

Theory algebras of relations

version 1.0 (February 22, 2007)

www.bucephalus.org

Contents

1	Abstract	3
2	Foreword	4

I Introduction 6

3	Introduction to Theory Algebras of Relations	7
3.1	What is a relation?	7
3.2	What is a theory algebra of relations?	8
3.3	Besides, what makes propositional logic a special case of relation algebra?	10
3.4	What is a theory algebra, anyway?	11

II The language of mathematics 12

4	Idiosyncratic features	13
4.1	Text arrangement, style and layout	13
4.2	Deviant use of common notions	13
4.3	Default syntax for operations	14
4.4	From ordinal to schematic mathematics	15
5	Basic concepts and notations	17
5.1	Stylistic conventions	17
5.2	Expressions	17
5.3	A sketch of mathematical semantics	18
5.4	Special terms and tuples	21
5.5	Formulas	21
5.6	Classes and numbers	23
5.7	Operations	26
5.8	Structures	29

III Quasi-hierarchies 31

6	Examples of quasi-structures	32
6.1	Introduction	32
6.2	Power algebras	32
6.3	A generalized quasi-boolean algebra on tuples	33
6.4	The theory algebra of propositional formulas	34
6.5	Quasi-ordered classes on numbers	36
6.6	The quasi-field of fractions	36
7	Quasi-structures and quasi-algebras	38
7.1	Special binary endorelations	38
7.2	Quasi-Hierarchies	38
7.3	Quasi-ordered classes and poclasses	38
7.4	Quasi-lattices	39

7.5	Distributivity and complementation	40
7.6	Quasi-algebras	41
7.7	Quotient structures	43

8 Additional junctors for quasi-boolean structures 45

8.1	Junctor sets	45
8.2	Multary sub- and equijunction	45

IV Records 52

9 Records, including tuples and schemas 53

9.1	Records	53
9.2	Tuples and other special records	54
9.3	Schemas	54

10 Operations on records 56

10.1	Values, domains and codomains	56
10.2	Projections	56
10.3	Relations between records	56
10.4	Distinct joins	58
10.5	Concatenation of tuples	58

11 Order structures and junctions on records 60

11.1	Poclasses of records	60
11.2	Lattices on records	60
11.3	Various domains of record classes	61
11.4	Record junctions	61
11.5	Properties of the junctions	64
11.6	Properties for compatible arguments	67
11.7	Big junctions	67
11.8	The general record structure	69
11.9	The complete boolean algebra of projections	70

V Schemas and their various products 72

12 Various schema and power products 73

12.1	Various products	73
12.2	Various ordinal products	74
12.3	Concatenation of tuple classes	75
12.4	Distributivity	76
12.5	Power products	79

13 Digressions 81

13.1	Digression: Product and coproduct constructions	81
13.2	Digression: Datastructures and formal languages as coproducts of schemas	82

14 Order structures on schemas 84

14.1	The operations	84
14.2	The included schema class	84
14.3	The boolean algebra of inclusions	84

VI Graphs and their distinct products 86

15	Schemas and graphs	87
15.1	Schemas of record classes	87
15.2	Graphs	88
16	Partitions and distinct products	90
16.1	Class partition (record)	90
16.2	Record and schema partition	90
16.3	Relative distinctness	92
16.4	Distinct products	93
16.5	Projections of graphs	95
16.6	Distributivity	95

VII Relations 98

17	Relations	99
17.1	Introduction	99
17.2	General definition	99
17.3	Ordinary relations as (schematic) relations and other notational variations	101
17.4	Relations as functions and vice versa	102
17.5	Tables and table representation	103
17.6	Double tables	104
17.7	More properties of relations	105
17.8	Projection relation class	106
18	Digression: null values, quasi-relations and expansions	108
18.1	Null values and quasi-relations	108
18.2	Quasi-relations as relations	109
18.3	Schema expansion of relations	110

VIII Operations on relations 112

19	Some basic operations on relations	113
19.1	Schema comparison	113

19.2	Identity relations	114
19.3	Empty and full relations	115
19.4	Complement	116
19.5	Boolean operations on equi-schematic relations	117
19.6	Distinct (cartesian) product	120
19.7	Concatenation of ordinary relations	121

20 Schema expansion and the semantic operations 122

20.1	Schema expansion	122
20.2	Semantic operations on compatible relations	124
20.3	Digression: performing semantic operations with tables	124
20.4	Properties of semantic operations	126
20.5	More derived semantic operations	134

21 Decreasing the schema of relations 135

21.1	Introduction and overview	135
21.2	Projection and coprojection	137
21.3	Properties of projections and coprojections	139
21.4	(Attribute) Eliminations	145
21.5	Properties of eliminations	145
21.6	Reductions	149
21.7	Properties of the reductions	151
21.8	Redundancy of attributes	155

22 Attribute translation 157

22.1	Introduction	157
22.2	Translation of records	157
22.3	Translations in $\mathbf{Proj}(\xi)$	159
22.4	Translation of relations	160
22.5	Translations in $\mathbf{Prel}(X)$	161

IX Theory algebras of relations 162

23 Theory algebras on $\mathbf{Prel}(X)$ 163

23.1	The semantic structures on $\mathbf{Prel}(X)$	163
23.2	Selective syntactic structures on $\mathbf{Prel}(X)$	164
23.3	The 3-carrier syntactic structures on $\mathbf{Prel}(X)$	165
23.4	The 2-carrier syntactic structures on $\mathbf{Prel}(X)$	166
23.5	The 1-carrier syntactic structures on $\mathbf{Prel}(X)$	166
23.6	Theory structures on $\mathbf{Prel}(X)$	167

A Summary of symbols 169

1 Abstract

By (schematic) **relation** we mean a generalization of two other relation concepts: the *n-ary* or *ordinary relations* from mathematics and the *partial tables* from database theory. A **theory algebra** is similar to a boolean algebra. But next to this “semantical” order structure it also comprises a “syntactical” structure, which is more or less a boolean algebra as well.

This text introduces these kind of relations, a couple of operations on them and investigates their properties. Certain classes of relations together with this operations constitute these (powerful, flexible and elegant) theory algebras.

For the development of the whole theory some preliminary chapters are also covered: On generalizations of partial order structures and lattices, called **quasi-hierarchies**. On **records**, operations and order structures on records. On **schemas** and their various products. On **graphs** (here defined as record classes) and certain operations on them.

2 Foreword

Foreword to version 1.0

This text is a generalization of the relation concept in mathematics and computer science in a context of a whole algebra. As it is given here, the subject is a self-contained, systematic and closed piece of mathematics.

On the other hand, it emerged as part of a bigger *foundation project*, one that puts the emphasis on “*relation*” rather than “*set*” or “*function*” as the most important and most basic concept for metamathematics. The main motivation for a *theory algebra of relations* then lies in the idea that standard predicate logic has an insufficient semantics. Given a first-order signature \mathfrak{S} and formulas φ, ψ on that grammar, there is a standard definition of the notions “ φ is *valid*” and “ φ *entails* ψ ”. But this semantic is only binary, a decision whether validity or entailment either holds or doesn’t hold. It induces a (quasi-) boolean algebra and tells us how and when a formula is valid or subvalent to another formula. But it doesn’t tell the “*meaning*” of the formula. The theory algebra $\mathfrak{Prel}(X)$ of relations does provide such a semantic. The schema X is a translation of the signature \mathfrak{S} and every (closed or open) formula denotes a unique relation in $\mathfrak{Prel}(X)$.

Actually, there have been earlier and well-elaborated other works to fill this gap in logical semantics. The one that comes closest to the design of theory algebras is probably the *cylindric algebra*.¹ However, our text here is quite isolated and doesn’t pay proper respect to these alternative approaches. The only excuse is the impression, that they are still too artificial for our main purposes and that there is a much more “*natural*” semantics. The relation concept in this text (together with terminology like “*record*”, “*schema*” etc.) is taken from relational database theory.² But again, the ongoing research in this area is neglected here as well.

In fact, in this text we don’t even deal with the connections to logical semantics. We just present a self-contained definition of these “*theory algebras of relations*” and we investigate their properties. Most readers will probably discover the postponed potential applications for predicate logic themselves, since the boolean properties are so similar.

This text will certainly need error updates, maybe even changes in content over time. Therefore it carries a *version number*, starting with “1.0”. The idea is, to increase the minor number for each minor error update. The major number is reserved for significant changes.

Thanks to the numerous unselfish projects and people who created a whole new dimension and platform for scientific work and communication over the last years.

¹Leon Henkin, J. Donald Monk, Alfred Tarski: *Cylindric Algebras*, (two parts) North-Holland 1971.

²The foundation paper is: E.F. Codd: *A Relational Model of Data for Large Shared Data Banks*, *Communications of the ACM*, Vol.13/6, June 1970. A classic text book is: C.J. Date: *An introduction to Database Systems*, Addison Wesley, several editions.

Special thanks to the GNU/Linux and $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ community. This text was entirely written with *Kile*, a free $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ editor.

Part I

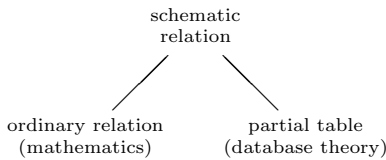
Introduction

3 Introduction to Theory Algebras of Relations

3.0.1 Overview

This introduction has three subsections:

- (1) First, we define a “relation” as a *schematic relation*. This definition is a generalization of the usual relation concept in mathematics, that we call *ordinary relation*. It also includes the relation concept from database theory, called *partial table* in our terminology.



- (2) Relations are interesting only when we can operate with them, i.e. in the context of an algebra of relations. We are particularly interested in operations that altogether constitute a *theory algebra of relations*, as we call it.
- (3) One motivation for these algebras of relations is propositional logic, which can be reconstructed as a special example. On the other hand, relation algebras can be seen as generalizations of propositional logic in the sense that the restriction on binary values is repealed and arbitrary values are allowed for each atom.

3.1.2 What is a table?

If $R = [X, \Gamma]$ is relation with $X = [X_i | i \in I]$, and both the attribute class $I = \{i_1, \dots, i_n\}$ and the graph $\Gamma = \{x_1, \dots, x_m\}$ are finite, then R is called a *table* (17.5.3), often written as

$i_1 : X_{i_1}$	\dots	$i_n : X_{i_n}$
x_{1,i_1}	\dots	x_{1,i_n}
\vdots		\vdots
x_{m,i_1}	\dots	x_{m,i_n}

The head row of the table is the schema X , each subsequent row is one of the members x_j of the graph, where

$$X = \begin{bmatrix} i_1 \mapsto X_{i_1} \\ \vdots \\ i_n \mapsto X_{i_n} \end{bmatrix} \quad \text{and} \quad x_j = \begin{bmatrix} i_1 \mapsto x_{j,i_1} \\ \vdots \\ i_n \mapsto x_{j,i_n} \end{bmatrix} \quad \text{for } j = 1, \dots, m$$

For example, a table is given by

account_no : Integer	owner : String	balance : Real
12340	"alice"	243.25
12342	"bob"	-12.05
12347	"carl"	2000.00

3.1 What is a relation?

3.1.1 What is a schematic relation?

- (1) A *record* (9.1.2) $\xi = [\xi_i | i \in I]$ has for each *index* or *attribute* i of its *domain* I a *value* ξ_i . If $I = \{i_1, \dots, i_n\}$ is finite, we also write ξ as

$$\begin{bmatrix} i_1 \mapsto \xi_{i_1} \\ \vdots \\ i_n \mapsto \xi_{i_n} \end{bmatrix}$$

- (2) A *schema* (9.3.1) is a record $X = [X_i | i \in I]$, where every value X_i is a class.
- (3) The *cartesian product* (12.1.2) $\otimes X$ of such a schema X is the class of all records that fit into X . More precisely,

$$\otimes X = \{[x_i | i \in I] \mid x_i \in X_i \text{ for all } i \in I\}$$

- (4) A *schematic relation* (17.2.1) R is essentially made of a schema $X = [X_i | i \in I]$ and a *graph* Γ , where $\Gamma \subseteq \otimes X$. Our default notation for such an R is

$$\begin{bmatrix} X \\ \Gamma \end{bmatrix} \quad \text{or} \quad [X, \Gamma]$$

For every $x \in \otimes X$ we write $x \in R$, “ x is a member of R ” or “ R holds for x ”, iff $x \in \Gamma$. Otherwise, we write $x \notin R$.

3.1.3 What is a partial table?

A *partial table* (18.1.2) is a table, but it allows gaps or so-called *null* values in place of proper values. For example

account_no : Integer	owner : String	balance : Real
12340		243.25
12342	"bob"	
12347	"carl"	2000.00

is a partial table, where the **owner** component of the first record and the **balance** in the second are not specified.

A partial table is very much the definition of a “relation” in database theory, where it is used to manage and retrieve (large amounts of) information. However, the interpretation of these partial tables is a controversial issue in database theory. The relational model was originally introduced by Codd³, but soon the various versions of the *structured query language* or *SQL* applied a somewhat different understanding, which is sometimes criticized for being less consequent and logical from the original point of view. Nowadays SQL has become the standard in practical and commercial systems, but this dominance has its critics.⁴

In our text we will not refer to this discussion. Our genuine

³E. F. Codd, “A relational model of data for large shared data banks”, 1970, Communications of the ACM, 13(6):377-387.

⁴For an introduction to these issues, see e.g. the article “Relational model” on en.wikipedia.org.

motivation is a generalization and reconstruction of logical calculi rather than the implementation of a database management system. However, in digression 18 we present an interpretation of null values, that allows us to properly subsume partial tables under schematic relations. In our approach we obtain

$$\text{tables} \subseteq \text{partial tables} \subseteq \text{schematic relations}$$

That way, we are probably more close to Codd and his predecessors than to the SQL approach.

3.1.4 What is an ordinary relation?

It is pretty much the standard in mathematics to define:

- (1) A (n -ary) *tuple* (5.4.1) $\xi = \langle \xi_1, \dots, \xi_n \rangle$ is a sequence of n components ξ_1, \dots, ξ_n .
- (2) A (n -ary) *cartesian product* (5.6.9) of classes C_1, \dots, C_n is the class of all n -tuples that fit in, i.e.

$$C_1 \times \dots \times C_n = \{ \langle x_1, \dots, x_n \rangle \mid x_1 \in C_1, \dots, x_n \in C_n \}$$

- (3) An *ordinary relation* R (5.7.10) is essentially given by its *domain*, which is a cartesian product $C_1 \times \dots \times C_n$, and its *graph* Γ , which is a subclass of this domain.

In our default notation (5.7.10) we write such an ordinary relation as

$$R = \left[\begin{array}{c} C_1 \rightsquigarrow \dots \rightsquigarrow C_n \\ \langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi \end{array} \right]$$

where " $C_1 \rightsquigarrow \dots \rightsquigarrow C_n$ " is the type expression for R and φ is a formula that determines which $\langle x_1, \dots, x_n \rangle$ is a member of Γ .

For example

$$R := \left[\begin{array}{c} \mathbb{Z} \rightsquigarrow \mathbb{Z} \rightsquigarrow \mathbb{Z} \\ \langle a, b, c \rangle \rightsquigarrow a + b = c \end{array} \right]$$

defines R to be the ordinary relation that holds for three integers a, b, c iff c is the sum of a and b . "Ordinary" can be rephrased "ordinal arity". This example R is "3-ary" or "ternary".

3.1.5 What makes every ordinary relation a schematic relation?

A tuple $\langle \xi_1, \dots, \xi_n \rangle$ can be (re-)defined (9.2.1) as a finite record on the ordinal number⁵ $\{1, \dots, n\}$, i.e. we could declare

$$\langle \xi_1, \dots, \xi_n \rangle := [\xi_i \mid i \in \{1, \dots, n\}]$$

That way and with some changes in the notation, every ordinary relation turns into a schematic relation: The cartesian class product $C_1 \times \dots \times C_n$ now is the special case of the more general cartesian product applied to the *ordinal schema* $\langle C_1, \dots, C_n \rangle$, i.e.

$$C_1 \times \dots \times C_n = \otimes \langle C_1, \dots, C_n \rangle$$

The change of an ordinal relation

$$R = \left[\begin{array}{c} C_1 \rightsquigarrow \dots \rightsquigarrow C_n \\ \langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi \end{array} \right]$$

into a proper schematic relation is then merely a change in the notation

$$R = \left[\begin{array}{c} \langle C_1, \dots, C_n \rangle \\ \{ \langle x_1, \dots, x_n \rangle \in \otimes \langle C_1, \dots, C_n \rangle \mid \varphi \} \end{array} \right]$$

so that we may state that it actually *is* a schematic relation, i.e.

$$\text{ordinary relation} \subseteq \text{schematic relation}$$

3.1.6 What is a relation?

A schematic relation is a generalization of both the "relation" from mathematics and the "relation" in database theory, at least in some particular version of database theory. Due to this general character we usually say *relation* instead of *schematic relation*.

3.2 What is a theory algebra of relations?

3.2.1 What do $\mathbf{Proj}(X)$, $\mathbf{Rel}(X)$ and $\mathbf{Prel}(X)$ stand for?

$\mathbf{P}(C) := \{A \mid A \subseteq C\}$ is the *power class* (5.6.13) of a given class C .

Let $X = [X_i \mid i \in I]$ be a proper schema from now on. Then

$$\mathbf{Proj}(X) := \{[X_j \mid j \in J] \mid J \subseteq I\}$$

is the *projection class* (10.2.6) of X

$$\mathbf{Rel}(X) := \left\{ \left[\begin{array}{c} X \\ \Gamma \end{array} \right] \mid \Gamma \subseteq \otimes X \right\}$$

is the *relation class* (17.2.6) on X

$$\mathbf{Prel}(X) := \bigcup_{Y \in \mathbf{Proj}(X)} \mathbf{Rel}(Y)$$

is the *projection relation class* (17.8.1) on X

3.2.2 What makes $\mathbf{Proj}(X)$ a boolean algebra?

For two records, in particular for two schemas $Y = [Y_j \mid j \in J]$ and $Z = [Z_k \mid k \in K]$ we define (10.3.2)

$$Y \leq Z \quad \text{iff} \quad J \subseteq K \text{ and } Y_j = Z_j \text{ for all } j \in J$$

Thus $\mathbf{Proj}(X)$ is the class of all the schemas Y with $Y \leq X$. And together with \leq , this class makes a poclass (i.e. \leq is a *partial order* on $\mathbf{Proj}(X)$).

$\langle \mathbf{Proj}(X), \leq \rangle$ even turns out to be a boolean algebra (11.9.4) with

$$\clubsuit Y \wedge Z := [X_i \mid i \in (J \cap K)] \quad \text{the meet of } Y \text{ and } Z$$

⁵ More often, the n -th ordinal number is defined as $\{0, 1, \dots, n-1\}$ rather than $\{1, \dots, n\}$, but that wouldn't make a significant difference.

- ♣ $Y \vee Z := [X_i | i \in (J \cup K)]$ the join of Y and Z
- ♣ $X \setminus Y := [X_i | i \in (I \setminus J)]$ the complement of Y
- ♣ $\langle \rangle$ the bottom element
- ♣ X the top element

for all $Y = [X_j | j \in J]$ and $Z = [X_k | k \in K]$ in $\mathbf{Proj}(X)$.

3.2.3 ———What makes $\mathbf{Rel}(X)$ a boolean algebra?———

We can inherit the inclusion \subseteq from the algebra of classes, i.e. we put (19.5.1)

$$\begin{bmatrix} X \\ \Gamma \end{bmatrix} \subseteq \begin{bmatrix} X \\ \Sigma \end{bmatrix} \quad \text{iff} \quad \Gamma \subseteq \Sigma$$

That way $\mathbf{Rel}(X)$ turns into a poclass, isomorph to $(\mathbf{P}(\otimes X), \subseteq)$. As such, it even is a (complete) boolean algebra (19.5.10), denoted by $\mathfrak{Rel}(X)$, with

$$\begin{bmatrix} X \\ \Gamma \end{bmatrix} \cap \begin{bmatrix} X \\ \Sigma \end{bmatrix} := \begin{bmatrix} X \\ \Gamma \cap \Sigma \end{bmatrix} \quad \begin{bmatrix} X \\ \Gamma \end{bmatrix} \cup \begin{bmatrix} X \\ \Sigma \end{bmatrix} := \begin{bmatrix} X \\ \Gamma \cup \Sigma \end{bmatrix}$$

as meet and join and

$$\neg \begin{bmatrix} X \\ \Gamma \end{bmatrix} := \begin{bmatrix} X \\ (\otimes X) \setminus \Gamma \end{bmatrix} \quad \perp_X := \begin{bmatrix} X \\ \emptyset \end{bmatrix} \quad \top_X := \begin{bmatrix} X \\ \otimes X \end{bmatrix}$$

as complement, bottom and top, respectively.

3.2.4 ———What makes $\mathbf{Prel}(X)$ a quasi-boolean algebra?———

$\mathbf{Prel}(X)$ is a superclass of $\mathbf{Rel}(Y)$, for every $Y \leq X$, and each $\mathfrak{Rel}(Y)$ is a boolean algebra. So let us seek for a structure $\mathfrak{Prel}(X)$ on the carrier class $\mathbf{Prel}(X)$ which is a superstructure of each of the $\mathfrak{Rel}(Y)$. And this time, let us use the square symbols “ $\sqsubseteq, \equiv, \sqcap, \sqcup$ ” to denote the according operations. The unary \neg remains the same with the definition from in 3.2.3.

For all $R = [Y, \Gamma]$ and $S = [Z, \Sigma]$ in $\mathbf{Prel}(X)$ the new operations must satisfy the embedding properties (20.4.4): If $Y = Z$, then

$$R \sqcap S := R \cap S \quad R \sqcup S := R \cup S \quad R \sqsubseteq S \text{ iff } R \subseteq S$$

The problem is the definition of the more general case, where $Y \neq Z$ may occur.

Later on, we will succeed in defining these operations properly and $\mathfrak{Prel}(X)$ will be a kind of boolean algebra as well, but only a “quasi” kind. Different to \subseteq , this \sqsubseteq is not a partial order, but a only a *quasi-order*. It is not *canonic* (or *anti-symmetric*) anymore, i.e. R and S may be equivalent ($R \equiv S$, i.e. $R \sqsubseteq S$ and $S \sqsubseteq R$), but still not the same.⁶

For example, for $Y \neq Z$, the two relations \perp_Y and \perp_Z are equivalent, however not identical.

3.2.5 ———What makes $\mathbf{Prel}(X)$ a theory algebra?———

That it is not just one, but a special combination of two quasi-

order structures is the characteristic feature of a *theory algebra*. For this particular structure $\mathfrak{Prel}(X)$ this means:

- ♣ The *semantic* structure is given by the *subvalence* relation \sqsubseteq (and the derived \equiv, \sqcap, \sqcup etc.), which essentially compares the graphs of the given relations.
- ♣ The *syntactic* structure on the other hand is given by the *subschema* relation \trianglelefteq (and the derived $\hat{=}, \uparrow, \downarrow$ etc.).

The definition (19.1.2) of \trianglelefteq on $\mathbf{Prel}(X)$ is simply

$$\begin{bmatrix} Y \\ \Gamma \end{bmatrix} \trianglelefteq \begin{bmatrix} Z \\ \Sigma \end{bmatrix} \quad \text{iff} \quad Y \leq Z$$

Again, \trianglelefteq is not a proper partial, but only a quasi-order on $\mathbf{Prel}(X)$, because there are *equi-schematic* (19.1.2) relations ($R \hat{=} S$, i.e. $R \trianglelefteq S$ and $S \trianglelefteq R$), which are not identical. For example, $\perp_X \hat{=} \top_X$.

For this particular theory algebra $\mathfrak{Prel}(X)$, identity means “bi-equivalence” in the sense that

$$\begin{bmatrix} Y \\ \Gamma \end{bmatrix} = \begin{bmatrix} Z \\ \Sigma \end{bmatrix} \quad \text{iff} \quad \begin{bmatrix} Y \\ \Gamma \end{bmatrix} \hat{=} \begin{bmatrix} Z \\ \Sigma \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} Y \\ \Gamma \end{bmatrix} \equiv \begin{bmatrix} Z \\ \Sigma \end{bmatrix}$$

Similar to the operations \sqcap, \sqcup etc. we could introduce say “ \wedge, \vee ” etc. by specifying the “...” in the definitions

$$\begin{bmatrix} Y \\ \Gamma \end{bmatrix} \wedge \begin{bmatrix} Z \\ \Sigma \end{bmatrix} := \begin{bmatrix} Y \wedge Z \\ \dots \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} Y \\ \Gamma \end{bmatrix} \vee \begin{bmatrix} Z \\ \Sigma \end{bmatrix} := \begin{bmatrix} Y \vee Z \\ \dots \end{bmatrix}$$

etc. But we use a different approach, one that is less elegant from an algebraic point of view, but more practical for the application fields we have in mind. We define $\|, \uparrow, \downarrow$ not as binary functions on $\mathbf{Prel}(X)$, but as binary operations of the mixed type

$$\mathbf{Prel}(X) \times \mathbf{Proj}(X) \longrightarrow \mathbf{Prel}(X)$$

So let $R = [Y, \Gamma] \in \mathbf{Prel}(X)$ and $Z \in \mathbf{Proj}(X)$ be given.

First of all, the *expansion* (20.1.1) of R by Z has the form

$$R \parallel Z := \begin{bmatrix} Y \vee Z \\ \Gamma' \end{bmatrix}$$

where Γ' is uniquely determined by demanding that $R \parallel Z \equiv R$ shall hold.⁷

On the other hand, we should have a similar *equivalent reduction* (21.1.2) of R onto Z ,

$$R \Downarrow Z := \begin{bmatrix} Z \\ \Gamma'' \end{bmatrix}$$

where again Γ'' is determined by $R \Downarrow Z \equiv R$. But different to the expansion, such a graph for an equivalent schema reduction does not always exist. However, we have two unique reductions onto Z , that come closest to R , namely

- ♣ a *supremum reduction* (21.6.1) $R \Downarrow Z$, that is the least subvalent relation on Z , i.e. the least of all the $R' \in \mathbf{Rel}(Z)$ with $R \sqsubseteq R'$, and similarly,
- ♣ a *infimum reduction* (21.6.1) $R \Uparrow Z$, which is the greatest subvalent element of $\mathbf{Rel}(Z)$.

⁶ Important structures on *poclasses* (7.2.2), in particular *lattices* (7.4.9) and *boolean lattices/algebras* (7.6.2), have become standard concepts in mathematics. But a generalization of order- and lattice theory on the basis of *quasi-ordered classes* (7.2.2) that introduces *quasi-lattices* (7.4.9) and *quasi-boolean algebras* (7.6.2), is not that common. We therefore give an overview of such a generalization in the special chapter of III.

⁷ In fact, this Γ' is given by $\Gamma' = \{\xi \vee v \mid \xi \in \Gamma, v \in \otimes(Z \setminus Y)\}$. In the main text we actually turn around the order of the definitions and use $\|$ to define $\sqsubseteq, \equiv, \sqcap$ etc. For $R = [Y, \Gamma]$ and $S = [Z, \Sigma]$ we put (20.2.2): $R \sqsubseteq S$ iff $(R \parallel Z) \subseteq (S \parallel Y)$, $R \sqcap S := (R \parallel Z) \cap (S \parallel Y)$, etc.

So we always have

$$\begin{array}{ccccccc}
 R \uparrow Z & \sqsubseteq & R & \equiv & R \parallel Z & \sqsubseteq & R \downarrow Z \\
 \in & & \in & & \in & & \in \\
 \mathbf{Rel}(Z) & & \mathbf{Rel}(Y) & & \mathbf{Rel}(Y \vee Z) & & \mathbf{Rel}(Z)
 \end{array}$$

3.2.6 What else?

The main subject of this text is the introduction and investigation of these theory algebras of relations. In order to provide suitable tools for practical applications, we provide more operations on relations and discuss some other relational structures. But most of them are merely variations of the same theme.

We will not discuss possible applications and implementations of these algebras.⁸ We only demonstrate how propositional logic can be reconstructed as a special case of relation algebra. Or the other way round, how relation algebras can be understood as arbitrary value generalizations of the two-valued propositional logic.

3.3 Besides, what makes propositional logic a special case of relation algebra?

3.3.1 What is the boolean table of a propositional formula?

Two examples of *propositional formulas* (6.4.1) are

$$-[a \wedge -b] \quad \text{and} \quad [a \vee [a \wedge b] \vee -b]$$

Their semantics can be defined by so-called *truth tables* or *boolean tables*. For example

a	b	
0	0	1
1	0	0
0	1	1
1	1	1

is the same boolean table for both given formulas (which is why they are called *equivalent*). Beside the head row, the table has four rows, each one for each possible *valuation* of the two *atoms* a and b with either a **0** (or **false**) or a **1** (or **true**). And the value in the extra right column tells us if the particular valuation turns the formula into a true (=1) or false (=0) statement.

3.3.2 How does a formula turn into a relation?

Each of the four valuations can be defined as a record, which maps the atoms a, b to values of $\mathbb{B} = \{0, 1\}$. The collection of these four records can accordingly be defined as the cartesian product $\otimes X$ of the schema

$$X := \begin{bmatrix} a \mapsto \mathbb{B} \\ b \mapsto \mathbb{B} \end{bmatrix}$$

So

$$\otimes X = \left\{ \begin{bmatrix} a \mapsto 0 \\ b \mapsto 0 \end{bmatrix}, \begin{bmatrix} a \mapsto 1 \\ b \mapsto 0 \end{bmatrix}, \begin{bmatrix} a \mapsto 0 \\ b \mapsto 1 \end{bmatrix}, \begin{bmatrix} a \mapsto 1 \\ b \mapsto 1 \end{bmatrix} \right\}$$

Now if φ is one of the two example formulas, say $-[a \wedge -b]$, then three of the four valuations hold for φ and the semantics of φ is represented by the (schematic) relation

$$\begin{bmatrix} X \\ \left\{ \begin{bmatrix} a \mapsto 0 \\ b \mapsto 0 \end{bmatrix}, \begin{bmatrix} a \mapsto 0 \\ b \mapsto 1 \end{bmatrix}, \begin{bmatrix} a \mapsto 1 \\ b \mapsto 1 \end{bmatrix} \right\} \end{bmatrix}$$

according to 3.1.1, and element of $\mathbf{Rel}(X)$.

And this relation can as well we written in the table form of 3.1.2

a : \mathbb{B}	b : \mathbb{B}
0	0
0	1
1	1

That way, we can say that every formula φ with atom class A is just a representation of an element of $\mathbf{Rel}(\llbracket \mathbb{B} | A \rrbracket)$ where $\llbracket \mathbb{B} | A \rrbracket$ is only another notation (9.2.4) for the schema $X = [X_a | a \in A]$ with $X_a = \mathbb{B}$ for all $a \in A$.

More general and in this interpretation, the class $\mathbf{Form}(A)$ of all propositional formulas with atoms taken from a class A quasi is just a class of representations for the relations in $\mathbf{Prel}(\llbracket \mathbb{B} | A \rrbracket)$.

3.3.3 What makes propositional logic an algebra of relations?

Both structures are quite the same or isomorph in the sense that the formula constructors \neg, \wedge, \vee behave like the \neg, \sqcap, \sqcup .

For example, for $A = \{a, b, c, \dots\}$ let $\varphi_1, \varphi_2 \in \mathbf{Form}(A)$ be two formulas with their corresponding $T_1, T_2 \in \mathbf{Prel}(\llbracket \mathbb{B} | A \rrbracket)$ be given by

$$\varphi_1 = -[a \wedge -b] \quad \varphi_2 = [-b \vee [c \wedge -c]]$$

$$T_1 = \begin{bmatrix} a & b \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix} \quad T_2 = \begin{bmatrix} b & c \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \end{bmatrix}$$

Then

⁸Please, check out www.bucephalus.org for more information.

$$T_1 \sqcap T_2 = \begin{array}{|c|c|c|} \hline a & b & c \\ \hline 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ \hline \end{array} \text{ is the table of } [\varphi_1 \wedge \varphi_2]$$

$$T_1 \sqcup T_2 = \begin{array}{|c|c|c|} \hline a & b & c \\ \hline 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ \hline \end{array} \text{ is the table of } [\varphi_1 \vee \varphi_2]$$

$$\neg T_1 = \begin{array}{|c|c|} \hline a & b \\ \hline 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ \hline \end{array} \text{ is the table of } \neg\varphi_1$$

That way, $\mathbf{Prel}(\mathbb{B}|A)$ is a quasi-boolean algebra. It behaves as propositional logic.

3.3.4 —Are there expansions and reductions for propositions?—

Even more, $\mathbf{Prel}(\mathbb{B}|A)$ is a theory algebra according to 3.2.5 with the well-defined $\sqsubseteq, \parallel, \uparrow, \downarrow$ etc. But here, we can simplify their type from

$$\mathbf{Prel}(\mathbb{B}|A) \times \mathbf{Proj}(\mathbb{B}|A) \longrightarrow \mathbf{Prel}(\mathbb{B}|A)$$

to

$$\mathbf{Prel}(\mathbb{B}|A) \times \mathbf{P}(A) \longrightarrow \mathbf{Prel}(\mathbb{B}|A)$$

because each $\mathbb{B}|A' \in \mathbf{Proj}(\mathbb{B}|A)$ is entirely given by A' only.

So if we take T_1 again, we have e.g.

$$T_1 \parallel \{c\} = \begin{array}{|c|c|} \hline a & b \\ \hline 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ \hline \end{array} \parallel \{b, c\} = \begin{array}{|c|c|c|} \hline a & b & c \\ \hline 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ \hline \end{array}$$

Now $\varphi_1 = \neg[a \wedge \neg b]$ is a formula for T_1 . And $T_1 \parallel \{c\}$ is a truth table for φ_1 as well, it only has one more atom c . If we want a proper definition of \parallel on formulas, we need to look for a transformation that returns an equivalent form, but adds the specified atoms. This task can be solved e.g. by the following approaches:

$$\varphi \parallel \{a_1, \dots, a_n\} := [\varphi \wedge [a_1 \vee \neg a_1 \vee \dots \vee a_n \vee \neg a_n]]$$

$$\varphi \parallel \{a_1, \dots, a_n\} := [\varphi \vee [\mathbf{0} \wedge a_1 \wedge \dots \wedge a_n]]$$

We can define the expansion within the usual framework of propositional logic. But this is not possible anymore with the reductions \uparrow and \downarrow . They perform an operation that is usually called *quantification*: \uparrow is similar to “ \forall ” and \downarrow is similar to “ \exists ”.

3.4 What is a theory algebra, anyway?

3.4.1 —What is a theory algebra?—

In 3.2.5 we already mentioned that the characteristic feature of theory algebras would be a certain combination of two more or less lattice-like structures, one “*syntactic*” and one “*semantic*” structure.

It is common in logic to formally characterize a *theory* T as a pair, made of

- the *syntax* or *language* Λ , which is defined as a formal language, i.e. a class of *sentences*, (*closed*) *formulas* or *expressions*, defined by means of a *signature*.
- And a *semantic*, *axiom* class or *theorem* class Θ , which is a selection of all the sentences, namely the ones that are (supposed to be) valid.

This pair character of theories induces two order structures on the class of all theories:

- A syntactical order, $T_1 \sqsubseteq T_2$ iff $\Lambda_1 \subseteq \Lambda_2$, and
- a semantical order, where $T_1 \sqsubseteq T_2$ iff $\Theta_1 \subseteq \Theta_2$,

for any two theories T_1, T_2 with languages Λ_1, Λ_2 and theorem classes Θ_1, Θ_2 , respectively.

The semantical order, the *consequence* or *entailment* relation, is a (quasi-) boolean lattice, theories can be negated, conjunctively and disjunctively combined to new theories. All that is part of daily life in mathematics. But the syntactical order is very lattice-like as well. For certain classes of theories, this is a (quasi-) boolean lattice as well. The combination of these two lattices makes a certain kind of structure with very specific features.

A *theory algebra* is the abstraction of this kind of structures on theories. It is an attempt to capture the features and properties of these double order structures.

3.4.2 —Again, what is a theory algebra of relations?—

So now we have a text on the “theory algebras of relations”, but without an appropriate definition of a “theory algebra”. However we choose to keep the “theory” in the title, because there are some other designs of “relation algebras” out there and it ought to be useful to have a distinguishing and descriptive component in the title.

⁹ For more information on these enriched versions of propositional logic as theory algebras, see other introductions on www.bucephalus.org, e.g. “*The Algebra of Worlds*”.

¹⁰ In a paper called “Axioms of Theory Algebras” we presented an elaboration of this principle. But it was too closely related to propositional logic, and certain features are chosen too specific to cover a structure such as $\mathfrak{Prel}(\mathfrak{X})$ as well.

Part II

The language of mathematics

4 Idiosyncratic features

4.0.3 Reading advice

In 5, we introduce the basic syntax of the language we are about to use, i.e. we provide a formal definition for *expressions*. The syntax is complete in the sense, that it is sufficient to express every mathematical statement that can be made at all. We intended to use standard terminology and symbolism, as far as there is one. However, in some cases we departed from conventions or created something unusual and these novelties shall be motivated and explained first here in section 4.

4.1 Text arrangement, style and layout

4.1.1 Text pieces and addressing

The main parts are called *chapters* and they are identified by big roman numbers. Every proper sentence of this text occurs in a *paragraph*, and each paragraph is uniquely identified by three decimal numbers “*n.n.n*” as its index. The overall structuring of this text is done as follows:

index	titles of the parts and subparts
II	this chapter
4	this section
4.1	this subsection
4.1.1	this paragraph

To provide some reference and structure inside paragraphs, we use

(1), (2), (3), ... (a), (b), (c), ... (i), (ii), (iii), ...

and in case there is no order required, we use item list:

- ♣ top level item
- ♠ second level item
- ♡ third level item
- ◇ fourth level item

4.1.2 Text style features

This text uses a couple of features to support efficient reading and the search of more important parts and key words.

this indicates the definition of a new notion, which is also listed in the *index* register at the end

this indicates the definition of an important new symbol or formalization, which is also listed in the *summary of symbols* register at the end

4.1.3 Two-dimensional expressions

If expressions become long, we sometimes use compact two-dimensional¹¹ versions. For class expressions, for example, we sometimes use

$$\left\{ \begin{array}{l} x_1, \\ x_2, \\ \vdots \\ x_n \end{array} \right\} \quad \text{or} \quad \left\{ \begin{array}{l} x \in C \\ \varphi_1, \\ \vdots \\ \varphi_n \end{array} \right\} \quad \text{and} \quad \left\{ \begin{array}{l|l} x \in C & \varphi_1, \\ & \vdots \\ & \varphi_n \end{array} \right\}$$

(often without the commas) for the usual $\{x_1, x_2, \dots, x_n\}$ or $\{x \in C \mid \varphi_1, \dots, \varphi_n\}$. Similarly for tuples $\langle \dots, \dots, \dots \rangle$ and other kind of expressions (\dots, \dots, \dots) we may write

$$\langle \dots \rangle \quad \text{and} \quad \left(\begin{array}{c} \dots \\ \dots \\ \dots \end{array} \right)$$

instead.

4.2 Deviant use of common notions

4.2.1 General concepts

Operation In algebra, an “operation” is usually a binary endo-function on a given class C , i.e. a function of type $C \times C \rightarrow C$. Instead we use *operation* as an informal collective notion for “relation, constant or function”. Accordingly, we can say that a *structure* is made of one or more classes with a couple of operations on them.

Class A *class* is any collection of elements. Usually in mathematics, the term “set” is preferred over “class”, as a kind of well-defined class, since an all too naive class concept can lead to paradoxes. But we do not care about the “dangers” here and consequently favor the more general concept. You may read “set” everywhere you find a “class”, the difference does not concern this text.

Identifier We occasionally say something like “let I be a class of identifiers”. In a proper mathematical sense that only means that I can be just any class. The term *identifier* will remain an undefined notion, like *point* is undefined in geom-

¹¹ G. Frege introduced a two-dimensional notation for mathematical logic in his *Begriffsschrift*. That didn’t survive and G. Peano’s linear symbolism (with \exists and \forall etc.) became established instead. But our proposal here is made in times of \TeX \LaTeX and other tools, where beautiful layout doesn’t take extra efforts and really supports efficient reading.

¹² It is probably fine to think of identifiers as strings like “example1”, including decimal numerals like “2345” as numerical strings. This is useful, because we define *records* as mappings from identifiers i_1, \dots, i_n to objects x_{i_1}, \dots, x_{i_n} , and *tuples* as special records, where the identifiers i_1, \dots, i_n are the integers $1, \dots, n$.

¹³ We could take strings as identifiers, but these strings can themselves be defined as character tuples. The same way we can define decimal numerals as tuples of the decimal digits 0,1,...,9. Leaving identifiers undefined provides us with a certain flexibility.

etry.¹²The point is, that I should be a pretty primitive type in practice, e.g. strings, integers, or even just finitely many character.¹³

which is defined by the formula φ . The predicator expression defines what we call the predicator. Neither the predicator notion nor a separate notation like our \rightsquigarrow are common concepts and that is a gap in the tradition.¹⁸Note, that we again make a clear distinction between the (untyped) predicator and the (typed) relation.¹⁹

4.3 Default syntax for operations

4.3.1 Standard notation for operations

We propose

$$\text{symbol} := \left[\begin{array}{c} \text{syntax} \\ \text{semantics} \end{array} \right]$$

as our standard form for the definition of functions and relations.¹⁴

- The “symbol” is an identifier, sometimes together with some graphical features pictured by means of placeholders $\textcircled{1}$, $\textcircled{2}$, \dots

For example, “ $\textcircled{1} + \textcircled{2}$ ” means that “+” is used as a binary operation symbol in infix notation. “ $\textcircled{1}^f$ ” explains, that the unary operation f is used in superscript notation when applied to arguments. As it is common in mathematics, we consider “ $f(\textcircled{1})$ ” as the default arrangement for the application of an operation f . And in that case we simply write “ f ” again on the left side of such a definition.

- The “syntax” is a type expression, i.e.

- $D \longrightarrow C$ for functions, and
- $D_1 \rightsquigarrow \dots \rightsquigarrow D_n$ for n -ary relations¹⁵¹⁶

- The “semantics” is one of the following:

- A map(ping) expression¹⁷ $a \mapsto \tau$, saying that any value for the variable a turns into the value which is defined by the term τ . A map expression on its own defines a mapping and we make a clear distinction between the (untyped) mapping and the (typed) function.
- A predicator expression $a \rightsquigarrow \varphi$, saying that any value for the variable a turns into the (false or true) expression,

For the time being we assume, that operations are ordinary or n -ary in the sense that their domain is an n -ary cartesian product $D_1 \times \dots \times D_n$. An then, we use tuples $\langle a_1, \dots, a_n \rangle$ of variables in place of the single variable a and summarize:

- The default n -ary function expression has the form

$$\left[\begin{array}{c} D_1 \times \dots \times D_n \longrightarrow C \\ \langle a_1, \dots, a_n \rangle \mapsto \tau \end{array} \right]$$

where f is an identifier, the D 's and C are class expressions, the a 's are variables and τ is a term with no more free variables than the given a 's.

- The default n -ary relation expression is

$$\left[\begin{array}{c} D_1 \rightsquigarrow \dots \rightsquigarrow D_n \\ \langle a_1, \dots, a_n \rangle \rightsquigarrow \varphi \end{array} \right]$$

where R is an identifier, the D 's are class expressions, the a 's are variables again and φ is a formula with no more free variables than the given a 's.

4.3.2 Examples

- The square function on the class \mathbb{Z} of integers can be defined in our standard notation by

$$\textcircled{1}^2 := \left[\begin{array}{c} \mathbb{Z} \longrightarrow \mathbb{Z} \\ n \mapsto n \cdot n \end{array} \right]$$

- The usual linear order relation on the class $\mathbb{N} := \{0, 1, \dots\}$ of natural numbers is given by

¹⁴ In my opinion there is the unfortunate tradition in mathematics that even the most precise authors not even try to invent a standard form at least for functions and use tiresome prose instead. Masses of clumsy sentences could be condensed into a proper notation and that can really ease the reading of texts. On the other hand, we try to be rather pragmatic than formalistic here ourselves and don't apply the proposed notation everywhere.

¹⁵ Different to function type expressions $D \longrightarrow C$, there is no common standard for (ordinary) relation type expressions yet. *Paul Taylor: A practical foundation of mathematics, 2000*, suggests $D_1 \longleftarrow D_2$ instead of our $D_1 \rightsquigarrow D_2$.

¹⁶ We occasionally also need a relation type expression like $R : D_1 \rightsquigarrow \dots \rightsquigarrow D_n$ for the cases $n = 1$ and $n = 0$. In this text we often say that, for an operation symbol \bullet , a dotted $\textcircled{1} \bullet \textcircled{2} \bullet \dots \bullet \textcircled{n}$ is just a convention for the more precise $\prod_{i=1}^n \textcircled{i}$, which can actually be written for $n = 1$ and $n = 0$. So when it comes to relation type expressions, it would be $R : \overset{1}{\rightsquigarrow}_{i=1} D_i$ and $R : \overset{0}{\rightsquigarrow}_{i=1} D_i$. But it is probably

nicer and makes sense to agree (see 5.6.8) on $R : \mathbf{Pty}(D_1)$ for the unary and $R : \mathbb{B}$ for the nullary case. (Recall, that $\mathbb{B} = \{0, 1\}$ and there are exactly two nullary relations.) Later on (in 17.2.6), we introduce a notation that will also cover these cases. In this notation we would write $R : \mathbf{Rel}((D_1))$ for unary and $R : \mathbf{Rel}(\langle \rangle)$ for nullary relation type expressions.

¹⁷ Another tradition (going back to A. Church) uses the lambda expressions $\lambda a. \tau$ instead of our map expression $a \mapsto \tau$.

¹⁸ Be aware, that there is a difference between the formula φ and the predicator expression $a \rightsquigarrow \varphi$. The variable a occurs free in φ , but is bound in $a \rightsquigarrow \varphi$.

Recall, that a map expression $a \mapsto \tau$ can be applied to an argument c , (often written $(a \mapsto \tau)(c)$). The result of this application is the term τ' , obtained from τ by replacing c for every free occurrence of a in τ . Similarly, we can apply a predicator expression $a \rightsquigarrow \varphi$ to an argument c , written again as $(a \rightsquigarrow \varphi)(c)$, and the result is φ' , where each free occurrence of a is replaced by c .

¹⁹ To memorize this non-standard taxonomy of typed and untyped operations we could use “function=domain+codomain+mapping” and “relation=domain+predicator”.

$$\begin{aligned} \textcircled{1} \leq \textcircled{2} &:= \left[\begin{array}{c} \mathbb{N} \leftrightarrow \mathbb{N} \\ \langle n, m \rangle \rightsquigarrow (\exists d \in \mathbb{N} . n + d = m) \end{array} \right] \\ &= \left[\begin{array}{c} \mathbb{N} \leftrightarrow \mathbb{N} \\ \langle n, m \rangle \rightsquigarrow (m - n \geq 0) \end{array} \right] \end{aligned}$$

- (3) If $\mathbb{Z}^+ := \mathbb{N} \setminus \{0\}$ denotes the positive integers, the division rest function can be defined recursively by

$$\textcircled{1} \bmod \textcircled{2} := \left[\begin{array}{c} \mathbb{N} \times \mathbb{Z}^+ \longrightarrow \mathbb{N} \\ \langle n, m \rangle \mapsto \begin{cases} n & \text{if } n < m \\ (n - m) \bmod m & \text{else} \end{cases} \end{array} \right]$$

- (4) An example of a ternary (3-ary) relation on \mathbb{N} is the well-known “ m and n are equivalent modulo d ”, for $d \neq 0$. We put

$$\textcircled{1} \equiv \textcircled{2} \text{ MOD } \textcircled{3} := \left[\begin{array}{c} \mathbb{N} \leftrightarrow \mathbb{N} \leftrightarrow \mathbb{Z}^+ \\ \langle n, m, d \rangle \rightsquigarrow (n \bmod d = m \bmod d) \end{array} \right]$$

- (5) Structures, basically a collection of operations on given classes, can be defined accordingly. For example, take a look at definition 5.8.4 to see how compact and lucid these declarations can become once the notation is accepted.

4.3.3 More complicated examples

Higher-order operations (i.e. operations with operations as arguments or results) can be elegantly formalized. Consider the previous relation

$$\textcircled{1} \equiv \textcircled{2} \text{ MOD } \textcircled{3} : \mathbb{N} \leftrightarrow \mathbb{N} \leftrightarrow \mathbb{Z}^+$$

again. It could alternatively be defined as a function

$$\text{MOD } \textcircled{1} : \mathbb{Z}^+ \longrightarrow (\mathbb{N} \leftrightarrow \mathbb{N})$$

which returns, for every positive d , the equivalence relation $(\textcircled{1} \equiv \textcircled{2} \text{ MOD } d)$ on the natural numbers. Written as an anonymous function this is

$$\left[\begin{array}{c} \mathbb{Z}^+ \longrightarrow (\mathbb{N} \leftrightarrow \mathbb{N}) \\ d \mapsto \left[\begin{array}{c} \mathbb{N} \leftrightarrow \mathbb{N} \\ \langle n, m \rangle \rightsquigarrow (n \bmod d = m \bmod d) \end{array} \right] \end{array} \right]$$

Alternatively the symbol definition can be integrated in the whole definition:

$$\text{MOD } \textcircled{1} := \left[\begin{array}{c} \mathbb{Z}^+ \longrightarrow (\mathbb{N} \leftrightarrow \mathbb{N}) \\ d \mapsto \left(\textcircled{1} \equiv \textcircled{2} \text{ MOD } d \right) \\ \mathbb{N} \leftrightarrow \mathbb{N} \\ \langle n, m \rangle \rightsquigarrow (n \bmod d = m \bmod d) \end{array} \right]$$

4.4 From ordinal to schematic mathematics

4.4.1 Variations and generalizations

Having proposed a standard for relation expressions, we are about to modify it again in the main part of this text, when we generalize the ordinary relation concept (in definition 17.2.1). We still use the general form

$$\left[\begin{array}{c} \text{syntax} \\ \text{semantics} \end{array} \right]$$

for (typed) operations. But our default form for ordinary relations

$$\left[\begin{array}{c} D_1 \leftrightarrow \dots \leftrightarrow D_n \\ \langle a_1, \dots, a_n \rangle \rightsquigarrow \varphi \end{array} \right]$$

will be changed:

- The “syntax”, formerly given by the ordinary relation type expression $D_1 \leftrightarrow \dots \leftrightarrow D_n$, will now be the class tuple $\langle D_1, \dots, D_n \rangle$. A class tuple is the special ordinary case of a [schema](#)²⁰ and such a schema is the type of the new generalized type of relation. The relations domain is always the cartesian product $\otimes X$ of its schema X . So the information is unchanged really.
- The “semantics” may be specified by the [graph](#) of the relation, i.e. the characteristic subclass of its domain. The [predicator](#) expression $x \rightsquigarrow \varphi$ can then be replaced by the class expression $\{x \in \otimes X \mid \varphi\}$. But obviously, the information again is still the same.

4.4.2 From “ordinary” to “schematic”

Modern mathematics is “ordinary” in the sense that operations have “ordinal arity”: their arguments are arranged as tuples and they are identified by their order in which they appear in writing. This ordinal design can be generalized towards a “schematic” form of mathematics, where the components are freed from their ordinal determination and arbitrary index classes are allowed instead. The key terms of this transformation are the following:

- The main role of the [tuple](#) is now taken by the [record](#), i.e. functions that return certain values ξ_i for given indices i . We often use the form

$$\left[\begin{array}{c} i_1 \mapsto \xi_{i_1} \\ \vdots \\ i_n \mapsto \xi_{i_n} \end{array} \right]$$

for finite records. Now, a tuple is just a special case of a finite record, defined as

$$\langle \xi_1, \dots, \xi_n \rangle := \left[\begin{array}{c} 1 \mapsto \xi_1 \\ \vdots \\ n \mapsto \xi_n \end{array} \right]$$

- A [schema](#) is the name of a record, in which each value is a class.
- The ordinary cartesian product operator \times now becomes a special case of the more general cartesian product \otimes , i.e. $D_1 \times \dots \times D_n$ can be redefined as $\otimes(D_1, \dots, D_n)$, and such a cartesian product is defined for every schema, not just for

²⁰ The “schema” is standard terminology in relational database theory.

class tuples $\langle D_1, \dots, D_n \rangle$.

- Consequently, “ordinary relations” have now become special instances of “schematic relations”.

This “schematization” of ordinary mathematics can be taken much further and one highlight will be the translation of whole structures into one relation. But all this will be properly introduced in due course.

5 Basic concepts and notations

5.1 Stylistic conventions

5.1.1 Definition Stylistic conventions

iff	is an abbreviation of “if, and only if”
<u>new word</u>	this is how we mark the definition of new notions in the text
new symbol	indicates the definition of an important new symbol or formalization

5.2 Expressions

5.2.1 Definition Expressions

An identifier or symbol is a basic building unit for expressions.
 An expression is

- either a primitive expression
 - i.e. an *identifier* or *symbol* ι itself
- or a complex expression

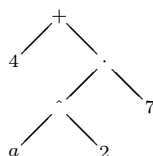
 - i.e. an application of the (*operator*) expression o to its (*argument*) expressions ξ_1, \dots, ξ_n .

5.2.2 Example

For example, suppose

$$+ \quad \cdot \quad \wedge \quad a \quad 2 \quad 4 \quad 7$$

are seven identifiers, then

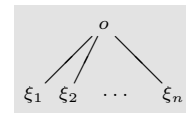


is an expression.

5.2.3 Remark Expressions, identifiers and symbols

- The “tree” definition 5.2.1 of complex expressions and the example expression in 5.2.2 may look strange and is probably not what one usually imagines.

Well, first of all note, that a *tree* is not the *picture of a tree*. The point is only, that the complex expression can be uniquely decomposed into its components again, and that this deconstruction is a primitive syntactical operation. Instead of writing



we could have written

$$o(\xi_1, \dots, \xi_n) \quad \text{or} \quad (o, \xi_1, \dots, \xi_n)$$

and that would turn our example into

$$+(4, \cdot(\wedge(a, 2))) \quad \text{or} \quad (+, 4, (\cdot, (\wedge, a, 2), 7))$$

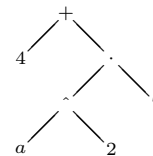
But that might still look strange, because one rather expects some picture like

$$“4 + a^2 \cdot 7”$$

- For a proper understanding of the expression concept, it is important to understand the following distinction:

- An *utterance* is the material appearance of an expression. This can be as speech, i.e. a sequence of phonemes, as computer code, i.e. a sequence of keyboard characters, or as a graphic on screen or paper, i.e. a two-dimensional display of graphemes.
- A *parse tree* reflects the inner structure of an expression.

For example, “*FourPlusSmallASquareTimesSeven*” or the L^AT_EX code “`4+2^2\cdot 7`” or the graphical “ $4 + a^2 \cdot 7$ ” are utterances that, given the usual syntax rules, stands for a parse tree which might be represented by



It is important to note, that mathematics is (almost) always communicated by means of utterances, but that the notion *expression* means the parse tree, i.e. the unambiguous syntactical structure.

In computer science, this utterance–parse-tree conversion is an issue and solutions are implemented in parsers, compilers, speech recognition devices etc. But in mathematics, a proper understanding of this process is usually taken for granted and entirely left to the reader. We will also take this easy way, usually just talk about *expressions* and trust on the readers intuition and education.

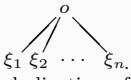
- We will not define *identifiers*. They could comprise decimal numerals like “2435”. But they could as well be restricted to digits and letters, in which case “2435” is not a primitive, but a complex expression, made of four digits. However, for our purposes it is probably the best to associate identifiers with short strings like “**sample**”, including numerical strings like “2435” and special characters like “+”, “ Σ ”, or “ f ”.

We summarize these remarks as follows.

5.2.4 Definition Meta-syntactical conventions

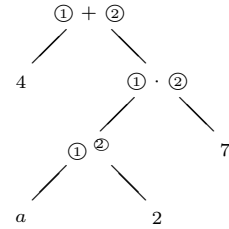
$o\xi$ or $o(\xi)$ for the unary case and more general

$o(\xi_1, \dots, \xi_n)$ for the n -ary case is the default notation for the application of an operator o on arguments ξ_1, \dots, ξ_n . In other words, it is the default graphical representation for the expression $\xi_1 \xi_2 \dots \xi_n$. But depending on the symbolization of o , there are many deviant notations and alternative configurations in mathematics. For all these non-default cases we use the following



$\textcircled{1}$, $\textcircled{2}$, ... placeholders for anonymous expressions or the definition of syntax features. For example, " $\textcircled{1} \leq \textcircled{2}$ " explains, that the identifier \leq is used as a binary operator symbol in infix notation.

() Parentheses are used to disambiguate the formal structure of expressions.



5.2.7 Definition Two-dimensional expressions

If expressions become long, we sometimes use compact two-dimensional versions. In particular, we may write

$\begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{pmatrix}$ or $\begin{pmatrix} \xi_1, \\ \xi_2, \\ \vdots \\ \xi_n \end{pmatrix}$ for (ξ_1, \dots, ξ_n) ,

$\left\{ \begin{matrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{matrix} \right\}$ or $\left\{ \begin{matrix} \xi_1, \\ \xi_2, \\ \vdots \\ \xi_n \end{matrix} \right\}$ for $\{\xi_1, \dots, \xi_n\}$,

$\left\{ \begin{matrix} \xi : C \\ \varphi_1 \\ \vdots \\ \varphi_n \end{matrix} \right\}$ or $\left\{ \begin{matrix} \xi : C & \varphi_1 \\ & \vdots \\ & \varphi_n \end{matrix} \right\}$ for $\{\xi : C \mid \varphi_1, \dots, \varphi_n\}$

etc.

5.2.5 Remark

- Placeholders enable us to turn an identifier (or token) (like "+" or " \leq " or "pr") into a proper symbol, i.e. the token plus its graphical outfit when the according operator is applied (e.g. " $\textcircled{1} + \textcircled{2}$ " or " $\textcircled{1} \leq \textcircled{2}$ " or "pr $\textcircled{1}$ "). Occasionally there are more identifiers to make one symbol (e.g. the three ones "if", "then", and "else" in the symbol "if $\textcircled{1}$ then $\textcircled{2}$ else $\textcircled{3}$ ") or even no identifier at all (e.g. the binary symbol " $\textcircled{1} \textcircled{2}$ " for the default operator application or " $\textcircled{1} \textcircled{2}$ " for the exponentiation). However, in the sequel we will not be precise on the distinction between identifiers and symbols and just talk about symbols in general.
- Parentheses are no symbols in the just mentioned sense, but only auxiliary tokens to generate expressions from given utterances (see 5.2.3(1)). Due to the common conventions, we do use parentheses explicitly in particular in " $f(x)$ ", i.e. applications of an operator f on an argument x . But it usually doesn't lead to ambiguities to write " (fx) " or " fx " instead. There are however some exceptions: we use parentheses in " $f(x_1, \dots, x_n)$ " as an abbreviation for " $f((x_1, \dots, x_n))$ ", which is standard notation in mathematics.
- When choosing words or acronyms for identifiers, we try to follow the standard if there is some. For example, "dom", "false" and "id". In our own creations we loosely apply the rule to
 - start with a capital letter, e.g. "Rel", if the whole thing stands for a class, in this case a class of relations,
 - start with a small letter, e.g. "rel", if the whole expression stand for a single say relation, and
 - use capital letters exclusively, e.g. "REL", for the proper class of relations (i.e. a class too big to be a set).

5.2.8 Definition Substitution

$\xi[x/v]$ the "substitution of v for x in ξ " or " ξ with x replaced by v ", denotes the expression ξ , where every occurrence of the identifier or symbol x is replaced by the expression v .

5.2.9 Example Example

$$(4 + a^2 \cdot 7)[a/-1] \text{ is } 4 + (-1)^2 \cdot 7$$

5.2.10 Remark

We define expression classes, e.g. the propositional formula class **Form**(A) in 6.4.1, in a traditional way: in prose. But later on in 13.2, we present a proper formal and much more elegant way to (context-free) formal languages, namely as co-products of schemas (12.1.2).

5.3 A sketch of mathematical semantics

5.3.1 Remark Introduction

The explanations here in this subsection 5.3 about the semantics of mathematics are not very thorough. One goal of the whole text is the attempt to fill this gap later on.

5.2.6 Example Example

So if we consider example 5.2.2 again, this time with symbols instead of identifiers, then " $4 + a^2 \cdot 7$ " or more precisely " $4 + ((a^2) \cdot 7)$ " would be a graphical representation for the expression

5.3.2 Definition — Types, values and environments

An expressions ξ can have a type τ , which is a class, and a value ω , which is an element of τ .
 An environment (or context) is a collection of expressions together with their possible types and values.
 In a given environment, an expression is called (well-) typed if it has a well-defined class as type, (well-) instantiated if it has a well-defined value, and (well-) defined if it is both well-typed and well-instantiated and its value is a proper element of its type.
 There is also an alternative terminology that we are going to use: If, in a given environment, an expression does have a unique value assigned, we call it closed and its potential value is a constant (value). Otherwise, the expression is open and its value is a variable (value).

5.3.3 Remark

- (1) Not all the expressions need to be typed and instantiated explicitly in an environment. Often, their types and values are derivable. For example, if “5” is the usual well-defined integer and “ $\textcircled{1} \cdot \textcircled{2}$ ” the usual integer multiplication, we can evaluate the value and type for “5 · 5” according to the usual method from 5.3.4.
- (2) If again “5” and “ $\textcircled{1} \leq \textcircled{2}$ ” are the usual constant operators on the class \mathbb{Z} of integers and v is a new identifier in the given environment, then “ $v \leq 5$ ” is an open expression, or more precisely, an open *formula* (5.3.11). “ $\forall v . v \leq 5$ ” is a closed formula, however not a well-typed one, because v is not yet typed. “ $\forall v : \mathbb{Z} . v \leq 5$ ” on the other hand would be typed and the whole formula is well-defined. Also, if “ $v : \mathbb{Z}$ ” has modified the environment before so that v is typed, then “ $\forall v . v \leq 5$ ” would as well be a well-defined formula and its value would be **true**.
- (3) The previous example also shows that an environment is a somehow dynamic concept, often varying between an expression and its subexpressions because some operators like the ones in 5.3.5, but also closing operators like “ \forall ” or “ \mapsto ” create (temporary) bindings for their arguments.
- (4) Two special and important environments in mathematics are created in the following two cases.
 - (a) The definition of a (*concrete*) *structure*, see for example the boolean value algebra in 5.8.4, introduces a couple of well-defined identifiers or symbols.
 - (b) A *theory* or *abstract structure* first introduces a *signature*, i.e. same class variables and some typed operator variables, and second names some axioms, i.e. closed formulas based on this vocabulary.

5.3.4 Remark — Type and value evaluation

Given an environment ε and an expression ξ .

- (1) If ξ is an identifier ι , the type of ξ is the type of ι and the value of ξ is the value of ι in ε .
- (2) If ξ is a complex expression $o(\xi_1, \dots, \xi_n)$, then
 - (a) ξ can be well-defined only if o is an operator expression and the codomain types of the ξ_1, \dots, ξ_n are compatible with the domain type of o ,
 - (b) the type of ξ is the codomain type of o , and
 - (c) the value of ξ is the result of the ξ_1, \dots, ξ_n under o .

As it is common in mathematics and some computer languages (while forbidden in others), we allow symbols to be overloaded, i.e. one symbol may be used to represent several different operators, if the domains of these operators are clearly distinct. For example, “ $\textcircled{1} \leq \textcircled{2}$ ” denotes the linear order between integers as well as real numbers, we will use it to compare two

records etc. In all these cases, it is the type of the used arguments that determines which of the operators is intended in a given application.

5.3.5 Definition — Operators on environments

$\xi := \omega$ is the general form of an assignment (or instantiation) and it defines ω to be the (new) value of ξ in the current environment. Usually, ξ is an identifier or symbol, but occasionally it may be a more complex expression.

$\xi : \tau$ is the general form of an type expression, which is a constraint to the possible values of ξ , that now has to be element of the class τ .

$\xi := \begin{bmatrix} \tau \\ \omega \end{bmatrix}$ stands for $(\xi : \tau) := \omega$ or $\xi := (\omega : \tau)$ and is our default notation for a full operator definition, i.e. a simultaneous type constraint and instantiation.

5.3.6 Remark

- (1) “ $\xi : \tau$ ” is what we call a *type expression*, but sometimes, that notion only applies to the class expression “ τ ”. This ambiguity shouldn’t cause major problems in the sequel.
- (2) Somehow, the type expression “ $\xi : C$ ” is similar to the element expression “ $\xi \in C$ ”. But “ $\xi \in C$ ” denotes a truth value, “ $\xi : C$ ” denotes the value ξ and causes a kind of *side effect* on the environment that restricts the possible values for ξ . However and despite these fundamental differences, both expressions are often confused and it is very common to use the element expression instead of a type expression. We will also do so and write something like “ $\{n \in \mathbb{N} \mid n \geq 7\}$ ” (see 5.6.2) or “ $\forall n \in \mathbb{N} . n + 7 > 5$ ” (see 5.5.5).
- (3) According to the derivability of types (5.3.4) there is a certain freedom on the location of type expression. Using the “ $\textcircled{1} : \textcircled{2}$ ” notation is a side effect on the environment but does not change the value. Let us take the two most important kinds of operators to explain that:
 - (a) A function is a mapping with an explicit domain and codomain (see 5.7.3) But this could also be written as a map expression with implicit type expressions. For example, the square function $\textcircled{1}^2$ on the integers is in our default notation (see 5.7.3) written as “ $\begin{bmatrix} \mathbb{Z} \longrightarrow \mathbb{Z} \\ n \mapsto n \cdot n \end{bmatrix}$ ”. But one would get the same information with “ $(n : \mathbb{Z}) \mapsto ((n \cdot n) : \mathbb{Z})$ ” or “ $(n \mapsto n \cdot n) : (\mathbb{Z} \longrightarrow \mathbb{Z})$ ”.
 - (b) An ordinary relation in our default notation (see 5.7.10), e.g. the linear order on the natural numbers

$$\begin{bmatrix} \mathbb{N} \rightsquigarrow \mathbb{N} \\ \langle n, m \rangle \rightsquigarrow (\exists d \in \mathbb{N} . n + d = m) \end{bmatrix}$$

can as well be written as

$$\langle n : \mathbb{N}, m : \mathbb{N} \rangle \rightsquigarrow (\exists d \in \mathbb{N} . n + d = m)$$

or

$$(\langle n, m \rangle \rightsquigarrow (\exists d \in \mathbb{N} . n + d = m)) : \mathbb{N} \rightsquigarrow \mathbb{N}$$

However, these forms are usually not our first choice and we

use the “ $\begin{bmatrix} \textcircled{1} \\ \textcircled{2} \end{bmatrix}$ ” as the default version.

- (4) An assignment “ $x := \xi$ ” is a kind of abbreviation. It allows us to use the simple “ x ” for the complex “ ξ ” in new expressions. From a sematical point of view, this is no real contribution and no new information. So somehow, the assignment “ $x := \xi$ ” is similar to the equation “ $x = \xi$ ”. But actually, this is not so. The equation

is a statement, thus a truth value, but the assignment — if we want it to have any semantical meaning at all — denotes the entity ξ , and the fact that it also defines x , is just a *side effect* on the given environment. For example, “ $(7 + (x := 3)) \cdot 2$ ” is a well-defined expression, meaning “20”, with the side effect that x is defined to be 3. On the other hand, “if $x = 0$ then 1 else -1 ” is a well-defined expression, but “if $x := 0$ then -1 else 1” is not. However, we will usually avoid to use assignments inside other expressions, although it often makes sense (see example 4.3.3 again).

- (5) When “ ξ ” in “ $\xi := \omega$ ” is not only an identifier or symbol, but a more complex expression, then all the free variables in “ ξ ” are defined via pattern matching. For example, “ $\langle x, y, z \rangle := \langle 5, 5^2, 5^3 \rangle$ ” is in a certain sense just an abbreviation for the three subsequent assignments “ $x := 5$ ”, “ $y := 5^2$ ”, and “ $z := 5^3$ ”.
- (6) We said that each expression can have a value and a type. According to the previous remarks, these are as follows.
- (a) The type of an assignment “ $\xi := \omega$ ” is the type of “ ξ ”, which is identical with the type of “ ω ”, if “ ξ ” isn’t typed yet. Its value is ω itself.
- (b) The type of a type expression “ $\xi : \tau$ ” is τ , its value is the value of “ ξ ”, if such a value is defined in the current environment, and undefined otherwise.

5.3.7 Definition Local definitions

let ① **in** ② or alternatively
 ② **where** ① is our formalization of a local definition, where the assignments made in ① only hold for the expression ②.

5.3.8 Example local definition

For every two real numbers $p, q \in \mathbb{R}$ there are two complex numbers $x_1, x_2 \in \mathbb{C}$ that satisfy the equation $x^2 + p \cdot x + q = 0$. The function r with $r(p, q) = \langle x_1, x_2 \rangle$ is defined by

$$r := \left[\begin{array}{l} \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{C} \times \mathbb{C} \\ \langle p, q \rangle \mapsto \langle -\frac{p}{2} + w, -\frac{p}{2} - w \rangle \text{ where } w := \sqrt{\frac{p^2}{4} - q} \end{array} \right]$$

5.3.9 Definition Two kinds of values

We distinguish two kinds of values in mathematics:

(1) classes, i.e. collections or sets (see 5.6) or

(2) operators (see 5.7), i.e. values that may take arguments from some domain and return new values from some codomain; besides they may modify the given environment.

Accordingly, every expression is a class expression or an operator expression.

5.3.10 Remark

- (1) We introduce the essential kinds of operators in 5.7:
- (a) untyped *mappings* and typed (*partial functions*),
- (b) untyped *predicators* and typed *ordinary relations*, operators with the truth value codomain, which is abstracted away in the notation, so that only the domain remains.

Other basic concepts which we are about to define can be derived:

- (i) *constants* as nullary functions,
- (ii) *truth values* as particular constants,
- (iii) *tuples* as functions with an ordinal number domain,
- (iv) *records* (to be introduced in the main text) as generalized tuples,
- (v) *structures* as tuples or records,
- (vi) *environments* themselves as records or a pair of (partial) functions that attach types and values to expressions,

etc. Other familiar concepts in mathematics depend on their actual definition. For example, a *number* might be a class (as given by the ordinal number definition $0 := \emptyset$ and $n + 1 := n \cup \{n\}$) or a *numeral* (i.e. a tuples of digits), but also as a member of a given class. *Expressions* themselves can be formalized as *trees* (i.e. recursively defined tuples on the class of identifiers). *Identifiers* again may be implemented in various ways as well (see 5.2.3(3)).

Also, there are a couple of predefined operators like the conditional (5.4.3), the descriptor (5.4.4), operators on tuples, classes, formulas etc (see below).

- (2) Our *operator* terminology is quite uncommon. (In algebra this usually means a binary endofunction.) But it is very helpful in our text here and there is no standard equivalent to denote the common features of functions, relations and their relatives. It also allows some elegant reconstructions, such as the introduction of “*structures*” as “certain operators on given classes”.

5.3.11 Definition Terms and formulas

Expressions are also divided into

(1) formulas, i.e. expressions with *truth* or *boolean value* (see 5.5.2)

(2) terms, i.e. expressions that stand for “(real) things” and all other values.

5.3.12 Remark

- (1) For example, given the context of the usual integer structure (5.8.5), then “5”, “3+2” and “ $(x \mapsto 9-x)(4)$ ” are three example terms that denote the same value five. “ $3 \leq 5$ ” and “ $\exists n . n \leq 5$ ” are two examples of formulas, both (of value) true.
- (2) We now have two classifications for expressions: The *term-formula* difference and the distinction between *operator expressions* and *class expressions*.
- (3) The term-formula distinction has its origin in a long philosophical tradition that distinguishes two meaningful categories in every proper language: concepts and statements (neither title is a standard). Concepts denote more or less abstract things (“a horse”, “the tail of my horse”, “the intelligence of horses”, etc.) and statements are true or false assertions (“This horse is not intelligent.”). In mathematics this view is still dominant and in mathematical logic there usually is a constructive hierarchy of first terms and formulas second. But this design is somewhat artificial and does not really capture the real live of mathematics. Take for example the conditional expression “if φ then θ_1 else θ_2 ” (5.4.3), which is a basic ingredient of the mathematical language, but is hard to fit into the philosophy because it is a term that has a formula as argument, thus turns around the hierarchy. Many programming languages don’t suffer from this tradition anymore and subsume formulas under terms, e.g. as *boolean terms*.
- (4) There is a close relation between terms and formulas on one hand and the taxonomy of operators (5.7) on the other:
- (a) A *mapping* “ $x \mapsto \theta$ ” is the closure of the open term “ θ ”, the free variable “ x ” in “ θ ” is now bound and “ $x \mapsto \theta$ ” can now be applied to an argument “ y ”, written “ $(x \mapsto \theta)(y)$ ”.

A *function* is a typed mapping.

- (b) Similarly, a *predicator* “ $x \rightsquigarrow \varphi$ ” is the closure of the open formula “ φ ” and a *relation* is a typed predicator.

$\begin{cases} \theta_1 & \text{if } \varphi_1 \\ \vdots & \vdots \\ \theta_n & \text{if } \varphi_n \\ \theta_{n+1} & \text{else} \end{cases}$	is a more compact notation for the nested conditional expression $\begin{aligned} & \text{(if } \varphi_1 \\ & \text{then } \theta_1 \\ & \text{else (if } \varphi_2 \\ & \quad \text{then } \theta_2 \\ & \quad \text{else (... (if } \varphi_n \\ & \quad \quad \text{then } \theta_n \\ & \quad \quad \text{else } \theta_{n+1} \text{)...)})} \end{aligned}$
---	--

5.4 Special terms and tuples

5.4.1 Definition Tuples

$\langle x_1, \dots, x_n \rangle$	n -tuple, for $n \in \mathbb{N}$.
$\langle \rangle$	is the <u>empty tuple</u>
$\text{lg}(x)$	$:= n$ is the <u>length</u> of a given tuple $x = \langle x_1, \dots, x_n \rangle$
$x(i)$	$:= x_i$ the <u>i-th component</u> of a tuple $x = \langle x_1, \dots, x_n \rangle$ and $i \in \mathbf{n}$
$x \uparrow y$	$:= \langle x_1, \dots, x_n, y_1, \dots, y_m \rangle$ is the <u>concatenation</u> of two given tuples $x = \langle x_1, \dots, x_n \rangle$ and $y = \langle y_1, \dots, y_m \rangle$.

5.4.2 Remark Tuples and products

- (1) 2-, 3-, 4-, etc -tuples are also called (ordered) pairs, triples, quadruples, etc.
- (2) Note the clear difference between an element x and an unary tuple $\langle x \rangle$. Accordingly, the three expressions

$$\langle a, b, c \rangle \quad \langle a, \langle b, c \rangle \rangle \quad \langle \langle a \rangle, b, c \rangle$$

denote three different things. In many approaches, this distinction is blurred, but we will have to be consequent on this matter.

- (3) Many texts and programming language consider tuples (also called *sequences, lists, stacks, arrays, strings, etc.*, depending on the available operations) as zero-based and rather write $\langle x_0, \dots, x_{n-1} \rangle$ than $\langle x_1, \dots, x_n \rangle$. But we follow the older tradition in mathematics which is to start with 1.
- (4) In 9.2.1, we redefine tuples like $\langle x_1, \dots, x_n \rangle$ as (a special notation for) the surjective function (see 5.7.5)

$$\langle x_1, \dots, x_n \rangle := \left[\begin{array}{l} \mathbf{n} \longrightarrow \{x_1, \dots, x_n\} \\ i \mapsto x_i \end{array} \right]$$

with $\mathbf{n} := \{1, 2, \dots, n\}$. That way, tuples can be subsumed under the more general notion of *records* (see 9).

5.4.3 Definition Conditional terms

if φ then θ_1 else θ_2	is the usual form of a <u>conditional expression</u> , where φ is a formula and θ_1, θ_2 are terms.
--	---

$\begin{cases} \theta_1 & \text{if } \varphi_1 \\ \vdots & \vdots \\ \theta_n & \text{if } \varphi_n \\ \theta_{n+1} & \text{if } \varphi_{n+1} \end{cases}$	stands for	$\begin{cases} \theta_1 & \text{if } \varphi_1 \\ \vdots & \vdots \\ \theta_n & \text{if } \varphi_n \\ \text{undefined} & \text{else} \end{cases}$
But usually, the $\varphi_1, \dots, \varphi_n$ exhaust all possible cases and the “ undefined ” scenario never occurs.		

5.4.4 Definition Descriptions

the x with φ	is the general form of an <u>(untyped) description</u> , where x is a variable and φ is a formula (with x as free variable). It denotes the allegedly unique element x that makes φ true.
the $x : C$ with φ	or alternatively
the $x \in C$ with φ	is the <u>typed description</u> that restricts the x to be a member of the class C .
sing(C)	stands for “ the x with $x \in C$” and denotes the unique element of C and is well-defined iff C is a singleton (i.e. a one-element class, 5.6.14(1))

5.4.5 Remark

- (1) For example “**the $n : \mathbb{N}$ with $n \cdot n = 27$ ”** is a description for 9 and “**the $n : \mathbb{N}$ with $n \cdot n = 10$ ”** is undefined, because 10 doesn't have a square root in \mathbb{N} .
- (2) In the constitutive days of modern logic, Russell and others used the iota symbol “ ι ” for descriptions, Ackermann and Hilbert the “ ϵ ”, but most mathematical texts don't use an explicit formalism for descriptions at all. However it is an essential feature in (meta)mathematics, in particular for the function concept (5.7.4).

5.5 Formulas

5.5.1 Definition Equation

$\textcircled{1} = \textcircled{2}$	is the general form of an <u>equation</u> , expressing that $\textcircled{1}$ and $\textcircled{2}$ are <u>equal</u> or <u>identical</u> .
-------------------------------------	--

5.5.2 Definition — truth values

false or **0** is the zero bit or false value

true or **1** is the unit bit or true value

\mathbb{B} := $\{0, 1\}$ is the truth or bit or boolean value class

5.5.3 Definition — Junction of formulas

Let $\varphi, \varphi_1, \varphi_2$ be given formulas.

$\neg\varphi$ “not φ ” (negation)

$\varphi_1 \wedge \varphi_2$ “ φ_1 and φ_2 ” (conjunction)

$\varphi_1 \vee \varphi_2$ “ φ_1 or φ_2 ” (conjunction)

$\varphi_1 \rightarrow \varphi_2$:= $\neg\varphi_1 \vee \varphi_2$ (subjunction)

$\varphi_1 \leftrightarrow \varphi_2$:= $(\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$ (equijunction)

5.5.4 Remark

5.5.5 Definition — Quantifications

$\exists x . \varphi$ is a existentialization, where x is a variable and φ a formula, is saying that “there is (at least) one x such that φ (is true)”.

$\forall x . \varphi$ is a generalization, where x is a variable and φ a formula, is saying that “for all x , φ (is true)”.

$\exists x : C . \varphi$ is the explicitly typed existentialization that restricts the x to be members of the class C .

$\forall x : C . \varphi$ is the explicitly typed generalization that restricts the x to be members of the class C .

$\exists x \in C . \varphi$ stands for $\exists x : C . \varphi$ and

$\forall x \in C . \varphi$ stands for $\forall x : C . \varphi$.

5.5.6 Remark

- (1) Again we often use the words “for each” and “for all” instead of the just defined symbols.

5.5.7 Definition — Sub- and equivalence of closed formulas

$\varphi_1 \Rightarrow \varphi_2$ “ φ_1 entails φ_2 ” or “ φ_2 follows (or derives) from φ_1 ” is the subvalence (or entailment or consequence) relation, and

$\varphi_1 \Leftrightarrow \varphi_2$ “ φ_1 is equivalent to φ_2 ” is the equivalence relation, on closed formulas (i.e. sentences) φ_1, φ_2 .

5.5.8 Remark

- (1) For every $n \in \mathbb{N}$ and formulas $\varphi_1, \dots, \varphi_n$ we put

$$\varphi_1 \wedge \dots \wedge \varphi_n := \begin{cases} \mathbf{true} & \text{if } n = 0 \\ \varphi_1 \wedge (\varphi_1 \wedge \dots \wedge \varphi_n) & \text{else} \end{cases}$$

$$\varphi_1 \vee \dots \vee \varphi_n := \begin{cases} \mathbf{false} & \text{if } n = 0 \\ \varphi_1 \vee (\varphi_1 \vee \dots \vee \varphi_n) & \text{else} \end{cases}$$

- (2) We also suggest an n -ary generalization for the sub- and equijunction:

$$\varphi_1 \rightarrow \dots \rightarrow \varphi_n := (\varphi_1 \rightarrow \varphi_2) \wedge \dots \wedge (\varphi_{n-1} \rightarrow \varphi_n)$$

$$\varphi_1 \leftrightarrow \dots \leftrightarrow \varphi_n := (\varphi_1 \rightarrow \dots \rightarrow \varphi_n) \wedge (\varphi_n \rightarrow \dots \rightarrow \varphi_1)$$

occasionally useful to express a chain condition or an equivalence class condition, respectively.

- (3) We are not consequently formalistic and often use “not”, “and”, “or” etc. instead of the just defined “ \neg ” etc. Also

$$\text{“} \varphi_1, \dots, \varphi_n \text{”}$$

is an often used alternative notation for “ $\varphi_1 \wedge \dots \wedge \varphi_n$ ” to express the simultaneous truth of multiple formulas. This is often applied in class expressions “ $\{x \mid \varphi_1, \dots, \varphi_n\}$ ”.

- (1) Note the difference between the *subjunction* \rightarrow , which takes two formulas and returns a new one, and the *subvalence relation* \Rightarrow , which compares two closed formulas (also called sentences) and is an either true or false statement. The same goes for the *equijunction* \leftrightarrow and the *equivalence* \Leftrightarrow .

- (2) However and despite the differences we often inaccurately use the phrases “ $\textcircled{1}$ implies $\textcircled{2}$ ” or even “if $\textcircled{1}$ then $\textcircled{2}$ ” to verbalize both “ $\textcircled{1} \rightarrow \textcircled{2}$ ” and “ $\textcircled{1} \Rightarrow \textcircled{2}$ ”. And similarly we follow the common habit and use “iff” for both \leftrightarrow and \Leftrightarrow .

- (3) There are two different approaches to define the subvalence relation \Rightarrow :

(a) Semantically, \Rightarrow is introduced as the model-theoretical relation \models , saying that $\varphi_1 \models \varphi_2$ iff each model of φ_1 is a model of φ_2 as well.

(b) Syntactically, \Rightarrow is introduced as the proof-theoretical relation \vdash , saying that $\varphi_1 \vdash \varphi_2$ iff φ_2 can be derived by transforming φ_1 according to simple replacement rules (such as 5.5.9, 5.5.10, 5.5.11)

Most of the time (i.e. in *first-order predicate logic*), these two approaches are equivalent. That allows us to use our \Rightarrow for both \models and \vdash .

- (4) Now given the subvalence relation, the equivalence can be properly defined as its equivalence relation in the sense that

$\varphi_1 \Leftrightarrow \varphi_2$ iff $(\varphi_1 \Rightarrow \varphi_2 \text{ and } \varphi_2 \Rightarrow \varphi_1)$.

5.5.9 Lemma Boolean laws

For all formulas $\varphi, \psi, \chi, \varphi_1, \dots, \varphi_n$ with $n \in \mathbb{N}$ and every bijective function $i : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ the following statements hold:

- (1) $\mathbf{0} \Rightarrow \varphi$ (least element)
- (2) $\varphi \Rightarrow \mathbf{1}$ (greatest element)
- (3) $\mathbf{1} \wedge \varphi \Leftrightarrow \varphi$ (neutral element of \wedge)
- (4) $\mathbf{0} \vee \varphi \Leftrightarrow \varphi$ (neutral element of \vee)
- (5) $\varphi \wedge \neg\varphi \Leftrightarrow \mathbf{0}$ (conjunctive complement)
- (6) $\varphi \vee \neg\varphi \Leftrightarrow \mathbf{1}$ (disjunctive complement)
- (7) $\varphi_1 \wedge \dots \wedge \varphi_n \Leftrightarrow \varphi_{i(1)} \wedge \dots \wedge \varphi_{i(n)}$ (commutativity of \wedge)
- (8) $\varphi_1 \vee \dots \vee \varphi_n \Leftrightarrow \varphi_{i(1)} \vee \dots \vee \varphi_{i(n)}$ (commutativity of \vee)
- (9) $\varphi \wedge \varphi \Leftrightarrow \varphi$ (idempotency of \wedge)
- (10) $\varphi \vee \varphi \Leftrightarrow \varphi$ (idempotency of \vee)
- (11) $\varphi \wedge (\psi \wedge \chi) \Leftrightarrow (\varphi \wedge \psi) \wedge \chi$ (associativity of \wedge)
- (12) $\varphi \vee (\psi \vee \chi) \Leftrightarrow (\varphi \vee \psi) \vee \chi$ (associativity of \vee)
- (13) $\psi \wedge (\varphi_1 \vee \dots \vee \varphi_n) \Leftrightarrow (\psi \wedge \varphi_1) \vee \dots \vee (\psi \wedge \varphi_n)$
(\wedge distributes over \vee)
- (14) $\psi \vee (\varphi_1 \wedge \dots \wedge \varphi_n) \Leftrightarrow (\psi \vee \varphi_1) \wedge \dots \wedge (\psi \vee \varphi_n)$
(\vee distributes over \wedge)
- (15) $\neg(\varphi_1 \wedge \dots \wedge \varphi_n) \Leftrightarrow \neg\varphi_1 \vee \dots \vee \neg\varphi_n$ (de Morgan's law)
- (16) $\neg(\varphi_1 \vee \dots \vee \varphi_n) \Leftrightarrow \neg\varphi_1 \wedge \dots \wedge \neg\varphi_n$ (de Morgan's law)

5.5.10 Lemma Laws on sub- and equijunctions

For all formulas φ and ψ holds:

- (1) $\varphi \rightarrow \psi \Leftrightarrow \neg\varphi \vee \psi$
- (2) $\varphi \leftrightarrow \psi \Leftrightarrow (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
- (3) $\varphi \leftrightarrow \psi \Leftrightarrow (\neg\varphi \wedge \neg\psi) \vee (\varphi \wedge \psi)$
- (4) $\varphi \leftrightarrow \mathbf{0} \Leftrightarrow \neg\varphi$
- (5) $\varphi \leftrightarrow \mathbf{1} \Leftrightarrow \varphi$
- (6) $\varphi \rightarrow \mathbf{0} \Leftrightarrow \neg\varphi$
- (7) $\varphi \rightarrow \mathbf{1} \Leftrightarrow \mathbf{1}$
- (8) $\mathbf{0} \rightarrow \varphi \Leftrightarrow \mathbf{1}$
- (9) $\mathbf{1} \rightarrow \varphi \Leftrightarrow \varphi$
- (10) $\varphi \Rightarrow \psi$ iff $\varphi \rightarrow \psi \Leftrightarrow \mathbf{1}$
- (11) $\varphi \Leftrightarrow \psi$ iff $\varphi \leftrightarrow \psi \Leftrightarrow \mathbf{1}$

5.5.11 Lemma Laws on quantifications

Let v, w be variables and C, D class symbols and φ, ψ formulas.

(1) The following statements hold in general:

- (a) $\neg\forall v \in C. \varphi \Leftrightarrow \exists v \in C. \neg\varphi$
- (b) $\neg\exists v \in C. \varphi \Leftrightarrow \forall v \in C. \neg\varphi$
- (c) $(\forall v \in C. \varphi) \wedge (\forall v \in C. \psi) \Leftrightarrow \forall v \in C. (\varphi \wedge \psi)$
- (d) $(\exists v \in C. \varphi) \wedge (\exists v \in C. \psi) \Rightarrow \exists v \in C. (\varphi \wedge \psi)$
- (e) $(\forall v \in C. \varphi) \vee (\forall v \in C. \psi) \Leftrightarrow \forall v \in C. (\varphi \vee \psi)$
- (f) $(\exists v \in C. \varphi) \vee (\exists v \in C. \psi) \Leftrightarrow \exists v \in C. (\varphi \vee \psi)$

(2) The following statements are true as well:

- (a) $\forall v \in C. \forall w \in D. \varphi \Leftrightarrow \forall w \in D. \forall v \in C. \varphi$
- (b) $\exists v \in C. \exists w \in D. \varphi \Leftrightarrow \exists w \in D. \exists v \in C. \varphi$
- (c) $\exists v \in C. \forall w \in D. \varphi \Rightarrow \forall w \in D. \exists v \in C. \varphi$
- (d) $\forall v \in C. \exists w \in D. \varphi \Leftarrow \exists w \in D. \forall v \in C. \varphi$

(3) If w does not occur in φ , then

- (a) $\forall w \in C. \varphi \Leftrightarrow \varphi$ (redundant variable)
- (b) $\exists w \in C. \varphi \Leftrightarrow \varphi$ (redundant variable)
- (c) $\forall v \in C. \varphi \Leftrightarrow \forall w \in C. \varphi[v/w]$ (renaming)
- (d) $\exists v \in C. \varphi \Leftrightarrow \exists w \in C. \varphi[v/w]$ (renaming)

(4) If v is not a free variable in ψ , then

- (a) $\forall v \in C. \psi \Leftrightarrow \psi$
- (b) $\exists v \in C. \psi \Leftrightarrow \psi$
- (c) $\forall v \in C. (\varphi \wedge \psi) \Leftrightarrow (\forall v \in C. \varphi) \wedge \psi$
- (d) $\exists v \in C. (\varphi \wedge \psi) \Leftrightarrow (\exists v \in C. \varphi) \wedge \psi$
- (e) $\forall v \in C. (\varphi \vee \psi) \Leftrightarrow (\forall v \in C. \varphi) \vee \psi$
- (f) $\exists v \in C. (\varphi \vee \psi) \Leftrightarrow (\exists v \in C. \varphi) \vee \psi$
- (g) $\forall v \in C. (\varphi \rightarrow \psi) \Leftrightarrow (\exists v \in C. \varphi) \rightarrow \psi$
- (h) $\forall v \in C. (\psi \rightarrow \varphi) \Leftrightarrow \psi \rightarrow \forall v \in C. \varphi$
- (i) $\exists v \in C. (\varphi \rightarrow \psi) \Leftrightarrow (\forall v \in C. \varphi) \rightarrow \psi$
- (j) $\exists v \in C. (\psi \rightarrow \varphi) \Leftrightarrow \psi \rightarrow \exists v \in C. \varphi$

(5) If C denotes a non-empty class, then

- (a) $\forall v \in C. \varphi \Rightarrow \exists v \in C. \varphi$

5.6 Classes and numbers

5.6.1 Introduction

- (1) A **class** is a collection of its **members** (or **elements**). A precise definition is presented by the syntax of class expressions, as introduced below.
- (2) A **set** is a special kind of ("small") class, which must be specified according to certain constraints (e.g. specified by the *Zermelo-Fränkel axioms*) that guarantee the class to be well-defined. Most mathematics can be done with the set concept. However, we prefer to talk about "classes".
- (2) A **class family** is (another name for) a class of classes.

5.6.2 Definition Class expressions

$\{x \mid \varphi\}$

"the class of all x satisfying φ " is the default form of a class expression, where x is a variable and φ a formula with no more free variables than x .

$\{x \in C \mid \varphi(x)\}$

is a common version for the proper class expression $\{x : C \mid \varphi(x)\}$.

$\{x_1, \dots, x_n\}$

$:= \{x \mid x = x_1 \vee \dots \vee x = x_n\}$ is the default form of a **finite class expression**

\emptyset

$:= \{\}$ is the **empty class**

$\{\theta \mid \varphi\}$

is a generalization of the default class expression, where θ is any term, φ a formula, and θ and φ have the same free variables x_1, \dots, x_n . This form is equal to the proper class expression $\{y \mid (y = \theta \wedge \varphi)\}$, where y is a variable different to the x_i .

$\{\theta \mid \varphi_1, \dots, \varphi_n\}$

stands for $\{\theta \mid \varphi_1 \wedge \dots \wedge \varphi_n\}$

$\{\theta_1, \dots, \theta_n \mid \varphi\}$

stands for $\bigcup_{i=1}^n \{\theta_i \mid \varphi\}$.

5.6.3 Definition — The element relation

$e \in C$ “ e is in or belongs to (class) C ” is an instance of the element or membership relation, where e is a term and C a class expression. If a default class expression for C is given by $C = \{x \mid \varphi\}$, the relation is defined as: $e \in C$ iff $\varphi[x/e]$ is true.

$e \notin C$ iff $\neg(e \in C)$, “ e is not in C ”.

5.6.4 Definition — Operations on classes

$C_1 \subseteq C_2$ iff $\forall x. (x \in C_1 \rightarrow x \in C_2)$ (inclusion)

$C_1 \not\subseteq C_2$ iff $\neg(C_1 \subseteq C_2)$

$C_1 \subset C_2$ iff $(C_1 \subseteq C_2 \wedge C_1 \neq C_2)$ (proper incl.)

$C_1 \not\subset C_2$ iff $\neg(C_1 \subset C_2)$

$C_1 \supseteq C_2$ iff $(C_2 \subseteq C_1)$

$C_1 \cap C_2 := \{x \mid x \in C_1 \wedge x \in C_2\}$ (intersection)

$C_1 \cup C_2 := \{x \mid x \in C_1 \vee x \in C_2\}$ (union)

$C_1 \uplus C_2 := C_1 \cup C_2$, only defined if $C_1 \cap C_2 = \emptyset$

(disjunct union)

$\bigcap \mathcal{K} := \{x \mid \forall C \in \mathcal{K}. x \in C\}$ (big intersection)

$\bigcup \mathcal{K} := \{x \mid \exists C \in \mathcal{K}. x \in C\}$ (big union)

$C_1 \setminus C_2 := \{x \mid x \in C_1 \wedge x \notin C_2\}$ (difference)

$\mathbf{P}(C) := \{D \mid D \subseteq C\}$ (power class)

$\mathbf{Fin}(C) := \{D \mid D \subseteq C, D \text{ is finite}\}$

(finite classes)

$\mathbf{Sg}(C) := \{\{x\} \mid x \in C\}$ (singleton class)

sing(C) (see 5.4.4) is the unique member of the given class C , which is undefined in case C is not a singleton (5.6.14(1)).

5.6.5 Lemma — Properties of class differences

For all classes A, B, C holds

(1) $A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$

(2) $A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C)$

(3) $(B \cap C) \setminus A = (B \setminus A) \cap (C \setminus A)$

(4) $(B \cup C) \setminus A = (B \setminus A) \cup (C \setminus A)$

(5) $(A \setminus B) \setminus C = A \setminus (B \cup A)$

(6) $A \setminus (B \setminus C) = (A \setminus B) \cup (A \cap C)$

(7) $(A \setminus B) \cap C = (A \cap C) \setminus B$

(8) $(A \setminus B) \cup C = (A \cup C) \setminus (B \setminus C)$

5.6.6 Definition — Oppositions

For every class \mathcal{K} of classes we define

$$\nabla \mathcal{K} := \bigcup_{C \in \mathcal{K}} (C \setminus \bigcup \{C\})$$

the opposition of \mathcal{K}

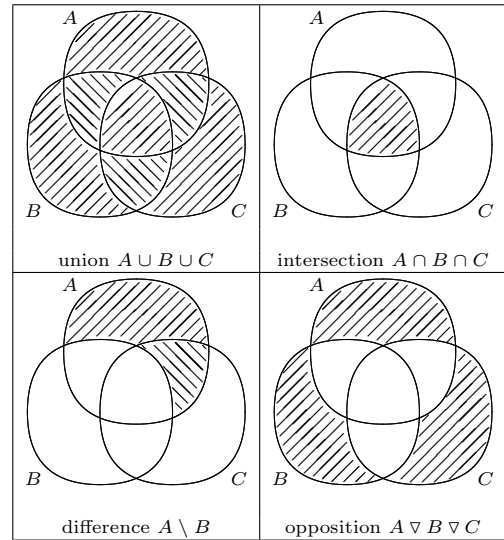
For any finite number of classes C_1, \dots, C_n we put

$$C_1 \nabla \dots \nabla C_n := \nabla \{C_1, \dots, C_n\}$$

the opposition of C_1, \dots, C_n

5.6.7 Remark — opposition and other class operations

- (1) For three given classes A, B, C the following *Venn diagrams* (the shadowed area is the result of the operation) visualize the basic operations on classes:



- (2) The opposition $A \nabla B = (A \setminus B) \cup (B \setminus A)$ is more often called the symmetric difference, but there is no real standard symbol.

- (3) The result of the opposition is the class of all the elements that occur in exactly one of the argument classes. In other words, for every class family \mathcal{K} holds:

$$\nabla \mathcal{K} = \{x \in \bigcup \mathcal{K} \mid x \in C \text{ for exactly one } C \in \mathcal{K}\}$$

- (4) Note, that the opposition shows some unusual behavior for a class operation, e.g.

$$A \nabla B \nabla C \neq (A \nabla B \nabla C) \cup (A \cap B \cap C)$$

$$= (A \nabla B) \nabla C$$

$$= A \nabla (B \nabla C)$$

- (5) An often used (but hardly ever formalized) operation is the inverse operation of “ $x \mapsto \{x\}$ ”, used to extract the (unique) element x of a given class C . We use the descriptor “**sing**(C)” (5.4.4) to achieve that. The result is well-defined iff C is a singleton.

5.6.8 Definition Operator classes

$D \dashrightarrow C$	the <u>partial function class</u> from D to C (see 5.7.15)
$D \longrightarrow C$	the <u>function class</u> from D to C (see 5.7.3)
$D_1 \rightsquigarrow \dots \rightsquigarrow D_n$	the (ordinary) <u>relation class</u> on D_1, \dots, D_n (see 5.7.10), which is written
$\text{Pty}(D_1)$	in case $n = 1$, because the unary relation is a <u>property</u> on D_1 , and
\mathbb{B}	for $n = 0$, because the nullary relation is a constant one of two possible truth values (5.5.2).

5.6.9 Definition Cartesian products

$C_1 \times \dots \times C_n$	$:= \left\{ \langle x_1, \dots, x_n \rangle \mid x_1 \in C_1, \dots, x_n \in C_n \right\}$ is the <u>cartesian product</u> of classes C_1, \dots, C_n
C^n	$:= \left\{ \langle x_1, \dots, x_n \rangle \mid x_1, \dots, x_n \in C \right\}$ the n -th (cartesian) <u>power</u> of a class C and $n \in \mathbb{N}$
C^*	$:= \bigcup_{n \in \mathbb{N}} C^n$ the <u>tuple class</u> (or <u>Kleene closure</u>) of a class C
$\{\langle \rangle\}$	$= C^0 = \emptyset^0$ is the <u>nullary product</u> , which is the same for every class C

5.6.10 Remark Cartesian products

- The " $C_1 \times \dots \times C_n$ " notation is properly writable only for $n \geq 2$. If we actually need to write an unary product, we use " C^1 ". (In section 12 we generalize the product notion and provide alternative notations.)
- We already emphasized (5.4.2) the difference between an element x and an unary tuple $\langle x \rangle$. Accordingly, a class C and its first power $C^1 = \{\langle x \rangle \mid x \in C\}$ are different. As another consequence, our strict version of the cartesian product is not associative:

$$A \times (B \times C) \quad (A \times B) \times C \quad A \times B \times C$$
are three different tuple classes.
- In 12.3.2 we define, for any two tuple classes Ξ and Ξ' the concatenation

$$\Xi \dagger \Xi' := \{ \xi \dagger \xi' \mid \xi \in \Xi, \xi' \in \Xi' \}$$

This provides us with the means to combine cartesian products

$$(A \times B) \dagger (C \times D) = A \times B \times C \times D$$

and to attach classes to existing cartesian product

$$A^1 \dagger (B \times C) = (A \times B) \dagger C^1 = A \times B \times C$$

- Often used in computer science are

- Char**, a finite class of characters, made of elements like

'0' ... '9' 'A' ... 'Z' 'a' ... 'z' ...

Often used ensembles are the 7-bit or 8-bit ASCII code

with either 128 or 256 different elements.

- String** := **Char**^{*}, the strings, where the usual notation for these character tuples is

"Hallo!" instead of $\langle 'H', 'a', 'l', 'l', 'o', '!', '\rangle$

5.6.11 Definition Number classes

\mathbb{N}	$:= \{0, 1, 2, \dots\}$	the <u>natural numbers</u>
\mathbb{Z}	$:= \{\dots, -1, 0, 1, 2, \dots\}$	the <u>integers</u>
\mathbb{Q}	$:= \{n/d \mid n, d \in \mathbb{Z}, d \neq 0\}$	the <u>rational numbers</u>
\mathbb{R}		stands for the <u>real numbers</u>
\mathbf{n}	$:= \{1, 2, \dots, n-1, n\}$	for every $n \in \mathbb{N}$

5.6.12 Definition Operations on numbers

0	1	+	-	·	/	≤	<	min	max	...
are the usual operations on numbers										

5.6.13 Definition Cardinality of classes

card (C)	the <u>cardinality</u> of C , a (possibly transfinite) cardinal number
---------------------	--

5.6.14 Definition Properties of classes

A class C is called
(1) a <u>singleton</u> , if card (C) = 1, i.e. $C = \{e\}$ for some e .
(2) <u>finite</u> , if card (C) $\in \mathbb{N}$
(3) <u>countable</u> or <u>enumerable</u> , if card (C) $\leq \mathbf{card}(\mathbb{N})$

5.6.15 Remark Cardinal number arithmetic

A generalization of the arithmetic of natural numbers (i.e. \mathbb{N} together with the linear order \leq , addition $+$, and multiplication \cdot) is the arithmetic of cardinal numbers. The finite cardinal numbers are exactly the natural numbers $0, 1, 2, \dots$. But there are also infinite (or transfinite) cardinals, beginning with the cardinality $\aleph_0 := \mathbf{card}(\mathbb{N})$, which is the cardinality of enumerable classes, and \aleph_1 , the cardinality of the real numbers. The arithmetic operations can be generalized and we have, e.g.

$$\begin{aligned} 2 < 9 & \quad 2 < \aleph_0 & \quad \aleph_0 < \aleph_1 \\ 2 + 9 = 11 & \quad 2 + \aleph_0 = \aleph_0 & \quad \aleph_0 + \aleph_0 = \aleph_0 \\ 2 \cdot 9 = 18 & \quad 2 \cdot \aleph_0 = \aleph_0 \end{aligned}$$

The addition and multiplication can also be generalized to an arbitrary number of arguments. For any class $\{\kappa_i \mid i \in I\}$ of cardinal numbers,

$$\begin{aligned} \sum \{ \kappa_i \mid i \in I \} & \text{ or } \sum_{i \in I} \kappa_i & \text{ denotes the } \underline{\text{sum}} & \text{ of the } \kappa_i \\ \prod \{ \kappa_i \mid i \in I \} & \text{ or } \prod_{i \in I} \kappa_i & \text{ denotes the } \underline{\text{product}} & \text{ of the } \kappa_i \end{aligned}$$

where in particular

$$\sum \emptyset = 0 \quad \text{and} \quad \prod \emptyset = 1$$

The generalization is given by the fact that for finite cardinal number records

$$\sum_{i \in \{i_1, \dots, i_n\}} \kappa_i = \kappa_{i_1} + \dots + \kappa_{i_n}$$

$$\prod_{i \in \{i_1, \dots, i_n\}} \kappa_i = \kappa_{i_1} \cdot \dots \cdot \kappa_{i_n}$$

5.7 Operations

5.7.1 Definition Mappings

$x \mapsto \theta$	is a <u>map(ping) (expression)</u> , where x is a variable and θ a term.
$(x \mapsto \theta)(\sigma)$	is the <u>application</u> of the map on a given term σ . The result of this application is the term θ' , obtained from θ by replacing each free occurrence of x in θ by σ .
$\langle x_1, \dots, x_n \rangle \mapsto \theta$	is a <u>n-ary</u> or ordinary <u>map(ping) expression</u> , where $n \in \mathbb{N}$, the x_1, \dots, x_n are pairwise different variables and θ is a term with no more free variables than the x_i .
$\mu(\sigma_1, \dots, \sigma_n)$	is the <u>application</u> of the n -ary map $\mu = (\langle x_1, \dots, x_n \rangle \mapsto \theta)$ on a tuple $\langle \sigma_1, \dots, \sigma_n \rangle$ of terms. The result is θ' , obtained from θ by simultaneously replacing each free occurrence of an x_i in θ by the according σ_i .
$\begin{bmatrix} \xi_1 \mapsto \theta_1 \\ \vdots \\ \xi_n \mapsto \theta_n \end{bmatrix}$	stands for $x \mapsto \begin{cases} \theta_1 & \text{if } x = \xi_1 \\ \theta_2 & \text{if } x = \xi_2 \\ \vdots \\ \theta_n & \text{if } x = \xi_n \end{cases}$

5.7.2 Remark

- (1) For example, “ $x \mapsto (7 \cdot x + 2)$ ” is a map expression, which, applied to 5, returns $(x \mapsto (7 \cdot x + 2))(5) = 7 \cdot 5 + 2 = 37$ under default interpretations of the symbols involved.
- (2) There is also a tradition to write “ $x \mapsto \theta$ ” as a so-called lambda expression “ $\lambda x. \theta$ ”.

5.7.3 Definition Functions

$f := \begin{bmatrix} D \longrightarrow C \\ x \mapsto \theta \end{bmatrix}$	is our default form of a <u>(total) function</u> definition, saying that the identifier f is defined to be a (total) function from a class D into a class C . θ is a term of type C with no more free variables than x .
$\text{dom}(f)$	$:= D$ is the <u>domain</u> of f
$\text{cod}(f)$	$:= C$ is the <u>codomain</u> of f
$f(a)$ or fa	“ <u>f of a”</u> , the default notation for the <u>application</u> of f onto a , is the unique value or image $b \in C$ for the given argument $a \in D$, given by $b = (x \mapsto \theta)(a)$.
$\textcircled{1} f \textcircled{2}$ $\textcircled{1}^g$ $h \textcircled{2}$	etc., usually on the left side of a function definition, denotes that an application of a n -ary function f is not written in its usual prefix notation “ $f(\langle x_1, \dots, x_n \rangle)$ ”, but in the specified way, i.e. that we write “ xfy ” instead of “ $f(x, y)$ ”, “ x^g ” instead of “ $g(x)$ ”, “ h_y^x ” instead of “ $h(x, y)$ ”, etc.

5.7.4 Remark

- (1) In set-theoretical terms a function f is usually defined as a triple $f = \langle D, C, \Gamma \rangle$, where D and C are classes and $\Gamma \subseteq D \times C$ is left total and right unique (i.e. for each $a \in D$ there is exactly one $b \in C$ with $\langle a, b \rangle \in \Gamma$). This definition is easily converted to our default function form via

$$f = \begin{bmatrix} D \longrightarrow C \\ x \mapsto (\text{the } b : C \text{ with } \langle x, b \rangle \in \Gamma) \end{bmatrix}$$

- (2) Functions are usually defined as ordinary or n -ary functions, which means that the domain is meant to be an n -ary cartesian product $D_1 \times \dots \times D_n$. So our standard form of an ordinal function definition becomes

$$\begin{bmatrix} D_1 \times \dots \times D_n \longrightarrow C \\ \langle x_1, \dots, x_n \rangle \mapsto \theta \end{bmatrix}$$

And the application of such an n -ary f to and argument $\langle a_1, \dots, a_n \rangle$ is usually written

$$f(a_1, \dots, a_n) \quad \text{as a short form for} \quad f(\langle a_1, \dots, a_n \rangle)$$

5.7.5 Definition Properties of functions

A function $f : X \longrightarrow Y$ is

- (1) injective or an injection,
if $x_1 \neq x_2$ implies $f(x_1) \neq f(x_2)$, for all $x_1, x_2 \in X$.
- (2) surjective or a surjection,
if $Y = \{f(x) \mid x \in X\}$.
- (3) bijjective or a bijection,
if it is both injective and surjective.

5.7.6 Definition Operations on functions

$\langle \rangle$	$:= (\text{the } f : \emptyset \longrightarrow \emptyset)$
	is the <u>empty function</u> (or <u>empty tuple</u> , see 5.4.1)
$f _Z$	$:= \begin{bmatrix} Z \longrightarrow C \\ x \mapsto f(x) \end{bmatrix}$
	is the <u>(domain) restriction</u> of a given function $f : D \longrightarrow C$ onto some $Z \subseteq D$.
$g \circ f$	$:= \begin{bmatrix} A \longrightarrow C \\ x \mapsto g(f(x)) \end{bmatrix}$
	is the <u>composition</u> of two given functions $f : A \longrightarrow B$ and $g : B \longrightarrow C$.
id_C	$:= \begin{bmatrix} C \longrightarrow C \\ c \mapsto c \end{bmatrix}$
	is the <u>identity function</u> of a given class C .

5.7.7 Remark Operations on functions

- Let $f : D \longrightarrow C$ be a function. The essential characterization of a function (see 5.7.4) is the fact that for each $a \in D$ there is a unique $b \in C$ defined by f . Accordingly, if $D = \emptyset$, then f is uniquely defined for every class C . And if $C = \emptyset$, then $f : D \longrightarrow C$ is defined only if $D = \emptyset$, too (otherwise, there would be a $a \in D$ without a corresponding $f(a)$ in C). In particular, the *empty function*, i.e. the unique function of type $\emptyset \longrightarrow \emptyset$, is well defined.
- The *empty function* is identical with the *empty tuple*, both written " $\langle \rangle$ ", because tuples will be redefined (see 5.6.10) as surjective functions $\mathbf{n} \longrightarrow C$, and for $\mathbf{n} = \mathbf{0} = \emptyset$ there is just one surjective function of type $\mathbf{n} \longrightarrow C$, the one with $C = \emptyset$, and that is the empty function.

5.7.8 Definition Predicators

$x \rightsquigarrow \varphi$	is a <u>predicator (expression)</u> , where x is a variable and φ a formula.
$(x \rightsquigarrow \varphi)(\theta)$	is the application of the predicator on a given term θ . The result of this application is the formula φ' , obtained from φ by replacing each free occurrence of x in φ by θ .
$\langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi$	is a <u>n-ary</u> or <u>ordinary predicator (expression)</u> , where $n \in \mathbb{N}$, the x_1, \dots, x_n are pairwise different variables and φ is a formula with no more free variables than the x_i .
$\pi(\theta_1, \dots, \theta_n)$	is the application of the n -ary predicator $\pi = (\langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi)$ on a term tuple $(\theta_1, \dots, \theta_n)$. The result is φ' , obtained from φ by simultaneously replacing each free occurrence of an x_i in φ by the according θ_i .
$\begin{bmatrix} \xi_1 \rightsquigarrow \varphi_1 \\ \vdots \\ \xi_n \rightsquigarrow \varphi_n \end{bmatrix}$	stands for $x \rightsquigarrow \begin{cases} \varphi_1 & \text{if } x = \xi_1 \\ \varphi_2 & \text{if } x = \xi_2 \\ \vdots & \vdots \\ \varphi_n & \text{if } x = \xi_n \end{cases}$

5.7.9 Remark

- For example, a predicator π is defined by

$$\pi := (n \rightsquigarrow \exists r : \mathbb{N} . r \cdot r = n)$$

So $\pi(9)$ becomes **true** and $\pi(10)$ is **false** (which is why we would rather write $\neg\pi(10)$ in concrete situations).

- Not every predicator application is well-defined. For the previously given example, $\pi(\langle 9, 10 \rangle)$ usually isn't. A *relation* now is a predicator together with a domain. Each of the domain members is guaranteed to be a well-defined argument for an application. In fact, we hardly ever use predicators as such, but only as part of a relation.

5.7.10 Definition Ordinary relations

$R := \begin{bmatrix} D_1 \rightsquigarrow \dots \rightsquigarrow D_n \\ \langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi \end{bmatrix}$	is our default form of an <u>ordinary</u> or <u>n-ary relation</u> definition, where the D_i are class expressions, the x_i are variables and φ is a formula with no more free variables than the x_i .
$\text{dom}(R) := D_1 \times \dots \times D_n$	is the <u>domain</u> of R
$\text{gr}(R) := \left\{ \begin{array}{l} \langle x_1, \dots, x_n \rangle \in D_1 \times \dots \times D_n \\ \varphi(x_1, \dots, x_n) \end{array} \right\}$	is the <u>graph</u> of R .
$\langle \theta_1, \dots, \theta_n \rangle \in R$ or $R(\theta_1, \dots, \theta_n)$	" R holds for $\langle \theta_1, \dots, \theta_n \rangle$ ", are two ways of expressing the same fact, that the application of $\langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi$ on $\langle \theta_1, \dots, \theta_n \rangle \in D_1 \times \dots \times D_n$ is true.
$\langle \theta_1, \dots, \theta_n \rangle \notin R$ or $\neg R(\theta_1, \dots, \theta_n)$	" R <u>does not hold</u> for $\langle \theta_1, \dots, \theta_n \rangle$ ", on the other hand are saying that the application of $\langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi$ on $\langle \theta_1, \dots, \theta_n \rangle \in D_1 \times \dots \times D_n$ is false.
① R ②	indicates the use of the infix notation " xRy " for the application of a binary relation R rather than the default prefix notation " $R(x, y)$ ".

5.7.11 Remark

According to 5.6.8 and 5.7.8, the default form for an unary relation is then given by

$$\begin{bmatrix} \text{Pty}(D) \\ x \rightsquigarrow \varphi \end{bmatrix}$$

5.7.12 Definition Special notations

$x_1, \dots, x_n R y$	is an often used abbreviation for “ $x_1 R y \wedge \dots \wedge x_n R y$ ”, where $\textcircled{1} R \textcircled{2} : A \rightsquigarrow B$ is a binary relation in infix notation, $x_1, \dots, x_n \in A$, and $y \in B$. For example, “ $a, b, c \in C$ ”.
$x_0 R_1 x_1 R_2 \dots R_n x_n$	is an often used abbreviation for “ $x_0 R_1 x_1 \wedge x_1 R_2 x_2 \wedge \dots \wedge x_{n-1} R_n x_n$ ”, where the R_i are binary relations in infix notation and the x_i are defined arguments. For example, “ $\emptyset \neq A \subseteq B \subset C$ ”.

5.7.13 Definition image and coimage classes of binary relations

$R[Z]$	$:= \{y \in Y \mid x \in (Z \cap X) \text{ and } \langle x, y \rangle \in R\}$ is the R -image class of Z , for $R : X \rightsquigarrow Y$ and a class Z
$R^{-1}[Z]$	$:= \{x \in X \mid y \in (Z \cap Y) \text{ and } \langle x, y \rangle \in R\}$ is the R -coimage class of Z , for $R : X \rightsquigarrow Y$ and a class Z .

5.7.14 Remark

- (1) In set-theoretical terms an n -ary relation R is defined as a subclass $\Gamma \subseteq D_1 \times \dots \times D_n$ of a cartesian product of classes. This version is easily translated into our default form by putting

$$R = \left[\begin{array}{c} D_1 \rightsquigarrow \dots \rightsquigarrow D_n \\ \langle x_1, \dots, x_n \rangle \rightsquigarrow \langle x_1, \dots, x_n \rangle \in \Gamma \end{array} \right]$$

- (2) Every n -ary relation $R : D_1 \rightsquigarrow \dots \rightsquigarrow D_n$ can be represented by an n -ary function

$$\chi_R := \left[\begin{array}{c} D_1 \times \dots \times D_n \longrightarrow \mathbb{B} \\ \langle x_1, \dots, x_n \rangle \mapsto \begin{cases} \mathbf{1} & \text{if } R(x_1, \dots, x_n) \\ \mathbf{0} & \text{else} \end{cases} \end{array} \right]$$

the characteristic function of R . On the other hand, every function $f : D \longrightarrow C$ can be represented by a binary relation

$$\left[\begin{array}{c} D \rightsquigarrow C \\ \langle x, y \rangle \rightsquigarrow f(x) = y \end{array} \right]$$

So somehow, one of the two concepts could be reduced to the other one. But in mathematical practice, it is common to understand them as distinct notions (see the remarks in 5.3.12).

- (3) However we introduce partial functions as a generalization of the function concept which will be convenient in certain circumstances (see definition 5.7.15). As usual, a partial function is a kind of function f from D into C , where $f(a)$ is not necessarily defined for every $a \in D$. Let us redefine a (total) function as a partial function $f : D \dashrightarrow C$, which has a unique $f(a) \in C$ for each $a \in D$.

5.7.15 Definition Partial functions

$f := \left[\begin{array}{c} D \dashrightarrow C \\ x \mapsto \theta \end{array} \right]$	is our default form of a <u>partial function definition</u> , saying that the identifier f is defined to be a partial function from a class D into a class C , such that every $x \in D$ is mapped onto $\theta \in C$. θ is a term of type C with no more free variables than x .
$f(a)$ or fa	f of a , is the unique value or image $b \in C$ for the given argument $a \in D$, given by $b = (x \mapsto \theta)(a)$.
$\text{dom}(f)$	$:= D$ is the <u>domain</u> of f
$\text{cod}(f)$	$:= C$ is the <u>codomain</u> of f
$\text{def}(f)$	$:= \left\{ a \in D \mid \begin{array}{l} f(a) \in C \\ \text{is well-defined} \end{array} \right\}$ is the <u>defined domain</u> of f

5.7.16 Remark

- (1) A typical example of a partial function is the division on rational numbers

$$\textcircled{1} / \textcircled{2} := \left[\begin{array}{c} \mathbb{Q} \times \mathbb{Q} \dashrightarrow \mathbb{Q} \\ \langle n, d \rangle \mapsto \text{the } a : \mathbb{Q} \text{ with } a \cdot d = n \end{array} \right]$$

with $\text{def}(/) = \mathbb{Q} \setminus \{0\}$.

- (2) The generalization of functions to partial functions is avoided in certain mathematical traditions, where “meaningless” or “undefined” expressions are tried to be excluded a priori. On the other hand, there are also strategies to deal with the undefined cases. A partial function $f : D \dashrightarrow C$ can easily be translated into a total function

$$f' := \left[\begin{array}{c} D \longrightarrow C \uplus \{\perp\} \\ x \mapsto \begin{cases} f(x) & \text{if } x \in \text{def}(f) \\ \perp & \text{else} \end{cases} \end{array} \right]$$

where $\perp \notin C$ is a any element of the codomain. Another approach (used e.g. in Unix and C), is the introduction of two channels for every output: a *standard output* and a *standard error*. In case of *undefined* arguments, a side effect is issued to the standard error.

5.7.17 Definition Operations on partial functions

f^{-1}	$:= \left[\begin{array}{c} C \dashrightarrow D \\ b \mapsto \text{the } a : D \text{ with } f(a) = b \end{array} \right]$ is the <u>inverse</u> of a partial function $f : D \dashrightarrow C$
$g \circ f$	$:= \left[\begin{array}{c} A \dashrightarrow C \\ x \mapsto g(f(x)) \end{array} \right]$ is the <u>composition</u> of two given partial functions $f : A \dashrightarrow B$ and $g : B \dashrightarrow C$.

5.8.5 Definition Number structures

$\mathfrak{Z} := \langle \mathbb{Z}, 0, 1, +, -, \cdot, \leq \rangle$ is the usual (linearly ordered) integer ring.

$\mathfrak{Q} := \langle \mathbb{Q}, 0, 1, +, -, \cdot, /, \leq \rangle$ is the usual (linearly ordered) field of rational numbers.

5.8.6 Definition

If $\mathfrak{S} = \langle \mathcal{K}, \mathcal{O} \rangle$ and $\mathfrak{S}' = \langle \mathcal{K}', \mathcal{O}' \rangle$ are two structures, then

$$\mathfrak{S} \uparrow \mathfrak{S}' := \langle \mathcal{K} \cup \mathcal{K}', \mathcal{O} \cup \mathcal{O}' \rangle$$

is the combination of \mathfrak{S} and \mathfrak{S}'

5.8.7 Definition

Given two ordinary structures in the normal form of 5.8.2

$$\mathfrak{S} = \langle C_1, \dots, C_k, c_1, \dots, c_l, f_1, \dots, f_m, R_1, \dots, R_n \rangle$$

$$\mathfrak{S}' = \langle C'_1, \dots, C'_k, c'_1, \dots, c'_l, f'_1, \dots, f'_m, R'_1, \dots, R'_n \rangle$$

(1) \mathfrak{S} and \mathfrak{S}' are called similar, iff

- (a) $k' = k$
- (b) $l' = l$ and for each $i \in \{1, \dots, l\}$ holds:

$$c_i : C_b \text{ implies } c'_i : C'_b$$
- (c) $m' = m$ and for each $i \in \{1, \dots, m\}$ holds:

$$\text{if } f_i : C_{a_1} \times \dots \times C_{a_j} \longrightarrow C_b$$

$$\text{then } f'_i : C'_{a_1} \times \dots \times C'_{a_j} \longrightarrow C'_b$$
- (d) $n' = n$ and for each $i \in \{1, \dots, n\}$ holds:

$$\text{if } R_i : C_{a_1} \rightsquigarrow \dots \rightsquigarrow C_{a_j}$$

$$\text{then } R'_i : C'_{a_1} \rightsquigarrow \dots \rightsquigarrow C'_{a_j}$$

(2) \mathfrak{S} is a substructure of \mathfrak{S}' , iff

- (a) \mathfrak{S} and \mathfrak{S}' are similar
- (b) $C_i \subseteq C'_i$ for each $i \in \{1, \dots, k\}$
- (c) $c_i = c'_i$ for each $i \in \{1, \dots, l\}$
- (d) $f_i(x) = f'_i(x)$, for all $i \in \{1, \dots, m\}$ and $x \in \text{dom}(f_i)$
- (e) $R_i(x)$ iff $R'_i(x)$, for all $i \in \{1, \dots, n\}$ and $x \in \text{dom}(R_i)$

5.8.8 Example

- (1) Being similar means for two structures, that they have an equivalent signature or type system.
- (2) \mathfrak{Z} and \mathfrak{Q} are not similar, \mathfrak{Q} has one more function than \mathfrak{Z} , but $\langle \mathbb{Z}, 0, 1, +, -, \cdot, \leq \rangle$ and $\langle \mathbb{Q}, 0, 1, +, -, \cdot, \leq \rangle$ are similar. In fact, the former is even a substructure of the latter.
- (3) $\langle \mathbb{B}, 0, 1, \wedge, \vee \rangle$ and $\langle \mathbb{Z}, 0, 1, \cdot, + \rangle$ are similar.

5.8.9 Definition

Given two similar structures, each with just one carrier, i.e. they have the general form

$$\mathfrak{S} = \langle C, c_1, \dots, c_l, f_1, \dots, f_m, R_1, \dots, R_n \rangle$$

$$\mathfrak{S}' = \langle C', c'_1, \dots, c'_l, f'_1, \dots, f'_m, R'_1, \dots, R'_n \rangle$$

A function

$$\mathfrak{h} : C \longrightarrow C'$$

is said to be

(a) a homomorphism from \mathfrak{S} into \mathfrak{S}' , written

$$\mathfrak{h} : \mathfrak{S} \longrightarrow \mathfrak{S}'$$

if it has the following properties:

- (i) $\mathfrak{h}(c_i) = c'_i$ for each $i \in \{1, \dots, l\}$
- (ii) $\mathfrak{h}(f_i(x_1, \dots, x_j)) = f'_i(\mathfrak{h}(x_1), \dots, \mathfrak{h}(x_j))$ for all $i \in \{1, \dots, m\}$ and $\langle x_1, \dots, x_j \rangle \in \text{dom}(f_i)$
- (iii) $R_i(x_1, \dots, x_j)$ implies $R'_i(\mathfrak{h}(x_1), \dots, \mathfrak{h}(x_j))$ for all $i \in \{1, \dots, n\}$ and $\langle x_1, \dots, x_j \rangle \in \text{dom}(R_i)$

(b) an isomorphism from \mathfrak{S} into \mathfrak{S}' , written

$$\mathfrak{h} : \mathfrak{S} \cong \mathfrak{S}'$$

if \mathfrak{h} is a bijective homomorphism.

\mathfrak{S} and \mathfrak{S}' are called isomorph, written

$$\mathfrak{S} \cong \mathfrak{S}'$$

if there is an isomorphism $\mathfrak{h} : \mathfrak{S} \cong \mathfrak{S}'$.

5.8.10 Example

- (1) If two structure are isomorph, then each formula holds in one structure iff it is true in the other one two. From a logical point of view, two isomorph structures are quasi-identical.
- (2) For example, for every record $\eta = [\eta_i | i \in I]$ (see 9.1.2), $\mathfrak{Proj}(\eta) \cong \mathfrak{P}(I)$, i.e. the record projection structure $\mathfrak{Proj}(\eta)$ (see 11.9.1) is isomorph to the power class algebra $\mathfrak{P}(I)$ (see 6.2.2).
- (3) In the proof of 19.5.10 we show that, for every schema $X = [X_i | i \in I]$ (see 9.3.1), $\mathfrak{Rel}(X) \cong \mathfrak{P}(\otimes X)$, i.e. the equi-schematic relation algebra over X (see 19.5.9) is isomorph to the power class algebra $\mathfrak{P}(\otimes X)$ of the cartesian product $\otimes X$ of X (see 12.1.2). And since $\mathfrak{P}(\otimes X)$ is a complete boolean algebra, $\mathfrak{Rel}(X)$ must be one, too.

Part III

Quasi-hierarchies

6 Examples of quasi-structures

6.1 Introduction

6.1.1 introduction

Order- and lattice-theoretical concepts belong to the basic toolkit of modern mathematics. Well-known key notions are:

- ♣ *poclasses* (or *posets*), i.e. *partially ordered classes*, i.e. classes with a transitive, reflexive and antisymmetric relation,
- ♣ *lattices*, i.e. poclasses with well-defined binary *meet* and *join* functions,
- ♣ *complete lattices*, i.e. lattices with *g.l.b* and *l.u.b* for every subclass,
- ♣ *boolean lattices* or *boolean algebras*, i.e. *distributive* lattices with *complement* functions.

Many structures of the main text can be subsumed under these headings as well. For example, the record class structure $\langle \mathbf{Rec}, \leq \rangle$ turns out to be a poclass and $\mathfrak{Rel}(X)$ is a complete boolean lattice, for every schema X .

However, there are other structures in the main text that almost fit in, but not exactly. These *quasi-structures* behave very much like poclasses, but instead they are based on *quasi-ordered classes*: they are still transitive and reflexive, but not necessarily antisymmetric.

Therefore, we generalize the key concepts and introduce *quasi-lattices*, *boolean quasi-lattices* and others. For example, the relation structure $\mathfrak{Pre}(X)$, the most important structure of our main text, comprises two *complete boolean quasi-lattices*. Of course, quasi-ordered classes are also well-known and well-investigated things in mathematics. But usually, these structures are turned into proper poclasses by either using their *quotient* structure (i.e. taking the equivalence classes as the new elements) or applying a *canonizer* (i.e. all equivalent representations are replaced by some unique *canonic normal form*).

But we cannot do all that with our structures without losing essential properties. In other words, if we want to integrate our relation algebra into the framework of lattices and boolean algebras, we need to generalize these concepts first.

6.1.2 overview of this chapter

Section 7 is a summarizing survey of this generalized order- and lattice theory. But that is just a dense and dry succession of definitions and statements. Therefore, we try to motivate the new concepts by a couple of important examples from mathematics and computer science here in section 6 first.

For these new key terms we add the references “(n.n.n)” which point to the paragraph that actually defines the according new notions.

6.2 Power algebras

6.2.1 Repetition

Recall 5.6.13, that every class C defines

$$\clubsuit \mathbf{P}(C) := \{A \mid A \subseteq C\}$$

$$\clubsuit \mathbf{Fin}(C) := \{A \mid A \subseteq C, A \text{ is finite}\}$$

6.2.2 Definition

Given a class C .

$$\mathfrak{P}(C) := \langle \mathbf{P}(C), \subseteq, \emptyset, \mathbf{1}, \cap, \cup, \bigcap, \bigcup, \complement \rangle$$

the power algebra or subclass algebra of C ,

where

$$(a) \mathbf{1} := C \text{ is the full class,}$$

$$(b) \cap \text{ is defined on the entire domain } \mathbf{P}(C) \text{ by putting } \bigcap \emptyset := C \text{ and}$$

$$(c) \complement \mathbf{1} := \mathbf{1} \setminus \mathbf{1} \text{ is the complement function.}$$

$$\mathfrak{Fin}(C) := \langle \mathbf{Fin}(C), \subseteq, \emptyset, \cap, \cup, \setminus \rangle$$

is the finite power algebra or finite subclass algebra of C .

6.2.3 Remark the power class algebra

$\langle \mathbf{P}(C), \subseteq \rangle$, i.e. the class of all its subclasses, ordered by the usual set inclusion, is the standard example of a *poclass* (7.2.2). In other words, \subseteq is a (*partial*) *order* (7.1.3) on $\mathbf{P}(C)$.

We generalize these notions: our basic concept is the *quasi-order* (7.1.3) (i.e. a *transitive* and *reflexive* binary endorelation), that may or may not be *antisymmetric* (also called *canonic*). A (*partial*) *order* is then a canonic quasi-relation.

The particular quasi-ordered class $\langle \mathbf{P}(C), \subseteq \rangle$ has also the following features:

- ♣ A *bottom element* (7.4.1) $\emptyset \in \mathbf{P}(C)$, which is less than any other member.
- ♣ A *top element* (7.4.1) $\mathbf{1} := C$, which is greater than any other member.
- ♣ A *meet function* (7.4.3) $\mathbf{1} \cap \mathbf{2} : \mathbf{P}(C) \times \mathbf{P}(C) \rightarrow \mathbf{P}(C)$ that returns the *greatest lower bound* for each pair of classes.
- ♣ A *join function* (7.4.3) $\mathbf{1} \cup \mathbf{2} : \mathbf{P}(C) \times \mathbf{P}(C) \rightarrow \mathbf{P}(C)$ that returns the *least upper bound* for each pair of classes.
- ♣ A *infimum function* (7.4.7) $\bigcap : \mathbf{P}(\mathbf{P}(C)) \rightarrow \mathbf{P}(C)$ that returns the *greatest lower bound* for every class of classes.
- ♣ A *supremum function* (7.4.7) $\bigcup : \mathbf{P}(\mathbf{P}(C)) \rightarrow \mathbf{P}(C)$ that returns the *least upper bound* for every class of classes.
- ♣ It has a *complement function* (7.5.2) $\complement : \mathbf{P}(C) \rightarrow \mathbf{P}(C)$ that returns the *complement* $\complement A$ of every class $A \in \mathbf{P}(C)$.

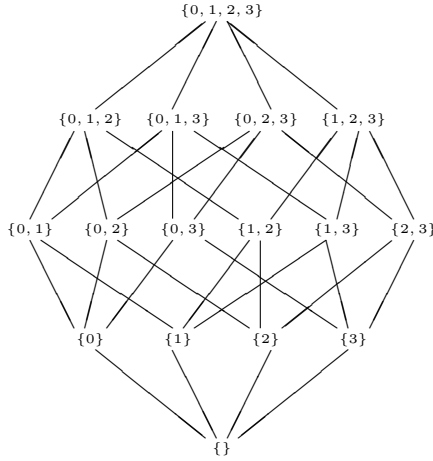
For *canonic* quasi-ordered classes, all these things are uniquely defined, if they exists at all. But when a quasi-ordered class is not canonic, there is usually more than one possible definition (as we will soon demonstrate).

A quasi-ordered class with all these features, which is also *distributive* (7.5.1), is a *complete quasi-boolean lattice*.

$\langle \mathbf{P}(C), \subseteq \rangle$ is a standard example of a complete quasi-boolean lattice. Canonic ones like this one are more often called *complete boolean lattices*.

6.2.4 Remark — the finite power class algebra

In case C is a finite class, every member of $\mathbf{P}(C)$ is a finite class, so $\mathbf{P}(C) = \mathbf{Fin}(C)$. If C is as small as $\{0, 1, 2, 3\}$, the structure $\langle \mathbf{Fin}(C), \subseteq \rangle$ can be represented by its *order diagram* or *Hasse diagram* (7.3.4):



But if C is an infinite class, $\mathbf{Fin}(C) \neq \mathbf{P}(C)$, and $\langle \mathbf{Fin}(C), \subseteq \rangle$ is not a complete boolean lattice anymore: There still is the bottom element \emptyset , and \cap and \cup are the meet and join function. But there is no top element anymore, thus no complement function. However, there still is

- a *relative complement function* (7.5.5) $\textcircled{1} \setminus \textcircled{2} : \mathbf{Fin}(C) \times \mathbf{Fin}(C) \rightarrow \mathbf{Fin}(C)$ that returns the *relative complement* $A \setminus B$ of B in A in the sense that always

$$(A \setminus B) \cap B = \emptyset \quad \text{and} \quad (A \setminus B) \cup B = A$$

Distributive lattices with these properties are sometimes called *generalized boolean algebras* (7.5.6) and $\langle \mathbf{Fin}(C), \subseteq \rangle$ is the typical standard example of such a structure. ²²

6.3 A generalized quasi-boolean algebra on tuples

6.3.1 Example

For every class C we defined the *tuple class* on C by

$$C^* := \{ \langle x_1, \dots, x_n \rangle \mid n \in \mathbb{N} \text{ and } x_1, \dots, x_n \in C \}$$

Such a tuple $\langle x_1, \dots, x_n \rangle$ can be used to represent the finite class $\{x_1, \dots, x_n\}$. All the class operations on $\mathbf{Fin}(C)$ can be defined for C^* in a similar fashion as follows. For each $e \in C$ and all $a = \langle a_1, \dots, a_n \rangle$ and $b = \langle b_1, \dots, b_m \rangle$ we put

- $e \in_* a$ iff $e \in \{a_1, \dots, a_n\}$

- $a \subseteq_* b$ iff $\{a_1, \dots, a_n\} \subseteq \{b_1, \dots, b_m\}$
- $a \equiv_* b$ iff $\{a_1, \dots, a_n\} = \{b_1, \dots, b_m\}$
- $a \setminus_* b$ is a , but every component in b is deleted
- $a \cup_* b := \langle a_1, \dots, a_n, b_1, \dots, b_m \rangle$
- $a \cap_* b := a \setminus_* (a \setminus_* b)$

For example, for $C = \mathbb{N}$ we obtain

$$\begin{aligned} 7 &\in_* \langle 1, 5, 2, 7, 4, 6, 32 \rangle \\ \langle 2, 4, 6 \rangle &\subseteq_* \langle 2, 2, 3, 3, 4, 4, 5, 5, 6, 6 \rangle \\ \langle 2, 4, 6 \rangle &\equiv_* \langle 2, 2, 4, 4, 6, 6 \rangle \equiv_* \langle 6, 4, 2 \rangle \\ \langle 2, 2, 3, 3, 4, 4, 5, 5, 6, 6 \rangle \setminus_* \langle 2, 4, 6, 8 \rangle &= \langle 3, 3, 5, 5 \rangle \\ \langle 2, 4, 4, 6 \rangle \cup_* \langle 2, 2, 3, 3, 4 \rangle &= \langle 2, 4, 4, 6, 2, 2, 3, 3, 4 \rangle \\ \langle 2, 4, 4, 6 \rangle \cap_* \langle 2, 2, 3, 3, 4 \rangle &= \langle 2, 4, 4, 6 \rangle \setminus_* \langle 6 \rangle = \langle 2, 4, 4 \rangle \\ \langle 2, 2, 3, 3, 4 \rangle \cap_* \langle 2, 4, 4, 6 \rangle &= \langle 2, 2, 3, 3, 4 \rangle \setminus_* \langle 3, 3 \rangle = \langle 2, 2, 4 \rangle \end{aligned}$$

The structure on C^* is very similar to the structure on $\mathbf{Fin}(C)$. More precisely, the codomain function

$$\mathbf{cod} := \left[\begin{array}{c} C^* \rightarrow \mathbf{Fin}(C) \\ \langle a_1, \dots, a_n \rangle \mapsto \{a_1, \dots, a_n\} \end{array} \right]$$

is a *quasi-isomorphism* in the sense that

$$\begin{aligned} e \in_* a &\text{ iff } a \in \mathbf{cod}(a) \\ a \subseteq_* b &\text{ iff } \mathbf{cod}(a) \subseteq \mathbf{cod}(b) \\ a \equiv_* b &\text{ iff } \mathbf{cod}(a) = \mathbf{cod}(b) \\ \mathbf{cod}(a \setminus_* b) &= \mathbf{cod}(a) \setminus \mathbf{cod}(b) \\ \mathbf{cod}(a \cap_* b) &= \mathbf{cod}(a) \cap \mathbf{cod}(b) \\ \mathbf{cod}(a \cup_* b) &= \mathbf{cod}(a) \cup \mathbf{cod}(b) \end{aligned}$$

for each $e \in C$ and all $a, b \in C^*$.

But unlike $\langle \mathbf{Fin}(C), \subseteq \rangle$, the structure $\langle C^*, \subseteq_* \rangle$ is not a *pclass* (transitive, reflexive, and antisymmetric), but only a *quasi-ordered class* (7.2.2) (i.e. transitive and reflexive), that is just not *canonic* or *antisymmetric* (7.1.2).

The general criterion (7.3.3) for canonicity tells us that the quasi-ordered class $\langle C^*, \subseteq_* \rangle$ is a pclass iff the equivalence relation \equiv_* is the identity relation. And this is obviously not the case here: $\langle 1, 2 \rangle \equiv_* \langle 2, 1 \rangle$, but $\langle 1, 2 \rangle \neq \langle 2, 1 \rangle$.

This non-canonicity makes that many notions uniquely defined on $\langle \mathbf{Fin}(C), \subseteq \rangle$ are defined on $\langle C^*, \subseteq_* \rangle$ only up to equivalence.

For example, two finite classes, say $\{1, 2, 3\}$ and $\{2, 3, 4\}$ do have a unique *greatest lower bound* (7.4.3) $\{2, 3\}$. So the *meet function* (7.4.3) on $\langle \mathbf{Fin}(C), \subseteq \rangle$, that takes any two finite classes and returns their greatest lower bound, is uniquely defined, namely as \cap . The two tuples $\langle 1, 2, 3 \rangle$ and $\langle 2, 3, 4 \rangle$ on the other hand have many *greatest lower bounds* $\langle 2, 3 \rangle$, $\langle 3, 2 \rangle$, $\langle 2, 2, 3 \rangle$ etc. which are all equivalent, i.e. $\langle 2, 3 \rangle \equiv_* \langle 3, 2 \rangle$ etc. Accordingly, a *meet function* does exist on $\langle C^*, \subseteq_* \rangle$ too, in our case \cap_* . But this meet function is not unique. We could have defined another meet function \cap'_* by $a \cap'_* b := b \setminus_* (b \setminus_* a)$. Then

²²According to Garrett Birkhoff: *Lattice Theory*, the definition of a *generalized boolean algebra* goes back to M.H. Stone: *The theory of representations for Boolean algebras*, *Trans. AMS* 40 (1936), p 35-52.

$$\langle 2, 4, 4, 6 \rangle \cap'_* \langle 2, 2, 3, 3, 4 \rangle = \langle 2, 2, 4 \rangle$$

$$\langle 2, 4, 4, 6 \rangle \cap_* \langle 2, 2, 3, 3, 4 \rangle = \langle 2, 4, 4 \rangle$$

The results of the two meet functions are always equivalent, here $\langle 2, 2, 4 \rangle \equiv_* \langle 2, 4, 4 \rangle$, but they are not equal in general.

If a quasi-ordered class like $\langle C^*, \subseteq_* \rangle$, also written together with the equivalence relation as $\langle C^*, \subseteq_*, \equiv_* \rangle$, has a bottom element (here: $\langle \rangle$), a meet and a join function (like \cap_* and \cup_*), and a relative complement function (here \setminus_*), we call it a *generalized boolean quasi-lattice* (7.5.4), and the concrete instance

$$\langle C^*, \subseteq_*, \equiv_*, \langle \rangle, \cap_*, \cup_*, \setminus_* \rangle$$

is a *generalized quasi-boolean algebra* (7.6.2).

6.4 The theory algebra of propositional formulas

6.4.1 propositional formulas

Given a class A of so-called **atoms** (usually a countable set of identifiers or symbols). The class

$$\mathbf{Form}(A)$$

of all **propositional formulas** on A is inductively defined to comprise the following expressions:

α	for every $\alpha \in A$	(atomic formulas)
\mathbf{f}		(false value)
\mathbf{t}		(true value)
$\neg\varphi$	for every $\varphi \in \mathbf{Form}(A)$	(negation)
$[\varphi_1 \wedge \dots \wedge \varphi_n]$	$\varphi_1, \dots, \varphi_n \in \mathbf{Form}(A)$	(conjunction)
$[\varphi_1 \vee \dots \vee \varphi_n]$	$\varphi_1, \dots, \varphi_n \in \mathbf{Form}(A)$	(disjunction)

For example, if $a, b, c, \dots \in A$, then

$$\begin{aligned} & [a \vee b \vee a \vee \neg c] \\ & [\neg a \wedge \neg [b \vee a] \wedge c \wedge \mathbf{t} \wedge \neg a] \\ & \quad \neg [\vee] \\ & [a \vee \mathbf{f}] \\ & [\mathbf{f} \vee \neg \mathbf{t} \vee [\wedge] \vee \mathbf{t}] \end{aligned}$$

are example formulas (on A).

Note that $\mathbf{Form}(A)$ is an infinite class, even if $A = \emptyset$.

6.4.2 boolean relations on propositional formulas

For two formulas φ and ψ we write

- ♣ $\varphi \Rightarrow \psi$ iff φ **entails** ψ , where this **entailment** or **subvalence** relation \Rightarrow on the class of formulas is defined as usual.
- ♣ $\varphi \Leftrightarrow \psi$ iff φ and ψ are **equivalent**, i.e. iff $\varphi \Rightarrow \psi$ and $\psi \Rightarrow \varphi$.

For example

$$[a \wedge \neg a] \Leftrightarrow \mathbf{f}$$

$$[a \wedge \neg b] \Rightarrow [a \vee \neg b]$$

$$[a \wedge \neg b] \Leftrightarrow \neg[\neg a \vee b]$$

$$[a \wedge b] \Rightarrow b$$

$$[a \wedge \mathbf{f}] \Rightarrow a$$

$$[a \wedge \mathbf{f}] \Leftrightarrow \neg \mathbf{t}$$

$$[\wedge] \Leftrightarrow \mathbf{t}$$

The relation \Rightarrow is *transitive* and *reflexive*, but it is not *antisymmetric* (7.1.2). For example $\mathbf{f} \Rightarrow \neg \mathbf{t}$ and $\neg \mathbf{t} \Rightarrow \mathbf{f}$, i.e. $\mathbf{f} \Leftrightarrow \neg \mathbf{t}$, but $\mathbf{f} \neq \neg \mathbf{t}$. So there are equivalent formulas which are not equal. In other words, $\langle \mathbf{Form}(A), \Rightarrow \rangle$ is a *quasi-ordered class*, but it is not *canonic*, i.e. not a *poclass* (7.2.2).

6.4.3 the standard quasi-boolean algebra on propositional formulas

For a given atom class A we define the

$$\mathfrak{Form}(A) := \langle \mathbf{Form}(A), \Rightarrow, \Leftrightarrow, \mathbf{f}, \mathbf{t}, \cap, \cup, \sqcup, \sqcap, \neg \rangle$$

the **standard quasi-boolean algebra** on $\mathbf{Form}(A)$, where

$$\begin{aligned} \textcircled{1} \cap \textcircled{2} & := \left[\begin{array}{l} \mathbf{Form}(A) \times \mathbf{Form}(A) \longrightarrow \mathbf{Form}(A) \\ \langle \varphi, \psi \rangle \mapsto [\varphi \wedge \psi] \end{array} \right] \\ \textcircled{1} \sqcup \textcircled{2} & := \left[\begin{array}{l} \mathbf{Form}(A) \times \mathbf{Form}(A) \longrightarrow \mathbf{Form}(A) \\ \langle \varphi, \psi \rangle \mapsto [\varphi \vee \psi] \end{array} \right] \\ \neg \textcircled{1} & := \left[\begin{array}{l} \mathbf{Form}(A) \longrightarrow \mathbf{Form}(A) \\ \varphi \mapsto \neg \varphi \end{array} \right] \end{aligned}$$

Such a *quasi-boolean algebra* is “quasi the same” as a proper *boolean algebra*, except that things are not “canonic”, i.e. identity is replaced by equivalence.

For example, *de Morgans law* (7.6.10) $\neg(x \cap y) = (\neg x) \sqcup (\neg y)$ only holds in a weaker, more general version $\neg(x \cap y) \Leftrightarrow (\neg x) \sqcup (\neg y)$.

Our standard quasi-boolean algebra on $\mathbf{Form}(A)$ is indeed a *quasi-boolean algebra* (7.6.2), because in particular

- ♣ \mathbf{f} is a *bottom element* (7.4.1) in the sense that it is a lower bound of every $\varphi \in \mathbf{Form}(A)$, i.e. $\mathbf{f} \Rightarrow \varphi$.
- ♣ \mathbf{t} is a *top element* (7.4.1), accordingly.
- ♣ \cap is a *meet function* (7.4.3) in the sense that $\varphi \cap \psi$ is a (*quasi*-)greatest lower bound for every two formulas φ and ψ . Again, note that next to $\varphi \cap \psi := [\varphi \wedge \psi]$ there are more greatest lower bounds, e.g. $[\psi \wedge \psi]$, $\neg[\neg \varphi \vee \neg \psi]$, etc. But they are all equivalent.
- ♣ \sqcup is a *join function* (7.4.3) that always returns a *least upper bound* accordingly.
- ♣ \neg is a *complement function*, because $\neg \varphi := \neg \varphi$ is indeed a *complement* of φ . But again, φ has more equivalent complements like $\neg[\varphi \wedge \mathbf{t}]$, $\neg \neg \varphi$, etc.

The fact that it is indeed possible to construct such a quasi-boolean algebra on top of the quasi-ordered class $\langle \mathbf{Form}(A), \Rightarrow \rangle$ is emphasized by saying that $\langle \mathbf{Form}(A), \Rightarrow \rangle$ is a *quasi-boolean lattice* (7.4.9).

For proper poclasses $\langle \mathcal{C}, \sqsubseteq \rangle$ there is no difference here: if they exist at all, the bottom and top element and the three functions are uniquely defined, the *boolean algebra* is uniquely determined, so the *boolean lattice* $\langle \mathcal{C}, \sqsubseteq \rangle$ “is” the *boolean algebra*. But if $\langle \mathcal{C}, \sqsubseteq \rangle$ is only a quasi-ordered class, then it either is or is not a *quasi-boolean lattice*. If it is, then there are usually more than one *boolean algebras* that can be defined on $\langle \mathcal{C}, \sqsubseteq \rangle$.

In our example, we could have defined another version of a quasi-boolean algebra, say

$$\langle \mathbf{Form}(A), \Rightarrow, \Leftrightarrow, [\vee], -[\vee], \sqcap', \sqcup', \neg' \rangle$$

where, for all $\varphi, \psi \in \mathbf{Form}(A)$,

$$\varphi \sqcap' \psi := [\psi \wedge \mathbf{t} \wedge \varphi \wedge \psi]$$

$$\varphi \sqcup' \psi := [\varphi \vee \psi \vee \mathbf{f}]$$

$$\neg' \varphi := -[\varphi \wedge \varphi]$$

6.4.4 Exercise

Is the quasi-boolean lattice $\langle \mathbf{Form}(A), \Rightarrow \rangle$ *complete* (7.4.9), i.e. does it have an *infimum* and *supremum function* (7.4.7)?

6.4.5 atomic relations on propositional formulas

The atom class function

$$\@ : \mathbf{Form}(A) \longrightarrow \mathbf{P}(A)$$

returns all the atoms that occur in a formula. For example

$$\@[(a \vee b \vee a \vee -c)] = \{a, b, c\}$$

$$\@[(-a \wedge -[b \vee a] \wedge c \wedge \mathbf{t} \wedge -a)] = \{a, b, c\}$$

$$\@[-[\vee]] = \emptyset$$

We define the subatomic and equiatomic relation on formulas by putting

$$\varphi \trianglelefteq \psi \quad \text{iff} \quad \@(\varphi) \subseteq \@(\psi)$$

$$\varphi \triangleq \psi \quad \text{iff} \quad \@(\varphi) = \@(\psi)$$

For example,

$$\mathbf{t} \trianglelefteq [a \vee -b]$$

$$[a \vee b \vee c \vee -a] \triangleq [a \wedge b \wedge c]$$

Similar to the subvalence relation, the subatomic relation is *transitive* and *reflexive*, but not *antisymmetric*. $\langle \mathbf{Form}(A), \trianglelefteq \rangle$ is a *quasi-ordered class*, but not a *canonic one*, i.e. not a *poclass*.

6.4.6 Exercise

Does $\langle \mathbf{Form}(A), \trianglelefteq \rangle$ have a *top* and *bottom elements*, a *meet* and *join function*, or even a *complement function*? Is it a *quasi-lattice* or even a *boolean one*?

6.4.7 atom expansion and reduction

Consider $\mathfrak{Form}(A) = \langle \mathbf{Form}(A), \Rightarrow, \Leftrightarrow, \mathbf{f}, \mathbf{t}, \sqcap, \sqcup, \neg \rangle$ again, for some arbitrary A , our standard quasi-boolean algebra on $\mathbf{Form}(A)$.

We observe that the operations \sqcap, \sqcup, \neg accumulate the atoms of their arguments in the sense that

$$\@(\varphi \sqcap \psi) = \@(\varphi) \cup \@(\psi)$$

$$\@(\varphi \sqcup \psi) = \@(\varphi) \cup \@(\psi)$$

$$\@(\neg \varphi) = \@(\varphi)$$

Operating on formulas increases the atom classes of the involved formulas. It is easy to define an operation that only expands the atoms of its argument and return an equivalent formula. On the other hand, we can also try to construct an operation that decreases the atom class. But this equivalent reduction is not that easy to generate. It often doesn't even exist at all.

An (atomic) expander is an operation

$$\textcircled{1} \parallel \textcircled{2} : \mathbf{Form}(A) \times A^* \longrightarrow \mathbf{Form}(A)$$

That takes a formula φ and atoms $\langle \alpha_1, \dots, \alpha_n \rangle \in A^*$ and returns a formula $\varphi' := \varphi \parallel \langle \alpha_1, \dots, \alpha_n \rangle$ such that

♣ $\@ \varphi' = \@ \varphi \cup \{ \alpha_1, \dots, \alpha_n \}$ and

♣ $\varphi' \Leftrightarrow \varphi$.

One instance of such an atom expander is obviously given by the definition

$$\varphi \parallel \langle \alpha_1, \dots, \alpha_n \rangle := [\varphi \vee [\mathbf{f} \wedge \alpha_1 \wedge \dots \wedge \alpha_n]]$$

We call the result the expansion of φ by $\alpha_1, \dots, \alpha_n$.

An equivalent reduction on the other hand should take a formula φ and atoms $\alpha_1, \dots, \alpha_n$ and return a formula φ' , called an equivalent reduction of φ onto $\alpha_1, \dots, \alpha_n$, such that

♣ $\@ \varphi' = \{ \alpha_1, \dots, \alpha_n \}$ and

♣ $\varphi' \Leftrightarrow \varphi$

For example, if we put $\varphi := [b \vee c \vee [a \wedge -a]]$ and $\varphi' := [b \vee c]$ then φ' is indeed an equivalent reduction of φ onto b, c .

But if we put $\varphi := [b \vee c \vee -a]$, no equivalent reduction of φ onto b, c does exist.

6.4.8 infimum and supremum reduction

Thus an *equivalent reduction* is not a well-defined notion in general. However, there are two other reduction concepts:

♣ An infimum reductor is a function

$$\textcircled{1} \uparrow \textcircled{2} : \mathbf{Form}(A) \times A^* \longrightarrow \mathbf{Form}(A)$$

such that, for all formulas φ and atoms α_i

♣ $\@(\varphi \uparrow \langle \alpha_1, \dots, \alpha_n \rangle) = \{ \alpha_1, \dots, \alpha_n \}$

♣ $\varphi \uparrow \langle \alpha_1, \dots, \alpha_n \rangle$ is a greatest lower bound of φ in $\mathbf{Form}(\{ \alpha_1, \dots, \alpha_n \})$

♣ Similarly, a supremum reductor is a function

$$\textcircled{1} \downarrow \textcircled{2} : \mathbf{Form}(A) \times A^* \longrightarrow \mathbf{Form}(A)$$

such that

♣ $\@(\varphi \downarrow \langle \alpha_1, \dots, \alpha_n \rangle) = \{ \alpha_1, \dots, \alpha_n \}$

♣ $\varphi \downarrow \langle \alpha_1, \dots, \alpha_n \rangle$ is a least upper bound of φ in $\mathbf{Form}(\{ \alpha_1, \dots, \alpha_n \})$

Different to the implementation of an expander, a method to generate these two reductions is quite a complicated process and we will not discuss the matter here.²³

6.5 Quasi-ordered classes on numbers

6.5.1 linearly ordered number classes

Recall, that $\mathbb{N} := \{0, 1, 2, \dots\}$ is the *natural number class*, $\mathbb{Z} := \{\dots, -2, -1, 0, 1, 2, \dots\}$, the class of *integers*, \mathbb{Q} the *rational number class*, and $\leq, <, +, -, \cdot, /$ are the usual arithmetic operations.

\leq is a *linear order* (7.1.3) and $<$ is a *strict linear order* (7.1.3) on \mathbb{N} , \mathbb{Z} , and \mathbb{Q} . In other words $\langle \mathbb{N}, \leq \rangle$, $\langle \mathbb{Z}, \leq \rangle$, and $\langle \mathbb{Q}, \leq \rangle$ are examples of *linearly ordered classes* or *chains* (7.2.2).

6.5.2 number structures as lattices

A linearly ordered class is a special case of a (*partially*) *ordered class* or *poclass* (7.2.2). Poclasses can have certain properties (7.4), such as being a *lattice* (7.4.9) or even a *boolean lattice* (7.5.4). Let us find out, in how far these properties hold for the given number structures.

$\langle \mathbb{N}, \leq \rangle$ has a *bottom element* (7.4.1), namely 0, because $0 \leq n$, for all $n \in \mathbb{N}$. But it has no *top element*, because for each natural number there is an even bigger one.

$\langle \mathbb{Z}, \leq \rangle$ has neither a bottom nor a top element. But we can add two new elements ∞ and $-\infty$, put

$$\mathbb{Z}^\infty := \mathbb{Z} \cup \{-\infty, \infty\}$$

and redefine the arithmetic operations on the extended class \mathbb{Z}^∞ as usual. For example, $-\infty \leq n$, $\infty + 9 = \infty$, $-\infty \cdot 7 = -\infty$, $5/\infty = 0$. That way $\langle \mathbb{Z}^\infty, \leq \rangle$ has a bottom and a top element, what makes it a so-called *bounded* (7.4.9) chain.

A *lower bound* for two given natural numbers n and m is any number k with $k \leq n$ and $k \leq m$. For example, 1, 2, and 3 are lower bounds of $n = 23$ and $m = 3$. In this case, 3 is the *greatest lower bound*. Now, any binary function that always returns the greatest lower bound for each pair of arguments is called a *meet function* (7.4.3) (or *conjunctive*). In $\langle \mathbb{N}, \leq \rangle$, each number pair has a unique greatest lower bound, so there is a unique meet function, namely **min** : $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$, the usual minimum function.

Similarly, the binary function **max** always returns the *least upper bound* of two given numbers and such a function is a *join function* (7.4.3) on $\langle \mathbb{N}, \leq \rangle$.

Now a poclass with a meet as well as a join function, such as $\langle \mathbb{N}, \leq, \mathbf{min}, \mathbf{max} \rangle$, is called a *lattice* (7.4.9). Obviously, $\langle \mathbb{Z}, \leq, \mathbf{min}, \mathbf{max} \rangle$ and $\langle \mathbb{Z}^\infty, \leq, \mathbf{min}, \mathbf{max} \rangle$ are also lattices. $\langle \mathbb{Z}^\infty, \leq, -\infty, \infty, \mathbf{min}, \mathbf{max} \rangle$ is even a *bounded lattice* (7.4.9).

The meet function takes two arguments, its generalization for any — even an infinite — class of arguments is called an *infimum function* (7.4.7) (or *big conjunctive*). $\langle \mathbb{N}, \leq \rangle$ has an infimum function, because for every $N \subseteq \mathbb{N}$, there is a unique

greatest lower bound of N , namely the minimum of N . But $\langle \mathbb{Z}, \leq \rangle$ doesn't have an infimum function. For example, the infinite class $\{-2, -4, -6, -8, \dots\}$ does not have a (greatest) lower bound in \mathbb{Z} . And for similar reasons, $\langle \mathbb{N}, \leq \rangle$ doesn't have a *supremum function* which would return the greatest upper bound of every given subclass of \mathbb{N} .

$\langle \mathbb{Z}^\infty, \leq \rangle$ actually has both, an infimum and a supremum function. Therefore, $\langle \mathbb{Z}^\infty, \leq \rangle$ (together with these implicitly defined functions) is an example of a *complete lattice* (??).

6.5.3 Exercise

Which of the three lattices (on \mathbb{N} , \mathbb{Z} , or \mathbb{Z}^∞) is a *distributed* (7.5.1), *complemented* (7.5.2), or even *boolean* (7.5.4) lattice?

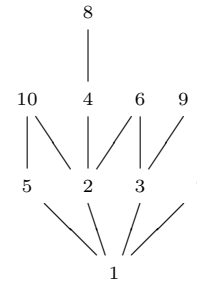
6.5.4 integers and the division relation

An often mentioned example of a *poclass* (7.2.2) is $\langle N, | \rangle$, where $N \subseteq \mathbb{N}$ and the binary relation is defined by

$$\textcircled{1} \mid \textcircled{2} := \left[\begin{array}{c} \mathbb{N} \leftrightarrow \mathbb{N} \\ \langle m, n \rangle \rightsquigarrow (m \cdot d = n \text{ for some } d > 0) \end{array} \right]$$

where $m \mid n$ reads “ m divides n ”. For example, $7 \mid 63$, because $7 \cdot 9 = 63$.

If say $N = \{1, 2, \dots, 10\}$, the poclass $\langle N, | \rangle$ is represented by the following *order* or *Hasse diagram* (7.3.4)



6.5.5 Exercise

Does $\langle \mathbb{N}, | \rangle$ have a *bottom element* (7.4.1), a *top element* (7.4.1), a *meet function* (7.4.3), a *join function* (7.4.3)? Is it a *lattice* (7.4.9)?

6.6 The quasi-field of fractions

6.6.1 fractions

We define the clas of *fractions* as

$$\mathbb{F} := \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$$

and we write

$$\frac{n}{d} \quad \text{for} \quad \langle n, d \rangle \in \mathbb{F}$$

²³ More material can be found on www.bucephalus.org, along with software system that compute the results of all these operations on propositional formulas.

i.e. for the fraction with *nominator* n and *denominator* d . Furthermore, we define a couple of relations and arithmetic operations on \mathbb{F} :

$$\begin{aligned} \textcircled{1} \lesssim \textcircled{2} &:= \left[\begin{array}{c} \mathbb{F} \rightsquigarrow \mathbb{F} \\ \langle n/d, n'/d' \rangle \rightsquigarrow n \cdot d' \leq n' \cdot d \end{array} \right] \\ \textcircled{1} \simeq \textcircled{2} &:= \left[\begin{array}{c} \mathbb{F} \rightsquigarrow \mathbb{F} \\ \langle n/d, n'/d' \rangle \rightsquigarrow n \cdot d' = n' \cdot d \end{array} \right] \\ \mathbf{0} &:= 0/1 \\ \mathbf{1} &:= 1/1 \\ \textcircled{1} + \textcircled{2} &:= \left[\begin{array}{c} \mathbb{F} \times \mathbb{F} \longrightarrow \mathbb{F} \\ \langle n/d, n'/d' \rangle \mapsto n \cdot d' + n' \cdot d / d \cdot d' \end{array} \right] \\ \textcircled{1} - \textcircled{2} &:= \left[\begin{array}{c} \mathbb{F} \times \mathbb{F} \longrightarrow \mathbb{F} \\ \langle n/d, n'/d' \rangle \mapsto n \cdot d' - n' \cdot d / d \cdot d' \end{array} \right] \\ \textcircled{1} \cdot \textcircled{2} &:= \left[\begin{array}{c} \mathbb{F} \times \mathbb{F} \longrightarrow \mathbb{F} \\ \langle n/d, n'/d' \rangle \mapsto n \cdot n' / d \cdot d' \end{array} \right] \\ \textcircled{1} \div \textcircled{2} &:= \left[\begin{array}{c} \mathbb{F} \times \mathbb{F} \dashrightarrow \mathbb{F} \\ \langle n/d, n'/d' \rangle \mapsto \begin{cases} n \cdot d' / d \cdot n' & \text{if } d \neq 0 \\ \text{undefined} & \text{if } d = 0 \end{cases} \end{array} \right] \end{aligned}$$

The overall structure

$$\langle \mathbb{F}, \lesssim, \simeq, \mathbf{0}, \mathbf{1}, +, -, \cdot, \div \rangle$$

is the quasi-ordered field of fractions.

6.6.2 the quasi-field

The title *quasi-field* indicates that this structure behaves like a proper field, when the identity $=$ in the defining axioms is replaced by the equivalence relation. Our more general term for these kind of algebras is *quasi-algebra*.

\lesssim is a *quasi-linear order* (7.1.3) (transitive, reflexive, and total), but not a proper *linear order* (7.1.3), because it is not

antisymmetric (7.1.2). In other words, the *equivalence relation* (7.3.1) \simeq of \lesssim is not the equality, there are cases of equivalent, but still different fractions, such as $5/2$ and $10/4$.

Accordingly, we could have defined the constants $\mathbf{0}$, $\mathbf{1}$ and the other operations on \mathbb{F} differently, and still obtain a quasi-ordered field. For example, if we would have defined

$$\mathbf{0} := 0/57 \quad \mathbf{1} := 13/13 \quad n/d \cdot n'/d' := 7 \cdot n \cdot n' / 7 \cdot d \cdot d' \quad \dots$$

then $\mathbf{1}$ still is the neutral element of the multiplication etc.

6.6.3 the quasi-lattice

The quasi-linearly ordered class $\langle \mathbb{F}, \lesssim \rangle$ is a *quasi-lattice*. Again (7.4.5), the *minimum function* (7.4.4) \mathbf{min} is one example of a possible *meet function*, and \mathbf{max} is a *join function* for the construction of such a quasi-lattice.

6.6.4 Exercise

Is this lattice on fractions *bounded* or even *complete* (hint: *real numbers*)?

6.6.5 rational numbers

As usual, the *rational numbers* can be introduced as equivalence classes of fractions together with a generalization of the arithmetic operations on \mathbb{Z} . We put

$$\begin{aligned} \frac{n}{d} &:= \{ n'/d' \mid n'/d' \simeq n/d \} \\ \mathbb{Q} &:= \{ \frac{n}{d} \mid n \in \mathbb{Z}, d \in (\mathbb{Z} \setminus \{0\}) \} \\ \mathbf{0} &:= \frac{0}{1} \\ \mathbf{1} &:= \frac{1}{1} \\ \textcircled{1} + \textcircled{2} &:= \left[\begin{array}{c} \mathbb{F} \times \mathbb{F} \longrightarrow \mathbb{F} \\ \langle \frac{n}{d}, \frac{n'}{d'} \rangle \mapsto \frac{n \cdot d' + n' \cdot d}{d \cdot d'} \end{array} \right] \\ &\vdots \\ &\vdots \end{aligned}$$

The overall structure on \mathbb{Q} is the linearly ordered field of rational numbers.

7 Quasi-structures and quasi-algebras

7.1 Special binary endorelations

7.1.1 Definition

A (binary) (endo-)relation on C is a relation $\rho : C \rightsquigarrow C$. We agree to write these kind of relations in infix notation by default, i.e. we write

$$\textcircled{1} \rho \textcircled{2} \quad \text{rather than} \quad \rho(\textcircled{1}, \textcircled{2})$$

A structured class $\langle C, \rho \rangle$ is a class C with a relation on C (or more general, on a superclass of C).

7.1.2 Definition

A binary endorelation $\rho : Q \rightsquigarrow Q$ on a given class Q is said to be

- (1) transitive, if $x\rho y$ and $y\rho z$ implies $x\rho z$, for all $x, y, z \in Q$
- (2) reflexive, if $x\rho x$, for all $x \in Q$
- (3) irreflexive, if $x\rho x$ doesn't hold for any $x \in Q$
- (4) symmetric, if $x\rho y$ implies $y\rho x$, for all $x, y \in Q$
- (5) asymmetric, if $x\rho y$ implies, that $y\rho x$ doesn't hold, for all $x \in Q$
- (6) antisymmetric, if $x\rho y$ and $y\rho x$ implies $x = y$, for all $x, y \in Q$
- (7) total, if $x\rho y$ or $y\rho x$, for all $x, y \in Q$
- (8) connex, if either $x\rho y$, or $y\rho x$, or $x = y$, for all $x, y \in Q$

7.1.3 Definition

A binary endorelation ρ on Q , i.e. $\rho : Q \rightsquigarrow Q$ is called

- (1) an equivalence relation (on Q), if ρ is transitive, reflexive, and symmetric
- (2) an equality or identity relation (on Q), if ρ is transitive, reflexive, symmetric, and antisymmetric
- (3) a quasi-order (on Q), if ρ is transitive and reflexive
- (4) a (partial) order (on Q), if ρ is transitive, reflexive, and antisymmetric
- (5) a quasi-linear order (or linear quasi-order) (on Q), if ρ is transitive, reflexive, and total
- (6) a linear order (on Q), if ρ is transitive, reflexive, total, and antisymmetric

7.2 Quasi-Hierarchies

7.2.1 Definition

If $\rho : D \rightsquigarrow D$ is a binary endorelation on a class D and $C \subseteq D$, we define

$$\rho|_C := \left[\begin{array}{c} C \rightsquigarrow C \\ \langle c_1, c_2 \rangle \rightsquigarrow c_1 \rho c_2 \end{array} \right]$$

the restriction of ρ onto C .

7.2.2 Definition

Let $\rho : D \rightsquigarrow D$ and $C \subseteq D$. We say that

- (1) $\langle C, \rho \rangle$ is a quasi-hierarchy, if $\rho|_C$ is a transitive relation on C
- (2) $\langle C, \rho \rangle$ is a hierarchy, if $\rho|_C$ is a transitive and antisymmetric relation on C
- (3) $\langle C, \rho \rangle$ is a quasi-class or a partition, if $\rho|_C$ is an equivalence relation on C
- (4) $\langle C, \rho \rangle$ is a canonic class or a anti-chain, if $\rho|_C$ is a equality relation on C
- (5) $\langle C, \rho \rangle$ is a quasi-ordered class, if $\rho|_C$ is a quasi-order on C
- (6) $\langle C, \rho \rangle$ is a canonic quasi-ordered class or (partially) ordered class or poclass, if $\rho|_C$ is a (partial) order on C
- (7) $\langle C, \rho \rangle$ is a quasi-linearly ordered class or quasi-chain, if $\rho|_C$ is quasi-linear order on C
- (8) $\langle C, \rho \rangle$ is a linearly ordered class or chain, if $\rho|_C$ is a linear order on C

7.2.3 Lemma

Let $\rho : D \rightsquigarrow D$ and $C' \subseteq C \subseteq D$.

- (1) If $\langle C, \rho \rangle$ is a quasi-hierarchy, then $\langle C', \rho \rangle$ is a quasi-hierarchy.
- (2) If $\langle C, \rho \rangle$ is a hierarchy, then $\langle C', \rho \rangle$ is a hierarchy.
- (3) If $\langle C, \rho \rangle$ is a quasi-class, then $\langle C', \rho \rangle$ is a quasi-class.
- (4) If $\langle C, \rho \rangle$ is a canonic class, then $\langle C', \rho \rangle$ is a canonical class.
- (5) If $\langle C, \rho \rangle$ is a quasi-ordered class, then $\langle C', \rho \rangle$ is a quasi-ordered class.
- (6) If $\langle C, \rho \rangle$ is a poclass, then $\langle C', \rho \rangle$ is a poclass.
- (7) If $\langle C, \rho \rangle$ is a quasi-chain, then $\langle C', \rho \rangle$ is a quasi-chain.
- (8) If $\langle C, \rho \rangle$ is a chain, then $\langle C', \rho \rangle$ is a chain.

7.3 Quasi-ordered classes and poclasses

7.3.1 Definition derived equivalence

If \sqsubseteq is a quasi-order on a given class Q , then

$$\left[\begin{array}{c} Q \rightsquigarrow Q \\ \langle x, y \rangle \rightsquigarrow x \sqsubseteq y \text{ and } y \sqsubseteq x \end{array} \right]$$

is its equivalence relation.

7.3.2 Remark notation

We often write

$$\langle C, \sqsubseteq, \equiv \rangle \quad \text{instead of} \quad \langle C, \sqsubseteq \rangle$$

to denote a quasi-ordered class, where \equiv is the equivalence relation of \sqsubseteq .

On the other hand, the graphic symbol of the derived equivalence relation is often designed similar to the quasi-order symbol, and then we often use the pair notation again. The following table lists some common examples.

the relation	its equivalence relation
\sqsubseteq	\equiv
\succcurlyeq	\approx
\preceq	$=$
\sqsubset	\equiv
\Rightarrow	\Leftrightarrow
\top	\neq

7.3.3 Lemma

If \sqsubseteq is a quasi-order and \equiv its equivalence relation, then

- (1) \equiv is indeed a well-defined equivalence relation (see 7.1.3)
- (2) \sqsubseteq is canonic iff \equiv is the identity relation.

In other words, a quasi-ordered class is canonic (i.e. a poclass) iff the equivalence relation is the equality.

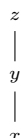
7.3.4 Remark Order or Hasse diagrams

Finite poclasses can be represented by so-called order diagrams or Hasse diagrams:

The fact, that $x \rho y$ holds for two elements x and y is expressed by the feature that x stands below y in the diagram and there is an upwards path from x to y .



The transitivity reduces the number of lines that have to be drawn. For example, the diagram



says that $x \rho y$ and $y \rho z$, and also that $x \rho z$.

The reflexivity, i.e. that $x \rho x$ for all x , is not explicitly represented in the diagram. It is just always assumed.

For examples see: the order diagram of a finite power class algebra in 6.2.4 or the diagram of integers with the division relation in 6.5.4.

7.3.5 Remark Quasi-order diagrams

For finite quasi-ordered classes we generalize the idea of order diagrams to quasi-order diagrams: To limit the number of edges in the diagram we first put all equivalent elements in a circle and then we draw the order diagram of these equivalence classes. In other words, the quasi-order diagram is the diagram of the quotient structure (see 7.7.3).

7.4 Quasi-lattices

7.4.1 Definition

Let $\langle Q, \sqsubseteq \rangle$ be a quasi-ordered class.

- (1) $\perp \in Q$ is a bottom or zero or least element in $\langle Q, \sqsubseteq \rangle$, if $\perp \sqsubseteq x$, for all $x \in Q$.
- (2) $\top \in Q$ is a top or unit or greatest element in $\langle Q, \sqsubseteq \rangle$, if $x \sqsubseteq \top$, for all $x \in Q$.

7.4.2 Lemma quasi-uniqueness of bottom and top

Let $\langle Q, \sqsubseteq \rangle$ be a quasi-ordered class.

- (1) If \perp_1 and \perp_2 are two bottom elements in $\langle Q, \sqsubseteq \rangle$, then $\perp_1 \equiv \perp_2$.
- (2) If \top_1 and \top_2 are two top elements in $\langle Q, \sqsubseteq \rangle$, then $\top_1 \equiv \top_2$.

7.4.3 Definition

Let $\langle Q, \sqsubseteq \rangle$ be a quasi-ordered class.

- (1) A meet function or conjunction on $\langle Q, \sqsubseteq \rangle$ is a function $\textcircled{1} \sqcap \textcircled{2} : Q \times Q \rightarrow Q$ that always returns a greatest lower bound in the sense that for all $x, y \in Q$ holds:
 - (a) $x \sqcap y \sqsubseteq x$ and $x \sqcap y \sqsubseteq y$
 - (b) If $z \in Q$ with $z \sqsubseteq x$ and $z \sqsubseteq y$ and $x \sqcap y \sqsubseteq z$, then $z \equiv x \sqcap y$.
- (2) A join function or disjunction on $\langle Q, \sqsubseteq \rangle$ is a function $\textcircled{1} \sqcup \textcircled{2} : Q \times Q \rightarrow Q$ that always returns a least upper bound in the sense that for all $x, y \in Q$ holds:
 - (a) $x \sqsubseteq x \sqcup y$ and $y \sqsubseteq x \sqcup y$
 - (b) If $z \in Q$ with $x \sqsubseteq z$ and $x \sqsubseteq z$ and $z \sqsubseteq x \sqcup y$, then $z \equiv x \sqcup y$.

7.4.4 Definition

If $\langle Q, \lesssim \rangle$ is a quasi-linearly ordered class, then

$$\mathbf{min} := \begin{matrix} Q \times Q \\ \langle x, y \rangle \mapsto \begin{cases} x & \text{if } x \simeq y \\ y & \text{else} \end{cases} \end{matrix}$$

is its minimum function

$$\mathbf{max} := \begin{matrix} Q \times Q \\ \langle x, y \rangle \mapsto \begin{cases} y & \text{if } x \simeq y \\ x & \text{else} \end{cases} \end{matrix}$$

is its maximum function

7.4.5 Lemma

Every quasi-linearly ordered class $\langle Q, \lesssim \rangle$ is a quasi-lattice. A meet and join function is given by **min** and **max**, respectively.

7.4.6 Lemma —quasi-uniqueness of meet and join—

Let $\langle Q, \sqsubseteq \rangle$ be a quasi-ordered class.

- (1) If \sqcap_1 and \sqcap_2 are two meet functions in $\langle Q, \sqsubseteq \rangle$, then $x \sqcap_1 y \equiv x \sqcap_2 y$, for all $x, y \in Q$.
- (2) If \sqcup_1 and \sqcup_2 are two join functions in $\langle Q, \sqsubseteq \rangle$, then $x \sqcup_1 y \equiv x \sqcup_2 y$, for all $x, y \in Q$.

7.4.7 Definition

Let $\langle Q, \sqsubseteq \rangle$ be a quasi-ordered class.

- (1) A infimum function or big conjunctor on $\langle Q, \sqsubseteq \rangle$ is a function

$$\prod \textcircled{1} : \mathbf{P}(Q) \longrightarrow Q$$

that always returns a greatest lower bound in the sense that for all $S \subseteq Q$ holds:

- (a) $\prod S \subseteq y$, for all $y \in S$
- (b) If $z \in Q$ with $z \subseteq y$ for all $y \in S$ and $\prod S \subseteq z$, then $z \equiv \prod S$.

- (2) A supremum function or big disjuncter on $\langle Q, \sqsubseteq \rangle$ is a function

$$\coprod \textcircled{1} : \mathbf{P}(Q) \longrightarrow Q$$

that always returns a least upper bound in the sense that for all $S \subseteq Q$ holds:

- (a) $y \subseteq \coprod S$, for all $y \in S$
- (b) If $z \in Q$ with $y \subseteq z$ for all $y \in S$ and $z \subseteq \coprod S$, then $z \equiv \coprod S$.

7.4.8 Lemma —quasi-uniqueness of infimum and supremum—

Let $\langle Q, \sqsubseteq \rangle$ be a quasi-ordered class.

- (1) If \prod_1 and \prod_2 are two infimum functions in $\langle Q, \sqsubseteq \rangle$, then $\prod_1 S \equiv \prod_2 S$, for all $S \subseteq Q$.
- (2) If \coprod_1 and \coprod_2 are two supremum functions in $\langle Q, \sqsubseteq \rangle$, then $\coprod_1 S \equiv \coprod_2 S$, for all $S \subseteq Q$.

7.4.9 Definition

A quasi-ordered class $\langle Q, \sqsubseteq \rangle$ is called

- (1) bounded, if both a bottom and top element exist
- (2) a quasi-lattice, if a meet and join function exist
- (3) a lattice, if it is a canonic quasi-lattice, in other words a quasi-lattice on a pclass
- (4) a complete quasi-lattice, if an infimum and supremum function exist
- (5) a complete lattice, accordingly, i.e. if it is a complete canonic quasi-lattice

7.4.10 Lemma —canonicity theorem—

On a given quasi-ordered class, bottom and top elements, small and big con- and disjunctors may or may not exist. But in case of a pclass, each of these items is unique if it exists at all.

7.5 Distributivity and complementation

7.5.1 Definition

Let $\langle Q, \sqsubseteq \rangle$ be a quasi-lattice. Let \sqcap and \sqcup be a meet and join function on it. We say that $\langle Q, \sqsubseteq \rangle$ is distributive, if

$$(x \sqcap y) \sqcup z \equiv (x \sqcup z) \sqcap (y \sqcup z)$$

$$(x \sqcup y) \sqcap z \equiv (x \sqcap z) \sqcup (y \sqcap z)$$

for all $x, y, z \in Q$.

7.5.2 Definition —complement—

Let $\langle Q, \sqsubseteq \rangle$ be a bounded quasi-lattice. Let \perp, \top, \sqcap , and \sqcup be a bottom, top, meet, and join on $\langle Q, \sqsubseteq \rangle$, respectively. A negator or complement function on $\langle Q, \sqsubseteq \rangle$ is a function

$$\neg \textcircled{1} : Q \longrightarrow Q$$

such that for all $x \in Q$ holds:

$$\neg x \sqcap x \equiv \perp \quad \text{and} \quad \neg x \sqcup x \equiv \top$$

$\langle Q, \sqsubseteq \rangle$ itself is a complemented quasi-lattice, if such a complement function exists.

7.5.3 Lemma —quasi-uniqueness of complementations—

Let $\langle Q, \sqsubseteq \rangle$ be a complemented quasi-lattice and let \neg_1 and \neg_2 be two complement functions on it. If $\langle Q, \sqsubseteq \rangle$ is distributive, then $\neg_1 x \equiv \neg_2 x$, for all $x \in Q$.

7.5.4 Definition —

A boolean quasi-lattice (or quasi-boolean lattice) is a complemented distributive quasi-lattice.
A boolean lattice is a canonic boolean quasi-lattice, i.e. a complemented distributive lattice.

7.5.5 Definition —relative complement—

Let $\langle Q, \sqsubseteq \rangle$ be a quasi-lattice with bottom element and let \perp, \sqcap, \sqcup be a bottom, meet and join, respectively. A relative complement function on $\langle Q, \sqsubseteq \rangle$ is a function

$$\textcircled{1} - \textcircled{2} : Q \times Q \rightarrow Q$$

such that for all $x, y \in Q$ holds:

$$(x - y) \sqcap y \equiv \perp \quad \text{and} \quad (x - y) \sqcup y \equiv x$$

$x - y$ is the relative complement of y in x , or simply x without or minus y .

7.5.6 Definition —generalized boolean quasi-lattice—

A generalized boolean quasi-lattice is a distributive quasi-lattice that has a bottom element and a relative complement function.

A generalized boolean lattice is a canonic generalized boolean quasi-lattice, i.e. a distributive lattice with a bottom element and a relative complement function.

7.5.7 Lemma —

(1) If a bounded quasi-lattice has a relative complement function $\textcircled{1} - \textcircled{2}$, it also has a complement function, namely $\neg \textcircled{1} := \textcircled{1} - \textcircled{1}$.

(2) A generalized boolean quasi-lattice is a boolean quasi-lattice iff it has a top element.

7.5.8 Remark —

- (1) Note, that due to 7.4.2 and 7.4.6, the concrete choice of \perp, \top, \sqcap , and \sqcup is not relevant in the definitions 7.5.1, 7.5.2 and 7.5.5, respectively.
- (2) There are non-distributive complemented quasi-lattices, where two complement functions do not always produce equivalent results. However, these structures are not relevant for us.
- (3) Unfortunately, the definition of “generalized boolean” notion is less common than the very established and standardized “boolean” structures and “lattices”. A simplified slogan to memorize the concept is the definition of *generalized boolean quasi-lattices* as “boolean quasi-lattices with or without top element”.

7.6 Quasi-algebras

7.6.1 Remark —Introduction—

A quasi-ordered class $\langle Q, \sqsubseteq \rangle$ together with a couple of junctions (e.g. \top, \sqcap, \sqcup etc. as defined earlier) are called quasi-algebras. “Quasi” is a generalization of the usual “non-quasi” or “canonical” notion in the sense that the equivalence relation \equiv is a generalization of the identity relation $=$.

If $\langle Q, \sqsubseteq \rangle$ is a poclass, then a junctor like a meet function \sqcap is uniquely defined if it exists at all. So, if $\langle Q, \sqsubseteq \rangle$ is say a lattice, then the algebra $\langle Q, \sqsubseteq, \sqcap, \sqcup \rangle$ is given by $\langle Q, \sqsubseteq \rangle$ only. In other words, a “lattice” is a “lattice algebra”. That makes the term “lattice algebra” a pleonasm and superfluous.

However, if $\langle Q, \sqsubseteq \rangle$ is a quasi-lattice, then there may be more than one way to define a “quasi-lattice algebra” $\langle Q, \sqsubseteq, \sqcap, \sqcup \rangle$. In other words, a “quasi-lattice” is not yet a whole “quasi-lattice algebra”.

So, when the traditional order and lattice theory, which is based on poclasses, is generalized for quasi-ordered classes, it makes sense to distinguish “quasi-structures” from their concrete “quasi-structure algebras” instances.

We define the following particular quasi-algebras:

7.6.2 Definition —quasi-algebras—

(1) A (distributive) quasi-lattice algebra is a structure

$$\langle Q, \sqsubseteq, \equiv, \sqcap, \sqcup \rangle$$

where $\langle Q, \sqsubseteq \rangle$ is a (distributive) quasi-lattice, \equiv is the equivalence relation of \sqsubseteq and \sqcap is a meet and \sqcup is a join function.

(2) A (distributive) bounded quasi-lattice algebra is a structure

$$\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup \rangle$$

where $\langle Q, \sqsubseteq, \equiv, \sqcap, \sqcup \rangle$ is a (distributive) quasi-lattice algebra and \perp is a bottom and \top is a top element.

(3) A (distributive) complete quasi-lattice algebra is a structure

$$\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \sqcap, \sqcup \rangle$$

where $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup \rangle$ is a (distributive) bounded quasi-lattice algebra and \sqcap is an infimum and \sqcup is a supremum function.

(4) A quasi-boolean algebra is a structure

$$\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$$

where $\langle Q, \sqsubseteq \rangle$ is a boolean quasi-lattice, \neg is a complement function, and $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup \rangle$ is a bounded quasi-lattice algebra (which is distributive).

(5) A complete quasi-boolean algebra is a structure

$$\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \sqcap, \sqcup, \neg \rangle$$

which is both, a quasi-boolean algebra and a complete quasi-lattice algebra (which is distributive).

(6) A generalized quasi-boolean algebra is a structure

$$\langle Q, \sqsubseteq, \equiv, \perp, \sqcap, \sqcup, - \rangle$$

where $\langle Q, \sqsubseteq, \equiv, \sqcap, \sqcup \rangle$ is a distributive quasi-lattice algebra, \perp is a bottom element and $-$ is a relative complement function on Q .

7.6.3 Remark

\prod and \coprod in the previous definitions both have $\mathbf{P}(Q)$ as their domain, not just Q . According to the structure notation 5.8.2, we should therefore include $\mathbf{P}(Q)$ among the list of carrier classes and write e.g.

$$\langle Q, \mathbf{P}(Q), \sqsubseteq, \equiv, \perp, \sqcap, \sqcup, \prod, \coprod, \neg \rangle$$

for complete quasi-boolean algebras. But it is custom to omit the “ $\mathbf{P}(Q)$ ” and we will usually share this habit.

7.6.4 Lemma

If $\langle Q, \sqsubseteq \rangle$ is a quasi-ordered class, there is

- (1) $x \sqsubseteq y$ (reflexivity of \sqsubseteq)
 - (2) $x \sqsubseteq y$ and $y \sqsubseteq z$ implies $x \sqsubseteq z$ (transitivity of \sqsubseteq)
 - (3) $x \equiv y$ (reflexivity of \equiv)
 - (4) $x \equiv y$ and $y \equiv z$ implies $x \equiv z$ (transitivity of \equiv)
 - (5) $x \equiv y$ implies $y \equiv x$ (symmetry of \equiv)
 - (6) $x \equiv y$ iff $(x \sqsubseteq y$ and $y \sqsubseteq x)$ (\equiv is equivalence of \sqsubseteq)
- for all $x, y, z \in Q$.

7.6.5 Lemma

If $\mathfrak{Q} = \langle Q, \sqsubseteq, \equiv, \sqcap, \sqcup \rangle$ is a quasi-lattice algebra, there is

- (1) $x \sqsubseteq y$ iff $x \sqcap y \equiv x$ (order as meet)
- (2) $x \sqsubseteq y$ iff $x \sqcup y \equiv y$ (order as join)
- (3) $x \sqcap y \sqsubseteq x$ (meet is lower bound)
- (4) $x \sqsubseteq x \sqcup y$ (join is upper bound)
- (5) $(x \sqcap y) \sqcap z \equiv x \sqcap (y \sqcap z)$ (quasi-associativity of meet)
- (6) $(x \sqcup y) \sqcup z \equiv x \sqcup (y \sqcup z)$ (quasi-associativity of join)
- (7) $x \sqcap y \equiv y \sqcap x$ (quasi-commutativity of meet)
- (8) $x \sqcup y \equiv y \sqcup x$ (quasi-commutativity of join)
- (9) $x \sqcap x \equiv x$ (quasi-idempotency of meet)
- (10) $x \sqcup x \equiv x$ (quasi-idempotency of join)
- (11) $x \sqcap (x \sqcup y) \equiv x$ (quasi-absorption)
- (12) $x \sqcup (x \sqcap y) \equiv x$ (quasi-absorption)

And in case \mathfrak{Q} is distributive, we also have

- (13) $x \sqcap (y \sqcup z) \equiv (x \sqcap y) \sqcup (x \sqcap z)$ (distr. of \sqcap over \sqcup)
 - (14) $x \sqcup (y \sqcap z) \equiv (x \sqcup y) \sqcap (x \sqcup z)$ (distr. of \sqcup over \sqcap)
- for all $x, y, z \in Q$:

7.6.6 Lemma

If $\langle Q, \sqsubseteq, \equiv, \perp, \sqcap, \sqcup, \top, \bot \rangle$ is a bounded quasi-lattice algebra, there is

- (1) $\perp \sqsubseteq x$ (bottom is least element)
 - (2) $x \sqsubseteq \top$ (top is greatest element)
 - (3) $x \sqcap \perp \equiv \perp$ (bottom cancels meet)
 - (4) $x \sqcup \top \equiv \top$ (top cancels join)
 - (5) $x \sqcup \perp \equiv x$ (bottom is quasi-neutral element of join)
 - (6) $x \sqcap \top \equiv x$ (top is quasi-neutral element of meet)
- for all $x \in Q$.

7.6.7 Definition

If $\langle Q, \sqsubseteq, \equiv, \perp, \sqcap, \sqcup, \top, \bot \rangle$ is a bounded quasi-lattice algebra, we define the following notation for every $n \geq 0$:

$$\begin{aligned} \textcircled{1} \sqcap \dots \sqcap \textcircled{n} &:= \begin{cases} \top & \text{if } n = 0 \\ \textcircled{1} \sqcap (\textcircled{2} \sqcap \dots \sqcap \textcircled{n}) & \text{if } n > 0 \end{cases} \\ \textcircled{1} \sqcup \dots \sqcup \textcircled{n} &:= \begin{cases} \top & \text{if } n = 0 \\ \textcircled{1} \sqcup (\textcircled{2} \sqcup \dots \sqcup \textcircled{n}) & \text{if } n > 0 \end{cases} \end{aligned}$$

And for all $n, m \in \mathbb{N}$ we also put:

$$\begin{aligned} \prod_{i=n}^m x_i &:= x_n \sqcap x_{n+1} \sqcap \dots \sqcap x_m \\ \coprod_{i=n}^m x_i &:= x_n \sqcup x_{n+1} \sqcup \dots \sqcup x_m \end{aligned}$$

7.6.8 Lemma

Let $\langle Q, \sqsubseteq, \equiv, \perp, \sqcap, \sqcup, \top, \bot \rangle$ be a bounded distributive quasi-lattice algebra.

For all $n, m \in \mathbb{N}$ and $x, y_n, \dots, y_m \in Q$ holds

- (1) $x \sqcap \prod_{i=n}^m y_i \equiv \prod_{i=n}^m (x \sqcap y_i)$
- (2) $x \sqcup \prod_{i=n}^m y_i \equiv \prod_{i=n}^m (x \sqcup y_i)$

More general, for all $n_1, m_1, n_2, m_2 \in \mathbb{N}$ and $x_{n_1}, \dots, x_{m_1}, y_{n_2}, \dots, y_{m_2} \in Q$ holds

- (3) $\left(\prod_{i=n_1}^{m_1} x_i \right) \sqcap \left(\prod_{j=n_2}^{m_2} y_j \right) \equiv \prod_{i=n_1}^{m_1} \prod_{j=n_2}^{m_2} (x_i \sqcap y_j)$
- (4) $\left(\prod_{i=n_1}^{m_1} x_i \right) \sqcup \left(\prod_{j=n_2}^{m_2} y_j \right) \equiv \prod_{i=n_1}^{m_1} \prod_{j=n_2}^{m_2} (x_i \sqcup y_j)$

7.6.9 Lemma

If $\langle Q, \sqsubseteq, \equiv, \perp, \sqcap, \sqcup, \prod, \coprod \rangle$ is a complete quasi-lattice algebra then

- (1) $\top \equiv \prod \emptyset$ (empty infimum)
- (2) $\perp \equiv \coprod \emptyset$ (empty supremum)
- (3) $x \sqcap y \equiv \prod \{x, y\}$ (small as big conjunction)
- (4) $x \sqcup y \equiv \coprod \{x, y\}$ (small as big disjunction)

and in case of distributivity, there also is

- (5) $x \sqcap \prod S \equiv \prod \{x \sqcap y \mid y \in S\}$ (distr. of \sqcap over \prod)
- (6) $x \sqcup \prod S \equiv \prod \{x \sqcup y \mid y \in S\}$ (distr. of \sqcup over \prod)

for all $x, y \in Q$ and $S \subseteq Q$.

7.6.10 Lemma

If $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ is a quasi-boolean algebra, there is

- (1) $\perp \equiv \neg \top$ (negated top)
 - (2) $\top \equiv \neg \perp$ (negated bottom)
 - (3) $\neg x \sqcap x \equiv \perp$ (meet of complements)
 - (4) $\neg x \sqcup x \equiv \top$ (join of complements)
 - (5) $\neg \prod_{i=1}^n x_i \equiv \bigsqcup_{i=1}^n \neg x_i$ (de Morgan's law)
 - (6) $\neg \bigsqcup_{i=1}^n x_i \equiv \prod_{i=1}^n \neg x_i$ (de Morgan's law)
 - (7) $\neg \neg x \equiv x$ (double negation)
 - (8) $x \sqsubseteq y$ iff $\neg y \sqsubseteq \neg x$ (order inversion)
 - (9) $x \sqsubseteq y$ iff $\neg x \sqcup y \equiv \top$ (top criterion of order)
 - (10) $x \sqsubseteq y$ iff $x \sqcap \neg y \equiv \perp$ (bottom criterion of order)
- for all $x, x_1, \dots, x_n, y \in Q$ with $n \in \mathbb{N}$.

7.6.11 Lemma

If $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \prod, \bigsqcup, \neg \rangle$ is a complete quasi-boolean algebra, there is

- (1) $\neg \prod S \equiv \bigsqcup \{\neg y \mid y \in S\}$ (de Morgan's law)
 - (2) $\neg \bigsqcup S \equiv \prod \{\neg y \mid y \in S\}$ (de Morgan's law)
- for all $S \subseteq Q$.

7.6.12 Lemma

(1) If $\langle Q, \sqsubseteq, \equiv, \sqcap, \sqcup \rangle$ is a quasi-lattice algebra, then the following statements are equivalent for all $x, y \in Q$:

- (a) $x \sqsubseteq y$
- (b) $x \sqcup y \equiv y$
- (c) $x \sqcap y \equiv x$

(2) If $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ is a quasi-boolean algebra, then the following statements are equivalent for all $x, y \in Q$:

- (a) $x \sqsubseteq y$
- (b) $\neg y \sqsubseteq \neg x$
- (c) $x \sqcap \neg y \equiv \perp$
- (d) $\neg x \sqcup y \equiv \top$

7.6.13 Lemma

If $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup \rangle$ is a bounded quasi-lattice algebra and $x, y \in Q$, then

- (1) $(x \equiv \top \text{ and } y \equiv \top)$ iff $x \sqcap y \equiv \top$
- (2) $(x \equiv \perp \text{ and } y \equiv \perp)$ iff $x \sqcup y \equiv \perp$
- (3) $(x \equiv \top \text{ or } y \equiv \top)$ implies $x \sqcup y \equiv \top$
- (4) $(x \equiv \perp \text{ or } y \equiv \perp)$ implies $x \sqcap y \equiv \perp$

If $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \prod, \bigsqcup \rangle$ is a complete quasi-lattice algebra and $S \subseteq Q$, then

- (5) $(\forall y \in S. y \equiv \top)$ iff $\prod S \equiv \top$
- (6) $(\forall y \in S. y \equiv \perp)$ iff $\bigsqcup S \equiv \perp$
- (7) $(\exists y \in S. y \equiv \perp)$ implies $\prod S \equiv \perp$
- (8) $(\exists y \in S. y \equiv \top)$ implies $\bigsqcup S \equiv \top$

If $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ is a quasi-boolean algebra and $x \in Q$, then

- (9) $\neg x \equiv \top$ iff $x \equiv \perp$
- (10) $\neg x \equiv \perp$ iff $x \equiv \top$

7.6.14 Lemma

If $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \prod, \bigsqcup \rangle$ is a complete quasi-lattice algebra, there is

- (1) $(y \sqsubseteq x \text{ for all } y \in S)$ iff $\prod S \sqsubseteq x$
 - (2) $(y \sqsubseteq x \text{ for one } y \in S)$ implies $\prod S \sqsubseteq x$
 - (3) $(x \sqsubseteq y \text{ for all } y \in S)$ iff $x \sqsubseteq \bigsqcup S$
 - (4) $(x \sqsubseteq y \text{ for one } y \in S)$ implies $x \sqsubseteq \bigsqcup S$
- for all $x, y, z \in Q$ and $S \subseteq Q$.

7.6.15 Lemma

If $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, - \rangle$ is a generalized quasi-boolean algebra, there is

- (1) $\perp - x \equiv \perp$
 - (2) $x - \perp \equiv x$
 - (3) $x - (y \sqcap z) \equiv (x - y) \sqcup (x - z)$
 - (4) $x - (y \sqcup z) \equiv (x - y) \sqcap (x - z)$
 - (5) $(y \sqcap z) - x \equiv (y - x) \sqcap (z - x)$
 - (6) $(y \sqcup z) - x \equiv (y - x) \sqcup (z - x)$
 - (7) $(x - y) - z \equiv x - (y \sqcup z)$
 - (8) $x - (y - z) \equiv (x - y) \sqcup (x \sqcap z)$
 - (9) $(x - y) \sqcap z \equiv (x \sqcap z) - y$
 - (10) $(x - y) \sqcup z \equiv (x \sqcup z) - (y - z)$
- for all $x, y, z \in Q$.

7.6.16 Lemma

If $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ is a quasi-boolean algebra, then

$$\textcircled{1} - \textcircled{2} := \begin{bmatrix} Q \times Q \longrightarrow Q \\ \langle x, y \rangle \mapsto x \sqcap \neg y \end{bmatrix}$$

is a relative complement function with

- (1) $x - \top \equiv \top$
- (2) $\top - x \equiv \neg x$

for all $x \in Q$.

7.7 Quotient structures**7.7.1** Definition

Let \sim be an equivalence relation on a given class Q .

(1) For every $x \in Q$ we define

$$x/\sim := \{y \in Q \mid y \sim x\}$$

the equivalence class of x modulo \sim

(2) We define

$$Q/\sim := \{x/\sim \mid x \in Q\}$$

the quotient class of Q modulo \sim

7.7.2 Definition

Given an equivalence relation \sim on a class Q .

- (1) Let f be an n -ary function on Q , i.e. $f : Q^n \rightarrow Q$. We say that f is \sim -compatible, if $x_1 \sim y_1, \dots, x_n \sim y_n$ implies $f(x_1, \dots, x_n) \sim f(y_1, \dots, y_n)$, for all $x_1, \dots, x_n, y_1, \dots, y_n \in Q$.

And in that case the following function is well-defined

$$f/\sim := \left[\begin{array}{c} (Q/\sim)^n \rightarrow (Q/\sim) \\ \langle x_1/\sim, \dots, x_n/\sim \rangle \mapsto f(x_1, \dots, x_n)/\sim \end{array} \right]$$

the quotient function of f modulo \sim

- (2) Let R be a n -ary relation on Q , i.e. $R : \mathbf{Rel}((Q \mid n))$. We say that R is \sim -compatible, if $x_1 \sim y_1, \dots, x_n \sim y_n$ implies $R(x_1, \dots, x_n) \Leftrightarrow R(y_1, \dots, y_n)$, for all $x_1, \dots, x_n, y_1, \dots, y_n \in Q$.

And in that case the following n -ary relation is well-defined

$$R/\sim := \left[\begin{array}{c} (Q/\sim) \rightsquigarrow \dots \rightsquigarrow (Q/\sim) \\ \langle x_1/\sim, \dots, x_n/\sim \rangle \rightsquigarrow R(x_1, \dots, x_n) \end{array} \right]$$

the quotient relation of R modulo \sim

7.7.3 Definition

Given an ordinary structure

$$\mathfrak{C} = \langle C, c_1, \dots, f_1, \dots, R_1, \dots \rangle$$

i.e. C is a class, the c_i are constants from C , the f_i are ordinary functions on C and the R_i are ordinary relations on C .

If \sim is an equivalence relation on C , we say that \mathfrak{C} is \sim -compatible or \sim is a congruence relation on \mathfrak{C} , if each f_i is \sim -compatible and each R_i is \sim -compatible. And in that case, we define

$$\mathfrak{C}/\sim := \langle C/\sim, c_1/\sim, \dots, f_1/\sim, \dots, R_1/\sim \rangle$$

the quotient structure of \mathfrak{C} modulo \sim

7.7.4 Lemma

For every quasi-ordered class $\Omega = \langle Q, \sqsubseteq, \equiv \rangle$ holds:

- (1) Ω/\equiv is a (well-defined) poclass.
- (2) Ω is a quasi-lattice iff Ω/\equiv is a lattice.
And if that is the case, then
 - (a) Ω is bounded iff Ω/\equiv is bounded
 - (b) Ω is distributive iff Ω/\equiv distributive
 - (c) Ω is complemented iff Ω/\equiv complemented
 - (d) Ω is boolean iff Ω/\equiv is boolean
 - (e) Ω is complete iff Ω/\equiv is complete
 - (f) Ω is generalized boolean iff Ω/\equiv is generalized boolean

8 Additional junctors for quasi-boolean structures

8.0.5 Remark

We introduce two additional junctors that can be derived from every quasi-boolean algebra: the *subjunctive* “ \rightarrow ” and the *equijunctive* “ \leftrightarrow ”, both not in their common binary version, but more general as *multiary* functions.

This whole section 8 is not an essential part of the overall text. It can and probably should be skipped on a first reading.

8.1 Junctor sets

8.1.1 Remark

Given a quasi-boolean lattice $\langle Q, \sqsubseteq, \equiv \rangle$. We can add the usual operations $\perp, \top, \sqcap, \sqcup, \neg$ to turn it into a quasi-boolean algebra, as described before. This set of operations is (*quasi-*) *complete* in the sense that we can define any (\sqsubseteq -related) n -ary junctor $j : Q^n \rightarrow Q$ in terms of these five.²⁴ But this set is not *minimal*, because we can remove some of them and the remaining ones are still complete. For example, we can remove \perp and \sqcap , because they are given in terms of \top, \sqcup, \neg via $\perp \equiv \neg\top$ and $x \sqcap y \equiv \neg(\neg x \sqcup \neg y)$.

There are plenty of alternative definitions for alternative junctor set, in particular in the logic tradition, and there are also more general investigations that compare these systems. For our part here we concentrate on two additional derived junctors that are useful for our purposes, such as logic.

8.2 Multiary sub- and equijunction

8.2.1 introduction

In particular in logic, two other junctors are often used:²⁵ the *subjunctive* \rightarrow and the *equijunction* \leftrightarrow . Their usual idea is the resemblance with the sub- and equivalence, respectively, in the sense that

$$x \rightarrow y \equiv \top \quad \text{iff} \quad x \sqsubseteq y$$

$$x \leftrightarrow y \equiv \top \quad \text{iff} \quad x \equiv y$$

The following definition is accordingly (see 8.2.18), but is generalized to arbitrary numbers of arguments.

This *multi-ary* generalization is not very common, maybe a novelty, but for us it has two advantages: For (propositional) logic (6.4.1) the restrictions to binary forms like $[\textcircled{1} \wedge \textcircled{2}]$ do not really make a modern calculus, because they make (dis-

junctive or conjunctive) normal forms very nested and awkward. And since we use multiary forms like $[\textcircled{1} \wedge \dots \wedge \textcircled{n}]$ in our formula syntax anyway, it seems more elegant to remove binary restrictions altogether and have the $[\textcircled{1} \rightarrow \dots \rightarrow \textcircled{n}]$ and $[\textcircled{1} \leftrightarrow \dots \leftrightarrow \textcircled{n}]$ available, as well. Secondly, these two multiary junctors are direct representations of two important notions: “ $\textcircled{1}, \dots, \textcircled{n}$ are in order (or a chain)” and “the $\textcircled{1}, \dots, \textcircled{n}$ are an equivalence class”, respectively.

8.2.2 Definition

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra. We define

$$\langle \rightarrow \textcircled{1} \rangle := \left[\begin{array}{c} Q^* \rightarrow Q \\ \langle x_1, \dots, x_n \rangle \mapsto \prod_{i=1}^{n-1} (\neg x_i \sqcup x_{i+1}) \end{array} \right] \quad \text{the subjunctive}$$

$$\langle \leftrightarrow \textcircled{1} \rangle := \left[\begin{array}{c} Q^* \rightarrow Q \\ \langle x_1, \dots, x_n \rangle \mapsto \left(\prod_{i=1}^n \neg x_i \right) \sqcup \left(\prod_{i=1}^n x_i \right) \end{array} \right] \quad \text{the equijunctive}$$

8.2.3 Definition notation

We usually write

$$x_1 \rightarrow \dots \rightarrow x_n \quad \text{instead of} \quad (\rightarrow \langle x_1, \dots, x_n \rangle)$$

$$x_1 \leftrightarrow \dots \leftrightarrow x_n \quad \text{instead of} \quad (\leftrightarrow \langle x_1, \dots, x_n \rangle)$$

and if these expressions have to be written explicitly for $n \leq 1$, we write

$$\langle \rightarrow \rangle \quad \text{for} \quad (\rightarrow \langle \rangle) \quad \langle \leftrightarrow \rangle \quad \text{for} \quad (\leftrightarrow \langle \rangle)$$

$$\langle \rightarrow x \rangle \quad \text{for} \quad (\rightarrow \langle x \rangle) \quad \langle \leftrightarrow x \rangle \quad \text{for} \quad (\leftrightarrow \langle x \rangle)$$

Furthermore, we also use yet another version and write

$$\prod_{i=n}^m x_i \quad \text{for} \quad x_m \rightarrow x_{m+1} \rightarrow \dots \rightarrow x_n$$

$$\prod_{i=n}^m x_i \quad \text{for} \quad x_m \leftrightarrow x_{m+1} \leftrightarrow \dots \leftrightarrow x_n$$

for arbitrary $m, n \in \mathbb{N}$.

8.2.4 Remark

Of course, if $n > m$ then

$$\prod_{i=n}^m x_i = (\rightarrow) = \prod_{i=n}^m (\neg x_i \sqcup x_{i+1}) = \top$$

²⁴In terms of propositional logic, this completeness of operations means that every truth table can be represented by a formula.

²⁵There is no real standard for their names. Our “subjunctive” and “equijunction” here is not very common, but it fits nicely into the overall terminology, in particular due to their familiarity with the “subvalence” (“ \Rightarrow ” or “ \sqsubseteq ”) and “equivalence” (“ \Leftrightarrow ” or “ \equiv ”), respectively. The symbols “ \rightarrow ” and “ \leftrightarrow ” have become more or less standard in mathematics now, although there is another tradition in logic that often uses “ \subset ” for “ \rightarrow ”.

and

$$\prod_{i=n}^m x_i = (\leftrightarrow) = \left(\prod_{i=n}^m \neg x_i \right) \sqcup \left(\prod_{i=n}^m x_i \right) = \top \sqcup \top \equiv \top$$

according to the definition of nullary conjunctions (in 7.6.7).

And if $n = m$ then

$$\prod_{i=n}^m x_i = (\rightarrow x_n) = \prod_{i=n}^{n-1} (\neg x_i \sqcup x_{i+1}) = \top$$

and

$$\prod_{i=n}^m x_i = (\leftrightarrow x_n) = \left(\prod_{i=n}^{n-1} \neg x_i \right) \sqcup \left(\prod_{i=n}^{n-1} x_i \right) = \top \sqcup \top \equiv \top$$

8.2.5 Lemma

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra. For all $x, y, z \in Q$ and $x_1, \dots, x_n, y_1, \dots, y_m \in Q$ with $n, m \in \mathbb{N}$ holds:

(1) Nullary and unary cases:

- (a) $(\rightarrow) \equiv \top$
- (b) $(\leftrightarrow) \equiv \top$
- (c) $(\rightarrow x) \equiv \top$
- (d) $(\leftrightarrow x) \equiv \top$

(2) The common binary case:

- (a) $x \rightarrow y \equiv \neg x \sqcup y$
- (b) $x \leftrightarrow y \equiv (x \rightarrow y) \sqcap (y \rightarrow x)$

But also:

- (c) $x \leftrightarrow y \equiv (\neg x \sqcap \neg y) \sqcup (x \sqcap y)$
- (d) $x \leftrightarrow y \equiv x \rightarrow y \rightarrow x \equiv y \rightarrow x \rightarrow y$

(3) Unipotency:

- (a) $x \rightarrow x \equiv \top$
- (b) $x \leftrightarrow x \equiv \top$

(4) For the combinations with \perp and \top holds:

- (a) $\perp \rightarrow x \equiv \top$
- (b) $\top \rightarrow x \equiv x$
- (c) $x \rightarrow \perp \equiv \neg x$
- (d) $x \rightarrow \top \equiv \top$
- (e) $\perp \leftrightarrow x \equiv \neg x$
- (f) $\top \leftrightarrow x \equiv x$
- (g) $x \leftrightarrow \perp \equiv \neg x$
- (h) $x \leftrightarrow \top \equiv x$

See also 8.2.16 for more general laws.

8.2.6 Proof of 8.2.5

(1) Immediate consequences of definition 8.2.2; see 8.2.4.

(2) (a) is an application of definition 8.2.2. For (b) holds:

$$\begin{aligned} & (x \rightarrow y) \sqcap (y \rightarrow x) \\ & \equiv (\neg x \sqcup y) \sqcap (\neg y \sqcup x) && \text{due to (a)} \\ & \equiv (\neg x \sqcap \neg y) \sqcup (\neg x \sqcap \neg x) \sqcup (y \sqcap \neg y) \sqcup (y \sqcap x) && \text{distributivity} \\ & \equiv (\neg x \sqcap \neg y) \sqcup \perp \sqcup \perp \sqcup (y \sqcap x) && \text{complement} \\ & \equiv (\neg x \sqcap \neg y) \sqcup (y \sqcap x) && \text{because } \perp \text{ is neutral for } \sqcup \\ & \equiv (\neg x \sqcap \neg y) \sqcup (x \sqcap y) && \text{commutativity of } \sqcup \\ & \equiv x \leftrightarrow y && \text{def. of } \leftrightarrow \end{aligned}$$

(c) is definition 8.2.2 for two arguments, and (d) is true because

$$x \leftrightarrow y$$

$$\begin{aligned} & \equiv (\neg x \sqcap \neg y) \sqcup (x \sqcap y) \\ & \equiv (\neg x \sqcup x) \sqcap (\neg x \sqcup y) \sqcap (\neg y \sqcup x) \sqcap (\neg y \sqcup y) && \text{due to the distributivity} \\ & \equiv \top \sqcap (\neg x \sqcup y) \sqcap (\neg y \sqcup x) \sqcap \top \\ & \equiv (\neg x \sqcup y) \sqcap (\neg y \sqcup x) \\ & \equiv x \rightarrow y \rightarrow x && \text{definition 8.2.2} \end{aligned}$$

and similarly for $x \leftrightarrow y \equiv y \rightarrow x \rightarrow y$.

(3) A proof for (a) is

$$\begin{aligned} & x \rightarrow x \\ & \equiv \neg x \sqcup x && \text{due to (2)(a)} \\ & \equiv \top && \text{complement} \end{aligned}$$

A proof of (b) is

$$\begin{aligned} & x \leftrightarrow x \\ & \equiv (\neg x \sqcap \neg x) \sqcup (x \sqcap x) && \text{def. 8.2.2} \\ & \equiv \neg x \sqcup x && \text{idempotency of } \sqcap \\ & \equiv \top && \text{complement} \end{aligned}$$

(4) For the combination with \perp and \top we use (2)(a) and the common laws for \perp and \top (7.6.6 and 7.6.10) and obtain:

- (a) $\perp \rightarrow x \equiv \neg \perp \sqcup x \equiv \top \sqcup x \equiv \top$
- (b) $\top \rightarrow x \equiv \neg \top \sqcup x \equiv \perp \sqcup x \equiv x$
- (c) $x \rightarrow \perp \equiv \neg x \sqcup \perp \equiv \neg x$
- (d) $x \rightarrow \top \equiv \neg x \sqcup \top \equiv \top$
- (e) $\perp \leftrightarrow x \equiv (\neg \perp \sqcap \neg x) \sqcup (\perp \sqcap x) \equiv \neg x \sqcup \perp \equiv \neg x$
- (f) $\top \leftrightarrow x \equiv (\neg \top \sqcap \neg x) \sqcup (\top \sqcap x) \equiv \perp \sqcup x \equiv x$
- (g) $x \leftrightarrow \perp \equiv (\neg x \sqcap \neg \perp) \sqcup (x \sqcap \perp) \equiv \neg x \sqcup \perp \equiv \neg x$
- (h) $x \leftrightarrow \top \equiv (\neg x \sqcap \neg \top) \sqcup (x \sqcap \top) \equiv \perp \sqcup x \equiv x$

8.2.7 Lemma

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra. For all $x_1, \dots, x_n \in Q$ with $n \in \mathbb{N}$ holds:

- (1) $\neg(x_1 \rightarrow \dots \rightarrow x_n) \equiv (x_1 \sqcap \neg x_2) \sqcup \dots \sqcup (x_{n-1} \sqcap \neg x_n)$
- (2) $\neg(x_1 \leftrightarrow \dots \leftrightarrow x_n) \equiv (x_1 \sqcup \dots \sqcup x_n) \sqcap (\neg x_1 \sqcup \dots \sqcup \neg x_n)$

8.2.8 Proof of 8.2.7

In both cases, de Morgans law is applied to definition 8.2.2:

$$\begin{aligned} & \neg(x_1 \rightarrow \dots \rightarrow x_n) \\ & \equiv \neg((\neg x_1 \sqcup x_2) \sqcap \dots \sqcap (\neg x_{n-1} \sqcup x_n)) \\ & \equiv \neg(\neg x_1 \sqcup x_2) \sqcup \dots \sqcup \neg(\neg x_{n-1} \sqcup x_n) \\ & \equiv (\neg \neg x_1 \sqcap \neg x_2) \sqcup \dots \sqcup (\neg \neg x_{n-1} \sqcap \neg x_n) \\ & \equiv (x_1 \sqcap \neg x_2) \sqcup \dots \sqcup (x_{n-1} \sqcap \neg x_n) \end{aligned}$$

and

$$\begin{aligned} & \neg(x_1 \leftrightarrow \dots \leftrightarrow x_n) \\ & \equiv \neg((\neg x_1 \sqcap \dots \sqcap \neg x_n) \sqcup (x_1 \sqcap \dots \sqcap x_n)) \\ & \equiv \neg(\neg x_1 \sqcap \dots \sqcap \neg x_n) \sqcap \neg(x_1 \sqcap \dots \sqcap x_n) \\ & \equiv (\neg \neg x_1 \sqcup \dots \sqcup \neg \neg x_n) \sqcap (\neg x_1 \sqcup \dots \sqcup \neg x_n) \\ & \equiv (x_1 \sqcup \dots \sqcup x_n) \sqcap (\neg x_1 \sqcup \dots \sqcup \neg x_n) \end{aligned}$$

8.2.9 Remark

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra. For the commutativity and associativity of \rightarrow and \leftrightarrow holds:

(1) \rightarrow is not commutative.

Even binary subjunctives like $x \rightarrow y$ and $y \rightarrow x$ are not equivalent in general. For example, $\top \rightarrow \perp \equiv \perp \not\equiv \top \equiv \perp \rightarrow \top$.

(2) \leftrightarrow is commutative in general, i.e.

(a) $x_{\nu(1)} \leftrightarrow \dots \leftrightarrow x_{\nu(n)} \equiv x_1 \leftrightarrow \dots \leftrightarrow x_n$

for every $n \in \mathbb{N}$, all $x_1, \dots, x_n \in Q$ and every bijection $\nu : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$.

This derives from the commutativity of both \sqcap and \sqcup , because

$$\begin{aligned} & x_{\nu(1)} \leftrightarrow \dots \leftrightarrow x_{\nu(n)} \\ & \equiv (\neg x_{\nu(1)} \sqcap \dots \sqcap \neg x_{\nu(n)}) \sqcup (x_{\nu(1)} \sqcap \dots \sqcap x_{\nu(n)}) \\ & \equiv (\neg x_1 \sqcap \dots \sqcap \neg x_n) \sqcup (x_1 \sqcap \dots \sqcap x_n) \\ & \equiv x_1 \leftrightarrow \dots \leftrightarrow x_n \end{aligned}$$

(3) \rightarrow is not associative.

For example, $x \rightarrow y \rightarrow z$ and $(x \rightarrow y) \rightarrow z$ are often not equivalent:

$$\begin{aligned} \perp \rightarrow \perp \rightarrow \perp & \equiv \top \not\equiv \perp \equiv (\perp \rightarrow \perp) \rightarrow \perp \\ \top \rightarrow \perp \rightarrow \perp & \equiv \perp \not\equiv \top \equiv (\top \rightarrow \perp) \rightarrow \perp \end{aligned}$$

However, we have a law saying that

(a) $(x \rightarrow y) \rightarrow z \sqsubseteq x \rightarrow (y \rightarrow z)$ for all $x, y, z \in Q$

because

$$\begin{aligned} (x \rightarrow y) \rightarrow z & \equiv \neg(\neg x \sqcup y) \sqcup z \\ & \equiv (x \sqcap \neg y) \sqcup z \\ & \sqsubseteq \neg y \sqcup z \\ & \sqsubseteq \neg x \sqcup \neg y \sqcup z \\ & \equiv \neg x \sqcup (\neg y \sqcup z) \\ & \equiv x \rightarrow (y \rightarrow z) \end{aligned}$$

(4) \leftrightarrow is not associative.

For example, $x \leftrightarrow y \leftrightarrow z$ and $(x \leftrightarrow y) \leftrightarrow z$ are often not equivalent:

$$\begin{aligned} \perp \leftrightarrow \perp \leftrightarrow \perp & \equiv \top \not\equiv \perp \equiv (\perp \leftrightarrow \perp) \leftrightarrow \perp \\ \top \leftrightarrow \perp \leftrightarrow \perp & \equiv \perp \not\equiv \top \equiv (\top \leftrightarrow \perp) \leftrightarrow \perp \end{aligned}$$

However, we have a law saying that

(a) $(x \leftrightarrow y) \leftrightarrow z \equiv x \leftrightarrow (y \leftrightarrow z)$ for all $x, y, z \in Q$

because

$$\begin{aligned} & (x \leftrightarrow y) \leftrightarrow z \\ & \equiv (\neg(x \leftrightarrow y) \sqcap \neg z) \sqcup ((x \leftrightarrow y) \sqcap z) \\ & \equiv (((\neg x \sqcap y) \sqcup (x \sqcap \neg y)) \sqcap \neg z) \sqcup (((\neg x \sqcap \neg y) \sqcup (x \sqcap y)) \sqcap z) \\ & \equiv (\neg x \sqcap y \sqcap \neg z) \sqcup (x \sqcap \neg y \sqcap \neg z) \sqcup (\neg x \sqcap \neg y \sqcap z) \sqcup (x \sqcap y \sqcap z) \\ & \equiv (\neg x \sqcap \neg y \sqcap z) \sqcup (\neg x \sqcap y \sqcap \neg z) \sqcup (x \sqcap \neg y \sqcap \neg z) \sqcup (x \sqcap y \sqcap z) \\ & \equiv (\neg x \sqcap ((\neg y \sqcap z) \sqcup (y \sqcap \neg z))) \sqcup (x \sqcap ((\neg y \sqcap \neg z) \sqcup (y \sqcap z))) \\ & \equiv (\neg x \sqcap \neg(y \leftrightarrow z)) \sqcup (x \sqcap (y \leftrightarrow z)) \\ & \equiv x \leftrightarrow (y \leftrightarrow z) \end{aligned}$$

8.2.10 Lemma

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra and $y \in Q$ and $x_1, \dots, x_n \in Q$ with $n \in \mathbb{N}$.

(1) \rightarrow distributes over \sqcap and \sqcup as follows:

$$(a) \bigcap_{i=1}^n (y \rightarrow x_i) \equiv y \rightarrow \left(\bigcap_{i=1}^n x_i \right)$$

$$(b) \bigcap_{i=1}^n (x_i \rightarrow y) \equiv \left(\bigcup_{i=1}^n x_i \right) \rightarrow y$$

$$(c) \bigcup_{i=1}^n (y \rightarrow x_i) \equiv y \rightarrow \left(\bigcup_{i=1}^n x_i \right)$$

$$(d) \bigcup_{i=1}^n (x_i \rightarrow y) \equiv \left(\bigcap_{i=1}^n x_i \right) \rightarrow y$$

(2) For similar constructions with \leftrightarrow holds:

$$(a) \bigcap_{i=1}^n (y \leftrightarrow x_i) \equiv (y \leftrightarrow x_1 \leftrightarrow \dots \leftrightarrow x_n)$$

$$(b) \bigcap_{i=1}^n (y \leftrightarrow x_i) \equiv \left(\bigcup_{i=1}^n x_i \right) \rightarrow y \rightarrow \left(\bigcap_{i=1}^n x_i \right)$$

8.2.11 Proof of 8.2.10

(1) We have:

$$\begin{aligned} & \bigcap_{i=1}^n (y \rightarrow x_i) \\ & \equiv \bigcap_{i=1}^n (\neg y \sqcup x_i) && \text{due to 8.2.5(2)(a)} \\ & \equiv \neg y \sqcup \left(\bigcap_{i=1}^n x_i \right) && \text{due to 7.6.5(14)} \\ & \equiv y \rightarrow \left(\bigcap_{i=1}^n x_i \right) && \text{again, 8.2.5(2)(a)} \end{aligned}$$

$$\begin{aligned} & \bigcap_{i=1}^n (x_i \rightarrow y) \\ & \equiv \bigcap_{i=1}^n (\neg x_i \sqcup y) && \text{due to 8.2.5(2)(a)} \\ & \equiv (\neg x_1 \sqcap \dots \sqcap \neg x_n) \sqcup y && \text{due to 7.6.5(14)} \\ & \equiv \neg(x_1 \sqcup \dots \sqcup x_n) \sqcup y && \text{due to 7.6.10(5)} \\ & \equiv \left(\bigcup_{i=1}^n x_i \right) \rightarrow y && \text{due to 8.2.5(2)(a)} \end{aligned}$$

$$\begin{aligned} & \bigcup_{i=1}^n (y \rightarrow x_i) \\ & \equiv \bigcup_{i=1}^n (\neg y \sqcup x_i) && \text{due to 8.2.5(2)(a)} \\ & \equiv \neg y \sqcup \left(\bigcup_{i=1}^n x_i \right) && \text{because } \sqcup \text{ is associative, commutative and idempotent} \\ & \equiv y \rightarrow \left(\bigcup_{i=1}^n x_i \right) && \text{again, 8.2.5(2)(a)} \end{aligned}$$

$$\begin{aligned} & \bigcup_{i=1}^n (x_i \rightarrow y) \\ & \equiv \bigcup_{i=1}^n (\neg x_i \sqcup y) && \text{due to 8.2.5(2)(a)} \\ & \equiv (\neg x_1 \sqcup \dots \sqcup \neg x_n) \sqcup y && \text{because } \sqcup \text{ is associative, commutative and idempotent} \\ & \equiv \neg(x_1 \sqcap \dots \sqcap x_n) \sqcup y && \text{due to 7.6.10(6)} \\ & \equiv \left(\bigcap_{i=1}^n x_i \right) \rightarrow y && \text{due to 8.2.5(2)(a)} \end{aligned}$$

(2)(a) We proof by induction on n :

♣ For $n = 0$ holds:

$$\bigcap_{i=1}^n (y \leftrightarrow x_i) \equiv \top \equiv (\leftrightarrow y) \equiv (y \leftrightarrow x_1 \leftrightarrow \dots \leftrightarrow x_n)$$

♣ For $n = 1$ holds:

$$\bigcap_{i=1}^n (y \leftrightarrow x_i) \equiv (y \leftrightarrow x_1) \equiv (y \leftrightarrow x_1 \leftrightarrow \dots \leftrightarrow x_n)$$

♣ For $n \mapsto n + 1$ holds:

$$\bigcap_{i=1}^{n+1} (y \leftrightarrow x_i)$$

$$\begin{aligned}
&\equiv \left(\prod_{i=1}^n (y \leftrightarrow x_i) \right) \sqcap (y \leftrightarrow x_{n+1}) \\
&\equiv (y \leftrightarrow x_1 \leftrightarrow \dots \leftrightarrow x_n) \sqcap (y \leftrightarrow x_{n+1}) \quad \text{induction step} \\
&\equiv \left(\begin{array}{c} (\neg y \sqcap \neg x_1 \sqcap \dots \sqcap \neg x_n) \\ \sqcup \\ (y \sqcap x_1 \sqcap \dots \sqcap x_n) \end{array} \right) \sqcap \left(\begin{array}{c} (\neg y \sqcap \neg x_{n+1}) \\ \sqcup \\ (y \sqcap x_{n+1}) \end{array} \right) \\
&\quad \text{definition 8.2.2} \\
&\equiv \left(\begin{array}{c} ((\neg y \sqcap \neg x_1 \sqcap \dots \sqcap \neg x_n) \sqcap (\neg y \sqcap \neg x_{n+1})) \sqcup \\ ((\neg y \sqcap \neg x_1 \sqcap \dots \sqcap \neg x_n) \sqcap (y \sqcap x_{n+1})) \sqcup \\ ((y \sqcap x_1 \sqcap \dots \sqcap x_n) \sqcap (\neg y \sqcap \neg x_{n+1})) \sqcup \\ ((y \sqcap x_1 \sqcap \dots \sqcap x_n) \sqcap (y \sqcap x_{n+1})) \end{array} \right) \\
&\quad \text{distributivity 7.6.8(3)} \\
&\equiv \left(\begin{array}{c} (\neg y \sqcap \neg y \sqcap \neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap \neg x_{n+1}) \sqcup \\ (\neg y \sqcap y \sqcap \neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap x_{n+1}) \sqcup \\ (y \sqcap \neg y \sqcap x_1 \sqcap \dots \sqcap x_n \sqcap \neg x_{n+1}) \sqcup \\ (y \sqcap y \sqcap x_1 \sqcap \dots \sqcap x_n \sqcap x_{n+1}) \end{array} \right) \\
&\quad \text{associativity and commutativity of } \sqcap \\
&\equiv \left(\begin{array}{c} (\neg y \sqcap \neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap \neg x_{n+1}) \sqcup \\ \perp \sqcup \\ \perp \sqcup \\ (y \sqcap x_1 \sqcap \dots \sqcap x_n \sqcap x_{n+1}) \end{array} \right) \\
&\equiv \left(\begin{array}{c} (\neg y \sqcap \neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap \neg x_{n+1}) \sqcup \\ (y \sqcap x_1 \sqcap \dots \sqcap x_n \sqcap x_{n+1}) \end{array} \right) \\
&\equiv y \leftrightarrow x_1 \leftrightarrow \dots \leftrightarrow x_n \quad \text{def. 8.2.2, again}
\end{aligned}$$

(2)(b) We obtain

$$\begin{aligned}
&\prod_{i=1}^n (y \leftrightarrow x_i) \\
&\equiv \prod_{i=1}^n ((x_i \rightarrow y) \sqcap (y \rightarrow x_i)) \quad \text{due to 8.2.5(2)(b)} \\
&\equiv \left(\prod_{i=1}^n (x_i \rightarrow y) \right) \sqcap \left(\prod_{i=1}^n (y \rightarrow x_i) \right) \\
&\quad \text{because } \sqcap \text{ is associative and commutative} \\
&\equiv \left(\left(\prod_{i=1}^n x_i \right) \rightarrow y \right) \sqcap \left(y \rightarrow \left(\prod_{i=0}^n x_i \right) \right) \\
&\quad \text{due to (1)(a) and (1)(b)} \\
&\equiv \left(\prod_{i=1}^n x_i \right) \rightarrow y \rightarrow \left(\prod_{i=1}^n x_i \right) \\
&\quad \text{def. 8.2.2 of } \rightarrow
\end{aligned}$$

8.2.12 Lemma

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra. For all $x, y, z \in Q$ and $x_1, \dots, x_n, y_1, \dots, y_m \in Q$ with $n, m \in \mathbb{N}$ holds:

(1) Some alternative characterizations of \rightarrow are

$$\begin{aligned}
\text{(a)} \quad \prod_{i=1}^n x_i &\equiv \prod_{i=1}^{n-1} (x_i \rightarrow x_{i+1}) \\
\text{(b)} \quad \prod_{i=1}^n x_i &\equiv \left(\prod_{i=1}^j x_i \right) \sqcap \left(\prod_{i=j}^n x_i \right) \quad \text{for } 1 \leq j \leq n
\end{aligned}$$

(2) Other characterizations of \leftrightarrow are

$$\begin{aligned}
\text{(a)} \quad \prod_{i=1}^n x_i &\equiv \prod_{i=1}^{n-1} (x_i \leftrightarrow x_{i+1}) \\
\text{(b)} \quad \prod_{i=1}^n x_i &\equiv \left(\prod_{i=1}^j x_i \right) \sqcap \left(\prod_{i=j}^n x_i \right) \quad \text{for } 1 \leq j \leq n \\
\text{(c)} \quad \prod_{i=1}^n x_i &\equiv \prod_{i=1}^n \prod_{j=1}^n (x_i \rightarrow x_j) \\
\text{(d)} \quad \prod_{i=1}^n x_i &\equiv \prod_{i=1}^n \prod_{j=1}^n (x_i \leftrightarrow x_j) \\
\text{(e)} \quad \prod_{i=1}^n x_i &\equiv \left(\prod_{i=1}^n x_i \right) \sqcap \left(\prod_{i=0}^{n-1} x_{n-i} \right) \\
\text{(f)} \quad \prod_{i=1}^n x_i &\equiv \left(\prod_{i=1}^n x_i \right) \rightarrow \left(\prod_{i=1}^n x_i \right)
\end{aligned}$$

8.2.13 Proof of 8.2.12

(1)(a) We derive

$$\begin{aligned}
&\prod_{i=1}^n x_i \\
&\equiv x_1 \rightarrow \dots \rightarrow x_n \\
&\equiv (\neg x_1 \sqcup x_2) \sqcap \dots \sqcap (\neg x_{n-1} \sqcup x_n) \quad \text{def. 8.2.2} \\
&\equiv (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \\
&\quad \text{due to 8.2.5(2)(a)} \\
&\equiv \prod_{i=1}^{n-1} (x_i \rightarrow x_{i+1})
\end{aligned}$$

(1)(b) We choose some $j \in \{1, \dots, n\}$ and obtain

$$\begin{aligned}
&\prod_{i=1}^n x_i \\
&\equiv (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \quad \text{due to (1)(a)} \\
&\equiv \left((x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{i-1} \rightarrow x_i) \sqcap \right. \\
&\quad \left. (x_i \rightarrow x_{i+1}) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \right) \\
&\equiv \left((x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{i-1} \rightarrow x_i) \right) \sqcap \\
&\quad \left((x_i \rightarrow x_{i+1}) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \right) \\
&\quad \text{associativity of } \sqcap \\
&\equiv (x_1 \rightarrow \dots \rightarrow x_i) \sqcap (x_i \rightarrow \dots \rightarrow x_n) \\
&\quad \text{again, (1)(a)} \\
&\equiv \left(\prod_{i=1}^j x_i \right) \sqcap \left(\prod_{i=j}^n x_i \right)
\end{aligned}$$

(2)(a) We proof by induction on n .

♣ If $n = 0$ then

$$\prod_{i=1}^{-1} (x_i \leftrightarrow x_{i+1}) \equiv \top \equiv (\leftrightarrow) \equiv \prod_{i=1}^0 x_i$$

♣ If $n = 1$ then

$$\prod_{i=1}^0 (x_i \leftrightarrow x_{i+1}) \equiv \top \equiv (\leftrightarrow x_1) \equiv \prod_{i=1}^1 x_i$$

♣ If $n \mapsto n+1$ then

$$\begin{aligned}
&\prod_{i=1}^n (x_i \leftrightarrow x_{i+1}) \\
&\equiv \left(\prod_{i=1}^{n-1} (x_i \leftrightarrow x_{i+1}) \right) \sqcap (x_n \leftrightarrow x_{n+1}) \\
&\equiv \left(\prod_{i=1}^{n-1} x_i \right) \sqcap (x_n \leftrightarrow x_{n+1}) \quad \text{induction step}
\end{aligned}$$

$$\begin{aligned}
&\equiv \left(\left(\left(\prod_{i=1}^n \neg x_i \right) \sqcup \left(\prod_{i=1}^n x_i \right) \right) \right) \\
&\equiv \left(\left(\prod_{i=1}^n \neg x_i \sqcap \neg x_n \sqcap \neg x_{n+1} \right) \sqcup \left(\prod_{i=1}^n \neg x_i \sqcap x_n \sqcap x_{n+1} \right) \right) \\
&\equiv \left(\prod_{i=1}^n \neg x_i \sqcap \neg x_n \sqcap \neg x_{n+1} \right) \sqcup \left(\prod_{i=1}^n \neg x_i \sqcap x_n \sqcap x_{n+1} \right) \\
&\equiv \left(\prod_{i=1}^{n+1} \neg x_i \right) \sqcup \perp \sqcup \perp \sqcup \left(\prod_{i=1}^{n+1} x_i \right) \\
&\equiv \left(\prod_{i=1}^{n+1} \neg x_i \right) \sqcup \left(\prod_{i=1}^{n+1} x_i \right) \\
&\equiv \overset{n+1}{\leftrightarrow} x_i \quad \text{def. 8.2.2, again}
\end{aligned}$$

(2)(b) We select an arbitrary $j \in \{1, \dots, n\}$.

If $j = n$ then

$$\overset{n}{\leftrightarrow} x_i \equiv \left(\overset{n}{\leftrightarrow} x_i \right) \sqcap \left(\overset{n}{\leftrightarrow} x_i \right) \equiv \left(\overset{j}{\leftrightarrow} x_i \right) \sqcap \left(\overset{n}{\leftrightarrow} x_i \right)$$

because $\overset{n}{\leftrightarrow} x_i \equiv (\leftrightarrow x_n) \equiv \top$.

Otherwise, $j < n$ and we obtain

$$\begin{aligned}
&\overset{n}{\leftrightarrow} x_i \\
&\equiv \prod_{i=1}^{n-1} (x_i \leftrightarrow x_{i+1}) \quad \text{due to (2)(a)} \\
&\equiv \left(\prod_{i=1}^{j-1} (x_i \leftrightarrow x_{i+1}) \right) \sqcap \left(\prod_{i=j}^{n-1} (x_i \leftrightarrow x_{i+1}) \right) \\
&\equiv \left(\overset{j}{\leftrightarrow} x_i \right) \sqcap \left(\overset{n}{\leftrightarrow} x_i \right) \quad \text{due to (2)(a), again}
\end{aligned}$$

(2)(c) We obtain

$$\begin{aligned}
&\overset{n}{\leftrightarrow} x_i \\
&\equiv \left(\prod_{i=1}^n \neg x_i \right) \sqcup \left(\prod_{j=1}^n x_j \right) \quad \text{def. 8.2.2} \\
&\equiv \prod_{i=1}^n \prod_{j=1}^n (\neg x_i \sqcup x_j) \quad \text{due to 7.6.8(4)} \\
&\equiv \prod_{i=1}^n \prod_{j=1}^n (x_i \rightarrow x_j) \quad \text{due to 8.2.5(2)(a)}
\end{aligned}$$

(2)(d) We have

$$\begin{aligned}
&\overset{n}{\leftrightarrow} x_i \\
&\equiv \prod_{i=1}^n \prod_{j=1}^n (x_i \rightarrow x_j) \quad \text{according to (c)} \\
&\equiv \left(\prod_{i=1}^n \prod_{j=1}^n (x_i \rightarrow x_j) \right) \sqcap \left(\prod_{i=1}^n \prod_{j=1}^n (x_i \rightarrow x_j) \right) \\
&\equiv \left(\prod_{i=1}^n \prod_{j=1}^n (x_i \rightarrow x_j) \right) \sqcap \left(\prod_{i=1}^n \prod_{j=1}^n (x_j \rightarrow x_i) \right) \quad \text{idempotency of } \sqcap \\
&\equiv \prod_{i=1}^n \prod_{j=1}^n ((x_i \rightarrow x_j) \sqcap (x_j \rightarrow x_i)) \\
&\equiv \prod_{i=1}^n \prod_{j=1}^n (x_i \leftrightarrow x_j) \quad \text{due to the associativity and commutativity of } \sqcap \\
&\equiv \prod_{i=1}^n \prod_{j=1}^n (x_i \leftrightarrow x_j) \quad \text{due to 8.2.5(2)(b)}
\end{aligned}$$

(2)(e) First of all, let us note that

$$\begin{aligned}
&\overset{n-1}{\rightarrow} x_{n-i} \\
&\equiv \prod_{i=0}^{n-1} (x_{n-i} \rightarrow x_{n-(i+1)}) \quad \text{due to (1)(a)} \\
&\equiv (x_n \rightarrow x_{n-1}) \sqcap \dots \sqcap (x_3 \rightarrow x_2) \sqcap (x_2 \rightarrow x_1) \\
&\equiv (x_2 \rightarrow x_1) \sqcap \prod (x_3 \rightarrow x_2) \sqcap \dots \sqcap (x_n \rightarrow x_{n-1}) \\
&\equiv \prod_{i=1}^{n-1} (x_{i+1} \rightarrow x_i)
\end{aligned}$$

We apply this in

$$\begin{aligned}
&\overset{n}{\leftrightarrow} x_i \\
&\equiv \prod_{i=1}^{n-1} (x_i \leftrightarrow x_{i+1}) \quad \text{due to (a)} \\
&\equiv \prod_{i=1}^{n-1} ((x_i \rightarrow x_{i+1}) \sqcap (x_{i+1} \rightarrow x_i)) \\
&\equiv \prod_{i=1}^{n-1} ((x_i \rightarrow x_{i+1}) \sqcap (x_{i+1} \rightarrow x_i)) \quad \text{due to 8.2.5(2)(b)}
\end{aligned}$$

$$\begin{aligned}
&\equiv \left(\prod_{i=1}^{n-1} (x_i \rightarrow x_{i+1}) \right) \sqcap \left(\prod_{i=1}^{n-1} (x_{i+1} \rightarrow x_i) \right) \\
&\equiv \left(\overset{n-1}{\rightarrow} x_i \right) \sqcap \left(\overset{n-1}{\rightarrow} x_{n-i} \right)
\end{aligned}$$

due to (1)(a) and the above statement

(2)(f) We derive

$$\begin{aligned}
&\overset{n}{\leftrightarrow} x_i \\
&\equiv \left(\prod_{i=1}^n \neg x_i \right) \sqcup \left(\prod_{j=1}^n x_j \right) \quad \text{def. 8.2.2} \\
&\equiv \neg \left(\prod_{i=1}^n x_i \right) \sqcup \left(\prod_{j=1}^n x_j \right) \quad \text{due to 7.6.10(5)} \\
&\equiv \left(\prod_{i=1}^n x_i \right) \rightarrow \left(\prod_{j=1}^n x_j \right) \quad \text{due to 8.2.5(2)(a)}
\end{aligned}$$

8.2.14 Lemma

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra. For all $x_1, \dots, x_n \in Q$ with $n \in \mathbb{N}$ holds:

- (1) $x_1 \rightarrow \dots \rightarrow x_n \rightarrow \perp \equiv \neg x_1 \sqcap \dots \sqcap \neg x_n$
- (2) $x_1 \rightarrow \dots \rightarrow x_n \rightarrow \top \equiv x_1 \rightarrow \dots \rightarrow x_n$
- (3) $\perp \rightarrow x_1 \rightarrow \dots \rightarrow x_n \equiv x_1 \rightarrow \dots \rightarrow x_n$
- (4) $\top \rightarrow x_1 \rightarrow \dots \rightarrow x_n \equiv x_1 \sqcap \dots \sqcap x_n$

8.2.15 Proof of 8.2.14

First of all, let us establish two auxiliary laws:

- (a) $(x \rightarrow y) \sqcap \neg y \equiv \neg x \sqcap \neg y$
because $(x \rightarrow y) \sqcap \neg y \equiv (\neg x \sqcup y) \sqcap \neg y \equiv (\neg x \sqcap \neg y) \sqcup (y \sqcap \neg y) \equiv (\neg x \sqcap \neg y) \sqcup \perp \equiv \neg x \sqcap \neg y$
- (b) $x \sqcap (x \rightarrow y) \equiv x \sqcap y$
because $x \sqcap (x \rightarrow y) \equiv x \sqcap (\neg x \sqcup y) \equiv (x \sqcap \neg x) \sqcup (x \sqcap y) \equiv \perp \sqcup (x \sqcap y) \equiv x \sqcap y$

are true for all $x, y \in Q$.

(1) If $n = 0$ we use 8.2.5(1)(a) to derive

$$x_1 \rightarrow \dots \rightarrow x_n \rightarrow \perp \equiv (\rightarrow \perp) \equiv \top \equiv \neg x_1 \sqcap \dots \sqcap \neg x_n$$

If $n = 1$ we use 8.2.5(4)(c) to derive

$$x_1 \rightarrow \dots \rightarrow x_n \rightarrow \perp \equiv (x_1 \rightarrow \perp) \equiv \neg x_1 \equiv \neg x_1 \sqcap \dots \sqcap \neg x_n$$

If $n \geq 1$ we apply (a) to derive

$$\begin{aligned}
&x_1 \rightarrow \dots \rightarrow x_n \rightarrow \perp \\
&\equiv (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \sqcap (x_n \rightarrow \perp) \\
&\equiv (x_1 \rightarrow x_2) \sqcap \dots \sqcap \underbrace{(x_{n-1} \rightarrow x_n) \sqcap \neg x_n}_{\equiv \neg x_{n-1} \sqcap \neg x_n} \\
&\equiv (x_1 \rightarrow x_2) \sqcap \dots \sqcap \underbrace{(x_{n-2} \rightarrow x_{n-1}) \sqcap \neg x_{n-1}}_{\equiv \neg x_{n-2} \sqcap \neg x_{n-1}} \sqcap \neg x_n \\
&\vdots \\
&\vdots \\
&\equiv \underbrace{(x_1 \rightarrow x_2) \sqcap \neg x_2}_{\equiv \neg x_1 \sqcap \neg x_2} \sqcap \neg x_3 \sqcap \dots \sqcap \neg x_n \\
&\equiv \neg x_1 \sqcap \neg x_2 \sqcap \neg x_3 \sqcap \dots \sqcap \neg x_n
\end{aligned}$$

(2) We derive

$$\begin{aligned}
&x_1 \rightarrow \dots \rightarrow x_n \rightarrow \top \\
&\equiv (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \sqcap (x_n \rightarrow \top) \\
&\equiv (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \sqcap \top \quad \text{due to 8.2.5(4)(d)} \\
&\equiv (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \\
&\equiv x_1 \rightarrow \dots \rightarrow x_n
\end{aligned}$$

(3) We derive

$$\perp \rightarrow x_1 \rightarrow \dots \rightarrow x_n$$

$$\begin{aligned}
&\equiv (\perp \rightarrow x_1) \sqcap (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \\
&\equiv \top \sqcap (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \quad \text{due to 8.2.5(4)(a)} \\
&\equiv (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \\
&\equiv x_1 \rightarrow \dots \rightarrow x_n
\end{aligned}$$

(4) If $n = 0$ we use 8.2.5(1)(a) to derive

$$\top \rightarrow x_1 \rightarrow \dots \rightarrow x_n \equiv (\top \rightarrow) \equiv \top \equiv x_1 \sqcap \dots \sqcap x_n$$

If $n = 1$ we use 8.2.5(4)(a) to derive

$$\top \rightarrow x_1 \rightarrow \dots \rightarrow x_n \equiv (\top \leftrightarrow x_1) \equiv x_1 \equiv \top \equiv x_1 \sqcap \dots \sqcap x_n$$

If $n \geq 1$ we apply (b) to derive

$$\begin{aligned}
&\top \rightarrow x_1 \rightarrow \dots \rightarrow x_n \\
&\equiv (\top \rightarrow x_1) \sqcap (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \\
&\equiv \underbrace{x_1 \sqcap (x_1 \rightarrow x_2)}_{\equiv x_1 \sqcap x_2} \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \\
&\equiv x_1 \sqcap \underbrace{x_2 \sqcap (x_2 \rightarrow x_3)}_{\equiv x_2 \sqcap x_3} \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \\
&\vdots \\
&\vdots \\
&\equiv x_1 \sqcap \dots \sqcap x_{n-2} \sqcap \underbrace{x_{n-1} \sqcap (x_{n-1} \rightarrow x_n)}_{\equiv x_{n-1} \sqcap x_n} \\
&\equiv x_1 \sqcap \dots \sqcap x_{n-2} \sqcap x_{n-1} \sqcap x_n
\end{aligned}$$

8.2.16 Lemma

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra. For all $x_1, \dots, x_n, y_1, \dots, y_m \in Q$ with $n, m \in \mathbb{N}$ holds:

- (1) $x_1 \rightarrow \dots \rightarrow x_n \rightarrow \perp \rightarrow y_1 \rightarrow \dots \rightarrow y_m$
 $\equiv (\neg x_1 \sqcap \dots \sqcap \neg x_n) \sqcap (y_1 \rightarrow \dots \rightarrow y_m)$
- (2) $x_1 \rightarrow \dots \rightarrow x_n \rightarrow \top \rightarrow y_1 \rightarrow \dots \rightarrow y_m$
 $\equiv (x_1 \rightarrow \dots \rightarrow x_n) \sqcap (y_1 \sqcap \dots \sqcap y_m)$
- (3) $x_1 \leftrightarrow \dots \leftrightarrow x_n \leftrightarrow \perp \leftrightarrow y_1 \leftrightarrow \dots \leftrightarrow y_m$
 $\equiv \neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap \neg y_1 \sqcap \dots \sqcap \neg y_m$
- (4) $x_1 \leftrightarrow \dots \leftrightarrow x_n \leftrightarrow \top \leftrightarrow y_1 \leftrightarrow \dots \leftrightarrow y_m$
 $\equiv x_1 \sqcap \dots \sqcap x_n \sqcap y_1 \sqcap \dots \sqcap y_m$

8.2.17 Proof of 8.2.16

(1) There is

$$\begin{aligned}
&x_1 \rightarrow \dots \rightarrow x_n \rightarrow \perp \rightarrow y_1 \rightarrow \dots \rightarrow y_m \\
&\equiv (x_1 \rightarrow \dots \rightarrow x_n \rightarrow \perp) \sqcap (\perp \rightarrow y_1 \rightarrow \dots \rightarrow y_m) \\
&\quad \text{due to 8.2.12(1)(b)} \\
&\equiv (\neg x_1 \sqcap \dots \sqcap \neg x_n) \sqcap (y_1 \rightarrow \dots \rightarrow y_m) \\
&\quad \text{due to 8.2.14(1) and (3)}
\end{aligned}$$

(2) Similar to (1) we obtain

$$\begin{aligned}
&x_1 \rightarrow \dots \rightarrow x_n \rightarrow \top \rightarrow y_1 \rightarrow \dots \rightarrow y_m \\
&\equiv (x_1 \rightarrow \dots \rightarrow x_n \rightarrow \top) \sqcap (\top \rightarrow y_1 \rightarrow \dots \rightarrow y_m) \\
&\equiv (x_1 \rightarrow \dots \rightarrow x_n) \sqcap (y_1 \sqcap \dots \sqcap y_m) \\
&\quad \text{due to 8.2.14(2) and (4)}
\end{aligned}$$

(3) We first apply definition 8.2.2 and obtain

$$\begin{aligned}
&x_1 \leftrightarrow \dots \leftrightarrow x_n \leftrightarrow \perp \leftrightarrow y_1 \leftrightarrow \dots \leftrightarrow y_m \\
&\equiv \left((\neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap \neg \perp \sqcap \neg y_1 \sqcap \dots \sqcap \neg y_m) \right) \\
&\quad \sqcup (x_1 \sqcap \dots \sqcap x_n \sqcap \perp \sqcap y_1 \sqcap \dots \sqcap y_m) \\
&\equiv (\neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap \top \sqcap \neg y_1 \sqcap \dots \sqcap \neg y_m) \sqcup \perp \\
&\equiv \neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap \neg y_1 \sqcap \dots \sqcap \neg y_m
\end{aligned}$$

(4) Again, we start with definition 8.2.2 and obtain

$$\begin{aligned}
&x_1 \leftrightarrow \dots \leftrightarrow x_n \leftrightarrow \top \leftrightarrow y_1 \leftrightarrow \dots \leftrightarrow y_m \\
&\equiv \left((\neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap \neg \top \sqcap \neg y_1 \sqcap \dots \sqcap \neg y_m) \right) \\
&\quad \sqcup (x_1 \sqcap \dots \sqcap x_n \sqcap \top \sqcap y_1 \sqcap \dots \sqcap y_m) \\
&\equiv \left((\neg x_1 \sqcap \dots \sqcap \neg x_n \sqcap \perp \sqcap \neg y_1 \sqcap \dots \sqcap \neg y_m) \right) \\
&\quad \sqcup (x_1 \sqcap \dots \sqcap x_n \sqcap y_1 \sqcap \dots \sqcap y_m) \\
&\equiv \perp \sqcup (x_1 \sqcap \dots \sqcap x_n \sqcap y_1 \sqcap \dots \sqcap y_m) \\
&\equiv x_1 \sqcap \dots \sqcap x_n \sqcap y_1 \sqcap \dots \sqcap y_m
\end{aligned}$$

8.2.18 Lemma

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra. For all $x_1, \dots, x_n \in Q$ with $n \in \mathbb{N}$ holds:

- (a) $x_1 \rightarrow \dots \rightarrow x_n \equiv \top$ iff $x_1 \sqsubseteq \dots \sqsubseteq x_n$
- (b) $x_1 \leftrightarrow \dots \leftrightarrow x_n \equiv \top$ iff $x_1 \equiv \dots \equiv x_n$

8.2.19 Proof of 8.2.18

(a) We derive

$$\begin{aligned}
&x_1 \rightarrow \dots \rightarrow x_n \equiv \top \\
&\text{iff } (x_1 \rightarrow x_2) \sqcap \dots \sqcap (x_{n-1} \rightarrow x_n) \equiv \top \\
&\quad \text{due to 8.2.12(1)(a)} \\
&\text{iff } x_1 \rightarrow x_2 \equiv \top \text{ and } \dots \text{ and } x_{n-1} \rightarrow x_n \equiv \top \\
&\quad \text{due to 7.6.13(1)} \\
&\text{iff } \neg x_1 \sqcup x_2 \equiv \top \text{ and } \dots \text{ and } \neg x_{n-1} \sqcup x_n \equiv \top \\
&\quad \text{due to 8.2.5(2)(a)} \\
&\text{iff } x_1 \sqsubseteq x_2 \text{ and } \dots \text{ and } x_{n-1} \sqsubseteq x_n \\
&\quad \text{due to 7.6.12(2)} \\
&\text{iff } x_1 \sqsubseteq \dots \sqsubseteq x_n
\end{aligned}$$

(b) We derive

$$\begin{aligned}
&x_1 \leftrightarrow \dots \leftrightarrow x_n \equiv \top \\
&\text{iff } (x_1 \rightarrow \dots \rightarrow x_n) \sqcap (x_n \rightarrow \dots \rightarrow x_1) \equiv \top \\
&\quad \text{due to 8.2.12(2)(e)} \\
&\text{iff } x_1 \rightarrow \dots \rightarrow x_n \equiv \top \text{ and } x_n \rightarrow \dots \rightarrow x_1 \equiv \top \\
&\quad \text{due to 7.6.13(1)} \\
&\text{iff } \left((x_1 \sqsubseteq x_2 \text{ and } \dots \text{ and } x_{n-1} \sqsubseteq x_n) \text{ and } \right) \\
&\quad \left((x_n \sqsubseteq x_{n-1} \text{ and } \dots \text{ and } x_2 \sqsubseteq x_1) \right) \\
&\quad \text{due to (a)} \\
&\text{iff } \left((x_1 \sqsubseteq x_2 \text{ and } x_2 \sqsubseteq x_1) \text{ and } \dots \text{ and } \right) \\
&\quad \left((x_{n-1} \sqsubseteq x_n \text{ and } x_n \sqsubseteq x_{n-1}) \right) \\
&\text{iff } x_1 \equiv x_2 \text{ and } \dots \text{ and } x_{n-1} \equiv x_n \\
&\quad \text{due to 7.6.4(6)} \\
&\text{iff } x_1 \equiv \dots \equiv x_n
\end{aligned}$$

8.2.20 Lemma

Let $\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$ be a quasi-boolean algebra. For all $w, x_1, \dots, x_n, y_1, \dots, y_m, z_1, \dots, z_k \in Q$ holds:

- (a) $x_1 \rightarrow \dots \rightarrow x_2 \rightarrow w \rightarrow y_1 \rightarrow \dots \rightarrow y_m \rightarrow w \rightarrow z_1 \rightarrow \dots \rightarrow z_k \equiv (x_1 \rightarrow \dots \rightarrow x_2 \rightarrow w \rightarrow z_1 \rightarrow \dots \rightarrow z_k) \sqcap (w \leftrightarrow y_1 \leftrightarrow \dots \leftrightarrow y_m)$
- (a) $x_1 \leftrightarrow \dots \leftrightarrow x_2 \leftrightarrow w \leftrightarrow y_1 \leftrightarrow \dots \leftrightarrow y_m \leftrightarrow w \leftrightarrow z_1 \leftrightarrow \dots \leftrightarrow z_k \equiv w \leftrightarrow x_1 \leftrightarrow \dots \leftrightarrow x_2 \leftrightarrow y_1 \leftrightarrow \dots \leftrightarrow y_m \leftrightarrow z_1 \leftrightarrow \dots \leftrightarrow z_k$

Left as exercise.

8.2.21 Proof of 8.2.20 _____

Part IV
Records

9 Records, including tuples and schemas

9.1 Records

9.1.1 Remark Introduction

A **record** assigns values²⁶ to certain **indices** or **attributes**.

Mathematically, we introduce records as surjective functions. The indices are always **identifiers**, i.e. primitive entities like short character strings. (We will talk about identifiers in this sense, but we won't need a formal definition of the identifier concept.) The values on the other hand might be quite complex.

We will agree that a “**P-record**” is a record, where all values are P. For example, a “*function record*” is a record, where each value is a function, in an “*integer record*” every value is an integer etc. Accordingly, we will even use the (awkward) term “*record-record*” to denote higher-order records, where each value is a record itself.

There are two important kinds of record that will be given an own name: class records will be called **schemas**. And records with the indices $1, 2, \dots, n$ are the usual tuples. Often tuples are also called *vectors* or *arrays*, but we will only use the tuple notion.

9.1.2 Definition record

A **record** is a surjective function $\xi : I \rightarrow \Xi$.

The usual representation of such a record is

$$[i \mapsto \xi_i | i \in I] \quad \text{or} \quad [\xi_i | i \in I]$$

where $\xi_i := \xi(i)$, for each $i \in I$. Since ξ is surjective, Ξ is implicitly given with $\Xi = \{\xi_i | i \in I\}$ and both notations contain all the information.

Given such a record $\xi = [\xi_i | i \in I]$ then

- (1) $I = \mathbf{dom}(\xi)$, the **domain** of ξ , is also called the **index** or **attribute class**.
- (2) The cardinality of I is the **dimension** of ξ .
- (3) The ξ_i are called **values**.
- (4) The value class Ξ of ξ is reconstructed from ξ via $\Xi = \{\xi(i) | i \in \mathbf{dom}(\xi)\}$. Taking ξ as a surjective function, Ξ is also called the **codomain** of ξ and given by $\Xi = \mathbf{cod}(\xi)$.

9.1.3 Remark notation

Recall 5.6.2, that the general class expression “ $\{i \in I | \varphi\}$ ” is the common version of the more correct “ $\{i : I | \varphi\}$ ”, since “ $i \in I$ ” is in fact not an element of I as intended, but a true

or false statement. Similarly, “ $[\xi_i | i : I]$ ” would be the formally more appropriate version of our actual definition “ $[\xi_i | i \in I]$ ”. However, having mentioned this convention, we may follow the usual habits.

According to our convention (4.1.3 and 5.2.7) to use more compact two-dimensional versions for (long) linear expressions, we occasionally write

$$\begin{array}{l} \left[\begin{array}{l} i \mapsto \xi_i \\ i \in I \end{array} \right] \quad \text{instead of} \quad [i \mapsto \xi_i | i \in I] \\ \left[\begin{array}{l} \xi_i \\ i \in I \end{array} \right] \quad \text{instead of} \quad [\xi_i | i \in I] \end{array}$$

9.1.4 Definition record class

REC denotes the class of all records.

9.1.5 Definition finite records

A record $\xi : I \rightarrow \Xi$ is **finite** iff it is of finite dimension, i.e. I is finite (and thus Ξ is finite, too, since ξ is a surjective function).

If $I = \{i_1, \dots, i_n\}$ is the domain of such a finite record, we often represent ξ by

$$\left[\begin{array}{l} i_1 \mapsto \xi(i_1) \\ \vdots \\ i_n \mapsto \xi(i_n) \end{array} \right]$$

n is called the **arity** and ξ is said to be n -ary. As usual, 0-ary, 1-ary, 2-ary, etc records are called **nullary**, **unary**, **binary**, etc. Unary records are also called **singular** or **literal**.

9.1.6 Example record

Let us define a finite record by

$$\text{today} := \left[\begin{array}{l} \text{year} \mapsto 2003 \\ \text{month} \mapsto 1 \\ \text{day} \mapsto 17 \\ \text{title} \mapsto \text{friday} \end{array} \right]$$

We can apply **today** as a function and get

$$\begin{array}{ll} \text{today}(\text{year}) = 2003 & \text{today}(\text{month}) = 1 \\ \text{today}(\text{day}) = 17 & \text{today}(\text{title}) = \text{friday} \end{array}$$

Writing **today** in our standard function form would result in a rather awkward expression like

²⁶ In 5 we introduced the title *value* as the general name for mathematical entities, the semantical counterpart of the syntactical *expression*. This metamathematical preliminaries will not be used explicitly in the sequel and *values* will be associated with *values of records* for the remainder of the text.

$$\left[\begin{array}{l} \{\text{year, month, day, title}\} \longrightarrow \{1, 17, 2003, \text{friday}\} \\ i \mapsto \begin{cases} 2003 & \text{if } i = \text{year} \\ 1 & \text{if } i = \text{month} \\ 17 & \text{if } i = \text{day} \\ \text{friday} & \text{if } i = \text{title} \end{cases} \end{array} \right]$$

So obviously, we usually prefer the initial version to represent finite records.

9.1.7 Remark

The order in which we list the single maps in a finite record expression doesn't matter. For example, there is

$$\left[\begin{array}{l} a \mapsto \xi_a \\ b \mapsto \xi_b \end{array} \right] = \left[\begin{array}{l} b \mapsto \xi_b \\ a \mapsto \xi_a \end{array} \right]$$

But if there is some default order on the index class, we will usually list them accordingly.

9.1.8 Remark Classes as records

From a certain point of view, records are very similar to classes. An arbitrary class C is often represented as $C = \{c_i \mid i \in I\}$ or $C = \{c_i \mid i \in I\}$. I is an *index class* and each element of C is addressed by its own unique index which is a kind of name. Often, such a $C = \{c_i \mid i \in I\}$ is called a *family*, in particular if the c_i are classes themselves. The resemblance with the record $[c_i \mid i \in I]$ is obvious, however classes and records are different things with different operations defined on them. But occasionally, we will take one for the other.

Note, that for every class C there is at least one appropriate index class, namely C itself. The *identity function* (5.7.6)

$$\text{id}_C = \left[\begin{array}{l} C \longrightarrow C \\ c \mapsto c \end{array} \right]$$

is a surjective function and thus a record with

$$\text{id}_C = [c \mapsto c \mid c \in C]$$

9.2 Tuples and other special records

9.2.1 Definition tuples as records

For each $n \in \mathbb{N}$, every n -tuple $\langle \xi_1, \dots, \xi_n \rangle$ (with components ξ_1, \dots, ξ_n) can be (re-)defined as a record ξ with $I = \{1, \dots, n\} = \mathbf{n}$ as its index class or domain. In other words,

$$\langle \xi_1, \dots, \xi_n \rangle := \left[\begin{array}{l} 1 \mapsto \xi_1 \\ \vdots \\ \vdots \\ n \mapsto \xi_n \end{array} \right]$$

9.2.2 Definition the empty record / function / tuple

$\langle \rangle$ is the nullary or empty record.

9.2.3 Remark

$\langle \rangle$ was already introduced as the empty tuple (5.4.1) and the empty function (5.7.6). This unique function of type $\emptyset \longrightarrow \emptyset$ is a well-defined function, and it is surjective in a trivial sense. So it is indeed a record and all the titles *empty tuple*, *empty function*, and *empty record* turn out to be different names for the same thing.

9.2.4 Definition univalent record

A record $\xi : I \longrightarrow \Xi$ is univalent, if

$$\xi(i) = \xi(j) \quad \text{for all } i, j \in I$$

So if ξ is not empty, its codomain Ξ is a singleton $\Xi = \{c\}$, for some value c .

When we write such a nonempty univalent record in the " $[\xi_i \mid i \in I]$ " notation, we could as well omit the index i , since it all ξ_i are the same c anyway. Therefore, we often write it as

$$[c \mid I]$$

9.2.5 Example univalent record

A boring menu for an entire week in record form is given by

$$[\text{soup} \mid \{\text{monday}, \dots, \text{sunday}\}] = \left[\begin{array}{l} \text{monday} \mapsto \text{soup} \\ \vdots \\ \vdots \\ \text{sunday} \mapsto \text{soup} \end{array} \right]$$

9.2.6 Definition univalent tuple

For every c and all $n \in \mathbb{N}$ we define an additional simplified notation for the univalent n -tuple with component c , namely

$$\langle c \mid n \rangle := [c \mid \mathbf{n}] = \underbrace{\langle c, c, \dots, c \rangle}_{n \text{ times}} = \left[\begin{array}{l} 1 \mapsto c \\ \vdots \\ \vdots \\ n \mapsto c \end{array} \right]$$

9.2.7 Example univalent tuple

$$\langle -2 \mid 5 \rangle = \langle -2, -2, -2, -2, -2 \rangle$$

9.3 Schemas

9.3.1 Definition schema

A record $X = [X_i \mid i \in I]$ is called a schema, if all the X_i are classes. For each $i \in I$, the class X_i is the i -th domain of X .

9.3.2 Remark

We use "schemas" rather than "schemata" as the plural for "schema".

9.3.3 Definition properties

A schema $X = [X_i | i \in I]$ is called

- (1) proper, if
 $X_i \neq \emptyset$, for all $i \in I$
- (2) finite, if
it is a finite record, i.e. if I is finite
- (3) locally finite or singular finite, if
 X_i is finite, for every $i \in I$
- (4) completely finite, if
it is both finite and locally finite.

9.3.4 Example schema

A proper finite schema is given by

$$\text{Dates} := \left[\begin{array}{l} \text{year} \mapsto \mathbb{Z} \\ \text{month} \mapsto \{1, \dots, 12\} \\ \text{day} \mapsto \{1, \dots, 31\} \\ \text{title} \mapsto \{\text{monday}, \dots, \text{sunday}\} \end{array} \right]$$

For example, the `month`-domain is $\{1, \dots, 12\}$. In other words, $\text{Dates}(\text{month}) = \{1, \dots, 12\}$. `Dates` is not completely finite, because the `year`-domain \mathbb{Z} is infinite.

10 Operations on records

10.1 Values, domains and codomains

10.1.1 Definition the usual function operations on records.

Let $\xi = [\xi_i | i \in I]$ be a record. Since a record is a function, the following notions are well-defined (as in 5.7.3):

$\xi(i)$	$:= \xi_i$	the <u>value</u> of ξ at $i \in I$ or the i -th component of ξ ,
$\text{dom}(\xi)$	$:= I$	the <u>domain</u> of ξ , also called the <u>index</u> or <u>attribute class</u> of ξ ,
$\text{cod}(\xi)$	$:= \{\xi_i i \in I\}$	the <u>codomain</u> or <u>value class</u> of ξ .

10.2 Projections

10.2.1 Definition projection.

For every record $\xi = [\xi_i | i \in I]$ and every class $J \subseteq I$ we define

$$\mathbf{pr}(\xi, J) := [\xi_i | i \in J] \quad \text{the projection of } \xi \text{ onto } J.$$

10.2.2 Example projection.

For

$$\xi = \langle a, b, c, d \rangle = \begin{bmatrix} 1 \mapsto a \\ 2 \mapsto b \\ 3 \mapsto c \\ 4 \mapsto d \end{bmatrix}$$

we have

$$\mathbf{pr}(\xi, \{2, 4\}) = \begin{bmatrix} 2 \mapsto b \\ 4 \mapsto d \end{bmatrix} \quad \text{and} \quad \mathbf{pr}(\xi, \emptyset) = \langle \rangle$$

10.2.3 Remark Remark.

Note the difference between the projection $\mathbf{pr}(\xi, J)$ and the domain restriction $\xi \upharpoonright_J$ (see 5.7.6). $\xi \upharpoonright_J$ is a well-defined function again, but it might not be surjective anymore, i.e. not a record. The projection is a domain as well as a codomain restriction and its result is a record.

10.2.4 Lemma projection.

For every record $\xi = [\xi_i | i \in I]$ and every class $J \subseteq I$

- (1) $\mathbf{pr}(\mathbf{pr}(\xi, J), J) = \mathbf{pr}(\xi, J)$ (idempotency)
- (2) $\mathbf{pr}(\xi, J) = \xi$ iff $J = I$ (neutral projection)

10.2.5 Proof of 10.2.4 Proof of 10.2.4.

Both properties are immediate consequences of definition 10.2.1.

10.2.6 Definition projection class.

For every record ξ we define

$$\mathbf{Proj}(\xi) := \{\mathbf{pr}(\xi, J) | J \subseteq \text{dom}(\xi)\}$$

the projection class of ξ .

10.2.7 Example projection class.

For $\xi = \langle a, b \rangle$ we obtain

$$\begin{aligned} \mathbf{Proj}(\xi) &= \left\{ \langle \rangle, [1 \mapsto a], [2 \mapsto b], \begin{bmatrix} 1 \mapsto a \\ 2 \mapsto b \end{bmatrix} \right\} \\ &= \left\{ \langle \rangle, \langle a \rangle, [2 \mapsto b], \langle a, b \rangle \right\} \end{aligned}$$

10.3 Relations between records

10.3.1 Remark introduction.

Suppose we have two records ξ and v . We introduce the following notions to compare the two: ξ and v are *distinct* iff they have no index in common. They are *compatible* iff there is no common index with a different value. And ξ is *smaller* or a *subrecord* of v iff we obtain ξ by deleting some indices (and their corresponding values) from v , i.e. iff ξ is a projection of v onto the domain of ξ .

10.3.2 Definition

We define

$$\textcircled{1} \not\sim \textcircled{2} := \left[\begin{array}{c} \text{REC} \leftrightarrow \text{REC} \\ \langle \{[\xi_i | i \in I], [v | j \in J]\} \rangle \rightsquigarrow I \cap J = \emptyset \end{array} \right]$$

the distinctness relation

$$\textcircled{1} \smile \textcircled{2} := \left[\begin{array}{c} \text{REC} \leftrightarrow \text{REC} \\ \langle \{[\xi_i | i \in I], [v | j \in J]\} \rangle \rightsquigarrow \\ \xi_k = v_k \text{ for all } k \in I \cap J \end{array} \right]$$

the compatibility relation

$$\textcircled{1} \leq \textcircled{2} := \left[\begin{array}{c} \text{REC} \leftrightarrow \text{REC} \\ \langle \{[\xi_i | i \in I], [v | j \in J]\} \rangle \rightsquigarrow \\ I \subseteq J \text{ and } \xi_i = v_i \text{ for all } i \in I \end{array} \right]$$

the subrecord or smaller relation

$$\textcircled{1} < \textcircled{2} := \left[\begin{array}{c} \text{REC} \leftrightarrow \text{REC} \\ \langle \{[\xi_i | i \in I], [v | j \in J]\} \rangle \rightsquigarrow \\ I \subset J \text{ and } \xi_i = v_i \text{ for all } i \in I \end{array} \right]$$

the proper subrecord or strict smaller relation

10.3.3 Definition notation

For two relations ξ and v we write (see also 19.4.1)

$$\xi \not\sim v \quad \xi \not\smile v \quad \xi \not\leq v \quad \xi \not< v$$

And for two or more records ξ_1, \dots, ξ_n we write

$$\xi_1 \not\sim \dots \not\sim \xi_n \quad \text{iff} \quad \xi_i \not\sim \xi_j \text{ for all } i \text{ and } j \text{ with } i \neq j$$

$$\xi_1 \smile \dots \smile \xi_n \quad \text{iff} \quad \xi_i \smile \xi_j \text{ for all } i \text{ and } j$$

$$\xi_1 \leq \dots \leq \xi_n \quad \text{iff} \quad \xi_1 \leq \xi_2 \text{ and } \dots \text{ and } \xi_{n-1} \leq \xi_n$$

$$\xi_1 < \dots < \xi_n \quad \text{iff} \quad \xi_1 < \xi_2 \text{ and } \dots \text{ and } \xi_{n-1} < \xi_n$$

10.3.4 Definition

A class Ξ of records is called

- (1) (pairwise) distinct
iff $\xi \neq v$ implies $\xi \not\sim v$, for all $\xi, v \in \Xi$

- (2) (pairwise) compatible
iff $\xi \smile v$, for all $\xi, v \in \Xi$

A record-record $\rho = [\rho_k | k \in K]$ is called

- (1) (pairwise) distinct
iff $\{\rho_k | k \in K\}$ is (pairwise) distinct.

- (2) (pairwise) compatible
iff $\{\rho_k | k \in K\}$ is (pairwise) compatible.

10.3.5 Remark

Note, that the definition of $\xi_1 \not\sim \dots \not\sim \xi_n$ and $\xi_1 \smile \dots \smile \xi_n$ in 10.3.3 diverges from definition 5.7.12, where

$$x_1 R x_2 R \dots R x_n \quad \text{means} \quad x_1 R x_2 \wedge x_2 R x_3 \wedge \dots \wedge x_{n-1} R x_n$$

Here, the we have another motivation for the n -ary sequence notation: if $\Xi = \{\xi_1, \dots, \xi_n\}$ is a finite record class, then

$$\xi_1 \not\sim \dots \not\sim \xi_n \quad \text{iff} \quad \{\xi_1, \dots, \xi_n\} \text{ is (pairwise) distinct}$$

$$\xi_1 \smile \dots \smile \xi_n \quad \text{iff} \quad \{\xi_1, \dots, \xi_n\} \text{ is (pairwise) compatible}$$

10.3.6 Example

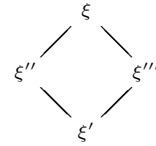
Suppose $a \neq b$ and given six records

$$\xi = \begin{bmatrix} 1 \mapsto a \\ 3 \mapsto a \\ 5 \mapsto a \\ 7 \mapsto a \end{bmatrix} \quad v = \begin{bmatrix} 2 \mapsto b \\ 4 \mapsto b \\ 6 \mapsto b \\ 8 \mapsto b \end{bmatrix} \quad \zeta = \begin{bmatrix} 1 \mapsto a \\ 2 \mapsto b \\ 3 \mapsto b \\ 4 \mapsto b \end{bmatrix}$$

$$\xi' = \langle \rangle \quad \xi'' = \begin{bmatrix} 1 \mapsto a \\ 5 \mapsto a \end{bmatrix} \quad \xi''' = \begin{bmatrix} 3 \mapsto a \\ 7 \mapsto a \end{bmatrix}$$

We can see that:

- (1) $\xi \not\sim v$, because $\text{dom}(\xi) \cap \text{dom}(v) = \emptyset$. Thus $\xi \smile v$. In other words, ξ and v are distinct and compatible.
- (2) ξ and ζ are not distinct, because $\text{dom}(\xi) \cap \text{dom}(\zeta) = \{1, 4\}$. And they are not compatible either, because they disagree in their common index 3, i.e. $\xi(3) = a \neq b = \zeta(3)$.
- (3) v and ζ are not distinct, because $\text{dom}(v) \cap \text{dom}(\zeta) = \{2, 4\}$. But they are compatible, because $v(2) = b = \zeta(2)$ and $v(4) = b = \zeta(4)$.
- (4) ξ' is the empty tuple, and as such it is distinct to and compatible with every other record.
- (5) The class $\{\xi, \xi', \xi'', \xi'''\}$ is not distinct (because e.g. $\xi \not\sim \xi''$), but it is compatible. The \leq -order on this class is represented in the following order diagram:



$$\text{i.e. } \xi' \leq \xi'', \xi' \leq \xi, \text{ etc.}$$

- (6) $\xi' \not\sim \xi \not\sim v$

10.3.7 Remark compatibility notation

The symbol \smile is the so-called “smile”. Records “smile” iff they “agree”. Accordingly, we could have used the “frown” symbol \frown for the opposite case. But we use the $\not\smile$ instead, according to a general convention that denotes the complement of a relation R by \bar{R} .

10.3.8 Remark

We state some obvious facts about the new relations:

- (1) The compatibility relation \smile is reflexive, symmetric, but not transitive (see the following counterexample 10.3.9). So \smile is not an equivalence relation.
- (2) Distinctness implies compatibility, i.e. $\xi \not\sim v$ implies $\xi \smile v$, but not vice versa in general.
- (3) If ξ and v are \leq -comparable, i.e. $\xi \leq v$ or $v \leq \xi$, then

- (a) $\xi \smile v$, and

- (b) $\xi \not\sim v$ iff $(\xi = \langle \rangle \text{ or } v = \langle \rangle)$.

- (4) For two tuples $\xi = \langle \xi_1, \dots, \xi_n \rangle$ and $v = \langle v_1, \dots, v_m \rangle$,

there is

$$\xi \leq v \quad \text{iff} \quad (n \leq m \text{ and } \langle \xi_1, \dots, \xi_n \rangle = \langle v_1, \dots, v_n \rangle)$$

In other words, $\xi \leq v$ iff ξ is an initial section of v .

10.3.9 Example compatibility

The following three records demonstrate, that the compatibility is not transitive in general. For

$$\xi = \begin{bmatrix} 1 \mapsto a \\ 2 \mapsto b \end{bmatrix} \quad v = \begin{bmatrix} 2 \mapsto b \\ 3 \mapsto c \end{bmatrix} \quad \zeta = \begin{bmatrix} 1 \mapsto d \\ 2 \mapsto b \end{bmatrix}$$

there is

$$\xi \smile v \quad \text{and} \quad v \smile \zeta \quad \text{but} \quad \xi \not\smile \zeta$$

10.3.10 Lemma

$\mathbf{Proj}(\xi)$ is compatible, for every record ξ .

10.3.11 Proof of 10.3.10

Let $\xi = [\xi_i | i \in I]$ be a given record, $v = [v_j | j \in J] \in \mathbf{Proj}(\xi)$, and $\zeta = [\zeta_k | k \in K] \in \mathbf{Proj}(\xi)$. For every $l \in J \cap K$ holds: $v_l = \zeta_l = \xi_l$. So $v \smile \zeta$, each two members of $\mathbf{Proj}(\xi)$ are compatible, thus $\mathbf{Proj}(\xi)$ is compatible.

10.3.12 Lemma the new relations in terms of projections

Let $\xi = [\xi_i | i \in I]$ and $v = [v_j | j \in J]$ be two records. Then

- (1) $\xi \check{\vee} v \Leftrightarrow \mathbf{pr}(\xi, I \cap J) = \langle \rangle \Leftrightarrow \mathbf{pr}(v, I \cap J) = \langle \rangle$
- (2) $\xi \smile v \Leftrightarrow \mathbf{pr}(\xi, I \cap J) = \mathbf{pr}(v, I \cap J)$
- (3) $\xi \leq v \Leftrightarrow \xi = \mathbf{pr}(v, I \cap J) \Leftrightarrow \xi \in \mathbf{Proj}(v)$

10.3.13 Proof of 10.3.12

Direct consequences of definitions 10.2.1 and 10.3.2.

10.4 Distinct joins

10.4.1 Definition

For each two distinct records $\xi = [\xi_i | i \in I]$ and $v = [v_j | j \in J]$ we define

$$\xi \check{\vee} v := \left[k \mapsto \begin{cases} \xi_k & \text{if } k \in I \\ v_k & \text{if } k \in J \end{cases} \mid k \in (I \cup J) \right]$$

the distinct join of ξ and v .

A generalization for $n \geq 0$ (pairwise) distinct records is defined by

$$\xi_1 \check{\vee} \dots \check{\vee} \xi_n := \begin{cases} \langle \rangle & \text{if } n = 0 \\ \xi_1 & \text{if } n = 1 \\ (\xi_1 \check{\vee} \xi_2) \check{\vee} \xi_3 \check{\vee} \dots \check{\vee} \xi_n & \text{if } n > 1 \end{cases}$$

And for every (pairwise) distinct record class Ξ we define

$$\check{\vee} \Xi := \left[\frac{k \mapsto \mathbf{the} \xi_k \mathbf{with} \xi \in \Xi \text{ and } k \in \mathbf{dom}(\xi)}{k \in \bigcup \{\mathbf{dom}(\xi) \mid \xi \in \Xi\}} \right]$$

the distinct join of Ξ

10.4.2 Remark

- (1) In case the arguments are finite records, the result of a distinct join is intuitive, simply a merge or combination: For example, given

$$\xi = \begin{bmatrix} p \mapsto \xi_p \\ q \mapsto \xi_q \end{bmatrix} \quad v = \begin{bmatrix} r \mapsto v_r \\ s \mapsto v_s \end{bmatrix}$$

then $\xi \check{\vee} v$ and

$$\xi \check{\vee} v = \begin{bmatrix} p \mapsto \xi_p \\ q \mapsto \xi_q \\ r \mapsto v_r \\ s \mapsto v_s \end{bmatrix}$$

- (2) Later on (11.4.3) we define another generalization: the *join* \vee , that is defined for every two records, not just for distinct ones. But it is very useful to have the distinct version and notation $\check{\vee}$ available, because it allows us to emphasize the distinctness of the arguments. We often apply it in *distinct decompositions* of a given record or *record partitions* (see 16 in particular). Besides, $\check{\vee}$ has some properties that do not hold anymore for \vee , e.g. the associativity (see 11.5.3).
- (3) The distinct join is associative, commutative, and $\langle \rangle$ is its neutral element. That makes the n -ary definition in 10.4.1 natural and obvious.
- (4) The big version $\check{\vee}$ is just a generalization of $\check{\vee}$ for an arbitrary number of records in the obvious sense that

$$\xi \check{\vee} v = \check{\vee} \{\xi, v\}$$

10.5 Concatenation of tuples

10.5.1 Definition concatenation

For all tuples $\xi = \langle \xi_1, \dots, \xi_n \rangle$ and $v = \langle \xi_1, \dots, \xi_m \rangle$ we define

$$\xi \dagger v := \langle \xi_1, \dots, \xi_n, v_1, \dots, v_n \rangle$$

the concatenation of ξ and v .

More general, we define for $n \leq 0$ tuples ξ_1, \dots, ξ_n their concatenation

$$\begin{aligned} \xi_1 \dagger \dots \dagger \xi_n &:= \\ \prod_{i=1}^n \xi_i &:= \begin{cases} \langle \rangle & \text{if } n = 0 \\ \xi_1 \dagger (\xi_2 \dagger \dots \dagger \xi_n) & \text{else} \end{cases} \end{aligned}$$

10.5.2 Example concatenation

$$\langle 1, 3, 5 \rangle \dagger \langle 2, 4, 6 \rangle = \langle 1, 3, 5, 2, 4, 6 \rangle$$

$$\langle 1, 2, 3 \rangle \dagger \langle \rangle = \langle 1, 2, 3 \rangle$$

$$\langle 1, 3, 5 \rangle \dagger \langle \rangle \dagger \langle 2, 4, 6 \rangle \dagger \langle 1, 3, 5 \rangle = \langle 1, 3, 5, 2, 4, 6, 1, 3, 5 \rangle$$

10.5.3 Lemma properties of the concatenation

$\langle X^*, \dagger, \langle \rangle \rangle$ is a monoid, for every class X .

10.5.4 Proof of 10.5.3

By definition, $\langle X^*, \dagger, \langle \rangle \rangle$ is a *monoid*, if the following two properties are satisfied:

(1) \dagger is associative:

$$(\xi \dagger v) \dagger \zeta = \xi \dagger (v \dagger \zeta) \quad \text{for all } \xi, v, \zeta \in X^*$$

(2) $\langle \rangle$ is the neutral element:

$$\langle \rangle \dagger \xi = \xi \dagger \langle \rangle \quad \text{for all } \xi \in X^*$$

And these are quite obvious facts.

10.5.5 Remark

The concatenation looks similar to the distinct join of finite distinct records in 10.4.2. But the concatenation is not simply the distinct join of two tuples. Two tuples are distinct only, if at least one of them is empty. To actually define the concatenation formally as a distinct join, we need an *attribute translation* of the second argument. For example

$$\begin{aligned} \langle a, b, c \rangle \dagger \langle b, b, d \rangle &= \begin{bmatrix} 1 \mapsto a \\ 2 \mapsto b \\ 3 \mapsto c \end{bmatrix} \dagger \begin{bmatrix} 1 \mapsto b \\ 2 \mapsto b \\ 3 \mapsto d \end{bmatrix} \\ &= \begin{bmatrix} 1 \mapsto a \\ 2 \mapsto b \\ 3 \mapsto c \end{bmatrix} \dot{\vee} \begin{bmatrix} 4 \mapsto b \\ 5 \mapsto b \\ 6 \mapsto d \end{bmatrix} = \langle a, b, c, b, b, d \rangle \end{aligned}$$

Attribute translations are also operations on records, but we will postpone their proper introduction until 22.2.3.

11 Order structures and junctions on records

11.1 Poclases of records

11.1.1 Lemma

- (1) $\langle \mathbf{REC}, \leq \rangle$ is a poclass.
- (2) $\langle \Xi, \leq \rangle$ is a poclass, for every record class Ξ .

11.1.2 Proof of 11.1.1

- (1) $\langle \mathbf{REC}, \leq \rangle$ satisfies the following three defining properties of a poclass (7.1.3 and 7.2.2):
 - \leq is transitive on \mathbf{REC} : If $\xi = [\xi_i | i \in I]$, $v = [v_j | j \in J]$, and $\zeta = [\zeta_k | k \in K]$ are three records with $\xi \leq v$ and $v \leq \zeta$, then $I \subseteq J$ and $J \subseteq K$, $\xi_i = v_i$ for all $i \in I$ and $v_j = \zeta_j$ for all $j \in J$, and thus $I \subseteq K$ and $\xi_i = \zeta_i$ for all $i \in I$, which means $\xi \leq \zeta$.
 - \leq is reflexive on \mathbf{REC} : $\xi \leq \xi$ for every record ξ , which is obvious.
 - \leq is antisymmetric on \mathbf{REC} : $\xi \leq v$ and $v \leq \xi$ implies $\xi = v$ for all records ξ and v , and that is correct, too.
- (2) Recall from 7.2.3: If $\langle D, \rho \rangle$ is any given poclass and $C \subseteq D$, then $\langle C, \rho \rangle$ is a poclass, too. So if Ξ is a record class, then $\langle \Xi, \leq \rangle$ is a poclass.

11.2 Lattices on records

11.2.1 Remark [overview](#)

We will now investigate, in how far these poclasses $\langle \Xi, \leq \rangle$ are lattice-like structures. In other words, we try to define the according operations (called *junctions*) such as the *meet* $\xi \wedge v$ and *join* $\xi \vee v$. We define $\mathbf{Rec}(I, C)$, which is a more specific type than the general \mathbf{REC} , but has the same structural properties and is useful for illustrations (see example 11.2.3 below).

We will see soon, that neither $\langle \mathbf{REC}, \leq \rangle$ nor $\langle \mathbf{Rec}(I, C), \leq \rangle$ is a lattice.²⁷ A key concept here is *compatibility*: For example, $\xi \vee v$ is the least upper bound of ξ and v iff $\xi \smile v$, i.e. iff ξ and v are compatible. So in order to have the full range of lattice operations available, we later concentrate on record classes Ξ , which are compatible and operationally closed. Such classes are the projection classes $\mathbf{Proj}(\xi)$. For every record ξ , the poclass $\langle \mathbf{Proj}(\xi), \leq \rangle$ is indeed a lattice, even a complete boolean one.

11.2.2 Definition

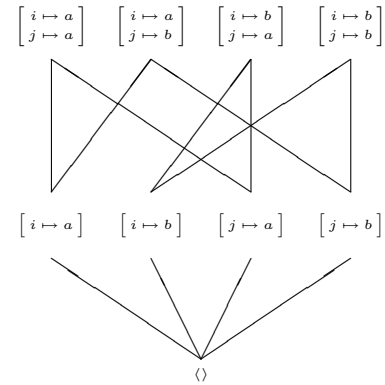
For every two class I and C we define

$$\mathbf{Rec}(I, C) := \{ \xi \in \mathbf{REC} \mid \text{dom}(\xi) \subseteq I, \text{cod}(\xi) \subseteq C \}$$

the record class on I and C .

11.2.3 Example

Let us take $I = \{i, j\}$ and $C = \{a, b\}$. The whole structure of $\mathbf{Rec}(I, C)$ is represented by the following order diagram:



The diagram of this simple example class is typical for all classes $\mathbf{Rec}(I, C)$ and \mathbf{REC} itself:

- (1) There is no *greatest* or *top* element.
- (2) But there is a *least* or *bottom* element, namely $\langle \rangle$.
- (3) Every two records ξ and v do have common *lower bounds*, i.e. there are records ζ such with $\zeta \leq \xi$ and $\zeta \leq v$. And of all these lower bounds ζ , there is a unique *greatest lower bound* or *meet*, which is written $\xi \wedge v$. For example,

$$\begin{aligned} \begin{bmatrix} i \mapsto a \\ j \mapsto a \end{bmatrix} \wedge \begin{bmatrix} i \mapsto a \\ j \mapsto b \end{bmatrix} &= \begin{bmatrix} i \mapsto a \end{bmatrix} \\ \begin{bmatrix} i \mapsto a \\ j \mapsto a \end{bmatrix} \wedge \begin{bmatrix} i \mapsto b \\ j \mapsto b \end{bmatrix} &= \langle \rangle \end{aligned}$$

- (4) Not every two records ξ and v have common *upper bounds*, let alone a *least upper bound*. For example, take $[i \mapsto a]$ and $[i \mapsto b]$. But they have a least upper bound or *join* $\xi \vee v$ iff they are compatible. For example

$$\begin{aligned} \begin{bmatrix} i \mapsto a \end{bmatrix} \vee \begin{bmatrix} j \mapsto b \end{bmatrix} &= \begin{bmatrix} i \mapsto a \\ j \mapsto b \end{bmatrix} \\ \begin{bmatrix} i \mapsto a \\ j \mapsto a \end{bmatrix} \vee \begin{bmatrix} i \mapsto a \end{bmatrix} &= \begin{bmatrix} i \mapsto a \\ j \mapsto a \end{bmatrix} \end{aligned}$$

²⁷ $\langle \mathbf{REC}, \leq \rangle$ is a so-called complete partial order or cpo (i.e. has bottom element and each chain has a supremum) and a meet semi-lattice (i.e. the greatest lower bound is always defined). But we will not use this or any other terminology for this kind of poclass.

So the *join* operator or *disjuncter* is just a *partial function*, in our notation $\textcircled{1} \vee \textcircled{2} : \mathbf{REC} \times \mathbf{REC} \dashrightarrow \mathbf{REC}$.

However, we enforce a join $\xi \vee v$ for every two records: if they are incompatible, we simply delete the incompatible indices first. For example:

$$\begin{aligned} [i \mapsto a] \vee [i \mapsto b] &:= \langle \rangle \\ \begin{bmatrix} i \mapsto a \\ j \mapsto a \end{bmatrix} \vee \begin{bmatrix} i \mapsto a \\ j \mapsto b \end{bmatrix} &:= [i \mapsto a] \end{aligned}$$

But these joins of incompatible arguments are not (least) upper bounds of their arguments.

So the structure $\langle \mathbf{Rec}(I, C), \leq, \langle \rangle, \wedge, \vee \rangle$ is well-defined, with \wedge and \vee defined for all its record pairs. But it is just not a proper lattice.

11.3 Various domains of record classes

11.3.1 Definition

For every record class Ξ we define

$$\mathbf{D}_\Xi := \bigcup_{\xi \in \Xi} \mathbf{dom}(\xi) \quad \text{the (total) domain,}$$

$$\mathbf{D}_\Xi^\neq := \left\{ \begin{array}{l} k \in \mathbf{D}_\Xi \\ \exists v, \zeta \in \Xi. \\ v(k) \neq \zeta(k) \end{array} \right\} \quad \text{the incompatible domain,}$$

$$\mathbf{D}_\Xi^\sim := \mathbf{D}_\Xi \setminus \mathbf{D}_\Xi^\neq \quad \text{the compatible domain of } \Xi.$$

11.3.2 Remark notation

In the sequel and when there is only one record class Ξ involved in the given context, we sometimes leave the subscript for the new domain symbols and simply write \mathbf{D} , \mathbf{D}^\neq , and \mathbf{D}^\sim instead.

11.3.3 Example

Given two records

$$\xi = \begin{bmatrix} i \mapsto 1 \\ j \mapsto 2 \\ k \mapsto 3 \end{bmatrix} \quad \text{and} \quad v = \begin{bmatrix} j \mapsto 5 \\ k \mapsto 3 \\ l \mapsto 4 \end{bmatrix}$$

so

$$\mathbf{dom}(\xi) = \{i, j, k\} \quad \text{and} \quad \mathbf{dom}(v) = \{j, k, l\}$$

We obtain

$$\mathbf{D} = \{i, j, k, l\} \quad \mathbf{D}^\neq = \{j\} \quad \mathbf{D}^\sim = \{i, k, l\}$$

(So here \mathbf{D} is an abbreviation for $\mathbf{D}_{\{\xi, v\}}$ etc.)

11.3.4 Lemma

Let Ξ be a record class. Then

- (1) $\mathbf{D}_\Xi = \mathbf{D}_\Xi^\neq \cup \mathbf{D}_\Xi^\sim$ and $\mathbf{D}_\Xi^\neq \cap \mathbf{D}_\Xi^\sim = \emptyset$
 - (2) Ξ is (pairwise) compatible iff $\mathbf{D}_\Xi^\neq = \emptyset$
 - (3) $\mathbf{D}_\Xi^\neq = \bigcup_{\xi, v \in \Xi} \mathbf{D}_{\{\xi, v\}}^\neq$
 - (4) $\mathbf{D}_\Xi^\neq = \bigcup_{\Psi \subseteq \Xi} \mathbf{D}_\Psi^\neq$
 - (5) If $\Psi \subseteq \Xi$ then $\mathbf{D}_\Psi \subseteq \mathbf{D}_\Xi$ and $\mathbf{D}_\Psi^\neq \subseteq \mathbf{D}_\Xi^\neq$.
But $\mathbf{D}_\Psi^\sim \subseteq \mathbf{D}_\Xi^\sim$ is not true in general.
- And if ξ, v, ζ are records, then:
- (6) $\mathbf{D}_\emptyset = \mathbf{D}_\emptyset^\neq = \mathbf{D}_\emptyset^\sim = \emptyset$
 - (7) $\mathbf{D}_{\{\xi\}} = \mathbf{D}_{\{\xi\}}^\sim = \mathbf{dom}(\xi)$ and $\mathbf{D}_{\{\xi\}}^\neq = \emptyset$
 - (8) $\mathbf{D}_{\{\xi, v\}}^\sim = \mathbf{dom}(\xi \vee v)$
 - (9) $\mathbf{D}_{\{\xi, v, \zeta\}}^\neq = \mathbf{D}_{\{\xi, v\}}^\neq \cup \mathbf{D}_{\{\xi \vee v, \zeta\}}^\neq$

11.3.5 Proof of 11.3.4

All statements are more or less trivial consequences of definition 11.3.2.

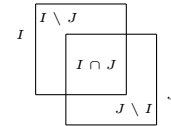
On the remark in (5) about the truth of “ $\Psi \subseteq \Xi$ implies $\mathbf{D}_\Psi^\sim \subseteq \mathbf{D}_\Xi^\sim$ ” consider the following counterexample: For $\Psi = \{\langle 7, 8, 9 \rangle, \langle 7, 8, 6 \rangle\} \subseteq \{\langle 7, 8, 9 \rangle, \langle 7, 8, 6 \rangle, \langle 9, 8, 7 \rangle\} = \Xi$ holds $\mathbf{D}_\Psi^\sim = \{1, 2\} \not\subseteq \{2\} = \mathbf{D}_\Xi^\sim$.

11.4 Record junctions

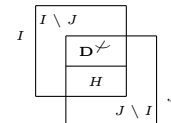
11.4.1 Remark introduction of the new junctors

The example 11.3.3 of two simple incompatible records is worth investigating, because it motivates the following binary junctor definitions.

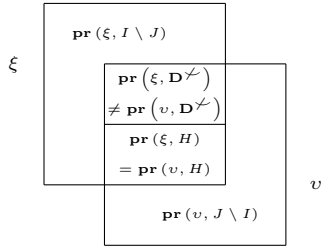
Let $\xi = [\xi_i | i \in I]$ and $v = [v_j | j \in J]$ be two records. Their (total) domain $\mathbf{D} = I \cup J$ partitions into the following parts:



But $I \cap J$ itself partitions into \mathbf{D}^\neq and the remaining class $H := (I \cap J) \setminus \mathbf{D}^\neq = I \cap J \cap \mathbf{D}^\sim$. The situation is illustrated by



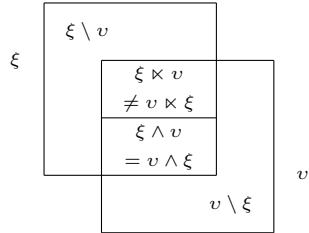
This partition of the total domain induces the following partition of records, induced by the two given ones:



The first three new junctors \setminus , \times and \wedge are just new notations for these projections; we put

$$\begin{aligned} \xi \setminus v &:= \mathbf{pr}(\xi, I \setminus J) \\ \xi \times v &:= \mathbf{pr}(\xi, \mathbf{D}^c) \\ \xi \wedge v &:= \mathbf{pr}(\xi, H) \end{aligned}$$

The corresponding diagram with the new notation is



The other three junctors ∇ , \vee and \surd are different distinct joins of the previous ones:

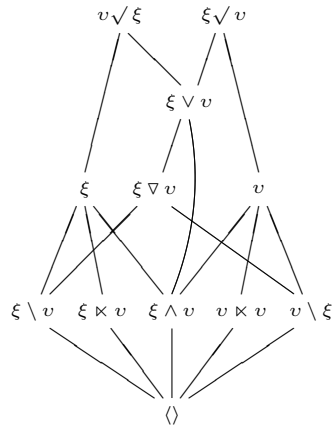
$$\begin{aligned} \xi \nabla v &:= (\xi \setminus v) \dot{\vee} (v \setminus \xi) \\ \xi \vee v &:= (\xi \setminus v) \dot{\vee} (v \setminus \xi) \dot{\vee} (\xi \wedge v) \\ \xi \surd v &:= (\xi \setminus v) \dot{\vee} (v \setminus \xi) \dot{\vee} (\xi \wedge v) \dot{\vee} (v \times \xi) \end{aligned}$$

so that

$$\begin{aligned} v \nabla \xi &= \xi \nabla v \\ v \vee \xi &= \xi \vee v \end{aligned}$$

$$v \surd \xi = (\xi \vee v) \dot{\vee} (v \times \xi) \neq (\xi \vee v) \dot{\vee} (\xi \times v) = \xi \surd v$$

The closure of ξ and v , i.e. the class of all the results produced by these new junctors is displayed by the following diagram



The diagram shows the most general situation. In special cases, e.g. $\xi \smile v$ or $\xi \leq v$, certain members of the diagram collapse into one (see 11.4.5).

This closure of ξ and v is their closure indeed in the usual sense:

- The three records $\xi, v, \langle \rangle$ are junction results as well because

$$\begin{aligned} \xi &= \xi \wedge \xi = \xi \vee \xi = \xi \surd \xi \\ \langle \rangle &= \xi \times \xi = \xi \nabla \xi \end{aligned}$$

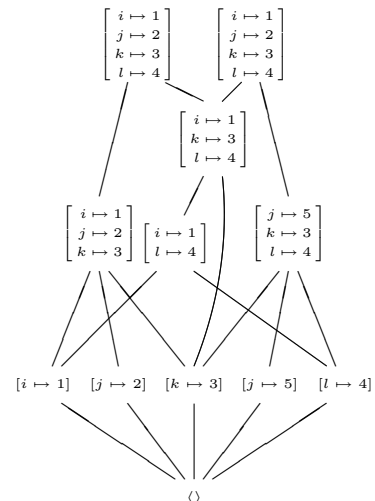
- Applying the junctions again (and again) does not produce new results.

11.4.2 Example

As an example, let us take the incompatible two records

$$\xi = \begin{bmatrix} i \mapsto 1 \\ j \mapsto 2 \\ k \mapsto 3 \end{bmatrix} \quad \text{and} \quad v = \begin{bmatrix} j \mapsto 5 \\ k \mapsto 3 \\ l \mapsto 4 \end{bmatrix}$$

from example 11.3.3 again. Their closure (where each record position in the picture corresponds to the record positions of the diagram in 11.4.1) is given by



11.4.3 Definition

We define

$$\textcircled{1} \setminus \textcircled{2} := \left[\begin{array}{c} \mathbf{REC} \times \mathbf{REC} \longrightarrow \mathbf{REC} \\ \langle \xi, v \rangle \mapsto \mathbf{pr}(\xi, \mathbf{dom}(\xi) \setminus \mathbf{dom}(v)) \end{array} \right]$$

the subtraction, reading “ $\textcircled{1}$ without $\textcircled{2}$ ”

$$\textcircled{1} \wedge \textcircled{2} := \left[\begin{array}{c} \mathbf{REC} \times \mathbf{REC} \longrightarrow \mathbf{REC} \\ \langle \xi, v \rangle \mapsto \mathbf{pr}(\xi, (\mathbf{dom}(\xi) \cap \mathbf{dom}(v)) \setminus \mathbf{D}_{\{\xi, v\}}^\neq) \end{array} \right]$$

the meet, reading “ $\textcircled{1}$ and $\textcircled{2}$ ”

$$\textcircled{1} \times \textcircled{2} := \left[\begin{array}{c} \mathbf{REC} \times \mathbf{REC} \longrightarrow \mathbf{REC} \\ \langle \xi, v \rangle \mapsto \mathbf{pr}(\xi, \mathbf{D}_{\{\xi, v\}}^\neq) \end{array} \right]$$

the rejector, reading “(the part of) $\textcircled{1}$ rejected by $\textcircled{2}$ ”

$$\textcircled{1} \nabla \textcircled{2} := \left[\begin{array}{c} \mathbf{REC} \times \mathbf{REC} \longrightarrow \mathbf{REC} \\ \langle \xi, v \rangle \mapsto (\xi \setminus v) \dot{\vee} (v \setminus \xi) \end{array} \right]$$

the opposition, reading “ $\textcircled{1}$ opposed to $\textcircled{2}$ ”

$$\textcircled{1} \vee \textcircled{2} := \left[\begin{array}{c} \mathbf{REC} \times \mathbf{REC} \longrightarrow \mathbf{REC} \\ \langle \xi, v \rangle \mapsto (\xi \wedge v) \dot{\vee} (\xi \nabla v) \end{array} \right]$$

the join, reading “ $\textcircled{1}$ or $\textcircled{2}$ ”

$$\textcircled{1} \checkmark \textcircled{2} := \left[\begin{array}{c} \mathbf{REC} \times \mathbf{REC} \longrightarrow \mathbf{REC} \\ \langle \xi, v \rangle \mapsto (\xi \vee v) \vee v \end{array} \right]$$

the updater, reading “ $\textcircled{1}$ updated with $\textcircled{2}$ ”

11.4.4 Remark

This definition of six junctors is a lot of notation and symbolism to digest. There doesn't seem to be a standard notion of an *algebra of records* or these kind of operations and our suggestions here don't pretend to be a final solution. But they should become plausible and easy to memorize if one understands the following phenomena:

- (1) Our idiosyncratic creations of the *updater* \checkmark and *rejector* \times are motivated by the most common circumstances where records are involved in mathematics and computer science. An *environment* (or *context*) ξ is a record that holds all the current *bindings*, i.e. values for identifiers. And if at that point some new (local or global) definitions v are made, then the *update* $\xi \checkmark v$ is the new environment, while the *rejection* $\xi \times v$ holds all the deleted bindings.

For example, let $\xi = \left[\begin{array}{l} c \mapsto 23 \\ x \mapsto 5 \end{array} \right]$ be the current environment.

When the command sequence “ $x:=7$; $y:=c+2$,” is called, ξ needs to be updated by $v = \left[\begin{array}{l} x \mapsto 7 \\ y \mapsto 25 \end{array} \right]$, the new environ-

ment is $\xi \checkmark v = \left[\begin{array}{l} c \mapsto 23 \\ x \mapsto 7 \\ y \mapsto 25 \end{array} \right]$ and the lost bindings are given

by $\xi \times v = \left[\begin{array}{l} x \mapsto 5 \end{array} \right]$.

- (2) The other four junctors $\wedge, \vee, \setminus, \nabla$ resemble the class operations with the corresponding symbol, respectively. Each of these junctors is produced by the same method: Given two records $\xi = [\xi_i | i \in I]$ and $v = [v_j | j \in J]$, then first determine their incompatible indices \mathbf{D}^\neq and second take \mathbf{D}^\neq off the new domain. In detail that is

junction	its domain
$\xi \vee v$	$(I \cup J) \setminus \mathbf{D}^\neq$
$\xi \wedge v$	$(I \cap J) \setminus \mathbf{D}^\neq$
$\xi \setminus v$	$(I \setminus J) \setminus \mathbf{D}^\neq = I \setminus J$
$\xi \nabla v$	$(I \nabla J) \setminus \mathbf{D}^\neq = I \nabla J$

This resemblance between the record junction and the class operation becomes even more apparent in case $\xi \smile v$, that is $\mathbf{D}^\neq = \emptyset$.

- (3) If $\xi \smile v$, then \checkmark and \times become superfluous in the sense that $\xi \checkmark v = \xi \vee v$ and $\xi \times v = \langle \rangle$. The closure then is a proper boolean lattice (see 11.4.5 below), a fact that holds more general for every compatible record class Ξ .
- (4) The six chosen symbols reflect another property: the three *symmetric* symbols “ \wedge ”, “ \vee ” and “ ∇ ” denote *commutative* junctors. The remaining *asymmetric* symbols “ \setminus ”, “ \checkmark ” and “ \times ” represent *non-commutative* junctors.

11.4.5 Remark closure of compatible records

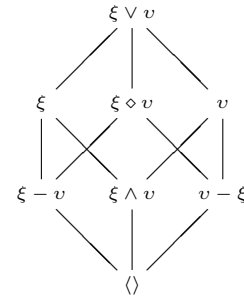
Let ξ and v be two records which are compatible. In that case, certain elements of the diagram in 11.4.1 reduce to a single one, namely

$$v \checkmark \xi = \xi \checkmark v = \xi \vee v$$

and

$$\xi \times v = v \times \xi = \langle \rangle$$

That makes the junctors \checkmark and \times superfluous, the remaining closure is entirely given by the diagram



11.4.6 Example

Two compatible records ξ and v are given by

$$\xi = \left[\begin{array}{l} i \mapsto 1 \\ j \mapsto 2 \\ k \mapsto 3 \end{array} \right] \quad \text{and} \quad v = \left[\begin{array}{l} j \mapsto 2 \\ k \mapsto 3 \\ l \mapsto 4 \end{array} \right]$$

(i) \Leftrightarrow (vi) is true according to 11.3.4(2).

(vi) \Leftrightarrow (iv) is true because $\xi \times v = \mathbf{pr}(\xi, \mathbf{D}_{\{\xi, v\}}^{\neq})$ according to definition 11.4.3, so that $\xi \times v = \langle \rangle$ iff $\mathbf{D}_{\{\xi, v\}}^{\neq} = \emptyset$.

(iv) \Leftrightarrow (v) is true because $\xi \vee v = (\xi \vee v) \dot{\vee} (\xi \times v)$ due to (1)(e) and (1)(d). Thus $\xi \vee v = \xi \vee v$ iff $\xi \times v = \langle \rangle$.

(ii) \Leftrightarrow (iv) is true because we have the distinct decompositions (1)(b) and (d)

$$\xi = (\xi \setminus v) \dot{\vee} (\xi \wedge v) \dot{\vee} (\xi \times v)$$

$$\xi \vee v = (\xi \setminus v) \dot{\vee} (\xi \wedge v) \dot{\vee} (v \setminus \xi)$$

so that $\xi \leq \xi \vee v$ iff $\xi \times v \leq v \setminus \xi$, due to (2) and (2)(h) in particular. But $\mathbf{dom}(\xi \times v) \cap \mathbf{dom}(v \setminus \xi) = \emptyset$, so that $\xi \times v \leq v \setminus \xi$ iff $\xi \times v = \langle \rangle$.

(iii) \Leftrightarrow (vi) Assume that (iii) ξ and v have an upper bound, i.e. there is a record $\beta = [\beta_i | l \in L]$ with $\xi \leq \beta$ and $v \leq \beta$. According to 10.3.12 that means $\xi = \mathbf{pr}(\beta, I)$ and $v = \mathbf{pr}(\beta, J)$. That entails $\mathbf{pr}(\xi, I \cap J) = \mathbf{pr}(\beta, I \cap J) = \mathbf{pr}(v, I \cap J)$ and (vi) $\mathbf{D}_{\{\xi, v\}}^{\neq} = \emptyset$.

On the other hand, if (vi) $\mathbf{D}_{\{\xi, v\}}^{\neq} = \emptyset$ is the case, then $\xi \times v = v \times \xi = \langle \rangle$ and so we have the distinct decompositions

$$\xi = (\xi \setminus v) \dot{\vee} (\xi \wedge v)$$

$$v = (v \setminus \xi) \dot{\vee} (\xi \wedge v)$$

$$\xi \vee v = (\xi \setminus v) \dot{\vee} (v \setminus \xi) \dot{\vee} (\xi \wedge v)$$

so that $\xi \leq \xi \vee v$ and $v \leq \xi \vee v$ according to (2)(h). And that is precisely statement (iii).

(ii) \Leftrightarrow (vii) $\xi \leq \xi \vee v \Leftrightarrow \xi = \mathbf{pr}(\xi \vee v, I) \Leftrightarrow \xi \in \mathbf{Proj}(\xi \vee v)$ and the same holds for v .

(vii) \Leftrightarrow (viii) That (vii) entails (viii) is a trivial truth; simply put $\eta := \xi \vee v$. On the other hand, $\xi, v \in \mathbf{Proj}(\eta) \Rightarrow \xi \vee v \in \mathbf{Proj}(\eta) \Rightarrow \xi \vee v = \mathbf{pr}(\eta, I \cup J)$ and so $\xi, v \in \mathbf{Proj}(\xi \vee v)$.

(7) Order properties: Each of the statements (a) to (f) can be prove by applying the additivity rule (2)(h) to the distinct decompositions of each side of the \leq -expression.

For example, for (e) this is

$$\begin{aligned} \xi \nabla v &= (\xi \setminus v) \dot{\vee} (v \setminus \xi) && \text{due to (1)(c)} \\ &\leq (\xi \setminus v) \dot{\vee} (v \setminus \xi) \dot{\vee} (\xi \wedge v) && \text{due to (2)(h)} \\ &= \xi \vee v && \text{due to (1)(d)} \end{aligned}$$

Proving the other statements with the same method is just a trivial exercise.

(8) Extrema:

(a) According to definition 10.3.2, $\langle \rangle \leq \xi \Leftrightarrow (\emptyset \subseteq I \text{ and } \langle \rangle(i) = \xi_i \text{ for all } i \in \emptyset)$. And that is always true in a trivial sense.

(b) According to (7)(a), $\xi \wedge v \leq \xi$ and $\xi \wedge v \leq v$, i.e. $\xi \wedge v$ is a lower bound of both ξ and v .

Suppose, there is another lower bound $\beta = [\beta_i | l \in L]$ of ξ and v with $\xi \wedge v \leq \beta$. $\beta \leq \xi$ and $\beta \leq v$ implies $L \subseteq I$ and $L \subseteq J$, in other words $L \subseteq I \cap J$. $\xi \wedge v \leq \beta$ implies $(I \cap J) \setminus \mathbf{D}_{\{\xi, v\}}^{\neq}$. But if there would be an $k \in \mathbf{D}_{\{\xi, v\}}^{\neq}$ with $k \in L$, then either $\beta \not\leq \xi$ or $\beta \not\leq v$ would be the case and β couldn't be the lower bound. Therefore L has to be $I \cap J$ and β is $\xi \wedge v$. $\xi \wedge v$ is indeed the greatest of all lower bounds.

(c) From (6)(1) and (6)(3) we know that $\xi \smile v$ iff ξ and v do have an upper bound at all. So if (1) $\xi \vee v$ is the least upper bound, it is certainly an upper bound and (2) $\xi \smile v$ must hold.

On the other hand, suppose that (2) $\xi \smile v$ is true. Then $\xi \leq \xi \vee v$ and $v \leq \xi \vee v$ due to (6), $\xi \vee v$ is an upper bound. And it is the least of all upper bounds, because if we assume that there is another upper bound $\beta = [\beta_i | l \in L]$ with $\beta \leq \xi \vee v$, then

♣ $\xi \leq \beta$ implies $I \subseteq L$, $v \leq \beta$ implies $J \subseteq L$, and thus

$$I \cup J \subseteq L.$$

♣ $\beta \leq \xi \vee v$ implies $L \subseteq \mathbf{dom}(\xi \vee v) = I \cup J$

Both properties together demand that $L = \mathbf{dom}(\xi \vee v)$, and that means $\beta = \xi \vee v$ in the end. $\xi \vee v$ is indeed the least of all upper bounds.

(9) Associativity

(a) We first concentrate on the domains of the generated records and obtain

$$\begin{aligned} &\mathbf{dom}((\xi \wedge v) \wedge \zeta) \\ &= (\mathbf{dom}(\xi \wedge v) \cap K) \setminus \mathbf{D}_{\{\xi \wedge v, \zeta\}}^{\neq} && \text{def. 11.4.3 of } \wedge \\ &= \left((I \cap J) \setminus \mathbf{D}_{\{\xi, v\}}^{\neq} \right) \cap K \setminus \mathbf{D}_{\{\xi \wedge v, \zeta\}}^{\neq} && \text{again due to 11.4.3} \\ &= \left((I \cap J \cap K) \setminus \mathbf{D}_{\{\xi, v\}}^{\neq} \right) \setminus \mathbf{D}_{\{\xi \wedge v, \zeta\}}^{\neq} && \text{due to 5.6.5(7), i.e. } (A \wedge B) \cap C = (A \cap C) \wedge B \\ &= \{j \in (I \cap J \cap K) \mid \xi_j = v_j\} \setminus \mathbf{D}_{\{\xi \wedge v, \zeta\}}^{\neq} \\ &= \{j \in (I \cap J \cap K) \mid \xi_j = v_j \text{ and } (\xi \wedge v)(j) = \zeta_j\} \\ &= \{j \in (I \cap J \cap K) \mid \xi_j = v_j = \zeta_j\} \\ &= (I \cap J \cap K) \setminus \mathbf{D}_{\{\xi, v, \zeta\}}^{\neq} \end{aligned}$$

Similar reasoning gives us

$$\mathbf{dom}(\xi \wedge (v \wedge \zeta)) = (I \cap J \cap K) \setminus \mathbf{D}_{\{\xi, v, \zeta\}}^{\neq}$$

so that

$$\mathbf{dom}((\xi \wedge v) \wedge \zeta) = \mathbf{dom}(\xi \wedge (v \wedge \zeta))$$

Making use of (3)(a) the commutativity of \wedge and 11.4.3 the definition of \wedge , we can now derive the original statement

$$\begin{aligned} (\xi \wedge v) \wedge \zeta &= [v_j | j \in \mathbf{dom}(v \wedge \xi)] \wedge \zeta \\ &= [v_j | j \in \mathbf{dom}((v \wedge \xi) \wedge \zeta)] \\ &= [v_j | j \in \mathbf{dom}((v \wedge \zeta) \wedge \xi)] \\ &= [v_j | j \in \mathbf{dom}(v \wedge \zeta)] \wedge \xi \\ &= (v \wedge \zeta) \wedge \xi \\ &= \xi \wedge (v \wedge \zeta) \end{aligned}$$

(b) See the proof of 11.6.1(4)(b).

(c) See the proof of 11.6.1(4)(c).

(10) Distributivity:

(a) If $v \smile \zeta$ then

$$\begin{aligned} &\xi \wedge (v \vee \zeta) \\ &= \xi \wedge ((v \setminus \zeta) \dot{\vee} (v \wedge \zeta) \dot{\vee} (\xi \wedge v)) && \text{due to (1)(d)} \\ &= (\xi \wedge (v \setminus \zeta)) \dot{\vee} (\xi \wedge (v \wedge \zeta)) \dot{\vee} (\xi \wedge (\xi \wedge v)) && \text{due to (2)(d) and (2)(g)} \\ &= (\xi \wedge (v \setminus \zeta)) \vee (\xi \wedge (v \wedge \zeta)) \vee (\xi \wedge (\xi \wedge v)) && \text{due to (2)(b) and (9)(b)} \\ &= (\xi \wedge (v \setminus \zeta)) \vee \left(\begin{array}{c} (\xi \wedge (v \wedge \zeta)) \\ \vee \\ (\xi \wedge (v \wedge \zeta)) \end{array} \right) \vee (\xi \wedge (\xi \wedge v)) && \text{due to (4)(e)} \\ &= \left(\begin{array}{c} (\xi \wedge (v \setminus \zeta)) \\ \vee \\ (\xi \wedge (v \wedge \zeta)) \end{array} \right) \vee \left(\begin{array}{c} (\xi \wedge (v \wedge \zeta)) \\ \vee \\ (\xi \wedge (\xi \wedge v)) \end{array} \right) && \text{due to (9)(b)} \\ &= \left(\begin{array}{c} (\xi \wedge (v \setminus \zeta)) \\ \dot{\vee} \\ (\xi \wedge (v \wedge \zeta)) \end{array} \right) \vee \left(\begin{array}{c} (\xi \wedge (v \wedge \zeta)) \\ \dot{\vee} \\ (\xi \wedge (\xi \wedge v)) \end{array} \right) && \text{due to (2)(b)} \\ &= (\xi \wedge ((v \setminus \zeta) \dot{\vee} (v \wedge \zeta))) \vee (\xi \wedge ((v \wedge \zeta) \dot{\vee} (\xi \wedge v))) \end{aligned}$$

$$= (\xi \wedge v) \vee (\xi \wedge \zeta)$$

due to (2)(d)

because $v \smile \zeta$ makes $v \times \zeta = \zeta \times v = \langle \rangle$ so that
 $v = (v \setminus \zeta) \dot{\vee} (v \wedge \zeta)$ and $\zeta = (v \wedge \zeta) \dot{\vee} (\zeta \setminus v)$
according to (1)(b)

(b) See the proof of 11.6.1(5)(b).

11.5.3 Remark

Note the following phenomena that differ from the common properties of lattice-like operations:

(1) \vee is associative for compatible arguments (11.5.1(9)(b)), but it is not associative in general. For example, let

$$\xi = \begin{bmatrix} a \mapsto 1 \\ b \mapsto 2 \end{bmatrix} \quad v = \begin{bmatrix} a \mapsto -1 \\ b \mapsto 2 \end{bmatrix} \quad \zeta = \begin{bmatrix} a \mapsto -1 \\ b \mapsto 2 \end{bmatrix}$$

then

$$(\xi \vee v) \vee \zeta = \begin{bmatrix} a \mapsto -1 \\ b \mapsto 2 \end{bmatrix} \neq \begin{bmatrix} b \mapsto 2 \end{bmatrix} = \xi \vee (v \vee \zeta)$$

(2) ∇ is associative for compatible arguments (11.5.1(9)(c)), but not in general. For example, for

$$\xi = [i \mapsto 1] \quad v = [i \mapsto 2] \quad \zeta = [i \mapsto 3]$$

we obtain

$$\xi \nabla (v \nabla \zeta) = [i \mapsto 1] \neq [i \mapsto 3] = (\xi \nabla v) \nabla \zeta$$

(3) The condition $v \smile \zeta$ is imperative for the distributivity law 11.5.1(10)(a) $\xi \wedge (v \vee \zeta) = (\xi \wedge v) \vee (\xi \wedge \zeta)$ to hold in general. Consider the following counterexample, where $v \not\smile \zeta$ with

$$\xi = \begin{bmatrix} b \mapsto 1 \\ c \mapsto 1 \\ d \mapsto 1 \end{bmatrix} \quad v = \begin{bmatrix} a \mapsto 1 \\ b \mapsto 1 \\ c \mapsto 1 \end{bmatrix} \quad \zeta = \begin{bmatrix} c \mapsto 3 \\ d \mapsto 1 \\ e \mapsto 1 \end{bmatrix}$$

For these records we obtain

$$\begin{aligned} \xi \wedge (v \vee \zeta) &= \begin{bmatrix} b \mapsto 1 \\ c \mapsto 1 \\ d \mapsto 1 \end{bmatrix} \wedge \begin{bmatrix} a \mapsto 1 \\ b \mapsto 1 \\ d \mapsto 1 \\ e \mapsto 1 \end{bmatrix} = \begin{bmatrix} b \mapsto 1 \\ d \mapsto 1 \end{bmatrix} \\ &\neq \begin{bmatrix} b \mapsto 1 \\ c \mapsto 1 \\ d \mapsto 1 \end{bmatrix} = \begin{bmatrix} b \mapsto 1 \\ c \mapsto 1 \end{bmatrix} \vee \begin{bmatrix} d \mapsto 1 \end{bmatrix} = (\xi \wedge v) \vee (\xi \wedge \zeta) \end{aligned}$$

11.6 Properties for compatible arguments

11.6.1 Lemma —binary junctions of compatible arguments—

Let $\xi = [\xi_i | i \in I]$, $v = [v_j | j \in J]$, $\zeta = [\zeta_k | k \in K]$ be three (pairwise) compatible records, i.e. $\xi \smile v \smile \zeta$. Next to the properties of 11.5.1, we now have the following additional facts.

(1) Compatible junction as projections: ξ, v, ζ have an upper bound η (i.e. there is a record η with $\xi, v, \zeta \leq \eta$). And for every upper bound η holds

- (a) $\xi, v, \zeta \in \mathbf{Proj}(\eta)$
- (b) $\xi = \mathbf{pr}(\eta, I)$
- (c) $\xi \setminus v = \mathbf{pr}(\eta, I \setminus J)$
- (d) $\xi \wedge v = \mathbf{pr}(\eta, I \cap J)$
- (e) $\xi \vee v = \mathbf{pr}(\eta, I \cup J)$
- (f) $\xi \nabla v = \mathbf{pr}(\eta, I \nabla J)$

(2) Rejector and updater: they become superfluous in the sense that

- (a) $\xi \times v = \langle \rangle$
- (b) $\xi \surd v = \xi \vee v$

(3) Subtraction together with meet and join:

- (a) $\xi \setminus (v \wedge \zeta) = (\xi \setminus v) \vee (\xi \setminus \zeta)$
- (b) $\xi \setminus (v \vee \zeta) = (\xi \setminus v) \wedge (\xi \setminus \zeta)$
- (c) $(v \wedge \zeta) \setminus \xi = (v \setminus \xi) \wedge (\zeta \setminus \xi)$
- (d) $(v \vee \zeta) \setminus \xi = (v \setminus \xi) \vee (\zeta \setminus \xi)$
- (e) $(\xi \setminus v) \setminus \zeta = \xi \setminus (v \vee \zeta)$
- (f) $\xi \setminus (v \setminus \zeta) = (\xi \setminus v) \vee (\xi \wedge \zeta)$
- (g) $(\xi \setminus v) \wedge \zeta = (\xi \wedge \zeta) \setminus v$
- (h) $(\xi \setminus v) \vee \zeta = (\xi \vee \zeta) \setminus (v \setminus \zeta)$

(4) Associativity:

- (a) $(\xi \wedge v) \wedge \zeta = \xi \wedge (v \wedge \zeta)$
- (b) $(\xi \vee v) \vee \zeta = \xi \vee (v \vee \zeta)$
- (c) $(\xi \nabla v) \nabla \zeta = \xi \nabla (v \nabla \zeta)$

(5) Distributivity:

- (a) $\xi \wedge (v \vee \zeta) = (\xi \wedge v) \vee (\xi \wedge \zeta)$
- (b) $\xi \vee (v \wedge \zeta) = (\xi \vee v) \wedge (\xi \vee \zeta)$

11.6.2 Proof of 11.6.1

(1) According to 11.5.1(6) and (8)(c), $\xi \smile v$ means that ξ and v have an upper bound, which is every record η with $\xi \vee v \leq \eta$. \leq is transitive and similar statements hold for three arguments as well: ξ, v, ζ have a least upper bound and that is given by any η with $(\xi \vee v) \vee \zeta \leq \eta$. For each such $\eta = [\eta_l | l \in L]$, we have $\xi, v, \zeta \in \mathbf{Proj}(\eta)$, the order structure on $\mathbf{Proj}(\eta)$ resembles the \subseteq -order structure on $\mathbf{P}(L)$ (i.e. they are isomorph), so that each of the record junctions $\setminus, \wedge, \vee, \nabla$ is a projections of η onto the according domain, which is a result of the according operation $\setminus, \cap, \cup, \nabla$, respectively.

(2) $\xi \smile v$ implies $\mathbf{D}_{\{\xi, v\}}^\times = \emptyset$, so that

$$(a) \quad \xi \times v = \mathbf{pr}(\xi, \mathbf{D}_{\{\xi, v\}}^\times) = \mathbf{pr}(\xi, \emptyset) = \langle \rangle$$

(b) Applying the distinct decompositions from 11.5.1(1), we obtain $\xi \surd v = (\xi \vee v) \dot{\vee} (\xi \times v) = (\xi \vee v) \dot{\vee} \langle \rangle = \xi \vee v$.

(3) Using (1), we are able to translate every record junction on $\mathbf{Proj}(\eta)$ into a set operation on the junction domains. And each of the eight statements (a)–(h) corresponds to the according one of the eight statements in 5.6.5. For example, for (a) we have

$$\begin{aligned} \xi \setminus (v \wedge \zeta) &= \mathbf{pr}(\eta, I \setminus (J \cap K)) \\ &= \mathbf{pr}(\eta, (I \setminus J) \cup (I \setminus K)) \quad \text{due to 5.6.5(1)} \\ &= (\xi \setminus v) \vee (\xi \setminus \zeta) \end{aligned}$$

(4) see 11.5.1(9).

(5) see 11.5.1(10).

11.7 Big junctions

11.7.1 Remark —introduction—

We will now generalize the binary junctors to their *big* versions and introduce \bigvee , \bigwedge and \bigtriangledown . The recipe is given in 11.4.4(2). The domain of the big junction is the according class operation on the arguments, but without the incompatible indices, i.e.

junction	its domain
$\bigvee \Xi$	$\left(\bigcup_{\xi \in \Xi} \mathbf{dom}(\xi) \right) \setminus \mathbf{D}_{\Xi}^{\neq} = \mathbf{D}_{\Xi} \setminus \mathbf{D}_{\Xi}^{\neq} = \mathbf{D}_{\Xi}^{\sim}$
$\bigwedge \Xi$	$\left(\bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) \right) \setminus \mathbf{D}_{\Xi}^{\neq}$
$\bigtriangledown \Xi$	$\left(\bigtriangledown_{\xi \in \Xi} \mathbf{dom}(\xi) \right) \setminus \mathbf{D}_{\Xi}^{\neq} = \bigtriangledown_{\xi \in \Xi} \mathbf{dom}(\xi)$

Deleting the compatible domain \mathbf{D}_{Ξ}^{\neq} in each case ensures that the value $\Xi(k)$ of the new record is well defined for each k of the according domain.

11.7.2 Definition

Given a record class Ξ . For each $k \in \mathbf{D}_{\Xi}^{\sim}$ we define
$\Xi(k) := \mathbf{sing}\{\xi(k) \mid \xi \in \Xi, k \in \mathbf{dom}(\xi)\}$
the <u>value</u> of Ξ at k
And so we can introduce
$\bigvee \Xi := [\Xi(k) \mid k \in \mathbf{D}_{\Xi}^{\sim}]$
the <u>supremum</u> or <u>(big) disjunction</u> of Ξ
$\bigwedge \Xi := \left[\Xi(k) \mid k \in \left(\bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) \right) \setminus \mathbf{D}_{\Xi}^{\neq} \right]$
the <u>infimum</u> of <u>(big) conjunction</u> of Ξ
$\bigtriangledown \Xi := [\Xi(k) \mid k \in \bigtriangledown_{\xi \in \Xi} \mathbf{dom}(\xi)]$
the <u>(big) opposition</u> of Ξ

11.7.3 Example big junctions

An example record class is

$$\Xi = \left\{ \left[\begin{array}{l} i \mapsto 1 \\ j \mapsto 2 \\ k \mapsto 3 \end{array} \right], \left[\begin{array}{l} j \mapsto -2 \\ k \mapsto 3 \\ l \mapsto 4 \end{array} \right], \left[\begin{array}{l} j \mapsto 2 \\ k \mapsto 3 \\ l \mapsto -4 \\ m \mapsto 5 \end{array} \right] \right\}$$

with

$$\begin{aligned} \mathbf{D}_{\Xi}^{\sim} &= \bigcup_{\xi \in \Xi} \mathbf{dom}(\xi) = \{i, j, k, l, m\} \\ \bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) &= \{j, k\} \\ \bigtriangledown_{\xi \in \Xi} \mathbf{dom}(\xi) &= \{i, m\} \\ \mathbf{D}_{\Xi}^{\neq} &= \{j, l\} \\ \mathbf{D}_{\Xi}^{\sim} &= \{i, k, m\} \end{aligned}$$

so that

$$\bigvee \Xi = \left[\begin{array}{l} i \mapsto 1 \\ k \mapsto 3 \\ m \mapsto 5 \end{array} \right] \quad \bigwedge \Xi = [k \mapsto 3] \quad \bigtriangledown \Xi = \left[\begin{array}{l} i \mapsto 1 \\ m \mapsto 5 \end{array} \right]$$

11.7.4 Definition

For arbitrary records ξ_1, \dots, ξ_n we define

$$\begin{aligned} \xi_1 \vee \dots \vee \xi_n &:= \bigvee \{\xi_1, \dots, \xi_n\} \\ \xi_1 \wedge \dots \wedge \xi_n &:= \bigwedge \{\xi_1, \dots, \xi_n\} \end{aligned}$$

11.7.5 Remark

- (1) $\bigvee \Xi$ and $\bigtriangledown \Xi$ are well-defined for all record classes Ξ . But $\bigwedge \Xi$ is well-defined only if $\Xi \neq \emptyset$, because $\bigcap \emptyset$ is undefined. Accordingly, $\xi_1 \wedge \dots \wedge \xi_n$ exists iff $n \geq 0$, while $\xi_1 \vee \dots \vee \xi_n = \langle \rangle$ for $n = 0$. In 11.9.1 we introduce a useful mutation of \bigwedge that is defined on its whole domain.
- (2) Definition 11.7.4 is well-defined generalization of the binary \wedge and \vee , because 11.7.6(1)(a) $\xi \vee v = \bigvee \{\xi, v\}$ and $\xi \wedge v = \bigwedge \{\xi, v\}$. But we need to keep in mind, that parentheses are not arbitrary in case of incompatible records, because the join is not associative.
- (3) A similar generalization of the opposition would not be consistent, because $\xi \triangledown v = \bigtriangledown \{\xi, v\}$ is not true in general. The opposition is nilpotent, not idempotent, so $\xi \triangledown \xi = \langle \rangle$, but $\bigtriangledown \{\xi, \xi\} = \bigtriangledown \{\xi\} = \xi$.

11.7.6 Lemma properties of the big junctions

- (1) Generalization: For every two records ξ and v
 - (a) $\xi \vee v = \bigvee \{\xi, v\}$
 - (b) $\xi \wedge v = \bigwedge \{\xi, v\}$

- (2) Empty argument class:
 - (a) $\bigvee \emptyset = \langle \rangle$
 - (b) $\bigwedge \emptyset$ is undefined
 - (c) $\bigtriangledown \emptyset = \langle \rangle$

- (3) Compatibility: For every record class Ξ the following statements are equivalent
 - (i) Ξ is compatible
 - (ii) $\xi \leq \bigvee \Xi$ for each $\xi \in \Xi$
 - (iii) Ξ has an upper bound
 - (iv) $\mathbf{D}_{\Xi}^{\neq} = \emptyset$
 - (v) $\Xi \subseteq \mathbf{Proj}(\bigvee \Xi)$
 - (vi) $\Xi \subseteq \mathbf{Proj}(\eta)$ for some record η

- (4) Order: For every record class Ξ holds
 - (a) $\bigwedge \Xi \leq \bigvee \Xi$
 - (b) $\bigtriangledown \Xi \leq \bigvee \Xi$

- (5) Extrema: For every record class Ξ holds
 - (a) If $\Xi \neq \emptyset$ then $\bigwedge \Xi$ is the *greatest lower bound* of Ξ
 - (b) The following statements are equivalent:
 - (i) Ξ is compatible
 - (ii) $\bigvee \Xi$ is the *least upper bound* of Ξ

(6) Compatible big junctions as projections: Let η be a record and $\Xi \subseteq \mathbf{Proj}(\eta)$. That means, that Ξ is compatible and

$$(a) \quad \bigvee \Xi = \mathbf{pr} \left(\eta, \bigcup_{\xi \in \Xi} \mathbf{dom}(\xi) \right) = \mathbf{pr}(\eta, \mathbf{D}_\Xi)$$

$$(b) \quad \bigwedge \Xi = \mathbf{pr} \left(\eta, \bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) \right)$$

$$(c) \quad \nabla \Xi = \mathbf{pr} \left(\eta, \bigvee_{\xi \in \Xi} \mathbf{dom}(\xi) \right)$$

(7) Subtraction and distributivity: For a record η and all $\Xi \subseteq \mathbf{Proj}(\eta)$ and $v \in \mathbf{Proj}(\eta)$ holds

$$(a) \quad v \setminus \bigwedge \Xi = \bigvee \{v \setminus \xi \mid \xi \in \Xi\} \quad \text{for all } \Xi \neq \emptyset$$

$$(b) \quad v \setminus \bigvee \Xi = \bigwedge \{v \setminus \xi \mid \xi \in \Xi\} \quad \text{for all } \Xi \neq \emptyset$$

$$(c) \quad (\bigwedge \Xi) \setminus v = \bigwedge \{\xi \setminus v \mid \xi \in \Xi\} \quad \text{for all } \Xi \neq \emptyset$$

$$(d) \quad (\bigvee \Xi) \setminus v = \bigvee \{\xi \setminus v \mid \xi \in \Xi\}$$

$$(e) \quad v \wedge \bigvee \Xi = \bigvee \{v \wedge \xi \mid \xi \in \Xi\}$$

$$(f) \quad v \vee \bigwedge \Xi = \bigwedge \{v \vee \xi \mid \xi \in \Xi\} \quad \text{for all } \Xi \neq \emptyset$$

11.7.7 Proof of 11.7.6

(1) Generalization: If $\xi = [\xi_i \mid i \in I]$ and $v = [v_j \mid j \in J]$ then

$$\begin{aligned} \bigvee \{\xi, v\} &= [\Xi(k) \mid k \in \mathbf{D}_{\{\xi, v\}}^-] \\ &= \left[k \mapsto \begin{cases} \xi_k & \text{if } k \in I \\ v_k & \text{if } k \in J \end{cases} \mid k \in \mathbf{D}_{\{\xi, v\}}^- \right] \\ &= \xi \vee v \end{aligned}$$

$$\begin{aligned} \bigwedge \{\xi, v\} &= [\Xi(k) \mid k \in (I \cap J) \setminus \mathbf{D}_{\{\xi, v\}}^-] \\ &= \mathbf{pr}(\xi_k, k \in (I \cap J) \setminus \mathbf{D}_{\{\xi, v\}}^-) \\ &= \xi \wedge v \end{aligned}$$

(2) Empty argument class: $\mathbf{D}_\emptyset^- = \emptyset$, $\bigcup \{\mathbf{dom}(\xi) \mid \xi \in \emptyset\} = \nabla \{\mathbf{dom}(\xi) \mid \xi \in \emptyset\} = \emptyset$ and $\bigcap \{\mathbf{dom}(\xi) \mid \xi \in \emptyset\}$ is undefined, so that $\bigvee \emptyset = \nabla \emptyset = \langle \rangle$ and $\bigwedge \emptyset$ is undefined.

(3) Compatibility: To proof the mutual equivalence of all six statements, we use the following steps (i) \Leftrightarrow (iv) \Leftrightarrow (ii) \Leftrightarrow (v) \Leftrightarrow (vi) \Leftrightarrow (iii), which are justified as follows.

(i) \Leftrightarrow (iv) See 11.3.4.

(iv) \Leftrightarrow (ii) If $\mathbf{D}_\Xi^- = \emptyset$ then $\mathbf{dom}(\xi) \subseteq \mathbf{D}_\Xi = \mathbf{dom}(\bigvee \Xi)$ for each $\xi \in \Xi$. On the other hand, if $\mathbf{D}_\Xi^- \neq \emptyset$ then there is a $v \in \Xi$ and $j \in \mathbf{dom}(v)$ with $j \notin \mathbf{D}_\Xi = \mathbf{dom}(\bigvee \Xi)$. So $\mathbf{dom}(v) \not\subseteq \mathbf{dom}(\bigvee \Xi)$, i.e. $v \not\leq \bigvee \Xi$.

(ii) \Leftrightarrow (v) See 10.3.12(3).

(v) \Leftrightarrow (vi) Obviously, (v) implies (vi) when $\eta := \bigvee \Xi$. On the other hand, (vi) is saying that Ξ has an upper bound η , so $\xi \leq \bigvee \Xi \leq \eta$ for all $\xi \in \Xi$, and that is (v).

(vi) \Leftrightarrow (iii) Ξ has a lower bound η iff $\xi \leq \eta$ for all $\xi \in \Xi$ iff $\xi \in \mathbf{Proj}(\eta)$, again according to 10.3.12(3).

(4) Order:

(a) We have

$$\begin{aligned} \bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) &\subseteq \bigcup_{\xi \in \Xi} \mathbf{dom}(\xi) \\ \Rightarrow \bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) \setminus \mathbf{D}_\Xi^- &\subseteq \bigcup_{\xi \in \Xi} \mathbf{dom}(\xi) \setminus \mathbf{D}_\Xi^- \\ \Rightarrow \left[\frac{\Xi(k)}{k \in \bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) \setminus \mathbf{D}_\Xi^-} \right] &\leq \left[\frac{\Xi(k)}{k \in \bigcup_{\xi \in \Xi} \mathbf{dom}(\xi) \setminus \mathbf{D}_\Xi^-} \right] \\ \Rightarrow \bigwedge \Xi &\leq \bigvee \Xi \end{aligned}$$

(b) Similar to (a) we obtain

$$\begin{aligned} \nabla \{\mathbf{dom}(\xi) \mid \xi \in \Xi\} &\subseteq \bigcup \{\mathbf{dom}(\xi) \mid \xi \in \Xi\} \\ \Rightarrow \nabla \Xi &\leq \bigvee \Xi \end{aligned}$$

(5) Extrema:

(a) Every non-empty Ξ has lower bounds, at least $\langle \rangle$. Suppose, $v = [v_j \mid j \in J]$ is an arbitrary lower bound of Ξ , so $v \leq \xi$ for all $\xi \in \Xi$, i.e. $J \subseteq \bigcap \{\mathbf{dom}(\xi) \mid \xi \in \Xi\}$. Now if $k \in \mathbf{D}_\Xi^-$, there are $\xi', \xi'' \in \Xi$ with $\xi'(k) \neq \xi''(k)$. And k cannot be in J , because otherwise $v \not\leq \xi'$ or $v \not\leq \xi''$. So $\mathbf{D}_\Xi^- \cap J = \emptyset$, in other words, $J \subseteq \bigcap \{\mathbf{dom}(\xi) \mid \xi \in \Xi\} \setminus \mathbf{D}_\Xi^- = \mathbf{dom}(\bigwedge \Xi)$. Furthermore, $v \leq \bigwedge \Xi$, $\bigwedge \Xi$ is not only a lower bound of Ξ , but the unique greatest lower one.

(b) If Ξ is compatible, $\mathbf{D}_\Xi^- = \emptyset$ and $\bigvee \Xi = [\Xi(k) \mid k \in \mathbf{D}_\Xi]$. For every $\xi \in \Xi$, $\mathbf{dom}(\xi) \subseteq \mathbf{D}_\Xi$ and each $i \in \mathbf{dom}(\xi)$, $\xi(i) = \Xi(i)$, so $\xi \leq \bigvee \Xi$. But not only is $\bigvee \Xi$ an upper bound of Ξ , it also is the unique least upper bound, because for every upper bound v , the criterion $\xi \leq v$ for all $\xi \in \Xi$ would imply that $\mathbf{dom}(\xi) \subseteq \mathbf{dom}(v)$, i.e. $\mathbf{D}_\Xi \subseteq \mathbf{dom}(v)$ and thus $\bigvee \Xi \leq v$.

On the other hand, if Ξ is incompatible, it has at least two incompatible members. And we know from 11.5.1(a), that these two records don't have an upper bound. So Ξ cannot have an upper bound, let alone a least one.

(6) Compatible big junctions as projections: $\Xi \subseteq \eta$ means that $\xi \leq \eta$ for all $\xi \in \Xi$ and the compatibility of Ξ is a trivial consequence. Also, $\Xi(k) = \eta(k)$ for all $k \in \mathbf{D}_\Xi$, and (a), (b), (c) follow immediately from definition 11.7.2.

$$\begin{aligned} v \setminus \bigwedge \Xi &= \mathbf{pr}(\eta, J) \setminus \mathbf{pr} \left(\eta, \bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) \right) && \text{due to (6)} \\ &= \mathbf{pr} \left(\eta, J \setminus \bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) \right) \\ &= \mathbf{pr} \left(\eta, \bigcup_{\xi \in \Xi} (J \setminus \mathbf{dom}(\xi)) \right) && \text{see 5.6.5(1)} \\ &= \bigvee_{\xi \in \Xi} \mathbf{pr}(\eta, J \setminus \mathbf{dom}(\xi)) && \text{due to (6)(b)} \\ &= \bigvee_{\xi \in \Xi} \mathbf{pr}(v, J \setminus \mathbf{dom}(\xi)) \\ &= \bigvee_{\xi \in \Xi} (v \setminus \xi) \end{aligned}$$

This is a proof for (a) and the other statements (b)–(f) are proved likewise. The same idea is always, that in case we operate on $\mathbf{Proj}(\eta)$, the structure on records is isomorph to the power class structure on the domain of η , $\mathbf{P}(K)$. The record operations behave similar to the according class operations.

11.8 The general record structure

11.8.1 Definition

$$\mathfrak{REC} := \langle \mathbf{REC}, \emptyset, \smile, \leq, \langle \rangle, \setminus, \wedge, \times, \nabla, \vee, \sqrt{}, \bigwedge, \bigvee, \nabla \rangle$$

is the (general) record structure

11.8.2 Remark

Definition 11.8.1 combines the relations and junctions on records into an overall structure. For the type of the operations in \mathfrak{REC} holds

$$\left. \begin{array}{l} \textcircled{1} \emptyset \textcircled{2} \\ \textcircled{1} \smile \textcircled{2} \\ \textcircled{1} \leq \textcircled{2} \end{array} \right\} : \mathbf{REC} \rightsquigarrow \mathbf{REC} \quad \langle \rangle \in \mathbf{REC}$$

$$\left. \begin{array}{l} \textcircled{1} \setminus \textcircled{2} \\ \textcircled{1} \wedge \textcircled{2} \\ \textcircled{1} \times \textcircled{2} \\ \textcircled{1} \nabla \textcircled{2} \\ \textcircled{1} \vee \textcircled{2} \\ \textcircled{1} \checkmark \textcircled{2} \end{array} \right\} : \mathbf{REC} \times \mathbf{REC} \longrightarrow \mathbf{REC}$$

$$\bigwedge \textcircled{1} : (\mathbf{P}(\mathbf{REC}) \setminus \{\emptyset\}) \longrightarrow \mathbf{REC}$$

$$\left. \begin{array}{l} \bigvee \textcircled{1} \\ \bigtriangledown \textcircled{1} \end{array} \right\} : \mathbf{P}(\mathbf{REC}) \longrightarrow \mathbf{REC}$$

Recall, that $\bigwedge \emptyset$ is not defined in general, so \emptyset is excluded from the domain of \bigwedge in $\mathfrak{R}\mathfrak{E}\mathfrak{L}$.

11.9 The complete boolean algebra of projections

11.9.1 Definition

For every record η we define

$$\mathfrak{P}\text{roj}(\eta) := \langle \mathbf{Proj}(\eta), \leq, \langle \rangle, \eta, \wedge, \vee, \bigwedge, \bigvee, - \rangle$$

the record projection structure of η

where $\bigwedge \emptyset := \eta$ and $-\textcircled{1} := \eta \setminus \textcircled{1}$ is the complement.

11.9.2 Remark

- (1) According to definition 5.8.1 for the usual form of an ordinary structure, we should have mentioned $\mathbf{P}(\mathbf{Proj}(\eta))$ as a second carrier class, because that is the domain of \bigwedge and \bigvee . But, as it is common for complete boolean algebras (see e.g. $\mathbf{P}(C)$ in 6.2.2), this class is not mentioned explicitly.
- (2) Recall 11.7.6(2)(b) and 11.7.5(1), that \bigwedge turns any record class Ξ into a single record $\bigwedge \Xi$, but it is not defined for $\Xi = \emptyset$. But when \bigwedge becomes a part of $\mathfrak{P}\text{roj}(\eta)$, this gap is filled with the definition $\bigwedge \emptyset := \eta$. A full definition of this modified \bigwedge is given by

$$\bigwedge \textcircled{1} := \left[\begin{array}{l} \mathbf{P}(\mathbf{Proj}(\eta)) \longrightarrow \mathbf{Proj}(\eta) \\ \Xi \mapsto \bigwedge(\{\eta\} \cup \Xi) \end{array} \right]$$

(For a proof that this definition does the job, see in the proof of 11.9.4 under (a).)

- (3) In 11.9.4 we will see that $\mathfrak{P}\text{roj}(\eta)$ is a complete boolean algebra, for every record $\eta = [\eta_i | i \in I]$. This is basically true due to the isomorphy

$$\mathfrak{P}\text{roj}(\eta) \cong \mathfrak{P}(I)$$

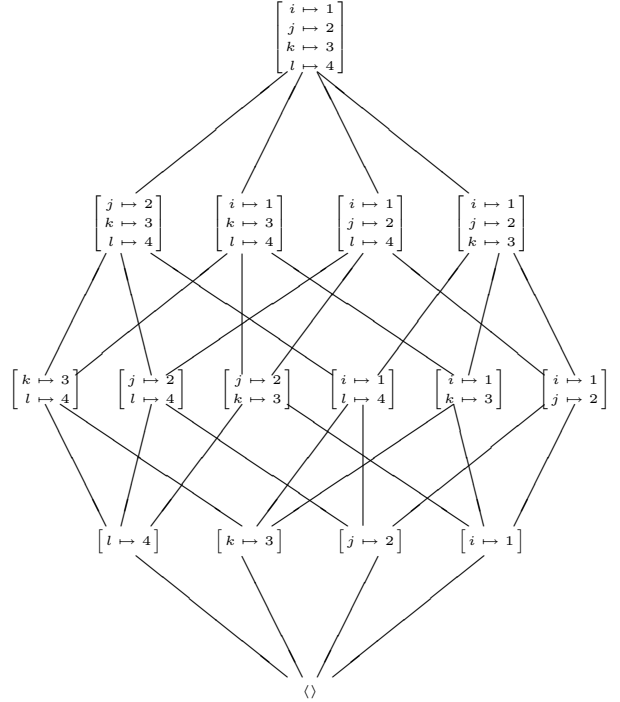
We illustrate this similarity in the following example 11.9.3.

11.9.3 Example lattice of projections

Take for example the record

$$\eta = \begin{bmatrix} i \mapsto 1 \\ j \mapsto 2 \\ k \mapsto 3 \\ l \mapsto 4 \end{bmatrix}$$

$\mathbf{Proj}(\eta)$ contains $2^4 = 16$ subrecords and they are ordered according to the following Hasse diagram:



This structure is isomorph to the power lattice algebra (see 6.2.4) of a four-element class, i.e. $\mathbf{P}(\{1, 2, 3, 4\})$.

11.9.4 Lemma

$\mathfrak{P}\text{roj}(\eta)$ is a complete boolean algebra, for every record η .

11.9.5 Proof of 11.9.4

Suppose, η is given by $\eta = [\eta_l | l \in L]$. In the sequel, let $v, \zeta \in \mathbf{Proj}(\eta)$ and $\Xi \subseteq \mathbf{Proj}(\eta)$.

- (a) $\langle \mathbf{Proj}(\eta), \leq \rangle$ is a poclass.
 $\langle \mathbf{REC}, \leq \rangle$ is a poclass, according to 11.1.1, so $\mathbf{Proj}(\eta) \subseteq \mathbf{REC}$ together with \leq is a poclass, too.
- (b) $\langle \rangle$ is the least element.
 This is true according to 11.5.1(8)(a).
- (c) η is the greatest element.
 True, because $\xi \leq \eta$, for all $\xi \in \mathbf{Proj}(\eta)$.
- (d) $\bigwedge(\{\eta\} \cup \Xi)$ is the greatest lower bound of Ξ .
 If $\Xi = \emptyset$, then $\bigwedge(\{\eta\} \cup \Xi) = \bigwedge \{\eta\} = \eta$, and η is the greatest lower bound of η .
 On the other hand, if $\Xi \neq \emptyset$, then the fact that $\{\eta\} \cup \Xi \subseteq \mathbf{Proj}(\eta)$ allows us to apply 11.7.6(8)(b) and we obtain

$$\begin{aligned} \bigwedge(\{\eta\} \cup \Xi) &= \mathbf{pr} \left(\eta, L \cap \bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) \right) \\ &= \mathbf{pr} \left(\eta, \bigcap_{\xi \in \Xi} \mathbf{dom}(\xi) \right) \\ &= \bigwedge \Xi \end{aligned}$$

and $\bigwedge \Xi$ is the greatest lower bound of Ξ for $\Xi \neq \emptyset$, according to 11.7.6(5)(a).

- (e) $v \wedge \zeta$ is the greatest lower bound of v and ζ .
That is due to 11.7.6(1)(b), saying that $v \wedge \zeta = \bigwedge \{v, \zeta\}$, and $\bigwedge \{v, \zeta\}$ is the greatest lower bound of v and ζ , as proved in (d).
- (f) $\bigvee \Xi$ is the least upper bound of Ξ .
 Ξ is compatible according to 11.7.6(3), thus $\bigvee \Xi$ is the least upper bound of Ξ according to 11.7.6(5)(b).
- (g) $v \vee \zeta$ is the least upper bound of v and ζ .
 $v \vee \zeta = \bigvee \{v, \zeta\}$ according to 11.7.6(1)(a) and due to (g), that is the least upper bound.
- (h) The whole lattice is distributive.
See 11.6.1(5).
- (i) $-v$ is the complement of v .
The criteria 7.5.2 of a complement are satisfied:
 $(-v) \wedge v$
 $= (\eta \setminus v) \wedge v$

$$\begin{aligned} &= (\eta \wedge v) \setminus v && \text{due to 11.6.1(3)(g)} \\ &= v \setminus v && \text{because } v \leq \eta, \text{ so } \eta \wedge v = v \\ &= \langle \rangle && \text{due to 11.5.1(4)(a)} \end{aligned}$$

and

$$\begin{aligned} &(-v) \vee v \\ &= (\eta \setminus v) \vee v \\ &= (\eta \vee v) \setminus (v \setminus v) && \text{due to 11.6.1(3)(h)} \\ &= \eta \setminus \langle \rangle && \text{because } v \leq \eta, \text{ so } \eta \vee v = \eta \\ &= \eta && \text{due to 11.5.1(5)(a)} \end{aligned}$$

Alltogether, the poclass $\langle \mathbf{Proj}(\eta), \leq \rangle$ is indeed a complete complemented distributive lattice, i.e. a complete boolean lattice.

Part V

Schemas and their various products

12 Various schema and power products

12.1 Various products

12.1.1 Repetition

Recall from 9.3.1, that a schema is a record $X = [X_i | i \in I]$, where each value X_i is a class, called the i -th domain.

Furthermore, such an X was in 9.3.3 defined to be called

- ♣ proper, if all X_i are non-empty
- ♣ finite, if I is finite
- ♣ completely finite, if I as well as each X_i is finite

12.1.2 Definition various products of schemas

For every schema $X = [X_i | i \in I]$ we define

$$\otimes X := \{[x_j | j \in J] \mid J \subseteq I \text{ and } x_j \in X_j \text{ for all } j \in J\}$$

the star product or the (general) record class of X ,

$$\otimes X := \{[x_i | i \in I] \mid x_i \in X_i \text{ for all } i \in I\}$$

the (cartesian) product or the expanded record class of X ,

$$\oplus X := \{[i \mapsto x] \mid i \in I \text{ and } x \in X_i\}$$

the coproduct or disjunct union or the singular (or literal) record class of X .

12.1.3 Example product

A schema is given by

$$X = \left[\begin{array}{l} \mathbf{a} \mapsto \{2\} \\ \mathbf{b} \mapsto \{2, 3, 4\} \\ \mathbf{c} \mapsto \{5\} \end{array} \right]$$

We obtain

$$\otimes X = \left\{ \begin{array}{l} \langle \rangle, [\mathbf{a} \mapsto 2], [\mathbf{b} \mapsto 2], \\ [\mathbf{b} \mapsto 3], [\mathbf{b} \mapsto 4], [\mathbf{c} \mapsto 5], \\ \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{b} \mapsto 2 \end{array} \right], \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{b} \mapsto 3 \end{array} \right], \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{b} \mapsto 4 \end{array} \right], \\ \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{c} \mapsto 5 \end{array} \right], \left[\begin{array}{l} \mathbf{b} \mapsto 2 \\ \mathbf{c} \mapsto 5 \end{array} \right], \left[\begin{array}{l} \mathbf{b} \mapsto 3 \\ \mathbf{c} \mapsto 5 \end{array} \right], \left[\begin{array}{l} \mathbf{b} \mapsto 4 \\ \mathbf{c} \mapsto 5 \end{array} \right], \\ \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{b} \mapsto 2 \\ \mathbf{c} \mapsto 5 \end{array} \right], \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{b} \mapsto 3 \\ \mathbf{c} \mapsto 5 \end{array} \right], \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{b} \mapsto 4 \\ \mathbf{c} \mapsto 5 \end{array} \right] \end{array} \right\}$$

$$\otimes X = \left\{ \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{b} \mapsto 2 \\ \mathbf{c} \mapsto 5 \end{array} \right], \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{b} \mapsto 3 \\ \mathbf{c} \mapsto 5 \end{array} \right], \left[\begin{array}{l} \mathbf{a} \mapsto 2 \\ \mathbf{b} \mapsto 4 \\ \mathbf{c} \mapsto 5 \end{array} \right] \right\}$$

$$\oplus X = \left\{ \begin{array}{l} [\mathbf{a} \mapsto 2], [\mathbf{b} \mapsto 2], [\mathbf{b} \mapsto 3], \\ [\mathbf{b} \mapsto 4], [\mathbf{c} \mapsto 5] \end{array} \right\}$$

12.1.4 Lemma

For every schema $X = [X_i | i \in I]$ holds:

- (1) $\otimes X \subseteq \otimes X$ and $\oplus X \subseteq \otimes X$
- (2) Equivalent are:
 - (a) $\otimes X \cap \oplus X \neq \emptyset$
 - (b) $\otimes X = \oplus X = \otimes X \setminus \{\langle \rangle\}$
 - (c) $\mathbf{card}(I) = 1$
- (3) $\otimes X = \bigcup \{\otimes Y \mid Y \in \mathbf{Proj}(X)\}$

12.1.5 Proof of 12.1.4

From definition 12.1.2 we derive

$$\otimes X = \{x \in \otimes X \mid \mathbf{dom}(x) = I\}$$

$$\oplus X = \{x \in \otimes X \mid \mathbf{card}(\mathbf{dom}(x)) = 1\}$$

so that (1) and (2) follow. (3) is true, because

$$\begin{aligned} \otimes X &= \{[x_j | j \in J] \mid J \subseteq I \text{ and } x_j \in X_j \text{ for all } j \in J\} \\ &= \bigcup_{J \subseteq I} \{[x_j | j \in J] \mid x_j \in X_j \text{ for all } j \in J\} \\ &= \bigcup_{J \subseteq I} \otimes \mathbf{pr}(X, J) = \bigcup_{Y \in \mathbf{Proj}(X)} \otimes Y \end{aligned}$$

12.1.6 Remark Record tables

In case a proper schema $X = [X_i | i \in I]$ is indeed completely finite, we can represent each of its products by a record table. The $i_1, \dots, i_n \in I$ form the columns, and each record is a row. If a record is not defined for a given $i \in I$, the according cell is left empty. The following example demonstrates the general idea.

12.1.7 Example record tables

Consider the last example schema X again. X is proper and completely finite, and its various products can be represented by

a	b	c
2		
	2	
	3	
		5
2	2	
2	3	
2	4	
2		5
	2	5
	3	5
	4	5
2	2	5
2	3	5
2	4	5

 $\otimes X =$

a	b	c
2	2	5
2	3	5
2	4	5

 $\oplus X =$

a	b	c
2		
	2	
	3	
	3	
		5

12.1.8 Repetition

Recall from 5.6.15, that for every class $\kappa = \{\kappa_i \mid i \in I\}$ of cardinal numbers, $\prod_{i \in I} \kappa_i$ denotes the arithmetic product (with result 1 for $\kappa = \emptyset$), and $\sum \kappa$ or $\sum_{i \in I} \kappa_i$ is the arithmetic sum (with result 0 for $I = \emptyset$).

12.1.9 Lemma cardinalities

For every schema $X = [X_i \mid i \in I]$ holds

$$\text{card}(\otimes X) = \prod_{i \in I} \text{card}(X_i)$$

$$\text{card}(\oplus X) = \sum_{i \in I} \text{card}(X_i)$$

12.1.10 Proof of 12.1.9

Suppose, X is completely finite, i.e. $I = \{i_1, \dots, i_n\}$ and $\text{card}(X_k) = m_k \in \mathbb{N}$ for each $k \in I$. Then

$$\begin{aligned} \text{card}(\otimes X) &= \text{card} \left\{ \left[\begin{array}{c} i_1 \mapsto x_1 \\ \vdots \\ i_n \mapsto x_n \end{array} \middle| \begin{array}{c} x_1 \in X_1, \\ \vdots \\ x_n \in X_n \end{array} \right\} \\ &= m_1 \cdot m_2 \cdot \dots \cdot m_n \end{aligned}$$

and

$$\begin{aligned} \text{card}(\oplus X) &= \text{card}\{[i \mapsto x] \mid i \in I, x \in X_i\} \\ &= \text{card} \left(\left\{ \begin{array}{c} [i_1 \mapsto x] \\ x \in X_1 \end{array} \right\} \uplus \dots \uplus \left\{ \begin{array}{c} [i_n \mapsto x] \\ x \in X_n \end{array} \right\} \right) \\ &= m_1 + m_2 + \dots + m_n \end{aligned}$$

where the last step is true, because the n classes $\{[i_k \mapsto x] \mid x \in X_k\}$ are pairwise disjoint. (Therefore, $\oplus X$ is also called the *disjunct union* of the classes X_k .) So far for the completely finite X .

If X is not completely finite, we are dealing with arithmetic sums and products which have infinitely many arguments or the arguments are infinite themselves. In these cases our lemma is correct by definition, because Cantor introduced e.g. the product of transfinite cardinal numbers as the cardinality of the cartesian product of transfinite classes. But we did not define transfinite arithmetic and our proof remains incomplete here.

12.1.11 Remark cardinalities and special cases

We emphasize, that 12.1.9 also holds for the following special cases. Suppose, $X = [X_i \mid i \in I]$ is a given schema.

- (1) If I is empty, then $X = \langle \rangle$ is the empty tuple and so
 - (a) $\otimes X = \{\langle \rangle\}$. The cardinality $\text{card}(\otimes X)$ is 1, which is also the arithmetic product of zero arguments (i.e. 1 is the neutral element of \prod).
 - (b) $\oplus X = \emptyset$, obviously, and
 - (c) $\otimes X = \{\langle \rangle\}$.
- (2) If $I \neq \emptyset$ and $X_i = \emptyset$, for at least one $i \in I$, then $\otimes X = \emptyset$. In fact, this also holds the other way round.

It is worth to keep these things in mind:

12.1.12 Lemma

For every schema $X = [X_i \mid i \in I]$ holds:

- (1) $\otimes X = \emptyset$ iff X is not proper (i.e. $X_i = \emptyset$ for one $i \in I$)
- (2) $\otimes X = \{\langle \rangle\}$ iff X is empty (i.e. $I = \emptyset$)

12.1.13 Proof of 12.1.12

See 12.1.11.

12.1.14 Lemma cardinality of the star product

Given a schema $X = [X_i \mid i \in I]$. Then

$$\text{card}(\otimes X) = \sum_{J \subseteq I} \prod_{j \in J} \text{card}(X_j)$$

12.1.15 Proof of 12.1.14

$$\begin{aligned} \text{card}(\otimes X) &= \text{card}(\bigcup \{\otimes Y \mid Y \in \mathbf{Proj}(X)\}) && \text{due to 12.1.4(3)} \\ &= \sum_{Y \in \mathbf{Proj}(X)} \text{card}(\otimes Y) \\ &\quad \text{as the cartesian products are pairwise disjoint} \\ &= \sum_{J \subseteq I} \text{card}(\otimes [X_j \mid j \in J]) \\ &= \sum_{J \subseteq I} \prod_{j \in J} \text{card}(X_j) && \text{due to 12.1.9} \end{aligned}$$

12.1.16 Remark

In 11.2.2 we defined

$$\mathbf{Rec}(I, C) = \left\{ [\xi_j \mid j \in J] \middle| \begin{array}{l} J \subseteq I, \\ \xi_j \in C \text{ for all } j \in J \end{array} \right\}$$

We can now write this as

$$\otimes[C \mid I] = \mathbf{Rec}(I, C)$$

12.2 Various ordinal products

12.2.1 Remark

Recall, that a tuple is just a certain kind of record with a finite ordinal as index class. In other words,

$$\langle x_1, \dots, x_n \rangle = [x_i | i \in \{1, \dots, n\}]$$

So, a class tuple $\langle C_1, \dots, C_n \rangle$ is just a special kind of schema and the various products $\otimes\langle C_1, \dots, C_n \rangle$, $\otimes\langle C_1, \dots, C_n \rangle$, and $\oplus\langle C_1, \dots, C_n \rangle$ are well-defined. And if we write $\otimes\langle C_1, \dots, C_n \rangle$ as $C_1 \times \dots \times C_n$, we obtain the usual cartesian product of classes. Accordingly, we also introduce a “+” and “*” notation in analogy to “×”.

12.2.2 Definition — various ordinal products

Let $n \in \mathbb{N}$ and C_1, \dots, C_n be classes. We define

$C_1 \star \dots \star C_n := \bigstar_{i=1}^n C_i := \otimes\langle C_1, \dots, C_n \rangle$
the (ordinal) star product of C_1, \dots, C_n

$C_1 \times \dots \times C_n := \bigtimes_{i=1}^n C_i := \otimes\langle C_1, \dots, C_n \rangle$
the (ordinal or cartesian) product of C_1, \dots, C_n

$C_1 + \dots + C_n := \bigoplus_{i=1}^n C_i := \oplus\langle C_1, \dots, C_n \rangle$
the (ordinal) coproduct of C_1, \dots, C_n
or disjoint union of C_1, \dots, C_n

12.2.3 Remark — various ordinal products

So, for $n \in \mathbb{N}$ and classes C_1, \dots, C_n we obtain

$$C_1 \star \dots \star C_n = \left\{ \frac{[x_j | j \in J]}{J \subseteq \{1, \dots, n\} \text{ and } x_j \in C_j \text{ for each } j \in J} \right\}$$

$$C_1 \times \dots \times C_n = \left\{ \frac{\langle x_1, \dots, x_n \rangle}{x_1 \in C_1, \dots, x_n \in C_n} \right\}$$

$$C_1 + \dots + C_n = \left\{ \frac{[i \mapsto x]}{i \in \{1, \dots, n\}, x \in C_i} \right\}$$

12.2.4 Example — ordinal products

Recall that

$$\langle \{a, b\}, \{c, d\} \rangle = \left[\begin{array}{l} 1 \mapsto \{a, b\} \\ 2 \mapsto \{c, d\} \end{array} \right]$$

so that

$$\langle \{a, b\} \star \{c, d\} \rangle = \left[\begin{array}{l} \langle \rangle, \\ [1 \mapsto a], [1 \mapsto b], \\ [2 \mapsto c], [2 \mapsto d], \\ [1 \mapsto a], [1 \mapsto b], \\ [2 \mapsto c], [2 \mapsto d], \\ [1 \mapsto a], [1 \mapsto b], \\ [2 \mapsto d], [2 \mapsto d] \end{array} \right]$$

$$\langle \{a, b\} \times \{c, d\} \rangle = \left[\begin{array}{l} [1 \mapsto a], [1 \mapsto b], \\ [2 \mapsto c], [2 \mapsto d], \\ [1 \mapsto a], [1 \mapsto b], \\ [2 \mapsto d], [2 \mapsto d] \end{array} \right] = \left\{ \begin{array}{l} \langle a, c \rangle \\ \langle a, d \rangle \\ \langle b, c \rangle \\ \langle b, d \rangle \end{array} \right\}$$

$$\langle \{a, b\} + \{c, d\} \rangle = \left[\begin{array}{l} [1 \mapsto a], [1 \mapsto b], \\ [2 \mapsto c], [2 \mapsto d] \end{array} \right]$$

12.2.5 Lemma

Given $n \in \mathbb{N}$ classes C_1, \dots, C_n .

- (1) If $n = 0$ then
 - (a) $C_1 \star \dots \star C_n = \{\langle \rangle\}$
 - (b) $C_1 \times \dots \times C_n = \emptyset$
 - (c) $C_1 + \dots + C_n = \emptyset$
- (2) If $n = 1$ then
 - (a) $C_1 \star \dots \star C_n = \{\langle \rangle\} \cup \{\langle c \rangle \mid c \in C_1\}$
 - (b) $C_1 \times \dots \times C_n = \{\langle c \rangle \mid c \in C_1\}$
 - (c) $C_1 + \dots + C_n = \{\langle c \rangle \mid c \in C_1\}$
- (3) If $n > 1$ then
 - (a) $C_1 \times \dots \times C_n \subset C_1 \star \dots \star C_n$
 - (b) $C_1 + \dots + C_n \subset C_1 \star \dots \star C_n$
 - (c) $(C_1 + \dots + C_n) \cap (C_1 \times \dots \times C_n) = \emptyset$

12.2.6 Proof of 12.2.5

These properties immediately follow from definition 12.2.2.

12.3 Concatenation of tuple classes

12.3.1 Remark

Note, that none of our product definitions is associative. For our definition of the cartesian product holds $(A \times B) \times C \neq A \times B \times C \neq A \times (B \times C)$. $(A \times B) \times C$ is a class of pairs, $A \times B \times C$ is a class of triples. Also, the unary cartesian product $\otimes\langle C \rangle$ is the class $\{\langle c \rangle \mid c \in C\}$. And that is different to C itself.

But there is a tradition to define these things differently. Usually — this is a common habit in formal language theory — the difference between say the element c and the unary tuple $\langle c \rangle$ is neglected, a character ‘c’ and a string “c” are considered the same thing. But from a strictly formal point of view or for a computer programmer, this difference is real. And mathematicians that define cartesian products as associative, are not really able to define e.g. the set $\{\langle \langle a, b \rangle, c \rangle \mid a, b, c \in \mathbb{N}\}$ with their formalism.

We next define the *concatenation* \dagger which has the particular property that $(A \times B) \dagger (C \times D) = A \times B \times C \times D$.

12.3.2 Definition

If L_1, \dots, L_n are classes of tuples, then

$$L_1 \dagger \dots \dagger L_n := \prod_{i=1}^n L_i :=$$

$$\{x_1 \dagger \dots \dagger x_n \mid x_1 \in L_1, \dots, x_n \in L_n\}$$

the concatenation of L_1, \dots, L_n .

12.3.3 Lemma

(1) For tuple classes L_1, \dots, L_n we have

(a) $L_1 \dagger \dots \dagger L_n = \{\langle \rangle\}$ for $n = 0$

(b) $L_1 \dagger \dots \dagger L_n = L_1$ for $n = 1$

(2) $(\mathbf{P}(C^*), \{\langle \rangle\}, \dagger)$ is a *monoid*, for every class C , in the sense that

(a) $\textcircled{1} \dagger \textcircled{2} : \mathbf{P}(C^*) \times \mathbf{P}(C^*) \rightarrow \mathbf{P}(C^*)$ is associative

(b) $\{\langle \rangle\}$ is the neutral element for $\textcircled{1} \dagger \textcircled{2}$

(3) For $n \in \mathbb{N}$ classes C_1, \dots, C_n and $0 \leq i < n$ holds

(a) $C_1 \times \dots \times C_n = (C_1 \times \dots \times C_i) \dagger (C_{i+1} \times \dots \times C_n)$

(b) $C_1 \times \dots \times C_n \neq (C_1 \times \dots \times C_i) \times (C_{i+1} \times \dots \times C_n)$

12.3.4 Proof of 12.3.3

In 10.5.1 we defined

$$\xi_1 \dagger \dots \dagger \xi_n := \begin{cases} \langle \rangle & \text{if } n = 0 \\ \xi_1 \dagger (\xi_2 \dagger \dots \dagger \xi_n) & \text{else} \end{cases}$$

We apply this definition and obtain:

(1) For tuple classes L_1, \dots, L_n ,

(a) $n = 0$ implies $L_1 \dagger \dots \dagger L_n = \{\langle \rangle\}$

(b) $n = 1$ implies $L_1 \dagger \dots \dagger L_n = \{\xi_1 \dagger \langle \rangle \mid \xi_1 \in L_1\} = L_1$

(2) If $L, L', L'' \in \mathbf{P}(C^*)$ then

(a) $\textcircled{1} \dagger \textcircled{2}$ is associative, because:

$$\begin{aligned} L \dagger (L' \dagger L'') &= \{\xi \dagger (\xi' \dagger \xi'') \mid \xi \in L, \xi' \in L', \xi'' \in L''\} \\ &= \{(\xi \dagger \xi') \dagger \xi'' \mid \xi \in L, \xi' \in L', \xi'' \in L''\} \\ &= (L \dagger L') \dagger L'' \end{aligned}$$

(b) $\{\langle \rangle\}$ is the neutral element of $\textcircled{1} \dagger \textcircled{2}$, because $L \dagger \{\langle \rangle\} = \{\xi \dagger \langle \rangle \mid \xi \in L\} = L$ and $\{\langle \rangle\} \dagger L = L$, too.

(3) For every $i \in \{0, 1, \dots, n\}$ we have

$$\begin{aligned} &(C_1 \times \dots \times C_i) \dagger (C_{i+1} \times \dots \times C_n) \\ &= \{\langle x_1, \dots, x_i \rangle \dagger \langle x_{i+1}, \dots, x_n \rangle \mid x_1 \in C_1, \dots, x_n \in C_n\} \\ &= \{\langle x_1, \dots, x_i, x_{i+1}, \dots, x_n \rangle \mid x_1 \in C_1, \dots, x_n \in C_n\} \\ &= C_1 \times \dots \times C_n \end{aligned}$$

and

$$\begin{aligned} &(C_1 \times \dots \times C_i) \times (C_{i+1} \times \dots \times C_n) \\ &= \{\langle \langle x_1, \dots, x_i \rangle, \langle x_{i+1}, \dots, x_n \rangle \rangle \mid x_1 \in C_1, \dots, x_n \in C_n\} \\ &\neq \{\langle x_1, \dots, x_i, x_{i+1}, \dots, x_n \rangle \mid x_1 \in C_1, \dots, x_n \in C_n\} \\ &= C_1 \times \dots \times C_n \end{aligned}$$

12.4 Distributivity

12.4.1 Remark distributivity

The *distributivity* of the various products with class operations is e.g. saying that

$$\otimes \begin{bmatrix} a \mapsto \{2\} \\ b \mapsto \{2, 3\} \cap \{3, 4\} \\ c \mapsto \{5\} \end{bmatrix} = \otimes \begin{bmatrix} a \mapsto \{2\} \\ b \mapsto \{2, 3\} \\ c \mapsto \{5\} \end{bmatrix} \cap \otimes \begin{bmatrix} a \mapsto \{2\} \\ b \mapsto \{3, 4\} \\ c \mapsto \{5\} \end{bmatrix}$$

or more general and intuitive

$$\otimes \begin{bmatrix} \vdots & \vdots \\ \vdots & \vdots \\ k \mapsto C \cap D \\ \vdots & \vdots \end{bmatrix} = \otimes \begin{bmatrix} \vdots & \vdots \\ \vdots & \vdots \\ k \mapsto C \\ \vdots & \vdots \end{bmatrix} \cap \otimes \begin{bmatrix} \vdots & \vdots \\ \vdots & \vdots \\ k \mapsto D \\ \vdots & \vdots \end{bmatrix}$$

Similar statements hold for \otimes and \oplus instead of \otimes , and \cup, \cap, \sqcap , and partially for \setminus instead of \cap .

We obtain an elegant formal expression of these distributivity laws when we apply the updater \surd defined in . Recall, that for a given schema $X = [X_i \mid i \in I]$, an identifier k and a class C holds

$$X \surd [k \mapsto C] = [\hat{X}_i \mid i \in I \cup \{k\}] \quad \text{where} \quad \hat{X}_i := \begin{cases} X_i & \text{if } i \neq k \\ C & \text{if } i = k \end{cases}$$

(Note, that it doesn't matter if k is a member of I or not.) A proper form for the previous distributivity law is then

$$\otimes (X \surd [k \mapsto C \cap D]) = \otimes (X \surd [k \mapsto C]) \cap \otimes (X \surd [k \mapsto D])$$

12.4.2 Example distributivity over intersections

$$\begin{aligned} &\otimes \begin{bmatrix} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \cap \{3, 4\} \end{bmatrix} \\ &= \otimes \begin{bmatrix} a \mapsto \{1\} \\ b \mapsto \{3\} \end{bmatrix} \\ &= \left\{ \langle \rangle, [a \mapsto 1], [b \mapsto 3], \left[\begin{array}{l} a \mapsto 1 \\ b \mapsto 3 \end{array} \right] \right\} \\ &= \left\{ \begin{array}{l} \langle \rangle, [a \mapsto 1], \\ [b \mapsto 2], [b \mapsto 3], \end{array} \right\} \cap \left\{ \begin{array}{l} \langle \rangle, [a \mapsto 1], \\ [b \mapsto 3], [b \mapsto 4], \end{array} \right\} \\ &= \left\{ \begin{array}{l} [a \mapsto 1] \\ [b \mapsto 2] \end{array} \right\}, \left\{ \begin{array}{l} [a \mapsto 1] \\ [b \mapsto 3] \end{array} \right\} \right\} \cap \left\{ \begin{array}{l} [a \mapsto 1] \\ [b \mapsto 3] \end{array} \right\}, \left\{ \begin{array}{l} [a \mapsto 1] \\ [b \mapsto 4] \end{array} \right\} \right\} \\ &= \otimes \begin{bmatrix} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \end{bmatrix} \cap \otimes \begin{bmatrix} a \mapsto \{1\} \\ b \mapsto \{3, 4\} \end{bmatrix} \end{aligned}$$

and

$$\begin{aligned}
& \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \cap \{3, 4\} \end{array} \right] \\
= & \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{3\} \end{array} \right] \\
= & \left\{ \left[\begin{array}{l} a \mapsto 1 \\ b \mapsto 3 \end{array} \right] \right\} \\
= & \left\{ \left[\begin{array}{l} a \mapsto 1 \\ b \mapsto 2 \end{array} \right], \left[\begin{array}{l} a \mapsto 1 \\ b \mapsto 3 \end{array} \right] \right\} \cap \left\{ \left[\begin{array}{l} a \mapsto 1 \\ b \mapsto 3 \end{array} \right], \left[\begin{array}{l} a \mapsto 1 \\ b \mapsto 4 \end{array} \right] \right\} \\
= & \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \end{array} \right] \cap \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{3, 4\} \end{array} \right]
\end{aligned}$$

and

$$\begin{aligned}
& \oplus \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \cap \{3, 4\} \end{array} \right] \\
= & \oplus \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{3\} \end{array} \right] \\
= & \{[a \mapsto 1], [b \mapsto 3]\} \\
= & \{[a \mapsto 1], [b \mapsto 2], [b \mapsto 3]\} \cap \{[a \mapsto 1], [b \mapsto 3], [b \mapsto 4]\} \\
= & \oplus \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \end{array} \right] \cap \oplus \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{3, 4\} \end{array} \right]
\end{aligned}$$

12.4.3 Lemma distributivity

Let $X = [X_i | i \in I]$ be a schema and k an identifier.

(1) For every schema $[C_j | j \in J]$ with $J \neq \emptyset$ holds

$$(a) \otimes \left(X \surd \left[k \mapsto \bigcap_{j \in J} C_j \right] \right) = \bigcap_{j \in J} \otimes (X \surd [k \mapsto C_j])$$

$$(b) \oplus \left(X \surd \left[k \mapsto \bigcap_{j \in J} C_j \right] \right) = \bigcap_{j \in J} \oplus (X \surd [k \mapsto C_j])$$

$$(c) \otimes \left(X \surd \left[k \mapsto \bigcap_{j \in J} C_j \right] \right) = \bigcap_{j \in J} \otimes (X \surd [k \mapsto C_j])$$

(2) For every schema $[C_j | j \in J]$ holds

$$(a) \otimes \left(X \surd \left[k \mapsto \bigcup_{j \in J} C_j \right] \right) = \bigcup_{j \in J} \otimes (X \surd [k \mapsto C_j])$$

and if $J \neq \emptyset$ then

$$(b) \oplus \left(X \surd \left[k \mapsto \bigcup_{j \in J} C_j \right] \right) = \bigcup_{j \in J} \oplus (X \surd [k \mapsto C_j])$$

$$(c) \otimes \left(X \surd \left[k \mapsto \bigcup_{j \in J} C_j \right] \right) = \bigcup_{j \in J} \otimes (X \surd [k \mapsto C_j])$$

(3) For every two classes C_1 and C_2 holds

$$\begin{aligned}
& \otimes (X \surd [k \mapsto C_1 \setminus C_2]) \\
= & \otimes (X \surd [k \mapsto C_1]) \setminus \otimes (X \surd [k \mapsto C_2])
\end{aligned}$$

12.4.4 Proof of 12.4.3

Given the schema $X = [X_i | i \in I]$ and the identifier k (which may or may not be a member of I). We put

$$\hat{I} := I \cup \{k\} \quad \bar{I} := I \setminus \{k\} \quad \bar{X} := \mathbf{pr}(X, \bar{I})$$

so that, for every class C ,

$$X \surd [k \mapsto C] = \bar{X} \dot{\vee} [k \mapsto C]$$

(1) Now we also have a given schema $[C_j | j \in J]$ with $J \neq \emptyset$. We put

$$C := \bigcap_{j \in J} C_j$$

$$\hat{X} := X \surd [k \mapsto C]$$

$$= \bar{X} \dot{\vee} [k \mapsto C]$$

$$= \left[i \mapsto \begin{cases} X_i & \text{if } i \neq k \\ C & \text{if } i = k \end{cases} \mid i \in \hat{I} \right]$$

and for each $j \in J$ we put

$$\sigma_j := X \surd [k \mapsto C_j]$$

$$= \bar{X} \dot{\vee} [k \mapsto C_j]$$

$$= \left[i \mapsto \begin{cases} X_i & \text{if } i \neq j \\ C_j & \text{if } i = j \end{cases} \mid i \in \hat{I} \right]$$

With these notations our original statements now become

$$(a) \otimes \hat{X} = \bigcap_{j \in J} \otimes \sigma_j$$

$$(b) \oplus \hat{X} = \bigcap_{j \in J} \oplus \sigma_j$$

$$(c) \otimes \hat{X} = \bigcap_{j \in J} \otimes \sigma_j$$

and their proof is as follows

$\otimes \hat{X}$

$$\begin{aligned}
& = \left\{ [x_i | i \in \hat{I}] \mid \begin{array}{l} x_i \in X_i \text{ for } i \in \bar{I}, \\ x_k \in \bigcap \{C_j \mid j \in J\} \end{array} \right\} \\
& = \bigcap_{j \in J} \left\{ [x_i | i \in \hat{I}] \mid \begin{array}{l} x_i \in X_i \text{ for } i \in \bar{I}, \\ x_k \in C_j \end{array} \right\} \\
& = \bigcap_{j \in J} \otimes \sigma_j
\end{aligned}$$

$\oplus \hat{X}$

$$\begin{aligned}
& = \{[i \mapsto x] \mid i \in \hat{I}, x \in \hat{X}_i\} \\
& = \oplus \bar{X} \cup \oplus [k \mapsto C] \\
& = \oplus \bar{X} \cup \bigcap_{j \in J} \oplus [k \mapsto C_j] \\
& = \bigcap_{j \in J} (\oplus \bar{X} \cup \oplus [k \mapsto C_j]) \\
& = \bigcap_{j \in J} \oplus (\bar{X} \dot{\vee} [k \mapsto C_j]) \\
& = \bigcap_{j \in J} \oplus \sigma_j
\end{aligned}$$

For a proof of (c) let us first note, that for each $L \subseteq \hat{I}$, exactly one of the following two situations is the case:

(i) $k \notin L$ and that implies

$$\mathbf{pr}(\hat{X}, L) = \mathbf{pr}(\bar{X}, L) = \mathbf{pr}(\sigma_j, L) \quad \text{for each } j \in J$$

so that

$$\otimes \mathbf{pr}(\hat{X}, L) = \otimes \mathbf{pr}(\sigma_j, L) = \bigcap_{j \in J} \otimes \mathbf{pr}(\sigma_j, L)$$

because $J \neq \emptyset$ and $\otimes \mathbf{pr}(\sigma_j, L)$ is the same for each $j \in J$.

(ii) $k \in L$ and that implies

$$\begin{aligned}
\mathbf{pr}(\hat{X}, L) &= \mathbf{pr}(\hat{X}, L \setminus \{k\}) \dot{\vee} [k \mapsto C] \\
&= \mathbf{pr}(\hat{X}, L \setminus \{k\}) \dot{\vee} \left[k \mapsto \bigcap_{j \in J} C_j \right]
\end{aligned}$$

so that

$$\begin{aligned}
\otimes \mathbf{pr}(\hat{X}, L) &= \otimes \left(\mathbf{pr}(\hat{X}, L \setminus \{k\}) \dot{\vee} \left[k \mapsto \bigcap_{j \in J} C_j \right] \right) \\
&= \left\{ [x_l | l \in L] \mid \begin{array}{l} x_l \in X_l \text{ for } l \neq k \\ x_k \in \bigcap_{j \in J} C_j \end{array} \right\} \\
&= \bigcap_{j \in J} \left\{ [x_l | l \in L] \mid \begin{array}{l} x_l \in X_l \text{ for } l \neq k \\ x_k \in C_j \end{array} \right\} \\
&= \bigcap_{j \in J} \otimes \mathbf{pr}(\sigma_j, L)
\end{aligned}$$

So anyway, for every $L \subseteq \hat{I}$ we have

$$\otimes \mathbf{pr}(\hat{X}, L) = \bigcap_{j \in J} \otimes \mathbf{pr}(\sigma_j, L)$$

and thus we obtain the following derivation for (c)

$$\begin{aligned}
\otimes \hat{X} &= \bigcup_{L \subseteq \hat{I}} \otimes \mathbf{pr}(\hat{X}, L) && \text{due to 12.1.4} \\
&= \bigcup_{L \subseteq \hat{I}} \bigcap_{j \in J} \otimes \mathbf{pr}(\sigma_j, L) \\
&= \bigcup_{L \subseteq \hat{I}} \bigcap_{j \in J} \left\{ [x_l | l \in L] \mid \begin{array}{l} x_l \in X_l \text{ for } l \in L \setminus \{k\} \\ x_l \in C_j \text{ for } l \in \{k\} \end{array} \right\} \\
&= \left\{ [x_l | l \in L] \mid \begin{array}{l} L \subseteq \hat{I} \\ x_l \in X_l \text{ for } l \in L \setminus \{k\} \\ x_l \in \bigcap_{j \in J} C_j \text{ for } l \in \{k\} \end{array} \right\} \\
&= \bigcap_{j \in J} \bigcup_{L \subseteq \hat{I}} \left\{ [x_l | l \in L] \mid \begin{array}{l} x_l \in X_l \text{ for } l \in L \setminus \{k\} \\ x_l \in C_j \text{ for } l \in \{k\} \end{array} \right\} \\
&= \bigcap_{j \in J} \bigcup_{L \subseteq \hat{I}} \otimes \mathbf{pr}(\sigma_j, L) \\
&= \bigcap_{j \in J} \otimes \sigma_j && \text{due to 12.1.4, again}
\end{aligned}$$

- (2) Again, we have a given schema $[C_j | j \in J]$ and this time we put

$$\begin{aligned}
C &:= \bigcup_{j \in J} C_j \\
\hat{X} &:= X \sqrt{[k \mapsto C]} \\
&= \bar{X} \dot{\vee} [k \mapsto C] \\
&= \left[i \mapsto \begin{cases} X_i & \text{if } i \neq k \\ C & \text{if } i = k \end{cases} \mid i \in \hat{I} \right]
\end{aligned}$$

and for each $j \in J$ we put

$$\begin{aligned}
\sigma_j &:= X \sqrt{[k \mapsto C_j]} \\
&= \bar{X} \dot{\vee} [k \mapsto C_j] \\
&= \left[i \mapsto \begin{cases} X_i & \text{if } i \neq j \\ C_j & \text{if } i = j \end{cases} \mid i \in \hat{I} \right]
\end{aligned}$$

With these notations our statements are

$$(a) \otimes \hat{X} = \bigcup_{j \in J} \otimes \sigma_j$$

That is proven similar to (1)(a):

$$\begin{aligned}
\otimes \hat{X} &= \left\{ [x_i | i \in \hat{I}] \mid \begin{array}{l} x_i \in X_i \text{ for } i \in \bar{I}, \\ x_k \in \bigcup_{j \in J} C_j \end{array} \right\} \\
&= \bigcup_{j \in J} \left\{ [x_i | i \in \hat{I}] \mid \begin{array}{l} x_i \in X_i \text{ for } i \in \bar{I}, \\ x_k \in C_j \end{array} \right\} \\
&= \bigcup_{j \in J} \otimes \sigma_j
\end{aligned}$$

$$(b) \text{ If } J \neq \emptyset \text{ then } \oplus \hat{X} = \bigcup_{j \in J} \oplus \sigma_j.$$

That is true because

$$\begin{aligned}
\oplus \hat{X} &= \{ [i \mapsto x] \mid i \in \hat{I}, x \in \hat{X}_i \} \\
&= \left\{ \frac{[i \mapsto x]}{i \in \bar{I}, x \in \bar{X}_i} \right\} \cup \left\{ [k \mapsto x] \right\}
\end{aligned}$$

$$\begin{aligned}
&= \left\{ \frac{[i \mapsto x]}{i \in \bar{I}, x \in \bar{X}_i} \right\} \cup \bigcup_{j \in J} \left\{ \frac{[k \mapsto x]}{x \in C_j} \right\} \\
&= \bigcup_{j \in J} \left(\left\{ \frac{[i \mapsto x]}{i \in \bar{I}, x \in \bar{X}_i} \right\} \cup \left\{ \frac{[k \mapsto x]}{x \in C_j} \right\} \right) \\
&= \bigcup_{j \in J} \otimes (\bar{X} \dot{\vee} [k \mapsto C_j]) \\
&= \bigcup_{j \in J} \otimes \sigma_j
\end{aligned}$$

$$(c) \text{ If } J \neq \emptyset \text{ then } \otimes \hat{X} = \bigcup_{j \in J} \otimes \sigma_j.$$

Similar to the reasoning for the proof of (1)(c), let us note that for each $L \subseteq \hat{I}$ we have

(i) either $k \notin L$, so that

$$\mathbf{pr}(\hat{X}, L) = \mathbf{pr}(\bar{X}, L) = \mathbf{pr}(\sigma_j, L) \quad \text{for each } j \in J$$

and thus

$$\begin{aligned}
\otimes \mathbf{pr}(\hat{X}, L) &= \otimes \mathbf{pr}(\sigma_j, L) \quad \text{for each } j \in J \\
&= \bigcup_{j \in J} \otimes \mathbf{pr}(\sigma_j, L) \quad \text{because } J \neq \emptyset
\end{aligned}$$

(ii) or $k \in L$, so that

$$\begin{aligned}
\mathbf{pr}(\hat{X}, L) &= \mathbf{pr}(\hat{X}, L \setminus \{k\}) \dot{\vee} [k \mapsto C] \\
&= \mathbf{pr}(\hat{X}, L \setminus \{k\}) \dot{\vee} \left[k \mapsto \bigcup_{j \in J} C_j \right]
\end{aligned}$$

so that

$$\begin{aligned}
\otimes \mathbf{pr}(\hat{X}, L) &= \otimes \left(\mathbf{pr}(\hat{X}, L \setminus \{k\}) \dot{\vee} \left[k \mapsto \bigcup_{j \in J} C_j \right] \right) \\
&= \left\{ [x_l | l \in L] \mid \begin{array}{l} x_l \in X_l \text{ for } l \neq k \\ x_k \in \bigcup_{j \in J} C_j \end{array} \right\} \\
&= \bigcup_{j \in J} \left\{ [x_l | l \in L] \mid \begin{array}{l} x_l \in X_l \text{ for } l \neq k \\ x_k \in C_j \end{array} \right\} \\
&= \bigcup_{j \in J} \otimes \mathbf{pr}(\sigma_j, L)
\end{aligned}$$

So anyway, for every $L \subseteq \hat{I}$ holds

$$\otimes \mathbf{pr}(\hat{X}, L) = \bigcup_{j \in J} \otimes \mathbf{pr}(\sigma_j, L)$$

Using that in the derivation for (c) gives us

$$\begin{aligned}
\otimes \hat{X} &= \bigcup_{L \subseteq \hat{I}} \otimes \mathbf{pr}(\hat{X}, L) && \text{due to 12.1.4} \\
&= \bigcup_{L \subseteq \hat{I}} \bigcup_{j \in J} \otimes \mathbf{pr}(\sigma_j, L) \\
&= \bigcup_{j \in J} \bigcup_{L \subseteq \hat{I}} \otimes \mathbf{pr}(\sigma_j, L) \\
&= \bigcup_{j \in J} \otimes \sigma_j && \text{due to 12.1.4, again}
\end{aligned}$$

- (3) Finally, we take any two classes C_1 and C_2 and obtain

$$\begin{aligned}
&\otimes (X \sqrt{[k \mapsto C_1 \setminus C_2]}) \\
&= \left\{ [x_i \mapsto i \in \hat{I}] \mid \begin{array}{l} i \in X_i \text{ for } i \in \bar{I}, \\ x_k \in C_1 \setminus C_2 \end{array} \right\} \\
&= \left\{ \frac{[x_i \mapsto i \in \hat{I}]}{i \in X_i \text{ for } i \in \bar{I},} \right\} \setminus \left\{ \frac{[x_i \mapsto i \in \hat{I}]}{x_k \in C_1} \right\} \\
&= \otimes (X \sqrt{[k \mapsto C_1]}) \setminus \otimes (X \sqrt{[k \mapsto C_2]})
\end{aligned}$$

12.4.5 Remark and example —distributivity over differences—

Note, that there is neither a statement

$$\otimes X = \otimes X \surd [k \mapsto C_1] \setminus \otimes X \surd [k \mapsto C_2]$$

nor

$$\oplus X = \oplus X \surd [k \mapsto C_1] \setminus \oplus X \surd [k \mapsto C_2]$$

in 12.4.3(3), because that would be false in general. Let us illustrate that with an example, similar to the one in 12.4.2.

The cartesian product is distributive:

$$\begin{aligned} & \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \setminus \{3, 4\} \end{array} \right] \\ = & \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2\} \end{array} \right] \\ = & \left\{ \begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 2 \end{array} \right\} \\ = & \left\{ \begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 2 \end{array} \right\}, \left\{ \begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 3 \end{array} \right\} \setminus \left\{ \begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 3 \end{array} \right\}, \left\{ \begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 4 \end{array} \right\} \\ = & \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \end{array} \right] \setminus \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{3, 4\} \end{array} \right] \end{aligned}$$

But that neither holds for the star product

$$\begin{aligned} & \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \setminus \{3, 4\} \end{array} \right] \\ = & \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2\} \end{array} \right] \\ = & \left\{ \langle \rangle, [\mathbf{a} \mapsto 1], [\mathbf{b} \mapsto 2], \left[\begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 2 \end{array} \right] \right\} \\ \neq & \left\{ [\mathbf{b} \mapsto 2], \left[\begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 2 \end{array} \right] \right\} \\ = & \left\{ \begin{array}{l} \langle \rangle, [\mathbf{a} \mapsto 1], \\ [\mathbf{b} \mapsto 2], [\mathbf{b} \mapsto 3], \end{array} \right\} \setminus \left\{ \begin{array}{l} \langle \rangle, [\mathbf{a} \mapsto 1], \\ [\mathbf{b} \mapsto 3], [\mathbf{b} \mapsto 4], \end{array} \right\} \\ = & \left\{ \begin{array}{l} \left[\begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 2 \end{array} \right], \left[\begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 3 \end{array} \right] \end{array} \right\} \setminus \left\{ \begin{array}{l} \left[\begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 3 \end{array} \right], \left[\begin{array}{l} \mathbf{a} \mapsto 1 \\ \mathbf{b} \mapsto 4 \end{array} \right] \end{array} \right\} \\ = & \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \end{array} \right] \setminus \otimes \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{3, 4\} \end{array} \right] \end{aligned}$$

nor for the coproduct

$$\begin{aligned} & \oplus \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \setminus \{3, 4\} \end{array} \right] \\ = & \oplus \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2\} \end{array} \right] \\ = & \{[\mathbf{a} \mapsto 1], [\mathbf{b} \mapsto 2]\} \\ \neq & \{[\mathbf{b} \mapsto 2]\} \\ = & \{[\mathbf{a} \mapsto 1], [\mathbf{b} \mapsto 2], [\mathbf{b} \mapsto 3]\} \setminus \{[\mathbf{a} \mapsto 1], [\mathbf{b} \mapsto 3], [\mathbf{b} \mapsto 4]\} \\ = & \oplus \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{2, 3\} \end{array} \right] \setminus \oplus \left[\begin{array}{l} a \mapsto \{1\} \\ b \mapsto \{3, 4\} \end{array} \right] \end{aligned}$$

12.5 Power products

12.5.1 Definition power products

For all classes C and I , where at least one of the two is non-empty, we define

$$C^I := \otimes [C | I] = \left\{ \frac{[x_i | i \in I]}{x_i \in C \text{ for all } i \in I} \right\}$$

the I -th power (product) of C .

For every every non-empty class C and all $n \in \mathbb{N}$ we define

$$C^n := \otimes \langle C | n \rangle = \left\{ \frac{\langle x_1, \dots, x_n \rangle}{x_i \in C \text{ for } i = 1, \dots, n} \right\}$$

the n -th power (product) of C or

the n -tuple class on C .

And for every non-empty class C , we define

$$C^* := \bigcup_{n \in \mathbb{N}} C^n = \left\{ \frac{\langle x_1, \dots, x_n \rangle}{n \in \mathbb{N}, \forall i \in I. x_i \in C} \right\}$$

the Kleene closure or the tuple class on C .

12.5.2 Remark power products with empty classes

We have the following special cases:

- (1) $I = \emptyset$ and $C \neq \emptyset$. Then $C^I = \{\langle \rangle\}$, i.e. the power product is made of one element, the empty tuple.
- (2) $I \neq \emptyset$ and $C = \emptyset$. Then $C^I = \emptyset$.

Now if $I = C = \emptyset$, both these cases would apply and result in an impossible situation. Therefore, $I = C = \emptyset$ is excluded from the definition of C^I .

12.5.3 Example power product

$$\begin{aligned} \{a, b\}^{\{c, d\}} &= \left\{ \left[\begin{array}{l} c \mapsto a \\ d \mapsto a \end{array} \right], \left[\begin{array}{l} c \mapsto b \\ d \mapsto a \end{array} \right], \right. \\ & \left. \left[\begin{array}{l} c \mapsto a \\ d \mapsto b \end{array} \right], \left[\begin{array}{l} c \mapsto b \\ d \mapsto b \end{array} \right] \right\} \\ \{a, b\}^2 &= \{ \langle a, a \rangle, \langle b, a \rangle, \langle a, b \rangle, \langle b, b \rangle \} \\ \{a, b\}^* &= \left\{ \begin{array}{l} \langle \rangle, \\ \langle a \rangle, \langle b \rangle, \\ \langle a, a \rangle, \langle b, a \rangle, \langle a, b \rangle, \langle b, b \rangle, \\ \langle a, a, a \rangle, \langle b, a, a \rangle, \langle a, b, a \rangle, \dots, \\ \vdots \end{array} \right\} \end{aligned}$$

12.5.4 Remark

Usually, the expression C^I for two classes C and I is defined to be the class of all functions with domain I and codomain C , i.e.

$$C^I := \{f | f : I \rightarrow C\}$$

In our design however, this is actually

$$C^I := \{f : I \rightarrow D \mid D \subseteq C \text{ and } f \text{ is surjective}\}$$

But there is an obvious bijection between these two function classes and usually no context in practice, where this difference in definition becomes significant. So we will allow to consider our slightly modified definition as conform with the common standard.

The notation C^I is tradition in mathematics. Most likely it is again motivated by the following statement about its cardinalities in an attempt to merge set theory and arithmetic.

12.5.5 Remark cardinality of power products

For all classes C and I holds

$$\mathbf{card}(C^I) = \mathbf{card}(C)^{\mathbf{card}(I)}$$

where the right hand side denotes the power operation of (possibly infinite) cardinal numbers. Note that this correspondence even holds for the case $C = I = \emptyset$, since neither \emptyset^\emptyset nor 0^0 are defined.

In particular, the case $I = \{1, \dots, n\}$ for some $n \in \mathbb{N}$ and C non-empty makes

$$\mathbf{card}(C^n) = (\mathbf{card}(C))^n$$

13 Digressions

13.0.6 Introduction

In the first of the two following digressions we generalize the binary product and coproduct definition towards a “product construction” and “coproduct construction”. This kind of abstraction is often done in category theory, for example. Our definitions depart slightly from the standard ones, but we will find that the differences are insignificant. In other words, our binary product (as defined in 12.2.2) is a proper product construction, and our coproduct (defined in 12.2.2 as well) is a proper coproduct construction.

In the second digression in subsection 13.2, we introduce a practical application for schema coproducts: the definition of formal languages or data structures.

13.1 Digression: Product and coproduct constructions

13.1.1 Definition

Given two classes A and B .

(1) A (binary) (cartesian) product construction for A and B comprises

- (i) P the so-called *product class*
- (ii) $p_1 : P \rightarrow A$ the *first projector*
- (iii) $p_2 : P \rightarrow B$ the *second projector*
- (iv) $\textcircled{1} \cdot \textcircled{2} : A \times B \rightarrow P$ the *constructor*

such that

- (a) $p_1(a \cdot b) = a$ for all $a \in A$ and $b \in B$
- (b) $p_2(a \cdot b) = b$ for all $a \in A$ and $b \in B$

(2) A (binary) coproduct construction for A and B comprises

- (i) S the so-called *coproduct class* (or *sum* or *disjunct union*)
- (ii) $i_1 : A \rightarrow S$ the *first injector*
- (iii) $i_2 : B \rightarrow S$ the *second injector*
- (iv) $\delta : S \rightarrow ((\{1\} \times A) \uplus (\{2\} \times B))$ the *destructor*

such that

- (a) $\delta(i_1(a)) = \langle 1, a \rangle$ for all $a \in A$
- (b) $\delta(i_2(b)) = \langle 2, b \rangle$ for all $b \in B$

13.1.2 Remark

(i) In category theory and other mathematical branches, a “product” usually is defined as the combination of P , p_1 and p_2 , the “constructor” is not mentioned explicitly.

²⁸ Recall 5.4.1, that $\langle a, b \rangle(1) = a$ and $\langle a, b \rangle(2) = b$ for arbitrary pairs $\langle a, b \rangle$.

²⁹ Recall 5.6.13, that $\mathbf{sing}(C)$ denotes the unique member of a singleton C ; and 5.6.6, that $\nabla \mathcal{K}$ is opposition of a class family \mathcal{K} ; in particular $\nabla \{C_1, C_2\} = C_1 \nabla C_2 = (C_1 \setminus C_2) \cup (C_2 \setminus C_1)$.

Similarly, a “coproduct” is made of P , i_1 and i_2 . A “destructor” is absent.

(ii) The default product construction is the one, where the constructor is the identity (see ??).

Similarly, the most common coproduct construction is the one where the deconstructor is the identity (see 13.1.5).

(iii) A “good” product and coproduct construction usually have the additional property, that they motivate the product and sum of cardinal numbers, respectively. More precisely, if A and B are given classes and P and S are their product and coproduct classes, then $\mathbf{card}(P) = \mathbf{card}(A) \cdot \mathbf{card}(B)$ and $\mathbf{card}(S) = \mathbf{card}(A) + \mathbf{card}(B)$. One way to characterize such a “good” product or coproduct is to demand that the constructor or deconstructor must be a bijection.

13.1.3 Example

The default product construction of given classes A and B is the ordinary cartesian product itself, i.e.²⁸

(i) $A \times B$ is the product class,

(ii) $\textcircled{1}(1) : A \times B \rightarrow A$ is the first and

(iii) $\textcircled{2}(2) : A \times B \rightarrow B$ is the second projection.

(iv) The constructor is of course the identity function on $A \times B$.

13.1.4 Example

The ordinary cartesian product is based on the tuple concept; but there is also a set version.²⁹ For given classes A and B their Kuratowski product construction comprises:

(i) $P := \{\{\{a\}, \{a, b\}\} \mid a \in A, b \in B\}$ is the product class

(ii) $p_1(\pi) := \mathbf{sing}(\bigcap \pi)$ for each $\pi \in P$ is the first projection. So, $p_1(\{\{a\}, \{a, b\}\}) = \mathbf{sing}(\{a\}) = a$.

(iii) $p_2(\pi) := \mathbf{sing}(\bigvee \pi)$ for each $\pi \in P$ is the second projection. So, $p_2(\{\{a\}, \{a, b\}\}) = \mathbf{sing}(\{a\} \nabla \{a, b\}) = \mathbf{sing}(\{b\}) = b$.

(iv) $a \cdot b := \{\{a\}, \{a, b\}\}$ for $a \in A$ and $b \in B$ is the constructor

13.1.5 Example

The default coproduct construction for given A and B puts

(i) $S := (\{1\} \times A) \uplus (\{2\} \times B)$ as the coproduct class

(ii) $i_1(a) := \langle 1, a \rangle$ for each $a \in A$ is the first injection

(iii) $i_2(b) := \langle 2, a \rangle$ for each $b \in B$ is the second injection

(iv) $\delta(n, x) := \langle n, x \rangle$ is the deconstructor

13.1.6 Example

The binary version of our coproduct definition in 12.2.2 is a coproduct construction in the sense of 13.1.1. For two given classes A and B we put

$S := A + B = \{[1 \mapsto a] \mid a \in A\} \uplus \{[2 \mapsto b] \mid b \in B\}$ as the coproduct class

(ii) $i_1(a) := [1 \mapsto a]$ for all $a \in A$ is the first injection

(iii) $i_2(b) := [2 \mapsto b]$ for all $b \in B$ is the first injection

(iv) $\delta([n \mapsto x]) := \langle n, x \rangle$ for all $[n \mapsto x] \in S$ is the deconstructor.

13.2 Digression: Datastructures and formal languages as coproducts of schemas

13.2.1 Introduction

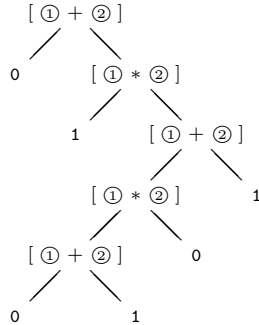
Often in mathematics and metamathematics, we need to define *formal languages*. In computer science, these expression classes are often called *data structures*.³⁰

Suppose, we want to define “(simple) arithmetic terms”. This term class **Arith** is a formal language and should contain expressions like

$$[0 + [[[0 + 1] * 0] + 1]]$$

i.e. expressions that start with *zero* and *unit* and allow more complex constructions by means of *addition* and *multiplication*, embraced by square brackets.

We already discussed the mathematical nature of expressions in 5.2, where we emphasized (5.2.3(2)) that expressions are actually parse trees and the example above is just a camouflage for something like



We are going to introduce a convenient and elegant “coproduct definition” for data structures in general. For **Arith** such a definition is given in 13.2.4. But before that, let us start with a “vulgar definition” for **Arith** in 13.2.2.

13.2.2 vulgar definition of arithmetic terms

The *arithmetic term* class **Arith** is defined to be the smallest class containing the following expressions:

- (1) 0 (zero)
 (2) 1 (unit)

- (3) $[\xi + v]$ for all $\xi, v \in \mathbf{Arith}$ (addition)
 (4) $[\xi * v]$ for all $\xi, v \in \mathbf{Arith}$ (multiplication)

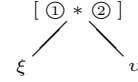
13.2.3 Remark

The arithmetic term class is a union of four disjunct classes

$$\mathbf{Arith} = \{0\} \uplus \{1\} \uplus \left\{ \frac{[\xi + v]}{\xi, v \in \mathbf{Arith}} \right\} \uplus \left\{ \frac{[\xi * v]}{\xi, v \in \mathbf{Arith}} \right\}$$

Even more so — and that is essential for data structures in computer programs — we can immediately “see” for each arithmetic term, to which of the four classes it belongs. Our example arithmetic term in 13.2.1 is an *addition* term, because “Ⓛ + Ⓜ” is the root of parse tree.

This is a characteristic property of data structures compare to other classes in general: each element is a transparent construction; either a primitive one, like 0, or a more complex one like $[\xi * v]$. And the complex one is a combination of an operator symbol or constructor “[Ⓛ * Ⓜ]” and an argument tuple, in this case “ $\langle \xi, v \rangle$ ”. And since “[$\xi * v$]” is in fact just the linear representation for the tree node



the mathematical formalization would be more appropriately something like a constructor–arguments pair

$$\langle [Ⓛ * Ⓜ], \langle \xi, v \rangle \rangle$$

or a singular record

$$\langle [Ⓛ * Ⓜ] \mapsto \langle \xi, v \rangle \rangle$$

We take this record representation “[$[Ⓛ * Ⓜ] \mapsto \langle \xi, v \rangle$]” as the general form of expressions, i.e. elements of data structures. And similar to the definition of “constants” as “functions with zero arguments”, we can also subsume primitive expressions under this form and represent “0” by “[$0 \mapsto \langle \rangle$]”.

Applying this approach to **Arith**, we now obtain the following disjunct union

$$\begin{aligned} \mathbf{Arith} = & \{[0 \mapsto \langle \rangle]\} \uplus \\ & \{[1 \mapsto \langle \rangle]\} \uplus \\ & \{[[Ⓛ + Ⓜ] \mapsto \langle \xi, v \rangle] \mid \xi, v \in \mathbf{Arith}\} \uplus \\ & \{[[Ⓛ * Ⓜ] \mapsto \langle \xi, v \rangle] \mid \xi, v \in \mathbf{Arith}\} \end{aligned}$$

And since

$$\begin{aligned} \{[0 \mapsto \langle \rangle]\} &= \oplus [0 \mapsto \{\langle \rangle\}] \\ \{[1 \mapsto \langle \rangle]\} &= \oplus [1 \mapsto \{\langle \rangle\}] \\ \{[[Ⓛ + Ⓜ] \mapsto \langle \xi, v \rangle] \mid \xi, v \in \mathbf{Arith}\} &= \oplus [[Ⓛ + Ⓜ] \mapsto \mathbf{Arith}^2] \\ \{[[Ⓛ * Ⓜ] \mapsto \langle \xi, v \rangle] \mid \xi, v \in \mathbf{Arith}\} &= \oplus [[Ⓛ * Ⓜ] \mapsto \mathbf{Arith}^2] \end{aligned}$$

³⁰ We only think of formal languages as *context-free* languages. There is of course a more general notion, as any standard book on *formal languages* tells us. But the syntax of most mathematical formalisms or computer languages is indeed definable by a context-free grammar.

with $\mathbf{Arith}^2 = \mathbf{Arith} \times \mathbf{Arith}$ as defined in 12.5.1, we obtain

$$\begin{aligned} \mathbf{Arith} &= \oplus[0 \mapsto \{\langle \rangle\}] \uplus \\ &\oplus[1 \mapsto \{\langle \rangle\}] \uplus \\ &\oplus[[\textcircled{1} + \textcircled{2}] \mapsto \mathbf{Arith}^2] \uplus \\ &\oplus[[\textcircled{1} * \textcircled{2}] \mapsto \mathbf{Arith}^2] \\ &= \oplus \left[\begin{array}{l} 0 \mapsto \{\langle \rangle\} \\ 1 \mapsto \{\langle \rangle\} \\ [\textcircled{1} + \textcircled{2}] \mapsto \mathbf{Arith}^2 \\ [\textcircled{1} * \textcircled{2}] \mapsto \mathbf{Arith}^2 \end{array} \right] \end{aligned}$$

And that is the final version of the following “coproduct definition” of arithmetic terms.

13.2.4 Coproduct definition of arithmetic terms

The *arithmetic term* class is defined as

$$\mathbf{Arith} := \oplus \left[\begin{array}{l} 0 \mapsto \{\langle \rangle\} \\ 1 \mapsto \{\langle \rangle\} \\ [\textcircled{1} + \textcircled{2}] \mapsto \mathbf{Arith}^2 \\ [\textcircled{1} * \textcircled{2}] \mapsto \mathbf{Arith}^2 \end{array} \right]$$

13.2.5 Remark

As a second example for the definition of an expression class, let us consider the definition of propositional formulas. If a given class A contains a and b , an example of such a propositional formula on A shall be given by

$$\neg[\neg a \wedge [0 \vee \neg[0 \vee 1] \vee a] \wedge 1 \wedge b]$$

These kind of expressions are covered by the following definition.

13.2.6 Coproduct definition of propositional formulas

For every class A of identifiers we define

$$\mathbf{Form}(A) := \oplus \left[\begin{array}{l} 0 \mapsto \{\langle \rangle\} \\ 1 \mapsto \{\langle \rangle\} \\ \textcircled{1} \mapsto A \\ \neg \textcircled{1} \mapsto \mathbf{Form}(A) \\ [\textcircled{1} \wedge \dots \wedge \textcircled{n}] \mapsto \mathbf{Form}(A)^* \\ [\textcircled{1} \vee \dots \vee \textcircled{n}] \mapsto \mathbf{Form}(A)^* \end{array} \right]$$

the *propositional formula* class on A .

13.2.7 Remark

This definition of $\mathbf{Form}(A)$ is another example of a schema coproduct definition. It is recursive, as it is typical for many data structures, but that does not cause a problem whatsoever. And again, the whole class is a disjoint union of six different kinds of constructions: the *zero bit*, the *unit bit*, the *atomic formula*, the *negation*, the *conjunction* and the *disjunction*, respectively.

Once again, there is a precise tree form and a convenient written version of each construction:

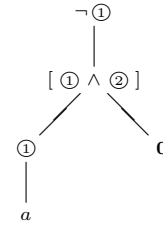
- Each *zero bit* formula from $\oplus[0 \mapsto \{\langle \rangle\}]$ has the precise form $[0 \mapsto \langle \rangle]$, which is simply written “0”, as usual.
- Each *negation* formula, each element from $\oplus[\neg \textcircled{1} \mapsto \mathbf{Form}(A)]$ has the precise form $[\neg \textcircled{1} \mapsto \varphi]$, for some $\varphi \in \mathbf{Form}(A)$, and the usual written representation “ $\neg\varphi$ ”.
- Each *conjunction* formula from $\oplus[[\textcircled{1} \wedge \dots \wedge \textcircled{n}] \mapsto \mathbf{Form}(A)^*]$ has the actual form $[[\textcircled{1} \wedge \dots \wedge \textcircled{n}] \mapsto \langle \varphi_1, \dots, \varphi_n \rangle]$, for some $\langle \varphi_1, \dots, \varphi_n \rangle \in \mathbf{Form}(A)^*$, and is usually written as “[$\varphi_1 \wedge \dots \wedge \varphi_n$]”.
- Each *atomic* formula is an element of $\oplus[\textcircled{1} \mapsto A]$. Accordingly, an atomic formula has the form $[\textcircled{1} \mapsto a]$, for some $a \in A$, but it is usually written as “ a ” itself.

13.2.8 Example

Alltogether, a propositional formula like $\neg[a \wedge 0]$ has the actual form

$$[\neg \textcircled{1} \mapsto [[\textcircled{1} \wedge \textcircled{2}] \mapsto \langle [\textcircled{1} \mapsto a], [0 \mapsto \langle \rangle] \rangle]]$$

which is the formalization of the tree



14 Order structures on schemas

14.1 The operations

14.1.1 Definition

For two schemas $X = [X_i | i \in I]$ and $Y = [Y_i | i \in I]$ with identical index class I we define

$X \subseteq Y$:iff $X_i \subseteq Y_i$ for all $i \in I$	<u>inclusion</u>
$X \not\subseteq Y$:iff $X_i \not\subseteq Y_i$ for some $i \in I$	
$X \cap Y$	$:= [X_i \cap Y_i i \in I]$	<u>intersection</u>
$X \cup Y$	$:= [X_i \cup Y_i i \in I]$	<u>union</u>
$X \setminus Y$	$:= [X_i \setminus Y_i i \in I]$	<u>difference</u>

For every non-empty class \mathcal{Y} of schemas, where all members have the same index class I , we define

$\bigcap \mathcal{Y}$	$:= \bigcap_{Y \in \mathcal{Y}} Y$	$:= [\bigcap \{Y_i Y \in \mathcal{Y}\} i \in I]$	
			the <u>(big) intersection</u> of \mathcal{Y}
$\bigcup \mathcal{Y}$	$:= \bigcup_{Y \in \mathcal{Y}} Y$	$:= [\bigcup \{Y_i Y \in \mathcal{Y}\} i \in I]$	
			the <u>(big) union</u> of \mathcal{Y}

14.1.2 Remark

Note, that $\bigcap \mathcal{Y}$ is defined iff $\mathcal{Y} \neq \emptyset$.

14.1.3 Example

Given two schemas

$$X = \begin{bmatrix} p \mapsto \{a, b, c\} \\ q \mapsto \emptyset \\ r \mapsto \{a\} \end{bmatrix} \quad Y = \begin{bmatrix} p \mapsto \{a, b\} \\ q \mapsto \{a, b\} \\ r \mapsto \{a, b\} \end{bmatrix}$$

with the same index class $I = \{p, q, r\}$. Then

$$X \not\subseteq Y \quad \text{and} \quad Y \not\subseteq X$$

and

$$\begin{aligned} X \cap Y &= \begin{bmatrix} p \mapsto \{a, b\} \\ q \mapsto \emptyset \\ r \mapsto \{a\} \end{bmatrix} & X \cup Y &= \begin{bmatrix} p \mapsto \{a, b, c\} \\ q \mapsto \{a, b\} \\ r \mapsto \{a, b\} \end{bmatrix} \\ X \setminus Y &= \begin{bmatrix} p \mapsto \{c\} \\ q \mapsto \emptyset \\ r \mapsto \emptyset \end{bmatrix} & Y \setminus X &= \begin{bmatrix} p \mapsto \emptyset \\ q \mapsto \{a, b\} \\ r \mapsto \{b\} \end{bmatrix} \end{aligned}$$

14.2 The included schema class

14.2.1 Definition

If $X = [X_i | i \in I]$ is a schema, then

$$\mathbf{Incl}(X) := \{[Y_i | i \in I] \mid Y_i \subseteq X_i \text{ for all } i \in I\}$$

is the included schema class or inclusions of X .

14.2.2 Example

Taking the schema

$$X = \begin{bmatrix} p \mapsto \{a, b\} \\ q \mapsto \emptyset \\ r \mapsto \{a\} \end{bmatrix}$$

then

$$\mathbf{Incl}(X) = \left\{ \begin{array}{l} \begin{bmatrix} p \mapsto \emptyset \\ q \mapsto \emptyset \\ r \mapsto \emptyset \end{bmatrix}, \begin{bmatrix} p \mapsto \{a\} \\ q \mapsto \emptyset \\ r \mapsto \emptyset \end{bmatrix}, \begin{bmatrix} p \mapsto \{b\} \\ q \mapsto \emptyset \\ r \mapsto \emptyset \end{bmatrix}, \\ \begin{bmatrix} p \mapsto \{a, b\} \\ q \mapsto \emptyset \\ r \mapsto \emptyset \end{bmatrix}, \begin{bmatrix} p \mapsto \emptyset \\ q \mapsto \emptyset \\ r \mapsto \{a\} \end{bmatrix}, \begin{bmatrix} p \mapsto \{a\} \\ q \mapsto \emptyset \\ r \mapsto \{a\} \end{bmatrix}, \\ \begin{bmatrix} p \mapsto \{b\} \\ q \mapsto \emptyset \\ r \mapsto \{a\} \end{bmatrix}, \begin{bmatrix} p \mapsto \{a, b\} \\ q \mapsto \emptyset \\ r \mapsto \{a\} \end{bmatrix} \end{array} \right\}$$

14.3 The boolean algebra of inclusions

14.3.1 Definition

The inclusion algebra of a given schema $X = [X_i | i \in I]$ is defined as

$$\mathfrak{Incl}(X) := \langle \mathbf{Incl}(X), \subseteq, [\emptyset | I], X, \cap, \cup, \setminus, \mathfrak{C} \rangle$$

where \cap is defined on its whole domain $\mathbf{P}(\mathbf{Incl}(X))$ by putting $\bigcap \emptyset := X$ and $\mathfrak{C}Y := X \setminus Y$ is the complement of every $Y \in \mathbf{Incl}(X)$.

14.3.2 Theorem

$\mathfrak{Incl}(X)$ is a complete boolean algebra, for every schema X .

14.3.3 Proof of 14.3.2

Let $W, Y, Z \in \mathbf{Incl}(X)$, then

- (a) $(W \subseteq Y \text{ and } Y \subseteq Z)$ implies $(W_i \subseteq Y_i \subseteq Z_i \text{ for all } i \in I)$ implies $(W_i \subseteq Z_i \text{ for all } i \in I)$ implies $W \subseteq Z$. \subseteq is transitive.
- (b) $W \subseteq W$, because $W_i \subseteq W_i$ for all $i \in I$. \subseteq is reflexive.
- (c) $(W \subseteq Y \text{ and } Y \subseteq W)$ implies $(W_i = Y_i \text{ for all } i \in I)$ implies $W = Y$. \subseteq is antisymmetric.

Altogether, \subseteq is a (partial) order relation on $\mathbf{Incl}(X)$. Furthermore

- (d) $[\emptyset|I] \subseteq W$, because $\emptyset \subseteq W_i$ for all $i \in I$. $[\emptyset|I]$ is the least element.
- (e) $W \subseteq X$, because $W_i \subseteq X_i$ for all $i \in I$. X is the greatest element.

For every $\mathcal{Y} \subseteq \mathbf{Incl}(X)$

- (f) $\bigcap \mathcal{Y} \subseteq Y$ for every $Y \in \mathcal{Y}$. And for every $Z \in \mathbf{Incl}(X)$ holds: If $Z \subseteq Y$ for all $Y \in \mathcal{Y}$, then $Z_i \subseteq \bigcap \{Y_i \mid Y \in \mathcal{Y}\}$ for all $i \in I$, and so $Z \subseteq \bigcap \mathcal{Y}$. In other words, $\bigcap \mathcal{Y}$ is the greatest lower bound of \mathcal{Y} .
- (g) $\bigcup \mathcal{Y}$ is the least upper bound of \mathcal{Y} , similar to (f).

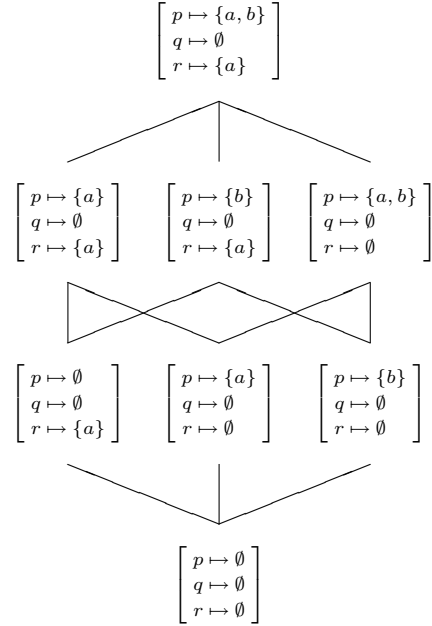
There is $\textcircled{1} \cap \textcircled{2} = \bigcap \{\textcircled{1}, \textcircled{2}\}$ and $\textcircled{1} \cup \textcircled{2} = \bigcup \{\textcircled{1}, \textcircled{2}\}$. So far, we have a complete lattice, which is distributive, because the underlying class algebra is distributive. And finally, for every $Y \in \mathbf{Incl}(X)$ we have

- (h) $(\mathcal{C}Y) \cap Y = (X \setminus Y) \cap Y = [\emptyset|I]$ and $(\mathcal{C}Y) \cup Y = (X \setminus Y) \cup Y = X$. That makes \mathcal{C} the complement operator.

That completes the proof.

14.3.4 Example

If we take the schema X again from the previous example 14.2.2, the order diagram of the complete boolean lattice on $\mathbf{Incl}(X)$ is



Part VI

Graphs and their distinct products

15 Schemas and graphs

a	b	d
5	3	
4		
5	7	1
2		2

15.1 Schemas of record classes

Next, combine the entries of each column as a separate class.

a	b	d
5	3	
4		
5	7	1
2		2
$\{2, 4, 5\}$	$\{3, 7\}$	$\{1, 2\}$

15.1.1 Definition

Let Ξ be a class of records. We define

$$\textcircled{\Xi} := \bigcup \{ \text{dom}(\xi) \mid \xi \in \Xi \}$$

the attribute class of Ξ

$$\text{dom}_\alpha(\Xi) := \{ \xi(\alpha) \mid \xi \in \Xi, \alpha \in \text{dom}(\xi) \}$$

the α -domain of Ξ , for each $\alpha \in \textcircled{\Xi}$

$$\mathbf{x}(\Xi) := [\text{dom}_\alpha(\Xi) \mid \alpha \in \textcircled{\Xi}]$$

the schema of Ξ

The schema of Ξ is finally given by

$$\mathbf{x}(\Xi) = \left[\begin{array}{l} a \mapsto \{2, 4, 5\} \\ b \mapsto \{3, 7\} \\ d \mapsto \{1, 2\} \end{array} \right]$$

15.1.2 Example

Given the record class

$$\Xi = \left\{ \langle \rangle, \left[\begin{array}{l} a \mapsto 5 \\ b \mapsto 3 \end{array} \right], \left[a \mapsto 4 \right], \left[\begin{array}{l} a \mapsto 5 \\ b \mapsto 7 \\ d \mapsto 1 \end{array} \right], \left[\begin{array}{l} a \mapsto 2 \\ d \mapsto 2 \end{array} \right] \right\}$$

then

$$\begin{aligned} \textcircled{\Xi} &= \emptyset \cup \{a, b\} \cup \{a\} \cup \{a, b, d\} \cup \{a, d\} \\ &= \{a, b, d\} \end{aligned}$$

$$\text{dom}_a(\Xi) = \{2, 4, 5\}$$

$$\text{dom}_b(\Xi) = \{3, 7\}$$

$$\text{dom}_d(\Xi) = \{1, 2\}$$

$$\mathbf{x}(\Xi) = \left[\begin{array}{l} a \mapsto \{2, 4, 5\} \\ b \mapsto \{3, 7\} \\ d \mapsto \{1, 2\} \end{array} \right]$$

Note, that $\mathbf{x}(\Xi)$ is proper and $\Xi \subseteq \textcircled{\mathbf{x}(\Xi)}$.

15.1.3 Remark Table construction

Consider the record class Ξ from the previous example 15.1.2 again. The construction of its schema can be done with a table. First, list each record of Ξ as a row in a table, the tables head row is the attribute class. We obtain

15.1.4 Lemma

- (1) $\mathbf{x}(\Xi)$ is a proper schema, for every record class Ξ .
- (2) $\Xi \subseteq \textcircled{\mathbf{x}(\Xi)}$, for every record class Ξ .
- (3) For every schema X holds
 - (a) $\mathbf{x}(\textcircled{X}) = X$
 - (b) $\mathbf{x}(\oplus X) = X$
 And if X is proper, then
 - (c) $\mathbf{x}(\otimes X) = X$

- (4) For every schema X and record class Ξ holds:

$$\Xi \subseteq \textcircled{X} \quad \text{iff} \quad \mathbf{x}(\Xi) \subseteq \text{pr}(X, \text{dom}(X) \cap \textcircled{\Xi})$$

15.1.5 Proof of 15.1.4

- (1) Obviously, $\mathbf{x}(\Xi)$ is a well defined schema. If $\textcircled{\Xi} = \emptyset$, then $\mathbf{x}(\Xi) = \langle \rangle$, which is a proper schema. If $\textcircled{\Xi} \neq \emptyset$, then for every $\alpha \in \textcircled{\Xi}$ holds: $\text{dom}_\alpha(\Xi) = \{ \xi(\alpha) \mid \xi \in \Xi, \alpha \in \text{dom}(\xi) \} \neq \emptyset$. So in every case, $\mathbf{x}(\Xi)$ is proper.
- (2) According to definition 15.1.1 and 12.1.2 we have

$$\begin{aligned} \textcircled{\mathbf{x}(\Xi)} &= \textcircled{[\text{dom}_\alpha(\Xi) \mid \alpha \in \textcircled{\Xi}]} \\ &= \textcircled{\left\{ [\xi_j \mid j \in J] \mid \begin{array}{l} J \subseteq \textcircled{\Xi} \text{ and} \\ \xi_j \in \text{dom}_j(\Xi) \text{ for all } j \in J \end{array} \right\}} \end{aligned}$$

So let $v = [v_j \mid j \in J]$. $\textcircled{\Xi} = \bigcup \{ \text{dom}(\xi) \mid \xi \in \Xi \}$ implies $J \subseteq \textcircled{\Xi}$. And for each $j \in J$, $\text{dom}_j(\Xi) = \{ \xi(j) \mid \xi \in \Xi, j \in \text{dom}(\xi) \}$ implies $v_j \in \text{dom}_j(\Xi)$. Thus $v \in \textcircled{\mathbf{x}(\Xi)}$.

- (3) Let $X = [X_i \mid i \in I]$ be the given schema.

- (a) For the schema of its star product holds:

$$\otimes X = \left\{ \frac{[\xi_j | j \in J]}{J \subseteq I, \xi_j \in X_j \text{ for all } j \in J} \right\}$$

$$\@(\otimes X) = \bigcup \{J \mid J \subseteq I\} = I$$

$$\begin{aligned} \mathbf{dom}_i(\otimes X) &= \{\xi_i \mid \xi \in \otimes X, i \in \mathbf{dom}(\xi)\} \\ &= X_i \quad \text{for each } i \in I \end{aligned}$$

$$\mathbf{x}(\otimes X) = [X_i | i \in I] = X$$

(b) For the schema of the coproduct holds:

$$\oplus X = \left\{ \frac{[i \mapsto \xi]}{i \in I, \xi \in X_i} \right\}$$

$$\@(\oplus X) = \bigcup \{\{i\} \mid i \in I\} = I$$

$$\begin{aligned} \mathbf{dom}_i(\oplus X) &= \{\xi \mid [i \mapsto \xi] \in \oplus X\} \\ &= X_i \quad \text{for each } i \in I \end{aligned}$$

$$\mathbf{x}(\oplus X) = [X_i | i \in I] = X$$

(c) And if X is proper, then

$$\begin{aligned} \otimes X &= \{[\xi_i | i \in I] \mid \xi_i \in X_i \text{ for all } i \in I\} \\ &\neq \emptyset \quad (\text{see 12.1.12}) \end{aligned}$$

$$\begin{aligned} \@(\otimes X) &= \bigcup \{\mathbf{dom}(\xi) \mid \xi \in \otimes X\} \\ &= I \quad (\text{because } \otimes X \neq \emptyset) \end{aligned}$$

$$\begin{aligned} \mathbf{dom}_i(\otimes X) &= \{\xi_i \mid \xi \in \otimes X\} \\ &= X_i \quad \text{for each } i \in I \end{aligned}$$

$$\mathbf{x}(\otimes X) = [X_i | i \in I] = X$$

(4) Let $X = [X_i | i \in I]$ be the given schema and $\mathbf{x}(\Xi) = [Y_j | j \in J] = Y$ the schema of the given record class Ξ . We need to show that

$$\Xi \subseteq \otimes X \text{ iff } Y \subseteq \mathbf{pr}(X, I \cap J)$$

We proof this equivalence in both directions:

- (a) Suppose, $\Xi \subseteq \otimes X$. That means, that for every $\xi \in \Xi$, $\mathbf{dom}(\xi) \subseteq J$ and $\xi(j) \in X_j$ for all $j \in \mathbf{dom}(\xi)$. $J := \bigcup \{\mathbf{dom}(\xi) \mid \xi \in \Xi\}$ implies $J \subseteq I$ and $Y_j := \{\xi(j) \mid \xi \in \Xi, j \in \mathbf{dom}(\xi)\}$ implies $Y_j \subseteq X_j$, for all $j \in J$. So $Y \subseteq \mathbf{pr}(X, I \cap J)$.
- (b) On the other hand, suppose $Y \subseteq \mathbf{pr}(X, I \cap J)$. Then $J = I \cap J$, i.e. $J \subseteq I$. $\xi \in \Xi$ implies $\mathbf{dom}(\xi) \subseteq J \subseteq I$ and $j \in J$ implies $\xi(j) \in Y_j \subseteq X_j$. Thus $\Xi \subseteq \otimes X$.

15.2 Graphs

15.2.1 Definition

A record class Ξ is also called a graph. More specifically, we call Ξ a

- (1) star graph, Kleene graph, general graph or quasi-relation graph, if $\Xi \subseteq \otimes X$ for some schema X
- (2) cartesian graph, expanded graph or relation graph, if $\Xi \subseteq \otimes \mathbf{x}(\Xi) \subseteq \otimes X$ for some schema X
- (3) cograph or singular graph, if $\Xi \subseteq \oplus X$ for some schema X

15.2.2 Example

Let us take the record class Ξ from example 15.1.2 again

$$\Xi = \left\{ \langle \rangle, \left[\begin{array}{l} a \mapsto 5 \\ b \mapsto 3 \end{array} \right], [a \mapsto 4], \left[\begin{array}{l} a \mapsto 5 \\ b \mapsto 7 \\ d \mapsto 1 \end{array} \right], \left[\begin{array}{l} a \mapsto 2 \\ d \mapsto 2 \end{array} \right] \right\}$$

(1) Ξ certainly is a star graph, because with 15.1.4(2) we can derive

$$\Xi \subseteq \otimes \mathbf{x}(\Xi) = \otimes \left[\begin{array}{l} a \mapsto \{2, 4, 5\} \\ b \mapsto \{3, 7\} \\ d \mapsto \{1, 2\} \end{array} \right]$$

(2) Ξ is certainly not a cartesian graph. The members of Ξ have different domains: $\emptyset, \{a, b\}, \{a\}, \{a, b, d\}, \{a, d\}$. But for every cartesian product, they would have to be identical.

(3) Ξ is coproduct neither. Therefore, every of its members would have to be a singular record, i.e. with exactly one index. But one $[a \mapsto 4]$ is actually singular, the others are not.

15.2.3 Lemma

Every record class Ξ is

- (1) a star graph
- (2) a cartesian graph iff $\Xi \subseteq \otimes \mathbf{x}(\Xi)$
- (3) a cartesian graph iff $\mathbf{dom}(\xi) = \mathbf{dom}(v)$ for all $\xi, v \in \Xi$
- (4) a cograph iff $\Xi \subseteq \oplus \mathbf{x}(\Xi)$
- (5) a cograph iff $\mathbf{card}(\mathbf{dom}(\xi)) = 1$ for all $\xi \in \Xi$

15.2.4 Proof of 15.2.3

(1) Each record class Ξ has a schema X such that $\Xi \subseteq \otimes X$, namely $X := \mathbf{x}(\Xi)$. $\Xi \subseteq \otimes \mathbf{x}(\Xi)$ holds, according to 15.1.4(2).

(2) In case $\Xi = \emptyset$, Ξ is a cartesian graph and $\Xi \subseteq \{\langle \rangle\} = \otimes(\langle \rangle) = \otimes \mathbf{x}(\Xi)$.
Otherwise, $\Xi \neq \emptyset$, and then

Ξ is a cartesian graph

iff $\Xi \subseteq \otimes X$ for some schema X

iff $\Xi \subseteq \{[x_i | i \in I] \mid x_i \in X_i \text{ for all } i \in I\}$

for some schema $X = [X_i | i \in I]$

iff $\Xi \subseteq \{[x_i | i \in \@(\Xi)] \mid x_i \in X_i \text{ for all } i \in \@(\Xi)\}$

for some schema $X = [X_i | i \in \@(\Xi)]$

iff $\Xi \subseteq \otimes \mathbf{x}(\Xi) \subseteq \otimes X$

for some schema $X = [X_i | i \in \@(\Xi)]$

iff $\Xi \subseteq \otimes \mathbf{x}(\Xi)$

So in every case, Ξ is a cartesian graph iff $\Xi \subseteq \otimes \mathbf{x}(\Xi)$.

(3) If Ξ is a cartesian graph, then $\mathbf{dom}(\xi) = \mathbf{dom}(v) = \@(\Xi)$, for all $\xi, v \in \Xi$ (regardless, if Ξ is empty or not). On the other hand, if $\mathbf{dom}(\xi) = \mathbf{dom}(v)$ for all $\xi, v \in \Xi$, then $\xi \in \otimes \mathbf{x}(\Xi)$ for all $\xi \in \Xi$, and Ξ is a cartesian graph according to (2).

(4) For every record class Ξ holds

Ξ is a cograph
 iff $\Xi \subseteq \oplus X$ for some schema X
 iff $\Xi \subseteq \{[i \mapsto x_i] \mid i \in I, x_i \in X_i\}$
 for some schema $X = [X_i \mid i \in I]$
 iff $\Xi \subseteq \{[i \mapsto x_i] \mid i \in @(\Xi), x_i \in \mathbf{dom}_i(\Xi)\}$
 iff $\Xi \subseteq \oplus \mathbf{x}(\Xi)$

(5) If Ξ is a cograph, then every member has the form $[i \mapsto x_i]$, i.e. $\mathbf{card}(\mathbf{dom}(\xi)) = 1$. On the other hand, if $\mathbf{card}(\mathbf{dom}(\xi)) = 1$ is the case, then $\xi \in \oplus \mathbf{x}(\Xi)$, i.e. $\Xi \subseteq \oplus \mathbf{x}(\Xi)$, so that Ξ is a cograph according to (4).

16 Partitions and distinct products

16.0.5 Repetition

- (1) Recall (10.3.2 and 10.3.4), that we call two records $\xi = [\xi_i | i \in I]$ and $v = [v_j | j \in J]$ **distinct**, written $\xi \dot{\neq} v$, iff $I \cap J = \emptyset$. In case they are distinct, their **distinct join**

$$\xi \dot{\vee} v := \left[k \mapsto \begin{cases} \xi_k & \text{if } k \in I \\ v_k & \text{if } k \in J \end{cases} \mid k \in I \cup J \right]$$

is well-defined (according to 10.4.1). And more general, a class Ξ of records is **(pairwise) distinct**, if $\xi \dot{\neq} v$ for all $\xi, v \in \Xi$ with $\xi \neq v$. And in that case, its **distinct join**

$$\dot{\vee} \Xi \text{ is well-defined}$$

- (2) Recall 9.1.1, that a “**record–record**” is a record

$$\rho = [\rho_k | k \in K] = [[\rho_{k,l} | l \in L_k] | k \in K]$$

where every $\rho_k = [\rho_{k,l} | l \in L_k]$ is a record itself.

So a special record–record is a **schema record**

$$\sigma = [\sigma_k | k \in K] = [[\sigma_{k,l} | l \in L_k] | k \in K]$$

where each $\sigma_k = [\sigma_{k,l} | l \in L_k]$ is a schema.

- (3) Recall 15.2.1, that a record class Ξ is also called a **graph**.

16.1 Class partition (record)

16.1.1 Definition

A **class partition (record)** is a schema $P = [P_k | k \in K]$, such that

$$P_i \cap P_j = \emptyset \text{ for all } i, j \in K \text{ with } i \neq j$$

P is also called a **class partition of C** , where C is the class defined by

$$C := \bigcup \{P_k \mid k \in K\}$$

Furthermore, P is called a **proper class partition**, if P is a proper schema.

16.1.2 Remark and example class partitions

Let $C = \{1, 2, 3, 4, 5, 6\}$. Usually in mathematics, a *partition* of C is a distribution of C 's members into mutually disjoint subclasses, e.g. $\{\{1, 3\}, \{2, 4, 5\}, \{6\}\}$. We use the same idea here, except that each subclass in a partition is indexed, i.e. a *class partition (record)* of C is a schema, say

$$P = \left[\begin{array}{l} a \mapsto \{1, 3\} \\ b \mapsto \{2, 4, 5\} \\ c \mapsto \{6\} \end{array} \right]$$

or with an alternative index class

$$P' = \left[\begin{array}{l} 1 \mapsto \{1, 3\} \\ 2 \mapsto \{2, 4, 5\} \\ 3 \mapsto \{6\} \end{array} \right] = \langle \{1, 3\}, \{2, 4, 5\}, \{6\} \rangle$$

P and P' are (class) partitions of C . And they are *proper* be-

cause they are proper schemas. Two examples of non–proper partitions of C are

$$P'' = \left[\begin{array}{l} a \mapsto \{1, 3\} \\ b \mapsto \{2, 4, 5\} \\ c \mapsto \{6\} \\ d \mapsto \emptyset \end{array} \right]$$

and

$$P''' = \langle \{1, 3\}, \emptyset, \{2, 4, 5\}, \emptyset, \{6\} \rangle$$

16.1.3 Lemma inclusion criterion

For every schema $P = [P_k | k \in K]$ the following statements are equivalent:

- (1) P is a class partition.
- (2) There is a class partition $P' = [P'_k | k \in K]$ with $P \subseteq P'$.

16.1.4 Proof of 16.1.3

(1) implies (2): Suppose, P is a class partition. There is $P \subseteq P$.

(2) implies (1): Suppose $P' = [P'_k | k \in K]$ is a class partition with $P \subseteq P'$. P' being a class partition means $P'_i \cap P'_j = \emptyset$ for all $i, j \in K$ with $i \neq j$. $P \subseteq P'$ means $P_k \subseteq P'_k$ for all $k \in K$. So $P_i \cap P_j = \emptyset$ for all $i, j \in K$ with $i \neq j$, P is a class partition as well.

16.2 Record and schema partition

16.2.1 Definition

If $\rho = [\rho_k | k \in K]$ is a record–record, then

$$\mathbf{doms}(\rho) := [\mathbf{dom}(\rho_k) | k \in K]$$

is the **domain schema** or **index schema** of ρ .

16.2.2 Example

- (1) A record–record ρ and its domain schema are given by

$$\rho = \left[\begin{array}{l} a \mapsto \left[\begin{array}{l} d \mapsto w \\ e \mapsto x \end{array} \right] \\ b \mapsto \langle \rangle \\ c \mapsto \left[\begin{array}{l} f \mapsto y \\ g \mapsto z \end{array} \right] \end{array} \right] \quad \mathbf{doms}(\rho) = \left[\begin{array}{l} a \mapsto \{d, e\} \\ b \mapsto \emptyset \\ c \mapsto \{f, g\} \end{array} \right]$$

- (2) A tuple of tuples is a special case of a record–record:

$$\rho' = \langle \langle 1, 2, 3 \rangle, \langle 2, 3, 4 \rangle, \langle \rangle, \langle \rangle, \langle 3, 4, 5 \rangle \rangle$$

$$\mathbf{doms}(\rho') = \langle \{1, 2, 3\}, \{1, 2, 3\}, \emptyset, \emptyset, \{1, 2, 3\} \rangle$$

16.2.3 Definition

A record partition is a record–record $\pi = [\pi_k | k \in K]$, such that

$$\pi_i \not\bowtie \pi_j \text{ for all } i, j \in K \text{ with } i \neq j$$

In that case, it is also called a (record) partition of ξ , where ξ is the record, well–defined by

$$\xi := \bigvee \{ \pi_k \mid k \in K \}$$

Furthermore, π is called a proper record partition, if

$$\pi_k \neq \langle \rangle \text{ for every } k \in K$$

A (record) partition $\pi = [\pi_k | k \in K]$ of a schema X is also called a schema partition of X .

$$\mathbf{dom}(\xi) = \{a, b, c, d, e\}$$

No value of π is $\langle \rangle$, P is not a proper schema, so π is a proper record partition of ξ .

Examples of non–proper record partitions of ξ are given by

$$\pi' = \left[\begin{array}{l} p \mapsto \left[\begin{array}{l} a \mapsto \xi_a \\ c \mapsto \xi_c \end{array} \right] \\ q \mapsto \left[\begin{array}{l} b \mapsto \xi_b \\ d \mapsto \xi_d \end{array} \right] \\ r \mapsto \left[\begin{array}{l} d \mapsto \xi_d \\ e \mapsto \xi_e \end{array} \right] \\ s \mapsto \langle \rangle \end{array} \right]$$

and

$$\pi'' = \left\langle \left[\begin{array}{l} a \mapsto \xi_a \\ c \mapsto \xi_c \end{array} \right], \langle \rangle, \left[b \mapsto \xi_b \right], \langle \rangle, \left[\begin{array}{l} d \mapsto \xi_d \\ e \mapsto \xi_e \end{array} \right] \right\rangle$$

Their domain schemas

$$P' := [\mathbf{dom}(\pi'_k) | k \in \mathbf{dom}(\pi')] = \left[\begin{array}{l} p \mapsto \{a, c\} \\ q \mapsto \{b\} \\ r \mapsto \{d, e\} \\ s \mapsto \emptyset \end{array} \right]$$

$$P'' := [\mathbf{dom}(\pi''_k) | k \in \mathbf{dom}(\pi'')] = \left[\begin{array}{l} 1 \mapsto \{a, c\} \\ 2 \mapsto \emptyset \\ 3 \mapsto \{b\} \\ 4 \mapsto \emptyset \\ 5 \mapsto \{d, e\} \end{array} \right] = \langle \{a, c\}, \emptyset, \{b\}, \emptyset, \{d, e\} \rangle$$

are class partitions of $\mathbf{dom}(\xi)$, but not proper.

16.2.4 Lemma

For every record–record $\rho = [\rho_k | k \in K]$ holds

- (1) ρ is a record partition iff $\mathbf{doms}(\rho)$ is a class partition
- (2) ρ is a proper record partition iff $\mathbf{doms}(\rho)$ is a proper class partition

16.2.5 Proof of 16.2.4

Let $\rho = [\rho_k | k \in K] = [[\rho_{k,l} | l \in L_k] | k \in K]$ be the given record–record. Then $\mathbf{doms}(\rho) = [L_k | k \in K]$ and we obtain:

- (1) ρ is a record partition iff $(\rho_i \not\bowtie \rho_j \text{ for all } i, j \in K \text{ with } i \neq j)$ iff $(L_i \cap L_j = \emptyset \text{ for all } i, j \in K \text{ with } i \neq j)$ iff $\mathbf{doms}(\rho)$ is a class partition.
- (2) ρ is proper iff $\rho_k \neq \langle \rangle$ for all $k \in K$ iff $L_k \neq \emptyset$ for all $k \in K$ iff $\mathbf{doms}(\rho)$ is proper.

16.2.6 Remark and example record partitions

Let ξ be a record, given by

$$\xi = \left[\begin{array}{l} a \mapsto \xi_a \\ b \mapsto \xi_b \\ c \mapsto \xi_c \\ d \mapsto \xi_d \\ e \mapsto \xi_e \end{array} \right]$$

We cut up ξ into pairwise distinct records, say

$$\left[\begin{array}{l} a \mapsto \xi_a \\ c \mapsto \xi_c \end{array} \right] \quad \left[b \mapsto \xi_b \right] \quad \left[\begin{array}{l} d \mapsto \xi_d \\ e \mapsto \xi_e \end{array} \right]$$

We index these records, i.e. we define a record–record such as

$$\pi = \left[\begin{array}{l} p \mapsto \left[\begin{array}{l} a \mapsto \xi_a \\ c \mapsto \xi_c \end{array} \right] \\ q \mapsto \left[b \mapsto \xi_b \right] \\ r \mapsto \left[\begin{array}{l} d \mapsto \xi_d \\ e \mapsto \xi_e \end{array} \right] \end{array} \right]$$

which is then a record partition of ξ .

Lemma 16.2.4 is easily confirmed:

$$P := \mathbf{doms}(\pi) = \left[\begin{array}{l} p \mapsto \{a, c\} \\ q \mapsto \{b\} \\ r \mapsto \{d, e\} \end{array} \right]$$

is a class partition, namely a class partition of

16.2.7 Remark

So if $\xi = [\xi_i | i \in I]$ is a record and $\rho = [[\rho_{k,l} | l \in L_k] | k \in K]$ is a record–record, the following statements are obviously equivalent:

- (1) ρ is a record partition of ξ
- (2) $\mathbf{doms}(\rho) = [L_k | k \in K]$ is a class partition (record) of I and $\rho_{k,l} = \xi_l$ for all $k \in K$ and $l \in L_k$.

16.2.8 Lemma inclusion criterion

For every schema record $\sigma = [\sigma_k | k \in K]$ the following statements are equivalent:

- (1) σ is a schema partition
- (2) There is a schema partition $\sigma' = [\sigma'_k | k \in K]$ with $\sigma \subseteq \sigma'$.

16.2.9 Proof of 16.2.8

(1) implies (2): If σ is a schema partition, then $\sigma' = [\sigma'_k | k \in K] := \sigma$ obviously satisfies $\sigma_k \subseteq \sigma'_k$ for all $k \in K$.

(2) implies (1): Suppose $\sigma' = [\sigma'_k | k \in K]$ is a schema partition with $\sigma_k \subseteq \sigma'_k$ for all $k \in K$. σ' being a schema partition means that $\mathbf{dom}(\sigma'_i) \cap \mathbf{dom}(\sigma'_j) = \emptyset$ for all $i, j \in K$ with $i \neq j$. $\sigma_k \subseteq \sigma'_k$ presupposes $\mathbf{dom}(\sigma_k) = \mathbf{dom}(\sigma'_k)$ (definition 14.1.1 of the schema inclusion \subseteq). So $\mathbf{dom}(\sigma_i) \cap \mathbf{dom}(\sigma_j) = \emptyset$ for all $i, j \in K$ with $i \neq j$. σ is a schema partition.

16.2.10 Lemma

- (1) If $\xi = [\xi_i | i \in I]$ is a record and $[P_k | k \in K]$ a class partition of I , then

$$\xi = \dot{\bigvee}_{k \in K} \mathbf{pr}(\xi, P_k)$$

- (2) If $\xi = [\xi_i | i \in I]$ is a record, $J \subseteq I$, and $[P_k | k \in K]$ is a class partition of J , then

$$\mathbf{pr}(\xi, J) = \dot{\bigvee}_{k \in K} \mathbf{pr}(\xi, P_k)$$

- (3) Let $\pi = [\pi_k | k \in K]$ be a record partition and $P = [P_k | k \in K]$ a schema with $P_k \subseteq \mathbf{dom}(\pi_k)$ for each $k \in K$. Then

$$\mathbf{pr}\left(\dot{\bigvee}_{k \in K} \pi_k, \bigcup_{k \in K} P_k\right) = \dot{\bigvee}_{k \in K} \mathbf{pr}(\pi_k, P_k)$$

16.2.11 Proof of 16.2.10

- (1) $\xi = [\xi_i | i \in \bigcup_{k \in K} P_k] = \dot{\bigvee}_{k \in K} [\xi_i | i \in P_k] = \dot{\bigvee}_{k \in K} \mathbf{pr}(\xi, P_k)$

- (2) We have

$$\begin{aligned} \mathbf{pr}(\xi, J) &= \mathbf{pr}\left(\xi, \bigcup_{k \in K} P_k\right) \\ &= [\xi_i | i \in \bigcup_{k \in K} P_k] \\ &= \dot{\bigvee}_{k \in K} [\xi_i | i \in P_k] \end{aligned} \quad \text{due to (1)}$$

- (3) π is a record partition, so the $\mathbf{dom}(\pi_k)$ are pairwise disjoint. With $P_k \subseteq \mathbf{dom}(\pi_k)$, the P_k are pairwise disjoint as well and $[P_k | k \in K]$ is a class partition. We can apply (2) and obtain

$$\begin{aligned} \mathbf{pr}\left(\dot{\bigvee}_{k \in K} \pi_k, \bigcup_{k \in K} P_k\right) &= \dot{\bigvee}_{k \in K} \mathbf{pr}\left(\dot{\bigvee}_{k \in K} \pi_k, P_k\right) \\ &= \dot{\bigvee}_{k \in K} \mathbf{pr}(\pi_k, P_k) \end{aligned}$$

16.3 Relative distinctness

16.3.1 Definition

Two graphs (i.e. record classes) Γ and Δ are called relatively distinct, written $\Gamma \check{\Delta}$, if $\xi \check{\Delta} \nu$ for all $\xi \in \Gamma$ and $\nu \in \Delta$.

A graph record $[\Gamma_k | k \in K]$ is called

- relatively distinct, if $\Gamma_i \check{\Delta} \Gamma_j$ for all $i, j \in K$ with $i \neq j$.
- proper, if $\Gamma_k \neq \emptyset$ for all $k \in K$

Graphs $\Gamma_1, \dots, \Gamma_n$ are called relatively distinct, written $\Gamma_1 \check{\Delta} \dots \check{\Delta} \Gamma_n$, if $\langle \Gamma_1, \dots, \Gamma_n \rangle$ is relatively distinct.

16.3.2 Example

Let

$$\Gamma = \left\langle \langle \rangle, [a \mapsto 1], [a \mapsto 2] \right\rangle$$

$$\Delta = \emptyset$$

$$\Sigma = \left\langle \langle \rangle, [c \mapsto 4], [d \mapsto 3] \right\rangle$$

then $\Gamma \check{\Delta}$, $\Gamma \check{\Delta} \Sigma$, and $\Delta \check{\Delta} \Sigma$. In other words, $\Gamma \check{\Delta} \Delta \check{\Delta} \Sigma$.

Recall, that this is the *relative distinctness*. For the (pairwise) *distinctness* (see definition 10.3.4) of each of the three graphs holds: Δ and Σ are (pairwise) distinct each. But Γ is not, because

$$[a \mapsto 1] \not\check{\Delta} \left[\begin{array}{l} a \mapsto 2 \\ b \mapsto 3 \end{array} \right]$$

However, $\Gamma \check{\Delta} \Delta \check{\Delta} \Sigma$ is saying that a graph record with these three graphs as values, say $\langle \Gamma, \Delta, \Sigma \rangle$, is a relatively distinct graph record. If we generate

$$\langle @(\Gamma), @(\Delta), @(\Sigma) \rangle = \langle \{a, b\}, \emptyset, \{c, d\} \rangle$$

we see that this is a class partition. In fact, it is a class partition exactly because the graph record $\langle \Gamma, \Delta, \Sigma \rangle$ is relatively distinct. This gives us an alternative criterion of relative distinctness. Yet another equivalent criterion comes with the schema record

$$\begin{aligned} &\langle \mathbf{x}(\Gamma), \mathbf{x}(\Delta), \mathbf{x}(\Sigma) \rangle \\ &= \left\langle \left[\begin{array}{l} a \mapsto \{1, 2\} \\ b \mapsto \{3\} \end{array} \right], \langle \rangle, \left[\begin{array}{l} c \mapsto \{4\} \\ d \mapsto \{3\} \end{array} \right] \right\rangle \end{aligned}$$

This schema record is a schema partition.

16.3.3 Lemma

If $\Gamma = [\Gamma_k | k \in K]$ is a relatively distinct graph record, then every member of $\mathbf{Incl}(\Gamma)$, i.e. every graph record $\Gamma' = [\Gamma'_k | k \in K]$ with $\Gamma' \subseteq \Gamma$, is relatively distinct as well.

16.3.4 Proof of 16.3.3

This is quite obvious. If $\Gamma' \subseteq \Gamma$, then $\Gamma'_k \subseteq \Gamma_k$ for all $k \in K$. So the fact that $\Gamma_i \check{\Delta} \Gamma_j$ holds for all $i, j \in K$, $i \neq j$, immediately implies $\Gamma'_i \check{\Delta} \Gamma'_j$.

16.3.5 Lemma

Let $\Gamma = [\Gamma_k | k \in K]$ be a graph record. Then the following statements are equivalent:

- (1) Γ is relatively distinct.
- (2) $@(\Gamma_k) | k \in K$ is a class partition.
- (3) There is a class partition $[P_k | k \in K]$ such that $@(\Gamma_k) | k \in K \subseteq [P_k | k \in K]$.
- (4) $\mathbf{x}(\Gamma_k) | k \in K$ is a schema partition.
- (5) There is a schema partition $[\sigma_k | k \in K]$ with $\mathbf{x}(\Gamma_k) \subseteq \sigma_k$ for every $k \in K$.
- (6) There is a schema partition $[\sigma_k | k \in K]$ with $\Gamma \subseteq @(\sigma_k | k \in K)$.

16.3.6 Proof of 16.3.5

Let $\Gamma = [\Gamma_k | k \in K]$ be the given graph record.

(1) \Leftrightarrow (2) We have

Γ is relatively distinct

iff $\xi \dot{\bowtie} v$ for all $i, j \in K, i \neq j$ and all $\xi \in \Gamma_i$ and $v \in \Gamma_j$

iff $\mathbf{dom}(\xi) \cap \mathbf{dom}(v) = \emptyset$

for all $i, j \in K, i \neq j$ and all $\xi \in \Gamma_i$ and $v \in \Gamma_j$

iff $\bigcup \{ \mathbf{dom}(\xi) \mid \xi \in \Gamma_i \} \cap \bigcup \{ \mathbf{dom}(v) \mid v \in \Gamma_u \} = \emptyset$

for all $i, j \in K, i \neq j$

iff $\mathbb{@}(\Gamma_i) \cap \mathbb{@}(\Gamma_j) = \emptyset$ for all $i, j \in K, i \neq j$

iff $[\mathbb{@}(\Gamma_k) \mid k \in K]$ is a class partition

(2) \Leftrightarrow (3) see lemma 16.1.3.

(2) \Leftrightarrow (4) $[\mathbb{@}(\Gamma_k) \mid k \in K] = \mathbf{doms}([\mathbf{x}(\Gamma_k) \mid k \in K])$ so according to 16.2.4, $[\mathbf{x}(\Gamma_k) \mid k \in K]$ is a record/schema partition iff $[\mathbb{@}(\Gamma_k) \mid k \in K]$ is a class partition.

(4) \Leftrightarrow (5) see lemma 16.2.8.

(1) \Leftrightarrow (6) If Γ is relatively distinct, then $[\mathbf{x}(\Gamma_k) \mid k \in K]$ is a schema partition according to (4) and $\Gamma_k \subseteq \mathbb{@}\mathbf{x}(\Gamma_k)$ according to 15.1.4(2) for each $k \in K$.

On the other hand, if there is a schema partition $[\sigma_k \mid k \in K]$ with $\Gamma \subseteq [\mathbb{@}\sigma_k \mid k \in K]$, then all $k, l \in K$ with $k \neq l$ satisfy $\mathbf{dom}(\sigma_k) \cap \mathbf{dom}(\sigma_l) = \emptyset$ and thus $\mathbb{@}(\Gamma_k) \cap \mathbb{@}(\Gamma_l) = \emptyset$. In other words, $\Gamma_k \dot{\bowtie} \Gamma_l$ and Γ is relatively distinct.

16.3.7 Lemma

If $[\sigma_k \mid k \in K]$ is a schema partition, then each of the following three graph records

$[\mathbb{@}\sigma_k \mid k \in K]$ $[\mathbb{@}\sigma_k \mid k \in K]$ $[\mathbb{+}\sigma_k \mid k \in K]$
is relatively distinct.

16.3.8 Proof of 16.3.7

Let $i, j \in K$ with $i \neq j$ and let $\xi_i \in \mathbb{@}\sigma_i$ and $\xi_j \in \mathbb{@}\sigma_j$. Then $\mathbf{dom}(\xi_i) \subseteq \mathbf{dom}(\sigma_i)$ and $\mathbf{dom}(\xi_j) \subseteq \mathbf{dom}(\sigma_j)$. $[\sigma_k \mid k \in K]$ is a schema partition, so $\mathbf{dom}(\sigma_i) \cap \mathbf{dom}(\sigma_j) = \emptyset$. Therefore $\mathbf{dom}(\xi_i) \cap \mathbf{dom}(\xi_j) = \emptyset$, i.e. $\mathbb{@}\sigma_i \dot{\bowtie} \mathbb{@}\sigma_j$, i.e. $[\mathbb{@}\sigma_k \mid k \in K]$ is relatively distinct.

For every $k \in K$, $\mathbb{@}\sigma_k \subseteq \mathbb{+}\sigma_k$ and $\mathbb{+}\sigma_k \subseteq \mathbb{+}\sigma_k$, so $[\mathbb{+}\sigma_k \mid k \in K] \subseteq [\mathbb{@}\sigma_k \mid k \in K]$ and $[\mathbb{+}\sigma_k \mid k \in K] \subseteq [\mathbb{+}\sigma_k \mid k \in K]$. With 16.3.3 we infer that both graph records are relatively distinct, too.

16.4 Distinct products

16.4.1 Definition

If $\Gamma = [\Gamma_k \mid k \in K]$ is a relatively distinct graph record, then

$$\odot\Gamma := \bigcirc_{k \in K} \Gamma := \{ \dot{\bigvee} \rho \mid \rho \in \odot\Gamma \}$$

is the distinct product of Γ

And for graphs $\Gamma_1, \dots, \Gamma_n$ with $\Gamma_1 \dot{\bowtie} \dots \dot{\bowtie} \Gamma_n$ we define

$$\Gamma_1 \odot \dots \odot \Gamma_n := \bigcirc_{i=1}^n \Gamma_i := \odot(\Gamma_1, \dots, \Gamma_n)$$

the distinct product of $\Gamma_1, \dots, \Gamma_n$

16.4.2 Remark

(1) Note, that the $\odot\Gamma$ is well-defined, because the relative distinctness of Γ guarantees that $\dot{\bigvee} \rho$ exists for each $\rho = [\rho_k \mid k \in K] \in \odot\Gamma$.

(2) For graphs $\Gamma_1, \dots, \Gamma_n$ with $\Gamma_1 \dot{\bowtie} \dots \dot{\bowtie} \Gamma_n$ we obtain

$$\Gamma_1 \odot \dots \odot \Gamma_n = \begin{cases} \{\langle \rangle\} & \text{if } n = 0 \\ \Gamma_1 & \text{if } n = 1 \\ \left\{ \frac{\rho_1 \dot{\bigvee} \dots \dot{\bigvee} \rho_n}{\rho_1 \in \Gamma_1, \dots, \rho_n \in \Gamma_n} \right\} & \text{else} \end{cases}$$

(3) An immediate consequence of its definition are the associativity and commutativity (see lemma 16.4.3 below). We may place parentheses arbitrarily:

$$\Gamma_1 \odot \Gamma_2 \odot \Gamma_3 = (\Gamma_1 \odot \Gamma_2) \odot \Gamma_3 = \Gamma_1 \odot (\Gamma_2 \odot \Gamma_3)$$

where each of the three terms is well-defined iff the other two are.

16.4.3 Lemma

If Γ, Δ, Σ are pairwise distinct record classes, i.e. $\Gamma \dot{\bowtie} \Delta \dot{\bowtie} \Sigma$, then

- (1) $(\Gamma \odot \Delta) \odot \Sigma = \Gamma \odot (\Delta \odot \Sigma)$ (associativity of \odot)
- (2) $\Gamma \odot \Delta = \Delta \odot \Gamma$ (commutativity of \odot)
- (3) $\Gamma \odot \emptyset = \emptyset$ (\odot with empty class)
- (4) $\Gamma \odot \{\langle \rangle\} = \Gamma$ (\odot with empty product)

16.4.4 Proof of 16.4.3

- (1) For all $\xi \in \Gamma, v \in \Delta, \zeta \in \Sigma$ holds $\xi \dot{\bigvee} v \dot{\bigvee} \zeta = (\xi \dot{\bigvee} v) \dot{\bigvee} \zeta = \xi \dot{\bigvee} (v \dot{\bigvee} \zeta)$, so $\Gamma \odot \Delta \odot \Sigma = (\Gamma \odot \Delta) \odot \Sigma = \Gamma \odot (\Delta \odot \Sigma)$.
- (2) $\Gamma \odot \Delta = \{ \xi \dot{\bigvee} v \mid \xi \in \Gamma, v \in \Delta \} = \{ v \dot{\bigvee} \xi \mid v \in \Delta, \xi \in \Gamma \} = \Delta \odot \Gamma$.
- (3) $\Gamma \odot \emptyset = \{ \xi \dot{\bigvee} v \mid \xi \in \Gamma, v \in \emptyset \} = \emptyset$
- (4) $\Gamma \odot \Delta = \{ \xi \dot{\bigvee} v \mid \xi \in \Gamma, v \in \{\langle \rangle\} \} = \{ \xi \dot{\bigvee} \langle \rangle \mid \xi \in \Gamma \} = \Gamma$.

16.4.5 Example

Take the graphs Γ, Δ, Σ from 16.3.2 again. Due to their relative distinctness, the following two examples of distinct products are well defined:

$$\Gamma \odot \Sigma = \left\{ \langle \rangle, \left[a \mapsto 1 \right], \left[\begin{array}{l} a \mapsto 2 \\ b \mapsto 3 \end{array} \right], \left[\begin{array}{l} c \mapsto 4 \\ d \mapsto 3 \end{array} \right], \left[\begin{array}{l} a \mapsto 1 \\ c \mapsto 4 \\ d \mapsto 3 \end{array} \right], \left[\begin{array}{l} a \mapsto 2 \\ b \mapsto 3 \\ c \mapsto 4 \\ d \mapsto 3 \end{array} \right] \right\}$$

$$\Gamma \odot \Delta \odot \Sigma = \{ \gamma \dot{\bigvee} \delta \dot{\bigvee} \sigma \mid \gamma \in \Gamma, \delta \in \emptyset, \sigma \in \Sigma \} = \emptyset$$

16.4.6 Remark

Consider the cardinalities of the graphs and their products in the previous example 16.4.5:

$$\mathbf{card}(\Gamma \odot \Sigma) = 6 = 3 \cdot 2 = \mathbf{card}(\Gamma) \cdot \mathbf{card}(\Sigma)$$

$$\mathbf{card}(\Gamma \odot \Delta \odot \Sigma) = 0 = 3 \cdot 0 \cdot 2 = \mathbf{card}(\Gamma) \cdot \mathbf{card}(\Delta) \cdot \mathbf{card}(\Sigma)$$

This behavior is typical for distinct products (hence one motivation for the term “product”) as we will point out in lemma 16.4.10 below. Obviously, the distinct product $\Gamma \odot \Delta \odot \Sigma$ has

the same cardinality than the cartesian product $\Gamma \times \Delta \times \Sigma$. There is even a bijection between the cartesian and the distinct product and this correspondence allows us to uniquely reconstruct the arguments from the result.

16.4.7 Lemma

Let $\Gamma = [\Gamma_k | k \in K]$ be relatively distinct graph record, then

$$\pi_\Gamma := \begin{bmatrix} \otimes \Gamma \longrightarrow \odot \Gamma \\ \rho \mapsto \dot{\vee} \rho \end{bmatrix}$$

π_Γ is a bijection with the inverse

$$\pi_\Gamma^{-1} = \begin{bmatrix} \odot \Gamma \longrightarrow \otimes \Gamma \\ \xi \mapsto [\mathbf{pr}(\xi, \mathbf{dom}(\xi) \cap \otimes(\Gamma_k)) | k \in K] \end{bmatrix}$$

Furthermore, if $P = [P_k | k \in K]$ a class partition with $[\otimes(\Gamma_k) | k \in K] \subseteq P$, then

$$\pi_\Gamma^{-1} = \begin{bmatrix} \odot \Gamma \longrightarrow \otimes \Gamma \\ \xi \mapsto [\mathbf{pr}(\xi, \mathbf{dom}(\xi) \cap P_k) | k \in K] \end{bmatrix}$$

16.4.8 Proof of 16.4.7

First of all, let us mention that $\odot \Gamma$ and the function π_Γ are well defined because Γ is relatively distinct.

π_Γ is ought to be a bijection, i.e. it must surjective and injective. The surjectivity is a trivial truth, $\odot \Gamma$ is in 16.4.1 just defined to be the image class $\{\pi_\gamma(\rho) | \rho \in \otimes \Gamma\}$ of π_Γ . It remains to show that π_Γ is injective as well.

Let $\rho = [\rho_k | k \in K]$ and $\rho' = [\rho'_k | k \in K]$ be any two members of $\otimes \Gamma$, i.e. $\rho_k \in \Gamma_k$ and $\rho'_k \in \Gamma_k$ for all $k \in K$. We put $\xi := \pi_\Gamma(\rho) = \dot{\vee} \{\rho_k | k \in K\}$ and $\xi' := \pi_\Gamma(\rho')$. Suppose, $\rho \neq \rho'$. This means, there has to be a $k \in K$ with $\rho_k \neq \rho'_k$. These ρ_k and ρ'_k are records and not being equal means either (a) $\mathbf{dom}(\rho_k) \neq \mathbf{dom}(\rho'_k)$ or (b) $\mathbf{dom}(\rho_k) = \mathbf{dom}(\rho'_k)$ and $\rho_k(i) \neq \rho'_k(i)$ for some $i \in \mathbf{dom}(\rho_k)$. If (a) is the case, $\mathbf{dom}(\rho_k) \neq \mathbf{dom}(\rho'_k)$ implies $\mathbf{dom}(\xi) = \bigcup \{\mathbf{dom}(\rho_k) | k \in K\} \neq \bigcup \{\mathbf{dom}(\rho'_k) | k \in K\} = \mathbf{dom}(\xi')$, thus $\xi \neq \xi'$. If (b) is the case, then $\xi(i) \neq \xi'(i)$ for some $i \in \mathbf{dom}(\rho_k)$, thus again $\xi \neq \xi'$. So π_Γ is indeed injective.

Since π_Γ is a bijection, it has an inverse π_Γ^{-1} , i.e. $\pi_\Gamma^{-1} : \odot \Gamma \longrightarrow \otimes \Gamma$ with $\pi_\Gamma^{-1}(\pi_\Gamma(\rho)) = \rho$ for all $\rho \in \otimes \Gamma$. We gave a characterization of π_Γ^{-1} in our lemma above and we now need to ensure that this characterization is correct. So let $\rho = [\rho_k | k \in K] = [[\rho_{k,l} | l \in L_k] | k \in K] \in \otimes \Gamma$ and $\xi := \pi_\Gamma(\rho)$. Note, that $\mathbf{dom}(\xi) = \bigcup \{L_k | k \in K\}$ and $\mathbf{dom}(\xi) \cap \otimes(\Gamma_k) = L_k$ for every $k \in K$. So applying our definition of π_Γ^{-1} on ξ we obtain $\pi_\Gamma^{-1}(\xi) := [\mathbf{pr}(\xi, \mathbf{dom}(\xi) \cap \otimes(\Gamma_k)) | k \in K] = [\mathbf{pr}(\xi, L_k) | k \in K] = [\rho_k | k \in K] = \rho$. Our characterization of π_Γ^{-1} is indeed correct.

Finally, let us assume that $P = [P_k | k \in K]$ is a class partition with $[\otimes(\Gamma_k) | k \in K] \subseteq P$. Again let $\rho = [\rho_k | k \in K] = [[\rho_{k,l} | l \in L_k] | k \in K] \in \otimes \Gamma$ and $\xi := \pi_\Gamma(\rho)$. Note, that for all $k_1, k_2 \in K$, $k_1 = k_2$ implies $\Gamma_{k_1} \subseteq P_{k_2}$ and $k_1 \neq k_2$ implies $\Gamma_{k_1} \cap P_{k_2} = \emptyset$. That leads to $\mathbf{dom}(\xi) \cap P_k = L_k$ for each $k \in K$. So applying the second characterization of π_Γ^{-1} on ξ we obtain $\pi_\Gamma^{-1}(\xi) := [\mathbf{pr}(\xi, \mathbf{dom}(\xi) \cap P_k) | k \in K] = [\mathbf{pr}(\xi, L_k) | k \in K] = [\rho_k | k \in K] = \rho$.

16.4.9 Example

Consider the graphs

$$\Gamma = \left\{ \langle \rangle, [a \mapsto 1], \begin{bmatrix} a \mapsto 2 \\ b \mapsto 3 \end{bmatrix} \right\}$$

$$\Delta = \emptyset$$

$$\Sigma = \left\{ \langle \rangle, \begin{bmatrix} c \mapsto 4 \\ d \mapsto 3 \end{bmatrix} \right\}$$

again with $\Gamma \dot{\bowtie} \Delta \dot{\bowtie} \Sigma$. So

$$\langle \Gamma, \Sigma \rangle := \begin{bmatrix} 1 \mapsto \left\{ \langle \rangle, [a \mapsto 1], \begin{bmatrix} a \mapsto 2 \\ b \mapsto 3 \end{bmatrix} \right\} \\ 2 \mapsto \left\{ \langle \rangle, \begin{bmatrix} c \mapsto 4 \\ d \mapsto 3 \end{bmatrix} \right\} \end{bmatrix}$$

is a relatively distinct graph record and its distinct product $\Gamma \odot \Sigma$ is well-defined with the result given in 16.4.5.

The function $\pi_{\langle \Gamma, \Sigma \rangle}$ is a reconstruction of $\Gamma \odot \Sigma$. It is given by

$$\pi_{\langle \Gamma, \Sigma \rangle} = \begin{bmatrix} \Gamma \times \Sigma \longrightarrow \Gamma \odot \Sigma \\ \langle \langle \rangle, \langle \rangle \rangle \mapsto \langle \rangle \\ \langle \langle \rangle, \begin{bmatrix} c \mapsto 4 \\ d \mapsto 3 \end{bmatrix} \rangle \mapsto \begin{bmatrix} c \mapsto 4 \\ d \mapsto 3 \end{bmatrix} \\ \langle [a \mapsto 1], \langle \rangle \rangle \mapsto [a \mapsto 1] \\ \langle [a \mapsto 1], \begin{bmatrix} c \mapsto 4 \\ d \mapsto 3 \end{bmatrix} \rangle \mapsto \begin{bmatrix} a \mapsto 1 \\ c \mapsto 4 \\ d \mapsto 3 \end{bmatrix} \\ \langle \begin{bmatrix} a \mapsto 2 \\ b \mapsto 3 \end{bmatrix}, \langle \rangle \rangle \mapsto \begin{bmatrix} a \mapsto 2 \\ b \mapsto 3 \end{bmatrix} \\ \langle \begin{bmatrix} a \mapsto 2 \\ b \mapsto 3 \end{bmatrix}, \begin{bmatrix} c \mapsto 4 \\ d \mapsto 3 \end{bmatrix} \rangle \mapsto \begin{bmatrix} a \mapsto 2 \\ b \mapsto 3 \\ c \mapsto 4 \\ d \mapsto 3 \end{bmatrix} \end{bmatrix}$$

Obviously, it is a bijection, and its inverse is

$$\pi_{\langle \Gamma, \Sigma \rangle}^{-1} = \begin{bmatrix} \Gamma \odot \Sigma \longrightarrow \Gamma \times \Sigma \\ \xi \mapsto \langle \mathbf{pr}(\xi, \mathbf{dom}(\xi) \cap \{1\}), \mathbf{pr}(\xi, \mathbf{dom}(\xi) \cap \{2\}) \rangle \end{bmatrix}$$

Another example of a relatively distinct graph record is

$$\langle \Gamma, \Delta, \Sigma \rangle := \begin{bmatrix} 1 \mapsto \left\{ \langle \rangle, [a \mapsto 1], \begin{bmatrix} a \mapsto 2 \\ b \mapsto 3 \end{bmatrix} \right\} \\ 2 \mapsto \emptyset \\ 3 \mapsto \left\{ \langle \rangle, \begin{bmatrix} c \mapsto 4 \\ d \mapsto 3 \end{bmatrix} \right\} \end{bmatrix}$$

But here, their cartesian as well as their distinct product is empty, i.e. $\Gamma \times \Delta \times \Sigma = \Gamma \odot \Delta \odot \Sigma = \emptyset$. So $\pi_{\langle \Gamma, \Delta, \Sigma \rangle} : \emptyset \longrightarrow \emptyset$ is the empty function, but by definition it is a bijection which is identical with its inverse $\pi_{\langle \Gamma, \Delta, \Sigma \rangle}^{-1} : \emptyset \longrightarrow \emptyset$.

16.4.10 Lemma

If $\Gamma = [\Gamma_k | k \in K]$ is a relatively distinct graph record, then

$$\mathbf{card}(\odot\Gamma) = \prod_{k \in K} \mathbf{card}(\Gamma_k)$$

16.4.11 Proof of 16.4.10

According to 16.4.7, there is a bijection between $\odot\Gamma$ and $\otimes\Gamma$, so $\mathbf{card}(\odot\Gamma) = \mathbf{card}(\otimes\Gamma)$. And according to 12.1.9, $\mathbf{card}(\otimes\Gamma) = \prod_{k \in K} \mathbf{card}(\Gamma_k)$.

16.5 Projections of graphs**16.5.1** Definition

For every graph Γ and each class J we define

$$\mathbf{pr}(\Gamma, J) := \{\mathbf{pr}(\xi, \mathbf{dom}(\xi) \cap J) \mid \xi \in \Gamma\}$$

the projection of Γ onto J

16.5.2 Lemma

If $[\Gamma_k | k \in K]$ is a relatively distinct graph record and J any class, then

$$\mathbf{pr}(\odot[\Gamma_k | k \in K], J) = \odot[\mathbf{pr}(\Gamma_k, \mathbb{0}(\Gamma_k) \cap J) \mid k \in K]$$

16.5.3 Proof of 16.5.2

$$\begin{aligned} & \mathbf{pr}(\odot[\Gamma_k | k \in K], J) \\ &= \{\mathbf{pr}(\xi, \mathbf{dom}(\xi) \cap J) \mid \xi \in \odot[\Gamma_k | k \in K]\} && \text{def. 16.5.2} \\ &= \left\{ \frac{\mathbf{pr}\left(\dot{\bigcup}_{k \in K} \rho_k, \bigcup_{k \in K} (\mathbf{dom}(\rho_k) \cap J)\right)}{[\rho_k | k \in K] \in \otimes[\Gamma_k | k \in K]} \right\} \\ &= \left\{ \dot{\bigcup}_{k \in K} \mathbf{pr}(\rho_k, \mathbf{dom}(\rho_k) \cap J) \mid [\rho_k | k \in K] \in \otimes[\Gamma_k | k \in K] \right\} && \text{def. 16.4.1} \\ &= \left\{ \dot{\bigcup}_{k \in K} \rho'_k \mid [\rho'_k | k \in K] \in \otimes[\mathbf{pr}(\Gamma_k, \mathbb{0}(\Gamma_k) \cap J) \mid k \in K] \right\} && \text{due to 16.2.10(3)} \\ &= \odot[\mathbf{pr}(\Gamma_k, \mathbb{0}(\Gamma_k) \cap J) \mid k \in K] && \text{def. 16.5.2} \\ & && \text{def. 16.4.1 again} \end{aligned}$$

16.6 Distributivity**16.6.1** Lemma distributivity

Let X be a schema and $Y, Z \in \mathbf{Proj}(X)$ with $Y \dot{\bowtie} Z$. For all $\Gamma_1, \Gamma_2 \in \mathbf{P}(\otimes Y)$, every $\{\Gamma_i \mid i \in I\} \subseteq \mathbf{P}(\otimes Y)$ and each $\Sigma \in \mathbf{P}(\otimes Z)$ holds:

- (1) $(\Gamma_1 \cap \Gamma_2) \odot \Sigma = (\Gamma_1 \odot \Sigma) \cap (\Gamma_2 \odot \Sigma)$
- (2) $(\Gamma_1 \cup \Gamma_2) \odot \Sigma = (\Gamma_1 \odot \Sigma) \cup (\Gamma_2 \odot \Sigma)$
- (3) $(\Gamma_1 \setminus \Gamma_2) \odot \Sigma = (\Gamma_1 \odot \Sigma) \setminus (\Gamma_2 \odot \Sigma)$
- (4) $\bigcup \{\Gamma_i \mid i \in I\} \odot \Sigma = \bigcup \{\Gamma_i \odot \Sigma \mid i \in I\}$
- (5) If $I \neq \emptyset$ then

$$\bigcap \{\Gamma_i \mid i \in I\} \odot \Sigma = \bigcap \{\Gamma_i \odot \Sigma \mid i \in I\}$$

16.6.2 Proof of 16.6.1

Recall 5.7.18, that for every function $\varphi : D \rightarrow C$ and each $D' \subseteq D$, the f -image class of D' was written

$$\varphi[D'] := \{\varphi(d) \mid d \in D'\}$$

We will make use of the fact, that if $\varphi : D \rightarrow C$ is a bijection, then

- (a) for every $\{D_i \mid i \in I\} \subseteq \mathbf{P}(D)$ with $I \neq \emptyset$ holds

$$\varphi[\bigcap \{D_i \mid i \in I\}] = \bigcap \{\varphi[D_i] \mid i \in I\}$$
- (b) for every $\{D_i \mid i \in I\} \subseteq \mathbf{P}(D)$ holds

$$\varphi[\bigcup \{D_i \mid i \in I\}] = \bigcup \{\varphi[D_i] \mid i \in I\}$$
- (c) for all $D_1, D_2 \in \mathbf{P}(D)$ holds

$$\varphi[D_1 \setminus D_2] = \varphi[D_1] \setminus \varphi[D_2]$$

Furthermore, we will use the facts (which are special cases of 12.4.3) that

- (d) for every schema $[C_j | j \in J]$, $J \neq \emptyset$, and each class D ,

$$\bigcap \{C_j \mid j \in J\} \times D = \bigcap \{C_j \times D \mid j \in J\}$$
- (e) for every schema $[C_j | j \in J]$ and each class D ,

$$\bigcup \{C_j \mid j \in J\} \times D = \bigcup \{C_j \times D \mid j \in J\}$$
- (f) for all classes C_1, C_2 and D holds

$$(C_1 \setminus C_2) \times D = (C_1 \times D) \setminus (C_2 \times D)$$

If $\Gamma \in \mathbf{P}(\otimes Y)$ and $\Sigma \in \mathbf{P}(\otimes Z)$, then $Y \dot{\bowtie} Z$ implies $\otimes Y \dot{\bowtie} \otimes Z$ and $\Gamma \dot{\bowtie} \Sigma$, so that $\otimes Y \odot \otimes Z$ and $\Gamma \odot \Sigma$ are well-defined. Using the image class notation and the definition of the bijection in 16.4.7, we have

$$(g) \quad \Gamma \odot \Sigma = \pi_{(\otimes Y, \otimes Z)}[\Gamma \times \Sigma]$$

We can now put everything together:

- (4) For every $\{\Gamma_i \mid i \in I\} \subseteq \mathbf{P}(\otimes Y)$ and $\Sigma \in \mathbf{P}(\otimes Z)$ holds

$$\begin{aligned} & \bigcup \{\Gamma_i \mid i \in I\} \odot \Gamma \\ &= \pi_{(\otimes Y, \otimes Z)}[\bigcup \{\Gamma_i \mid i \in I\} \times \Sigma] && \text{due to (g)} \\ &= \pi_{(\otimes Y, \otimes Z)}[\bigcup \{\Gamma_i \times \Sigma \mid i \in I\}] && \text{due to (e)} \\ &= \bigcup \{\pi_{(\otimes Y, \otimes Z)}[\Gamma_i \odot \Sigma] \mid i \in I\} && \text{due to (b)} \\ &= \bigcup \{\Gamma_i \odot \Sigma \mid i \in I\} && \text{due to (g)} \end{aligned}$$

- (5) If additionally $I \neq \emptyset$, then

$$\begin{aligned} & \bigcap \{\Gamma_i \mid i \in I\} \odot \Gamma \\ &= \pi_{(\otimes Y, \otimes Z)}[\bigcap \{\Gamma_i \mid i \in I\} \times \Sigma] && \text{due to (g)} \\ &= \pi_{(\otimes Y, \otimes Z)}[\bigcap \{\Gamma_i \times \Sigma \mid i \in I\}] && \text{due to (d)} \\ &= \bigcap \{\pi_{(\otimes Y, \otimes Z)}[\Gamma_i \odot \Sigma] \mid i \in I\} && \text{due to (a)} \\ &= \bigcap \{\Gamma_i \odot \Sigma \mid i \in I\} && \text{due to (g)} \end{aligned}$$

- (1) For every $\Gamma_1, \Gamma_2 \in \mathbf{P}(\otimes Y)$ and $\Sigma \in \mathbf{P}(\otimes Z)$, (1) is just two

special cases of (5) with $I = \{1, 2\}$

- (2) Similar to (1), (2) is a special case of (4).
 (3) There is

$$\begin{aligned}
 & (\Gamma_1 \setminus \Gamma_2) \odot \Sigma \\
 = & \pi_{(\otimes Y, \otimes Z)}[(\Gamma_1 \setminus \Gamma_2) \times \Sigma] && \text{due to (g)} \\
 = & \pi_{(\otimes Y, \otimes Z)}[(\Gamma_1 \times \Sigma) \setminus (\Gamma_2 \times \Sigma)] && \text{due to (f)} \\
 = & \pi_{(\otimes Y, \otimes Z)}[\Gamma_1 \setminus \Sigma] \setminus \pi_{(\otimes Y, \otimes Z)}[\Gamma_2 \setminus \Sigma] && \text{due to (c)} \\
 = & (\Gamma_1 \odot \Sigma) \setminus (\Gamma_2 \odot \Sigma) && \text{due to (g)}
 \end{aligned}$$

16.6.3 Remark more general products

The distinct product is only partially defined, it is restricted to relatively distinct graphs. It is not difficult to find more general definitions of “graph products” (see the \bowtie operator below). But the constraint pays off: the distributivity with the class operations (given in 16.6.1) holds for the distinct product, but is violated for generalizations like the following \bowtie .

Suppose, we define for any two (not necessarily relatively distinct) record classes Γ and Σ a more general product \bowtie by putting³¹

$$\Gamma \bowtie \Sigma := \{\xi \vee v \mid \xi \in \Gamma, v \in \Sigma, \xi \sim v\}$$

Obviously, if $\Gamma \checkmark \Sigma$, then $\Gamma \bowtie \Sigma = \Gamma \odot \Sigma$ and \bowtie is a generalization of \odot .

But a distributivity statement like

$$(\Gamma_1 \cap \Gamma_2) \bowtie \Sigma = (\Gamma_1 \bowtie \Sigma) \cap (\Gamma_2 \bowtie \Sigma)$$

is not true in general. For example, let

$$\begin{aligned}
 \Gamma_1 &= \{[a \mapsto 1]\} \\
 \Gamma_2 &= \{[b \mapsto 2]\} \\
 \Sigma &= \{[a \mapsto 1], [b \mapsto 2]\}
 \end{aligned}$$

then

$$\begin{aligned}
 & (\Gamma_1 \cap \Gamma_2) \bowtie \Sigma \\
 = & \emptyset \bowtie \Sigma \\
 = & \emptyset \\
 \neq & \left\{ \left[\begin{array}{l} a \mapsto 1 \\ b \mapsto 2 \end{array} \right] \right\} \\
 = & \left\{ [a \mapsto 1], [b \mapsto 2] \right\} \cap \left\{ [b \mapsto 2], \left[\begin{array}{l} a \mapsto 1 \\ b \mapsto 2 \end{array} \right] \right\} \\
 = & (\Gamma_1 \bowtie \Sigma) \cap (\Gamma_2 \bowtie \Sigma)
 \end{aligned}$$

16.6.4 Remark

There is another form of distributivity in connection with distinct products. Suppose, X_1, \dots, X_n are schemas with $X_1 \checkmark \dots \checkmark X_n$. In other words, they make a schema partition of $X = X_1 \dot{\vee} \dots \dot{\vee} X_n$. Then it doesn't matter if we either first generate their cartesian products $\otimes X_i$ and then the distinct product of these cartesian graphs, or if we first join these (pairwise) distinct schemas and then generate their cartesian product. Put formally, $(\otimes X_1) \odot \dots \odot (\otimes X_n) = \otimes(X_1 \dot{\vee} \dots \dot{\vee} X_n) = \otimes X$. A similar statement holds for the

star product $\otimes X$.

16.6.5 Lemma

If $[\sigma_k | k \in K]$ is a schema partition of a schema X , then

- (1) $\otimes X = \odot[\otimes \sigma_k | k \in K]$
 (2) $\otimes X = \odot[\otimes \sigma_k | k \in K]$
 So in particular, if X_1, \dots, X_n are (two or more) schemas with $X_1 \checkmark \dots \checkmark X_n$, then
 (3) $(\otimes X_1) \odot \dots \odot (\otimes X_n) = \otimes(X_1 \dot{\vee} \dots \dot{\vee} X_n)$
 (4) $(\otimes X_1) \odot \dots \odot (\otimes X_n) = \otimes(X_1 \dot{\vee} \dots \dot{\vee} X_n)$

16.6.6 Proof of 16.6.5

Let $\sigma = [\sigma_k | k \in K] = [[\sigma_{k,l} | l \in L_k] | k \in K]$ and $X = [X_i | i \in I]$ be given, where σ is a schema partition of X .

- (1) Since σ is a schema partition of X , there is (see 16.2.7)

$$\sigma_{k,l} = X_l \text{ for all } k \in K \text{ and } l \in L_k$$

as mentioned in 16.2.7. From 16.2.3 and 16.2.4 we know that $[L_k | k \in K]$ is a class partition of I , so

$$\bigcup \{L_k | k \in K\} = I$$

This implies the following equation

$$\left\{ \bigcup_{k \in K} M_k \mid [M_k | k \in K] \subseteq [L_k | k \in K] \right\} = \{J \mid J \subseteq I\}$$

which we will use below. Now,

$$\otimes \sigma_k = \left\{ [\rho_j | j \in M] \mid \begin{array}{l} M \subseteq L_k, \\ \rho_j \in X_j \text{ for all } j \in M \end{array} \right\}$$

for each $k \in K$, so that

$$\begin{aligned}
 & \odot[\otimes \sigma_k | k \in K] \\
 = & \left\{ \dot{\vee} \rho \mid \rho \in \otimes[\otimes \sigma_k | k \in K] \right\} \\
 = & \left\{ \dot{\vee} \rho \mid \rho \in \left\{ \frac{[[\rho_j | j \in M_k] | k \in K]}{[M_k | k \in K] \subseteq [L_k | k \in K], \rho_j \in X_j \text{ for all } k \in K \text{ and } j \in M_k} \right\} \right\} \\
 = & \left\{ [\rho_j | j \in \bigcup_{k \in K} M_k] \mid \begin{array}{l} [M_k | k \in K] \subseteq [L_k | k \in K], \\ \rho_j \in X_j \text{ for all } j \in \bigcup_{k \in K} M_k \end{array} \right\} \\
 = & \left\{ [\rho_j | j \in J] \mid \begin{array}{l} J \subseteq I \\ \rho_j \in X_j \text{ for all } j \in J \end{array} \right\} \\
 = & \otimes[X_i | i \in I] \\
 = & \otimes X
 \end{aligned}$$

- (2) For every $k \in K$,

$$\otimes \sigma_k = \{[\rho_j | j \in L_k] \mid \rho_j \in X_j \text{ for all } j \in L_k\}$$

so that

$$\begin{aligned}
 & \odot[\otimes \sigma_k | k \in K] \\
 = & \left\{ \dot{\vee} \rho \mid \rho \in \otimes[\otimes \sigma_k | k \in K] \right\} \\
 = & \left\{ \dot{\vee} \rho \mid \rho \in \left\{ \frac{[[\rho_j | j \in L_k] | k \in K]}{\rho_j \in X_j \text{ for all } k \in K \text{ and } j \in L_k} \right\} \right\} \\
 = & \left\{ [\rho_j | j \in \bigcup_{k \in K} L_k] \mid \rho_j \in X_j \text{ for all } k \in K \text{ and } j \in L_k \right\} \\
 = & \{[\rho_j | j \in I] \mid \rho_j \in X_j \text{ for all } j \in I\} \\
 = & \otimes X
 \end{aligned}$$

If X_1, \dots, X_n are (two or more) schemas with $X_1 \checkmark \dots \checkmark X_n$, then $\langle X_1, \dots, X_n \rangle$ is a schema partition of $X_1 \dot{\vee} \dots \dot{\vee} X_n$. Thus

³¹ The notation \bowtie resembles the *natural join* operation \bowtie on relations, a common operation and notation in relational databank theory.

- (3) is just is special case of (1), and
 (4) derives from (2)

16.6.7 Example

Let

$$X = \begin{bmatrix} a \mapsto \{1, 2\} \\ b \mapsto \{3\} \\ c \mapsto \{2, 3\} \end{bmatrix}$$

A schema partition of X is given by $\langle X_1, X_2 \rangle$ with

$$X_1 = \begin{bmatrix} a \mapsto \{1, 2\} \\ b \mapsto \{3\} \end{bmatrix} \quad \text{and} \quad X_2 = \begin{bmatrix} c \mapsto \{2, 3\} \end{bmatrix}$$

A verification of 16.6.5(4) is then given by:

$$\begin{aligned} & (\otimes X_1) \odot (\otimes X_2) \\ = & \odot \langle \otimes X_1, \otimes X_2 \rangle \\ = & \left\{ \dot{\vee} \rho \mid \rho \in \langle \otimes X_1, \otimes X_2 \rangle \right\} \\ = & \left\{ \dot{\vee} \rho \mid \rho \in \left\{ \left[\begin{array}{l} 1 \mapsto \rho_1 \\ 2 \mapsto \rho_2 \end{array} \right] \mid \begin{array}{l} \rho_1 \in \otimes X_1 \\ \rho_2 \in \otimes X_2 \end{array} \right\} \right\} \\ = & \left\{ \dot{\vee} \rho \mid \rho \in \left\{ \left[\begin{array}{l} 1 \mapsto \left[\begin{array}{l} a \mapsto \rho_{1,a} \\ b \mapsto \rho_{1,b} \end{array} \right] \\ 2 \mapsto \left[\begin{array}{l} c \mapsto \rho_{2,c} \end{array} \right] \end{array} \right] \mid \begin{array}{l} \rho_{1,a} \in \{1, 2\} \\ \rho_{1,b} \in \{3\} \\ \rho_{2,c} \in \{2, 3\} \end{array} \right\} \right\} \\ = & \left\{ \left[\begin{array}{l} a \mapsto \rho_{1,a} \\ b \mapsto \rho_{1,b} \\ c \mapsto \rho_{2,c} \end{array} \right] \mid \begin{array}{l} \rho_{1,a} \in \{1, 2\} \\ \rho_{1,b} \in \{3\} \\ \rho_{2,c} \in \{2, 3\} \end{array} \right\} \\ = & \left\{ \left[\begin{array}{l} a \mapsto \rho_a \\ b \mapsto \rho_b \\ c \mapsto \rho_c \end{array} \right] \mid \begin{array}{l} \rho_a \in \{1, 2\} \\ \rho_b \in \{3\} \\ \rho_c \in \{2, 3\} \end{array} \right\} \\ = & \otimes \begin{bmatrix} a \mapsto \{1, 2\} \\ b \mapsto \{3\} \\ c \mapsto \{2, 3\} \end{bmatrix} \\ = & \otimes \left(\begin{bmatrix} a \mapsto \{1, 2\} \\ b \mapsto \{3\} \end{bmatrix} \dot{\vee} \begin{bmatrix} c \mapsto \{2, 3\} \end{bmatrix} \right) \\ = & \otimes (X_1 \dot{\vee} X_2) \end{aligned}$$

16.6.8 Lemma

For every schema $X = [X_i \mid i \in I]$ holds:

- (1) $\odot [\otimes [i \mapsto X_i] \mid i \in I] = \otimes X$
 (2) $\odot [\otimes [i \mapsto X_i] \mid i \in I] = \otimes X$
 (3) $\odot [\oplus [i \mapsto X_i] \mid i \in I] = \otimes X$

16.6.9 Proof of 16.6.8

$[i \mapsto X_i] \mid i \in I$ is obviously a schema partition of X . So

(1) is just a special case of 16.6.5(1), and

(2) is a special case of 16.6.5.

(3) There is $\oplus [i \mapsto X_i] = \otimes [i \mapsto X_i]$, for every $i \in I$, so that

$$\odot [\oplus [i \mapsto X_i] \mid i \in I] = \odot [\otimes [i \mapsto X_i] \mid i \in I] = \otimes X$$

according to (2).

16.6.10 Example

If we consider the schema

$$X = \begin{bmatrix} a \mapsto \{1, 2\} \\ b \mapsto \{3\} \\ c \mapsto \{2, 3\} \end{bmatrix}$$

from example 16.6.7 again and put

$$X_a := [a \mapsto \{1, 2\}] \quad X_b := [b \mapsto \{3\}] \quad X_c := [c \mapsto \{2, 3\}]$$

we obtain a confirmation of 16.6.8 by

$$\begin{aligned} & (\otimes X_a) \odot (\otimes X_b) \odot (\otimes X_c) \\ = & \left\{ \begin{array}{l} \langle \rangle, \\ [a \mapsto 1], \\ [a \mapsto 2] \end{array} \right\} \odot \left\{ \begin{array}{l} \langle \rangle, \\ [b \mapsto 3] \end{array} \right\} \odot \left\{ \begin{array}{l} \langle \rangle, \\ [c \mapsto 2], \\ [c \mapsto 3] \end{array} \right\} \\ = & \otimes X \\ & (\otimes X_a) \odot (\otimes X_b) \odot (\otimes X_c) \\ = & \left\{ \begin{array}{l} [a \mapsto 1], \\ [a \mapsto 2] \end{array} \right\} \odot \{ [b \mapsto 3] \} \odot \left\{ \begin{array}{l} [c \mapsto 2], \\ [c \mapsto 3] \end{array} \right\} \\ = & \otimes X \\ & (\oplus X_a) \odot (\oplus X_b) \odot (\oplus X_c) \\ = & \left\{ \begin{array}{l} [a \mapsto 1], \\ [a \mapsto 2] \end{array} \right\} \odot \{ [b \mapsto 3] \} \odot \left\{ \begin{array}{l} [c \mapsto 2], \\ [c \mapsto 3] \end{array} \right\} \\ = & \otimes X \end{aligned}$$

Part VII

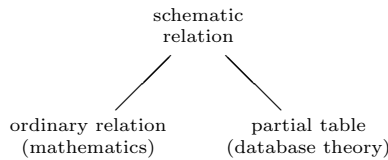
Relations

17 Relations

17.1 Introduction

17.1.1 Remark a more general relation concept

We already introduced and used quite a lot *ordinary (or ordinary) relations*, they are essential and standard in mathematics.³² In database theory there is another relation concept, the *partial table*, as we call it here. We will now introduce a more general definition of a (*schematic*) *relation* which is able to treat both ordinary relations and partial tables as special cases.



17.1.2 Remark representations

One main objective here in chapter VII is an understanding of the many ways in which relations can be represented. This whole variation may be tiresome at first, but it really pays off for a good understanding of the many facets of these things called relations.

- ♣ Our default and most general representation is conform to the standard appearance of operations

$$\left[\begin{array}{c} \text{syntax} \\ \text{semantics} \end{array} \right]$$

described in 4.3.1 and 5.3, which was already applied to ordinary relations. However, even this representation has some variations, which are supposed to be convenient, not confusing.

- ♣ Every relation R can be seen as a function, the *characteristic function* χ_R (17.4.2).
- ♣ A certain subclass of relations, called *tables* (17.5.3), can be represented as such.
- ♣ And a certain class of tables, called *completely finite relations*, can also be displayed by either their *boolean tables* (17.5.4) or their *double tables* (17.6.1).
- ♣ Allowing *null values* at first leads to a generalization and we call that a *quasi-relation*. But we will understand null values in a way that turns quasi-relations into relations again.

There are other suggestions for representations which are very suitable for implementations in common computer systems. But for now and our purposes here, that will do.

17.1.3 Remark partial tables and database theory

A *partial table* (or *Codd relation*) is a table that allows possible *null* values, similar to a *partial function* that is a function with possibly *undefined* arguments. Partial tables are pretty much the kind of relations used in database theory. However, the terminology is not standard, just “table” or “relation” is more common. Neither is the interpretation of partial tables and null values seem to be undisputed, not even in database theory itself.

Our interpretation is closely related to our concept of relation *expansions* and that smoothly fits into an elegant algebra that is very close to the behavior of logic. That way we are probably more close to Codd’s original ideas than the various implementations of the relational database language SQL.³³

In 18 we introduce *partial tables* as a particular finite version of *quasi-relations* in general. And we will see, how these quasi-relations are well covered by our relation concept from 17.

17.2 General definition

17.2.1 Definition relation

A *relation* R is essentially made of a schema X and a graph Γ , where $\Gamma \subseteq \otimes X$. The default form for R is

$$R = \left[\begin{array}{c} X \\ \Gamma \end{array} \right] \quad \text{or} \quad R = [X, \Gamma]$$

17.2.2 Definition relation

If $R = [X, \Gamma]$ is a relation with $X = [X_i | i \in I]$ we put

$@(R)$	$:= I$	the <u>attribute</u> or <u>index</u> class of R
$\mathbf{x}(R)$	$:= X$	the <u>schema</u> of R
$\mathbf{gr}(R)$	$:= \Gamma$	the <u>graph</u> of R
$\mathbf{dom}(R)$	$:= \otimes X$	the <u>domain</u> of R
$\mathbf{dom}_i(R)$	$:= X_i$	the <u>i-domain</u> of R , for each $i \in I$

17.2.3 Remark deconstruction

- (1) In other words, if R is an arbitrary relation R then

$$R = \left[\begin{array}{c} \mathbf{x}(R) \\ \mathbf{gr}(R) \end{array} \right] = \left[\begin{array}{c} [\mathbf{dom}_i(R) | i \in @(R)] \\ \mathbf{gr}(R) \end{array} \right]$$

³² At this point, it might be useful to recall the design of basic mathematical concepts in 4 and 5, in particular the introduction of *predicators* and *ordinary relations* in 5.7.

³³ See e.g. on *wikipedia.org* the according articles such as “Relational algebra”.

(2) Note, that for a given relation $R = [X, \Gamma]$ with $X = [X_i | i \in I]$, the domain of R and the domain of X are different things: $\mathbf{dom}(R) = \otimes X$ and $\mathbf{dom}(X) = @R = I$. By a *domain* of an operation we mean the class of its possible arguments. For the schema X that is I , and for R that is the record class $\otimes X$.

17.2.4 Example relation

Let us consider the following schema

$$\text{Date} := \left[\begin{array}{l} \text{year} \mapsto \mathbb{Z} \\ \text{month} \mapsto \{1, \dots, 12\} \\ \text{day} \mapsto \{1, \dots, 31\} \\ \text{title} \mapsto \{\text{monday}, \dots, \text{sunday}\} \end{array} \right]$$

and two elements of its cartesian product $\otimes \text{Date}$

$$d_1 := \left[\begin{array}{l} \text{year} \mapsto 2003 \\ \text{month} \mapsto 1 \\ \text{day} \mapsto 18 \\ \text{title} \mapsto \text{saturday} \end{array} \right] \quad d_2 := \left[\begin{array}{l} \text{year} \mapsto 2003 \\ \text{month} \mapsto 1 \\ \text{day} \mapsto 19 \\ \text{title} \mapsto \text{sunday} \end{array} \right]$$

A relation is then given by

$$\text{ThisWeekend} := \left[\begin{array}{c} \text{Date} \\ \{d_1, d_2\} \end{array} \right]$$

As we will define it properly below, a relation like this with a finite attribute class and a finite graph is usually better be represented by its graph table (diagram) (see below in 17.5.4), which is

year :	month :	day :	title :
\mathbb{Z}	$\{1, \dots, 12\}$	$\{1, \dots, 31\}$	$\{\text{monday}, \dots, \text{sunday}\}$
2003	1	18	saturday
2003	1	19	sunday

A decomposition of **ThisWeekend** returns:

$$\begin{aligned} @(\text{ThisWeekend}) &= \{\text{year}, \text{month}, \text{day}, \text{title}\} \\ \mathbf{x}(\text{ThisWeekend}) &= \text{Date} \\ \mathbf{dom}(\text{ThisWeekend}) &= \otimes \text{Date} \\ \mathbf{dom}_{\text{year}}(\text{ThisWeekend}) &= \mathbb{Z} \\ \mathbf{dom}_{\text{month}}(\text{ThisWeekend}) &= \{1, \dots, 12\} \\ \mathbf{dom}_{\text{day}}(\text{ThisWeekend}) &= \{1, \dots, 31\} \\ \mathbf{dom}_{\text{title}}(\text{ThisWeekend}) &= \{\text{monday}, \dots, \text{sunday}\} \\ \mathbf{gr}(\text{ThisWeekend}) &= \{d_1, d_2\} \end{aligned}$$

Another relation with the the same schema is the calendar of 2003, which has not only the two members d_1 and d_2 , but 365 records in its graph.

17.2.5 Definition membership

Let $R = [X, \Gamma]$ be a relation. For every $x \in \mathbf{dom}(R)$ we write as usual:

$$\begin{aligned} x \in R &\text{ iff } x \in \Gamma \text{ saying that } x \text{ is a } \underline{\text{member}} \text{ or } R \text{ or } \underline{\text{in}} \ R \\ x \notin R &\text{ iff } x \in \Gamma \text{ saying that } x \text{ is a not a member, } \\ &\text{ i.e. } x \text{ is } \underline{\text{not in}} \ R \end{aligned}$$

17.2.6 Definition relation class

REL denotes the overall relation class.

For every schema X we define

$$\mathbf{Rel}(X) := \left\{ \left[\begin{array}{c} X \\ \Gamma \end{array} \right] \mid \Gamma \subseteq \otimes X \right\}$$

the relation class on X .

17.2.7 Remark relation type expression

The fact that R is a relation with the schema X is expressed by the type expression

$$R : \mathbf{Rel}(X)$$

(Type expressions in general were defined in 5.3.5).

For example 17.2.4 we have the type expression

$$\text{ThisWeekend} : \mathbf{Rel}(\text{Date})$$

17.2.8 Remark empty schema relations

$\langle \rangle$ is a well defined schema, the empty schema. There are precisely two relations with this schema:

$$\mathbf{Rel}(\langle \rangle) = \left\{ \left[\begin{array}{c} \langle \rangle \\ \emptyset \end{array} \right], \left[\begin{array}{c} \langle \rangle \\ \{\langle \rangle\} \end{array} \right] \right\}$$

In 5.6.8 we suggested to write

$$\mathbb{E} \text{ instead of } \mathbf{Rel}(\langle \rangle)$$

and in 17.5.7 below we introduce yet another notation for its two members.

17.2.9 Definition proper relations

A relation $R = [X, \Gamma]$ is proper iff X is proper.

17.2.10 Remark proper relations

Recall 9.3.3 and 12.1.12, that for every schema $X = [X_i | i \in I]$ holds:

$$X \text{ is proper} \Leftrightarrow X_i \neq \emptyset \text{ for all } i \in I \Leftrightarrow \otimes X \neq \emptyset$$

and thus:

(1) $\mathbf{Rel}(X) = \{[X, \emptyset]\}$ iff X is not proper.

(2) If X is proper, then $\mathbf{Rel}(X)$ has at least two elements.

(3) No matter what the schema X looks like, $\mathbf{Rel}(X)$ is never empty.

We usually assume the schemas involved to be proper and things are mostly pretty pointless if they are not. However, we don't make that assumption a quiet agreement in the sequel and call the situation "proper" in case it is.

17.2.11 Definition

Let $R = [[X_i | i \in I], \Gamma]$ be a relation. We define

$$\mathbf{rng}_i(R) := \{x_i \mid x \in \Gamma\}$$

the i -th range of R , for each $i \in I$, and

$$\mathbf{rng}(R) := \{x_i \mid i \in I, x \in \Gamma\}$$

is the (overall) range of R .

17.2.12 Example

Let $\mathbf{10} = \{1, \dots, 10\}$ and $\gcd(n, m)$ be the greatest common divisor of two natural numbers n and m . Let D be the binary endorelation, given by

$$D = \left[\begin{array}{c} \mathbf{10} \rightsquigarrow \mathbf{10} \\ \langle n, m \rangle \rightsquigarrow (n < m \text{ and } \gcd(n, m) > 1) \end{array} \right]$$

Since D is a table, we can as well represent it by

$$D = \begin{array}{|c|c|} \hline \mathbf{1 : 10} & \mathbf{2 : 10} \\ \hline 2 & 4 \\ \hline 2 & 6 \\ \hline 2 & 8 \\ \hline 3 & 6 \\ \hline 3 & 9 \\ \hline 4 & 8 \\ \hline \end{array}$$

We have

$$\begin{aligned} \mathbf{dom}_1(D) &= \mathbf{10} & \mathbf{dom}_2(D) &= \mathbf{10} \\ \mathbf{rng}_1(D) &= \{2, 3, 4\} & \mathbf{rng}_2(D) &= \{4, 6, 8, 9\} \end{aligned}$$

and so

$$\mathbf{rng}(D) = \{2, 3, 4, 6, 8, 9\}$$

17.2.13 Remark

Note that for every relation $R = [[X_i | i \in I], \Gamma]$ holds

- (1) $\mathbf{rng}_i(R) \subseteq \mathbf{dom}_i(R)$, for each $i \in I$
- (2) $\mathbf{rng}(R) \subseteq \bigcup \{X_i \mid i \in I\}$

17.3 Ordinary relations as (schematic) relations and other notational variations

17.3.1 Definition notation

According to 5.7.10, the default form of an ordinary relation R is the *typed-predicator form*

$$R = \left[\begin{array}{c} D_1 \rightsquigarrow \dots \rightsquigarrow D_n \\ \langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi \end{array} \right]$$

To fit into the new default form of definition 17.2.1, called *schema-graph form*, we transform R into

$$R = \left[\begin{array}{c} \langle D_1, \dots, D_n \rangle \\ \{\langle x_1, \dots, x_n \rangle \in D_1 \times \dots \times D_n \mid \varphi\} \end{array} \right]$$

17.3.2 Remark ordinal relation class

For every ordinal schema (i.e. class tuple) $\langle D_1, \dots, D_n \rangle$ holds

$$\mathbf{Rel}(\langle D_1, \dots, D_n \rangle) = \begin{cases} D_1 \rightsquigarrow \dots \rightsquigarrow D_n & \text{if } n \geq 2 \\ \mathbf{Pty}(D_1) & \text{if } n = 1 \\ \mathbb{B} & \text{if } n = 0 \end{cases}$$

as defined in 5.6.8 and due to the convention in 17.3.1.

If some $R \in \langle D_1 \rightsquigarrow \dots \rightsquigarrow D_n \rangle$ is given, then $R \in \mathbf{Rel}(\langle D_1, \dots, D_n \rangle)$ according to 17.3.1. On the other hand, if some

$$R = \left[\begin{array}{c} \langle D_1, \dots, D_n \rangle \\ \Gamma \end{array} \right] \in \mathbf{Rel}(\langle D_1, \dots, D_n \rangle)$$

is given, we have

$$R = \left[\begin{array}{c} D_1 \rightsquigarrow \dots \rightsquigarrow D_n \\ \langle x_1, \dots, x_n \rangle \rightsquigarrow \langle x_1, \dots, x_n \rangle \in \Gamma \end{array} \right]$$

to write R in the default form for ordinal relations.

17.3.3 Example ordinal relation

The usual linear order \leq on the integers is a binary relation, which can be written in our standard notation for ordinal relations as

$$\textcircled{1} \leq \textcircled{2} = \left[\begin{array}{c} \mathbb{Z} \rightsquigarrow \mathbb{Z} \\ \langle n, m \rangle \rightsquigarrow \exists d \in \mathbb{N} . n + d = m \end{array} \right]$$

So that is the *typed-predicator form*: first the type $\mathbb{Z} \rightsquigarrow \mathbb{Z}$, then the predicator $\langle n, m \rangle \rightsquigarrow \exists d \in \mathbb{N} . n + d = m$. We have

$$\mathbf{dom}(\leq) = \mathbb{Z} \times \mathbb{Z} = \otimes \langle \mathbb{Z}, \mathbb{Z} \rangle$$

$$\mathbf{x}(\leq) = \langle \mathbb{Z}, \mathbb{Z} \rangle$$

$$\mathbf{@}(\leq) = \{1, 2\}$$

$$\mathbf{dom}_1(\leq) = \mathbb{Z}$$

$$\mathbf{dom}_2(\leq) = \mathbb{Z}$$

$$\mathbf{gr}(\leq) = \{\langle n, m \rangle \in (\mathbb{Z} \times \mathbb{Z}) \mid \exists d \in \mathbb{N} . n + d = m\}$$

In *schema-graph form* we have

$$\textcircled{1} \leq \textcircled{2} = \left[\begin{array}{c} \langle \mathbb{Z}, \mathbb{Z} \rangle \\ \{ \langle n, m \rangle \in (\mathbb{Z} \times \mathbb{Z}) \mid \exists d \in \mathbb{N} . n + d = m \} \end{array} \right]$$

17.3.4 Remark

We could as well use yet another variation, the *schema-predicator* form. For \leq that is

$$\textcircled{1} \leq \textcircled{2} = \left[\begin{array}{c} \langle \mathbb{Z}, \mathbb{Z} \rangle \\ \langle n, m \rangle \rightsquigarrow \exists d \in \mathbb{N} . n + d = m \end{array} \right]$$

which is a bit more compact and less redundant again. Let us summarize the motivation and taxonomy of the various forms.

17.3.5 Remark summary of the notations

In 4.3.1 and 5.3 we explained that we use

$$\left[\begin{array}{c} \text{syntax} \\ \text{semantics} \end{array} \right]$$

as the standard form for operators. Relations are also operators and fit in as well. But there is a certain degree of variation in the precise formalization.

- ♣ The syntax of a relation $R = [X, \Gamma]$ is given either by its *schema* X or its *domain* $\otimes X$ or by its *type* $\mathbf{Rel}(X)$. Each of the three specifications has the same information.
- ♣ The semantics of R is given by its *graph* Γ . But we have the same information if we replace the graph $\Gamma = \{x \in \otimes X \mid \varphi\}$ by a *predicator* $x \rightsquigarrow \varphi$, as we already did (in 4.3.1 and 5.7.10) for ordinary relations.

This gives us the following six variations for equivalent formalizations of a given relation R :

syntax \ semantics	graph form	predicator form
<u>schema form</u>	$\left[\begin{array}{c} X \\ \Gamma \end{array} \right]$ This is our default form for schematic relations in general, as defined in 17.2.1.	$\left[\begin{array}{c} X \\ x \rightsquigarrow \varphi \end{array} \right]$ This is also used in our text, because it often is less redundant than the default form.
<u>domain form</u>	$\left[\begin{array}{c} \otimes X \\ \Gamma \end{array} \right]$ We never use that.	$\left[\begin{array}{c} \otimes X \\ x \rightsquigarrow \varphi \end{array} \right]$ Again, we never use that.
<u>typed form</u>	$\left[\begin{array}{c} \mathbf{Rel}(X) \\ \Gamma \end{array} \right]$ is the general version and we never use that. There is the special version for ordinal relations: $\left[\begin{array}{c} X_1 \rightsquigarrow \dots \rightsquigarrow X_n \\ \Gamma \end{array} \right]$ but we never use that, either.	$\left[\begin{array}{c} \mathbf{Rel}(X) \\ x \rightsquigarrow \varphi \end{array} \right]$ is the general version and we never use that, either. The special version for ordinal relations $\left[\begin{array}{c} X_1 \rightsquigarrow \dots \rightsquigarrow X_n \\ x \rightsquigarrow \varphi \end{array} \right]$ is the default notation as defined in 5.7.10.

17.4 Relations as functions and vice versa

17.4.1 Repetition

- (1) Recall from 5.8.4, that

$$\mathbb{B} := \{ \mathbf{0}, \mathbf{1} \}$$

is the bit value or boolean value class.

- (2) For an earlier introduction of the characteristic function, recall 5.7.14.

17.4.2 Definition characteristic function

- (1) If $R = [X, \Gamma]$ is a relation, then

$$\chi_R := \left[\begin{array}{c} \otimes X \longrightarrow \mathbb{B} \\ x \mapsto \begin{cases} \mathbf{0} & \text{if } x \in \Gamma \\ \mathbf{1} & \text{if } x \notin \Gamma \end{cases} \end{array} \right]$$

is the characteristic function of R

- (2) On the other hand, if X is a schema and $\chi : \otimes X \longrightarrow \mathbb{B}$, then

$$\mathbf{rel}(\chi) := \left[\begin{array}{c} X \\ x \rightsquigarrow \xi(x) = 1 \end{array} \right] \quad \text{is the } \underline{\text{relation}} \text{ of } \chi$$

17.5 Tables and table representation

17.5.1 Repetition

- Recall from 9.3.3, that a schema $X = [X_i | i \in I]$ is called
- (1) finite iff I is finite (i.e. X as a record is a finite record)
 - (2) completely finite iff I is finite and X_i is finite, for each $i \in I$.

17.5.2 Definition finiteness properties

We say that a relation $R = [[X_i | i \in I], \Gamma]$

- (1) has a finite table or is a table iff I is finite and Γ is finite,
- (2) is completely finite iff X is completely finite.

In that case, I and each X_i is finite, and thus Γ is finite, too.

17.5.3 Definition table class

For every schema X we define

$$\mathbf{Tab}(X) := \left\{ \left[\begin{array}{c} Y \\ \Gamma \end{array} \right] \mid \begin{array}{l} Y \leq X \\ \mathbf{dom}(Y) \text{ is finite} \\ \Gamma \text{ is finite} \end{array} \right\}$$

the table class on X

17.5.4 Remark graph tables and boolean tables

If $R = [[X_i | i \in I], \Gamma]$ has a finite table (or is a table), i.e. if I and Γ are finite with

$$I = \{i_1, \dots, i_n\}$$

$$\Gamma = \left\{ \left[\begin{array}{c} i_1 \mapsto x_{1,1} \\ \vdots \\ i_n \mapsto x_{1,n} \end{array} \right], \dots, \left[\begin{array}{c} i_1 \mapsto x_{m,1} \\ \vdots \\ i_n \mapsto x_{m,n} \end{array} \right] \right\}$$

then it is usually much more convenient to represent R by its graph table (diagram)

$i_1 : X_i$	\dots	$i_n : X_n$
$x_{1,1}$	\dots	$x_{1,n}$
\vdots		\vdots
$x_{m,1}$	\dots	$x_{m,n}$

If R is even completely finite, i.e. if furthermore each domain is finite with

$$X_1 = \{x_{1,1}, \dots, x_{1,k_1}\}, \dots, X_n = \{x_{n,1}, \dots, x_{n,k_n}\}$$

then R can also be written as a boolean table (diagram),

i_1	\dots	i_n	
$x_{1,1}$	\dots	$x_{n,1}$	β_1
\vdots		\vdots	\vdots
x_{1,k_1}	\dots	x_{n,k_1}	β_{k_1}
\vdots		\vdots	\vdots
x_{1,k_n}	\dots	x_{n,k_n}	$\beta_{i_1 \dots i_n}$

The left side of the table contains all the $i_1 \dots i_n$ elements of the product $\otimes [X_i | i \in I]$. Each boolean or bit value $\beta \in \mathbb{B} = \{0, 1\}$ on the right side indicates, if the element on the left is a member of the graph ($\beta = 1$) or if it isn't ($\beta = 0$).

Note that for both kinds of tables, the relation can entirely be recovered from the table. However, the table diagrams are not unique in general, since there is no pre-defined order on the columns and rows.

17.5.5 Example

The relation **ThisWeekend** from example 17.2.4 was given by the graph table

year :	month :	day :	title :
\mathbb{Z}	$\{1, \dots, 12\}$	$\{1, \dots, 31\}$	$\{\text{monday}, \dots, \text{sunday}\}$
2003	1	18	saturday
2003	1	19	sunday

We could change the order of the rows and obtain

year :	month :	day :	title :
\mathbb{Z}	$\{1, \dots, 12\}$	$\{1, \dots, 31\}$	$\{\text{monday}, \dots, \text{sunday}\}$
2003	1	19	sunday
2003	1	18	saturday

but we consider these two different diagrams as graphical representations of the same table.

This relation is not completely finite, its **year**-domain is the infinite class \mathbb{Z} of integers. It cannot be represented by a boolean table. (At least not in the strict sense. One could use (horizontal and vertical) dots “...” to display it anyway.)

17.5.6 Example

If we take the schema

$$X = \left[\begin{array}{l} \text{one} \mapsto \{1, 2\} \\ \text{two} \mapsto \{2, 3, 4\} \\ \text{three} \mapsto \{5\} \end{array} \right]$$

and a subset Γ of $\otimes X$, say

$$\Gamma = \left\{ \left[\begin{array}{l} \text{one} \mapsto 1 \\ \text{two} \mapsto 3 \\ \text{three} \mapsto 5 \end{array} \right], \left[\begin{array}{l} \text{one} \mapsto 2 \\ \text{two} \mapsto 4 \\ \text{three} \mapsto 5 \end{array} \right] \right\}$$

then $R := [X, \Gamma]$ is represented by its graph table

$$R = \begin{array}{|c|c|c|} \hline \text{one} : \{1, 2\} & \text{two} : \{2, 3, 4\} & \text{three} : \{5\} \\ \hline 1 & 3 & 5 \\ \hline 2 & 4 & 5 \\ \hline \end{array}$$

Since R is completely finite, we can as well represent it by its boolean table

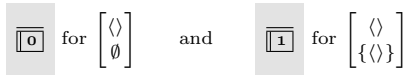
$$R = \begin{array}{|c|c|c|c|} \hline \text{one} & \text{two} & \text{three} & \\ \hline 1 & 2 & 5 & \mathbf{0} \\ 2 & 2 & 5 & \mathbf{0} \\ 1 & 3 & 5 & \mathbf{1} \\ 2 & 3 & 5 & \mathbf{0} \\ 1 & 4 & 5 & \mathbf{0} \\ 2 & 4 & 5 & \mathbf{1} \\ \hline \end{array}$$

We mentioned that the order of the rows and columns are not unique in general. So we can as well write the same boolean table as

$$R = \begin{array}{|c|c|c|c|} \hline \text{three} & \text{two} & \text{one} & \\ \hline 5 & 4 & 2 & \mathbf{1} \\ 5 & 4 & 1 & \mathbf{0} \\ 5 & 3 & 2 & \mathbf{0} \\ 5 & 3 & 1 & \mathbf{1} \\ 5 & 2 & 2 & \mathbf{0} \\ 5 & 2 & 1 & \mathbf{0} \\ \hline \end{array}$$

17.5.7 Remark table representation of empty schema relations.

As mentioned in 17.2.8, $\mathbf{Rel}(\langle \rangle)$ has two members. Their table diagrams are



respectively. (Later on in ??, we introduce yet another notation for these two important relations and write \perp and \top , respectively.)

17.5.8 Remark generalized table notation

Yet another and often convenient notation for a relation with a finite schema is a generalized table notation

$i_1 : X_1$...	$i_n : X_n$
x_1	...	x_n
$\varphi(x_1, \dots, x_n)$		

which stands for

$$\left[\begin{array}{c} \left[\begin{array}{c} i_1 \mapsto X_1 \\ \vdots \\ i_n \mapsto X_n \end{array} \right] \\ \left[\begin{array}{c} i_1 \mapsto x_1 \\ \vdots \\ i_n \mapsto x_n \end{array} \right] \rightsquigarrow \varphi(x_1, \dots, x_n) \end{array} \right]$$

where $\varphi(x_1, \dots, x_n)$ is a formula with all free variables among the x_1, \dots, x_n .

For example, we can define a “spring relation” for the according season (on the northern hemisphere) by

year :	month :	day :	title :
\mathbb{Z}	$\{1, \dots, 12\}$	$\{1, \dots, 31\}$	$\{\text{monday}, \dots, \text{sunday}\}$
y	m	d	t
$(m = 3 \wedge d \geq 20) \vee m = 4 \vee m = 5 \vee (m = 6 \wedge d \leq 21)$			

17.6 Double tables

17.6.1 Remark double tables

There is yet another variation of the one-dimensional boolean table representation: two-dimensional so-called double tables. Besides being more compact, they are useful in understanding operations like schema reductions and expansion (see 21.1).

Consider a completely finite relation R with a schema

$$X = \left[\begin{array}{c} i_1 \mapsto X_1 \\ \vdots \\ i_n \mapsto X_n \end{array} \right]$$

Its boolean table has the general form

$i_1 \dots i_n$	
$\otimes X$	boolean values of R

Now let us split X into two distinct schemas

$$Y = \left[\begin{array}{c} i_1 \mapsto X_1 \\ \vdots \\ i_k \mapsto X_k \end{array} \right] \quad \text{and} \quad Z = \left[\begin{array}{c} i_{k+1} \mapsto X_{k+1} \\ \vdots \\ i_n \mapsto X_n \end{array} \right]$$

i.e. $X = Y \dot{\vee} Z$.

In the double table of R we put the elements of $\otimes Y$ in the left, and the elements of $\otimes Z$ in the upper part:

		i_{k+1}
	$\otimes Z$	\vdots
		i_n
$\otimes Y$	boolean values of R	
$i_1 \dots i_k$		

17.6.2 Example double table

For example, let

$$X = \left[\begin{array}{c} \text{one} \mapsto \{1, 2\} \\ \text{two} \mapsto \{2, 3, 4\} \\ \text{three} \mapsto \{1, 2\} \\ \text{four} \mapsto \{2, 3, 4\} \end{array} \right]$$

A relation on X is

$$R = \left[\begin{array}{c} X \\ \left[\begin{array}{l} \text{one} \mapsto a \\ \text{two} \mapsto b \\ \text{three} \mapsto c \\ \text{four} \mapsto d \end{array} \right] \rightsquigarrow a \cdot b < c + d \end{array} \right]$$

i.e.

	one	two	three	four
$R =$	a	b	c	d
	$a \cdot b < c + d$			

and a schema split of X is given by

$$Y = \left[\begin{array}{l} \text{one} \mapsto \{1, 2\} \\ \text{two} \mapsto \{2, 3, 4\} \end{array} \right] \quad \text{and} \quad Z = \left[\begin{array}{l} \text{three} \mapsto \{1, 2\} \\ \text{four} \mapsto \{2, 3, 4\} \end{array} \right]$$

The boolean table of R has $2 \cdot 3 \cdot 2 \cdot 3 = 36$ rows (heading excluded) and is given by

one	two	three	four	
1	2	1	2	1
2	2	1	2	0
1	3	1	2	0
2	3	1	2	0
1	4	1	2	0
2	4	1	2	0
1	2	2	2	1
2	2	2	2	0
1	3	2	2	1
2	3	2	2	0
1	4	2	2	0
2	4	2	2	0
1	2	1	3	1
2	2	1	3	0
1	3	1	3	1
2	3	1	3	0
1	4	1	3	0
2	4	1	3	0
1	2	2	3	1
2	2	2	3	1
1	3	2	3	1
2	3	2	3	0
1	4	2	3	1
2	4	2	3	0
1	2	1	4	1
2	2	1	4	1
1	3	1	4	1
2	3	1	4	0
1	4	1	4	1
2	4	1	4	0
1	2	2	4	1
2	2	2	4	1
1	3	2	4	1
2	3	2	4	0
1	4	2	4	1
2	4	2	4	0

The double table of R with Y on the left and Z on top is more compact:

		1	2	1	2	1	2	three
		2	3	4	2	3	4	four
1	2	1	1	1	1	1	1	1
2	2	0	0	0	1	1	1	1
1	3	0	1	1	1	1	1	1
2	3	0	0	0	0	0	0	0
1	4	0	0	0	1	1	1	1
2	4	0	0	0	0	0	0	0
one	two							

17.7 More properties of relations

17.7.1 Definition measures

For every relation $R = [[X_i | i \in I], \Gamma]$ we define:

- (1) the dimension of R is the cardinality of the attributes (i.e. $\text{card}(I)$).
- (2) the cardinality of R is the cardinality of the graph (i.e. $\text{card}(\Gamma)$).

17.7.2 Remark

So a relation is a table iff both its dimension and its cardinality are finite.

For ordinary relations, the dimension is the arity of the relation.

17.7.3 Definition properties

A relation $R = [X, \Gamma]$ with $X = [X_i | i \in I]$ is called

- (1) empty, if $\Gamma = \emptyset$
- (2) full, if $\Gamma = \otimes X$
- (3) elementary, if Γ has exactly one element
- (4) literal, if both I and Γ are singletons.
- (5) singular, if X is singular (9.1.5), i.e. I is a singleton
- (6) an endorelation if the schema is univalent, i.e. all X_i are one and the same class.
- (7) (a) total in i , for $i \in I$, if $\text{rng}_i(R) = X_i$, and (b) total, if R is total in each $i \in I$.

17.7.4 Example

Consider the example relation D of 17.2.12 again,

$$D = \begin{array}{c|c} \mathbf{1 : 10} & \mathbf{2 : 10} \\ \hline 2 & 4 \\ \hline 2 & 6 \\ \hline 2 & 8 \\ \hline 3 & 6 \\ \hline 3 & 9 \\ \hline 4 & 8 \end{array}$$

D is neither total in 1 nor in 2, and thus not total altogether.

17.7.5 Remark

Note, that

- (1) a full relation is always total, but not the other way round. For example, the identity relation on the natural numbers $\textcircled{1} = \textcircled{2} : \mathbb{N} \leftrightarrow \mathbb{N}$ is an example of a total relation which is not full.

- (2) An improper relation (17.2.9 and 17.2.10) is always empty, but not the other way round.

17.7.6 Remark properties

The special relations defined in 17.7.3 have the following pattern:

endorelation $\begin{bmatrix} [C I] \\ \Gamma \end{bmatrix}$	empty relation $\begin{bmatrix} X \\ \emptyset \end{bmatrix}$	full relation $\begin{bmatrix} X \\ \otimes X \end{bmatrix}$
elementary relation $\begin{bmatrix} X \\ \{x\} \end{bmatrix}$	literal relation $\begin{bmatrix} [i \mapsto X_i] \\ \{[i \mapsto c]\} \end{bmatrix}$	singular relation $\begin{bmatrix} [i \mapsto X_i] \\ \otimes [i \mapsto C] \end{bmatrix}$

17.7.7 Remark

An *empty* relation has none, a *full* relation has all records as members. In the design of a relation algebra as a quasi-boolean algebra later on, empty and full relations become the least and greatest elements of these (quasi-)order structures. If we consider relation algebras as logical structures, empty relations represent “false” (they don’t hold for any record) and “true” (satisfied for all records).

Elementary, *literal* and *singular* relations will play the role of basic building bricks for relations. For example, every table can be (re-)constructed algebraically from a finite number of literal relations.

Endorelations, relations in which all domains are one and the same carrier class C , impose a kind of internal structure on C . *Binary endorelations*, i.e. relations of type $C \rightsquigarrow C$ play an important role in mathematics. Endorelations in general have a schema of the form $[C|I]$, where $I = \{1,2\}$ for the binary case. In relation algebras, the “compatibility” (19.1.2) of relations is an important constraint for many operations to be defined. But when all the relations involved are endorelations on the same C , they are always *compatible* (see 19.1.2 and 19.1.3).

17.8 Projection relation class

17.8.1 Definition

For every schema X we define

$$\mathbf{Prel}(X) := \bigcup_{Y \in \mathbf{Proj}(X)} \mathbf{Rel}(Y)$$

the projection relation class on X

17.8.2 Remark

For a given schema X , the class $\mathbf{Prel}(X)$ is a very important class of relations. Later on, we will see how each relation class $\mathbf{Prel}(X)$ comes along with a couple of operations that turn it into a *quasi-boolean* and a *theory algebra*, as we call it.

As the size of X increases, the number of relations in $\mathbf{Prel}(X)$

grows exponentially. In the next example, we take a very simple X , which allows us to list all these elements. The whole exercise is quite trivial, but it is worth studying, because it already reveals almost all the properties of these structures in general.

17.8.3 Example

A schema X is given by

$$X = \begin{bmatrix} a \mapsto \mathbb{B} \\ b \mapsto \mathbb{B} \end{bmatrix}$$

We obtain

$$\mathbf{Proj}(X) = \left\{ \langle \rangle, [a \mapsto \mathbb{B}], [b \mapsto \mathbb{B}], \begin{bmatrix} a \mapsto \mathbb{B} \\ b \mapsto \mathbb{B} \end{bmatrix} \right\}$$

For each of the four $Y \in \mathbf{Proj}(X)$ we obtain a class $\mathbf{Rel}(Y)$, which is given as follows.

- (1) $\mathbf{Rel}(\langle \rangle)$ has 2 members. As mentioned in 17.5.7, these can be represented as

$$\overline{0} \quad \text{and} \quad \overline{1}$$

respectively.

- (2) $\mathbf{Rel}([a \mapsto \mathbb{B}])$ has 4 members:

a	
0	0
1	0

a	
0	1
1	0

a	
0	0
1	1

a	
0	1
1	1

- (3) $\mathbf{Rel}([b \mapsto \mathbb{B}])$ has 4 members, too:

b	
0	0
1	0

b	
0	1
1	0

b	
0	0
1	1

b	
0	1
1	1

- (4) $\mathbf{Rel}\left(\begin{bmatrix} a \mapsto \mathbb{B} \\ b \mapsto \mathbb{B} \end{bmatrix}\right)$ has 16 members:

a	b	
0	0	0
1	0	0
0	1	0
1	1	0

a	b	
0	0	1
1	0	0
0	1	0
1	1	0

a	b	
0	0	0
1	0	1
0	1	0
1	1	0

a	b	
0	0	1
1	0	1
0	1	0
1	1	0

a	b	
0	0	0
1	0	0
0	1	1
1	1	0

a	b	
0	0	1
1	0	0
0	1	1
1	1	0

a	b	
0	0	0
1	0	1
0	1	1
1	1	0

a	b	
0	0	1
1	0	1
0	1	1
1	1	0

a	b	
0	0	0
1	0	0
0	1	0
1	1	1

a	b	
0	0	1
1	0	0
0	1	0
1	1	1

a	b	
0	0	0
1	0	1
0	1	0
1	1	1

a	b	
0	0	1
1	0	1
0	1	0
1	1	1

a	b	
0	0	0
1	0	0
0	1	1
1	1	1

a	b	
0	0	1
1	0	0
0	1	1
1	1	1

a	b	
0	0	0
1	0	1
0	1	1
1	1	1

a	b	
0	0	1
1	0	1
0	1	1
1	1	1

Finally, $\mathbf{Prel}(X)$ is the union of these four classes with $2 + 4 + 4 + 16 = 26$ members.

Such example X is an univalent bit-value schema, i.e. a schema of the form (9.2.4) $[\mathbb{B}|A]$, for some A , in this case $A = \{a, b\}$. The relation classes induced by these schemas are important and we introduce the following notation to denote them. For our example, that is $\mathbf{Prel}(X) = \mathbf{Brel}(\{a, b\})$.

17.8.4 Definition

For every class A we define

$$\mathbf{Brel}(A) := \mathbf{Prel}([\mathbb{B}|A])$$

the bit value relation class on A

17.8.5 Lemma

For every schema X , the following statements are equivalent:

- (1) X is completely finite.
- (2) Every $R \in \mathbf{Prel}(X)$ is completely finite.
- (3) $\mathbf{Prel}(X) = \mathbf{Tab}(X)$

17.8.6 Proof of 17.8.5

Let $X = [X_i | i \in I]$ be a given schema.

(1) \Leftrightarrow (2) is true, because

X is completely finite

$\Leftrightarrow (I \text{ is finite}) \text{ and } (X_i \text{ is finite, for each } i \in I)$

$\Leftrightarrow (\text{each } J \in \mathbf{P}(I) \text{ is finite}) \text{ and } (\text{each } X_i \text{ is finite})$

$\Leftrightarrow Y \text{ is completely finite, for every } Y \in \mathbf{Proj}(X)$

$\Leftrightarrow R \text{ is completely finite, for every } R \in \mathbf{Prel}(X)$

(3) \Leftrightarrow (1) is true, because

$$\mathbf{Prel}(X) = \mathbf{Tab}(X)$$

$$\Leftrightarrow \left\{ \begin{array}{l} [Y, \Gamma] \\ Y \in \mathbf{Proj}(X), \\ \Gamma \subseteq \otimes Y \end{array} \right\} = \left\{ \begin{array}{l} [Y, \Gamma] \\ Y \in \mathbf{Proj}(X), \\ \mathbf{dom}(Y) \text{ is finite,} \\ \Gamma \subseteq \otimes Y, \\ \Gamma \text{ is finite} \end{array} \right\}$$

$$\Leftrightarrow \forall Y \in \mathbf{Proj}(X) . \left(\begin{array}{l} \mathbf{dom}(Y) \text{ is finite and} \\ \forall \Gamma \subseteq \otimes Y . \Gamma \text{ is finite} \end{array} \right)$$

$$\Leftrightarrow \forall Y \in \mathbf{Proj}(X) . \left(\begin{array}{l} \mathbf{dom}(Y) \text{ is finite and} \\ \otimes Y \text{ is finite} \end{array} \right)$$

$\Leftrightarrow \forall Y \in \mathbf{Proj}(X) . (Y \text{ is completely finite})$

$\Leftrightarrow X \text{ is completely finite}$

18 Digression: null values, quasi-relations and expansions

18.0.7 Remark [introduction](#)

A “partial table” is very much the definition of a “relation” in database theory. Most *database management systems* nowadays use a (more or less) standard syntax and semantics, called *SQL* (structured query language). In this section, the notion “*database algebras*” refers to the underlying structure of these kind of systems.

This section is a “digression”, it introduces *null values*, *quasi-relations* and *partial tables*, and none of these concepts plays a role in the remainder of this text. In our interpretation they become an insignificant part of a more general design. This interpretation is very different to the approach in database algebras. So the statement that our “relation algebras” would generalize the mathematical as well as the database theoretical relation concept, is only true with some restrictions, at least when the SQL-like approach is taken.

18.1 Null values and quasi-relations

18.1.1 Remark [introduction](#)

Consider the following

$$Q_1 := \begin{array}{|c|c|c|} \hline \text{name : String} & \text{age : } \{0, 1, \dots, 255\} & \text{sex : } \{f, m\} \\ \hline \text{"peter"} & 57 & m \\ \hline \text{"mo"} & 40 & \\ \hline \text{"hanna"} & & f \\ \hline \end{array}$$

This is not a table or relation in the strict sense, because the data is incomplete in the last two record entries. Values for the attribute *sex* in the second, and *age* in the third row are missing. Database algebras usually use a **default** or **null value** “**null**” for these cases to fill the empty places.

The generalization of the relation concept that allows null values shall be called **quasi-relation**. In other words, for a quasi-relation Q with schema X , the graph Γ doesn't have to be a subclass of the cartesian product $\otimes X$, but may be a subclass of the bigger star product $\otimes X$. A table that allows such null values, like the one for Q_1 above, shall be called a **partial table**.

³⁴

18.1.2 Definition [quasi-relation](#)

As a generalization of the relation concept (17.2.1), a **quasi-relation** Q is given as a pair

$$[X, \Gamma] \quad \text{or} \quad \begin{array}{|c|} \hline X \\ \hline \Gamma \\ \hline \end{array}$$

where

- ♣ X is a schema
- ♣ $\Gamma \subseteq \otimes X$ is the so-called **quasi-graph** of Q .

Such a quasi-relation is a **partial table**, if both **dom**(X) and Γ are finite.

18.1.3 Example [introduction](#)

Given in this formal notation $[X, \Gamma]$ with schema X and $\Gamma \subseteq \otimes X$, our previous example Q_1 is given as

$$\left[\begin{array}{|c|} \hline \begin{array}{|c|} \hline \text{name} \mapsto \text{String} \\ \text{age} \mapsto \{0, \dots, 255\} \\ \text{sex} \mapsto \{f, m\} \\ \hline \end{array} \\ \hline \left\{ \left[\begin{array}{|c|} \hline \text{name} \mapsto \text{"peter"} \\ \text{age} \mapsto 57 \\ \text{sex} \mapsto m \\ \hline \end{array} \right], \left[\begin{array}{|c|} \hline \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \\ \hline \end{array} \right], \left[\begin{array}{|c|} \hline \text{name} \mapsto \text{"hanna"} \\ \text{sex} \mapsto f \\ \hline \end{array} \right] \right\} \\ \hline \end{array} \right]$$

18.1.4 Remark [two different meanings of null values](#)

There is a fundamental difference in the interpretation of null values in database algebras and in our approach.

- ♣ In database algebras, null values are a kind of default value of each domain component. A **null** stands for something like “unknown”. In order to consequently fit this into the overall design, many operations need to be extended for these exceptional cases. For example, the binary logic of **false** and **true** has to be accompanied by a third value **unknown** to deal with **null** occurrences.
- ♣ In our approach, a null value is some kind of placeholder or abbreviation for “all possible values”.

Consider the initial example Q_1 again and the following record

$$y := \begin{array}{|c|} \hline \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 0 \\ \text{sex} \mapsto f \\ \hline \end{array}$$

Is y a member of Q_1 ?

- ♣ In our approach, this is a clear “yes”: $y \in Q_1$ does hold. The question is understood as: “Could it be compatible with the information we have, that y does exist?”
- ♣ In database algebras, the answer is “unknown”: neither $y \in Q_1$ nor $y \notin Q_1$ does hold. The question is here understood as: “Does y exist in our data world Q_1 ?”

³⁴ Neither “quasi-relation” nor “partial table” is part of the standard terminology in database theory.

Our algebra is an attempt to generalize the binary semantics of propositional logic toward a multivalued semantics. At the same time we want to introduce an intuitive and efficient semantics for both propositional and predicate logic. Therefore our algebra of relations needs to have at least the properties of a boolean lattice. Database algebras do not provide these properties.

18.1.5 Remark quasi-relations are relations

Quasi-relations are “as-good-as relations”, every quasi-relation Q is actually supposed to stand for a proper relation $\mathbf{rel}(Q)$. From this point of view, quasi-relations are just another representation for relations. Accordingly, this whole section 18 on quasi-relations could be skipped for a more conceptual and less technical study. However, a familiarity with the arguments of the following discussion are very helpful for an overall understanding of the operations introduced later on. So in 18.3 we motivate the important concept of the *expansion* $R \parallel Y$ of a relation R by a compatible schema Y .

18.2 Quasi-relations as relations

18.2.1 Remark introduction

We will now see how a quasi-relation Q is converted into a proper relation $\mathbf{rel}(Q)$. The basic idea is to translate each incomplete record $x \in \otimes X$ into the class $x \parallel X \subseteq \otimes X$ of all full records which are compatible with x . $\mathbf{rel}(Q)$ is then essentially the union of these classes.

18.2.2 Example quasi-relation as relation

Consider the example Q_1 from 18.1.1 again. Let us start with the second record

$$x_2 = \begin{bmatrix} \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \end{bmatrix}$$

in Q_1 . The schema of Q_1 is

$$X = \begin{bmatrix} \text{name} \mapsto \mathbf{String} \\ \text{age} \mapsto \{0, \dots, 255\} \\ \text{sex} \mapsto \{\mathbf{f}, \mathbf{m}\} \end{bmatrix}$$

X has one index (namely “sex”) which doesn’t occur in x_2 . Since there is no clue which of the two sexes should belong to x_2 , we define x_2 to stand for both alternatives

$$\begin{bmatrix} \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \\ \text{sex} \mapsto \mathbf{f} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \\ \text{sex} \mapsto \mathbf{m} \end{bmatrix}$$

one for “sex=f” and one for “sex=m”. In other words, the table of the proper relation $\mathbf{rel}(Q_1)$ shall contain the two rows

"mo"	40	f
"mo"	40	m

We call this record class the expansion of x_1 in X and write

$$x_1 \parallel X := \left\{ \begin{bmatrix} \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \\ \text{sex} \mapsto \mathbf{f} \end{bmatrix}, \begin{bmatrix} \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \\ \text{sex} \mapsto \mathbf{m} \end{bmatrix} \right\}$$

Putting this method into a formula (see 18.2.4), we generate for every schema X and $x \in \otimes$ the expansion of x into X by applying

$$x \parallel X := \{x \dot{\vee} y \mid y \in \otimes \mathbf{pr}(X, \mathbf{dom}(X) \setminus \mathbf{dom}(x))\}$$

In case of our specific example X and $x_2 \in \otimes X$, we obtain

$$\begin{aligned} & \otimes \mathbf{pr}(X, \mathbf{dom}(X) \setminus \mathbf{dom}(x_2)) \\ &= \otimes \mathbf{pr}(X, \{\text{name}, \text{age}, \text{sex}\} \setminus \{\text{name}, \text{age}\}) \\ &= \otimes \mathbf{pr}(X, \{\text{sex}\}) \\ &= \otimes [\text{sex} \mapsto \{\mathbf{f}, \mathbf{m}\}] \\ &= \left\{ [\text{sex} \mapsto \mathbf{f}], [\text{sex} \mapsto \mathbf{m}] \right\} \end{aligned}$$

so that

$$\begin{aligned} x_2 \parallel X &= \left\{ x_2 \dot{\vee} [\text{sex} \mapsto \mathbf{f}], x_2 \dot{\vee} [\text{sex} \mapsto \mathbf{m}] \right\} \\ &= \left\{ \begin{bmatrix} \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \\ \text{sex} \mapsto \mathbf{f} \end{bmatrix}, \begin{bmatrix} \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \\ \text{sex} \mapsto \mathbf{m} \end{bmatrix} \right\} \end{aligned}$$

This definition of the expansion is more constructive. Perhaps more elegant is the characterization by means of the compatibility notion:

$$x_2 \parallel X = \{y \in \otimes X \mid y \smile x_2\}$$

The only $y \in \otimes X$ which are compatible with x_2 are

$$\begin{bmatrix} \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \\ \text{sex} \mapsto \mathbf{f} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \text{name} \mapsto \text{"mo"} \\ \text{age} \mapsto 40 \\ \text{sex} \mapsto \mathbf{m} \end{bmatrix}$$

So this characterization produces indeed the same result. This coincidence is true in general (and stated in 18.2.3 and 18.2.4).

So far for x_2 . Let us do the same for x_1 and x_3 :

$$x_3 = \begin{bmatrix} \text{name} \mapsto \text{"hanna"} \\ \text{sex} \mapsto \mathbf{f} \end{bmatrix}$$

stands for 256 records with attribute “age” ranging from 0 to 255. More formal

$$x_3 \parallel X = \left\{ \begin{bmatrix} \text{name} \mapsto \text{"hanna"} \\ \text{age} \mapsto 0 \\ \text{sex} \mapsto \mathbf{f} \end{bmatrix}, \dots, \begin{bmatrix} \text{name} \mapsto \text{"hanna"} \\ \text{age} \mapsto 255 \\ \text{sex} \mapsto \mathbf{f} \end{bmatrix} \right\}$$

The first record x_1 of Q_1 has all the attributes of X , its expansion is simply

$$x_1 \parallel X = \left\{ \begin{bmatrix} \text{name} \mapsto \text{"peter"} \\ \text{age} \mapsto 57 \\ \text{sex} \mapsto \mathbf{m} \end{bmatrix} \right\}$$

The proper relation $\mathbf{rel}(Q_1)$ of Q now is basically the union of all these record expansions

$$\mathbf{rel}(Q_1) := \begin{bmatrix} X \\ (x_1 \parallel X) \cup (x_2 \parallel X) \cup (x_3 \parallel X) \end{bmatrix}$$

It is given by the table

name : String	age : {0, ..., 255}	sex : {f, m}
"peter"	57	m
"mo"	40	f
"mo"	40	m
"hanna"	0	f
"hanna"	1	f
"hanna"	2	f
⋮	⋮	⋮
"hanna"	255	f

But here, too, we can apply the same result by

$$\text{rel}(Q_1) := \left[\begin{array}{c} X \\ y \rightsquigarrow y \smile x_1 \text{ or } y \smile x_2 \text{ or } y \smile x_2 \end{array} \right]$$

in other words

$$\text{rel}(Q_1) := \left[\begin{array}{c} X \\ y \rightsquigarrow \exists x \in X . y \smile x \end{array} \right]$$

We finally summarize the definitions properly.

18.2.3 Definition record expansion

Let $X = [X_i | i \in I]$ be a schema and $x \in \otimes X$. We define

$$x \parallel X := \{y \in \otimes X \mid y \smile x\}$$

the expansion of x into X .

18.2.4 Lemma record expansion

For every schema X and $x \in \otimes X$ holds

$$x \parallel X = \{x \dot{\vee} y \mid y \in \otimes \text{pr}(X, \text{dom}(X) \setminus \text{dom}(x))\}$$

18.2.5 Proof of 18.2.4

Let $X = [X_i | i \in I]$ be the given schema and $x = [x_j | j \in J] \in \otimes X$. For every $y = [y_i | i \in I] \in \otimes X$ holds $J \subseteq I$ and

$$y \smile x \Leftrightarrow x \dot{\vee} \text{pr}(y, I \setminus J) = y$$

Therefore

$$\begin{aligned} x \parallel X &= \{y \in \otimes X \mid y \smile x\} \\ &= \{x \dot{\vee} y \mid y \in \otimes \text{pr}(X, I \setminus J)\} \end{aligned}$$

18.2.6 Definition quasi-relation as relation

For every quasi-relation $Q = [X, \Gamma]$ we define

$$\text{rel}(Q) := \left[\begin{array}{c} X \\ y \rightsquigarrow \exists x \in \Gamma . y \smile x \end{array} \right]$$

the relation of Q .

18.2.7 Lemma quasi-relation as relation

For every quasi-relation $Q = [X, \Gamma]$ holds

$$\text{rel}(Q) = \left[\begin{array}{c} X \\ \bigcup_{x \in \Gamma} (x \parallel X) \end{array} \right]$$

18.2.8 Proof of 18.2.7

For each quasi-relation $Q = [X, \Gamma]$ we have

$$\begin{aligned} \text{rel}(Q) &= \left[\begin{array}{c} X \\ y \rightsquigarrow \exists \in \Gamma . y \smile x \end{array} \right] \\ &= \left[\begin{array}{c} X \\ \{y \in \otimes X \mid \exists x \in \Gamma . y \smile x\} \end{array} \right] \\ &= \left[\begin{array}{c} X \\ \bigcup_{x \in \Gamma} \{x \in \otimes X \mid y \smile x\} \end{array} \right] \\ &= \left[\begin{array}{c} X \\ \bigcup_{x \in \Gamma} (x \parallel X) \end{array} \right] \end{aligned}$$

18.3 Schema expansion of relations

18.3.1 Remark introduction

One very important operation in our upcoming algebra is the expansion $R \parallel Y$ of a given relation $R = [X, \Gamma]$ by a compatible (i.e. $X \smile Y$) schema Y . The schema of the result R' is then the join $X \vee Y$.

18.3.2 Example expansion

Let us take the example

$$R := \begin{array}{|l|l|} \hline \text{name : String} & \text{age : } \{0, \dots, 255\} \\ \hline \text{"peter"} & 57 \\ \hline \text{"mo"} & 40 \\ \hline \end{array}$$

with schema

$$X := \mathbf{x}(R) = \left[\begin{array}{l} \text{name} \mapsto \text{String} \\ \text{age} \mapsto \{0, \dots, 255\} \end{array} \right]$$

A second schema is given by

$$Y := \left[\text{sex} \mapsto \{\text{f}, \text{m}\} \right]$$

X and Y are compatible and their join is

$$X' := X \vee Y = \left[\begin{array}{l} \text{name} \mapsto \text{String} \\ \text{age} \mapsto \{0, \dots, 255\} \\ \text{sex} \mapsto \{\text{f}, \text{m}\} \end{array} \right]$$

The expansion of R by Y is a relation R' with the modified schema X' , but “equivalent” to R . And this means, that R' has the same graph as R , i.e.

$$R' = \begin{array}{|c|c|c|} \hline \text{name : String} & \text{age : } \{0, \dots, 255\} & \text{sex : } \{\text{f}, \text{m}\} \\ \hline \text{"peter"} & 57 & \\ \hline \text{"mo"} & 40 & \\ \hline \end{array}$$

But this is a quasi-relation and not a proper relation yet. Applying **rel** returns the standard form of R' , which is, as a table diagram, given by

$$R' = \begin{array}{|c|c|c|} \hline \text{name : String} & \text{age : } \{0, \dots, 255\} & \text{sex : } \{\text{f}, \text{m}\} \\ \hline \text{"peter"} & 57 & \text{f} \\ \hline \text{"peter"} & 57 & \text{m} \\ \hline \text{"mo"} & 40 & \text{f} \\ \hline \text{"mo"} & 40 & \text{m} \\ \hline \end{array}$$

18.3.3 Definition relation expansion

Let $R = [X, \Gamma]$ be a relation and Y a schema with $Y \smile X$. Then

$$R \parallel Y := \mathbf{rel} \left(\left[\begin{array}{l} X \vee Y \\ \Gamma \end{array} \right] \right)$$

is the expansion of R by Y .

18.3.4 Remark

An alternative characterization of this expansion is given by the fact that

$$R \parallel Y = \left[\begin{array}{l} X \vee Y \\ y \rightsquigarrow \exists x \in \Gamma . x \smile y \end{array} \right]$$

Actually, we prefer the latter equation as the definition (see 20.1.1) because it doesn't need to refer to quasi-relations and the **rel** operation.

18.3.5 Remark schema unification

This expansion concept is very important for the development of our whole algebra on relations later on.

Take, for example, two relations $R = [X, \Gamma]$ and $S = [Y, \Sigma]$. Lattice operations like \cup and \cap are very important and they are understood as the according operations on the graphs, e.g.

$$R \cup S := \left[\begin{array}{l} X \\ \Gamma \cup \Sigma \end{array} \right]$$

But of course, this is only a well-defined relation, if R and S are equischematic, i.e. if $Y = X$.

However, by applying the previous ideas, we can generalize \cup and \cap for the case that X and Y are compatible. We denote these generalizations by \sqcup and \sqcap , respectively. So

$$R \sqcup S := \mathbf{rel} \left(\left[\begin{array}{l} X \vee Y \\ \Gamma \cup \Sigma \end{array} \right] \right)$$

Without applying **rel** this can alternatively be expressed by

$$R \sqcup S := R' \cup S' \quad \text{where} \quad R' := R \parallel Y \quad \text{and} \quad S' := S \parallel X$$

R' and S' are equischematic, their common schema is $X \vee Y$, and that allows us to apply \cup .

Besides, this is also the basis for a definition of the equivalence of relations:

$$R \equiv S \quad \text{iff} \quad R' = S'$$

This recipe:

- ♣ first, make the relations equischematic, and
- ♣ second, apply a class operation on the new graphs

is a general method in the algebra we are going to develop below.

18.3.6 Remark

The schema unification of relations is similar to the denominator unification of two fractions: To compare or add two fractions

$$\alpha = \frac{n}{d} \quad \text{and} \quad \beta = \frac{m}{e}$$

we first produce

$$\alpha' := \frac{n \cdot e}{d \cdot e} \quad \text{and} \quad \beta' = \frac{m \cdot d}{d \cdot e}$$

and then we are able to compare:

$$\alpha \leq \beta \quad \text{iff} \quad n \cdot e \leq m \cdot d$$

and to add:

$$\alpha + \beta = \frac{(n \cdot e) + (m \cdot d)}{d \cdot e}$$

The general idea is, that α' and β' have the same denominator $d \cdot e$. Similarly, R' and S' have the same schema $X \vee Y$.

For the fraction arithmetic, it is important that $\alpha' = \alpha$ and $\beta' = \beta$. And the same is true for the relation algebra: $R' \equiv R$ and $S' \equiv S$. The result of an expansion is always equivalent to the original relation.

Part VIII

Operations on relations

19 Some basic operations on relations

19.1 Schema comparison

19.1.1 Repetition

Recall (definition 10.3.2), that for two schemas $X = [X_i | i \in I]$ and $Y = [Y_j | j \in J]$

- (1) $X \not\sim Y$ iff $I \cap J = \emptyset$ (distinct)
- (2) $X \sim Y$ iff $X_k = Y_k$ for all $k \in I \cap J$ (compatible)
- (3) $X \leq Y$ iff $I \subseteq J$ and $X_i = Y_i$ for all $i \in I$ (smaller)

19.1.2 Definition

Let $X = [X_i | i \in I]$ and $Y = [Y_j | j \in J]$ be two schemas and $R = [X, \Gamma]$ and $S = [Y, \Sigma]$ two relations. We define:

- (1) R is sub-schematic to S , written

$$R \trianglelefteq S$$

iff $X \leq Y$ (i.e. $I \subseteq J$ and $X_i = Y_i$ for all $i \in I$)

- (2) R is equi-schematic with S , written

$$R \triangleq S$$

iff $X = Y$

- (3) R and S are (schema) compatible, written

$$R \sim S$$

iff $X \sim Y$ (i.e. $X_k = Y_k$ for all $k \in I \cap J$)

- (4) R and S are (attribute or schema) distinct, written

$$R \not\sim S$$

iff $X \not\sim Y$ (i.e. $I \cap J = \emptyset$)

And again, we write

$$R \not\trianglelefteq S \quad R \not\triangleq S \quad R \not\sim S \quad R \not\not\sim S$$

if the according relation does not hold.

19.1.3 Definition

A class \mathcal{R} of relations is called

- (1) (pairwise) distinct, if
all $R, S \in \mathcal{R}$ with $R \neq S$ are distinct
- (2) (pairwise) compatible, if
all $R, S \in \mathcal{R}$ are compatible
- (3) (pairwise) equi-schematic, if
all $R, S \in \mathcal{R}$ are equi-schematic

19.1.4 Example

The just defined relations between relations and properties of relation classes all depend on the schemas only, the graphs are not concerned.

Let us take the following example relations in graph table notation:

$$R_1 = \begin{array}{|c|c|} \hline a : \mathbb{N} & b : \mathbb{Z} \\ \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} \quad R_2 = \begin{array}{|c|c|} \hline a : \mathbb{N} & b : \mathbb{Z} \\ \hline 1 & 3 \\ \hline \end{array} \quad R_3 = \begin{array}{|c|c|} \hline b : \mathbb{Z} & c : \mathbb{N} \\ \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array}$$

$$R_4 = \begin{array}{|c|c|} \hline a : \mathbb{Z} & c : \mathbb{N} \\ \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} \quad R_5 = \begin{array}{|c|c|} \hline p : \mathbb{N} & q : \mathbb{Z} \\ \hline 2 & 3 \\ \hline 4 & 5 \\ \hline \end{array} \quad R_6 = \begin{array}{|c|c|} \hline r : \mathbb{N} & s : \mathbb{Z} \\ \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array}$$

- (1) R_1 and R_2 are equi-schematic, $R_1 \triangleq R_2$.

This example explains the introduction of \triangleq . For any two schemas X and Y , $X \leq Y$ and $Y \leq X$ implies $X = Y$. For relations, this is not true in general: $R_1 \trianglelefteq R_2$ and $R_2 \trianglelefteq R_1$ does not necessarily imply $R_1 = R_2$.

In more general terms: **(REC, \leq)** is a *poclass* (i.e. \leq is transitive, reflexive and antisymmetric on the class of all records, including schemas), but **(Rel, \trianglelefteq)** is only a *quasi-ordered class* (i.e. \trianglelefteq is transitive and reflexive, but not necessarily antisymmetric on the class of all relations).

- (2) $R_1 \sim R_3$ and $R_3 \sim R_4$, but $R_1 \not\sim R_4$ (because $\text{dom}_a(R_1) = \mathbb{N} \neq \mathbb{Z} = \text{dom}_a(R_4)$)

This example shows, that \sim is not transitive in general. So it is not an equivalence relation, although it is reflexive and symmetric.

- (3) R_1 and R_5 are distinct, i.e. $R_1 \not\sim R_5$, because their attribute classes $\{a, b\}$ and $\{p, q\}$ are disjoint.

R_1 and R_5 are also compatible, i.e. $R_1 \sim R_5$, because $\text{dom}_i(R_1) \cap \text{dom}_i(R_5) = \emptyset$ and $\text{dom}_i(R_1) = \text{dom}_i(R_5)$ is true for all $i \in \emptyset$ in a trivial sense.

Obviously, this example is symptomatic for the general rule: distinctness implies compatibility.

- (4) $\{R_1, R_5, R_6\}$ is distinct. Their attribute classes $\{a, b\}$, $\{p, q\}$, $\{r, s\}$ are pairwise disjoint and thus $R_1 \not\sim R_5$, $R_1 \not\sim R_6$, and $R_5 \not\sim R_6$.
 $\{R_1, R_5, R_6\}$ is also compatible.

19.1.5 Example

- (1) The empty class \emptyset is a distinct, compatible, and equi-schematic class of relations.
- (2) \mathbf{Rel} , the class of all relations is not distinct, not compatible, and not equischematic.

19.1.6 Lemma

- (1) $\langle \mathbf{Rel}, \trianglelefteq \rangle$ is a quasi-ordered class.
- (2) $\langle \mathcal{R}, \trianglelefteq \rangle$ is a quasi-ordered class, for each $\mathcal{R} \subseteq \mathbf{Rel}$.
- (3) \trianglelefteq is the derived equivalence relation of \trianglelefteq in the sense that $R \trianglelefteq S$ iff $R \trianglelefteq S$ and $S \trianglelefteq R$, for all relations R, S .

19.1.7 Proof of 19.1.6

Recall, that a *quasi-order relation* is a transitive and reflexive binary endorelation.

- (1) Let $R_1 = [X_1, \Gamma_1], R_2 = [X_2, \Gamma_2], R_3 = [X_3, \Gamma_3]$ be any three relations. If $R_1 \trianglelefteq R_2$ and $R_2 \trianglelefteq R_3$, then $X_1 \leq X_2$ and $X_2 \leq X_3$, so that $X_1 \leq X_3$ (according to 11.1.1) and thus $R_1 \trianglelefteq R_3$. This is the transitivity of \leq on \mathbf{REC} . And with $X_1 \leq X_1$ we have $R_1 \trianglelefteq R_1$, which is the reflexivity.
- (2) The same reasoning in (1) holds, when we restrict the three relations to come from \mathcal{R} . More general: Every subclass of a quasi-ordered class is a quasi-ordered class itself, with respect to the same relation (see 7.2.3).
- (3) On the class of schemas, \leq is antisymmetric, i.e. $X \leq Y$ and $Y \leq X$ implies $X = Y$. So two relations $R = [X, \Gamma]$ and $S = [Y, \Sigma]$ with $R \trianglelefteq S$ and $S \trianglelefteq R$ have $X \leq Y$ and $Y \leq X$, so $X = Y$ and thus $R \trianglelefteq S$.

19.1.8 Lemma

- (1) Every distinct class of relations is compatible.
- (2) Every equischematic class of relations is compatible.

19.1.9 Proof of 19.1.8

Both statements are immediate consequences of the definitions in 19.1.2.

19.1.10 Lemma

For every schema X holds:

- (1) $\mathbf{Rel}(X)$ is equi-schematic and compatible
 - (2) $\mathbf{Prel}(X)$ is compatible
- On the other hand, if \mathcal{R} is a relation class, then
- (3) \mathcal{R} is equi-schematic iff $\mathcal{R} \subseteq \mathbf{Rel}(Y)$, for some schema Y
 - (4) \mathcal{R} is compatible iff $\mathcal{R} \subseteq \mathbf{Prel}(Y)$, for some schema Y
- Furthermore
- (5) If \mathcal{R} is compatible, then $\mathcal{R} \subseteq \mathbf{Prel}(\{\mathbf{x}(R) \mid R \in \mathcal{R}\})$.

19.1.11 Proof of 19.1.10

Let $X = [X_i \mid i \in I]$ be any given schema.

- (1) $\mathbf{Rel}(X) = \{[X, \Gamma] \mid \Gamma \subseteq \otimes X\}$ is equi-schematic by definition, each member has the same schema X . Therefore it is compatible as well.
- (2) We have $\mathbf{Prel}(X) = \{[Y, \Gamma] \mid Y \in \mathbf{Proj}(X), \Gamma \subseteq \otimes Y\}$

and we know (from 10.3.10), that $\mathbf{Proj}(X)$ is compatible. So all $Y_1, Y_2 \in \mathbf{Proj}(X)$ are compatible and that makes all $[Y_1, \Gamma_1], [Y_2, \Gamma_2] \in \mathbf{Prel}(X)$ compatible as well.

- (3) If $\mathcal{R} = \emptyset$, then $\mathcal{R} \subseteq \mathbf{Rel}(Y)$ for every arbitrary schema Y . If $\mathcal{R} \neq \emptyset$ and $[X, \Gamma]$ is a member of \mathcal{R} , then every other member must have the same schema X , because \mathcal{R} is supposed to be equi-schematic. So $R \in \mathbf{Rel}(X)$ for all $R \in \mathcal{R}$, i.e. $\mathcal{R} \subseteq \mathbf{Rel}(X)$.

From lemma 11.7.6(3) we know that for every class \mathcal{X} of schemas, the following three statements are equivalent: (a) \mathcal{X} is compatible. (b) $\mathcal{X} \subseteq \mathbf{Proj}(Y)$, for some schema Y . (c) $\mathcal{X} \subseteq \mathbf{Proj}(X)$ for $X := \bigvee \mathcal{X}$.

We can apply that knowledge immediately for the compatibility of a relation class \mathcal{R} , because that only depends on the schemas of the elements in \mathcal{R} . As a result we obtain (4) and (5).

19.2 Identity relations**19.2.1** Definition identity

For every schema $X = [X_i \mid i \in I]$ we define

$$\mathbf{Id}_X := \left[\begin{array}{c} X \\ x \rightsquigarrow x(i) = x(j) \text{ for all } i, j \in I \end{array} \right]$$

the identity (relation) of X .

19.2.2 Remark

If C is a class, the usual identity $=_C$ on C is just a special case. We can (re)define

$$\textcircled{1} =_C \textcircled{2} := \mathbf{Id}_{\langle C, C \rangle} = \left[\begin{array}{c} C \rightsquigarrow C \\ \{\langle x, x \rangle \mid x \in C\} \end{array} \right]$$

19.2.3 Table representations

Recall, that $\mathbb{N} := \{0, 1, 2, \dots\}$ is the natural number and \mathbb{Q} is the rational number class. Let $A := \{-2, 0, 3, 7, 11\}$ and

$$X := \left[\begin{array}{l} a \mapsto A \\ n \mapsto \mathbb{N} \\ q \mapsto \mathbb{Q} \end{array} \right]$$

The graph of \mathbf{Id}_X contains exactly the records

$$x := \left[\begin{array}{l} a \mapsto \nu \\ n \mapsto \nu \\ q \mapsto \nu \end{array} \right] \quad \text{with } \nu \in A \cap \mathbb{N} \cap \mathbb{Q}$$

Accordingly, the *graph table* contains all the possible rows with

identical components:

$a : A$	$n : \mathbb{N}$	$q : \mathbb{Q}$
0	0	0
3	3	3
7	7	7
11	11	11

An often used special case in mathematics is the *double table* of $=_C$, for some class C . For example, the double table for $=_A$ is

	-2	0	3	7	11
-2	1	0	0	0	0
0	0	1	0	0	0
3	0	0	1	0	0
7	0	0	0	1	0
11	0	0	0	0	1

(Note, that this standard table in mathematics is a little variation of our precise definition of a double table. We left away the attributes in 1 and 2 for the left column and top row. In fact, it doesn't matter where they are place, because the identity is symmetric.) The 1's form a *diagonal* shape in the table. Sometimes, the identity relation itself is therefore called the *diagonal relation*.

19.2.4 Example identity

Recall, that $\mathbb{N} := \{0, 1, 2, \dots\}$ and $\mathbb{Z} := \{\dots, -1, 0, 1, \dots\}$. So

$$\mathbf{Id}_{(\mathbb{N}, \mathbb{Z})} = \begin{bmatrix} \mathbb{N} \rightsquigarrow \mathbb{Z} \\ \langle n, m \rangle \rightsquigarrow n = m \end{bmatrix}$$

And the ternary identity on $\mathbb{B} = \{0, 1\}$ is

$$\mathbf{Id}_{(\mathbb{B}, \mathbb{B}, \mathbb{B})} = \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

19.3 Empty and full relations

19.3.1 Definition empty and full relations

Let $X = [X_i | i \in I]$ be a schema, then

$$\perp_X := \begin{bmatrix} X \\ \emptyset \end{bmatrix}$$

is the empty, bottom or zero relation on X

$$\top_X := \begin{bmatrix} X \\ \otimes X \end{bmatrix}$$

is the full, top, or unit relation on X

In particular

$$\perp := \perp_{\langle \rangle} = \begin{bmatrix} \langle \rangle \\ \emptyset \end{bmatrix}$$

is the empty, bottom, or zero relation

$$\top := \top_{\langle \rangle} = \begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \end{bmatrix}$$

is the full, top, or unit relation

19.3.2 Table representations

$$\text{Let } X = \begin{bmatrix} a \mapsto \{p, q\} \\ b \mapsto \{r, s, t\} \end{bmatrix}.$$

The graph tables of \perp_X and \top_X are given by

$$\perp_X = \begin{array}{|c|c|} \hline a : \{p, q\} & b : \{r, s, t\} \\ \hline \end{array}$$

and

$$\top_X = \begin{array}{|c|c|} \hline a : \{p, q\} & b : \{r, s, t\} \\ \hline p & r \\ \hline q & r \\ \hline p & s \\ \hline q & s \\ \hline p & t \\ \hline q & t \\ \hline \end{array}$$

i.e. in \perp_X there is none, in \top_X every element of $\otimes X$ is in the table, \perp_X is *the least* and \top_X is *the greatest* relation on X .

Their boolean tables are typical as well:

$$\perp_X = \begin{array}{|c|c|} \hline a & b \\ \hline p & r \\ \hline q & r \\ \hline p & s \\ \hline q & s \\ \hline p & t \\ \hline q & t \\ \hline \end{array} \quad \text{and} \quad \top_X = \begin{array}{|c|c|} \hline a & b \\ \hline p & r \\ \hline q & r \\ \hline p & s \\ \hline q & s \\ \hline p & t \\ \hline q & t \\ \hline \end{array}$$

i.e. \perp_X shows 0's and \top_X has 1's only.

19.3.3 Table representations of \perp and \top

The empty cartesian product $\otimes\langle\rangle = \{\langle\rangle\}$ has an empty set of attributes. If we write \perp and \top in graph table notation, we have the problem that the table “disappears”.

Written as boolean tables, we suggest the following notation for \perp and \top :

$$\boxed{\mathbf{0}} = \perp \quad \text{and} \quad \boxed{\mathbf{1}} = \top$$

according to the construction rules for boolean tables.

19.3.4 Remark

In 17.7.3 we defined to call a relation $R = [X, \Gamma]$ **empty**, if $\Gamma = \emptyset$, and **full**, if $\Gamma = \otimes\Gamma$. In terms of our new notation, this can be expressed by: R is empty, if $R = \perp_X$, and full, if $R = \top_X$.

19.3.5 Remark

If $X = [X_i | i \in I]$ is a schema, then exactly one of the three following statements is true:

- (1) X is empty, i.e. $X = \langle\rangle$.
Then $\perp_X = \perp = [\langle\rangle, \emptyset] \neq [\langle\rangle, \{\langle\rangle\}] = \top = \top_X$ and $\mathbf{Rel}(X) = \{\perp, \top\}$.
- (2) X is not proper, i.e. $X_i = \emptyset$ for at least one $i \in I$.
Then $\otimes X = \emptyset$ and $\perp_X = [X, \emptyset] = [X, \otimes X] = \top_X$ and $\mathbf{Rel}(X) = \{\perp_X\} = \{\top_X\}$.
- (3) X is proper and not empty.
Then $\otimes X \neq \emptyset$, $\perp_X \neq \top_X$, and $\mathbf{Rel}(X) \supseteq \{\perp_X, \top_X\}$.

We obtain the following result:

- (4) $\perp_X \neq \top_X$ iff X is proper

19.4 Complement

19.4.1 Definition complement

For every relation $R = [X, \Gamma]$, we define

$$\neg R := \begin{bmatrix} X \\ \otimes X \setminus \Gamma \end{bmatrix}$$

the **complement** or **negation** of R or “**not** R ”.

The symbolic “deletion” of R , i.e.

$$\boxed{R}$$

is an often used alternative notation for $\neg R$.

19.4.2 Boolean table representations

Recall 5.8.4, that for each $\beta \in \mathbb{B} := \{0, 1\}$ we defined:

$$\neg\beta := \begin{cases} 0 & \text{if } \beta = 1 \\ 1 & \text{if } \beta = 0 \end{cases}$$

If a relation R is completely finite and its boolean table has the form

i_1	\dots	i_n	
\dots	\dots	\dots	β_1
\dots	\dots	\dots	β_2
\dots	\dots	\dots	β_3
\vdots		\vdots	\vdots
\vdots		\vdots	\vdots
\dots	\dots	\dots	β_m

then $\neg R$ is given by

i_1	\dots	i_n	
\dots	\dots	\dots	$\neg\beta_1$
\dots	\dots	\dots	$\neg\beta_2$
\dots	\dots	\dots	$\neg\beta_3$
\vdots		\vdots	\vdots
\vdots		\vdots	\vdots
\dots	\dots	\dots	$\neg\beta_m$

19.4.3 Example

If $R =$

a	b	
p	r	$\mathbf{0}$
q	r	$\mathbf{1}$
p	s	$\mathbf{0}$
q	t	$\mathbf{0}$
p	t	$\mathbf{1}$
q	s	$\mathbf{1}$

then $\neg R =$

a	b	
p	r	$\mathbf{1}$
q	r	$\mathbf{0}$
p	s	$\mathbf{1}$
q	t	$\mathbf{1}$
p	t	$\mathbf{0}$
q	s	$\mathbf{0}$

19.4.4 Remark

If $R = [X, \Gamma]$ is a table, then $\neg R = [X, \otimes X \setminus \Gamma]$ is not necessarily a table anymore, because the new graph $\otimes X \setminus \Gamma$ might be infinite. So $\neg R$ might not be representable by a graph table, even if R is.

19.4.5 Remark

For every $R \in \mathbf{Rel}(X)$ and each $x \in \otimes X$ holds: $\neg R(x)$ iff $R(x)$ is not true. In other words, $x \in \neg R$ iff $x \notin R$.
The deletion notation is more convenient and common for binary endorelations. For example, if \leq is the usual linear order (say on the integers), we usually write $\not\leq$ rather than $\neg \leq$. So for every two integers n and m we have $n \not\leq m$ iff $n \leq m$ is not true.
For a linear order like the \leq on \mathbb{Z} , there is also the option to use $>$ for $\not\leq$. But note the difference between the *conversion* (or *inversion*) of a binary relation, where $R : X \rightsquigarrow Y$ turns into

$$\left[\begin{array}{c} Y \rightsquigarrow X \\ \langle y, x \rangle \rightsquigarrow \langle x, y \rangle \in R \end{array} \right]$$

and the complement. Complementation (or negation) and conversion (or inversion) are different operations. Besides, this turn of the symbol direction doesn't work for non-linear relations. Take the class inclusion \subseteq . Its complement $\not\subseteq$ is different to \supset . For example, $\{1, 2\} \not\subseteq \{2, 3\}$, but $\{1, 2\} \supset \{2, 3\}$ is not correct.

19.4.6 Lemma

$$\neg\neg R = R \quad \text{for every relation } R.$$

19.4.7 Proof of 19.4.6

If $R = [X, \Gamma]$, then

$$\neg\neg R = \neg \begin{bmatrix} X \\ \otimes X \setminus \Gamma \end{bmatrix} = \begin{bmatrix} X \\ \otimes X \setminus (\otimes X \setminus \Gamma) \end{bmatrix} = \begin{bmatrix} X \\ \Gamma \end{bmatrix} = R$$

19.4.8 Lemma

$$\begin{aligned} (1) \quad \neg \perp &= \top \\ (2) \quad \neg \top &= \perp \end{aligned}$$

19.4.9 Proof of 19.4.8

$$(1) \quad \neg \perp = \neg \begin{bmatrix} \langle \rangle \\ \emptyset \end{bmatrix} = \begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \setminus \emptyset \end{bmatrix} = \begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \end{bmatrix} = \top$$

$$(2) \quad \neg \top = \neg \begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \end{bmatrix} = \begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \setminus \{\langle \rangle\} \end{bmatrix} = \begin{bmatrix} \langle \rangle \\ \emptyset \end{bmatrix} = \perp$$

19.5 Boolean operations on equischematic relations

19.5.1 Definition boolean operations

For two relations $R = [X, \Gamma]$ and $S = [X, \Sigma]$ with identical schema X we define:

$$\begin{aligned} R \subseteq S & \text{ :iff } \Gamma \subseteq \Sigma && \text{(inclusion)} \\ R \cap S & := \begin{bmatrix} X \\ \Gamma \cap \Sigma \end{bmatrix} && \text{((small) intersection)} \\ R \cup S & := \begin{bmatrix} X \\ \Gamma \cup \Sigma \end{bmatrix} && \text{((small) union)} \end{aligned}$$

19.5.2 Definition

As usual, there is a whole range of relations that come along with \subseteq , namely:

$$\begin{aligned} \textcircled{1} \not\subseteq \textcircled{2} & \text{ is not } \textcircled{1} \subseteq \textcircled{2} \\ \textcircled{1} \subset \textcircled{2} & \text{ is } \textcircled{1} \subseteq \textcircled{2} \text{ and } \textcircled{2} \not\subseteq \textcircled{1} \\ \textcircled{1} \supseteq \textcircled{2} & \text{ is } \textcircled{2} \subseteq \textcircled{1} \end{aligned}$$

etc.

19.5.3 Definition big junctions

For every schema X and $\mathcal{R} \subseteq \mathbf{Rel}(X)$ we define

$$\bigcap \mathcal{R} := \bigcap_{R \in \mathcal{R}} R := \begin{cases} \top_X & \text{if } \mathcal{R} = \emptyset \\ \begin{bmatrix} X \\ \bigcap_{R \in \mathcal{R}} \mathbf{gr}(R) \end{bmatrix} & \text{if } \mathcal{R} \neq \emptyset \end{cases}$$

the (big) intersection of \mathcal{R}

$$\bigcup \mathcal{R} := \bigcup_{R \in \mathcal{R}} R := \begin{cases} \perp_X & \text{if } \mathcal{R} = \emptyset \\ \begin{bmatrix} X \\ \bigcup_{R \in \mathcal{R}} \mathbf{gr}(R) \end{bmatrix} & \text{if } \mathcal{R} \neq \emptyset \end{cases}$$

the (big) union of \mathcal{R}

19.5.4 Remark

The definition of \bigcap depends on the schema X , so we should carry this information in the notation to avoid ambiguities and write, say \bigcap_X instead of just \bigcap for the big intersection. However, most of the times X can be reconstructed from the expression $\bigcap \mathcal{R}$ via $X = \bigvee \{\mathbf{gr}(R) \mid R \in \mathcal{R}\}$. Unless \mathcal{R} is empty. In that case, $\bigcap \mathcal{R} = \top_X$ is not well-defined, if X is not obvious from the context. So in doubtful circumstances, let us keep in mind to write

$$\bigcap_X \textcircled{1} \quad \text{and} \quad \bigcup_X \textcircled{2}$$

for the big intersection and big union.

19.5.5 Remark graph table representations

The just defined operations on relations resemble the class operations on their graphs. In particular, the union and intersection of relations in graph table notation is the union and intersection of their rows. And as usual in class notation, multiple occurring elements of the union are only mentioned once. For example, for

$$R = \begin{array}{|c|c|} \hline a : \mathbb{N} & b : \mathbb{N} \\ \hline 1 & 2 \\ \hline 3 & 4 \\ \hline 5 & 6 \\ \hline 7 & 8 \\ \hline \end{array} \quad \text{and} \quad S = \begin{array}{|c|c|} \hline a : \mathbb{N} & b : \mathbb{N} \\ \hline 5 & 6 \\ \hline 7 & 8 \\ \hline 9 & 10 \\ \hline \end{array}$$

we obtain

$$R \cap S = \begin{array}{|c|c|} \hline a : \mathbb{N} & b : \mathbb{N} \\ \hline 5 & 6 \\ \hline 7 & 8 \\ \hline \end{array}$$

and

$$R \cup S = \begin{array}{|c|c|} \hline a : \mathbb{N} & b : \mathbb{N} \\ \hline 1 & 2 \\ \hline 3 & 4 \\ \hline 5 & 6 \\ \hline 7 & 8 \\ \hline 5 & 6 \\ \hline 7 & 8 \\ \hline 9 & 10 \\ \hline \end{array} = \begin{array}{|c|c|} \hline a : \mathbb{N} & b : \mathbb{N} \\ \hline 1 & 2 \\ \hline 3 & 4 \\ \hline 5 & 6 \\ \hline 7 & 8 \\ \hline 9 & 10 \\ \hline \end{array}$$

Furthermore, we have $R \not\subseteq S$, because not every row of R is also a row of S .

19.5.6 Repetition

Recall 5.8.4, that $\mathbb{B} := \{0, 1\}$ is the set of bit values, and together with $\wedge, \vee, \bigwedge, \bigvee$ and \neg it makes a complete boolean algebra. For example, $\neg 0 = 1, 0 \wedge 1 = 0$ and $\bigwedge \{0, 1\} = 0$.

19.5.7 Remark Boolean table representations

If X is completely finite, given by

$$X = \begin{array}{|c|} \hline i_1 \mapsto X_1 \\ \hline \vdots \\ \hline i_n \mapsto X_n \\ \hline \end{array}$$

and if $R, S \in \mathbf{Rel}(X)$ are given by their boolean tables

$$R = \begin{array}{|c|c|c|c|} \hline i_1 & \dots & i_n & \\ \hline \dots & \dots & \dots & \beta_1 \\ \hline \dots & \dots & \dots & \beta_2 \\ \hline \dots & \dots & \dots & \beta_3 \\ \hline \vdots & & \vdots & \vdots \\ \hline \vdots & & \vdots & \vdots \\ \hline \dots & \dots & \dots & \beta_m \\ \hline \end{array} \text{ and } S = \begin{array}{|c|c|c|c|} \hline i_1 & \dots & i_n & \\ \hline \dots & \dots & \dots & \beta'_1 \\ \hline \dots & \dots & \dots & \beta'_2 \\ \hline \dots & \dots & \dots & \beta'_3 \\ \hline \vdots & & \vdots & \vdots \\ \hline \vdots & & \vdots & \vdots \\ \hline \dots & \dots & \dots & \beta'_m \\ \hline \end{array}$$

then

$$R \subseteq S \text{ iff } \beta_i \leq \beta'_i \text{ for all } i \in \{1, \dots, m\}$$

and

$$R \cap S = \begin{array}{|c|c|c|c|} \hline i_1 & \dots & i_n & \\ \hline \dots & \dots & \dots & \beta_1 \wedge \beta'_1 \\ \hline \dots & \dots & \dots & \beta_2 \wedge \beta'_2 \\ \hline \dots & \dots & \dots & \beta_3 \wedge \beta'_3 \\ \hline \vdots & & \vdots & \vdots \\ \hline \dots & \dots & \dots & \beta_m \wedge \beta'_m \\ \hline \end{array}$$

$$R \cup S = \begin{array}{|c|c|c|c|} \hline i_1 & \dots & i_n & \\ \hline \dots & \dots & \dots & \beta_1 \vee \beta'_1 \\ \hline \dots & \dots & \dots & \beta_2 \vee \beta'_2 \\ \hline \dots & \dots & \dots & \beta_3 \vee \beta'_3 \\ \hline \vdots & & \vdots & \vdots \\ \hline \dots & \dots & \dots & \beta_m \vee \beta'_m \\ \hline \end{array}$$

19.5.8 Example

Let $X = \begin{array}{|c|} \hline a \mapsto \{p, q\} \\ \hline b \mapsto \{r, s, t\} \\ \hline \end{array}$ and $R, S \in \mathbf{Rel}(X)$, given by

$$R = \begin{array}{|c|c|c|} \hline a & b & \\ \hline p & r & 0 \\ \hline q & r & 0 \\ \hline p & s & 1 \\ \hline q & s & 1 \\ \hline p & t & 1 \\ \hline q & t & 0 \\ \hline \end{array} \text{ and } S = \begin{array}{|c|c|c|} \hline a & b & \\ \hline p & r & 0 \\ \hline q & r & 1 \\ \hline p & s & 1 \\ \hline q & s & 1 \\ \hline p & t & 0 \\ \hline q & t & 0 \\ \hline \end{array}$$

then $R \not\subseteq S$ and

$$R \cap S = \begin{array}{|c|c|c|} \hline a & b & \\ \hline p & r & 0 \\ \hline q & r & 0 \\ \hline p & s & 1 \\ \hline q & s & 1 \\ \hline p & t & 0 \\ \hline q & t & 0 \\ \hline \end{array} \text{ and } R \cup S = \begin{array}{|c|c|c|} \hline a & b & \\ \hline p & r & 0 \\ \hline q & r & 1 \\ \hline p & s & 1 \\ \hline q & s & 1 \\ \hline p & t & 1 \\ \hline q & t & 0 \\ \hline \end{array}$$

19.5.9 Definition

For every schema X , we define

$$\mathfrak{Rel}(X) := \langle \mathbf{Rel}(X), \subseteq, \perp_X, \top_X, \cap, \cup, \bigcap, \bigcup, \neg \rangle$$

the equi-schematic relation algebra over X or the algebra (on) $\mathbf{Rel}(X)$.

19.5.10 Lemma

$\mathfrak{Rel}(X)$ is a complete boolean algebra, for every schema X .

19.5.11 Remark

Recall (see 7 for the full characterization of complete boolean algebras), that theorem 19.5.10 is defined to mean:

- ♣ \subseteq is a (partial) order on $\mathbf{Rel}(X)$ which determines all the other operations
- ♣ \perp_X is the least element
- ♣ \top_X is the greatest element
- ♣ $R \cap S$ is the greatest lower bound of $R, S \in \mathbf{Rel}(X)$
- ♣ $R \cup S$ is the least upper bound of $R, S \in \mathbf{Rel}(X)$
- ♣ \cap and \cup are mutually distributive
- ♣ $\bigcap \mathcal{R}$ is the greatest lower bound of $\mathcal{R} \subseteq \mathbf{Rel}(X)$
- ♣ $\bigcup \mathcal{R}$ is the least upper bound of $\mathcal{R} \subseteq \mathbf{Rel}(X)$
- ♣ \neg is the complementation, i.e. $\neg R \cap R = \perp_X$ and $\neg R \cup R = \top_X$, for every $R \in \mathbf{Rel}(X)$.

19.5.12 Proof of 19.5.10

Recall (see 6), that for every given C , the power class algebra over C

$$\mathfrak{P}(C) = \langle \mathbf{P}(C), \subseteq, \emptyset, C, \cap, \cup, \bigcap, \bigcup, C \setminus \emptyset \rangle$$

is a complete boolean algebra. In more detail: The power class $\mathbf{P}(C)$ of C is the carrier class, \subseteq is the (partial) order relation, \emptyset is the bottom and C the top element, \cap and \cup are the meet and join with completions $\bigcap, \bigcup : \mathbf{P}(\mathbf{P}(C)) \rightarrow \mathbf{P}(C)$ (where $\bigcap \emptyset := C$), and $C \setminus \textcircled{1} : \mathbf{P}(C) \rightarrow \mathbf{P}(C)$ is the complementation.

Now let X be a given schema. We define

$$\varphi := \left[\begin{array}{l} \mathbf{P}(\otimes X) \longrightarrow \mathbf{Rel}(X) \\ \Gamma \mapsto [X, \Gamma] \end{array} \right]$$

Obviously, φ is a bijection and its inverse φ^{-1} is

$$\varphi^{-1} = \left[\begin{array}{l} \mathbf{Rel}(X) \longrightarrow \mathbf{P}(\otimes X) \\ [X, \Gamma] \mapsto \Gamma \end{array} \right]$$

i.e. $\varphi^{-1}(R) = \mathbf{gr}(R)$, for all $R \in \mathbf{Rel}(X)$.

It is easy to see that

- ♣ $\Gamma \subseteq \Sigma$ iff $\varphi(\Gamma) \subseteq \varphi(\Sigma)$ for all $\Gamma, \Sigma \in \mathbf{P}(\otimes X)$
- ♣ $\varphi(\emptyset) = \perp_X$
- ♣ $\varphi(\otimes X) = \top_X$
- ♣ $\varphi(\Gamma \cap \Sigma) = \varphi(\Gamma) \cap \varphi(\Sigma)$ for all $\Gamma, \Sigma \in \mathbf{P}(\otimes X)$
- ♣ $\varphi(\Gamma \cup \Sigma) = \varphi(\Gamma) \cup \varphi(\Sigma)$ for all $\Gamma, \Sigma \in \mathbf{P}(\otimes X)$
- ♣ $\varphi(\bigcap \mathcal{G}) = \bigcap \{\varphi(\Gamma) \mid \Gamma \in \mathcal{G}\}$ for every $\mathcal{G} \subseteq \mathbf{P}(\otimes X)$
- ♣ $\varphi(\bigcup \mathcal{G}) = \bigcup \{\varphi(\Gamma) \mid \Gamma \in \mathcal{G}\}$ for every $\mathcal{G} \subseteq \mathbf{P}(\otimes X)$
- ♣ $\varphi(\otimes X \setminus \Gamma) = \neg \varphi(\Gamma)$ for each $\Gamma \in \mathbf{P}(\otimes X)$

And that means altogether, that φ is an isomorphism from $\mathfrak{P}(\otimes X)$ into $\mathfrak{Rel}(X)$, written

$$\varphi : \mathfrak{P}(\otimes X) \cong \mathfrak{Rel}(X)$$

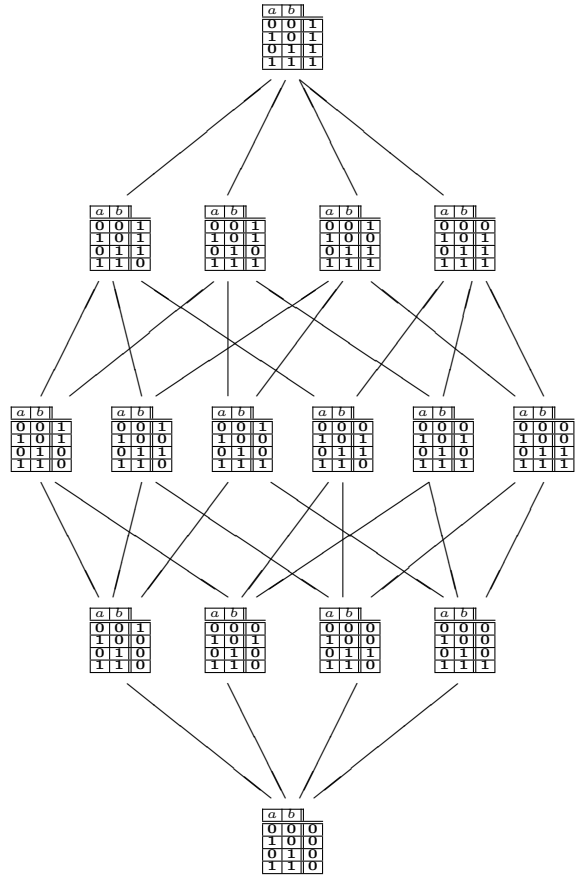
That implies, that $\mathfrak{Rel}(X)$ is a complete boolean algebra itself.

19.5.13 Example

Consider the schema X from example 17.8.3 again

$$X = \left[\begin{array}{l} a \mapsto \mathbb{B} \\ b \mapsto \mathbb{B} \end{array} \right]$$

The complete boolean lattice on $\mathbf{Rel}(X)$ has 16 members and is represented by the following order diagram:



19.5.14 Lemma — common properties of complete boolean algebras

Let X be a proper schema. For all $R, S, T \in \mathbf{Rel}(X)$ and all $\mathcal{R} \subseteq \mathbf{Rel}(X)$ holds:

- (1) $\top_X \cap R = R$ (\top_X is neutral element of \cap)
- (2) $\perp_X \cup R = R$ (\perp_X is neutral element of \cup)
- (3) $R \cap R = R$ (\cap is idempotent)
- (4) $R \cup R = R$ (\cup is idempotent)
- (5) $R \cap (S \cap T) = (R \cap S) \cap T$ (\cap is associative)
- (6) $R \cup (S \cup T) = (R \cup S) \cup T$ (\cup is associative)
- (7) $R \cap S = S \cap R$ (\cap is commutative)
- (8) $R \cup S = S \cup R$ (\cup is commutative)
- (9) $R \cap S = \bigcap \{R, S\}$ (\cap is special case of \bigcap)
- (10) $R \cup S = \bigcup \{R, S\}$ (\cup is special case of \bigcup)
- (11) $S \cap \bigcup \mathcal{R} = \bigcup \{S \cap R \mid R \in \mathcal{R}\}$ (full distributivity)
- (12) $S \cup \bigcap \mathcal{R} = \bigcap \{S \cup R \mid R \in \mathcal{R}\}$ (full distributivity)
- (13) $\neg \bigcap \mathcal{R} = \bigcup \{\neg R \mid R \in \mathcal{R}\}$ (de Morgan's law)
- (14) $\neg \bigcup \mathcal{R} = \bigcap \{\neg R \mid R \in \mathcal{R}\}$ (de Morgan's law)
- (15) $\perp_X \cap R \equiv \perp_X$ (\perp_X cancels \cap)
- (16) $\top_X \cup R \equiv \top_X$ (\top_X cancels \cup)
- (17) $\perp_X \subseteq R$ (\perp_X is a least element)
- (18) $R \subseteq \top_X$ (\top_X is a greatest element)
- (19) $\bigcap \{R\} = R$ (\bigcap is idempotent)
- (20) $\bigcup \{R\} = R$ (\bigcup is idempotent)

19.5.15 Proof of 19.5.14

These properties hold for complete boolean algebras in general (see 7). So they all follow from 19.5.10.

19.5.16 Remark

If the schema X is not proper, then $\otimes X = \emptyset$ (according to 12.1.12). For such an X , $\mathbf{Rel}(X)$ has exactly one member $R = [X, \emptyset]$ and $R = \perp_X = \top_X$ (according to 19.3.5(4)). Nevertheless, the algebra on $\mathbf{Rel}(X)$ still satisfies the properties of a complete boolean algebra. However, such an one-element boolean algebra is usually not a very useful one and it is also called degenerated. In all other case of proper schemas X , the algebra has at least two elements \perp_X and \top_X with $\perp_X \neq \top_X$.

19.6 Distinct (cartesian) product

19.6.1 Repetition

Two records $\xi = [\xi_i | i \in I]$ and $v = [v_j | j \in J]$ are distinct, written $\xi \check{\vee} v$, if $I \cap J = \emptyset$. Their distinct join is

$$\xi \check{\vee} v := [\zeta_k | k \in (I \cup J)] \quad \text{with} \quad \zeta_k := \begin{cases} \xi_k & \text{if } k \in I \\ v_k & \text{if } k \in J \end{cases}$$

Two relations $R = [X, \Gamma]$ and $S = [Y, \Sigma]$ are distinct, written $R \check{\vee} S$, if their schema is distinct, i.e. $X \check{\vee} Y$. A class \mathcal{R} of relations is (pairwise) distinct, if each two different members are distinct.

Two graphs (i.e. record classes) Γ and Σ are relatively distinct, written $\Gamma \check{\vee} \Sigma$, if $\xi \check{\vee} v$ for all $\xi \in \Gamma$ and $v \in \Sigma$. And that is exactly the case, when there are two distinct schemas X and Y with $\Gamma \subseteq \otimes X$ and $\Sigma \subseteq \otimes Y$.

If Γ and Σ are two relatively distinct graphs, then $\Gamma \odot \Sigma := \{\xi \check{\vee} v | \xi \in \Gamma, v \in \Sigma\}$ is their distinct product. The number of elements in the distinct product is given by $\mathbf{card}(\Gamma \odot \Sigma) = \mathbf{card}(\Gamma) \cdot \mathbf{card}(\Sigma)$.

19.6.2 Definition

For every two distinct relations $R = [X, \Gamma]$ and $S = [Y, \Sigma]$ we define

$$R \odot S := \left[\begin{array}{c} X \check{\vee} Y \\ \{x \check{\vee} y | x \in \Gamma, y \in \Sigma\} \end{array} \right] = \left[\begin{array}{c} X \check{\vee} Y \\ \Gamma \odot \Sigma \end{array} \right]$$

the distinct (cartesian) product of R and S

More general, if $\mathcal{R} = \{\mathcal{R}_k | k \in K\}$ is a (pairwise) distinct relation class with $\mathcal{R}_k = [X_k, \Gamma_k]$ for each $k \in K$, then

$$\odot \mathcal{R} := \bigodot_{k \in K} \mathcal{R}_k := \left[\begin{array}{c} \bigvee_{k \in K} X_k \\ \bigodot_{k \in K} \Gamma_k \end{array} \right]$$

the distinct (cartesian) product of \mathcal{R}

19.6.3 Remark

Obviously, the general distinct product on distinct relation

classes is indeed a generalization of the binary $\textcircled{1} \odot \textcircled{2}$ in the usual sense that $R \odot S = \odot \{R, S\}$, for all relations R and S with $R \check{\vee} S$.

19.6.4 Remark Graph table representation

Given two distinct tables, say

$$R = \begin{array}{|c|c|c|} \hline a : \mathbb{Z} & b : \mathbb{Z} & c : \mathbb{Z} \\ \hline -1 & -2 & -3 \\ \hline -2 & -3 & -4 \\ \hline -4 & -5 & -6 \\ \hline \end{array} \quad S = \begin{array}{|c|c|} \hline d : \mathbb{Z} & e : \mathbb{Z} \\ \hline 1 & 2 \\ \hline 2 & 3 \\ \hline \end{array}$$

Note, that $\{a, b, c\} \cap \{d, e\} = \emptyset$, so that $R \check{\vee} S$ is really the case. So we can produce the distinct product by joining each of the three records in R with each of the two records in S , obtaining a relation with $3 \cdot 2 = 6$ members in its graph.

$$R \odot S = \begin{array}{|c|c|c|c|c|} \hline a : \mathbb{Z} & b : \mathbb{Z} & c : \mathbb{Z} & d : \mathbb{Z} & e : \mathbb{Z} \\ \hline -1 & -2 & -3 & 1 & 2 \\ \hline -2 & -3 & -4 & 1 & 2 \\ \hline -4 & -5 & -6 & 1 & 2 \\ \hline -1 & -2 & -3 & 2 & 3 \\ \hline -2 & -3 & -4 & 2 & 3 \\ \hline -4 & -5 & -6 & 2 & 3 \\ \hline \end{array}$$

19.6.5 Lemma

If R, S, T are pairwise distinct relations, then

- (1) $R \odot (S \odot T) = (R \odot S) \odot T$ (associativity)
- (2) $R \odot S = S \odot R$ (commutativity)
- (3) $R \odot \perp = \perp_{\mathbf{x}(R)}$ (canceling element)
- (4) $R \odot \top = R$ (neutral element)

19.6.6 Proof of 19.6.5

The distinct product of relations is a “typed version” of the distinct product of (their) graphs. Their properties are closely related, the ones here resemble the statements in 16.4.3. Suppose $R = [X, \Gamma]$, $S = [Y, \Sigma]$, and $T = [Z, \Pi]$.

(1) Using 16.4.3(1) we obtain

$$\begin{aligned} R \odot (S \odot T) &= \left[\begin{array}{c} X \\ \Gamma \end{array} \right] \odot \left[\begin{array}{c} Y \check{\vee} Z \\ \Sigma \odot \Pi \end{array} \right] \\ &= \left[\begin{array}{c} X \check{\vee} Y \check{\vee} Z \\ \Gamma \odot \Sigma \odot \Pi \end{array} \right] \\ &= \left[\begin{array}{c} X \check{\vee} Y \\ \Gamma \odot \Sigma \end{array} \right] \odot \left[\begin{array}{c} Z \\ \Pi \end{array} \right] = (R \odot S) \odot T \end{aligned}$$

(2) Using 16.4.3(2) we obtain

$$R \odot S = \left[\begin{array}{c} X \check{\vee} Y \\ \Gamma \odot \Sigma \end{array} \right] = \left[\begin{array}{c} Y \check{\vee} X \\ \Sigma \odot \Gamma \end{array} \right] = S \odot R$$

(3) Using 16.4.3(3) we obtain

$$R \odot \perp = \left[\begin{array}{c} X \\ \Gamma \end{array} \right] \odot \left[\begin{array}{c} \langle \rangle \\ \emptyset \end{array} \right] = \left[\begin{array}{c} X \odot \langle \rangle \\ \Gamma \odot \emptyset \end{array} \right] = \left[\begin{array}{c} X \\ \emptyset \end{array} \right] = \perp_X$$

(4) Using 16.4.3(4) we obtain

$$R \odot \top = \left[\begin{array}{c} X \\ \Gamma \end{array} \right] \odot \left[\begin{array}{c} \langle \rangle \\ \{\langle \rangle\} \end{array} \right] = \left[\begin{array}{c} X \check{\vee} \langle \rangle \\ \Gamma \odot \{\langle \rangle\} \end{array} \right] = \left[\begin{array}{c} X \\ \Gamma \end{array} \right] = R$$

19.6.7 Lemma

If R and S are two distinct relations, then

- (1) $\neg(R \odot S) = (\neg R \odot \neg S) \cup (\neg R \odot S) \cup (R \odot \neg S)$
- (2) $\neg R \odot \neg S \subseteq \neg(R \odot S)$

19.6.8 Proof of 19.6.7

Let $R = [Y, \Gamma]$ and $S = [Z, \Sigma]$. R and S are distinct, so $Y \not\subseteq Z$. Let us put $X := Y \dot{\vee} Z$, then

$$\otimes X = \otimes(Y \dot{\vee} Z) = (\otimes Y) \odot (\otimes Z)$$

according to 16.6.5(4). Furthermore, let us put $\Gamma' := \otimes Y \setminus \Gamma$ and $\Sigma' := \otimes Z \setminus \Sigma$. So $\otimes Y$ is the disjunct union of Γ and Γ' , and $\otimes Z$ is the disjunct union of Σ and Σ' . $\otimes X$ is a disjunct union of the following four components:

$$\begin{aligned} \otimes X &= (\otimes Y) \odot (\otimes Z) \\ &= (\Gamma \cup \Gamma') \odot (\Sigma \cup \Sigma') \\ &= (\Gamma \odot \Sigma) \cup (\Gamma \odot \Sigma') \cup (\Gamma' \odot \Sigma) \cup (\Gamma' \odot \Sigma') \end{aligned}$$

With these definitions we have the following proof of (1):

$$\begin{aligned} \neg(R \odot S) &= \neg\left(\begin{bmatrix} Y \\ \Gamma \end{bmatrix} \odot \begin{bmatrix} Z \\ \Sigma \end{bmatrix}\right) \\ &= \neg\begin{bmatrix} X \\ \Gamma \odot \Sigma \end{bmatrix} \\ &= \begin{bmatrix} X \\ \otimes X \setminus (\Gamma \odot \Sigma) \end{bmatrix} \\ &= \begin{bmatrix} X \\ (\Gamma \odot \Sigma') \cup (\Gamma' \odot \Sigma) \cup (\Gamma' \odot \Sigma') \end{bmatrix} \\ &= \begin{bmatrix} X \\ \Gamma \odot \Sigma' \end{bmatrix} \cup \begin{bmatrix} X \\ \Gamma' \odot \Sigma \end{bmatrix} \cup \begin{bmatrix} X \\ \Gamma' \odot \Sigma' \end{bmatrix} \\ &= (R \odot \neg S) \cup (\neg R \odot S) \cup (\neg R \odot \neg S) \end{aligned}$$

Since (1) is true, (2) immediately follows:

$$\begin{aligned} \neg R \odot \neg S &\subseteq (R \odot \neg S) \cup (\neg R \odot S) \cup (\neg R \odot \neg S) \\ &= \neg(R \odot S) \end{aligned}$$

19.7 Concatenation of ordinary relations**19.7.1** Repetition

Recall 5.4.1, that $\langle x_1, \dots, x_n \rangle \dagger \langle y_1, \dots, y_m \rangle = \langle x_1, \dots, x_n, y_1, \dots, y_m \rangle$ is the *concatenation* of two tuples.

19.7.2 Definition

For every two ordinary relations

$$R = \begin{bmatrix} X_1 \rightsquigarrow \dots \rightsquigarrow X_n \\ \Gamma \end{bmatrix} \text{ and } S = \begin{bmatrix} Y_1 \rightsquigarrow \dots \rightsquigarrow Y_m \\ \Sigma \end{bmatrix}$$

we define

$$R \dagger S := \begin{bmatrix} X_1 \rightsquigarrow \dots \rightsquigarrow X_n \rightsquigarrow Y_1 \rightsquigarrow \dots \rightsquigarrow Y_m \\ \{\xi \dagger v \mid \xi \in \Gamma, v \in \Sigma\} \end{bmatrix}$$

is the *concatenation* of R and S .

19.7.3 Example

$$\begin{aligned} &\begin{bmatrix} \mathbb{N} \rightsquigarrow \mathbb{N} \\ \{(1, 2), \langle 3, 4 \rangle\} \end{bmatrix} \dagger \begin{bmatrix} \mathbb{N} \rightsquigarrow \mathbb{N} \\ \{\langle 5, 6 \rangle, \langle 7, 8 \rangle\} \end{bmatrix} \\ &= \begin{bmatrix} \mathbb{N} \rightsquigarrow \mathbb{N} \rightsquigarrow \mathbb{N} \rightsquigarrow \mathbb{N} \\ \{(1, 2, 5, 6), \langle 3, 4, 5, 6 \rangle, \langle 1, 2, 7, 8 \rangle, \langle 3, 4, 7, 8 \rangle\} \end{bmatrix} \end{aligned}$$

19.7.4 Lemma

For all ordinary relations R, S, T holds:

- (1) $(R \dagger S) \dagger T = R \dagger (S \dagger T)$ (associativity)
- (2) $R \dagger \top = R$ (\top is neutral element for \dagger)
- (3) $R \dagger \perp = \perp_{\mathbf{x}(R)}$ (\perp cancels \dagger)

19.7.5 Proof of 19.7.4

(1) Is an immediate consequence of 10.5.3(1), the associativity of the concatenation of tuples.

(2) If $R = [X_1 \rightsquigarrow \dots \rightsquigarrow X_n, \Gamma]$, then

$$R \dagger \top = R \dagger \begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \end{bmatrix} = \begin{bmatrix} X_1 \rightsquigarrow \dots \rightsquigarrow X_n \\ \{\xi \dagger \langle \rangle \mid \xi \in \Gamma\} \end{bmatrix} = R$$

(3) Again, if $R = [X_1 \rightsquigarrow \dots \rightsquigarrow X_n, \Gamma]$, then

$$R \dagger \perp = R \dagger \begin{bmatrix} \langle \rangle \\ \emptyset \end{bmatrix} = \begin{bmatrix} X_1 \rightsquigarrow \dots \rightsquigarrow X_n \\ \emptyset \end{bmatrix} = \perp_{\langle X_1, \dots, X_n \rangle}$$

20 Schema expansion and the semantic operations

20.1 Schema expansion

20.1.1 Definition expansion

Let $R = [X, \Gamma]$ be relation and Y a schema with $Y \smile X$. We define

$$R \parallel Y := \left[\begin{array}{c} X \vee Y \\ y \rightsquigarrow \exists x \in \Gamma. x \smile y \end{array} \right]$$

the expansion of R by Y

20.1.2 Remark table representations

If $R = [X, \Gamma]$ is a relation and Y a schema with $Y \smile X$, there are two intuitive methods to generate $R \parallel Y$ by means of table representations:

- ♣ If R is a table and Y is completely finite, we can use a graph table representation (see 20.1.6).
- ♣ If X and Y are both completely finite, we can apply boolean and double tables (see 20.1.5).

Both methods are based on the following lemma.

20.1.3 Lemma construction

Let $X = [X_i | i \in I]$ and $Y = [Y_j | j \in J]$ be two compatible schemas and $R = [X, \Gamma]$. If we put

$$Y' := Y \setminus X = \mathbf{pr}(Y, J \setminus I)$$

we have

- (1) $R \parallel Y = R \parallel Y'$
- (2) $R \parallel Y = \left[\begin{array}{c} X \dot{\vee} Y' \\ \Gamma \odot (\otimes Y') \end{array} \right]$
- (3) $R \parallel Y = R \odot \top_{Y'}$
- (4) The graph of the result has the following size:

$$\begin{aligned} \mathbf{card}(\mathbf{gr}(R \parallel Y)) &= \mathbf{card}(\Gamma) \cdot \mathbf{card}(\otimes Y') \\ &= \mathbf{card}(\Gamma) \cdot \prod_{j \in J \setminus I} \mathbf{card}(Y_j) \end{aligned}$$

20.1.4 Proof of 20.1.3

- (1) Obviously, $X \vee Y = X \dot{\vee} Y'$, so

$$\begin{aligned} R \parallel Y &= [X \vee Y, \{y \in \otimes(X \vee Y) \mid \exists x \in \Gamma. x \smile y\}] \\ &= [X \dot{\vee} Y', \{y \in \otimes(X \dot{\vee} Y') \mid \exists x \in \Gamma. x \smile y\}] \\ &= R \parallel Y' \end{aligned}$$

- (2) Again we have $X \vee Y = X \dot{\vee} Y'$. X and Y' are dis-

tinct, so every $y \in \otimes(X \vee Y')$ can uniquely be reconstructed by $y = x \dot{\vee} z$ with $x := \mathbf{pr}(y, I) \in \otimes X$ and $z := \mathbf{pr}(y, J \setminus I) \in \otimes Y'$. For such a y holds: $(\exists x \in \Gamma. x \smile y)$ iff $x \in \Gamma$. So

$$\left\{ \frac{y \in \otimes(X \vee Y)}{\exists x \in \Gamma. x \smile y} \right\} = \left\{ \frac{x \dot{\vee} z}{x \in \Gamma, z \in \otimes Y'} \right\}$$

and thus

$$\begin{aligned} R \parallel Y' &= [X \dot{\vee} Y', \{x \dot{\vee} y \mid x \in \Gamma, y \in \otimes Y'\}] \\ &= [X \vee Y', \Gamma \odot (\otimes Y')] \end{aligned}$$

- (3) We have

$$\begin{aligned} R \parallel Y &= \left[\begin{array}{c} X \dot{\vee} Y' \\ \Gamma \odot (\otimes Y') \end{array} \right] \quad \text{due to (2)} \end{aligned}$$

$$= \left[\begin{array}{c} X \\ \Gamma \end{array} \right] \odot \left[\begin{array}{c} Y' \\ \otimes Y' \end{array} \right] \quad \text{def. 19.6.2 of } \odot$$

$$= R \odot \top_{Y'} \quad \text{def. 19.3.1 of } \top_{Y'}$$

- (4) There is

$$\begin{aligned} &\mathbf{card}(\mathbf{gr}(R \parallel Y)) \\ &= \mathbf{card}(\Gamma \odot \otimes Y') \quad \text{see (2)} \\ &= \mathbf{card}(\Gamma) \cdot \mathbf{card}(\otimes Y') \quad \text{due to 16.4.10} \\ &= \mathbf{card}(\Gamma) \cdot \prod_{j \in J \setminus I} \mathbf{card}(Y_j) \quad \text{due to 12.1.9} \end{aligned}$$

20.1.5 Boolean and double table representation

If X and Y both completely finite schemas, $X \smile Y$, and $R = [X, \Gamma]$, then $R \parallel Y$ can be constructed intuitively by means of boolean and double tables.

Based on 20.1.3(3) $R \parallel Y = R \odot \top_{Y \setminus X}$ we obtain $R \parallel Y$ by performing the following steps:

- ♣ Given X and Y , say

$$X = \left[\begin{array}{c} a \mapsto \{1, 2\} \\ b \mapsto \{1, 2, 3\} \end{array} \right] \quad Y = \left[\begin{array}{c} b \mapsto \{1, 2, 3\} \\ c \mapsto \{-1, -2\} \\ d \mapsto \{-1, -2\} \end{array} \right]$$

and R as boolean table, say

$$R = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 1 & \mathbf{0} \\ \hline 2 & 1 & \mathbf{1} \\ \hline 1 & 2 & \mathbf{1} \\ \hline 2 & 2 & \mathbf{0} \\ \hline 1 & 3 & \mathbf{1} \\ \hline 2 & 3 & \mathbf{0} \\ \hline \end{array}$$

- ♣ First, generate $Y \setminus X$. In our example, that is

$$Y \setminus X = \left[\begin{array}{c} c \mapsto \{-1, -2\} \\ d \mapsto \{-1, -2\} \end{array} \right]$$

- ♣ Write $\top_{(Y \setminus X)}$ as a graph table, i.e.

$$\top_{(Y \setminus X)} = \begin{array}{|c|c|} \hline c : \{-1, -2\} & d : \{-1, -2\} \\ \hline -1 & -1 \\ \hline -2 & -1 \\ \hline -1 & -2 \\ \hline -2 & -2 \\ \hline \end{array}$$

$$\top_{(Y \setminus X)} = \begin{array}{|c|c|} \hline c : \{-1, -2\} & d : \{-1, -2\} \\ \hline -1 & -1 \\ \hline -2 & -1 \\ \hline -1 & -2 \\ \hline -2 & -2 \\ \hline \end{array}$$

For our purposes, it is even more intuitive, if we rotate this table and write it horizontally as

$$\top_{(Y \setminus X)} = \begin{array}{|c|c|c|c|c|} \hline -1 & -2 & -1 & -2 & c : \{-1, -2\} \\ \hline -1 & -1 & -2 & -2 & d : \{-1, -2\} \\ \hline \end{array}$$

Finally, we take R and $\top_{(Y \setminus X)}$, i.e.

$$R = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 1 & 0 \\ \hline 2 & 1 & 1 \\ \hline 1 & 2 & 1 \\ \hline 2 & 2 & 0 \\ \hline 1 & 3 & 1 \\ \hline 2 & 3 & 0 \\ \hline \end{array}$$

and

$$\begin{array}{|c|c|c|c|c|} \hline -1 & -2 & -1 & -2 & c : \{-1, -2\} \\ \hline -1 & -1 & -2 & -2 & d : \{-1, -2\} \\ \hline \end{array}$$

Generate the distinct cartesian product by using the graph table representations

$$R \odot \top_{(Y \setminus X)} = \begin{array}{|c|c|c|c|c|} \hline a : \mathbb{N} & b : \mathbb{N} & c : \{-1, -2\} & d : \{-1, -2\} & \\ \hline 2 & 1 & -1 & -1 & \\ \hline 1 & 2 & -1 & -1 & \\ \hline 1 & 3 & -1 & -1 & \\ \hline 2 & 1 & -2 & -1 & \\ \hline 1 & 2 & -2 & -1 & \\ \hline 1 & 3 & -2 & -1 & \\ \hline 2 & 1 & -1 & -2 & \\ \hline 1 & 2 & -1 & -2 & \\ \hline 1 & 3 & -1 & -2 & \\ \hline 2 & 1 & -1 & -2 & \\ \hline 1 & 2 & -1 & -2 & \\ \hline 1 & 3 & -1 & -2 & \\ \hline \end{array}$$

The result is a graph table representation of $R \parallel Y$.

Again, 20.1.3(4) is verified. The R has 3, $\top_{(Y \setminus X)}$ has $2 \cdot 2$ members, and thus $R \parallel Y$ has $3 \cdot 4 = 12$ members, i.e. rows in its graph table.

and construct the double table of $R \odot \top_{(Y \setminus X)}$, i.e.

$$\begin{array}{|c|c|c|c|c|c|} \hline & & -1 & -2 & -1 & -2 & c \\ \hline & & -1 & -1 & -2 & -2 & d \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & \\ \hline 2 & 1 & 1 & 1 & 1 & 1 & \\ \hline 1 & 2 & 1 & 1 & 1 & 1 & \\ \hline 2 & 2 & 0 & 0 & 0 & 0 & \\ \hline 1 & 3 & 1 & 1 & 1 & 1 & \\ \hline 2 & 3 & 0 & 0 & 0 & 0 & \\ \hline a & b & & & & & \\ \hline \end{array}$$

by placing R on the left, $\top_{(Y \setminus X)}$ at the top, and copying the bit values of R throughout each row of the double table. The result is the double table representation for $R \parallel Y$.

As predicted in 20.1.3(4), it has $3 \cdot (2 \cdot 2) = 12$ member records (i.e. twelve 1's in its boolean and double table).

20.1.6 Graph table representation

If $R = [X, \Gamma]$ is a table and Y a schema such that $Y \smile X$ and $Y \setminus X$ is completely finite, we can apply graph tables to generate $R \parallel Y$.

Based on 20.1.3(3) $R \parallel Y = R \odot \top_{Y \setminus X}$ and 19.6.4 the graph table representation of distinct cartesian products, we obtain $R \parallel Y$ by performing the following steps:

Given X and Y , say

$$X = \begin{bmatrix} a \mapsto \mathbb{N} \\ b \mapsto \mathbb{N} \end{bmatrix} \quad Y = \begin{bmatrix} b \mapsto \mathbb{N} \\ c \mapsto \{-1, -2\} \\ d \mapsto \{-1, -2\} \end{bmatrix}$$

such that $Y \setminus X$ is completely finite, which is indeed the case here, where

$$Y \setminus X = \begin{bmatrix} c \mapsto \{-1, -2\} \\ d \mapsto \{-1, -2\} \end{bmatrix}$$

Also given the table R , represented as a graph table

$$R = \begin{array}{|c|c|} \hline a : \mathbb{N} & b : \mathbb{N} \\ \hline 2 & 1 \\ \hline 1 & 2 \\ \hline 1 & 3 \\ \hline \end{array}$$

Generate the graph table representation of $\top_{(Y \setminus X)}$

20.1.7 Lemma compatibility

If X and Y are two compatible schemas and $R \in \mathbf{Rel}(X)$, then $R \parallel Y \smile R$.

20.1.8 Proof of 20.1.7

X and Y are compatible, so $X \vee Y \smile X$, implying $\mathbf{x}(R \parallel Y) \smile \mathbf{x}(R)$, so that $R \parallel Y \smile R$.

20.1.9 Lemma unification

If $R = [X, \Gamma]$ and $S = [Y, \Sigma]$ are two compatible relations, then

$$R \parallel Y = R \parallel (X \vee Y) \triangleq S \parallel (X \vee Y) = S \parallel X$$

More general, if \mathcal{R} is a class of (pairwise) compatible relations, $Z := \bigvee \{\mathbf{x}(R) \mid R \in \mathcal{R}\}$ is well-defined and

$$R \parallel Z \triangleq S \parallel Z \text{ for all } R, S \in \mathcal{R}$$

20.1.10 Proof of 20.1.7

$\mathbf{x}(R \parallel Y) = X \vee Y = \mathbf{x}(S \parallel X)$, so $R \parallel Y = R \parallel (X \vee Y) \triangleq S \parallel (X \vee Y) = S \parallel X$. And because $R \in \mathcal{R}$ implies $\mathbf{x}(R) \leq Z$, we have $\mathbf{x}(R \parallel Z) = X \vee Z = Z = Y \vee Z = \mathbf{x}(S \parallel Z)$ and thus $R \parallel Z \triangleq S \parallel Z$, for all $R, S \in \mathcal{R}$.

20.1.11 Lemma

Let $R = [X, \Gamma]$ be a relation.

(1) If X, Y, Z are pairwise compatible schemas, then $(R \parallel Y) \parallel Z = R \parallel (Y \vee Z)$ (schema accumulation)

(2) If Y is a schema such that $Y \smile X$, then $(R \parallel Y) \parallel Y = R \parallel Y$ (idempotency)

(3) If $Y \leq X$, then $R \parallel Y = R$ (neutral expansions)

20.1.12 Proof of 20.1.11

(1) We put

$$\begin{aligned} Y' &:= \mathbf{pr}(Y, @ (Y) \setminus @ (X)) \\ Z' &:= \mathbf{pr}(Z, @ (Z) \setminus (@ (X) \cup @ (Y))) \end{aligned}$$

so that

$$X \vee Y \vee Z = X \dot{\vee} Y' \dot{\vee} Z'$$

We derive

$$\begin{aligned} (R \parallel Y) \parallel Z & \\ &= (R \odot \top_{Y'}) \odot \top_{Z'} && \text{due to 20.1.3} \\ &= R \odot (\top_{Y'} \odot \top_{Z'}) && \text{due to 19.6.5} \\ &= R \odot \top_{(Y' \dot{\vee} Z')} \\ &= R \parallel (Y' \dot{\vee} Z') && \text{due to 20.1.3(3)} \\ &= R \parallel (Y \vee Z) && \text{due to 20.1.3(1)} \end{aligned}$$

(2) Applying (1), we obtain

$$(R \parallel Y) \parallel Y = R \parallel (Y \vee Y) = R \parallel Y$$

(3) $Y \leq X$ implies $Y \setminus X = \langle \rangle$. So applying 20.1.3 gives us

$$R \parallel Y = R \odot \top_{\langle \rangle} = R \odot \top = R$$

where the last step is due to 19.6.5(4).

20.2 Semantic operations on compatible relations

20.2.1 Remark [introduction](#)

Let $R = [X, \Gamma]$ and $S = [Y, \Sigma]$ be two relations. We already have the operations \subseteq, \cap, \cup available to compare and combine R and S . But they are only defined in case $X = Y$, i.e. if $R \triangleq S$.

We will now generalize the definition of \subseteq, \cap, \cup to more powerful operations $\sqsubseteq, \sqcap, \sqcup$, which are defined for the more general case $X \smile Y$, i.e. $R \smile S$.

We use the unification lemma 20.1.9, which tells us that $R \parallel Y \triangleq S \parallel X$. This way we introduce the new operations by means of the already existing ones. For example

$$R \sqcap S := (R \parallel Y) \cap (S \parallel X) = (R \parallel X \vee Y) \cap (S \parallel X \vee Y)$$

20.2.2 Definition [boolean operations, extended](#)

Let $R = [X, \Gamma]$ and $S = [Y, \Sigma]$ be two compatible relations. With $Z := X \vee Y$ we define

$$R \sqsubseteq S \quad \text{:iff} \quad (R \parallel Z) \subseteq (S \parallel Z) \quad \text{(subvalence)}$$

$$R \equiv S \quad \text{:iff} \quad (R \parallel Z) = (S \parallel Z) \quad \text{(equivalence)}$$

$$R \sqcap S \quad := \quad (R \parallel Z) \cap (S \parallel Z) \quad \text{(conjunction)}$$

$$R \sqcup S \quad := \quad (R \parallel Z) \cup (S \parallel Z) \quad \text{(disjunction)}$$

20.2.3 Definition

As usual, there is a whole range of relations mutating from \sqsubseteq and \equiv , namely:

$$\textcircled{1} \sqsubset \textcircled{2} \quad \text{is} \quad \text{not} \quad \textcircled{1} \sqsubseteq \textcircled{2}$$

$$\textcircled{1} \not\equiv \textcircled{2} \quad \text{is} \quad \text{not} \quad \textcircled{1} \equiv \textcircled{2}$$

$$\textcircled{1} \sqsubset \textcircled{2} \quad \text{is} \quad \textcircled{1} \sqsubseteq \textcircled{2} \quad \text{and} \quad \textcircled{1} \not\equiv \textcircled{2}$$

$$\textcircled{1} \sqsupset \textcircled{2} \quad \text{is} \quad \textcircled{2} \sqsubseteq \textcircled{1}$$

etc.

20.2.4 Definition

Let \mathcal{R} be a compatible relation class. Then

$$Z := \bigvee \{ \mathbf{x}(R) \mid R \in \mathcal{R} \}$$

is a well-defined schema and

$$\prod \mathcal{R} := \prod_{R \in \mathcal{R}} R := \begin{cases} \top & \text{if } \mathcal{R} = \emptyset \\ \bigcap_{R \in \mathcal{R}} (R \parallel Z) & \text{if } \mathcal{R} \neq \emptyset \end{cases}$$

is the (big) conjunction of \mathcal{R}

$$\coprod \mathcal{R} := \prod_{R \in \mathcal{R}} R := \begin{cases} \perp & \text{if } \mathcal{R} = \emptyset \\ \bigcup_{R \in \mathcal{R}} (R \parallel Z) & \text{if } \mathcal{R} \neq \emptyset \end{cases}$$

is the (big) disjunction of \mathcal{R}

20.2.5 Remark [Various table representations](#)

The various table representations for $\sqsubseteq, \equiv, \sqcap, \sqcup$ are constructed according to the given definitions in two steps: first, calculate $R \parallel Y$ and $S \parallel X$, then perform the method for $\subseteq, =, \cap, \cup$, respectively.

As there is nothing really new in this two-step method, we put the discussion of the graph and boolean table method in a separate subsection (see 20.3) that may safely be skipped.

20.3 Digression: performing semantic operations with tables

20.3.1 Remark [introduction](#)

We now take two tables R and S and perform the new operations, first in graph table notation and second by means of boolean table representations.

20.3.2 [Graph table representation](#)

Given two relations $R = [X, \Gamma]$ and $S = [Y, \Gamma]$ in graph table notation as

$$R = \begin{array}{|c|c|} \hline a : \{p, q, r\} & b : \{s, t\} \\ \hline p & s \\ \hline q & t \\ \hline \end{array}$$

and

$$S = \begin{array}{|c|c|c|} \hline b : \{s, t\} & c : \{u, v\} & d : \{w\} \\ \hline s & u & w \\ \hline s & v & w \\ \hline t & v & w \\ \hline \end{array}$$

Note, that R and S are compatible, because $@(R) \cap @(S) = \{b\}$ and $\mathbf{dom}_b(R) = \{s, t\} = \mathbf{dom}_b(S)$. We first join the schemas and obtain

$$Z := X \vee Y = \left[\begin{array}{l} a \mapsto \{p, q, r\} \\ b \mapsto \{s, t\} \\ c \mapsto \{u, v\} \\ d \mapsto \{w\} \end{array} \right]$$

We expand R and S with Z according to the method in 20.1.6. So

$$R \parallel Z = R \odot \top_{Z \setminus X}$$

where

$$Z \setminus X = \left[\begin{array}{l} c \mapsto \{u, v\} \\ d \mapsto \{w\} \end{array} \right]$$

so that

$$R \parallel Z = \begin{array}{|c|c|} \hline a : \{p, q, r\} & b : \{s, t\} \\ \hline p & s \\ \hline q & t \\ \hline \end{array} \odot \begin{array}{|c|c|} \hline c : \{u, v\} & d : \{w\} \\ \hline u & w \\ \hline v & w \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|c|} \hline a : \{p, q, r\} & b : \{s, t\} & c : \{u, v\} & d : \{w\} \\ \hline p & s & u & w \\ \hline q & t & u & w \\ \hline p & s & v & w \\ \hline q & t & v & w \\ \hline \end{array}$$

With the same method for $S \parallel Z$ we obtain

$$S \parallel Z = S \odot \top_{Z \setminus Y}$$

$$= S \odot \top_{[a \mapsto \{p, q, r\}]}$$

$$= \begin{array}{|c|c|c|} \hline b : \{s, t\} & c : \{u, v\} & d : \{w\} \\ \hline s & u & w \\ \hline s & v & w \\ \hline t & v & w \\ \hline \end{array} \odot \begin{array}{|c|} \hline a : \{p, q, r\} \\ \hline p \\ \hline q \\ \hline r \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|c|} \hline a : \{p, q, r\} & b : \{s, t\} & c : \{u, v\} & d : \{w\} \\ \hline p & s & u & w \\ \hline q & s & u & w \\ \hline r & s & u & w \\ \hline p & s & v & w \\ \hline q & s & v & w \\ \hline r & s & v & w \\ \hline p & t & v & w \\ \hline q & t & v & w \\ \hline r & t & v & w \\ \hline \end{array}$$

The relations are made equi-schematic and we construct $R \sqcap S$ as the intersection of the rows in $R \parallel Z$ and $S \parallel Z$, i.e.

$$R \sqcap S = \begin{array}{|c|c|c|c|} \hline a : \{p, q, r\} & b : \{s, t\} & c : \{u, v\} & d : \{w\} \\ \hline p & s & u & w \\ \hline p & s & v & w \\ \hline q & t & v & w \\ \hline \end{array}$$

Accordingly, the union of rows gives us

$$R \sqcup S = \begin{array}{|c|c|c|c|} \hline a : \{p, q, r\} & b : \{s, t\} & c : \{u, v\} & d : \{w\} \\ \hline p & s & u & w \\ \hline q & s & u & w \\ \hline r & s & u & w \\ \hline q & t & u & w \\ \hline p & s & v & w \\ \hline q & s & v & w \\ \hline r & s & v & w \\ \hline p & t & v & w \\ \hline q & t & v & w \\ \hline r & t & v & w \\ \hline \end{array}$$

And comparing the graph tables of $R \parallel Z$ and $S \parallel Z$, we see that not every row of the first is a member of the second, so $R \not\sqsubseteq S$ and $R \neq S$.

20.3.3

Boolean table representation

We use the same example relations $R = [X, \Gamma]$ and $S = [Y, \Sigma]$ from 20.3.2 again, but this time in boolean table representation:

$$R = \begin{array}{|c|c|} \hline a & b \\ \hline p & s & \mathbf{1} \\ \hline q & s & \mathbf{0} \\ \hline r & s & \mathbf{0} \\ \hline p & t & \mathbf{0} \\ \hline q & t & \mathbf{1} \\ \hline r & t & \mathbf{0} \\ \hline \end{array} \quad S = \begin{array}{|c|c|c|} \hline b & c & d \\ \hline s & u & w & \mathbf{1} \\ \hline t & u & w & \mathbf{0} \\ \hline s & v & w & \mathbf{1} \\ \hline t & v & w & \mathbf{1} \\ \hline \end{array}$$

We join the schemas and again

$$Z := X \vee Y = \left[\begin{array}{l} a \mapsto \{p, q, r\} \\ b \mapsto \{s, t\} \\ c \mapsto \{u, v\} \\ d \mapsto \{w\} \end{array} \right]$$

According to the method in 20.1.5 for the double table construction of $R \parallel Z$, we have

$$\top_{Z \setminus X} = \begin{array}{|c|c|} \hline c & d \\ \hline u & w \\ \hline v & w \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline u & v & c \\ \hline w & w & d \\ \hline \end{array}$$

so that

$$R \parallel Z = R \odot \top_{Z \setminus X}$$

$$= \begin{array}{|c|c|c|} \hline a & b & \\ \hline p & s & \mathbf{1} \\ \hline q & s & \mathbf{0} \\ \hline r & s & \mathbf{0} \\ \hline p & t & \mathbf{0} \\ \hline q & t & \mathbf{1} \\ \hline r & t & \mathbf{0} \\ \hline \end{array} \odot \begin{array}{|c|c|c|} \hline u & v & c \\ \hline w & w & d \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|c|} \hline & & u & v & c \\ \hline & & w & w & d \\ \hline p & s & \mathbf{1} & \mathbf{1} & \\ \hline q & s & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline r & s & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline p & t & \mathbf{0} & \mathbf{0} & \\ \hline q & t & \mathbf{1} & \mathbf{1} & \\ \hline r & t & \mathbf{0} & \mathbf{0} & \\ \hline a & b & & & \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline p & s & u & w & \mathbf{1} \\ \hline q & s & u & w & \mathbf{0} \\ \hline r & s & u & w & \mathbf{0} \\ \hline p & t & u & w & \mathbf{0} \\ \hline q & t & u & w & \mathbf{1} \\ \hline r & t & u & w & \mathbf{0} \\ \hline p & s & v & w & \mathbf{1} \\ \hline q & s & v & w & \mathbf{0} \\ \hline r & s & v & w & \mathbf{0} \\ \hline p & t & v & w & \mathbf{0} \\ \hline q & t & v & w & \mathbf{1} \\ \hline r & t & v & w & \mathbf{0} \\ \hline \end{array}$$

We follow the same procedure for $S \parallel Z$:

$$Z \setminus Y = [a \mapsto \{p, q, r\}]$$

and

$$\top_{Z \setminus Y} = \begin{array}{|c|} \hline a \\ \hline p \\ \hline q \\ \hline r \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline p & q & r & a \\ \hline \end{array}$$

so that

$$S \parallel Z = S \odot \top_{Z \setminus Y}$$

$$= \begin{array}{|c|c|c|} \hline b & c & d \\ \hline s & u & w & \mathbf{1} \\ \hline t & u & w & \mathbf{0} \\ \hline s & v & w & \mathbf{1} \\ \hline t & v & w & \mathbf{1} \\ \hline \end{array} \odot \begin{array}{|c|c|c|c|} \hline p & q & r & a \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|c|} \hline & & p & q & r & a \\ \hline s & u & w & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hline t & u & w & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline s & v & w & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hline t & v & w & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hline b & c & d & & & \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline p & s & u & w & \mathbf{1} \\ \hline q & s & u & w & \mathbf{1} \\ \hline r & s & u & w & \mathbf{1} \\ \hline p & t & u & w & \mathbf{0} \\ \hline q & t & u & w & \mathbf{0} \\ \hline r & t & u & w & \mathbf{0} \\ \hline p & s & v & w & \mathbf{1} \\ \hline q & s & v & w & \mathbf{1} \\ \hline r & s & v & w & \mathbf{1} \\ \hline p & t & v & w & \mathbf{1} \\ \hline q & t & v & w & \mathbf{1} \\ \hline r & t & v & w & \mathbf{1} \\ \hline \end{array}$$

Finally, we obtain $R \sqcap S = (R \parallel Z) \cap (S \parallel Z)$, i.e.

$$R \sqcap S = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline p & s & u & w & \mathbf{1} \wedge \mathbf{1} \\ \hline q & s & u & w & \mathbf{0} \wedge \mathbf{1} \\ \hline r & s & u & w & \mathbf{0} \wedge \mathbf{1} \\ \hline p & t & u & w & \mathbf{0} \wedge \mathbf{0} \\ \hline q & t & u & w & \mathbf{1} \wedge \mathbf{0} \\ \hline r & t & u & w & \mathbf{0} \wedge \mathbf{0} \\ \hline p & s & v & w & \mathbf{1} \wedge \mathbf{1} \\ \hline q & s & v & w & \mathbf{0} \wedge \mathbf{1} \\ \hline r & s & v & w & \mathbf{0} \wedge \mathbf{1} \\ \hline p & t & v & w & \mathbf{0} \wedge \mathbf{1} \\ \hline q & t & v & w & \mathbf{1} \wedge \mathbf{1} \\ \hline r & t & v & w & \mathbf{0} \wedge \mathbf{1} \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline p & s & u & w & \mathbf{1} \\ \hline q & s & u & w & \mathbf{0} \\ \hline r & s & u & w & \mathbf{0} \\ \hline p & t & u & w & \mathbf{0} \\ \hline q & t & u & w & \mathbf{0} \\ \hline r & t & u & w & \mathbf{0} \\ \hline p & s & v & w & \mathbf{1} \\ \hline q & s & v & w & \mathbf{0} \\ \hline r & s & v & w & \mathbf{0} \\ \hline p & t & v & w & \mathbf{0} \\ \hline q & t & v & w & \mathbf{1} \\ \hline r & t & v & w & \mathbf{0} \\ \hline \end{array}$$

and

$$R \sqcup S = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline p & s & u & w & \mathbf{1} \vee \mathbf{1} \\ \hline q & s & u & w & \mathbf{0} \vee \mathbf{1} \\ \hline r & s & u & w & \mathbf{0} \vee \mathbf{1} \\ \hline p & t & u & w & \mathbf{0} \vee \mathbf{0} \\ \hline q & t & u & w & \mathbf{1} \vee \mathbf{0} \\ \hline r & t & u & w & \mathbf{0} \vee \mathbf{0} \\ \hline p & s & v & w & \mathbf{1} \vee \mathbf{1} \\ \hline q & s & v & w & \mathbf{0} \vee \mathbf{1} \\ \hline r & s & v & w & \mathbf{0} \vee \mathbf{1} \\ \hline p & t & v & w & \mathbf{0} \vee \mathbf{1} \\ \hline q & t & v & w & \mathbf{1} \vee \mathbf{1} \\ \hline r & t & v & w & \mathbf{0} \vee \mathbf{1} \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline p & s & u & w & \mathbf{1} \\ \hline q & s & u & w & \mathbf{1} \\ \hline r & s & u & w & \mathbf{1} \\ \hline p & t & u & w & \mathbf{0} \\ \hline q & t & u & w & \mathbf{1} \\ \hline r & t & u & w & \mathbf{0} \\ \hline p & s & v & w & \mathbf{1} \\ \hline q & s & v & w & \mathbf{1} \\ \hline r & s & v & w & \mathbf{1} \\ \hline p & t & v & w & \mathbf{1} \\ \hline q & t & v & w & \mathbf{1} \\ \hline r & t & v & w & \mathbf{1} \\ \hline \end{array}$$

and we can see that $R \not\sqsubseteq S$, because $(R \parallel Z) \not\sqsubseteq (S \parallel Z)$, which is evident from their graph table comparison:

$$R \parallel Z = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline p & s & u & w & \mathbf{1} \\ \hline q & s & u & w & \mathbf{0} \\ \hline r & s & u & w & \mathbf{0} \\ \hline p & t & u & w & \mathbf{0} \\ \hline q & t & u & w & \mathbf{1} \\ \hline r & t & u & w & \mathbf{0} \\ \hline p & s & v & w & \mathbf{1} \\ \hline q & s & v & w & \mathbf{0} \\ \hline r & s & v & w & \mathbf{0} \\ \hline p & t & v & w & \mathbf{0} \\ \hline q & t & v & w & \mathbf{1} \\ \hline r & t & v & w & \mathbf{0} \\ \hline \end{array} \not\sqsubseteq \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline p & s & u & w & \mathbf{1} \\ \hline q & s & u & w & \mathbf{1} \\ \hline r & s & u & w & \mathbf{1} \\ \hline p & t & u & w & \mathbf{0} \\ \hline q & t & u & w & \mathbf{0} \\ \hline r & t & u & w & \mathbf{0} \\ \hline p & s & v & w & \mathbf{1} \\ \hline q & s & v & w & \mathbf{1} \\ \hline r & s & v & w & \mathbf{1} \\ \hline p & t & v & w & \mathbf{1} \\ \hline q & t & v & w & \mathbf{1} \\ \hline r & t & v & w & \mathbf{1} \\ \hline \end{array} = S \parallel Z$$

There is one row, where the left table has a 1 and the right one has a 0, but $\mathbf{1} \not\leq \mathbf{0}$.

20.4 Properties of semantic operations

20.4.1 Lemma

For two distinct relations R and S holds

$$R \odot S = R \sqcap S$$

For every pairwise distinct relation class \mathcal{R} holds

$$\odot \mathcal{R} = \prod \mathcal{R}$$

20.4.2 Proof of 20.4.1

First of all let us note that conjunctions are defined for compatible relations and distinct products for (pairwise) distinct relations. This is a proper generalization, because distinct records are compatible, according to 19.1.8(1).

Since $R \odot S = \odot \{R, S\}$, we only need the proof for $\odot \mathcal{R} = \prod \mathcal{R}$.

So suppose $\mathcal{R} = \{\mathcal{R}_k \mid k \in K\}$ with $\mathcal{R}_k = [X_k, \Gamma_k]$ for each $k \in K$. We put $X := \bigvee_{k \in K} X_k$ and

$$\Gamma_k^l := \begin{cases} \Gamma_k & \text{if } l = k \\ \otimes X_l & \text{if } l \neq k \end{cases} \quad \text{for all } k, l \in L$$

Then, for each $k \in K$, the graph record $\{\Gamma_k^l \mid l \in L\}$ is relatively distinct with a well-defined distinct product

$$\bigcirc_{l \in K} \Gamma_k^l = \Gamma_k \circ \left(\bigcirc_{l \in K \setminus \{k\}} \otimes X_l \right) = \Gamma_k \circ \otimes (X \setminus X_k)$$

according to 16.6.5(2). And since $\Gamma_k^l \subseteq \otimes X_l$ for all $k, l \in K$, we obtain

$$\bigcap_{k \in K} \Gamma_k^l = \Gamma_l$$

for each $l \in L$. The distributivity law of 16.6.1 gives us

$$\bigcap_{k \in K} \left(\bigcirc_{l \in K} \Gamma_k^l \right) = \bigcirc_{l \in K} \left(\bigcap_{k \in K} \Gamma_k^l \right) = \bigcirc_{l \in K} \Gamma_l$$

We put things together and obtain

$$\begin{aligned} & \prod \mathcal{R} \\ &= \bigcap_{k \in K} (\mathcal{R}_k \parallel X) && \text{def. 20.2.2 of } \prod \\ &= \bigcap_{k \in K} (\mathcal{R}_k \circ \top_{(X \setminus X_k)}) && \text{due to 20.1.3(3)} \\ &= \bigcap_{k \in K} \left[\begin{array}{c} X_k \dot{\vee} (X \setminus X_k) \\ \Gamma_k \circ \otimes (X \setminus X_k) \end{array} \right] && \text{due to 20.1.3(2)} \\ &= \bigcap_{k \in K} \left[\begin{array}{c} X \\ \bigcirc_{l \in K} \Gamma_k^l \end{array} \right] \\ &= \left[\begin{array}{c} X \\ \bigcap_{k \in K} \left(\bigcirc_{l \in K} \Gamma_k^l \right) \end{array} \right] && \text{def. 19.5.3 of } \bigcap \\ &= \left[\begin{array}{c} X \\ \bigcirc_{k \in K} \Gamma_k \end{array} \right] \\ &= \circ \mathcal{R} \end{aligned}$$

20.4.3 Remark _____generalization_____

Note, that $\mathbf{Rel}(X) \subseteq \mathbf{Prel}(X)$, for every schema X .

And in a certain sense, the algebra on $\mathbf{Rel}(X)$

$$(\mathbf{Rel}(X), \subseteq, \perp_X, \top_X, \cap, \cup, \prod, \prod, \neg)$$

is a kind of subalgebra or “quasi-subalgebra” of the structure

$$(\mathbf{Prel}(X), \subseteq, \perp, \top, \cap, \cup, \prod, \prod, \neg)$$

“Quasi” means “up to equivalence”. The precise details of the resemblance between these two structures is given in the following lemma.

20.4.4 Lemma _____generalization_____

Let X be a schema. For all $R, S \in \mathbf{Rel}(X)$ and every $\mathcal{R} \subseteq \mathbf{Rel}(X)$ holds:

- (1) $\perp_X \equiv \perp$
- (2) $\top_X \equiv \top$
- (3) $R \subseteq S$ iff $R \sqsubseteq S$
- (4) $R = S$ iff $R \equiv S$
- (5) $R \cap S = R \sqcap S$
- (6) $R \cup S = R \sqcup S$
- (7) $\prod \mathcal{R} \equiv \prod \mathcal{R}$
- (8) $\bigcup \mathcal{R} \equiv \prod \mathcal{R}$
- (9) $\prod \mathcal{R} = \prod \mathcal{R}$ iff $(\mathcal{R} \neq \emptyset \text{ or } X = \langle \rangle)$
- (10) $\bigcup \mathcal{R} = \prod \mathcal{R}$ iff $(\mathcal{R} \neq \emptyset \text{ or } X = \langle \rangle)$

20.4.5 Proof of 20.4.4 _____

(1) $\perp_X \parallel X = \perp \parallel X$, so $\perp_X \equiv \perp$, and the same holds for (2) $\top_X \equiv \top$.

For $R, S \in \mathbf{Rel}(X)$, there is $\mathbf{x}(R) \vee \mathbf{x}(S) = X \vee X = X$, so $R \parallel X = R$ and $S \parallel X = S$. Therefore, (3) $R \subseteq S$ iff $R \sqsubseteq S$, (4) $R = S$ iff $R \equiv S$, (5) $R \cap S = R \sqcap S$, and (6) $R \cup S = R \sqcup S$.

For $\mathcal{R} \subseteq \mathbf{Rel}(X)$, there is

$$\bigvee \{\mathbf{x}(R) \mid R \in \mathcal{R}\} = \begin{cases} X & \text{if } \mathcal{R} \neq \emptyset \\ \langle \rangle & \text{if } \mathcal{R} = \emptyset \end{cases}$$

- (a) If $\mathcal{R} \neq \emptyset$, then $\prod \mathcal{R} = \bigcap \{(R \parallel Z) \mid R \in \mathcal{R}\} = \bigcap \mathcal{R}$.
 - (b) If $\mathcal{R} = \emptyset$ and $X = \langle \rangle$, then $\prod \mathcal{R} = \top = \top_X = \bigcap \mathcal{R}$.
 - (c) If $\mathcal{R} = \emptyset$ and $X \neq \langle \rangle$, then $\prod \mathcal{R} = \top \neq \top_X = \bigcap \mathcal{R}$, but still $\prod \mathcal{R} \equiv \bigcap \mathcal{R}$, due to (2).
- So, (a), (b), (c) together proof (7) and (9).
(8) and (10) are proved similarly.

20.4.6 Remark _____accumulation_____

As a general rule, we can say that: junctions accumulate the according schemas and attribute classes. The following lemma 20.4.7 is a more precise version of this rule.

20.4.7 Lemma _____accumulation_____

Let X be a schema. Let $R, S, T \in \mathbf{Prel}(X)$ and $\mathcal{R} \subseteq \mathbf{Prel}(X)$.

- (1) $\mathbf{x}(\perp) = \langle \rangle$
- (2) $\mathbf{x}(\top) = \langle \rangle$
- (3) $\mathbf{x}(\neg R) = \mathbf{x}(R)$
- (4) $\mathbf{x}(R \cap S) = \mathbf{x}(R) \vee \mathbf{x}(S)$
- (5) $\mathbf{x}(R \cup S) = \mathbf{x}(R) \vee \mathbf{x}(S)$
- (6) $\mathbf{x}(\prod \mathcal{R}) = \bigvee \{\mathbf{x}(R) \mid R \in \mathcal{R}\}$
- (7) $\mathbf{x}(\bigcup \mathcal{R}) = \bigvee \{\mathbf{x}(R) \mid R \in \mathcal{R}\}$

20.4.8 Proof of 20.4.7 _____

All these statements are immediate consequences of the according operation definition.

20.4.9 Lemma equivalence

- (1) For every two compatible relations R and S ,
- $$R \equiv S \text{ iff } R \sqsubseteq S \text{ and } S \sqsubseteq R$$
- (2) If $R = [Y, \Gamma]$ is a relation and Z a schema with $Z \sim Y$, then
- $$R \equiv R \parallel Z$$
- (3) Let Y and Z be two schemas, such that Z is proper and $Y \not\sqsubseteq Z$. For all $R, S \in \mathbf{Prel}(Y)$ holds:
- (a) $R \sqsubseteq S$ iff $R \parallel Z \sqsubseteq S \parallel Z$
- (b) $R = S$ iff $R \parallel Z = S \parallel Z$
- (4) For every proper schema X and all $R, S \in \mathbf{Prel}(X)$
- (a) $R \sqsubseteq S$ iff $R \parallel X \sqsubseteq S \parallel X$
- (b) $R \equiv S$ iff $R \parallel X = S \parallel X$
- (5) For every schema X and all $R, S \in \mathbf{Prel}(X)$:
- $$R = S \text{ iff } R \equiv S \text{ and } R \hat{=} S$$

20.4.10 Proof of 20.4.9

- (1) Let $Z = \mathbf{x}(R) \vee \mathbf{x}(S)$. There is $R \equiv S$ iff $R \parallel Z = S \parallel Z$ iff $(R \parallel Z \sqsubseteq S \parallel Z \text{ and } R \parallel Z \sqsubseteq S \parallel Z)$ iff $(R \sqsubseteq S \text{ and } S \sqsubseteq R)$.
- (2) $\mathbf{x}(R) \vee \mathbf{x}(R \parallel Z) = Y \vee Z$, so $R \equiv R \parallel Z$ iff $R \parallel Y \vee Z = (R \parallel Z) \parallel Y \vee Z$. According to lemma 20.1.11(1), $(R \parallel Z) \parallel Y \vee Z = R \parallel (Z \vee (Y \vee Z)) = R \parallel Y \vee Z$. Thus $R \equiv R \parallel Z$.
- (3) Note, that distinct schemas are compatible, so $R \parallel Z$ and $S \parallel Z$ are well-defined. Z is a proper schema, so $\otimes Z \neq \emptyset$, according to 12.1.12(1). Let $R = [Y, \Gamma]$ and $S = [Y, \Sigma]$, then

$$R \parallel Z = [Y \vee Z, \{y' \vee z \mid y' \in \Gamma, z \in \otimes Z\}]$$

$$S \parallel Z = [Y \vee Z, \{y'' \vee z \mid y'' \in \Sigma, z \in \otimes Z\}]$$

For the proof of (a) we distinguish two cases:

- ♣ Suppose, $R \subseteq S$. Then $\Gamma \subseteq \Sigma$ and thus $\{y' \vee z \mid y' \in \Gamma, z \in \otimes Z\} \subseteq \{y'' \vee z \mid y'' \in \Sigma, z \in \otimes Z\}$ and $R \parallel Z \subseteq S \parallel Z$.
- ♣ Suppose, $R \not\subseteq S$. Then $\Gamma \not\subseteq \Sigma$, and because $\otimes Z \neq \emptyset$, $\{y' \vee z \mid y' \in \Gamma, z \in \otimes Z\} \not\subseteq \{y'' \vee z \mid y'' \in \Sigma, z \in \otimes Z\}$ and thus $R \parallel Z \not\subseteq S \parallel Z$.

Thus $R \subseteq S$ iff $R \parallel Z \subseteq S \parallel Z$.

(b) can be proved similarly with “=” instead of “ \subseteq ”, but it is also an immediate consequence of (a) and (1).

- (4) (a) Let $Y := \mathbf{x}(R) \vee \mathbf{x}(S)$ and $Z := X \setminus Y$. Then $Y \not\sqsubseteq Z$, and Z is proper, because X is proper. We derive

$$\begin{aligned} R &\sqsubseteq S \\ \Leftrightarrow R \parallel Y &\subseteq S \parallel Y && \text{def. 20.2.2} \\ \Leftrightarrow (R \parallel Y) \parallel Z &\subseteq (S \parallel Y) \parallel Z && \text{due to (3)(a)} \\ \Leftrightarrow R \parallel Y \vee Z &\subseteq S \parallel Y \vee Z && \text{due to 20.1.3(1)} \\ \Leftrightarrow R \parallel X &\subseteq S \parallel X \end{aligned}$$

(b) can be proved similarly, but it is also an immediate consequence of (a) and (1).

- (5) Let $R = [Y, \Gamma]$ and $S = [Z, \Sigma]$.

Suppose, $R = S$. Then $Y = Z$ and $\Gamma = \Sigma$, and thus $R \hat{=} S$ and $R \equiv S$.

On the other hand, suppose, $R \hat{=} S$ and $R \equiv S$. $R \hat{=} S$ means $Y = Z$. Therefore $R \parallel Z = R$ and $S \parallel Y = S$. So $R \equiv S$ implies $R = S$.

20.4.11 Repetition quasi-ordered class

A structure $\langle Q, \sqsubseteq, \equiv \rangle$ is a quasi-ordered class, if

- (1) Q is a class
- (2) \sqsubseteq is a quasi-order on Q in the sense that
- (a) $a \sqsubseteq b$ and $b \sqsubseteq c$ implies $a \sqsubseteq c$, for all $a, b, c \in Q$ (transitivity)
- (b) $a \sqsubseteq a$, for all $a \in Q$ (reflexivity)
- (3) \equiv is the equivalence (relation) of \sqsubseteq in the sense that
- $$a \equiv b \text{ iff } (a \sqsubseteq b \text{ and } b \sqsubseteq a) \text{ for all } a, b \in Q$$

In that case, \equiv is indeed an equivalence relation (i.e. transitive, reflexive and symmetric).

Because \equiv is given by \sqsubseteq through (3), a quasi-ordered class is more often simply given by $\langle Q, \sqsubseteq \rangle$ only.

20.4.12 Lemma quasi-order

$\langle \mathbf{Prel}(X), \sqsubseteq, \equiv \rangle$ is a quasi-ordered class, for every proper schema X .

20.4.13 Proof of 20.4.12

We need to show that, for a given schema X , $\langle \mathbf{Prel}(X), \sqsubseteq, \equiv \rangle$ satisfies the properties of 20.4.11.

Well, \sqsubseteq certainly is reflexive. And by applying 20.4.9 we have

$$\begin{aligned} R \sqsubseteq S \text{ and } S \sqsubseteq T \\ \text{implies } R \parallel X \subseteq S \parallel X \text{ and } S \parallel X \subseteq T \subseteq X \\ \text{implies } R \parallel X \subseteq T \parallel X \\ \text{implies } R \sqsubseteq T \end{aligned}$$

for all $R, S, T \in \mathbf{Prel}(X)$, i.e. \sqsubseteq is transitive, too.

And from 20.4.9(1), we already know that \equiv is the equivalence relation of \sqsubseteq .

20.4.14 Remark Remark

It is important to note, that $\langle \mathbf{Prel}(X), \sqsubseteq, \equiv \rangle$ is usually not a quasi-ordered class anymore, if the schema X is not proper.

Suppose, X is not proper, say

$$X = \left[\begin{array}{l} a \mapsto \{p, q\} \\ b \mapsto \emptyset \\ c \mapsto \{r, s\} \end{array} \right]$$

and $R, S, T \in \mathbf{Prel}(X)$ are given by

$$R = \frac{a : \{p, q\}}{\frac{p}{q}} \quad S = \frac{b : \emptyset}{\quad} \quad T = \frac{c : \{r, s\}}{r}$$

then

$$R \parallel \mathbf{x}(S) = \frac{a : \{p, q\} \quad b : \emptyset}{\quad} = S \parallel \mathbf{x}(R)$$

$$T \parallel \mathbf{x}(S) = \boxed{b : \emptyset \mid c : \{r, s\}} = S \parallel \mathbf{x}(T)$$

$$R \parallel \mathbf{x}(T) = \begin{array}{|c|c|} \hline a : & c : \\ \hline \{p, q\} & \{r, s\} \\ \hline p & r \\ \hline q & r \\ \hline p & s \\ \hline q & s \\ \hline \end{array} \neq \begin{array}{|c|c|} \hline a : & c : \\ \hline \{p, q\} & \{r, s\} \\ \hline p & r \\ \hline q & r \\ \hline \end{array} = T \parallel \mathbf{x}(R)$$

and so

$$R \equiv S \text{ and } S \equiv T \text{ but } R \not\equiv T$$

The transitivity of \equiv is violated and \equiv is not an equivalence relation anymore. Neither is \sqsubseteq a quasi-order on this example $\mathbf{Prel}(X)$.

20.4.15 Lemma —distributivity of expansion over boolean junctions—

Let X be a proper schema and $Y \in \mathbf{Proj}(X)$.

- (1) $\perp \parallel Y \equiv \perp$
- (2) $\top \parallel Y \equiv \top$
- (3) $(\neg R) \parallel Y = \neg(R \parallel Y)$ for every $R \in \mathbf{Prel}(X)$
- (4) $(R \sqcap S) \parallel Y = (R \parallel Y) \sqcap (S \parallel Y)$ for all $R, S \in \mathbf{Prel}(X)$
- (5) $(R \sqcup S) \parallel Y = (R \parallel Y) \sqcup (S \parallel Y)$ for all $R, S \in \mathbf{Prel}(X)$
- (6) $R \sqsubseteq S$ iff $R \parallel Y \sqsubseteq S \parallel Y$ for all $R, S \in \mathbf{Prel}(X)$
- (7) $R \equiv S$ iff $R \parallel Y \equiv S \parallel Y$ for all $R, S \in \mathbf{Prel}(X)$
- (8) For every $\mathcal{R} \subseteq \mathbf{Prel}(X)$

$$(\prod \mathcal{R}) \parallel Y \equiv \prod_{R \in \mathcal{R}} (R \parallel Y)$$

$$(\prod \mathcal{R}) \parallel Y = \prod_{R \in \mathcal{R}} (R \parallel Y) \text{ iff } (\mathcal{R} \neq \emptyset \text{ or } Y = \langle \rangle)$$

- (9) For every $\mathcal{R} \subseteq \mathbf{Prel}(X)$

$$(\prod \mathcal{R}) \parallel Y \equiv \prod_{R \in \mathcal{R}} (R \parallel Y)$$

$$(\prod \mathcal{R}) \parallel Y = \prod_{R \in \mathcal{R}} (R \parallel Y) \text{ iff } (\mathcal{R} \neq \emptyset \text{ or } Y = \langle \rangle)$$

20.4.16 Proof of 20.4.15

First of all, let us note that $U \smile Y$, for every $R = [U, \Gamma] \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$. So all operations, in particular all expansions, occurring in this lemma, are well-defined.

- (1) $\perp = [\langle \rangle, \emptyset]$, $\langle \rangle \smile Y$, and so $\perp \parallel Y \equiv \perp$, according to 20.4.9(2).

- (2) similar to (1)

- (3) If $R = [U, \Gamma]$, we put $Y' := Y \setminus U$ and obtain

$$\begin{aligned} & (\neg R) \parallel Y \\ &= \left[\begin{array}{c} U \\ \otimes U \setminus \Gamma \end{array} \right] \parallel Y \quad \text{def. 19.4.1 of } \neg \\ &= \left[\begin{array}{c} U \\ \otimes U \setminus \Gamma \end{array} \right] \parallel Y' \quad \text{due to 20.1.3(1)} \\ &= \left[\begin{array}{c} U \dot{\vee} Y' \\ (\otimes U \setminus \Gamma) \odot \otimes Y' \end{array} \right] \quad \text{due to 20.1.3(2)} \end{aligned}$$

$$= \left[\begin{array}{c} U \dot{\vee} Y' \\ (\otimes U \odot \otimes Y') \setminus (\Gamma \odot \otimes Y') \end{array} \right] \quad \text{due to 16.6.1(3)}$$

$$= \left[\begin{array}{c} U \dot{\vee} Y' \\ (\otimes(U \dot{\vee} Y')) \setminus (\Gamma \odot \otimes Y') \end{array} \right] \quad \text{due to 16.6.5(4)}$$

$$= \neg \left[\begin{array}{c} U \dot{\vee} Y' \\ \Gamma \odot \otimes Y' \end{array} \right] \quad \text{due to 19.4.1 again}$$

$$= \neg \left(\left[\begin{array}{c} U \\ \Gamma \end{array} \right] \parallel Y' \right) \quad \text{due to 20.1.3(2)}$$

$$= \neg(R \parallel Y') \quad \text{definition of } R$$

$$= \neg(R \parallel Y) \quad \text{due to 20.1.3(1)}$$

We now first proof (8) and (9), which turn out to be generalizations of (4) and (5), respectively.

- (8) Let $\mathcal{R} = \{\mathcal{R}_i \mid i \in I\} \subseteq \mathbf{Prel}(X)$ with $\mathcal{R}_i = [X_i, \Gamma_i]$ for $i \in I$. We put

$$Z := \bigvee \{X_i \mid i \in I\}$$

$$Y' := Y \setminus Z$$

$$Z_i := Z \setminus X_i \quad \text{for each } i \in I$$

$$Y_i := Y \setminus X_i \quad \text{for each } i \in I$$

$$Z'_i := Z \setminus (Y \vee X_i) \quad \text{for each } i \in I$$

so that

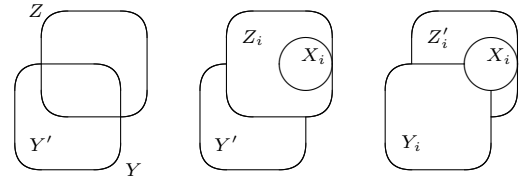
$$(a) \quad Z \vee Y = Z \dot{\vee} Y'$$

$$(b) \quad Z \vee Y = Y_i \dot{\vee} Z'_i \dot{\vee} X_i \quad \text{for each } i \in I$$

$$(c) \quad Z \vee Y = Y'_i \dot{\vee} Z_i \dot{\vee} X_i \quad \text{for each } i \in I$$

$$(d) \quad \Gamma_i \odot (\otimes Y_i) \odot (\otimes Z'_i) = \Gamma_i \odot (\otimes Y'_i) \odot (\otimes Z_i) \quad \text{for each } i \in I$$

as the following diagrams illustrate:



First, let us suppose that $\mathcal{R} \neq \emptyset$, i.e. $I \neq \emptyset$, then

$$\begin{aligned} & (\prod \mathcal{R}) \parallel Y \\ &= \left(\bigcap_{i \in I} (\mathcal{R}_i \parallel Z) \right) \parallel Y \quad \text{def. 20.2.4 of } \prod \end{aligned}$$

$$= \left(\bigcap_{i \in I} (\mathcal{R}_i \parallel Z) \right) \parallel Y' \quad \text{due to 20.1.3(1)}$$

$$= \left(\bigcap_{i \in I} (\mathcal{R}_i \parallel Z_i) \right) \parallel Y' \quad \text{again 20.1.3(1)}$$

$$= \left(\bigcap_{i \in I} \left[\begin{array}{c} X_i \dot{\vee} Z_i \\ \Gamma_i \odot (\otimes Z_i) \end{array} \right] \right) \parallel Y' \quad \text{20.1.3(2)}$$

$$= \left[\begin{array}{c} Z \\ \bigcap_{i \in I} (\Gamma_i \odot (\otimes Z_i)) \end{array} \right] \parallel Y' \quad \text{def. 19.5.3 of } \prod$$

$$\begin{aligned}
&= \left[\begin{array}{c} Z \dot{\vee} Y' \\ \left(\bigcap_{i \in I} (\Gamma_i \odot (\otimes Z_i)) \right) \odot (\otimes Y') \end{array} \right] && \text{due to 20.1.3(2)} \\
&= \left[\begin{array}{c} Z \dot{\vee} Y' \\ \bigcap_{i \in I} (\Gamma_i \odot (\otimes Z_i) \odot (\otimes Y')) \end{array} \right] && \text{due to 16.6.1(5)} \\
&= \left[\begin{array}{c} Z \dot{\vee} Y' \\ \bigcap_{i \in I} (\Gamma_i \odot (\otimes Y_i) \odot (\otimes Z'_i)) \end{array} \right] && \text{due to (d)} \\
&= \bigcap_{i \in I} \left[\begin{array}{c} Z \dot{\vee} Y' \\ (\Gamma_i \odot (\otimes Y_i) \odot (\otimes Z'_i)) \end{array} \right] && \text{due to 19.5.3} \\
&= \bigcap_{i \in I} \left[\begin{array}{c} (X_i \dot{\vee} Y_i) \dot{\vee} Z'_i \\ ((\Gamma_i \odot (\otimes Y_i)) \odot (\otimes Z'_i)) \end{array} \right] && \text{due to (b)} \\
&= \bigcap_{i \in I} \left(\left[\begin{array}{c} (X_i \dot{\vee} Y_i) \\ ((\Gamma_i \odot (\otimes Y_i))) \end{array} \right] \right) \parallel Z'_i && \text{due to 20.1.3(2)} \\
&= \bigcap_{i \in I} (\mathcal{R}_i \parallel Y_i) \parallel Z'_i && \text{due to 20.1.3(2) again}
\end{aligned}$$

On the other hand, if $\mathcal{R} = \emptyset$, then

$$\begin{aligned}
&(\prod \mathcal{R}) \parallel Y \\
&= \top \parallel Y && \text{definition 20.2.4 of } \prod \\
&\equiv \top && \text{due to (2)} \\
&= \prod \emptyset && \text{again, def. 20.2.4 of } \prod \\
&= \prod_{i \in \emptyset} (\mathcal{R}_i \parallel Y)
\end{aligned}$$

And obviously, this equivalence

$$(\prod \mathcal{R}) \parallel Y \equiv \prod \{R \parallel Y \mid R \in \mathcal{R}\}$$

turns into the identity

$$(\prod \mathcal{R}) \parallel Y = \prod \{R \parallel Y \mid R \in \mathcal{R}\}$$

exactly in case $Y = \langle \rangle$.

So putting things together we have, for every $\mathcal{R} \subseteq \mathbf{Prel}(X)$,

$$(\prod \mathcal{R}) \parallel Y \equiv \prod_{R \in \mathcal{R}} (R \parallel Y)$$

and

$$(\prod \mathcal{R}) \parallel Y = \prod_{R \in \mathcal{R}} (R \parallel Y) \quad \text{iff } (\mathcal{R} \neq \emptyset \text{ or } Y = \langle \rangle)$$

(9) Proof similar to (8). The central step in (8) is the application of the distributivity lemma 16.6.1(5) for distinct products over intersections. A similar statement 16.6.1(4) is true for unions (even including the empty class case). The rest of the proof is very much the same.

(4) We put $Z := \mathbf{x}(R) \vee \mathbf{x}(S)$ and obtain

$$\begin{aligned}
R \sqcap S &= (R \parallel Z) \cap (S \parallel Z) \\
&= \cap \{(R \parallel Z), (S \parallel Z)\} \\
&= \prod \{R, S\}
\end{aligned}$$

so (3) is just a special case of (8) with $\mathcal{R} = \{R, S\}$.

(5) Similar to (5).

(6) For arbitrary $R, S \in \mathbf{Prel}(X)$ holds $R \equiv R \parallel Y$, due to 20.4.9(2), and thus $R \sqsubseteq R \parallel Y$ and $R \parallel Y \sqsubseteq R$, due to 20.4.11(3). So, $R \sqsubseteq S$ implies $R \parallel Y \sqsubseteq R \sqsubseteq S \sqsubseteq S \parallel Y$ implies $R \parallel Y \sqsubseteq S \parallel Y$, since \sqsubseteq is transitive. On the other hand, $R \parallel Y \sqsubseteq S \parallel Y$ implies $R \sqsubseteq R \parallel Y \sqsubseteq S \parallel Y \sqsubseteq S$ implies $R \sqsubseteq S$. Thus $R \sqsubseteq S$ iff $R \parallel Y \sqsubseteq S \parallel Y$.

(7) Follows from (6) with 20.4.11(3).

20.4.17 Lemma

For every schema Y holds:

$$(1) \perp \parallel Y = \perp_Y$$

$$(2) \top \parallel Y = \top_Y$$

And if $R = [X, \Gamma]$ is a relation with $Y \sim X$, then

$$(3) R \parallel Y = R \sqcap \top_Y$$

$$(4) R \parallel Y = R \sqcup \perp_Y$$

20.4.18 Proof of 20.4.17

(1) We obtain

$$\perp \parallel Y = \left[\begin{array}{c} \langle \rangle \\ \emptyset \end{array} \right] \parallel Y = \left[\begin{array}{c} \langle \rangle \dot{\vee} Y \\ \emptyset \odot \otimes Y \end{array} \right] = \left[\begin{array}{c} Y \\ \emptyset \end{array} \right] = \perp_Y$$

by applying 20.1.3 and definition 19.3.1 of \perp and \perp_Y .

(2) Similar to (1) we obtain

$$\top \parallel Y = \left[\begin{array}{c} \langle \rangle \\ \{\langle \rangle\} \end{array} \right] \parallel Y = \left[\begin{array}{c} \langle \rangle \dot{\vee} Y \\ \{\langle \rangle\} \odot \otimes Y \end{array} \right] = \left[\begin{array}{c} Y \\ \otimes Y \end{array} \right] = \top_Y$$

(3) We obtain

$$\begin{aligned}
&R \sqcap \top_Y \\
&= (R \parallel (X \vee Y)) \cap (\top_Y \parallel (X \vee Y)) && \text{def. ?? of } \sqcap \\
&= (R \parallel (Y \setminus X)) \cap (\top_Y \parallel (X \setminus Y)) && \text{due to 20.1.3(1)} \\
&= \left(\left[\begin{array}{c} X \\ \Gamma \end{array} \right] \parallel (Y \setminus X) \right) \cap \left(\left[\begin{array}{c} Y \\ \otimes Y \end{array} \right] \parallel (X \setminus Y) \right) \\
& && \text{def. of } R \text{ and } \top_Y
\end{aligned}$$

$$= \left[\begin{array}{c} X \dot{\vee} (Y \setminus X) \\ \Gamma \odot \otimes (Y \setminus X) \end{array} \right] \cap \left[\begin{array}{c} Y \dot{\vee} (X \setminus Y) \\ \otimes Y \odot \otimes (X \setminus Y) \end{array} \right]$$

due to 20.1.3(2)

$$= \left[\begin{array}{c} X \dot{\vee} (Y \setminus X) \\ \Gamma \odot \otimes (Y \setminus X) \end{array} \right] \cap \left[\begin{array}{c} Y \dot{\vee} (X \setminus Y) \\ \otimes (Y \dot{\vee} (X \setminus Y)) \end{array} \right]$$

due to 16.6.5(4)

$$= \left[\begin{array}{c} X \vee Y \\ \Gamma \odot \otimes (Y \setminus X) \end{array} \right] \cap \left[\begin{array}{c} X \vee Y \\ \otimes (X \vee Y) \end{array} \right]$$

$$= \left[\begin{array}{c} X \vee Y \\ (\Gamma \odot \otimes (Y \setminus X)) \cap \otimes (X \vee Y) \end{array} \right]$$

def. 19.5.1 of \cap

$$= \left[\begin{array}{c} X \vee Y \\ \Gamma \odot \otimes (Y \setminus X) \end{array} \right]$$

because $(\Gamma \odot \otimes (Y \setminus X)) \sqsubseteq \otimes (X \vee Y)$

$$= R \parallel (Y \setminus X)$$

due to 20.1.3(2), again

$$= R \parallel Y$$

due to 20.1.3(1), again

(4) The proof is similar to (3) and we may skip some steps

$$\begin{aligned}
&R \sqcup \perp_Y \\
&= \left(\left[\begin{array}{c} X \\ \Gamma \end{array} \right] \parallel (Y \setminus X) \right) \cup \left(\left[\begin{array}{c} Y \\ \emptyset \end{array} \right] \parallel (X \setminus Y) \right)
\end{aligned}$$

$$\begin{aligned}
&= \left[\begin{array}{c} X \dot{\vee} (Y \setminus X) \\ \Gamma \odot \otimes (Y \setminus X) \end{array} \right] \cup \left[\begin{array}{c} Y \dot{\vee} (X \setminus Y) \\ \emptyset \odot \otimes (X \setminus Y) \end{array} \right] \\
&\quad \text{due to 20.1.3(2)} \\
&= \left[\begin{array}{c} X \vee Y \\ \Gamma \odot \otimes (Y \setminus X) \end{array} \right] \cup \left[\begin{array}{c} Y \vee X \\ \emptyset \end{array} \right] \\
&\quad \text{due to 16.4.3(3)} \\
&= \left[\begin{array}{c} X \vee Y \\ \Gamma \odot \otimes (Y \setminus X) \end{array} \right] \\
&= R \parallel Y
\end{aligned}$$

20.4.19 Lemma

Let X be a proper schema. In $\langle \mathbf{Prel}(X), \sqsubseteq \rangle$ holds for every $\mathcal{R} \subseteq \mathbf{Prel}(X)$ and all $R, S \in \mathbf{Prel}(X)$

- (1) $\prod \mathcal{R}$ is a greatest lower bound of \mathcal{R}
- (2) $\coprod \mathcal{R}$ is a least upper bound of \mathcal{R}
- (3) $R \sqcap S$ is a greatest lower bound of R and S
- (4) $R \sqcup S$ is a least upper bound of R and S

20.4.20 Proof of 20.4.19

- (1) Let $\mathcal{R} \in \mathbf{P}(\mathbf{Prel}(X))$. Then

- ♣ $\prod \mathcal{R}$ is a *lower bound* of \mathcal{R} :
According to 19.1.10, \mathcal{R} is compatible and $Y := \bigvee \{\mathbf{x}(R) \mid R \in \mathcal{R}\}$ is well defined. If $S \in \mathcal{R}$, then $\prod \mathcal{R} = \bigcap \{R \parallel Y \mid R \in \mathcal{R}\} \subseteq S \parallel Y$, so $\prod \mathcal{R} \subseteq S$.
- ♣ $\prod \mathcal{R}$ is a *greatest lower bound* of \mathcal{R} :
Let $T \in \mathbf{Prel}(X)$ with $\prod \mathcal{R} \subseteq T \subseteq S$ for all $S \in \mathcal{R}$. According to lemma 20.4.9(4)(a), this implies $\prod \mathcal{R} \parallel X \subseteq T \parallel X \subseteq S \parallel X$ for all $S \in \mathcal{R}$, so that $\bigcap \{R \parallel X \mid R \in \mathcal{R}\} \subseteq T \parallel X \subseteq S \parallel X$ for all $S \in \mathcal{R}$. Thus $\bigcap \{R \parallel X \mid R \in \mathcal{R}\} = T \parallel X$, i.e. $\prod \mathcal{R} \equiv T$, due to lemma 20.4.9(4)(b).

- (2) Proof similar to (1).
- (3) Is true because $R \sqcap S \equiv \prod \{R, S\}$ (see 20.4.21(9)) and statement (1).
- (4) This is true due to $R \sqcup S \equiv \coprod \{R, S\}$ (see 20.4.21(10)) and (2).

20.4.21 Lemma

Let X be a proper schema. For all $R, S, T \in \mathbf{Prel}(X)$ and all $\mathcal{R} \subseteq \mathbf{Prel}(X)$ holds:

- (1) $\top \sqcap R = R$ (\top is neutral element of \sqcap)
- (2) $\perp \sqcup R = R$ (\perp is neutral element of \sqcup)
- (3) $R \sqcap R = R$ (\sqcap is idempotent)
- (4) $R \sqcup R = R$ (\sqcup is idempotent)
- (5) $R \sqcap (S \sqcap T) = (R \sqcap S) \sqcap T$ (\sqcap is associative)
- (6) $R \sqcup (S \sqcup T) = (R \sqcup S) \sqcup T$ (\sqcup is associative)
- (7) $R \sqcap S = S \sqcap R$ (\sqcap is commutative)
- (8) $R \sqcup S = S \sqcup R$ (\sqcup is commutative)
- (9) $R \sqcap S = \prod \{R, S\}$ (\sqcap is special case of \prod)
- (10) $R \sqcup S = \coprod \{R, S\}$ (\sqcup is special case of \coprod)
- (11) $S \sqcap \prod \mathcal{R} = \prod \{S \sqcap R \mid R \in \mathcal{R}\}$ (full distributivity)
- (12) $S \sqcup \prod \mathcal{R} = \prod \{S \sqcup R \mid R \in \mathcal{R}\}$ (full distributivity)
- (13) $\neg \prod \mathcal{R} = \prod \{\neg R \mid R \in \mathcal{R}\}$ (de Morgan's law)
- (14) $\neg \coprod \mathcal{R} = \prod \{\neg R \mid R \in \mathcal{R}\}$ (de Morgan's law)
- (15) $\perp \sqcap R \equiv \perp$ (\perp quasi-cancels \sqcap)
- (16) $\top \sqcup R \equiv \top$ (\top quasi-cancels \sqcup)
- (17) $\perp \subseteq R$ (\perp is a least element)
- (18) $R \subseteq \top$ (\top is a greatest element)
- (19) $\prod \{R\} = R$ (\prod is idempotent)
- (20) $\coprod \{R\} = R$ (\coprod is idempotent)
- (21) $R \sqcap \neg R \equiv \perp$ (\sqcap -quasi-complement)
- (22) $R \sqcup \neg R \equiv \top$ (\sqcup -quasi-complement)

20.4.22 Proof of 20.4.21

Suppose $R, S, T \in \mathbf{Prel}(X)$ and $\mathcal{R} \subseteq \mathbf{Prel}(X)$ are given by $R = [Y, \Gamma]$, $S = [Y', \Gamma']$, $T = [Y'', \Gamma'']$, and $\mathcal{R} = \{R_k \mid k \in K\}$ with $R_k = [Y_k, \Gamma_k]$ for each $k \in K$.

- (1) $\top \sqcap R = \left(\left[\begin{array}{c} \langle \rangle \\ \{\langle \rangle\} \end{array} \right] \parallel Y \right) \cap \left(\left[\begin{array}{c} Y \\ \Gamma \end{array} \right] \parallel Y \right) = \left[\begin{array}{c} Y \\ \otimes Y \end{array} \right] \cap \left[\begin{array}{c} Y \\ \Gamma \end{array} \right] = \left[\begin{array}{c} Y \\ \Gamma \end{array} \right] = R$
- (2) $\perp \sqcup R = \left(\left[\begin{array}{c} \langle \rangle \\ \emptyset \end{array} \right] \parallel Y \right) \cup \left(\left[\begin{array}{c} Y \\ \Gamma \end{array} \right] \parallel Y \right) = \left[\begin{array}{c} Y \\ \emptyset \end{array} \right] \cup \left[\begin{array}{c} Y \\ \Gamma \end{array} \right] = \left[\begin{array}{c} Y \\ \Gamma \end{array} \right] = R$
- (3) $R \sqcap R = (R \parallel Y) \cap (R \parallel Y) = R \cap R = \left[\begin{array}{c} Y \\ \Gamma \cap \Gamma \end{array} \right] = \left[\begin{array}{c} Y \\ \Gamma \end{array} \right] = R$
- (4) $R \sqcup R = (R \parallel Y) \cup (R \parallel Y) = R \cup R = \left[\begin{array}{c} Y \\ \Gamma \cup \Gamma \end{array} \right] = \left[\begin{array}{c} Y \\ \Gamma \end{array} \right] = R$
- (5) Let us put $Z := Y \vee Y' \vee Y''$. There is $\mathbf{x}(R \sqcap S) = Y \vee Y'$ and $\mathbf{x}(S \sqcap T) = Y' \vee Y''$, so that

$$\begin{aligned}
& R \sqcap (S \sqcap T) \\
&= R \parallel (Y \vee (Y' \vee Y'')) \cap ((S \sqcap T) \parallel (Y \vee (Y' \vee Y''))) \\
&= R \parallel Z \cap ((S \sqcap T) \parallel Z) \\
&= R \parallel Z \cap (S \parallel Z \sqcap T \parallel Z) \\
&= R \parallel Z \cap (S \parallel Z \cap T \parallel Z) \\
&= (R \parallel Z \cap S \parallel Z) \cap T \parallel Z \\
&= (R \parallel Z \sqcap S \parallel Z) \cap T \parallel Z \\
&= ((R \sqcap S) \parallel Z) \cap T \parallel Z \\
&= ((R \sqcap S) \parallel ((Y \vee Y') \vee Y'')) \cap T \parallel ((Y \vee Y') \vee Y'') \\
&= (R \sqcap S) \sqcap T
\end{aligned}$$

(6) Same as (5) with “ \sqcap ” and “ \cap ” replaced by “ \sqcup ” and “ \cup ”, respectively.

(7) Obvious by now.

(8) See (7).

$$(9) R \sqcap S = (R \parallel (Y \vee Y')) \cap (S \parallel (Y \vee Y')) = \cap \{(R \parallel (Y \vee Y')), (S \parallel (Y \vee Y'))\} = \prod \{R, S\}$$

(10) Similar to (9).

(11) We define $Z := \bigvee \{Y_k \mid k \in K\}$ and obtain

$$\begin{aligned}
& S \sqcap \prod \mathcal{R} \\
&= (S \parallel Y' \vee Z) \cap (\prod \mathcal{R} \parallel Y' \vee Z) \\
&= (S \parallel Y' \vee Z) \cap \prod_{k \in K} (R_k \parallel Y' \vee Z) \quad \text{def. 20.2.2 of } \sqcap \\
&= (S \parallel Y' \vee Z) \cap \bigcup_{k \in K} (R_k \parallel Y' \vee Z \parallel Y' \vee Z) \quad \text{due to 20.4.15(9)} \\
&= (S \parallel Y' \vee Z) \cap \bigcup_{k \in K} (R_k \parallel Y' \vee Z) \quad \text{def. 20.2.4 of } \prod \\
&= \bigcup_{k \in K} ((S \parallel Y' \vee Z) \cap (R_k \parallel Y' \vee Z)) \quad \text{due to 20.1.11(1)} \\
&= \bigcup_{k \in K} ((S \parallel Y' \vee Z) \cap (R_k \parallel Y' \vee Z)) \quad \text{due to the distributivity of } \bigcup \text{ and } \cap \\
&= \bigcup_{k \in K} ((S \sqcap R_k) \parallel Y' \vee Z) \quad \text{due to 20.4.4(5)} \\
&= \prod_{k \in K} (S \sqcap R_k) \quad \text{due to 20.4.15(4)} \\
& \quad \text{again def. 20.2.4 of } \prod
\end{aligned}$$

(12) Similar to (11).

(13) Let $Z := \bigvee \{\mathbf{x}(R) \mid R \in \mathcal{R}\}$, then

$$\begin{aligned}
& \neg \prod \mathcal{R} \\
&= \neg \bigcap_{R \in \mathcal{R}} (R \parallel Z) \\
&= \bigcup_{R \in \mathcal{R}} \neg (R \parallel Z) \quad \text{due to 19.5.14(13)} \\
&= \bigcup_{R \in \mathcal{R}} (\neg R \parallel Z) \quad \text{due to ??(3)} \\
&= \prod_{R \in \mathcal{R}} (\neg R)
\end{aligned}$$

(14) Proof similar to (13).

$$(15) \perp \sqcap R = \left(\begin{bmatrix} \langle \rangle \\ \emptyset \end{bmatrix} \parallel Y \right) \cap \left(\begin{bmatrix} Y \\ \Gamma \end{bmatrix} \parallel Y \right) = \begin{bmatrix} Y \\ \emptyset \end{bmatrix} \cap \begin{bmatrix} Y \\ \Gamma \end{bmatrix} = \begin{bmatrix} Y \\ \emptyset \end{bmatrix} = \perp_Y \equiv \perp$$

$$(16) \top \sqcup R = \left(\begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \end{bmatrix} \parallel Y \right) \cup \left(\begin{bmatrix} Y \\ \Gamma \end{bmatrix} \parallel Y \right) = \begin{bmatrix} Y \\ \otimes Y \end{bmatrix} \cup \begin{bmatrix} Y \\ \Gamma \end{bmatrix} = \begin{bmatrix} Y \\ \otimes Y \end{bmatrix} = \top_Y \equiv \top$$

(17) $\perp \sqsubseteq R$ iff $\perp \parallel Y \subseteq R$ iff $[Y, \emptyset] \subseteq [Y, \Gamma]$ iff $\emptyset \subseteq \Gamma$, which is

always the case.

(18) $R \sqsubseteq \top$ iff $R \subseteq \top \parallel Y$ iff $[Y, \Gamma] \subseteq [Y, \otimes Y]$ iff $\Gamma \subseteq \otimes Y$, which is always the case.

(19) We apply (3) and obtain $\prod \{R\} = \prod \{R, R\} = R \sqcap R = R$

(20) Similar to (19).

(21) We have

$$\begin{aligned}
R \sqcap \neg R &= \begin{bmatrix} Y \\ \Gamma \end{bmatrix} \cap \begin{bmatrix} Y \\ (\otimes Y) \setminus \Gamma \end{bmatrix} \\
&= \begin{bmatrix} Y \\ \Gamma \cap ((\otimes Y) \setminus \Gamma) \end{bmatrix} \\
&= \begin{bmatrix} Y \\ \emptyset \end{bmatrix} \\
&= \perp_Y \\
&\equiv \perp
\end{aligned}$$

(22) We have

$$\begin{aligned}
R \sqcup \neg R &= \begin{bmatrix} Y \\ \Gamma \end{bmatrix} \sqcup \begin{bmatrix} Y \\ (\otimes Y) \setminus \Gamma \end{bmatrix} \\
&= \begin{bmatrix} Y \\ \Gamma \cup ((\otimes Y) \setminus \Gamma) \end{bmatrix} \\
&= \begin{bmatrix} Y \\ \otimes Y \end{bmatrix} \\
&= \top_Y \\
&\equiv \top
\end{aligned}$$

20.4.23 Definition — notation

The *monoid*-properties of \sqcap and \sqcup , i.e. the associativity (20.4.21(5) and (6)) and the existence of neutral elements (20.4.21(1) and (2)), allow us to use the following abbreviating notations:

$$\begin{aligned}
R_1 \sqcap \dots \sqcap R_n &:= \begin{cases} \top & \text{if } n = 0 \\ R_1 \sqcap (R_2 \sqcap \dots \sqcap R_n) & \text{if } n > 0 \end{cases} \\
R_1 \sqcup \dots \sqcup R_n &:= \begin{cases} \perp & \text{if } n = 0 \\ R_1 \sqcup (R_2 \sqcup \dots \sqcup R_n) & \text{if } n > 0 \end{cases}
\end{aligned}$$

for every list R_1, \dots, R_n of (pairwise) compatible relations.

20.4.24 Lemma

For every proper schema X and all $R, S \in \mathbf{Prel}(X)$, the following statements are equivalent:

- (1) $R \sqsubseteq S$
- (2) $\neg S \sqsubseteq \neg R$
- (3) $R \sqcup S \equiv S$
- (4) $R \sqcap S \equiv R$
- (5) $R \sqcap \neg S \equiv \perp$
- (6) $\neg R \sqcup S \equiv \top$

20.4.25 Proof of 20.4.24

Suppose, R and S are given as $R = [Y, \Gamma]$ and $S = [Z, \Sigma]$.

Below we will provide the proof by showing, that (1) is equivalent to (2), ..., (6), each. But first let us recall, that for every class C and all $A, B \in \mathbf{P}(C)$,

- (a) $A \subseteq B$ iff $C \setminus B \subseteq C \setminus A$
- (b) $A \subseteq B$ iff $A \cup B = B$
- (c) $A \subseteq B$ iff $A \cap B = A$
- (d) $A \subseteq B$ iff $A \cap (C \setminus B) = \emptyset$
- (e) $A \subseteq B$ iff $(C \setminus B) \cup A = C$

Furthermore, we have

$$\begin{aligned}
R \parallel Y \vee Z &= \begin{bmatrix} Y \\ \Gamma \end{bmatrix} \parallel (Z \setminus Y) && \text{due to 20.1.3(1)} \\
&= \begin{bmatrix} Y \dot{\vee} (Z \setminus Y) \\ \Gamma \circledast \otimes (Z \setminus Y) \end{bmatrix} && \text{due to 20.1.3(2)} \\
&= \begin{bmatrix} Y \vee Z \\ \Gamma \circledast \otimes (Z \setminus Y) \end{bmatrix}
\end{aligned}$$

and

$$\begin{aligned}
(\neg R) \parallel Y \vee Z &= \begin{bmatrix} Y \\ \otimes Y \setminus \Gamma \end{bmatrix} \parallel Y \vee Z && \text{def. 19.4.1 of } \neg \\
&= \begin{bmatrix} Y \dot{\vee} (Z \setminus Y) \\ (\otimes Y \setminus \Gamma) \circledast \otimes (Z \setminus Y) \end{bmatrix} && \text{due to 20.1.3} \\
&= \begin{bmatrix} Y \vee Z \\ (\otimes Y \circledast \otimes (Z \setminus Y)) \setminus (\Gamma \circledast \otimes (Z \setminus Y)) \end{bmatrix} && \text{due to 16.6.1(3)}
\end{aligned}$$

Similarly, we find that

$$\begin{aligned}
S \parallel Y \vee Z &= \begin{bmatrix} Y \vee Z \\ \Sigma \circledast \otimes (Y \setminus Z) \end{bmatrix} \\
(\neg S) \parallel Y \vee Z &= \begin{bmatrix} Y \vee Z \\ (\otimes Z \circledast \otimes (Y \setminus Z)) \setminus (\Sigma \circledast \otimes (Y \setminus Z)) \end{bmatrix}
\end{aligned}$$

Now, a proof of “(1) iff (2)” is given by

$$\begin{aligned}
R \sqsubseteq S & \\
\text{iff } R \parallel Y \vee Z \subseteq S \parallel Y \vee Z && \text{def. ?? of } \sqsubseteq \\
\text{iff } \begin{bmatrix} Y \vee Z \\ \Gamma \circledast \otimes (Z \setminus Y) \end{bmatrix} \subseteq \begin{bmatrix} Y \vee Z \\ \Sigma \circledast \otimes (Y \setminus Z) \end{bmatrix} & \\
\text{iff } \Gamma \circledast \otimes (Z \setminus Y) \subseteq \Sigma \circledast \otimes (Y \setminus Z) & \\
\text{iff } \left(\begin{array}{l} \otimes (Y \vee Z) \setminus (\Sigma \circledast \otimes (Y \setminus Z)) \\ \subseteq \otimes (Y \vee Z) \setminus (\Gamma \circledast \otimes (Z \setminus Y)) \end{array} \right) && \text{due to (a)} \\
\text{iff } \left(\begin{array}{l} \otimes (Z \dot{\vee} (Y \setminus Z)) \setminus (\Sigma \circledast \otimes (Y \setminus Z)) \\ \subseteq \otimes (Y \dot{\vee} (Z \setminus Y)) \setminus (\Gamma \circledast \otimes (Z \setminus Y)) \end{array} \right) & \\
\text{iff } \left(\begin{array}{l} (\otimes Z \circledast \otimes (Y \setminus Z)) \setminus (\Sigma \circledast \otimes (Y \setminus Z)) \\ \subseteq (\otimes Y \circledast \otimes (Z \setminus Y)) \setminus (\Gamma \circledast \otimes (Z \setminus Y)) \end{array} \right) && \text{due to 16.6.5(4)} \\
\text{iff } \left(\begin{array}{l} (\otimes Z \setminus \Sigma) \circledast \otimes (Y \setminus Z) \\ \subseteq (\otimes Y \setminus \Gamma) \circledast \otimes (Z \setminus Y) \end{array} \right) && \text{due to 16.6.1(3)} \\
\text{iff } \begin{bmatrix} Y \vee Z \\ (\otimes Z \setminus \Sigma) \circledast \otimes (Y \setminus Z) \end{bmatrix} \subseteq \begin{bmatrix} Y \vee Z \\ (\otimes Y \setminus \Gamma) \circledast \otimes (Z \setminus Y) \end{bmatrix} & \\
\text{iff } (\neg S) \parallel Y \vee Z \subseteq (\neg R) \parallel Y \vee Z & \\
\text{iff } \neg S \sqsubseteq \neg R &
\end{aligned}$$

A proof of “(1) iff (3)” is given by

$$\begin{aligned}
R \sqcup S \equiv S & \\
\text{iff } (R \sqcup S) \parallel Y \vee Z = S \parallel Y \vee Z && \text{def. 20.2.2 of } \equiv \\
\text{iff } (R \parallel Y \vee Z) \sqcup (S \parallel Y \vee Z) = S \parallel Y \vee Z && \text{due to 20.4.15(5)} \\
\text{iff } (R \parallel Y \vee Z) \cup (S \parallel Y \vee Z) \subseteq (S \parallel Y \vee Z) && \text{def. 20.2.2 of } \sqcup
\end{aligned}$$

$$\begin{aligned}
&\text{iff } \begin{bmatrix} Y \vee Z \\ \Gamma \circledast \otimes (Z \setminus Y) \end{bmatrix} \cup \begin{bmatrix} Y \vee Z \\ \Sigma \circledast \otimes (Y \setminus Z) \end{bmatrix} \subseteq \begin{bmatrix} Y \vee Z \\ \Sigma \circledast \otimes (Y \setminus Z) \end{bmatrix} \\
&\text{iff } (\Gamma \circledast \otimes (Z \setminus Y)) \cup (\Sigma \circledast \otimes (Y \setminus Z)) \subseteq (\Sigma \circledast \otimes (Y \setminus Z)) \\
&\text{iff } (\Gamma \circledast \otimes (Z \setminus Y)) \subseteq (\Sigma \circledast \otimes (Y \setminus Z)) && \text{due to (b)} \\
&\text{iff } R \sqsubseteq S
\end{aligned}$$

A proof of “(1) iff (4)” is similarly given by

$$\begin{aligned}
R \cap S \equiv S & \\
\text{iff } (\Gamma \circledast \otimes (Z \setminus Y)) \cap (\Sigma \circledast \otimes (Y \setminus Z)) \subseteq (\Sigma \circledast \otimes (Y \setminus Z)) \\
\text{iff } (\Gamma \circledast \otimes (Z \setminus Y)) \subseteq (\Sigma \circledast \otimes (Y \setminus Z)) && \text{due to (c)} \\
\text{iff } R \sqsubseteq S
\end{aligned}$$

A proof of “(1) iff (5)” is similar by using (d).
Finally, (e) can be used to show “(1) iff (6)”.

20.4.26 Lemma congruence

Let X be a proper schema.

If $R, R', S, S' \in \mathbf{Prel}(X)$ with $R \equiv S$ and $R' \equiv S'$, then

- (1) $R \sqsubseteq R'$ iff $S \sqsubseteq S'$
- (2) $R \cap S \equiv R' \cap S'$
- (3) $R \sqcup S \equiv R' \sqcup S'$
- (4) $\neg R \equiv \neg S$

If $\mathcal{R}, \mathcal{S} \subseteq \mathbf{Prel}(X)$ such that each $R \in \mathcal{R}$ has an equivalent $S \in \mathcal{S}$ and vice versa, then

- (5) $\bigcap \mathcal{R} \equiv \bigcap \mathcal{S}$
- (6) $\bigcup \mathcal{R} \equiv \bigcup \mathcal{S}$

20.4.27 Proof of 20.4.26

Should be clear by now.

20.4.28 Lemma total distributivity

Let $X = [X_i | i \in I]$ be a proper schema. Let $L = [L_k | k \in K]$ also be a proper schema and $\mathcal{R}_{kl} \in \mathbf{Prel}(X)$, for each $k \in K$ and $l \in L_k$. Then

- (1) $\prod_{k \in K} \prod_{l \in L_k} \mathcal{R}_{kl} = \prod_{\lambda \in \otimes L} \prod_{k \in K} \mathcal{R}_{k\lambda(k)}$
- (2) $\prod_{k \in K} \prod_{l \in L_k} \mathcal{R}_{kl} = \prod_{\lambda \in \otimes L} \prod_{k \in K} \mathcal{R}_{k\lambda(k)}$

20.4.29 Proof of 20.4.28

Left as exercise.

20.4.30 Lemma

Let X be a proper schema, $R \in \mathbf{Prel}(X)$, and $\mathcal{R} = \{\mathcal{R}_k | k \in K\} \subseteq \mathbf{Prel}(X)$. Then

- (1) $\neg R \equiv \perp$ iff $R \equiv \top$
- (2) $\neg R \equiv \top$ iff $R \equiv \perp$
- (3) $(\forall k \in K. \mathcal{R}_k \equiv \top)$ iff $\prod \mathcal{R} \equiv \top$
- (4) $(\forall k \in K. \mathcal{R}_k \equiv \perp)$ iff $\prod \mathcal{R} \equiv \perp$
- (5) $(\exists k \in K. \mathcal{R}_k \equiv \perp)$ implies $\prod \mathcal{R} \equiv \perp$
- (6) $(\exists k \in K. \mathcal{R}_k \equiv \top)$ implies $\prod \mathcal{R} \equiv \top$

20.4.31 Proof of 20.4.30

Left as exercise.

20.5 More derived semantic operations

20.5.1 Remark

In section 8 we introduced additional junctors derived from quasi-boolean algebras, in particular the multiary *subjunct* “ \rightarrow ” and *equijunct* “ \leftrightarrow ” (8.2.2). Here in our final subsection 20.5 of section 20, we introduce these junctors for our (quasi-boolean algebra) of compatible relations. At least the simple binary versions “ $\textcircled{1} \rightarrow \textcircled{2}$ ” and “ $\textcircled{1} \leftrightarrow \textcircled{2}$ ”. We also add the *subtraction* (or *difference*) “ $\textcircled{1} - \textcircled{2}$ ”.

In the introductory remark 8.0.5 of section 8, we discouraged the reader to study that section. So we introduce these new junctors here along with some lemmata and proofs, that don't refer to section 8.

20.5.2 Definition

For all compatible relations R and S we define:

$R - S$	$:= R \sqcap \neg S$	(subtraction or difference)
$R \rightarrow S$	$:= \neg R \sqcup S$	(subjunct)
$R \leftrightarrow S$	$:= (\neg R \sqcap \neg S) \sqcup (R \sqcap S)$	(equijunct)

20.5.3 Lemma —properties of sub- and equijunction—

For every proper schema X and all $R, S \in \mathbf{Prel}(X)$ holds:

- (1) $\mathbf{x}(R \rightarrow S) = \mathbf{x}(R) \vee \mathbf{x}(S)$ (subj. schema)
- (2) $\mathbf{x}(R \leftrightarrow S) = \mathbf{x}(R) \vee \mathbf{x}(S)$ (equij. schema)
- (3) $R \leftrightarrow S = (R \rightarrow S) \sqcap (S \rightarrow R)$ (double subj.)
- (4) $R \sqsubseteq S$ iff $R \rightarrow S \equiv \top$ (subvalence criterion)
- (5) $R \equiv S$ iff $R \leftrightarrow S \equiv \top$ (equivalence criterion)
- (6) $\perp \rightarrow R \equiv \top$ (left bottom)
- (7) $\top \rightarrow R = R$ (left top)
- (8) $R \rightarrow \perp = \neg R$ (right bottom)
- (9) $R \rightarrow \top \equiv \top$ (right top)
- (10) $R \leftrightarrow \perp = \neg R$ (negative literal)
- (11) $R \leftrightarrow \top = R$ (positive literal)
- (12) $R - S = \neg(R \rightarrow S)$ (subtraction and subjunct)

20.5.4 Proof of 20.5.3

Suppose $R = [Y, \Gamma]$ and $S = [Z, \Sigma]$. $R, S \in \mathbf{Prel}(X)$ implies the compatibility of R and S , so all the sub- and equijunction terms of the lemma are well-defined.

(1) We have

$$\begin{aligned} \mathbf{x}(R \rightarrow S) &= \mathbf{x}(\neg R \sqcup S) && \text{def. 20.5.2} \\ &= \mathbf{x}(\neg R) \vee \mathbf{x}(S) && \text{due to 20.4.7(5)} \\ &= \mathbf{x}(R) \vee \mathbf{x}(S) && \text{due to 20.4.7(3)} \end{aligned}$$

(2) Similar to (1), we have

$$\begin{aligned} \mathbf{x}(R \leftrightarrow S) &= (\mathbf{x}(R) \vee \mathbf{x}(S)) \vee (\mathbf{x}(R) \vee \mathbf{x}(S)) \\ &= \mathbf{x}(R) \vee \mathbf{x}(S) \end{aligned}$$

(3) Applying definition 20.5.2 of \rightarrow and \leftrightarrow and the boolean laws from 20.4.21 we obtain

$$\begin{aligned} R \leftrightarrow S &= (\neg R \sqcap \neg S) \sqcup (R \sqcap S) \\ &= (\neg R \sqcup R) \sqcap (\neg R \sqcup S) \sqcap (\neg S \sqcup R) \sqcap (\neg S \sqcup S) \\ &= \top_Y \sqcap (\neg R \sqcup S) \sqcap (\neg S \sqcup R) \sqcap \top_Z \\ &= (\neg R \sqcup S) \sqcap (\neg S \sqcup R) \\ &= (R \rightarrow S) \sqcap (S \rightarrow R) \end{aligned}$$

(4) We have

$$\begin{aligned} R \sqsubseteq S & \\ \Leftrightarrow \neg R \sqcup S \equiv \top & \text{due to 20.4.24} \\ \Leftrightarrow R \rightarrow S \equiv \top & \text{def. 20.5.2 of } \rightarrow \end{aligned}$$

(5) If $R \equiv S$, then $\neg R \sqcap \neg S \equiv \neg R$ and $R \sqcap S \equiv R$, so that $R \leftrightarrow S = (\neg R \sqcap \neg S) \sqcup (R \sqcap S) \equiv \neg R \sqcup R \equiv \top$. On the other hand, if $R \leftrightarrow S \equiv \top$, then $(R \rightarrow S) \sqcap (S \rightarrow R) \equiv \top$ according to (3), so $R \rightarrow S \equiv \top$ and $S \rightarrow R \equiv \top$, so that $R \sqsubseteq S$ and $S \sqsubseteq R$ according to (4), which means $R \equiv S$ according to 20.4.9(1).

(6) We have

$$\begin{aligned} \perp \rightarrow R &= \neg \perp \sqcup R && \text{def. 20.5.2} \\ &= \top \sqcup R && \text{due to 19.4.8(1)} \\ &\equiv \top && \text{due to 20.4.21(16)} \end{aligned}$$

(7) Similar to (6) we have

$$\top \rightarrow R = \neg \top \sqcup R = \perp \sqcup R = R$$

(8) We have

$$\begin{aligned} R \rightarrow \perp &= \neg R \sqcup \perp && \text{def. 20.5.2} \\ &= \neg R && \text{due to 20.4.21(2)} \end{aligned}$$

(9) Similar to (8) we have

$$R \rightarrow \top = \neg R \sqcup \top \equiv \top$$

(10) We have

$$\begin{aligned} R \leftrightarrow \perp &= (\neg R \sqcap \neg \perp) \sqcup (R \sqcap \perp) && \text{def. 20.5.2} \\ &= (\neg R \sqcap \top) \sqcup \perp && \text{in particular due to 20.4.21(15)} \\ &= \neg R \sqcup \perp && \text{due to 20.4.21(1)} \\ &= \neg R && \text{due to 20.4.21(2)} \end{aligned}$$

(11) Similar to (10) we obtain

$$\begin{aligned} R \leftrightarrow \top &= (\neg R \sqcap \neg \top) \sqcup (R \sqcap \top) \\ &= (\neg R \sqcap \perp) \sqcup R \\ &= \perp \sqcup R \\ &= R \end{aligned}$$

(12) We have

$$\begin{aligned} R - S &= R \sqcap \neg S && \text{def. 20.5.2} \\ &= \neg \neg(R \sqcap \neg S) && \text{due to 19.4.6} \\ &= \neg(\neg R \sqcup \neg \neg S) && \text{due to 20.4.21(13), (9), and (10)} \\ &= \neg(\neg R \sqcup S) && \text{due to 19.4.6} \\ &= \neg(R \rightarrow S) && \text{def. 20.5.2} \end{aligned}$$

21 Decreasing the schema of relations

21.1 Introduction and overview

21.1.1 Remark __Various operations to decrease the schema__

We already have an operation available to increase the schema of a relation, namely the *expander* \parallel . Decreasing the schema is a more subtle process and we will present six or seven different operations to do the job. One reason for this variety is the diversity of possible operation types to express the idea.

Let $X = [X_i | i \in I]$ be a schema, $R \in \mathbf{Rel}(X)$ and $Y \leq X$, i.e. $Y = \mathbf{pr}(X, J)$ for some $J \subseteq I$. Suppose we want to decrease the schema X of R and Y shall be the schema of the resulting relation R' . We call this

- ♣ a reduction of R onto Y , or
- ♣ a (co)projection of R onto J , or
- ♣ an elimination of $K := I \setminus J$ from R

That is already a lot of terminology for basically the same thing. Nevertheless, the variety is useful. In particular, as their actual formalizations uses very different approaches.

A reduction of R onto Y is the attempt to find a $R' \in \mathbf{Rel}(Y)$ equivalent to R . But such a R' exists only in rare cases. In general, we only have two closest results: the maximal subvalent member of $\mathbf{Rel}(Y)$, called the infimum reduction of R onto Y and written $R \uparrow Y$, and the minimal supervalent element, called the supremum reduction of R onto Y , noted as $R \downarrow Y$. It is very important to understand the arguments that lead to these concepts. In 21.1.2 and 21.1.3, we use double tables to motivate and introduce these reductions.

A projection of R onto J , written $\mathbf{pr}(R, J)$, is an often used operation on relations and seems the most obvious way of schema decreasing when we look at the graph table of R : to obtain $\mathbf{pr}(R, J)$, simply delete all the columns whose attributes are not in J .

It will turn out (in 21.6.4(1)(a)), that the projection resembles the supremum reduction: $\mathbf{pr}(R, J) = R \downarrow Y$. In order to have its dual concept available as well, we introduce the coprojection of R onto J , written $\mathbf{cpj}(R, J)$, with $\mathbf{cpj}(R, J) = R \uparrow Y$.

Finally, eliminations come in two flavors, accordingly: $R \downarrow K := \mathbf{pr}(R, J)$, the supremum elimination, read “the upper R without K ”, and $R \uparrow K := \mathbf{cpj}(R, J)$, the infimum elimination, read as “the lower R without K ”.

21.1.2 Remark .Equivalent reduction and redundant attributes.

In the sequel, let a relation $R = [X, \Gamma]$ and a schema Y with $Y \sim X$ be given.

We already defined a method to *expand* the schema X of R by Y , the resulting relation is $R \parallel Y$ and its expanded schema is $X \vee Y$. $R \parallel Y$ is distinguished from all the other elements of

$\mathbf{Rel}(X \vee Y)$ by the fact that it is the only one equivalent to R . We could have defined

$$R \parallel Y := \mathbf{the } S \in \mathbf{Rel}(Y) \text{ with } S \equiv R$$

Let us now define the inverse operation: a reduction, i.e. an operation that reduces the schema of relations. For now we take this term literally and restrict the definition to the case $Y \leq X$. So, in case $Y \leq X$, we define the equivalent reduction of R onto Y as

$$R \Downarrow Y := \mathbf{the } S \in \mathbf{Rel}(Y) \text{ with } S \equiv R$$

If such an $R \Downarrow Y$ exists, it is unique. However, in most cases it does not exist.

Double tables provide a nice and intuitive way to demonstrate the situation. We consider the example where X, Y, R are given by

$$X = \begin{bmatrix} a \mapsto \{1, 2\} \\ b \mapsto \{2, 3, 4\} \\ c \mapsto \{1, 2\} \\ d \mapsto \{2, 3, 4\} \end{bmatrix} \quad Y = \begin{bmatrix} a \mapsto \{1, 2\} \\ b \mapsto \{2, 3, 4\} \end{bmatrix}$$

$$R = \begin{bmatrix} X \\ x \rightsquigarrow x(a) \cdot x(b) < x(c) + x(d) \end{bmatrix}$$

The double table of R (according to 17.6.1) with Y as left and $X \setminus Y$ as top schema is

		1	2	1	2	1	2	c
		2	2	3	3	4	4	d
1	2	2 < 3	2 < 4	2 < 4	2 < 5	2 < 5	2 < 6	
2	2	4 < 3	4 < 4	4 < 4	4 < 5	4 < 5	4 < 6	
1	3	3 < 3	3 < 4	3 < 4	3 < 5	3 < 5	3 < 6	
2	3	6 < 3	6 < 4	6 < 4	6 < 5	6 < 5	6 < 6	
1	4	4 < 3	4 < 4	4 < 4	4 < 5	4 < 5	4 < 6	
2	4	8 < 3	8 < 4	8 < 4	8 < 5	8 < 5	8 < 6	
a	b							

$$= \begin{bmatrix} 1 & 2 & 1 & 2 & 1 & 2 & c \\ 2 & 2 & 3 & 3 & 4 & 4 & d \\ 1 & 2 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 0 & 0 & 0 & 1 & 1 \\ 1 & 3 & 0 & 1 & 1 & 1 & 1 \\ 2 & 3 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 1 & 1 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 \\ a & b & & & & & \end{bmatrix}$$

Let us try to construct $R \Downarrow Y$. The schema of $R \Downarrow Y$ is Y , so $R \Downarrow Y$ can be represented by a boolean table

$$R \Downarrow Y = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 2 & \beta_1 \\ \hline 2 & 2 & \beta_2 \\ \hline 1 & 3 & \beta_3 \\ \hline 2 & 3 & \beta_4 \\ \hline 1 & 4 & \beta_5 \\ \hline 2 & 4 & \beta_6 \\ \hline \end{array}$$

$$S \parallel X = \begin{array}{|c|c|c|c|c|c|c|c|} \hline & & & & & & & & & & c \\ \hline & & 1 & 2 & 1 & 2 & 1 & 2 & & & d \\ \hline & & 2 & 2 & 3 & 3 & 4 & 4 & & & \\ \hline 1 & 2 & \beta_1 & \beta_1 & \beta_1 & \beta_1 & \beta_1 & \beta_1 & & & \\ \hline 2 & 2 & \beta_2 & \beta_2 & \beta_2 & \beta_2 & \beta_2 & \beta_2 & & & \\ \hline 1 & 3 & \beta_3 & \beta_3 & \beta_3 & \beta_3 & \beta_3 & \beta_3 & & & \\ \hline 2 & 3 & \beta_4 & \beta_4 & \beta_4 & \beta_4 & \beta_4 & \beta_4 & & & \\ \hline 1 & 4 & \beta_5 & \beta_5 & \beta_5 & \beta_5 & \beta_5 & \beta_5 & & & \\ \hline 2 & 4 & \beta_6 & \beta_6 & \beta_6 & \beta_6 & \beta_6 & \beta_6 & & & \\ \hline a & b & & & & & & & & & \\ \hline \end{array}$$

It remains to define the $\beta_1, \dots, \beta_6 \in \mathbb{B}$. We want $R \Downarrow Y$ to satisfy $R \Downarrow Y \equiv R$, and that means $R \Downarrow Y \parallel X = R$. In double table representation (according to 20.1.5) that is

$$R \Downarrow Y \parallel X = \begin{array}{|c|c|c|c|c|c|c|c|} \hline & & & & & & & & & & c \\ \hline & & 1 & 2 & 1 & 2 & 1 & 2 & & & d \\ \hline & & 2 & 2 & 3 & 3 & 4 & 4 & & & \\ \hline 1 & 2 & \beta_1 & \beta_1 & \beta_1 & \beta_1 & \beta_1 & \beta_1 & & & \\ \hline 2 & 2 & \beta_2 & \beta_2 & \beta_2 & \beta_2 & \beta_2 & \beta_2 & & & \\ \hline 1 & 3 & \beta_3 & \beta_3 & \beta_3 & \beta_3 & \beta_3 & \beta_3 & & & \\ \hline 2 & 3 & \beta_4 & \beta_4 & \beta_4 & \beta_4 & \beta_4 & \beta_4 & & & \\ \hline 1 & 4 & \beta_5 & \beta_5 & \beta_5 & \beta_5 & \beta_5 & \beta_5 & & & \\ \hline 2 & 4 & \beta_6 & \beta_6 & \beta_6 & \beta_6 & \beta_6 & \beta_6 & & & \\ \hline a & b & & & & & & & & & \\ \hline \end{array}$$

i.e. in each of the six rows (of the center block), the entries for all cells must be identical. And $R \Downarrow Y \parallel X = R$ means that this double table must coincide with the one for R given earlier on. Obviously, there are no $\beta_1, \dots, \beta_6 \in \mathbb{B}$ to make that happen. The double table for R has rows with mixed values, e.g. in the second last row (the one for $a = 1$ and $b = 4$), both $\mathbf{0}$ and $\mathbf{1}$ occur. For this given R and Y , $R \Downarrow Y$ does not exist.

We will also say that $R \Downarrow Y$ exists iff each of the deleted attributes (here c and d) is redundant in R .

21.1.3 Remark Infimum and supremum reduction

$R \Downarrow Y$, the *equivalent* reduction onto Y might not always exist. But there is always:

- ♣ a maximal lower bound, i.e. the greatest $R' \in \mathbf{Rel}(Y)$ with $R' \sqsubseteq R$, called the infimum reduction of R onto Y and written $R \Downarrow Y$.
- ♣ and similarly, a minimal upper bound, written $R \Updownarrow Y$, the supremum reduction of R onto Y .

Let us take the previous example R and Y again. Let us consider a $S \in \mathbf{Rel}(Y)$, its general form was

$$S = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 2 & \beta_1 \\ \hline 2 & 2 & \beta_2 \\ \hline 1 & 3 & \beta_3 \\ \hline 2 & 3 & \beta_4 \\ \hline 1 & 4 & \beta_5 \\ \hline 2 & 4 & \beta_6 \\ \hline \end{array}$$

with $\beta_1, \dots, \beta_6 \in \mathbb{B}$ and so

Now suppose we want the β_1, \dots, β_6 such that $S \sqsubseteq R$, i.e. $S \parallel X \sqsubseteq R$. In double table representation that is

$$\begin{array}{|c|c|c|c|c|c|c|c|} \hline & & & & & & & & & & c \\ \hline & & 1 & 2 & 1 & 2 & 1 & 2 & & & d \\ \hline & & 2 & 2 & 3 & 3 & 4 & 4 & & & \\ \hline 1 & 2 & \beta_1 & \beta_1 & \beta_1 & \beta_1 & \beta_1 & \beta_1 & & & \\ \hline 2 & 2 & \beta_2 & \beta_2 & \beta_2 & \beta_2 & \beta_2 & \beta_2 & & & \\ \hline 1 & 3 & \beta_3 & \beta_3 & \beta_3 & \beta_3 & \beta_3 & \beta_3 & & & \\ \hline 2 & 3 & \beta_4 & \beta_4 & \beta_4 & \beta_4 & \beta_4 & \beta_4 & & & \\ \hline 1 & 4 & \beta_5 & \beta_5 & \beta_5 & \beta_5 & \beta_5 & \beta_5 & & & \\ \hline 2 & 4 & \beta_6 & \beta_6 & \beta_6 & \beta_6 & \beta_6 & \beta_6 & & & \\ \hline a & b & & & & & & & & & \\ \hline \end{array} \subseteq \begin{array}{|c|c|c|c|c|c|c|c|} \hline & & & & & & & & & & c \\ \hline & & 1 & 2 & 1 & 2 & 1 & 2 & & & d \\ \hline & & 2 & 2 & 3 & 3 & 4 & 4 & & & \\ \hline 1 & 2 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & & & \\ \hline 2 & 2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & & & \\ \hline 1 & 3 & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & & & \\ \hline 2 & 3 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & & \\ \hline 1 & 4 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & & & \\ \hline 2 & 4 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & & \\ \hline a & b & & & & & & & & & \\ \hline \end{array}$$

which means, that the β_i have to satisfy the following set of order statements

$$\begin{array}{l} \beta_1 \leq \mathbf{1} \quad \beta_1 \leq \mathbf{1} \quad \beta_1 \leq \mathbf{1} \quad \beta_1 \leq \mathbf{1} \quad \beta_1 \leq \mathbf{1} \quad \beta_1 \leq \mathbf{1} \\ \beta_2 \leq \mathbf{0} \quad \beta_2 \leq \mathbf{0} \quad \beta_2 \leq \mathbf{0} \quad \beta_2 \leq \mathbf{1} \quad \beta_2 \leq \mathbf{1} \quad \beta_2 \leq \mathbf{1} \\ \beta_3 \leq \mathbf{0} \quad \beta_3 \leq \mathbf{1} \quad \beta_3 \leq \mathbf{1} \quad \beta_3 \leq \mathbf{1} \quad \beta_3 \leq \mathbf{1} \quad \beta_3 \leq \mathbf{1} \\ \beta_4 \leq \mathbf{0} \quad \beta_4 \leq \mathbf{0} \quad \beta_4 \leq \mathbf{0} \quad \beta_4 \leq \mathbf{0} \quad \beta_4 \leq \mathbf{0} \quad \beta_4 \leq \mathbf{0} \\ \beta_5 \leq \mathbf{0} \quad \beta_5 \leq \mathbf{0} \quad \beta_5 \leq \mathbf{0} \quad \beta_5 \leq \mathbf{1} \quad \beta_5 \leq \mathbf{1} \quad \beta_5 \leq \mathbf{1} \\ \beta_6 \leq \mathbf{0} \quad \beta_6 \leq \mathbf{0} \quad \beta_6 \leq \mathbf{0} \quad \beta_6 \leq \mathbf{0} \quad \beta_6 \leq \mathbf{0} \quad \beta_6 \leq \mathbf{0} \end{array}$$

which is the case exactly if

$$\beta_1 \text{ is either } \mathbf{0} \text{ or } \mathbf{1} \text{ and } \beta_2 = \beta_3 = \beta_4 = \beta_5 = \beta_6 = \mathbf{0}$$

In parameterized boolean table notation that result goes: For every $S \in \mathbf{Rel}(Y)$,

$$S \sqsubseteq R \quad \text{iff} \quad S = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 2 & \beta_1 \\ \hline 2 & 2 & \mathbf{0} \\ \hline 1 & 3 & \mathbf{0} \\ \hline 2 & 3 & \mathbf{0} \\ \hline 1 & 4 & \mathbf{0} \\ \hline 2 & 4 & \mathbf{0} \\ \hline \end{array} \quad \text{with } \beta_1 \in \mathbb{B}$$

In other words, R has two lower bounds in $\mathbf{Rel}(Y)$: one S_1 with $\beta_1 = \mathbf{0}$ and one S_2 with $\beta_1 = \mathbf{1}$. Obviously, the maximum of these lower bounds is S_2 , where β_1 is put to its maximum value $\mathbf{1}$. S_2 is our wanted $R \Downarrow Y$.

An alternative way for saying that $R \Downarrow Y$ should be the maximal lower bound of R in $\mathbf{Rel}(Y)$ is the actual definition 21.6.1 of $R \Downarrow Y$ as

$$R \Downarrow Y := \bigcup \{S \in \mathbf{Rel}(Y) \mid S \sqsubseteq R\}$$

And indeed, for our example case, $R \Downarrow Y = S_1 \cup S_2$.

Using the same line of reasoning for the minimal upper bound, we find that

$$R \subseteq S \quad \text{iff} \quad S = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 2 & \mathbf{1} \\ \hline 2 & 2 & \mathbf{1} \\ \hline 1 & 3 & \mathbf{1} \\ \hline 2 & 3 & \beta_4 \\ \hline 1 & 4 & \mathbf{1} \\ \hline 2 & 4 & \beta_6 \\ \hline \end{array} \quad \text{with } \beta_4, \beta_6 \in \mathbb{B}$$

The minimum of these four upper bounds is given by assigning the minimum value $\mathbf{0}$ to the free parameters β_4, β_6 . Written with the new notation that is

$$R \Downarrow Y = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 2 & \mathbf{1} \\ \hline 2 & 2 & \mathbf{1} \\ \hline 1 & 3 & \mathbf{1} \\ \hline 2 & 3 & \mathbf{0} \\ \hline 1 & 4 & \mathbf{1} \\ \hline 2 & 4 & \mathbf{0} \\ \hline \end{array}$$

21.1.4 Double table method for supremum and infimum reduction

Following the arguments you might already have discovered a faster method to derive $R \uparrow Y$ and $R \Downarrow Y$ from the given double table of R :

- ♣ For $R \uparrow Y$ perform the boolean conjunction \wedge of all boolean values in each row.
- ♣ For $R \Downarrow Y$ perform the boolean disjunction \vee of all boolean values in each row.

For our example, this means:

$$R \uparrow Y = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 2 & \mathbf{1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1} \\ \hline 2 & 2 & \mathbf{0 \wedge 0 \wedge 0 \wedge 0 \wedge 1 \wedge 1 \wedge 1} \\ \hline 1 & 3 & \mathbf{0 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1} \\ \hline 2 & 3 & \mathbf{0 \wedge 0 \wedge 0 \wedge 0 \wedge 0 \wedge 0 \wedge 0} \\ \hline 1 & 4 & \mathbf{0 \wedge 0 \wedge 0 \wedge 0 \wedge 1 \wedge 1 \wedge 1} \\ \hline 2 & 4 & \mathbf{0 \wedge 0 \wedge 0 \wedge 0 \wedge 0 \wedge 0 \wedge 0} \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 2 & \mathbf{1} \\ \hline 2 & 2 & \mathbf{0} \\ \hline 1 & 3 & \mathbf{0} \\ \hline 2 & 3 & \mathbf{0} \\ \hline 1 & 4 & \mathbf{0} \\ \hline 2 & 4 & \mathbf{0} \\ \hline \end{array}$$

and

$$R \Downarrow Y = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 2 & \mathbf{1 \vee 1 \vee 1 \vee 1 \vee 1 \vee 1} \\ \hline 2 & 2 & \mathbf{0 \vee 0 \vee 0 \vee 0 \vee 1 \vee 1 \vee 1} \\ \hline 1 & 3 & \mathbf{0 \vee 1 \vee 1 \vee 1 \vee 1 \vee 1 \vee 1} \\ \hline 2 & 3 & \mathbf{0 \vee 0 \vee 0 \vee 0 \vee 0 \vee 0 \vee 0} \\ \hline 1 & 4 & \mathbf{0 \vee 0 \vee 0 \vee 0 \vee 1 \vee 1 \vee 1} \\ \hline 2 & 4 & \mathbf{0 \vee 0 \vee 0 \vee 0 \vee 0 \vee 0 \vee 0} \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline a & b & \\ \hline 1 & 2 & \mathbf{1} \\ \hline 2 & 2 & \mathbf{1} \\ \hline 1 & 3 & \mathbf{1} \\ \hline 2 & 3 & \mathbf{0} \\ \hline 1 & 4 & \mathbf{1} \\ \hline 2 & 4 & \mathbf{0} \\ \hline \end{array}$$

By the way, the notation “ \uparrow ” can be memorized by this method as “ $\uparrow = \parallel + \wedge$ ”. In lattice theory, the term “infimum” is synonymous to “meet” or “conjunction”. Accordingly, think of “ $\Downarrow = \parallel + \vee$ ” for the supremum reduction.

21.1.5 Remark overview

Next, we move on to state the proper formal definitions of the

mentioned operations and discuss their properties. We begin with the (co)projections (21.2) and continue with the eliminations (21.4), reductions (21.6) and redundancies of attributes (21.8).

So far, we only mentioned proper schema-decreasing operations in the sense that $R' \leq R$ for every R decreased to R' . But reductions, say a supremum reduction $R \Downarrow Y$, is defined as $\bigcap \{S \in \mathbf{Rel}(Y) \mid R \subseteq S\}$. That expression is not only well-defined for $Y \leq \mathbf{x}(R)$, but more general for every $Y \sim \mathbf{x}(R)$. In this respect, reductions will be more general than (co)projections and eliminations.

By the way, if $R = [X, \Gamma]$ and $Y \sim X$, we will see (in 21.6.4), that e.g. the general supremum reduction $R \Downarrow Y$ is a combined proper reduction and with a subsequent expansion: $R \Downarrow Y = R \Downarrow (Y \wedge X) \parallel Y$. We can understand these generalizations with the already known notions.

We will also see that the duality of, say supremum and infimum reductions is a proper duality indeed: one can be expressed in terms of the other, e.g. $R \Downarrow Y = \neg((\neg R) \uparrow Y)$, etc. We use this fact for the introduction of the coprojection as $\mathbf{cpj}(R, J) := \neg \mathbf{pr}(\neg R, J)$.

21.2 Projection and coprojection

21.2.1 Repetition

Recall from 16.5.2, that

$$\mathbf{pr}(\Gamma, J) := \{\mathbf{pr}(\xi, J) \mid \xi \in \Gamma\}$$

for every schema $X = [X_i \mid i \in I]$, each $\Gamma \subseteq \otimes X$ and all $J \subseteq I$.

21.2.2 Definition

Let $X = [X_i \mid i \in I]$ be a schema, $R = [X, \Gamma]$ a relation and $J \subseteq I$. We define

$$\mathbf{pr}(R, J) := \begin{bmatrix} \mathbf{pr}(X, J) \\ \mathbf{pr}(\Gamma, J) \end{bmatrix}$$

the projection of R onto J

$$\mathbf{cpj}(R, J) := \neg \mathbf{pr}(\neg R, J)$$

the coprojection of R onto J

21.2.3 Table representations

Let X be a schema, $R \in \mathbf{Rel}(X)$, and $J \subseteq \mathbf{dom}(X)$.

- ♣ If R is a table, we can compute $\mathbf{pr}(R, J)$ by simply deleting all the graph table columns that are not in J . See 21.2.4 for example. This possibility is a direct consequence of the projection definition 21.2.2.
- ♣ R being a table does in general not mean that $\neg R$ is a table, too. So the coprojection definition $\mathbf{cpj}(R, J) := \neg \mathbf{pr}(\neg R, J)$ is not a recipe for a graph table method. Not in general, anyway.
- ♣ If R is a completely finite relation, it can be represented by boolean and double tables and we generate both $\mathbf{pr}(R, J)$ and $\mathbf{cpj}(R, J)$ according to the method in 21.1.4. The justification and formal expression for this method is given in 21.2.9 below.

21.2.4 Graph table representation for the projection

If $R = [X, \Gamma]$ is a table, we can represent it by its graph table. For example

$$R = \begin{array}{c|c|c|c} a : \mathbb{N} & b : \mathbb{N} & c : \mathbb{N} & d : \mathbb{N} \\ \hline 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ \hline 1 & 1 & 3 & 3 \\ 5 & 5 & 7 & 7 \\ 9 & 9 & 11 & 11 \\ 2 & 2 & 4 & 4 \\ 6 & 6 & 8 & 8 \\ 10 & 10 & 12 & 12 \end{array}$$

If we want to compute $\mathbf{pr}(R, J)$ with $J = \{b, d\}$, we delete all the columns in this graph table, except the ones for b and d :

$$\mathbf{pr}(R, J) = \begin{array}{c|c} b : \mathbb{N} & d : \mathbb{N} \\ \hline 2 & 4 \\ 6 & 8 \\ 10 & 12 \\ \hline 1 & 3 \\ 5 & 7 \\ 9 & 11 \\ 2 & 4 \\ 6 & 8 \\ 10 & 12 \end{array}$$

There are some redundancies, and after removing them we obtain

$$\mathbf{pr}(R, J) = \begin{array}{c|c} b : \mathbb{N} & d : \mathbb{N} \\ \hline 2 & 4 \\ 6 & 8 \\ 10 & 12 \\ \hline 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{array}$$

21.2.5 Lemma

Let $X = [X_i | i \in I]$ be a schema, $R = [X, \Gamma]$ a relation and $J \subseteq I$. Then

$$\mathbf{cpj}(R, J) = \begin{bmatrix} \mathbf{pr}(X, J) \\ \otimes \mathbf{pr}(X, J) \setminus \mathbf{pr}(\otimes X \setminus \Gamma, J) \end{bmatrix}$$

21.2.6 Proof of 21.2.5

$$\begin{aligned} & \mathbf{cpj}(R, J) \\ &= \neg \mathbf{pr}(\neg R, J) && \text{def. 21.2.2 of the coprojection} \\ &= \neg \mathbf{pr} \left(\begin{bmatrix} X \\ \otimes X \setminus \Gamma \end{bmatrix}, J \right) && \text{def. 19.4.1 of } \neg \end{aligned}$$

$$\begin{aligned} &= \neg \begin{bmatrix} \mathbf{pr}(X, J) \\ \mathbf{pr}(\otimes X \setminus \Gamma, J) \end{bmatrix} && \text{def. 21.2.2 of the projection} \\ &= \begin{bmatrix} \mathbf{pr}(X, J) \\ \otimes \mathbf{pr}(X, J) \setminus \mathbf{pr}(\otimes X \setminus \Gamma, J) \end{bmatrix} && \text{def. 19.4.1 of } \neg, \text{ again} \end{aligned}$$

21.2.7 Lemma

Let $X = [X_i | i \in I]$ be a proper schema, $R = [X, \Gamma]$ a relation and $J \subseteq I$. We put $Y := \mathbf{pr}(X, J)$. Then

$$\begin{aligned} (1) \quad \mathbf{pr}(R, J) &= \begin{bmatrix} Y \\ y \rightsquigarrow \exists z \in \otimes(X \setminus Y) . y \dot{\vee} z \in \Gamma \end{bmatrix} \\ (2) \quad \mathbf{cpj}(R, J) &= \begin{bmatrix} Y \\ y \rightsquigarrow \forall z \in \otimes(X \setminus Y) . y \dot{\vee} z \in \Gamma \end{bmatrix} \end{aligned}$$

21.2.8 Proof of 21.2.7

Let $Y := \mathbf{pr}(X, J)$.

(1) There is

$$\otimes X = \otimes(Y \dot{\vee} (X \setminus Y)) = \{y \dot{\vee} z \mid y \in \otimes Y, z \in \otimes(X \setminus Y)\}$$

so that

$$\begin{aligned} & \mathbf{pr}(\Gamma, J) \\ &= \{\mathbf{pr}(x, J) \mid x \in \otimes X, x \in \Gamma\} \\ &= \{\mathbf{pr}(y \dot{\vee} z, J) \mid y \in \otimes Y, z \in \otimes(X \setminus Y), y \dot{\vee} z \in \Gamma\} \\ &= \{y \mid y \in \otimes Y, \exists z \in \otimes(X \setminus Y) . y \dot{\vee} z \in \Gamma\} \\ & \qquad \qquad \qquad \text{because } \mathbf{pr}(y \dot{\vee} z, J) = y \end{aligned}$$

Thus

$$\mathbf{pr}(R, J) = \begin{bmatrix} Y \\ y \rightsquigarrow \exists z \in \otimes(X \setminus Y) . y \dot{\vee} z \in \Gamma \end{bmatrix}$$

(2) We have

$$\begin{aligned} & \mathbf{cpj}(R, J) \\ &= \neg \mathbf{pr}(\neg R, J) \\ &= \neg \mathbf{pr} \left(\begin{bmatrix} X \\ \otimes X \setminus \Gamma \end{bmatrix}, J \right) \\ &= \neg \left[y \rightsquigarrow \exists z \in \otimes(X \setminus Y) . y \dot{\vee} z \in (\otimes X \setminus \Gamma) \right] && \text{due to (1)} \\ &= \neg \left[y \rightsquigarrow \neg \forall z \in \otimes(X \setminus Y) . y \dot{\vee} z \in \Gamma \right] \\ &= \left[y \rightsquigarrow \neg \neg \forall z \in \otimes(X \setminus Y) . y \dot{\vee} z \in \Gamma \right] \\ &= \begin{bmatrix} Y \\ y \rightsquigarrow \forall z \in \otimes(X \setminus Y) . y \dot{\vee} z \in \Gamma \end{bmatrix} \end{aligned}$$

21.2.9 Lemma

Let $X = [X_i | i \in I]$ be a proper schema, $R = [X, \Gamma]$ a relation and $J \subseteq I$. We put $Y := \mathbf{pr}(X, J)$. Then

$$(1) \chi_{\mathbf{pr}(R, J)} = \left[\begin{array}{c} \otimes Y \longrightarrow \mathbb{B} \\ y \mapsto \bigvee_{\mathbb{B}} \{ \chi_{R(y \dot{\vee} z)} \mid z \in \otimes(X \setminus Y) \} \end{array} \right]$$

$$(2) \chi_{\mathbf{cpj}(R, J)} = \left[\begin{array}{c} \otimes Y \longrightarrow \mathbb{B} \\ y \mapsto \bigwedge_{\mathbb{B}} \{ \chi_{R(y \dot{\vee} z)} \mid z \in \otimes(X \setminus Y) \} \end{array} \right]$$

21.2.10 Proof of 21.2.9

These are the characteristic functions of the relations from 21.2.7.

21.3 Properties of projections and coprojections

21.3.1 Lemma

Let $X = [X_i | i \in I]$ be a schema, $R = [X, \Gamma]$ a relation, and $J \subseteq I$. Then

- (1) $\mathbf{x}(\mathbf{pr}(R, J)) = \mathbf{pr}(\mathbf{x}(R), J)$
- (2) $\mathbf{x}(\mathbf{cpj}(R, J)) = \mathbf{pr}(\mathbf{x}(R), J)$
- (3) $\mathbb{@}(\mathbf{pr}(R, J)) = J$
- (4) $\mathbb{@}(\mathbf{cpj}(R, J)) = J$
- (5) $\mathbf{gr}(\mathbf{pr}(R, J)) = \mathbf{pr}(\mathbf{gr}(R), J)$
- (6) $\mathbf{gr}(\mathbf{cpj}(R, J)) = \otimes \mathbf{pr}(X, J) \setminus \mathbf{pr}(\otimes X \setminus \mathbf{gr}(R), J)$

21.3.2 Proof of 21.3.1

Statements (1)–(5) are immediate consequences of 21.2.2 and (6) derives from 21.2.5.

21.3.3 Lemma

Let $X = [X_i | i \in I]$ be a proper schema and $R = [X, \Gamma]$ a relation.

- (1) There is
 - (a) $\mathbf{pr}(R, I) = R$ (neutrality)
 - (b) $\mathbf{cpj}(R, I) = R$ (neutrality)

- (2) If $J \subseteq I$, then
 - (a) $\mathbf{pr}(\mathbf{pr}(R, J), J) = \mathbf{pr}(R, J)$ (idempotency)
 - (b) $\mathbf{cpj}(\mathbf{cpj}(R, J), J) = \mathbf{cpj}(R, J)$ (idempotency)

- (3) There is
 - (a) $\mathbf{pr}(R, \emptyset) = \begin{cases} \perp & \text{if } R \text{ is empty} \\ \top & \text{else} \end{cases}$ (empty criterion)
 - (b) $\mathbf{cpj}(R, \emptyset) = \begin{cases} \top & \text{if } R \text{ is full} \\ \perp & \text{else} \end{cases}$ (full criterion)

- (4) If I_1, I_2 make a partition of I , i.e. $I_1 \cap I_2 = \emptyset$ and $I_1 \cup I_2 = I$, then
 - (a) $R \subseteq \mathbf{pr}(R, I_1) \odot \mathbf{pr}(R, I_2)$
 - (b) $R \supseteq \mathbf{cpj}(R, I_1) \odot \mathbf{cpj}(R, I_2)$

- (5) As a generalization of (4) we have: if $[I_k | k \in K]$ is a class partition of I , then
 - (a) $R \subseteq \odot [\mathbf{pr}(R, I_k) | k \in K]$
 - (b) $R \supseteq \odot [\mathbf{cpj}(R, I_k) | k \in K]$

- (6) If $J_1 \subseteq J_2 \subseteq I$ then
 - (a) $\mathbf{pr}(\mathbf{pr}(R, J_2), J_1) = \mathbf{pr}(R, J_1)$
 - (b) $\mathbf{cpj}(\mathbf{cpj}(R, J_2), J_1) = \mathbf{cpj}(R, J_1)$

- (7) If $J_1 \subseteq J_2 \subseteq I$ and $Y := \mathbf{pr}(X, J_2 \setminus J_1)$, then
 - (a) $\mathbf{pr}(R, J_1) \odot \top_Y \supseteq \mathbf{pr}(R, J_2)$
 - (b) $\mathbf{cpj}(R, J_1) \odot \top_Y \subseteq \mathbf{cpj}(R, J_2)$

- (8) If $J_1 \subseteq J_2 \subseteq I$, then
 - (a) $\mathbf{pr}(R, J_2) \sqsubseteq \mathbf{pr}(R, J_1)$ (antitony)
 - (b) $\mathbf{cpj}(R, J_1) \sqsubseteq \mathbf{cpj}(R, J_2)$ (isotony)

- (9) If $J \subseteq I$, then
 - (a) $R \sqsubseteq \mathbf{pr}(R, J)$ (supervalence of projection)
 - (b) $R \supseteq \mathbf{cpj}(R, J)$ (subvalence of coprojection)
 - (c) $\mathbf{cpj}(R, J) \subseteq \mathbf{pr}(R, J)$

- (10) If $J \subseteq I$, $Y := \mathbf{pr}(X, J)$ and $S = [Y, \Sigma]$ is a relation,
 - (a) $R \subseteq S$ iff $\mathbf{pr}(R, J) \subseteq S$
 - (b) $S \subseteq R$ iff $S \subseteq \mathbf{cpj}(R, J)$

- (11) If $J \subseteq I$ and $S = [X, \Sigma]$ is a relation, then
 - (a) $R \subseteq S$ implies $\mathbf{pr}(R, J) \subseteq \mathbf{pr}(S, J)$
 - (b) $R \subseteq S$ implies $\mathbf{cpj}(R, J) \subseteq \mathbf{cpj}(S, J)$

21.3.4 Proof of 21.3.3

- (1) There is

$$(a) \quad \mathbf{pr}(R, I) = \left[\begin{array}{c} \mathbf{pr}(X, I) \\ \{ \mathbf{pr}(x, I) \mid x \in \Gamma \} \end{array} \right] = R$$

- (b) and

$$\begin{aligned} & \mathbf{cpj}(R, I) \\ &= \left[\begin{array}{c} \mathbf{pr}(X, I) \\ \otimes \mathbf{pr}(X, I) \setminus \mathbf{pr}(\otimes X \setminus \Gamma, I) \end{array} \right] \quad \text{due to 21.2.5} \\ &= \left[\begin{array}{c} X \\ \otimes X \setminus (\otimes X \setminus \Gamma) \end{array} \right] \\ &= R \end{aligned}$$

- (2) For $J \subseteq I$ holds
 - (a) $\mathbb{@}(\mathbf{pr}(R, J)) = J$, so we can apply (1)(a) and obtain $\mathbf{pr}(\mathbf{pr}(R, J), J) = \mathbf{pr}(R, J)$.
 - (b) similar proof
- (3) Recall, that $R = [X, \Gamma]$ is empty, if $\Gamma = \emptyset$, i.e. if $R = \perp_X$, and full, if $\Gamma = \otimes X$, i.e. if $R = \top_X$.
 - (a) There is $\mathbf{pr}(\xi, \emptyset) = \langle \rangle$, for every record ξ . So

$$\{\mathbf{pr}(x, \emptyset) \mid x \in \Gamma\} = \begin{cases} \{\langle \rangle\} & \text{if } \Gamma \neq \emptyset \\ \emptyset & \text{if } \Gamma = \emptyset \end{cases}$$

Thus

$$\begin{aligned} \mathbf{pr}(R, \emptyset) &= \begin{bmatrix} \mathbf{pr}(X, \emptyset) \\ \{\mathbf{pr}(x, \emptyset) \mid x \in \Gamma\} \end{bmatrix} \\ &= \begin{cases} [\langle \rangle, \emptyset] = \perp & \text{if } R \text{ is empty} \\ [\langle \rangle, \{\langle \rangle\}] = \top & \text{else} \end{cases} \end{aligned}$$

(b) We have

$$\begin{aligned} \mathbf{cpj}(R, \emptyset) &= \begin{bmatrix} \mathbf{pr}(X, \emptyset) \\ \otimes \mathbf{pr}(X, \emptyset) \setminus \mathbf{pr}(\otimes X \setminus \Gamma, \emptyset) \end{bmatrix} && \text{due to 21.2.5} \\ &= \begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \setminus \{\mathbf{pr}(x, \emptyset) \mid x \in \otimes X \setminus \Gamma\} \end{bmatrix} \\ &= \begin{cases} \begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \setminus \emptyset \end{bmatrix} = \top & \text{if } \Gamma = \otimes X \\ \begin{bmatrix} \langle \rangle \\ \{\langle \rangle\} \setminus \{\langle \rangle\} \end{bmatrix} = \perp & \text{if } \Gamma \neq \otimes X \end{cases} \end{aligned}$$

(4) Let $X_1 := \mathbf{pr}(X, I_1)$ and $X_2 := \mathbf{pr}(X, I_2)$, so $X = X_1 \dot{\vee} X_2$.

(a) There is

$$\begin{aligned} \Gamma &= \{\mathbf{pr}(\xi, I_1) \dot{\vee} \mathbf{pr}(\xi, I_2) \mid \xi \in \Gamma\} \\ &\subseteq \{\mathbf{pr}(\xi_1, I_1) \dot{\vee} \mathbf{pr}(\xi_2, I_2) \mid \xi_1, \xi_2 \in \Gamma\} \\ &= \{\mathbf{pr}(\xi_1, I_1) \mid \xi_1 \in \Gamma\} \odot \{\mathbf{pr}(\xi_2, I_2) \mid \xi_2 \in \Gamma\} \\ &= \mathbf{pr}(\Gamma, I_1) \odot \mathbf{pr}(\Gamma, I_2) \end{aligned}$$

so that

$$\begin{aligned} R &= \begin{bmatrix} X \\ \Gamma \end{bmatrix} \\ &\subseteq \begin{bmatrix} X_1 \dot{\vee} X_2 \\ \mathbf{pr}(\Gamma, I_1) \odot \mathbf{pr}(\Gamma, I_2) \end{bmatrix} \\ &= \begin{bmatrix} X_1 \\ \mathbf{pr}(\Gamma, I_1) \end{bmatrix} \odot \begin{bmatrix} X_2 \\ \mathbf{pr}(\Gamma, I_2) \end{bmatrix} \\ &= \mathbf{pr}(R, I_1) \odot \mathbf{pr}(R, I_2) \end{aligned}$$

(b) We have

$$\begin{aligned} \mathbf{cpj}(R, I_1) \odot \mathbf{cpj}(R, I_2) &= \left(\begin{bmatrix} X_1 \\ x_1 \rightsquigarrow \forall y_2 \in \otimes X_2 . x_1 \dot{\vee} y_2 \in \Gamma \end{bmatrix} \odot \begin{bmatrix} X_2 \\ x_2 \rightsquigarrow \forall y_1 \in \otimes X_1 . y_1 \dot{\vee} x_2 \in \Gamma \end{bmatrix} \right) \\ &= \begin{bmatrix} X \\ \left\{ x_1 \dot{\vee} x_2 \mid \begin{array}{l} x_1 \in \otimes X_1, x_2 \in \otimes X_2, \\ \forall y_2 \in \otimes X_2 . x_1 \dot{\vee} y_2 \in \Gamma, \\ \forall y_1 \in \otimes X_1 . y_1 \dot{\vee} x_2 \in \Gamma \end{array} \right\} \end{bmatrix} && \text{due to 21.2.7(2)} \end{aligned}$$

$$\begin{aligned} &\subseteq \begin{bmatrix} X \\ \left\{ x_1 \dot{\vee} x_2 \mid \begin{array}{l} x_1 \in \otimes X_1, x_2 \in \otimes X_2, \\ x_1 \dot{\vee} x_2 \in \Gamma \end{array} \right\} \end{bmatrix} \\ &= \begin{bmatrix} X \\ \Gamma \end{bmatrix} \\ &= R \end{aligned}$$

(5) For every $k \in K$ let $X_k := \mathbf{pr}(X, I_k)$.

(a) Recall, that for every $\xi \in \otimes X$ holds

$$\xi = \dot{\vee} \{\mathbf{pr}(\xi, I_k) \mid k \in K\}$$

Now

$$\begin{aligned} \Gamma &= \left\{ \dot{\vee}_{k \in K} \mathbf{pr}(\xi, I_k) \mid \xi \in \Gamma \right\} \\ &\subseteq \left\{ \dot{\vee}_{k \in K} \mathbf{pr}(\xi_k, I_k) \mid \xi_k \in \Gamma \text{ for each } k \in K \right\} \\ &= \odot \{\mathbf{pr}(\xi_k, I_k) \mid \xi_k \in \Gamma, k \in K\} \\ &= \odot \{\mathbf{pr}(\Gamma, I_k) \mid k \in K\} \end{aligned}$$

so that

$$\begin{aligned} R &= \begin{bmatrix} X \\ \Gamma \end{bmatrix} \\ &= \begin{bmatrix} \dot{\vee}_{k \in K} X_k \\ \odot \{\mathbf{pr}(\Gamma, I_k) \mid k \in K\} \end{bmatrix} \\ &= \odot \{\mathbf{pr}(R, I_k) \mid k \in K\} \end{aligned}$$

(b) Let $Y_k := X \setminus X_k$ for each $k \in K$, so that $X = X_k \dot{\vee} Y_k$.

There is

$$\begin{aligned} &\left\{ [x_k \mid k \in K] \mid \begin{array}{l} \forall k \in K . x_k \in \otimes X_k \\ \forall k \in K . \forall y_k \in \otimes Y_k . x_k \dot{\vee} y_k \in \Gamma \end{array} \right\} \\ &\subseteq \left\{ [x_k \mid k \in K] \mid \begin{array}{l} \forall k \in K . x_k \in \otimes X_k \\ \dot{\vee} [x_k \mid k \in K] \in \Gamma \end{array} \right\} \end{aligned}$$

so that

$$\begin{aligned} &\odot \{\mathbf{cpj}(R, I_k) \mid k \in K\} \\ &= \odot \left[\begin{bmatrix} X_k \\ \left\{ \frac{x_k \in \otimes X_k}{\forall y_k \in \otimes Y_k . x_k \dot{\vee} y_k \in \Gamma} \right\} \right] \Big|_{k \in K} \end{aligned} && \text{due to 21.2.7(2)} \end{aligned}$$

$$\begin{aligned} &= \begin{bmatrix} X \\ \left\{ \frac{\dot{\vee} [x_k \mid k \in K]}{\forall k \in K . x_k \in \otimes X_k, \forall k \in K . \forall y_k \in \otimes Y_k . x_k \dot{\vee} y_k \in \Gamma} \right\} \end{bmatrix} \\ &\subseteq \begin{bmatrix} X \\ \left\{ \dot{\vee} [x_k \mid k \in K] \mid \begin{array}{l} \forall k \in K . x_k \in \otimes X_k, \\ \dot{\vee} [x_k \mid k \in K] \in \Gamma \end{array} \right\} \end{bmatrix} \\ &= \begin{bmatrix} X \\ \Gamma \end{bmatrix} \\ &= R \end{aligned}$$

(6) For every record $\xi = [\xi_i \mid i \in I] \in \otimes X$ holds

$$\begin{aligned}
\mathbf{pr}(\mathbf{pr}(\xi, J_2), J_1) &= \mathbf{pr}(\mathbf{pr}([\xi_i | i \in I], J_2), J_1) \\
&= \mathbf{pr}([\xi_i | i \in (I \cap J_2)], J_1) \\
&= [\xi_i | i \in ((I \cap J_2) \cap J_1)] \\
&= [\xi_i | i \in J_1] \\
&= \mathbf{pr}(\xi, J_1)
\end{aligned}$$

It follows, that for every $\Sigma \subseteq \otimes X$,

$$\mathbf{pr}(\mathbf{pr}(\Sigma, J_2), J_1) = \mathbf{pr}(\Sigma, J_1)$$

(a) We obtain

$$\begin{aligned}
\mathbf{pr}(\mathbf{pr}(R, J_2), J_1) &= \mathbf{pr}\left(\left[\begin{array}{c} \mathbf{pr}(X, J_2) \\ \mathbf{pr}(\Gamma, J_2) \end{array}\right], J_1\right) \\
&= \left[\begin{array}{c} \mathbf{pr}(\mathbf{pr}(X, J_2), J_1) \\ \mathbf{pr}(\mathbf{pr}(\Gamma, J_2), J_1) \end{array}\right] \\
&= \left[\begin{array}{c} \mathbf{pr}(X, J_1) \\ \mathbf{pr}(\Gamma, J_1) \end{array}\right] \\
&= \mathbf{pr}(R, J_1)
\end{aligned}$$

(b) By applying 21.2.7(2) twice, we obtain

$$\begin{aligned}
&\mathbf{cpj}(\mathbf{cpj}(R, J_2), J_1) \\
&= \mathbf{cpj}\left(\left[\begin{array}{c} \mathbf{pr}(X, J_2) \\ y \rightsquigarrow \forall z \in \otimes \mathbf{pr}(X, I \setminus J_2) . \\ y \dot{\vee} z \in \Gamma \end{array}\right], J_1\right) \\
&= \left[\begin{array}{c} \mathbf{pr}(\mathbf{pr}(X, J_2), J_1) \\ x \rightsquigarrow \forall y' \in \otimes \mathbf{pr}(X, J_2 \setminus J_1) . \\ \forall z \in \otimes \mathbf{pr}(X, I \setminus J_2) . \\ x \dot{\vee} y' \dot{\vee} z \in \Gamma \end{array}\right] \\
&= \left[\begin{array}{c} \mathbf{pr}(X, J_1) \\ x \rightsquigarrow \forall x' \in \otimes \mathbf{pr}(X, I \setminus J_1) . x \dot{\vee} x' \in \Gamma \end{array}\right] \\
&= \mathbf{pr}(R, J_1)
\end{aligned}$$

(7) J_1 and $J_2 \setminus J_1$ make a partition of J_2 , so we can apply (4) as follows

(a) We have

$$\begin{aligned}
&\mathbf{pr}(R, J_2) \\
&\subseteq \mathbf{pr}(\mathbf{pr}(R, J_2), J_1) \odot \mathbf{pr}(\mathbf{pr}(R, J_2), J_2 \setminus J_1) \quad \text{due to (4)(a)} \\
&= \mathbf{pr}(R, J_1) \odot \mathbf{pr}(R, J_2 \setminus J_1) \quad \text{due to (6)(a)} \\
&\subseteq \mathbf{pr}(R, J_1) \odot \top_Y \quad \text{because } \mathbf{pr}(R, J_2 \setminus J_1) \subseteq \top_Y
\end{aligned}$$

(b) Let $X := \mathbf{pr}(X, J_1)$, $X_2 := \mathbf{pr}(X, J_2)$ and $Z := \mathbf{pr}(X, I \setminus J_2)$, so that $X_2 = X_1 \dot{\vee} Y$ and $X = X_2 \dot{\vee} Z = X_1 \dot{\vee} Y \dot{\vee} Z$. Now let $\xi \in \otimes X_2$, then

$$\begin{aligned}
&\xi \in \mathbf{cpj}(R, J_1) \odot \top_Y \\
&\Leftrightarrow \xi \in \left\{ x \left| \begin{array}{l} x \in \otimes X_1, \\ \forall x' \in \otimes (Y \dot{\vee} Z) . x \dot{\vee} x' \in \Gamma \end{array} \right. \right\} \odot \otimes Y \\
&\quad \text{due to 21.2.7(2) and def. 19.3.1 of } \top_Y \\
&\Leftrightarrow \xi \in \left\{ x \dot{\vee} y \left| \begin{array}{l} x \in \otimes X_1, y \in \otimes Y, \\ \forall x' \in \otimes (Y \dot{\vee} Z) . x \dot{\vee} x' \in \Gamma \end{array} \right. \right\} \\
&\Leftrightarrow \xi \in \left\{ x \dot{\vee} y \left| \begin{array}{l} x \in \otimes X_1, y \in \otimes Y, \\ \forall y' \in \otimes Y . \forall z \in \otimes Z . x \dot{\vee} y' \dot{\vee} z \in \Gamma \end{array} \right. \right\} \\
&\Rightarrow \xi \in \left\{ x \dot{\vee} y \left| \begin{array}{l} x \in \otimes X_1, y \in \otimes Y, \\ \forall z \in \otimes Z . z \dot{\vee} y \dot{\vee} z \in \Gamma \end{array} \right. \right\} \\
&\Leftrightarrow \xi \in \{u \in \otimes X_2 \mid \forall z \in \otimes Z . x \dot{\vee} z \in \Gamma\} \\
&\Leftrightarrow \xi \in \mathbf{cpj}(R, J_2) \quad \text{due to 21.2.7(2) again}
\end{aligned}$$

and so

$$\mathbf{cpj}(R, J_1) \odot \top_Y \subseteq \mathbf{cpj}(R, J_2)$$

(8) We put $X_1 := \mathbf{pr}(X, J_1)$, $X_2 := \mathbf{pr}(X, J_2)$ and $Y := \mathbf{pr}(X, J_2 \setminus J_1)$, so $X_2 = X_1 \dot{\vee} Y$.

(a) There is

$$\begin{aligned}
&\mathbf{pr}(R, J_2) \sqsubseteq \mathbf{pr}(R, J_1) \\
&\Leftrightarrow \mathbf{pr}(R, J_2) \parallel X_2 \subseteq \mathbf{pr}(R, J_1) \parallel X_2 \quad \text{def. 20.2.2 of } \sqsubseteq \\
&\Leftrightarrow \mathbf{pr}(R, J_2) \subseteq \mathbf{pr}(R, J_1) \parallel Y \quad \text{due to 20.1.3(1)} \\
&\Leftrightarrow \mathbf{pr}(R, J_2) \subseteq \mathbf{pr}(R, J_1) \odot \top_Y \quad \text{due to ??(3)}
\end{aligned}$$

and the last statement is proved to be true in (7)(a).

(b) Similar to (a) we have

$$\begin{aligned}
\mathbf{cpj}(R, J_1) &\equiv \mathbf{cpj}(R, J_1) \parallel Y \\
&= \mathbf{cpj}(R, J_1) \odot \top_Y \\
&\subseteq \mathbf{cpj}(R, J_2)
\end{aligned}$$

(9) (a) and (b) are immediate consequences of (1) and (8):

$$R = \mathbf{pr}(R, I) \sqsubseteq \mathbf{pr}(R, J)$$

$$R = \mathbf{cpj}(R, I) \supseteq \mathbf{cpj}(R, J)$$

\sqsubseteq is transitive, so (a) and (b) entail $\mathbf{cpj}(R, J) \sqsubseteq \mathbf{pr}(R, J)$. And because both sides have the same resulting schema, $\mathbf{cpj}(R, J) \subseteq \mathbf{pr}(R, J)$.

(10) So let $J \subseteq I$, $Y := \mathbf{pr}(X, J)$ and $S = [Y, \Sigma]$ be given.

(a) We have

$$\begin{aligned}
&R \sqsubseteq S \\
&\Leftrightarrow R \parallel X \subseteq S \parallel X \quad \text{def. 20.2.2 of } \sqsubseteq \\
&\Leftrightarrow R \subseteq S \odot \top_{X \setminus Y} \quad \text{due to 20.1.3} \\
&\Leftrightarrow \left[\begin{array}{c} X \\ \Gamma \end{array}\right] \subseteq \left[\begin{array}{c} X \\ \Sigma \odot \otimes (X \setminus Y) \end{array}\right] \\
&\Leftrightarrow \forall x \in \otimes X . (x \in \Gamma \rightarrow x \in \Sigma \odot \otimes (X \setminus Y)) \\
&\Leftrightarrow \left(\forall y \in \otimes Y . \forall z \in \otimes (X \setminus Y) . \right. \\
&\quad \left. (y \dot{\vee} z \in \Gamma \rightarrow y \dot{\vee} z \in \Sigma \odot \otimes (X \setminus Y)) \right) \\
&\Leftrightarrow \forall y \in \otimes Y . (\exists z \in \otimes (X \setminus Y) . y \dot{\vee} z \in \Gamma) \rightarrow y \in \Sigma \\
&\quad \text{because } X \text{ is proper and so is } X \setminus Y \\
&\Leftrightarrow \forall y \in \otimes Y . (y \in \mathbf{pr}(R, J) \rightarrow y \in S) \\
&\Leftrightarrow \mathbf{pr}(R, J) \subseteq S
\end{aligned}$$

(b) Similar to (a) we obtain

$$\begin{aligned}
&S \subseteq R \\
&\Leftrightarrow \left[\begin{array}{c} Y \\ \Sigma \odot \otimes (X \setminus Y) \end{array}\right] \subseteq \left[\begin{array}{c} X \\ \Gamma \end{array}\right] \\
&\Leftrightarrow \forall x \in \otimes X . (x \in \Sigma \odot \otimes (X \setminus Y) \rightarrow x \in \Gamma) \\
&\Leftrightarrow \left(\forall y \in \otimes Y . \forall z \in \otimes (X \setminus Y) . \right. \\
&\quad \left. (y \dot{\vee} z \in \Gamma \odot \otimes (X \setminus Y) \rightarrow y \dot{\vee} z \in \Gamma) \right) \\
&\Leftrightarrow \forall y \in \otimes Y . (y \in \Sigma \rightarrow \forall z \in \otimes (X \setminus Y) . y \dot{\vee} z \in \Gamma) \\
&\Leftrightarrow \left[\begin{array}{c} Y \\ \Sigma \end{array}\right] \subseteq \left[\begin{array}{c} Y \\ y \rightsquigarrow \forall z \in \otimes (X \setminus Y) . y \dot{\vee} z \in \Gamma \end{array}\right] \\
&\Leftrightarrow S \subseteq \mathbf{cpj}(R, J) \quad \text{due to 21.2.7(2)}
\end{aligned}$$

(11) Now $J \subseteq I$ and $S = [X, \Sigma]$.

(a) $R \subseteq S$ means that $\Gamma \subseteq \Sigma$, and so

$$\begin{aligned} \mathbf{pr}(R, J) &= \left[\begin{array}{c} \mathbf{pr}(X, J) \\ \{\mathbf{pr}(x, J) \mid x \in \Gamma\} \end{array} \right] \\ &\subseteq \left[\begin{array}{c} \mathbf{pr}(X, J) \\ \{\mathbf{pr}(x, J) \mid x \in \Sigma\} \end{array} \right] \\ &= \mathbf{pr}(S, J) \end{aligned}$$

(b) $R \subseteq S$ is equivalent to $\neg S \subseteq \neg R$. Therefore, $\mathbf{pr}(\neg S, J) \subseteq \mathbf{pr}(\neg R, J)$, according to (a). And applying the same law again gives us $\neg \mathbf{pr}(\neg R, J) \subseteq \neg \mathbf{pr}(\neg S, J)$. In terms of definition 21.2.2, this is $\mathbf{cpj}(R, J) \subseteq \mathbf{cpj}(S, J)$.

21.3.5 Remark

The schema X in lemma 21.3.3 was supposed to be proper. For some of the statements this is essential.

Suppose, X is not proper, e.g.

$$X := \left[\begin{array}{c} a \mapsto \mathbb{N} \\ b \mapsto \emptyset \end{array} \right]$$

For such an improper schema the cartesian product $\otimes X$ is empty and $\mathbf{Rel}(X)$ has only one member $R = [X, \emptyset]$ with $R = \perp_X = \top_X$ and $\neg R = R$ (see 19.5.16).

But $Y := \mathbf{pr}(X, \{a\}) = [a \mapsto \mathbb{N}]$ is a proper schema with $\otimes Y = \{[a \mapsto n] \mid n \in \mathbb{N}\} \neq \emptyset$ and $\perp_Y \neq \top_Y$. The projection and coprojection of R onto $\{a\}$ are given by

$$\begin{aligned} \mathbf{pr}(R, \{a\}) &= \left[\begin{array}{c} \mathbf{pr}(X, \{a\}) \\ \mathbf{pr}(\emptyset, \{a\}) \end{array} \right] \\ &= [Y, \emptyset] \\ &= \perp_Y \\ \mathbf{cpj}(R, \{a\}) &= \neg \mathbf{pr}(\neg R, \{a\}) \\ &= \neg \mathbf{pr}(R, \{a\}) \\ &= \neg \perp_Y \\ &= \top_Y \end{aligned}$$

so that

$$\mathbf{cpj}(\perp_X, \{a\}) \not\subseteq \mathbf{pr}(\perp_X, \{a\})$$

in violation with 21.3.3(9)(c).

21.3.6 Lemma

Let $X = [X_i \mid i \in I]$ be a proper schema and $R, S \in \mathbf{Prel}(X)$. If $K := \textcircled{\@}(R) \cap \textcircled{\@}(S)$ then

$$R \subseteq S \quad \text{iff} \quad \mathbf{pr}(R, K) \subseteq \mathbf{cpj}(S, K)$$

21.3.7 Proof of 21.3.6

Recall from 5.5.11, that for all variables x, y , class symbols C, D , and formulas φ, ψ holds:

(a) If x does not occur free in ψ and y does not occur free in ϕ , then

$$\forall x \in C . \forall y \in D . (\varphi \rightarrow \psi) \Leftrightarrow (\exists x \in C . \varphi) \rightarrow (\forall y \in D . \psi)$$

For the full proof of the lemma, let us put

$$\begin{aligned} J_1 &:= \textcircled{\@}(R) & Y_1 &:= \mathbf{x}(R) & \Gamma_1 &:= \mathbf{gr}(R) \\ J_2 &:= \textcircled{\@}(S) & Y_2 &:= \mathbf{x}(S) & \Gamma_2 &:= \mathbf{gr}(S) \end{aligned}$$

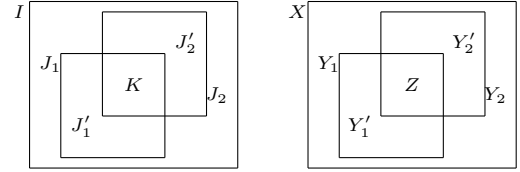
so that $K = J_1 \cap J_2$. Furthermore we put

$$Z := Y_1 \wedge Y_2$$

and

$$\begin{aligned} J'_1 &:= J_1 \setminus K & Y'_1 &:= Y_1 \setminus Z \\ J'_2 &:= J_2 \setminus K & Y'_2 &:= Y_2 \setminus Z \end{aligned}$$

The subclasses of I and the projections of X defined so far are displayed in the following two diagrams:



We derive

$$\begin{aligned} R &\subseteq S \\ \Leftrightarrow [Y_1, \Gamma_1] &\subseteq [Y_2, \Gamma_2] \\ \Leftrightarrow \left[\begin{array}{c} Y_1 \dot{\vee} Y'_2 \\ \Gamma_1 \otimes \otimes Y'_2 \end{array} \right] &\subseteq \left[\begin{array}{c} Y_2 \dot{\vee} Y'_1 \\ \Gamma_2 \otimes \otimes Y'_1 \end{array} \right] && \text{due to 20.1.3} \\ \Leftrightarrow \Gamma_1 \otimes \otimes Y'_2 &\subseteq \Gamma_2 \otimes \otimes Y'_1 \\ \Leftrightarrow \left(\forall x \in \otimes(Y_1 \dot{\vee} Y_2) . \right. && \left. (x \in (\Gamma_1 \otimes \otimes Y'_2) \rightarrow x \in (\Gamma_2 \otimes \otimes Y'_1)) \right) \\ \Leftrightarrow \left(\forall z \in \otimes Z . \forall y_1 \in \otimes Y'_1 . \forall y_2 \in \otimes Y'_2 . \right. && \left. \left((z \dot{\vee} y_1 \dot{\vee} y_2) \in (\Gamma_1 \otimes \otimes Y'_2) \right) \right. \\ && \left. \rightarrow (z \dot{\vee} y_1 \dot{\vee} y_2) \in (\Gamma_2 \otimes \otimes Y'_1) \right) \\ && \text{because } \otimes(Y_1 \dot{\vee} Y_2) = (\otimes Z) \otimes (\otimes Y'_1) \otimes (\otimes Y'_2) \\ \Leftrightarrow \left(\forall z \in \otimes Z . \forall y_1 \in \otimes Y'_1 . \forall y_2 \in \otimes Y'_2 . \right. && \left. ((z \dot{\vee} y_1) \in \Gamma_1 \rightarrow (z \dot{\vee} y_2) \in \Gamma_2) \right) \\ && \text{because } (z \dot{\vee} y_1 \dot{\vee} y_2) \in (\Gamma_1 \otimes \otimes Y'_2) \\ && \text{iff } (z \dot{\vee} y_1) \in \Gamma_1 \text{ and } (z \dot{\vee} y_1 \dot{\vee} y_2) \in (\Gamma_2 \otimes \otimes Y'_1) \\ && \text{iff } (z \dot{\vee} y_2) \in \Gamma_2 \\ \Leftrightarrow \forall z \in \otimes Z . \left(\begin{array}{c} (\exists y_1 \in \otimes Y'_1 . (z \dot{\vee} y_1) \in \Gamma_1) \\ \rightarrow (\forall y_2 \in \otimes Y'_2 . (z \dot{\vee} y_2) \in \Gamma_2) \end{array} \right) && \text{due to (a) above} \\ \Leftrightarrow \left(\{z \in \otimes Z \mid \exists y_1 \in \otimes Y'_1 . (z \dot{\vee} y_1) \in \Gamma_1\} \right. && \left. \subseteq \{z \in \otimes Z \mid \forall y_2 \in \otimes Y'_2 . (z \dot{\vee} y_2) \in \Gamma_2\} \right) \\ \Leftrightarrow \left(\left[\begin{array}{c} Z \\ z \rightsquigarrow \exists y_1 \in \otimes Y'_1 . (z \dot{\vee} y_1) \in \Gamma_1 \end{array} \right] \right. && \left. \subseteq \left[\begin{array}{c} Z \\ z \rightsquigarrow \forall y_2 \in \otimes Y'_2 . (z \dot{\vee} y_2) \in \Gamma_2 \end{array} \right] \right) \\ \Leftrightarrow \mathbf{pr}(R, K) &\subseteq \mathbf{cpj}(S, K) && \text{due to 21.2.7} \end{aligned}$$

21.3.8 Lemma

Let $X = [X_i | i \in I]$ be a proper schema, $J \subseteq I$ and $Y := \mathbf{pr}(X, J)$.

- (1) If $R \in \mathbf{Rel}(X)$ then
 - (a) $\mathbf{pr}(R, J) = \bigcap \{S \in \mathbf{Rel}(Y) \mid R \sqsubseteq S\}$
 - (b) $\mathbf{cpj}(R, J) = \bigcup \{S \in \mathbf{Rel}(Y) \mid S \sqsubseteq R\}$
- (2) If $R \in \mathbf{Rel}(Y)$ then
 - (a) $\bigcup \{S \in \mathbf{Rel}(X) \mid \mathbf{pr}(S, J) \subseteq R\} = R \odot \top_{(X \setminus Y)}$
 - (b) $\bigcap \{S \in \mathbf{Rel}(X) \mid R \subseteq \mathbf{cpj}(S, J)\} = R \odot \top_{(X \setminus Y)}$

21.3.9 Proof of 21.3.8

- (1) Let $R = [X, \Gamma]$.
 - (a) Let $R' := \mathbf{pr}(R, J)$ and $\mathcal{S} := \{S \in \mathbf{Rel}(Y) \mid R \sqsubseteq S\}$.
So we need to proof that $R' = \bigcap \mathcal{S}$.
There is $R' \in \mathbf{Rel}(Y)$ and $R \sqsubseteq \mathbf{pr}(R, J)$, due to 21.3.3(9)(a), so $R' \in \mathcal{S}$.
If $S' \in \mathcal{S}$, then $R \sqsubseteq S$ implies $R' \subseteq S$, due to 21.3.3(10)(a).
Thus R' is the minimal element of \mathcal{S} (with respect to the partial order \subseteq on $\mathbf{Rel}(Y)$) and therefore $R' = \bigcap \mathcal{S}$.
 - (b) Similar to (a) we proof $R' = \bigcup \mathcal{S}$, where this time $R' := \mathbf{cpj}(R, J)$ and $\mathcal{S} := \{S \in \mathbf{Rel}(Y) \mid S \sqsubseteq R\}$.
 $R' \in \mathbf{Rel}(Y)$ and $\mathbf{cpj}(R, J) \sqsubseteq R$, due to 21.3.3(9)(b), so $R' \in \mathcal{S}$.
If $S' \in \mathcal{S}$, then $S \sqsubseteq R$ implies $S \subseteq R'$, due to 21.3.3(10)(b).
Thus R' is the (\subseteq -) maximal element of \mathcal{S} and $R' = \bigcup \mathcal{S}$.
- (2) Now let $R = [Y, \Gamma]$.
 - (a) Let $\Gamma' := \Gamma \odot \otimes(X \setminus Y)$ and $\mathcal{S} := \{\Sigma \mid \Sigma \in \otimes X, \mathbf{pr}(\Sigma, J) \subseteq \Gamma\}$.
There is $\Gamma' \in \mathcal{S}$, because $\Gamma' \subseteq \otimes(Y \dot{\vee}(X \setminus Y)) = \otimes X$ and

$$\begin{aligned} \mathbf{pr}(\Gamma', J) &= \{\mathbf{pr}(\xi, J) \mid \xi \in (\Gamma \odot \otimes(X \setminus Y))\} \\ &= \{\mathbf{pr}(\xi_1 \dot{\vee} \xi_2, J) \mid \xi_1 \in \Gamma, \xi_2 \in \otimes(X \setminus Y)\} \\ &= \{\xi_1 \mid \xi_1 \in \Gamma, \xi_2 \in \otimes(X \setminus Y)\} \\ &= \Gamma \subseteq \Gamma' \end{aligned}$$

Furthermore, $\Sigma \in \mathcal{S}$ implies $\Sigma \subseteq \Gamma'$, because

$$\mathbf{pr}(\Sigma, J) = \{v \in \otimes Y \mid \exists \zeta \in \otimes(X \setminus Y) . v \dot{\vee} \zeta \in \Sigma\}$$

so that $\mathbf{pr}(\Sigma, J) \subseteq \Gamma$ implies $\Sigma \subseteq \Gamma \odot \otimes(X \setminus Y) = \Gamma'$.
From $\Gamma \in \mathcal{S}$ and ($\Sigma \in \mathcal{S}$ implies $\Sigma \subseteq \Gamma'$) we derive that $\Gamma' = \bigcup \mathcal{S}$.
So we obtain

$$\begin{aligned} &\bigcup \{S \in \mathbf{Rel}(X) \mid \mathbf{pr}(S, J) \subseteq \Gamma\} \\ &= \bigcup \left\{ \begin{bmatrix} X \\ \Sigma \end{bmatrix} \mid \mathbf{pr} \left(\begin{bmatrix} X \\ \Sigma \end{bmatrix}, J \right) \subseteq \begin{bmatrix} Y \\ \Gamma \end{bmatrix} \right\} \\ &= \bigcup \left\{ \begin{bmatrix} X \\ \Sigma \end{bmatrix} \mid \Sigma \subseteq \otimes X, \mathbf{pr}(\Sigma, J) \subseteq \Gamma \right\} \\ &= \left[\bigcup \{ \Sigma \mid \Sigma \subseteq \otimes X, \mathbf{pr}(\Sigma, J) \subseteq \Gamma \} \right] \\ &= \begin{bmatrix} X \\ \Gamma \odot \otimes(X \setminus Y) \end{bmatrix} && \text{because } \Gamma' = \bigcup \mathcal{S} \\ &= \begin{bmatrix} Y \dot{\vee}(X \setminus Y) \\ \Gamma \odot \otimes(X \setminus Y) \end{bmatrix} \\ &= R \odot \top_{(X \setminus Y)} && \text{due to 20.1.3} \end{aligned}$$

- (b) Proof is similar to (a).

21.3.10 Lemma

Let $X = [X | i \in I]$ be a proper schema, $J \subseteq I$ and $\mathcal{R} = \{\mathcal{R}_k \mid k \in K\} \subseteq \mathbf{Rel}(X)$, then

- (1) $\mathbf{pr}(\bigcap \mathcal{R}, J) \subseteq \bigcap_{k \in K} \mathbf{pr}(\mathcal{R}_k, J)$
- (2) $\mathbf{pr}(\bigcup \mathcal{R}, J) = \bigcup_{k \in K} \mathbf{pr}(\mathcal{R}_k, J)$
- (3) $\mathbf{cpj}(\bigcap \mathcal{R}, J) = \bigcap_{k \in K} \mathbf{cpj}(\mathcal{R}_k, J)$
- (4) $\mathbf{cpj}(\bigcup \mathcal{R}, J) \supseteq \bigcup_{k \in K} \mathbf{cpj}(\mathcal{R}_k, J)$

21.3.11 Proof of 21.3.10

Let $Y := \mathbf{pr}(X, J)$ and $\mathcal{R}_k = [X, \Gamma_k]$, for each $k \in K$.

- (1) For every $k \in K$ holds $\bigcap \mathcal{R} \subseteq \mathcal{R}_k$. Applying 21.3.3(11)(a), it follows that $\mathbf{pr}(\bigcap \mathcal{R}, J) \subseteq \mathbf{pr}(\mathcal{R}_k, J)$, for every $k \in K$, so that $\mathbf{pr}(\bigcap \mathcal{R}, J) \subseteq \bigcap \{\mathbf{pr}(\mathcal{R}_k, J) \mid k \in K\}$.

- (2) We have

$$\begin{aligned} &\mathbf{pr}(\bigcup \mathcal{R}, J) \\ &= \mathbf{pr} \left(\begin{bmatrix} X \\ \bigcup_{k \in K} \Gamma_k \end{bmatrix}, J \right) && \text{def. 19.5.3 of } \bigcup \\ &= \left[\begin{bmatrix} Y \\ \{\mathbf{pr}(x, J) \mid x \in \bigcup \{\Gamma_k \mid k \in K\}\} \end{bmatrix} \right] && \text{def. 21.2.2} \\ &= \left[\bigcup_{k \in K} \begin{bmatrix} Y \\ \{\mathbf{pr}(x, J) \mid x \in \Gamma_k\} \end{bmatrix} \right] \\ &= \bigcup_{k \in K} \begin{bmatrix} Y \\ \mathbf{pr}(\Gamma_k, J) \end{bmatrix} \\ &= \bigcup_{k \in K} \mathbf{pr}(\mathcal{R}_k, J) \end{aligned}$$

- (3) We have

$$\begin{aligned} &\mathbf{cpj}(\bigcap \mathcal{R}, J) \\ &= \neg \mathbf{pr}(\neg \bigcap \mathcal{R}, J) && \text{def. 21.2.2 of the coprojection} \\ &= \neg \mathbf{pr} \left(\bigcup_{k \in K} (\neg \mathcal{R}_k), J \right) && \text{de Morgan's law} \\ &= \neg \bigcup_{k \in K} \mathbf{pr}(\neg \mathcal{R}_k, J) && \text{due to (2)} \\ &= \bigcap_{k \in K} \neg \mathbf{pr}(\neg \mathcal{R}_k, J) && \text{de Morgan's law} \\ &= \bigcap_{k \in K} \mathbf{cpj}(\mathcal{R}_k, J) && \text{again def. 21.2.2} \end{aligned}$$

- (4) For every $k \in K$ holds $\mathcal{R}_k \subseteq \bigcup \mathcal{R}$. Applying 21.3.3(11)(b), it follows that $\mathbf{cpj}(\mathcal{R}_k, J) \subseteq \mathbf{cpj}(\bigcup \mathcal{R}, J)$, for every $k \in K$, so that $\bigcup \{\mathbf{cpj}(\mathcal{R}_k, J) \mid k \in K\} \subseteq \mathbf{cpj}(\bigcup \mathcal{R}, J)$.

21.3.12 Lemma

Let $X = [X_i | i \in I]$ be a proper schema, $R = [X, \Gamma]$ a relation and $J \subseteq I$.

- (1) $\mathbf{pr}(\neg R, J) = \neg \mathbf{cpj}(R, J)$
- (2) $\mathbf{cpj}(\neg R, J) = \neg \mathbf{pr}(R, J)$

21.3.13 Proof of 21.3.12

- (1) We have

$$\begin{aligned} &\mathbf{pr}(\neg R, J) \\ &= \neg \mathbf{pr}(\neg \neg R, J) && \text{due to 19.4.6} \\ &= \neg \mathbf{cpj}(\neg \neg R, J) && \text{def. 21.2.2} \end{aligned}$$

$$= \neg \mathbf{cpj}(R, J) \quad \text{due to 19.4.6, again}$$

$$(2) \text{ Similarly, } \mathbf{cpj}(\neg R, J) = \neg \mathbf{pr}(\neg \neg R, J) = \neg \mathbf{pr}(R, J)$$

21.3.14 Lemma

For every proper schema $X = [X_i | i \in I]$ and every $J \subseteq I$ holds:

$$(1) \mathbf{pr}(\perp_X, J) = \perp_{\mathbf{pr}(X, J)}$$

$$(2) \mathbf{cpj}(\perp_X, J) = \perp_{\mathbf{pr}(X, J)}$$

$$(3) \mathbf{pr}(\top_X, J) = \top_{\mathbf{pr}(X, J)}$$

$$(4) \mathbf{cpj}(\top_X, J) = \top_{\mathbf{pr}(X, J)}$$

21.3.15 Proof of 21.3.14

Let $Y := \mathbf{pr}(X, J)$. We first proof (1) and (3), and use the result in (2) and (4) by means of definition 21.2.2, (i.e. $\mathbf{cpj}(\textcircled{1}, \textcircled{2}) = \neg \mathbf{pr}(\neg \textcircled{1}, \textcircled{2})$).

$$(1) \mathbf{pr}(\perp_X, J) = \mathbf{pr}\left(\begin{bmatrix} X \\ \emptyset \end{bmatrix}, J\right) = \begin{bmatrix} \mathbf{pr}(X, J) \\ \mathbf{pr}(\emptyset, J) \end{bmatrix} = \begin{bmatrix} Y \\ \emptyset \end{bmatrix} = \perp_Y$$

$$(3) \mathbf{pr}(\top_X, J) = \mathbf{pr}\left(\begin{bmatrix} X \\ \otimes X \end{bmatrix}, J\right) = \begin{bmatrix} \mathbf{pr}(X, J) \\ \mathbf{pr}(\otimes X, J) \end{bmatrix} =$$

$$\begin{bmatrix} Y \\ \otimes Y \end{bmatrix} = \top_Y$$

$$(2) \mathbf{cpj}(\perp_X, J) = \neg \mathbf{pr}(\neg \perp_X, J) = \neg \mathbf{pr}(\top_X, J) = \neg \top_Y = \perp_Y$$

$$(3) \mathbf{cpj}(\top_X, J) = \neg \mathbf{pr}(\neg \top_X, J) = \neg \mathbf{pr}(\perp_X, J) = \neg \perp_Y = \top_Y$$

21.3.16 Lemma

Let $\mathcal{R} = \{\mathcal{R}_k \mid k \in K\}$ be a pairwise distinct relation class and $[J_k \mid k \in K]$ a schema with $J_k \subseteq \textcircled{\otimes}(\mathcal{R}_k)$ for each $k \in K$. So $[J_k \mid k \in K]$ is a class partition of the class $J := \bigcup_{k \in K} J_k$. Then

$$(1) \mathbf{pr}(\textcircled{\otimes} \mathcal{R}, J) = \textcircled{\otimes}_{k \in K} \mathbf{pr}(\mathcal{R}_k, J_k)$$

$$(2) \mathbf{cpj}(\textcircled{\otimes} \mathcal{R}, J) = \textcircled{\otimes}_{k \in K} \mathbf{cpj}(\mathcal{R}_k, J_k)$$

21.3.17 Proof of 21.3.16

For each $k \in K$ let $\mathcal{R}_k = [X_k, \Gamma_k]$ and $I_k := \textcircled{\otimes}(\mathcal{R}_k)$, so $J_k \subseteq I_k$. \mathcal{R} being pairwise distinct is saying that $I_l \cap I_m = \emptyset$ and thus $J_l \cap J_m = \emptyset$, for all $l, m \in K$ with $l \neq m$.

If we put $X := \bigvee_{k \in K} X_k$ and $Y := \mathbf{pr}(X, J)$, then

$$Y = \mathbf{pr}\left(\bigvee_{k \in K} X_k, \bigcup_{k \in K} J_k\right) = \bigvee_{k \in K} \mathbf{pr}(X_k, J_k)$$

according to 16.2.10(2).

For each $\xi \in \otimes X$ there is

$$\begin{aligned} \xi &= \bigvee_{k \in K} \mathbf{pr}(\xi, J_k) \\ &= \bigvee_{k \in K} (\mathbf{pr}(\xi, J_k) \dot{\vee} \mathbf{pr}(\xi, I_k \setminus J_k)) \\ &= \underbrace{\left(\bigvee_{k \in K} \mathbf{pr}(\xi, J_k)\right)}_{\in \otimes Y} \dot{\vee} \underbrace{\left(\bigvee_{k \in K} \mathbf{pr}(\xi, I_k \setminus J_k)\right)}_{\in \otimes (X \setminus Y)} \end{aligned}$$

So for $\textcircled{\otimes}_{k \in K} \Gamma_k \subseteq \otimes X$ we have

$$\begin{aligned} \mathbf{pr}\left(\textcircled{\otimes}_{k \in K} \Gamma_k, J\right) &= \{\mathbf{pr}(\xi, J) \mid \xi \in \textcircled{\otimes}_{k \in K} \Gamma_k\} \\ &= \left\{ \bigvee_{k \in K} \mathbf{pr}(\xi_k, J_k) \mid [\xi_k \mid k \in K] \in \textcircled{\otimes}_{k \in K} \Gamma_k \right\} \\ &= \textcircled{\otimes}_{k \in K} \mathbf{pr}(\Gamma_k, J_k) \end{aligned}$$

(1) We have

$$\begin{aligned} &\mathbf{pr}(\textcircled{\otimes} \mathcal{R}, J) \\ &= \mathbf{pr}\left(\begin{bmatrix} \bigvee_{k \in K} X_k \\ \textcircled{\otimes}_{k \in K} \Gamma_k \end{bmatrix}, J\right) \\ &= \begin{bmatrix} \mathbf{pr}(X, J) \\ \mathbf{pr}\left(\textcircled{\otimes}_{k \in K} \Gamma_k, J\right) \end{bmatrix} \\ &= \begin{bmatrix} Y \\ \{\mathbf{pr}(\xi, J) \mid \xi \in \textcircled{\otimes}_{k \in K} \Gamma_k\} \end{bmatrix} \\ &= \begin{bmatrix} Y \\ \left\{ \bigvee_{k \in K} \mathbf{pr}(\xi_k, J) \mid [\xi_k \mid k \in K] \in \textcircled{\otimes}_{k \in K} \Gamma_k \right\} \end{bmatrix} \\ &= \begin{bmatrix} Y \\ \left\{ \bigvee_{k \in K} v_k \mid [v_k \mid k \in K] \in \textcircled{\otimes}_{k \in K} \mathbf{pr}(\Gamma_k, J_k) \right\} \end{bmatrix} \\ &= \begin{bmatrix} \bigvee_{k \in K} \mathbf{pr}(X_k, J_k) \\ \textcircled{\otimes}_{k \in K} \mathbf{pr}(\Gamma_k, J_k) \end{bmatrix} \\ &= \textcircled{\otimes}_{k \in K} \mathbf{pr}(\mathcal{R}_k, J_k) \end{aligned}$$

(2) We have

$$\begin{aligned} &\mathbf{cpj}(\textcircled{\otimes} \mathcal{R}, J) \\ &= \mathbf{cpj}\left(\begin{bmatrix} X \\ \textcircled{\otimes}_{k \in K} \Gamma_k \end{bmatrix}, J\right) \\ &= \begin{bmatrix} Y \\ \left\{ v \in \otimes Y \mid \forall v' \in \otimes (X \setminus Y). v \dot{\vee} v' \in \textcircled{\otimes}_{k \in K} \Gamma_k \right\} \end{bmatrix} \\ &\quad \text{due to 21.2.7(2)} \\ &= \begin{bmatrix} Y \\ \left\{ \frac{v \in \otimes \left(\bigvee_{k \in K} \mathbf{pr}(X_k, J_k)\right)}{\forall v' \in \otimes \left(\bigvee_{k \in K} \mathbf{pr}(X_k, I_k \setminus J_k)\right)}. v \dot{\vee} v' \in \textcircled{\otimes}_{k \in K} \Gamma_k} \right\} \end{bmatrix} \end{aligned}$$

$$\begin{aligned}
&= \left[\begin{array}{c} \bigoplus_{k \in K} \mathbf{pr}(X_k, J_k) \\ \left\{ \begin{array}{l} \dot{\bigvee}_{k \in K} v_k \mid \left[\begin{array}{l} [v_k \mid k \in K] \in \bigotimes_{k \in K} (\otimes \mathbf{pr}(X_k, J_k)), \\ \forall [v'_k \mid k \in K] \in \bigotimes_{k \in K} (\otimes \mathbf{pr}(X_k, I_k \setminus J_k)) . \end{array} \right. \\ \left(\dot{\bigvee}_{k \in K} v_k \right) \dot{\bigvee} \left(\dot{\bigvee}_{k \in K} v'_k \right) \in \bigoplus_{k \in K} \Gamma_k \end{array} \right\} \end{array} \right] \\
&= \bigoplus_{k \in K} \left[\begin{array}{c} \mathbf{pr}(X_k, J_k) \\ \left\{ \begin{array}{l} v_k \in \otimes \mathbf{pr}(X_k, J_k) \\ \forall v'_k \in \otimes \mathbf{pr}(X_k, I_k \setminus J_k) . \\ v_k \dot{\bigvee} v'_k \in \Gamma_k \end{array} \right\} \end{array} \right] \\
&= \bigoplus_{k \in K} \mathbf{cpj}(\mathcal{R}_k, J_k) \quad \text{due to 21.2.7(2)}
\end{aligned}$$

21.4 (Attribute) Eliminations

21.4.1 Definition

Let $X = [X_i \mid i \in I]$ be a schema and $R \in \mathbf{Prel}(X)$.

For every $J \subseteq I$ we define

$$R \downarrow J := \mathbf{pr}(R, @ (R) \setminus J)$$

the (general) supremum elimination of J from R ,
or simply the upper R without J

$$R \uparrow J := \mathbf{cpj}(R, @ (R) \setminus J)$$

the (general) infimum elimination of J from R ,
or simply the lower R without J

For every $j \in I$ we define

$$R \downarrow j := R \downarrow \{j\}$$

the (singular) supremum elimination of j from R ,
or simply the upper R without j

$$R \uparrow j := R \uparrow \{j\}$$

the (singular) infimum elimination of j from R ,
or simply the lower R without j

21.4.2 Remark

While projection and coprojection specify the remaining attributes, supremum and infimum eliminations specify the attributes to be removed. From a constructive point of view, there is not much news here.

Eliminations are also similar to reductions. If $X = [X_i \mid i \in I]$ is a schema, $R = [X, \Gamma]$ is a relation and J a class, then

- (1) $R \downarrow J = R \downarrow \mathbf{pr}(X, I \cap J)$
- (2) $R \uparrow J = R \uparrow \mathbf{pr}(X, I \cap J)$

Reductions will be defined and investigated in 21.6. But they were already introduced in 21.1.1. By means of (1) and (2) the ideas and table methods from 21.1.1 can be applied to eliminations as well. So we don't bother with examples to illustrate the new concepts here and move on to list and proof some important properties. However similar the eliminations

are to (co)projections and reductions, there are some specific algebraic specialities for eliminations which are worth to be mentioned and require a proof.

21.5 Properties of eliminations

21.5.1 Lemma —basic properties general eliminations—

Let $X = [X_i \mid i \in I]$ be proper schema, $R \in \mathbf{Prel}(X)$, and $J \subseteq I$. Then

- (1) $\mathbf{x}(R \downarrow J) = \mathbf{pr}(\mathbf{x}(R), @ (R) \setminus J)$
- (2) $\mathbf{x}(R \uparrow J) = \mathbf{pr}(\mathbf{x}(R), @ (R) \setminus J)$
- (3) $@(R \downarrow J) = @(R) \setminus J$
- (4) $@(R \uparrow J) = @(R) \setminus J$
- (5) $\mathbf{gr}(R \downarrow J) = \mathbf{pr}(\mathbf{gr}(R), @ (R) \setminus J)$
- (6) $\mathbf{gr}(R \uparrow J) = \mathbf{pr}(\mathbf{gr}(R), @ (R) \setminus J)$

21.5.2 Proof of 21.5.1

Let $R = [Y, \Gamma]$ and $Y = [Y_l \mid l \in L]$, so that $\mathbf{x}(R) = Y$, $@(R) = \mathbf{dom}(Y) = L$, and $\mathbf{gr}(R) = \Gamma$.

- (1) Schema of supremum elimination:

$$\begin{aligned}
&\mathbf{x}(R \downarrow J) \\
&= \mathbf{x}(\mathbf{pr}(R, L \setminus J)) \quad \text{def. 21.4.1 of sup.elim.} \\
&= \mathbf{x} \left(\left[\begin{array}{l} \mathbf{pr}(Y, L \setminus J) \\ \mathbf{pr}(\Gamma, L \setminus J) \end{array} \right] \right) \quad \text{def. 21.2.2 of projection} \\
&= \mathbf{pr}(Y, L \setminus J)
\end{aligned}$$

- (2) Schema of infimum elimination: similar proof.
- (3) Attribute class of supremum elimination:

$$\begin{aligned}
&@(R \downarrow J) \\
&= \mathbf{dom}(\mathbf{x}(R \downarrow J)) \quad \text{def. 17.2.2 of attribute class} \\
&= \mathbf{dom}(\mathbf{pr}(Y, L \setminus J)) \quad \text{due to (1)} \\
&= L \setminus J
\end{aligned}$$

- (4) Attribute class of infimum elimination: similar proof.
- (5) Graph of supremum elimination:

$$\begin{aligned}
&\mathbf{gr}(R \downarrow J) \\
&= \mathbf{gr}(\mathbf{pr}(R, L \setminus J)) \quad \text{def. 21.4.1 of sup.elim.} \\
&= \mathbf{gr} \left(\left[\begin{array}{l} \mathbf{pr}(Y, L \setminus J) \\ \mathbf{pr}(\Gamma, L \setminus J) \end{array} \right] \right) \quad \text{def. 21.2.2 of projection} \\
&= \mathbf{pr}(\Gamma, L \setminus J)
\end{aligned}$$

- (6) Graph of infimum elimination:

$$\begin{aligned}
&\mathbf{gr}(R \uparrow J) \\
&= \mathbf{gr}(\mathbf{cpj}(R, L \setminus J)) \quad \text{def. 21.4.1 of inf.elim.} \\
&= \mathbf{gr} \left(\left[\begin{array}{l} \mathbf{pr}(Y, L \setminus J) \\ \otimes \mathbf{pr}(Y, L \setminus J) \setminus \mathbf{pr}(\otimes Y \setminus \Gamma, L \setminus J) \end{array} \right] \right) \quad \text{due to 21.2.5} \\
&= \otimes \mathbf{pr}(Y, L \setminus J) \setminus \mathbf{pr}(\otimes Y \setminus \Gamma, L \setminus J)
\end{aligned}$$

21.5.3 Lemma — properties of general eliminations

Let $X = [X_i | i \in I]$ be proper schema, $R, S \in \mathbf{Prel}(X)$, and $J, K \subseteq I$. Then

- (1) $R \downarrow J \downarrow K = R \downarrow (J \cup K)$ (accumulation of sup.elim.)
- (2) $R \uparrow J \uparrow K = R \uparrow (J \cup K)$ (accumulation of inf.elim.)
- (3) $R \downarrow J \downarrow K = R \downarrow K \downarrow J$ (commutativity of sup.elim.)
- (4) $R \uparrow J \uparrow K = R \uparrow K \uparrow J$ (commutativity of inf.elim.)
- (5) $R \downarrow J \downarrow J = R \downarrow J$ (idempotency of sup.elim.)
- (6) $R \uparrow J \uparrow J = R \uparrow J$ (idempotency of inf.elim.)
- (7) $R \downarrow \emptyset = R$ (empty sup.elim.)
- (8) $R \uparrow \emptyset = R$ (empty inf.elim.)
- (9) $R \downarrow J = R \downarrow (@ (R) \cap J)$ (essential att. of sup.elim.)
- (10) $R \uparrow J = R \uparrow (@ (R) \cap J)$ (essential att. of inf.elim.)
- (11) $R \downarrow J = R$ iff $@ (R) \cap J = \emptyset$ (neutral sup.elim.)
- (12) $R \uparrow J = R$ iff $@ (R) \cap J = \emptyset$ (neutral inf.elim.)
- (13) $J \subseteq K$ implies $R \downarrow J \supseteq R \downarrow K$ (antitony of sup.elim.)
- (14) $J \subseteq K$ implies $R \uparrow J \subseteq R \uparrow K$ (isotony of inf.elim.)
- (15) $R \downarrow J \supseteq R$ (supervalence of sup.elim.)
- (16) $R \uparrow J \subseteq R$ (subvalence of inf.elim.)
- (17) $R \uparrow J \downarrow K \subseteq R \downarrow K \uparrow J$ (non-commutativity of elim.)
- (18) $R \subseteq S$ implies $R \downarrow J \subseteq S \downarrow J$ (isotony of sup.elim.)
- (19) $R \subseteq S$ implies $R \uparrow J \subseteq S \uparrow J$ (isotony of inf.elim.)

21.5.4 Proof of 21.5.3

Let $R = [Y, \Gamma]$ and $Y = [Y_i | i \in I]$, so that $\mathbf{x}(R) = Y$ and $@(R) = \mathbf{dom}(Y) = L$.

(1) Accumulation of supremum elimination:

$$\begin{aligned}
 & R \downarrow J \downarrow K \\
 &= \mathbf{pr}(R, L \setminus J) \downarrow K && \text{def. 21.4.1} \\
 &= \mathbf{pr}(\mathbf{pr}(R, L \setminus J), (L \setminus J) \setminus K) && \text{def. 21.4.1} \\
 &= \mathbf{pr}(R, (L \setminus J) \setminus K) && \text{due to 21.3.3(6)(a)} \\
 &= \mathbf{pr}(R, L \setminus (J \cup K)) \\
 &= R \downarrow J \cup K && \text{def. 21.4.1, again}
 \end{aligned}$$

(2) Accumulation of infimum elimination: similar proof.

(3) Commutativity of supremum elimination:

$$\begin{aligned}
 & R \downarrow J \downarrow K \\
 &= R \downarrow (J \cup K) && \text{due to (1)} \\
 &= R \downarrow (K \cup J) \\
 &= R \downarrow K \downarrow J && \text{due to (1), again}
 \end{aligned}$$

(4) Commutativity of infimum elimination: similar proof.

(5) Idempotency of supremum elimination:

$$\begin{aligned}
 & R \downarrow J \downarrow J \\
 &= R \downarrow J \cup J && \text{due to (1)} \\
 &= R \downarrow J
 \end{aligned}$$

(6) Idempotency of infimum elimination: similar proof.

(7) Empty supremum elimination:

$$\begin{aligned}
 & R \downarrow \emptyset \\
 &= \mathbf{pr}(R, L \setminus \emptyset) && \text{def. 21.4.1} \\
 &= \mathbf{pr}(R, L) \\
 &= R && \text{due to 21.3.3(1)(a)}
 \end{aligned}$$

(8) Empty infimum elimination: similar proof.

(9) Essential attributes of supremum elimination:

$$\begin{aligned}
 & R \downarrow J \\
 &= \mathbf{pr}(R, L \setminus J) && \text{def. 21.4.1}
 \end{aligned}$$

$$= \mathbf{pr}(R, L \setminus (L \cap J))$$

$$= R \downarrow L \cap J$$

def. 21.4.1, again

(10) Essential attributes of infimum elimination: similar proof.

(11) Neutral supremum eliminations: If $L \cap J \neq \emptyset$, then $L \setminus J \neq L$, so $\mathbf{x}(R) = L \neq L \setminus J = \mathbf{x}(R \downarrow J)$ and $R \neq R \downarrow J$. On the other hand, if $L \cap J = \emptyset$, then $R \downarrow J = R \downarrow L \cap J = R \downarrow \emptyset = R$, due to (9) and (7).

(12) Neutral infimum eliminations: similar proof.

(13) Antitony of supremum elimination: If $J \subseteq K$, then $L \setminus J \supseteq L \setminus K$, so that

$$\begin{aligned}
 & R \downarrow J \\
 &= \mathbf{pr}(R, L \setminus J) \\
 &\supseteq \mathbf{pr}(R, L \setminus K) && \text{due to 21.3.3(8)(a)} \\
 &= R \downarrow K
 \end{aligned}$$

(14) Isotony of infimum elimination: similar proof.

(15) Supervalence of supremum elimination: $R \downarrow J = \mathbf{pr}(R, L \setminus J) \supseteq R$, due to 21.3.3(9)(a).

(16) Subvalence of infimum elimination: similar proof.

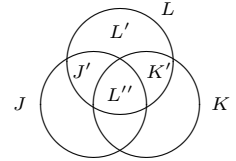
(17) Recall, that $R = [Y, \Gamma]$ with $Y = [Y_i | i \in I]$. We partition L into the following four distinct classes:

$$L' := L \setminus (J \cup K)$$

$$L'' := L \cap J \cap K$$

$$J' := (L \cap J) \setminus K$$

$$K' := (K \cap L) \setminus J$$



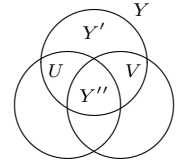
Accordingly, we partition Y into four distinct schemas:

$$Y' := \mathbf{pr}(Y, L')$$

$$Y'' := \mathbf{pr}(Y, L'')$$

$$U := \mathbf{pr}(Y, J')$$

$$V := \mathbf{pr}(Y, K')$$



We will use the following general logical rules (see. 5.5.11). If v and w are variables, C and D class symbols, and φ a formula, then

(a) $\exists v \in C . \forall w \in D . \varphi \Rightarrow \forall w \in D . \exists v \in C . \varphi$

(b) If C denotes a non-empty class, then $\forall v \in C . \varphi \Rightarrow \exists v \in C . \varphi$

We now give the full proof:

$$\begin{aligned}
 & R \uparrow J \downarrow K \\
 &= \mathbf{pr}(\mathbf{cpj}(R, L' \cup K'), L') && \text{def. 21.4.1} \\
 &= \mathbf{pr}\left(\left[\begin{array}{c} Y' \dot{\vee} V \\ \xi \rightsquigarrow \forall \zeta \in \otimes(U \dot{\vee} Y'') . \xi \dot{\vee} \zeta \in \Gamma \end{array}\right], L'\right) && \text{due to 21.2.7(2)} \\
 &= \left[\begin{array}{c} Y' \\ y' \rightsquigarrow \exists v \in \otimes V . \forall \zeta \in \otimes(U \dot{\vee} Y'') . y' \dot{\vee} v \dot{\vee} \zeta \in \Gamma \end{array}\right] && \text{due to 21.2.7(1)} \\
 &= \left[\begin{array}{c} Y' \\ y' \rightsquigarrow \exists v \in \otimes V . \forall u \in \otimes U . \forall y'' \in \otimes Y'' . \\ y' \dot{\vee} v \dot{\vee} u \dot{\vee} y'' \in \Gamma \end{array}\right] && \text{due to 16.6.5(4)}
 \end{aligned}$$

$$\begin{aligned}
&\subseteq \left[\begin{array}{c} Y' \\ y' \rightsquigarrow \forall u \in \otimes U . \exists v \in \otimes V . \forall y'' \in \otimes Y'' . \\ y' \dot{\vee} u \dot{\vee} v \dot{\vee} y'' \in \Gamma \end{array} \right] \\
&\hspace{15em} \text{due to (a)} \\
&\subseteq \left[\begin{array}{c} Y' \\ y' \rightsquigarrow \forall u \in \otimes U . \forall v \in \otimes V . \exists y'' \in \otimes Y'' . \\ y' \dot{\vee} u \dot{\vee} v \dot{\vee} y'' \in \Gamma \end{array} \right] \\
&\hspace{15em} \text{due to (b), with } \otimes Y'' \neq \emptyset \\
&\subseteq \left[\begin{array}{c} Y' \\ y' \rightsquigarrow \forall u \in \otimes U . \exists \zeta \in \otimes (V \dot{\vee} Y'') . \\ y' \dot{\vee} u \dot{\vee} \zeta \in \Gamma \end{array} \right] \\
&\hspace{15em} \text{due to 16.6.5(4), again} \\
&= \mathbf{cpj} \left(\left[\begin{array}{c} Y' \dot{\vee} V \\ \xi \rightsquigarrow \exists \zeta \in \otimes (V \dot{\vee} Y'') . \xi \dot{\vee} \zeta \in \Gamma \end{array} \right], L' \right) \\
&\hspace{15em} \text{due to 21.2.7(2), again} \\
&= \mathbf{cpj} (\mathbf{pr} (R, L' \cup J'), L') \hspace{5em} \text{due to 21.2.7(1), again} \\
&= R \downarrow K \uparrow J \hspace{15em} \text{def. 21.4.1, again}
\end{aligned}$$

(18) Recall, that $Y = \mathbf{x}(R)$ and $L = \mathbb{@}(R)$. So let $Y' = \mathbf{x}(S)$ and $L' = \mathbb{@}(S)$.

$$\begin{aligned}
&R \subseteq S \\
&\Leftrightarrow R \parallel (Y \vee Y') \subseteq S \parallel (Y \vee Y') \\
&\hspace{15em} \text{def. 20.2.2} \\
&\Leftrightarrow R \odot \top_{(Y' \setminus Y)} \subseteq S \odot \top_{(Y' \setminus Y)} \\
&\hspace{15em} \text{due to 20.1.3(3)} \\
&\Rightarrow \left(\begin{array}{c} \mathbf{pr} (R \odot \top_{(Y' \setminus Y)}, (L \cup L') \setminus J) \\ \subseteq \mathbf{pr} (S \odot \top_{(Y' \setminus Y)}, (L \cup L') \setminus J) \end{array} \right) \\
&\hspace{15em} \text{due to 21.3.3(11)} \\
&\Leftrightarrow \left(\begin{array}{c} \mathbf{pr} (R, L \setminus J) \odot \mathbf{pr} (\top_{(Y' \setminus Y)}, L' \setminus (L \cup J)) \\ \subseteq \mathbf{pr} (S, L' \setminus J) \odot \mathbf{pr} (\top_{(Y' \setminus Y)}, L \setminus (L' \cup J)) \end{array} \right) \\
&\hspace{15em} \text{due to 21.3.16(1)} \\
&\Leftrightarrow \left(\begin{array}{c} \mathbf{pr} (R, L \setminus J) \odot \top_{\mathbf{pr}(Y', L' \setminus (L \cup J))} \\ \subseteq \mathbf{pr} (S, L' \setminus J) \odot \top_{\mathbf{pr}(Y, L \setminus (L' \cup J))} \end{array} \right) \\
&\hspace{15em} \text{due to 21.3.14(3)} \\
&\Leftrightarrow \left(\begin{array}{c} (R \downarrow J) \odot \top_{\mathbf{pr}(Y', L' \setminus (L \cup J))} \\ \subseteq (S \downarrow J) \odot \top_{\mathbf{pr}(Y, L \setminus (L' \cup J))} \end{array} \right) \\
&\hspace{15em} \text{def. 21.4.1(1)} \\
&\Leftrightarrow \left(\begin{array}{c} (R \downarrow J) \parallel \mathbf{pr} (Y \vee Y', (L \cup L') \setminus J) \\ \subseteq (S \downarrow J) \parallel \mathbf{pr} (Y \vee Y', (L \cup L') \setminus J) \end{array} \right) \\
&\hspace{15em} \text{due to 20.1.3} \\
&\Leftrightarrow R \downarrow J \subseteq S \downarrow J \hspace{15em} \text{def. 20.2.2}
\end{aligned}$$

(19) Proof similar to (18).

21.5.5 Lemma

Let $X = [X_i | i \in I]$ be a proper schema.

(1) If $J \subseteq I$ and $\mathcal{R} \subseteq \mathbf{Prel}(X)$ is pairwise distinct, then

$$\begin{aligned}
\text{(a)} \quad &\left(\bigcirc_{k \in K} \mathcal{R}_k \right) \downarrow J = \bigcirc_{k \in K} (\mathcal{R}_k \downarrow J) \\
\text{(b)} \quad &\left(\bigcirc_{k \in K} \mathcal{R}_k \right) \uparrow J = \bigcirc_{k \in K} (\mathcal{R}_k \uparrow J)
\end{aligned}$$

(2) If $J \subseteq I$, $Y \leq X$, and if we put

$$Y' := \mathbf{pr} (Y, \mathbf{dom}(Y) \setminus J)$$

then

$$\begin{aligned}
\text{(a)} \quad &\top_Y \downarrow J = \top_{Y'} \\
\text{(b)} \quad &\top_Y \uparrow J = \top_{Y'} \\
\text{(c)} \quad &\perp_Y \downarrow J = \perp_{Y'} \\
\text{(d)} \quad &\perp_Y \uparrow J = \perp_{Y'}
\end{aligned}$$

(3) If $R \in \mathbf{Prel}(X)$ and $J \subseteq I$, then

$$\begin{aligned}
\text{(a)} \quad &R \downarrow J = \neg((\neg R) \uparrow J) \\
\text{(b)} \quad &R \uparrow J = \neg((\neg R) \downarrow J)
\end{aligned}$$

(4) If $J \subseteq I$, $Y \leq X$, $R \in \mathbf{Prel}(X)$, and if we put

$$Y' := \mathbf{pr} (Y, \mathbf{dom}(Y) \setminus J)$$

then

$$\begin{aligned}
\text{(a)} \quad &(R \parallel Y) \downarrow J = (R \downarrow J) \parallel Y' \\
\text{(b)} \quad &(R \parallel Y) \uparrow J = (R \uparrow J) \parallel Y'
\end{aligned}$$

21.5.6 Proof of 21.5.5

(1) Let $L_k = \mathbb{@}(\mathcal{R}_k)$, for each $k \in K$. \mathcal{R} being pairwise distinct means that for all $k, k' \in K$ with $k \neq k'$, $L_k \cap L_{k'} = \emptyset$, and that implies $(L_k \setminus J) \cap (L_{k'} \cap J) = \emptyset$.

$$\begin{aligned}
\text{(a)} \quad &\left(\bigcirc_{k \in K} \mathcal{R}_k \right) \downarrow J \\
&= \mathbf{pr} \left(\bigcirc_{k \in K} \mathcal{R}_k, \mathbb{@} \left(\bigcirc_{k \in K} \mathcal{R}_k \right) \setminus J \right) \hspace{5em} \text{def. 21.4.1} \\
&= \mathbf{pr} \left(\bigcirc_{k \in K} \mathcal{R}_k, \left(\bigcup_{k \in K} L_k \right) \setminus J \right) \\
&= \mathbf{pr} \left(\bigcirc_{k \in K} \mathcal{R}_k, \bigcup_{k \in K} (L_k \setminus J) \right) \\
&= \bigcirc_{k \in K} \mathbf{pr} (\mathcal{R}_k, L_k \setminus J) \hspace{5em} \text{due to 21.3.16(1)} \\
&= \bigcirc_{k \in K} (\mathcal{R}_k \downarrow J)
\end{aligned}$$

(b) Has a similar proof.

(2) These four statements are a translation of the four (co)project statements in 21.3.14 in term of eliminations. By using 21.3.14(1), we obtain a proof of (a)

$$\begin{aligned}
\top_Y \downarrow J &= \mathbf{pr} (\top_Y, \mathbf{dom}(Y) \setminus J) \\
&= \top_{\mathbf{pr}(Y, \mathbf{dom}(Y) \setminus J)} \\
&= \top_{Y'}
\end{aligned}$$

The proofs for (b), (c), and (d) are likewise.

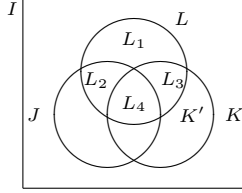
(3) Definition 21.2.2 of the coprojection and lemma 21.3.12 are the main tools in the following proofs for (a) and (b)

$$\begin{aligned}
R \downarrow J &= \mathbf{pr} (R, \mathbb{@}(R) \setminus J) \\
&= \mathbf{pr} (\neg \neg R, \mathbb{@}(R) \setminus J) \\
&= \neg \mathbf{cpj} (\neg R, \mathbb{@}(R) \setminus J) \\
&= \neg((\neg R) \uparrow J)
\end{aligned}$$

$$\begin{aligned}
R \uparrow J &= \mathbf{cpj}(R, @ (R) \setminus J) \\
&= \neg \mathbf{pr}(\neg R, @ (R) \setminus J) \\
&= \neg((\neg R) \downarrow J)
\end{aligned}$$

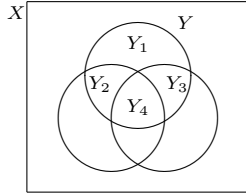
(4) Let us point out the following subclasses of I

$$\begin{aligned}
L &:= \mathbf{dom}(Y) \\
K &:= @ (R) \\
K' &:= K \setminus L \\
L_1 &:= L \setminus (K \cup L) \\
L_2 &:= (L \cap J) \setminus K \\
L_3 &:= (L \cap K) \setminus J \\
L_4 &:= L \cap J \cap K
\end{aligned}$$



so that L is the disjunct union of the L_1, L_2, L_3, L_4 . We also split up the schema X accordingly

$$\begin{aligned}
Y_i &:= \mathbf{pr}(Y, L_i) \\
&\text{for } i = 1, 2, 3, 4
\end{aligned}$$



so that $Y = Y_1 \dot{\vee} Y_2 \dot{\vee} Y_3 \dot{\vee} Y_4$ and $Y' = Y_1 \dot{\vee} Y_3$.

(a) We can now see that

$$\begin{aligned}
&(R \parallel Y) \uparrow J \\
&= (R \odot \top_{(Y_1 \dot{\vee} Y_2)}) \uparrow J && \text{due to 20.1.3(3)} \\
&= \mathbf{pr}(R \odot \top_{(Y_1 \dot{\vee} Y_2)}, (K \cup L_1 \cup L_2) \setminus J) \\
&&& \text{def. 21.4.1} \\
&= \mathbf{pr}(R \odot \top_{(Y_1 \dot{\vee} Y_2)}, K' \cup L_3 \cup L_1) \\
&= \mathbf{pr}(R, K' \cup L_3) \odot \mathbf{pr}(\top_{(Y_1 \dot{\vee} Y_2)}, L_1) \\
&&& \text{due to 21.3.16(1)} \\
&= \mathbf{pr}(R, K \setminus J) \odot \top_{Y_1} \\
&= (R \downarrow J) \odot \top_{Y_1} && \text{def. 21.4.1, again} \\
&= (R \downarrow J) \parallel Y_1 && \text{due to 20.1.3} \\
&= ((R \downarrow J) \parallel Y_3) \parallel Y_1 && \text{due to 20.1.11(3)} \\
&= R \downarrow J \parallel (Y_3 \vee Y_1) && \text{due to 20.1.11(1)} \\
&= R \downarrow J \parallel Y'
\end{aligned}$$

(b) Quite similar we obtain

$$\begin{aligned}
&(R \parallel Y) \uparrow J \\
&= \mathbf{cpj}(R \parallel Y, (L \cup K) \setminus J) && \text{def. 21.4.1} \\
&= \mathbf{cpj}(R \parallel Y, K' \cup L_3 \cup L_1) \\
&= \neg \mathbf{pr}(\neg(R \parallel Y), K' \cup L_3 \cup L_1) && \text{def. 21.2.2} \\
&= \neg \mathbf{pr}((\neg R) \parallel Y, K' \cup L_3 \cup L_1) \\
&&& \text{due to 20.4.15(3)} \\
&= \neg \mathbf{pr}((\neg R) \odot \top_{(Y_1 \dot{\vee} Y_2)}, K' \cup L_3 \cup L_1) \\
&&& \text{due to 20.1.3} \\
&= \neg(\mathbf{pr}(\neg R, K' \cup L_3) \odot \mathbf{pr}(\top_{(Y_1 \dot{\vee} Y_2)}, L_1)) \\
&&& \text{due to 21.3.16(1)} \\
&= \neg(\mathbf{pr}(\neg R, K \setminus J) \odot \top_{Y_1}) \\
&= \neg(((\neg R) \downarrow J) \odot \top_{Y_1}) && \text{def. 21.4.1} \\
&= \neg(((\neg R) \downarrow J) \parallel Y') && \text{as in the proof of (a)} \\
&= \neg((\neg R) \downarrow J) \parallel Y' && \text{due to 20.4.15(3)} \\
&= R \uparrow J \parallel Y' && \text{due to (3)(b)}
\end{aligned}$$

21.5.7 Lemma — general eliminations over junctions

Let $X = [X_i | i \in I]$ be a proper schema. For each $\mathcal{R} = \{\mathcal{R}_k | k \in K\} \subseteq \mathbf{Prel}(X)$, all $R, S \in \mathbf{Prel}(X)$, and every $J \subseteq I$ holds:

- (1) $\perp \downarrow J = \perp$
- (2) $\perp \uparrow J = \perp$
- (3) $\top \downarrow J = \top$
- (4) $\top \uparrow J = \top$
- (5) $(\neg R) \downarrow J = \neg(R \uparrow J)$
- (6) $(\neg R) \uparrow J = \neg(R \downarrow J)$
- (7) $(R \cap S) \downarrow J \subseteq (R \downarrow J) \cap (S \downarrow J)$
- (8) $(R \cap S) \uparrow J = (R \uparrow J) \cap (S \uparrow J)$
- (9) $(R \cup S) \downarrow J = (R \downarrow J) \cup (S \downarrow J)$
- (10) $(R \cup S) \uparrow J \supseteq (R \uparrow J) \cup (S \uparrow J)$
- (11) $(\prod \mathcal{R}) \downarrow J \subseteq \prod \{\mathcal{R}_k \downarrow J | k \in K\}$
- (12) $(\prod \mathcal{R}) \uparrow J = \prod \{\mathcal{R}_k \uparrow J | k \in K\}$
- (13) $(\coprod \mathcal{R}) \downarrow J = \coprod \{\mathcal{R}_k \downarrow J | k \in K\}$
- (14) $(\coprod \mathcal{R}) \uparrow J \supseteq \coprod \{\mathcal{R}_k \uparrow J | k \in K\}$

21.5.8 Proof of 21.5.7

(1) We have

$$\begin{aligned}
&\perp \downarrow J \\
&= \mathbf{pr}(\perp, @(\perp) \setminus J) && \text{def. 21.4.1} \\
&= \mathbf{pr}(\perp, \emptyset) \\
&= \perp_{\mathbf{pr}(\cdot, \emptyset)} && \text{due to 21.3.14(1)} \\
&= \perp_{\langle \rangle} \\
&= \perp && \text{def. 19.3.1}
\end{aligned}$$

(2),(3),(4) Proof similar to (1).

(5) We obtain

$$\begin{aligned}
&(\neg R) \downarrow J \\
&= \mathbf{pr}(\neg R, @ (R) \setminus J) && \text{def. 21.4.1} \\
&= \neg \mathbf{cpj}(R, @ (R) \setminus J) && \text{due to 21.3.12(1)} \\
&= \neg(R \uparrow J) && \text{def. 21.4.1, again}
\end{aligned}$$

(6) Proof similar to (5).

(7) This statement is just a special case of (11) proved below, because $R \cap S = \prod \{R, S\}$, due to 20.4.21(9).

(8) Special case of (12).

(9) Special case of (13).

(10) Special case of (14).

For the last four statements we need some preparations. The relation class $\mathcal{R} \subseteq \mathbf{Prel}(X)$ was given by $\mathcal{R} = \{\mathcal{R}_k | k \in K\}$. Now, for every $k \in K$ let

$$A_k = @(\mathcal{R}_k) \quad X_k = \mathbf{x}(\mathcal{R}_k) \quad \Gamma_k = \mathbf{gr}(\mathcal{R}_k)$$

so that $\mathcal{R}_k = [X_k, \Gamma_k]$. Furthermore, let

$$A := \bigcup \{A_k | k \in K\} \quad Z := \bigvee \{X_k | k \in K\}$$

So $@(\prod \mathcal{R}) = A$ and $\mathbf{x}(\prod \mathcal{R}) = Z$. Finally, let us put

$$Z' := \mathbf{pr}(Z, A \setminus J)$$

Note, that

$$\begin{aligned}
\mathbf{x}(\prod\{\mathcal{R}_k \downarrow J \mid k \in K\}) &= \\
\mathbf{x}(\prod\{\mathcal{R}_k \uparrow J \mid k \in K\}) &= \\
\mathbf{x}(\bigsqcup\{\mathcal{R}_k \downarrow J \mid k \in K\}) &= \\
\mathbf{x}(\bigsqcup\{\mathcal{R}_k \uparrow J \mid k \in K\}) &= Z'
\end{aligned}$$

And due to 21.5.5(4), we have for every $k \in K$,

- (a) $(\mathcal{R}_k \parallel Z) \downarrow J = (\mathcal{R}_k \downarrow J) \parallel Z'$
(b) $(\mathcal{R}_k \parallel Z) \uparrow J = (\mathcal{R}_k \uparrow J) \parallel Z'$

We now obtain the proofs for (11) to (14) as follows

(11) We have

$$\begin{aligned}
&(\prod \mathcal{R}) \downarrow J \\
&= \mathbf{pr}(\prod \mathcal{R}, A \setminus J) && \text{due to 21.4.1} \\
&= \mathbf{pr}(\bigcap\{\mathcal{R}_k \parallel Z \mid k \in K\}, A \setminus J) && \text{def. 20.2.4 of } \prod \\
&\subseteq \bigcap\{\mathbf{pr}(\mathcal{R}_k \parallel Z, A \setminus J) \mid k \in K\} && \text{due to 21.3.10(1)} \\
&= \bigcap\{(\mathcal{R}_k \parallel Z) \downarrow J \mid k \in K\} && \text{def. 21.4.1} \\
&= \bigcap\{(\mathcal{R}_k \downarrow J) \parallel Z' \mid k \in K\} && \text{due to (a)} \\
&= \prod\{\mathcal{R}_k \downarrow J \mid k \in K\} && \text{def. 20.2.4 of } \prod, \text{ again}
\end{aligned}$$

(12) We have

$$\begin{aligned}
&(\prod \mathcal{R}) \uparrow J \\
&= \mathbf{cpj}(\prod \mathcal{R}, A \setminus J) && \text{def. 21.4.1} \\
&= \mathbf{cpj}(\bigcap\{\mathcal{R}_k \parallel Z \mid k \in K\}, A \setminus J) && \text{def. 20.2.4} \\
&= \bigcap\{\mathbf{cpj}((\mathcal{R}_k \parallel Z), A \setminus J) \mid k \in K\} && \text{due to 21.3.10(3)} \\
&= \bigcap\{(\mathcal{R}_k \parallel Z) \uparrow J \mid k \in K\} && \text{def. 21.4.1} \\
&= \bigcap\{(\mathcal{R}_k \uparrow J) \parallel Z' \mid k \in K\} && \text{due to (b)} \\
&= \prod\{\mathcal{R}_k \uparrow J \mid k \in K\} && \text{def. 20.2.4}
\end{aligned}$$

(13) Using the justifications of the proof for (11), we obtain

$$\begin{aligned}
&(\bigsqcup \mathcal{R}) \downarrow J \\
&= \mathbf{pr}(\bigcup\{\mathcal{R}_k \parallel Z \mid k \in K\}, A \setminus J) \\
&= \bigcup\{\mathbf{pr}(\mathcal{R}_k \parallel Z, A \setminus J) \mid k \in K\} && \text{due to 21.3.10(2)} \\
&= \bigsqcup\{\mathcal{R}_k \downarrow J \mid k \in K\}
\end{aligned}$$

(14) Finally and similar to (12) we obtain

$$\begin{aligned}
&(\bigcup \mathcal{R}) \uparrow J \\
&= \mathbf{cpj}(\bigcup\{\mathcal{R}_k \parallel Z \mid k \in K\}, A \setminus J) \\
&\supseteq \bigcup\{\mathbf{cpj}(\mathcal{R}_k \parallel Z, A \setminus J) \mid k \in K\} && \text{due to 21.3.10(4)} \\
&= \bigsqcup\{\mathcal{R}_k \uparrow J \mid k \in K\}
\end{aligned}$$

21.5.9 Lemma

Let $X = [X_i \mid i \in I]$ be a proper schema, $R \in \mathbf{Prel}(X)$ and $J \subseteq I$. The following statements are equivalent:

- (1) $R \downarrow J \equiv R$
- (2) $R \uparrow J \equiv R$
- (3) $R \downarrow j \equiv R$ for every $j \in J$
- (4) $R \uparrow j \equiv R$ for every $j \in J$

21.5.10 Proof of 21.5.9

Suppose, $R = [Y, \Gamma]$ with $Y = [Y_i \mid i \in L]$.

(1) \Rightarrow (3) For every $j \in J$ holds:

- ♣ $R \sqsubseteq R \downarrow j$ due to 21.5.3(15)
- ♣ $R \downarrow j \sqsubseteq R \downarrow J$ due to ??(13)

so that

$$R \downarrow J \equiv R \text{ implies } R \downarrow j \equiv R$$

$\neg(1) \Rightarrow \neg(3)$ We derive

$$\begin{aligned}
&R \downarrow J \not\equiv R \\
&\Leftrightarrow R \sqsubset R \downarrow J && \text{due to 21.5.3(15)} \\
&\Leftrightarrow R \subset (R \downarrow J) \parallel Y && \text{def. 20.2.2} \\
&\Leftrightarrow R \subset (R \downarrow J) \odot \top_{\mathbf{pr}(Y, J \cap L)} && \text{due to 20.1.3} \\
&\Leftrightarrow \begin{bmatrix} Y \\ \Gamma \end{bmatrix} \subset \begin{bmatrix} \mathbf{pr}(Y, L \setminus J) \\ \mathbf{pr}(\Gamma, L \setminus J) \end{bmatrix} \odot \begin{bmatrix} \mathbf{pr}(Y, J \cap L) \\ \otimes \mathbf{pr}(Y, J \cap L) \end{bmatrix} \\
&\Leftrightarrow \Gamma \subset \mathbf{pr}(\Gamma, L \setminus J) \odot \otimes \mathbf{pr}(Y, J \cap L) \\
&\Leftrightarrow \exists \xi \in \Gamma. \exists v \in \otimes \mathbf{pr}(Y, J \cap L). \mathbf{pr}(\xi, L \setminus J) \dot{\vee} v \notin \Gamma
\end{aligned}$$

In that case and for such ξ and v we put $J' := \{j \in (J \cap L) \mid \xi_j \neq v_j\}$. J' is not empty, because then $\mathbf{pr}(\xi, L \setminus J) \dot{\vee} v = \xi \in \Gamma$. So there must be at least one $j \in J'$. And for this j we would have $\xi \in R$, but $\xi \notin R \downarrow j \parallel Y$, and thus $R \not\equiv R \downarrow j$.

(1) \Leftrightarrow (2) We have

$$\begin{aligned}
&R \downarrow J \equiv R \\
&\Leftrightarrow R \downarrow J \sqsubseteq R && \text{because } R \downarrow J \sqsupseteq R, \text{ due to 21.5.3(15)} \\
&\Leftrightarrow \mathbf{pr}(R \downarrow J, L \setminus J) \subseteq \mathbf{cpj}(R, L \setminus J) && \text{due to 21.3.6} \\
&\Leftrightarrow R \downarrow J \subseteq R \uparrow J && \text{due to 21.3.3(1)(a) and def. 21.4.1(1)} \\
&\Leftrightarrow \mathbf{pr}(R, L \setminus J) \subseteq \mathbf{cpj}(R \uparrow J, L \setminus J) && \text{def. 21.4.1(1) and due to 21.3.3(1)(b)} \\
&\Leftrightarrow R \sqsubseteq R \uparrow J && \text{due to 21.3.6 again} \\
&\Leftrightarrow R \uparrow J \equiv R && \text{because } R \uparrow J \sqsubseteq R \text{ is due to 21.5.3(16)}
\end{aligned}$$

(3) \Leftrightarrow (4) We just showed that $R \uparrow J \equiv R$ iff $R \downarrow J \equiv R$. As special case this implies $R \uparrow j \equiv R$ iff $R \downarrow j \equiv R$.

21.6 Reductions

21.6.1 Definition

Let X be a schema, $R \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$. We define

$$R \Downarrow Y := \text{the } S \in \mathbf{Rel}(Y) \text{ with } S \equiv R$$

the equivalent reduction of R onto Y

$$R \Downarrow Y := \bigcap\{S \in \mathbf{Rel}(Y) \mid R \sqsubseteq S\}$$

the supremum reduction of R onto Y

$$R \Uparrow Y := \bigcup\{S \in \mathbf{Rel}(Y) \mid S \sqsubseteq R\}$$

the infimum reduction of R onto Y

21.6.2 Lemma

If X is a relation, $R \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$, then

- (1) If $R \uparrow Y$ exists, then $\mathbf{x}(R \uparrow Y) = Y$
- (2) $\mathbf{x}(R \downarrow Y) = Y$
- (3) $\mathbf{x}(R \uparrow Y) = Y$

21.6.3 Proof of 21.6.2

(1) is obvious. (2) and (3) are due to definition 19.5.3 of \cap and \cup , which are operations on $\mathbf{Rel}(Y)$. Besides, there is no ambiguity here, which might occur for the expressions $\cap \emptyset$ and $\cup \emptyset$, because this empty class situation never occurs. There is always at least one $S \in \mathbf{Rel}(Y)$ which is subvalent (or supervalent, respectively) to R .

21.6.4 Lemma

Let $X = [X_i | i \in I]$ and $Y = [Y_j | j \in J]$ be two schemas and $R = [X, \Gamma]$ a relation.

- (1) If $Y \leq X$ then
 - (a) $R \downarrow Y = \mathbf{pr}(R, J)$
 - (b) $R \uparrow Y = \mathbf{cpj}(R, J)$
 - (c) $R \downarrow Y = R \downarrow (I \setminus J)$
 - (d) $R \uparrow Y = R \uparrow (I \setminus J)$
- (2) If $X \leq Y$ then
 - (a) $R \downarrow Y = R \parallel Y$
 - (b) $R \uparrow Y = R \parallel Y$
- (3) If $Y \sim X$, then
 - (a) $R \downarrow Y = \mathbf{pr}(R, J \cap I) \odot \top_{Y \setminus X}$
 - (b) $R \uparrow Y = \mathbf{cpj}(R, J \cap I) \odot \top_{Y \setminus X}$
 - (c) $R \downarrow Y = R \downarrow (I \setminus J) \odot \top_{Y \setminus X}$
 - (d) $R \uparrow Y = R \uparrow (I \setminus J) \odot \top_{Y \setminus X}$

21.6.5 Proof of 21.6.4

Recall, that for every class C and $A \in \mathbf{P}(C)$.

- (i) $\cap \{B \in \mathbf{P}(C) \mid A \subseteq B\} = A$
- (ii) $\cup \{B \in \mathbf{P}(C) \mid B \subseteq A\} = A$

Similar statements hold on $\mathbf{Rel}(Y)$ as well, i.e. for every $T = [Y, \Gamma] \in \mathbf{Rel}(Y)$ holds:

(i) We have

$$\begin{aligned} & \cap \{S \in \mathbf{Rel}(Y) \mid T \subseteq S\} \\ &= \cap \left\{ \begin{bmatrix} Y \\ \Sigma \end{bmatrix} \mid \Sigma \subseteq \otimes Y, \Gamma \subseteq \Sigma \right\} \\ &= \begin{bmatrix} Y \\ \cap \{ \Sigma \in \mathbf{P}(\otimes Y) \mid \Gamma \subseteq \Sigma \} \end{bmatrix} \\ &= [Y, \Gamma] \\ &= T \end{aligned}$$

due to (i) above

and similarly, we also have

- (ii) $\cup \{S \in \mathbf{Rel}(Y) \mid S \subseteq T\} = T$

Now we proof in detail:

- (1) If $Y \leq X$, then $J \subseteq I$.

(a) We obtain

$$R \downarrow Y$$

$$= \cap \{S \in \mathbf{Rel}(Y) \mid R \subseteq S\}$$

def. 21.6.1 of \downarrow

$$= \cap \{S \in \mathbf{Rel}(Y) \mid \mathbf{pr}(R, J) \subseteq \mathbf{cpj}(S, J)\}$$

due to 21.3.6

$$= \cap \{S \in \mathbf{Rel}(Y) \mid \mathbf{pr}(R, J) \subseteq S\}$$

due to 21.3.3(1)(b)

$$= \mathbf{pr}(R, J)$$

due to the initial remarks

(b) Similar to (a) we obtain

$$R \uparrow Y = \cup \{S \in \mathbf{Rel}(Y) \mid S \subseteq R\}$$

$$= \cup \{S \in \mathbf{Rel}(Y) \mid \mathbf{pr}(S, J) \subseteq \mathbf{cpj}(R, J)\}$$

$$= \cup \{S \in \mathbf{Rel}(Y) \mid S \subseteq \mathbf{cpj}(R, J)\}$$

$$= \mathbf{cpj}(R, J)$$

(c) We have

$$R \downarrow Y$$

$$= \mathbf{pr}(R, J)$$

due to (a)

$$= R \downarrow (I \setminus J)$$

def. 21.4.1(1)

(d) Similar to (c), $R \uparrow Y = \mathbf{cpj}(R, J) = R \uparrow (I \setminus J)$.

(2) Now $X \leq Y$.

(a) We have

$$R \downarrow Y$$

$$= \cap \{S \in \mathbf{Rel}(Y) \mid R \subseteq S\}$$

def. 21.6.1 of \downarrow

$$= \cap \{S \in \mathbf{Rel}(Y) \mid R \parallel (X \vee Y) \subseteq S \parallel (X \vee Y)\}$$

def. 20.2.2 of \subseteq

$$= \cap \{S \in \mathbf{Rel}(Y) \mid R \parallel Y \subseteq S \parallel Y\}$$

because $X \vee Y = Y$

$$= \cap \{S \in \mathbf{Rel}(Y) \mid R \parallel Y \subseteq S\}$$

because $S \parallel Y = S$

$$= R \parallel Y$$

due to (i)

(b) Proof similar to (a).

(3) Now $Y \sim X$.

(a) We have

$$R \downarrow Y$$

$$= \cap \{S \in \mathbf{Rel}(Y) \mid R \subseteq S\}$$

def. 21.6.1 of \downarrow

$$= \cap \{S \in \mathbf{Rel}(Y) \mid \mathbf{pr}(R, J \cap I) \subseteq \mathbf{cpj}(S, J \cap I)\}$$

due to 21.3.6

$$= \mathbf{pr}(R, J \cap I) \odot \top_{Y \setminus X}$$

due to 21.3.8(2)(b)

(b) Similar to (a) we obtain

$$R \uparrow Y$$

$$= \cup \{S \in \mathbf{Rel}(Y) \mid S \subseteq R\}$$

$$= \cup \{S \in \mathbf{Rel}(Y) \mid \mathbf{pr}(S, J \cap I) \subseteq \mathbf{cpj}(R, J \cap I)\}$$

$$= \mathbf{cpj}(R, J \cap I) \odot \top_{Y \setminus X}$$

due to 21.3.8(2)(a)

(c) We have

$$R \downarrow Y$$

$$= \mathbf{pr}(R, J \cap I) \odot \top_{Y \setminus X}$$

due to (a)

$$= \mathbf{pr}(R, I \setminus (J \cap I)) \odot \top_{Y \setminus X}$$

def. 21.4.1(1)

$$= \mathbf{pr}(R, I \setminus J) \odot \top_{Y \setminus X}$$

(d) Proof similar to (c).

21.6.6 Lemma

Let X be a proper schema, $R \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$.

- (1) $R \uparrow Y \sqsubseteq R \sqsubseteq R \downarrow Y$
- (2) The following statements are equivalent:
 - (a) $R \Downarrow Y$ is defined
 - (b) $R \uparrow Y = R \downarrow Y$
- (3) In case $R \Downarrow Y$ is actually defined, then

$$R \Downarrow Y = R \uparrow Y = R \downarrow Y$$

21.6.7 Proof of 21.6.6

Suppose, $R = [U, \Gamma]$ with $U = [U_k | k \in K]$ and $Y = [Y_j | j \in J]$.

- (1) We have

$$\begin{aligned}
 & R \uparrow Y \\
 &= \mathbf{pr}(R, K \cap J) \odot \top_{(Y \setminus U)} && \text{due to 21.6.4(3)(b)} \\
 &= \mathbf{pr}(R, K \cap J) \parallel (Y \setminus U) && \text{due to 20.1.3} \\
 &\equiv \mathbf{pr}(R, K \cap J) && \text{due to 20.4.9(2)} \\
 &\sqsubseteq R && \text{due to 21.3.3(9)(b)} \\
 &\sqsubseteq \mathbf{cpj}(R, K \cap J) && \text{due to 21.3.3(9)(a)} \\
 &\equiv \mathbf{cpj}(R, K \cap J) \parallel (Y \setminus U) && \text{due to 20.4.9(2)} \\
 &= \mathbf{cpj}(R, K \cap J) \odot \top_{(Y \setminus U)} && \text{due to 20.1.3} \\
 &= R \downarrow Y && \text{due to 21.6.4(3)(a)}
 \end{aligned}$$

- (2) $R \Downarrow Y$ is defined iff there is a $S \in \mathbf{Rel}(Y)$ with $S \equiv R$. In that case,

$$\begin{aligned}
 R \downarrow Y &= \bigcap \{S \in \mathbf{Rel}(Y) \mid R \sqsubseteq S\} \\
 &\equiv R \Downarrow Y \\
 &\equiv \bigcup \{S \in \mathbf{Rel}(Y) \mid S \sqsubseteq R\} \\
 &= R \uparrow Y
 \end{aligned}$$

- (3) Immediate consequence of (1) and (2).

21.6.8 Remark

The construction of the various reductions by means of double tables was already introduced in 21.1.1. At this point we have all the justifications available, so let us summarize the method for the general case.

Let $X = [X_i | i \in I]$ and $Y = [Y_j | j \in J]$ be two completely finite schemas with $X \smile Y$ and $R = [X, \Gamma]$ a relation.

For the construction of $R \downarrow Y$ and $R \uparrow Y$ we use lemma 21.6.4(3)

$$\begin{aligned}
 R \downarrow Y &= \mathbf{pr}(R, J \cap I) \odot \top_{Y \setminus X} \\
 R \uparrow Y &= \mathbf{cpj}(R, J \cap I) \odot \top_{Y \setminus X}
 \end{aligned}$$

So we make two steps:

- ♣ We first construct the $R' := \mathbf{pr}(R, J \cap I)$ (or $R'' := \mathbf{cpj}(R, J \cap I)$) according to the method 21.2.3.
- ♣ Then we generate $R' \odot \top_{Y \setminus X}$ (or $R'' \odot \top_{Y \setminus X}$) according to 20.1.5.

The constructions for $R \downarrow Y$ and $R \uparrow Y$ also give us a criterion for the existence of $R \Downarrow Y$. Due to 21.6.6(2), $R \Downarrow Y$ is defined iff $R \downarrow Y = R \uparrow Y$. And in that case, all these reductions are

identical according to 21.6.6(2).

21.7 Properties of the reductions

21.7.1 Lemma

Let X be a proper schema.

- (1) For $R = [U, \Gamma] \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$

- (a) $R \downarrow Y = R \downarrow (U \wedge Y) \parallel Y$
- (b) $R \uparrow Y = R \uparrow (U \wedge Y) \parallel Y$

- (2) For $R \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$ holds:

- (a) $R \uparrow Y = \neg((\neg R) \downarrow Y)$
- (b) $R \downarrow Y = \neg((\neg R) \uparrow Y)$

- (3) For $R \in \mathbf{Prel}(X)$ and $Y, Z \in \mathbf{Proj}(X)$ holds:

- (a) $R \downarrow Y = R \parallel Z \downarrow Y$
- (b) $R \uparrow Y = R \parallel Z \uparrow Y$

- (4) For $R, S \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$ holds:

- (a) $R \equiv S$ implies $R \downarrow Y = S \downarrow Y$
- (b) $R \equiv S$ implies $R \uparrow Y = S \uparrow Y$

- (5) For $R, S \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$ holds:

- (a) $R \sqsubseteq S$ implies $R \downarrow Y \sqsubseteq S \downarrow Y$
- (b) $R \sqsubseteq S$ implies $R \uparrow Y \sqsubseteq S \uparrow Y$

- (6) For $R \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$ holds:

- (a) $R \downarrow Y \sqsupseteq R$
- (b) $R \uparrow Y \sqsubseteq R$

- (7) For $R = [U, \Gamma] \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$ holds:

- (a) $U \Downarrow Y$ implies $R \downarrow Y = \begin{cases} \perp_Y & \text{if } R \equiv \perp \\ \top_Y & \text{else} \end{cases}$
- (b) $U \Downarrow Y$ implies $R \uparrow Y = \begin{cases} \top_Y & \text{if } R \equiv \top \\ \perp_Y & \text{else} \end{cases}$

- (8) For every $R \in \mathbf{Prel}(X)$ holds:

- (a) $R \downarrow \langle \rangle = \begin{cases} \perp & \text{if } R \text{ is empty} \\ \top & \text{else} \end{cases}$
- (b) $R \uparrow \langle \rangle = \begin{cases} \top & \text{if } R \text{ is full} \\ \perp & \text{else} \end{cases}$

- (9) For $R \in \mathbf{Prel}(X)$ and $Y, Z \in \mathbf{Proj}(X)$ holds:

- (a) $Y \leq Z$ implies $R \downarrow Y \sqsupseteq R \downarrow Z$
- (b) $Y \leq Z$ implies $R \uparrow Y \sqsubseteq R \uparrow Z$

- (10) For $R \in \mathbf{Prel}(X)$ and $Y \in \mathbf{Proj}(X)$ holds:

$$R \uparrow Y \sqsubseteq R \downarrow Y$$

(11) For every $R \in \mathbf{Prel}(X)$ and all $Y, Z \in \mathbf{Proj}(X)$ holds:

$$R \uparrow Y \downarrow Z \sqsubseteq R \downarrow Z \uparrow Y$$

(12) For all $R \in \mathbf{Prel}(X)$ holds:

$$R \uparrow X = R \parallel X = R \downarrow X = R \uparrow X$$

(13) For all $R \in \mathbf{Prel}(X)$ and $Y_1, \dots, Y_n \in \mathbf{Proj}(X)$ with $n \geq 1$ holds

$$(a) \quad R \downarrow Y_1 \downarrow \dots \downarrow Y_n = R \downarrow \left(\bigwedge_{i \in 1}^n Y_i \right) \parallel Y_n$$

$$(b) \quad R \uparrow Y_1 \uparrow \dots \uparrow Y_n = R \uparrow \left(\bigwedge_{i \in 1}^n Y_i \right) \parallel Y_n$$

21.7.2 Proof of 21.7.1

(1) (a) Recall, that $U, Y \in \mathbf{Proj}(X)$ implies $U \sim Y$ and that $U \wedge Y \leq U$. Now if $J := \mathbf{dom}(U)$ and $K := \mathbf{dom}(Y)$ then

$$\begin{aligned} & R \downarrow Y \\ &= \mathbf{pr}(R, J \cap K) \odot \top_{Y \setminus U} && \text{due to 21.6.4(3)(a)} \\ &= (R \downarrow (U \wedge Y)) \odot \top_{Y \setminus U} && \text{due to 21.6.4(1)(a)} \\ &= R \downarrow (U \wedge Y) \parallel (Y \setminus U) && \text{due to 20.1.3(3)} \\ &= R \downarrow (U \wedge Y) \parallel Y && \text{due to 20.1.3(1)} \end{aligned}$$

(b) Proof similar to (a).

(2) (a) We have

$$\begin{aligned} & R \uparrow Y \\ &= \bigcup \{S \mid S \in \mathbf{Rel}(Y), S \sqsubseteq R\} && \text{def. 21.6.1 of } \uparrow \\ &= \neg \neg \bigcup \{S \mid S \in \mathbf{Rel}(Y), S \sqsubseteq R\} && \text{due to 19.4.6} \\ &= \neg \bigcap \{\neg S \mid S \in \mathbf{Rel}(Y), S \sqsubseteq R\} && \text{due to de Morgans law on } \mathbf{Rel}(Y) \\ &= \neg \bigcap \{S \mid S \in \mathbf{Rel}(Y), \neg S \sqsubseteq R\} \\ &= \neg \bigcap \{S \in \mathbf{Rel}(Y) \mid \neg R \sqsubseteq S\} && \text{due to 20.4.24} \\ &= \neg((\neg R) \downarrow Y) && \text{def. 21.6.1 of } \downarrow \end{aligned}$$

(b) Proof similar to (a).

(3) (a) We have

$$\begin{aligned} & R \parallel Z \downarrow Y \\ &= \bigcap \{S \in \mathbf{Rel}(Y) \mid (R \parallel Z) \sqsubseteq S\} && \text{def. 21.6.1 of } \downarrow \\ &= \bigcap \{S \in \mathbf{Rel}(Y) \mid R \sqsubseteq S\} && \text{because } R \parallel Z \equiv R \\ &= R \downarrow Y && \text{def. 21.6.1, again} \end{aligned}$$

(b) We have

$$\begin{aligned} & R \uparrow Y \\ &= \neg((\neg R) \downarrow Y) && \text{due to (2)(a)} \\ &= \neg((\neg R) \parallel Z \downarrow Y) && \text{due to (a)} \\ &= \neg(\neg(R \parallel Z) \downarrow Y) && \text{due to 20.4.15(3)} \\ &= R \parallel Z \uparrow Y && \text{due to (2)(a), again} \end{aligned}$$

(4) (a) We have

$$\begin{aligned} & R \equiv S \\ &\Rightarrow R \parallel X = S \parallel X && \text{due to 20.4.9(4)(b)} \\ &\Rightarrow R \parallel X \downarrow Y = S \parallel X \downarrow Y \\ &\Rightarrow R \downarrow Y = S \downarrow Y && \text{due to (3)(a)} \end{aligned}$$

(b) Proof similar to (a).

(5) (a) Suppose $Y = [Y_j \mid j \in J]$, then

$$\begin{aligned} & R \sqsubseteq S \\ &\Rightarrow R \parallel X \sqsubseteq S \parallel X && \text{due to 20.4.9(4)(a)} \\ &\Rightarrow \mathbf{pr}(R \parallel X, J) \sqsubseteq \mathbf{pr}(S \parallel X, J) && \text{due to 21.3.3(11)(a)} \\ &\Rightarrow R \parallel X \downarrow Y \sqsubseteq S \parallel X \downarrow Y && \text{due to 21.6.4(1)(a)} \\ &\Rightarrow R \downarrow Y \sqsubseteq S \downarrow Y && \text{due to (3)(a)} \end{aligned}$$

(b) Proof similar to (b).

(6) (a) Suppose $Y = [Y_j \mid j \in J]$, then

$$\begin{aligned} & R \downarrow Y \\ &= R \parallel X \downarrow Y && \text{due to (3)(a)} \\ &= \mathbf{pr}(R \parallel X, J) && \text{due to 21.6.4(1)(a)} \\ &\sqsupseteq R \parallel X && \text{due to 21.3.3(9)(a)} \\ &\equiv R \end{aligned}$$

and so $R \downarrow Y \sqsupseteq R$.

(b) Proof similar to (a).

(7) (a) Suppose $U = [U_i \mid i \in I]$ and $Y = [Y_j \mid j \in J]$, then

$$\begin{aligned} & R \downarrow Y \\ &= \mathbf{pr}(R, L \cap J) \odot \top_{Y \setminus U} && \text{due to 21.6.4(3)(a)} \\ &= \mathbf{pr}(R, \emptyset) \odot \top_Y && \text{because } U \not\ll Y \\ &= \begin{cases} \perp \odot \top_Y = \perp_Y & \text{if } R \text{ is empty, i.e. } R \equiv \perp \\ \top \odot \top_Y = \top_Y & \text{else} \end{cases} && \text{due to 21.3.3(3)(a)} \end{aligned}$$

(b) Proof similar to (a).

(8) For every schema Z holds $Z \not\ll \langle \rangle$. So (8) is just the special case of (7), where $Y = \langle \rangle$.

(9) (a) Suppose $Y = [Y_j \mid j \in J]$ and $Z = [Z_k \mid k \in K]$, then $Y \leq Z$ means $J \subseteq K$. Then

$$\begin{aligned} & R \downarrow Y \\ &= R \parallel X \downarrow Y && \text{due to (3)(a)} \\ &= \mathbf{pr}(R \parallel X, J) && \text{due to 21.6.4(1)} \\ &\sqsupseteq \mathbf{pr}(R \parallel X, K) && \text{due to 21.3.3(8)(a)} \\ &= R \parallel X \downarrow Z \\ &= R \downarrow Z \end{aligned}$$

(b) Proof similar to (a).

(10) From 21.6.6 we obtain $R \uparrow Y \sqsubseteq R \downarrow Y$. And because $\mathbf{x}(R \uparrow Y) = \mathbf{x}(R \downarrow Y)$ we derive $R \uparrow Y \sqsubseteq R \downarrow Y$.

(11) Suppose that $R = [U, \Gamma]$ and $U = [U_i \mid i \in I]$, $Y = [Y_j \mid j \in J]$, and $Z = [Z_k \mid k \in K]$.

For the left hand side of the subvalence expression holds:

$$\begin{aligned} & R \uparrow Y \downarrow Z \\ &= R \uparrow (U \wedge Y) \parallel Y \downarrow Z && \text{due to (1)(b)} \\ &= R \uparrow (U \wedge Y) \downarrow Z && \text{due to (3)(a)} \\ &= R \uparrow (U \wedge Y) \downarrow (U \wedge Y \wedge Z) && \text{due to (1)(a)} \\ &\equiv R \uparrow (U \wedge Y) \downarrow (U \wedge Y \wedge Z) && \text{due to 20.4.9(2)} \end{aligned}$$

And because

$$\begin{aligned} @ (R) &= I \\ @ (R \uparrow (U \wedge Y)) &= \mathbf{dom}(U \wedge Y) \\ &= I \cap J \\ &= I \setminus (I \setminus J) \\ @ (R \uparrow (U \wedge Y) \downarrow (U \wedge Y \wedge Z)) &= I \cap J \cap K \\ &= (I \setminus (I \setminus J)) \setminus (I \setminus K) \end{aligned}$$

we obtain

$$R \uparrow (U \wedge Y) \downarrow (U \wedge Y \wedge Z) = R \uparrow (I \setminus J) \downarrow (I \setminus K)$$

by applying 21.6.4(1)(c) and (d).

Performing similar steps on the right hand side of the subvalence expression gives us

$$\begin{aligned} R \Downarrow Z \Uparrow Y &\equiv R \Downarrow (U \wedge Z) \Uparrow (U \wedge Y \wedge Z) \\ &= R \downarrow (I \setminus K) \uparrow (I \setminus J) \end{aligned}$$

Having translated the reductions into eliminations this way, we can now apply 21.5.3(17):

$$\begin{aligned} R \Uparrow Y \Downarrow Z &\equiv R \uparrow (I \setminus J) \downarrow (I \setminus K) \\ &\subseteq R \downarrow (I \setminus K) \uparrow (I \setminus J) \\ &\equiv R \Downarrow Z \Uparrow Y \end{aligned}$$

and thus

$$R \Uparrow Y \Downarrow Z \sqsubseteq R \Downarrow Z \Uparrow Y$$

(12) Suppose $R = [Y, \Gamma]$. Then $Y \leq X$ and

$$R \Downarrow X = R \parallel X \quad \text{due to 21.6.4(2)(a)}$$

$$R \Uparrow X = R \parallel X \quad \text{due to 21.6.4(2)(b)}$$

Furthermore, $R \parallel X \equiv R$ according to 20.4.9(2). So there is a (unique) $R \in \mathbf{Rel}(X)$ equivalent to R , namely $R \parallel X$, and so $R \parallel X = R \Downarrow X$.

(13) Let $R = [Y, \Gamma]$ with $U = [U_i | i \in I]$ and $\mathbf{dom}(Y_k) = J_i$ for $k = 1, \dots, n$.

(a) We proof by induction on n :

♣ For $n = 1$ we have

$$R \Downarrow Y_1 = R \Downarrow Y_1 \parallel Y_1$$

which is a true statement according to 20.1.11(3)

♣ For $n \mapsto n + 1$ we obtain

$$\begin{aligned} R \Downarrow Y_1 \Downarrow \dots \Downarrow Y_n \Downarrow Y_{n+1} & \\ = R \Downarrow \left(\bigwedge_{i=1}^n Y_i \right) \parallel Y_n \Downarrow Y_{n+1} & \quad \text{by induction} \\ = R \Downarrow \left(\bigwedge_{i=1}^n Y_i \right) \Downarrow Y_{n+1} & \quad \text{due to (3)(a)} \\ = R \Downarrow \left(\bigwedge_{i=1}^{n+1} Y_i \right) \parallel Y_{n+1} & \quad \text{due to (1)(a)} \end{aligned}$$

(b) Proof similar to (a).

21.7.3 Lemma

For each schema X , all $\mathcal{R} = \{\mathcal{R}_k | k \in K\} \subseteq \mathbf{Prel}(X)$ and $R, S \in \mathbf{Prel}(X)$, and every $Y \in \mathbf{Proj}(X)$ holds:

- (1) $\perp \Downarrow Y = \perp_Y$
- (2) $\perp \Uparrow Y = \perp_Y$
- (3) $\top \Downarrow Y = \top_Y$
- (4) $\top \Uparrow Y = \top_Y$
- (5) $(\neg R) \Downarrow Y = \neg(R \Uparrow Y)$
- (6) $(\neg R) \Uparrow Y = \neg(R \Downarrow Y)$
- (7) $(R \sqcap S) \Downarrow Y \subseteq (R \Downarrow Y) \cap (S \Downarrow Y)$
- (8) $(R \sqcap S) \Uparrow Y = (R \Uparrow Y) \cap (S \Uparrow Y)$
- (9) $(R \sqcup S) \Downarrow Y = (R \Downarrow Y) \cup (S \Downarrow Y)$
- (10) $(R \sqcup S) \Uparrow Y \supseteq (R \Uparrow Y) \cup (S \Uparrow Y)$
- (11) $(\prod \mathcal{R}) \Downarrow Y \subseteq \bigcap \{\mathcal{R}_k \Downarrow Y | k \in K\}$
- (12) $(\prod \mathcal{R}) \Uparrow Y = \bigcap \{\mathcal{R}_k \Uparrow Y | k \in K\}$
- (13) $(\prod \mathcal{R}) \Downarrow Y = \bigcup \{\mathcal{R}_k \Downarrow Y | k \in K\}$
- (14) $(\prod \mathcal{R}) \Uparrow Y \supseteq \bigcup \{\mathcal{R}_k \Uparrow Y | k \in K\}$

(1) We obtain

$$\begin{aligned} \perp \Downarrow Y & \\ &= \bigcap \{T \in \mathbf{Rel}(Y) | \perp \sqsubseteq T\} \quad \text{def. 21.6.1 of } \Downarrow \\ &= \bigcap \mathbf{Rel}(Y) \quad \text{due to 20.4.21(17)} \\ &= \perp_Y \end{aligned}$$

(2) Similar to (1) we obtain

$$\begin{aligned} \perp \Uparrow Y & \\ &= \bigcup \{T \in \mathbf{Rel}(Y) | T \sqsubseteq \perp\} \quad \text{def. 21.6.1 of } \Uparrow \\ &= \bigcup \{T \in \mathbf{Rel}(Y) | T \subseteq \perp_Y\} \\ &= \bigcup \{\top_Y\} \\ &= \perp_Y \end{aligned}$$

(3) Similar to (1) and (2).

(4) Similar to (1) and (2).

(5) We have

$$\begin{aligned} (\neg R) \Downarrow Y & \\ &= \neg \neg ((\neg R) \Downarrow Y) \quad \text{due to 19.4.6} \\ &= \neg(R \Uparrow Y) \quad \text{due to 21.7.1(2)(b)} \end{aligned}$$

(6) Similar to (5).

(7) Due to

$$R \sqcap S = \prod R, S$$

$$(R \parallel Y) \sqcap (S \parallel Y) = \prod \{(R \parallel Y), (S \parallel Y)\}$$

this case (7) is just a special case of (11) (proof see below).

(8) Is a special case of (12).

(9) Is a special case of (13).

(10) Is a special case of (14).

(11) We have

$$\begin{aligned} (\prod \mathcal{R}) \Downarrow Y & \\ &= (\prod \mathcal{R}) \parallel X \Downarrow Y \quad \text{due to 21.7.1(5)(c)} \\ &= \prod \{\mathcal{R}_k \parallel X | k \in K\} \Downarrow Y \quad \text{due to 20.4.15(8)} \\ &= \bigcap \{\mathcal{R}_k \parallel X | k \in K\} \Downarrow Y \quad \text{def. 20.2.4 of } \prod \\ &= \mathbf{pr} (\bigcap \{\mathcal{R}_k \parallel X | k \in K\}, J) \quad \text{due to 21.6.4(1)(a)} \\ &\subseteq \bigcap \{\mathbf{pr} (\mathcal{R}_k \parallel X, J) | k \in K\} \quad \text{due to 21.3.10(1)} \\ &= \bigcap \{\mathcal{R}_k \parallel X \Downarrow Y | k \in K\} \quad \text{due to 21.6.4(1)(a), again} \\ &= \bigcap \{\mathcal{R}_k \Downarrow Y | k \in K\} \quad \text{due to 21.7.1(5)(c), again} \end{aligned}$$

(12) Similar to (11), we obtain

$$\begin{aligned} (\prod \mathcal{R}) \Uparrow Y & \\ &= (\prod \mathcal{R}) \parallel X \Uparrow Y \quad \text{due to 21.7.1(5)(d)} \\ &= \prod \{\mathcal{R}_k \parallel X | k \in K\} \Uparrow Y \quad \text{due to 20.4.15(8)} \\ &= (\bigcap \{\mathcal{R}_k \parallel X | k \in K\}) \Uparrow Y \quad \text{def. 20.2.4 of } \prod \\ &= \mathbf{pr} (\bigcap \{\mathcal{R}_k \parallel X | k \in K\}, J) \quad \text{due to 21.6.4(1)(b)} \\ &= \bigcap \{\mathbf{pr} (\mathcal{R}_k \parallel X, J) | k \in K\} \quad \text{due to 21.3.10(1)} \\ &= \bigcap \{\mathcal{R}_k \parallel X \Uparrow Y | k \in K\} \quad \text{due to 21.6.4(1)(b), again} \\ &= \bigcap \{\mathcal{R}_k \Uparrow Y | k \in K\} \quad \text{due to 21.7.1(5)(d), again} \end{aligned}$$

(13) Similar to (11) and (12), by applying 21.3.10(3).

(14) Similar to (11) and (12), by applying 21.3.10(4).

21.7.4 Proof of 21.7.3

Let $X = [X_i | i \in I]$ and $\mathbf{dom}(Y) = J$, so that $Y = [X_j | j \in J]$.

21.7.5 Lemma

Let $R = [X, \Gamma]$ and $S = [Y, \Sigma]$ be two compatible relations. We put $Z := X \wedge Y$.

(1) The following statements are equivalent:

- (a) $R \sqsubseteq S$
- (b) $R \Downarrow Z \subseteq S \Uparrow Z$
- (c) $R \Downarrow Y \subseteq S \Uparrow X$

(2) The following statements are equivalent as well:

- (a) $R \equiv S$
- (b) $R \Downarrow Z = R \Uparrow Z = S \Downarrow Z = S \Uparrow Z$
- (c) $R \Updownarrow Z$ and $S \Updownarrow Z$ both exist and they are equal.

21.7.6 Proof of 21.7.5

Suppose, $X = [X_i | i \in I]$ and $Y = [Y_j | j \in J]$.

(1) (a) and (b) are equivalent, because

$$\begin{aligned} R \sqsubseteq S & \\ \Leftrightarrow \mathbf{pr}(R, I \cap J) \subseteq \mathbf{cpj}(S, I \cap J) & \text{ due to 21.3.6} \\ \Leftrightarrow R \Downarrow Z \subseteq S \Uparrow Z & \text{ due to 21.6.4(1)(a) and (b)} \end{aligned}$$

(b) and (c) are equivalent, because

$$\begin{aligned} R \Downarrow Y & \\ = R \Downarrow Z \parallel Y & \text{ due to 21.7.1(1)(a)} \\ \equiv R \Downarrow Z & \text{ due to 20.4.9(2)} \\ S \Uparrow X & \\ = S \Uparrow Z \parallel X & \text{ due to 21.7.1(1)(b)} \\ \equiv S \Uparrow Z & \text{ due to 20.4.9(2)} \end{aligned}$$

and thus

$$R \Downarrow Z \subseteq S \Uparrow Z \text{ iff } R \Downarrow Y \subseteq S \Uparrow X$$

(2) (a) and (b) are equivalent, because

$$\begin{aligned} R \equiv S & \\ \text{iff } R \sqsubseteq S \text{ and } S \sqsubseteq R & \text{ due to 20.4.9(1)} \\ \text{iff } R \Downarrow Z \subseteq S \Uparrow Z \text{ and } S \Downarrow Z \subseteq R \Uparrow Z & \text{ due to (1)} \\ \text{iff } R \Downarrow Z = R \Uparrow Z = S \Downarrow Z = S \Uparrow Z & \text{ because } R \Uparrow Z \subseteq R \Downarrow Z \\ & \text{and } S \Uparrow Z \subseteq S \Downarrow Z \\ & \text{according to 21.7.1(10)} \end{aligned}$$

(b) and (c) equivalent, due to 21.6.6

♣ $R \Updownarrow Z$ is defined iff $R \Uparrow Z = R \Downarrow Z$ (then also $= R \Updownarrow Z$).

♣ $S \Updownarrow Z$ is defined iff $S \Uparrow Z = S \Downarrow Z$ (then also $= S \Updownarrow Z$).

and that entails the equivalence of (b) and (c).

21.7.7 Lemma

Let $X = [X_i | i \in I]$ be a proper schema, $[\sigma_k | k \in K]$ a schema partition of X , and for each $k \in K$ let $R_k \in \mathbf{Prel}(\sigma_k)$ and $Y_k \in \mathbf{Proj}(\sigma_k)$. Then

$$(a) \left(\bigcirc_{k \in K} R_k \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) = \bigcirc_{k \in K} (R_k \parallel Y_k)$$

$$(b) \left(\bigcirc_{k \in K} R_k \right) \Downarrow \left(\dot{\bigvee}_{k \in K} Y_k \right) = \bigcirc_{k \in K} (R_k \Downarrow Y_k)$$

$$(c) \left(\bigcirc_{k \in K} R_k \right) \Uparrow \left(\dot{\bigvee}_{k \in K} Y_k \right) = \bigcirc_{k \in K} (R_k \Uparrow Y_k)$$

21.7.8 Proof of 21.7.7

First, let us state that every distinct product expression of 21.7.7 is well defined: the R_k are pairwise distinct by definition. And with $R_k \parallel Y_k \in \mathbf{Prel}(\sigma_k)$ and pairwise distinct σ_k , the $R_k \parallel Y_k$ are pairwise distinct as well and their distinct product is well-defined. The same holds for the supremum and infimum reduction expressions.

For the subsequent proof we will use the following fact from the algebra of classes:

(i) Let C and K be classes, $\{A_k \mid k \in K\} \subseteq \mathbf{P}(C)$, and $\{B_k \mid k \in K\} \subseteq \mathbf{P}(C)$, such that $k_1 \neq k_2$ implies $A_{k_1} \cap B_{k_2} = \emptyset$, for all $k_1, k_2 \in K$. Then

$$\left(\bigcup_{k \in K} A_k \right) \cap \left(\bigcup_{k \in K} B_k \right) = \bigcup_{k \in K} (A_k \cap B_k)$$

Now, for each $k \in K$, let $U_k = \mathbf{x}(R_k)$, $L_k = \mathbf{@}(R_k) = \mathbf{dom}(U_k)$, and $J_k = \mathbf{dom}(Y_k)$.

(a) The given statements is

$$\left(\bigcirc_{k \in K} R_k \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) = \bigcirc_{k \in K} (R_k \parallel Y_k)$$

According to 20.4.9(5), both sides of the stated equation are equal iff they are both equivalent and have the same schema. Now the equivalence holds due to

$$\begin{aligned} \left(\bigcirc_{k \in K} R_k \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) & \equiv \left(\bigcirc_{k \in K} R_k \right) \\ & \equiv \bigcirc_{k \in K} (R_k \parallel Y_k) \end{aligned}$$

and for the schemas holds

$$\begin{aligned} \mathbf{x} \left(\left(\bigcirc_{k \in K} R_k \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) \right) & = \left(\dot{\bigvee}_{k \in K} U_k \right) \vee \left(\dot{\bigvee}_{k \in K} Y_k \right) \\ & = \bigvee_{k \in K} (U_k \vee Y_k) \\ & = \mathbf{x} \left(\bigcirc_{k \in K} (R_k \parallel Y_k) \right) \end{aligned}$$

(b) We have

$$\begin{aligned} \left(\bigcirc_{k \in K} R_k \right) \Downarrow \left(\dot{\bigvee}_{k \in K} Y_k \right) & \\ = \mathbf{pr} \left(\bigcirc_{k \in K} R_k, \left(\bigcup_{k \in K} L_k \right) \cap \left(\bigcup_{k \in K} J_k \right) \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) & \text{ due to 21.6.4(3)(a) and 20.1.3(3)} \\ = \mathbf{pr} \left(\bigcirc_{k \in K} R_k, \bigcup_{k \in K} (L_k \cap J_k) \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) & \text{ due to (i)} \\ = \left(\bigcirc_{k \in K} \mathbf{pr}(R_k, L_k \cap J_k) \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) & \text{ due to 21.3.16(1)} \\ = \left(\bigcirc_{k \in K} (R_k \Downarrow (U_k \wedge Y_k)) \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) & \text{ due to 21.6.4(1)(a)} \\ = \bigcirc_{k \in K} (R_k \Downarrow (U_k \wedge Y_k) \parallel Y_k) & \text{ due to (a)} \\ = \bigcirc_{k \in K} (R_k \Downarrow Y_k) & \text{ due to 21.7.1(1)(a)} \end{aligned}$$

(c) Similar to the proof of (b) we have

$$\begin{aligned} \left(\bigcirc_{k \in K} R_k \right) \Uparrow \left(\dot{\bigvee}_{k \in K} Y_k \right) & \\ = \mathbf{cpj} \left(\bigcirc_{k \in K} R_k, \left(\bigcup_{k \in K} L_k \right) \cap \left(\bigcup_{k \in K} J_k \right) \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) & \text{ due to 21.6.4(3)(b) and 20.1.3(3)} \\ = \mathbf{cpj} \left(\bigcirc_{k \in K} R_k, \bigcup_{k \in K} (L_k \cap J_k) \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) & \text{ due to (i)} \\ = \left(\bigcirc_{k \in K} \mathbf{cpj}(R_k, L_k \cap J_k) \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) & \text{ due to 21.3.16(2)} \\ = \left(\bigcirc_{k \in K} (R_k \Uparrow (U_k \wedge Y_k)) \right) \parallel \left(\dot{\bigvee}_{k \in K} Y_k \right) & \text{ due to 21.6.4(1)(b)} \\ = \bigcirc_{k \in K} (R_k \Uparrow (U_k \wedge Y_k) \parallel Y_k) & \text{ due to (a)} \end{aligned}$$

$$= \bigcirc_{k \in K} (R_k \uparrow Y_k) \quad \text{due to 21.7.1(1)(b)}$$

21.8 Redundancy of attributes

21.8.1 Remark introduction

We saw in 21.5.3(15) and (16), that

$$R \uparrow j \sqsubseteq R \sqsubseteq R \downarrow j$$

for every relation R and attribute j . In fact, the left subvalence is proper iff the right one is, i.e. exactly one of the following situations is the case: Either

- (a) $R \uparrow j \equiv R \equiv R \downarrow j$ or
- (b) $R \uparrow j \sqsubset R \sqsubset R \downarrow j$.

In case of (a), we say that j is **redundant** for R . We can eliminate j from R without losing any information, where “same information” means “being equivalent”. Neither does it matter how we eliminate j from R — supremum or infimum elimination, it is all the same in this case.

21.8.2 Definition

Let $X = [X_i | i \in I]$ be a proper schema and $R \in \mathbf{Prel}(X)$.

(1) We call every $j \in I$

(a) **redundant for R** , iff

$$R \uparrow j = R \downarrow j$$

(b) **irredundant for R** , iff

$$R \uparrow j \neq R \downarrow j$$

And if j is an attribute of R , we call it (ir)redundant **in** R .

(2) Each $J \subseteq I$ is called

(a) **redundant for R** , if each $j \in J$ is redundant for R

(b) **irredundant for R** , if each $j \in J$ is irredundant for R

And if $J \subseteq @ (R)$ we call J (ir)redundant **in** R , accordingly.

(3) We define

$$\mathbf{redAt}(R) := \{j \in @ (R) \mid R \uparrow j = R \downarrow j\}$$

the **redundant attribute class** of R

$$\mathbf{irrAt}(R) := \{j \in @ (R) \mid R \uparrow j \neq R \downarrow j\}$$

the **irredundant attribute class** of R

Note that in definition 21.8.2, each single $j \in I$ is either redundant or irredundant for R . But a subclass $J \subseteq I$ that has both redundant and irredundant members, is itself neither redundant nor irredundant for R .

Another terminology for “redundant” is “negative” and for “irredundant” is “positive”. But we will not use these terms here.

21.8.4 Example and table methods

Consider a relation R with schema X , where

$$X = \begin{bmatrix} a \mapsto \mathbb{B} \\ b \mapsto \mathbb{B} \\ c \mapsto \mathbb{B} \\ d \mapsto \mathbb{B} \end{bmatrix} \quad \text{and} \quad R = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ \hline \end{array}$$

(Actually, R is the truth table of the propositional formula $[(a \vee \neg b) \wedge \neg c \wedge [d \vee \neg d]]$.)

Double tables (introduced in 17.6.1) provide an intuitive tool to decide, if a certain attribute, say a , is redundant. According to 21.6.4(1)(c) and (d), an elimination of a is a reduction onto $\mathbf{pr}(X, \{b, c, d\})$. And this is done with double tables. We first represent R as a double table with a as the only attribute in the top schema:

$$R = \begin{array}{|c|c|c|c|c|} \hline & & & 0 & 1 & a \\ \hline 0 & 0 & 0 & 1 & 1 & \\ 1 & 0 & 0 & 0 & 1 & \\ 0 & 1 & 0 & 0 & 0 & \\ 1 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 1 & 1 & \\ 1 & 0 & 1 & 0 & 1 & \\ 0 & 1 & 1 & 0 & 0 & \\ 1 & 1 & 1 & 0 & 0 & \\ \hline b & c & d & & & \\ \hline \end{array}$$

Boolean conjunction \wedge of the two values in each row produces the infimum elimination:

$$R \uparrow a = \begin{array}{|c|c|c|c|} \hline b & c & d & \\ \hline 0 & 0 & 0 & 1 \wedge 1 \\ 1 & 0 & 0 & 0 \wedge 1 \\ 0 & 1 & 0 & 0 \wedge 0 \\ 1 & 1 & 0 & 0 \wedge 0 \\ 0 & 0 & 1 & 1 \wedge 1 \\ 1 & 0 & 1 & 0 \wedge 1 \\ 0 & 1 & 1 & 0 \wedge 0 \\ 1 & 1 & 1 & 0 \wedge 0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline b & c & d & \\ \hline 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ \hline \end{array}$$

Similarly, we obtain the supremum elimination:

21.8.3 Remark Remark

$$R \downarrow a = \begin{array}{|c|c|c|} \hline b & c & d \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline b & c & d \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

We see here that the two resulting tables are different, so $R \uparrow a \neq R \downarrow a$ and thus a is irredundant in R .

On the other hand, if we perform the same process for the eliminations of d , we find $R \uparrow d = R \downarrow d$:

$$\begin{array}{|c|c|c|} \hline 0 & 1 & d \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline a & b & c \\ \hline \end{array} \quad \begin{array}{|c|c|c|} \hline a & b & c \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|} \hline a & b & c \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$R \qquad R \uparrow d \qquad R \downarrow d$

So d is redundant. If we do this test for the remaining two attributes b and c , we will find that the attribute class of R partitions into the following two subclasses:

$$\begin{aligned} \text{redAt}(R) &= \{d\} \\ \text{irrAt}(R) &= \{a, b, c\} \end{aligned}$$

21.8.5 Remark

We mentioned, that the table R of the previous example 21.8.4 is the *truth table* of the propositional formula

$$\varphi = [[a \vee \neg b] \wedge \neg c \wedge [d \vee \neg d]]$$

This provides us with another clue for the understanding of the redundancy concept. The subformula $[d \vee \neg d]$ is equivalent to $\mathbf{1}$. So it is a neutral element of this conjunction and may be deleted without changing the boolean semantics. In other words, $[[a \vee \neg b] \wedge \neg c]$ is still equivalent to φ . That d is *redundant* if φ is saying that there is an equivalent formula without d .

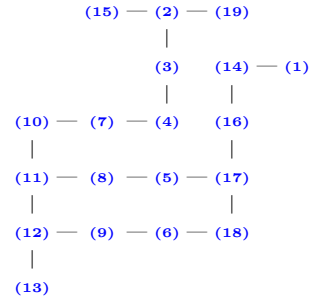
21.8.6 Lemma

Let $X = [X_i | i \in J]$ be a proper schema, $R \in \text{Pre1}(X)$ and $J \subseteq I$. With $K := @ (R)$ and $Y := \text{pr}(X, K \setminus J)$ the following statements are equivalent:

- (1) J is redundant for R
- (2) $K \cap J$ is redundant in R
- (3) $R \uparrow (K \cap J) = R \downarrow (K \cap J)$
- (4) $R \uparrow J = R \downarrow J$
- (5) $R \downarrow J \equiv R$
- (6) $R \uparrow J \equiv R$
- (7) $\text{pr}(R, K \setminus J) = \text{cpj}(R, K \setminus J)$
- (8) $\text{pr}(R, K \setminus J) \equiv R$
- (9) $\text{cpj}(R, K \setminus J) \equiv R$
- (10) $R \uparrow Y = R \downarrow Y$
- (11) $R \downarrow Y \equiv R$
- (12) $R \uparrow Y \equiv R$
- (13) $R \uparrow \downarrow Y$ is well-defined
- (14) Each $j \in J$ is redundant for R
- (15) Each $j \in (K \cap J)$ is redundant in R
- (16) $R \uparrow j = R \downarrow j$ for each $j \in J$
- (17) $R \downarrow j \equiv R$ for each $j \in J$
- (18) $R \uparrow j \equiv R$ for each $j \in J$
- (19) $K \cap J \subseteq \text{redAt}(R)$

21.8.7 Proof of 21.8.6

Our proof pictures the following network of equivalences:



The details of this argumentation:

- (14) \Leftrightarrow (16) due to definition 21.8.2(1)(a).
- (1) \Leftrightarrow (14) is exactly definition 21.8.2(2)(a).
- (2) \Leftrightarrow (15) again due to definition 21.8.2.
- (2) \Leftrightarrow (19) due to definition 21.8.2, once again.
- (3) \Leftrightarrow (4) holds due to 21.5.3(9) and (10), saying that $R \downarrow J = R \downarrow (K \cap J)$ and $R \uparrow J = R \uparrow (K \cap J)$.
- (10) \Leftrightarrow (11) \Leftrightarrow (12) \Leftrightarrow (13) is true according to 21.6.6.
- (5) \Leftrightarrow (8) \Leftrightarrow (11) due to 21.6.4.
- (6) \Leftrightarrow (9) \Leftrightarrow (12) due to 21.6.4.
- (4) \Leftrightarrow (7) \Leftrightarrow (10) due to 21.6.4.
- (17) \Leftrightarrow (18) \Leftrightarrow (16) \Leftrightarrow (17) \wedge (18). And (17) \Leftrightarrow (18), as we know by now. Thus (16) \Leftrightarrow (17) and (16) \Leftrightarrow (18).
- (5) \Leftrightarrow (17) \Leftrightarrow (18) \Leftrightarrow (6) due to 21.5.9.
- (2) \Leftrightarrow (3) By now we have proof for (1) \Leftrightarrow (4). It follows that: $K \cap J$ is redundant for R iff (3). And since $K \cap J \subseteq K$, the statements “ $K \cap J$ is redundant for R ” and “ $K \cap J$ is redundant in R ” are equivalent.

22 Attribute translation

22.1 Introduction

22.1.1 Example

Consider an example relation R , given as

$$R := \begin{array}{|c|c|c|} \hline \text{child} : H & \text{father} : H & \text{mother} : H \\ \hline c_1 & f_1 & m_1 \\ \hline c_2 & f_2 & m_2 \\ \hline c_3 & f_3 & m_3 \\ \hline \end{array}$$

where H is the class of human beings (or less suggestive and more formal, a class of names or strings). Also given a little *translator*, i.e. a function (or record) between attributes, by

$$\tau := \begin{array}{|c|} \hline \text{child} \mapsto \text{kind} \\ \hline \text{father} \mapsto \text{vater} \\ \hline \text{mother} \mapsto \text{mutter} \\ \hline \end{array}$$

We define $\text{tr}(R, \tau)$, the (*attribute*) translation of R by τ , so that in our particular example

$$\text{tr}(R, \tau) := \begin{array}{|c|c|c|} \hline \text{kind} : H & \text{vater} : H & \text{mutter} : H \\ \hline c_1 & f_1 & m_1 \\ \hline c_2 & f_2 & m_2 \\ \hline c_3 & f_3 & m_3 \\ \hline \end{array}$$

The result is a “German translation” of the original “English relation”. In this case, the translation is simply a substitution of the attributes in the table.

22.1.2 Example

But things are not always that plain. Taking the same relation R from 22.1.1, but another translator

$$\tau := \begin{array}{|c|} \hline \text{father} \mapsto \text{parent} \\ \hline \text{mother} \mapsto \text{parent} \\ \hline \end{array}$$

we could argue if $\text{tr}(R, \tau)$ can be well-defined at all. But we do want a very general translation concept and we apply the following principles for its definition.

First of all, if R has an attribute that doesn't occur in the domain of τ , we leave the attribute unchanged. In our example, *child* is such an attribute. So at this stage, our example becomes

$$\text{tr}(R, \tau) := \begin{array}{|c|c|c|} \hline \text{child} : H & \text{parent} : H & \text{parent} : H \\ \hline c_1 & f_1 & m_1 \\ \hline c_2 & f_2 & m_2 \\ \hline c_3 & f_3 & m_3 \\ \hline \end{array}$$

The result is not a well-defined relation, there cannot be two columns with the same attributes. So we merge the two *parent* columns into one. In order to do that, we treat each of the three graph records accordingly. For each of the three rows $\xi \in R$ we produce the translation $\text{tr}(\xi, \tau)$. But for, say the first member

$$\xi = \begin{array}{|c|} \hline \text{child} \mapsto c_1 \\ \hline \text{parent} \mapsto f_1 \\ \hline \text{parent} \mapsto m_1 \\ \hline \end{array}$$

the translation $\text{tr}(\xi, \tau)$ is defined only, if $f_1 = m_1$. But we assume instead, that fathers and mothers are always two different specimens. $\text{tr}(\xi, \tau)$ is undefined for this ξ and the other two members as well. Thus we end up with

$$\text{tr}(R, \tau) = \begin{array}{|c|c|} \hline \text{child} : H & \text{parent} : H \\ \hline \end{array}$$

However empty, the result is still well-defined. Nevertheless, there are also examples of relations R and translators τ , where $\text{tr}(R, \tau)$ is not reasonably definable.

22.1.3 Remark

In general, the τ -translation $\text{tr}(R, \tau)$ of a relation $R = [X, \Gamma]$ is based on the τ -translation of records. The general method has two steps:

- ♣ First, produce the τ -translation $X' := \text{tr}(X, \tau)$ of the schema. If X' is not well defined, the whole translation of R fails and aborts here. Otherwise:
- ♣ For each member $x \in \Gamma$, produce the τ -translation $x' := \text{tr}(x, \tau)$. If x' is well-defined, it becomes a member of the new graph Γ' . Eventually, $[X', \Gamma']$ is returned as the overall result.

In 22.2, we first develop a general definition of record translations. The translation of relations is fully introduced in 22.4.

22.2 Translation of records

22.2.1 Remark —Generalization of images and image classes—

Let $f : D \rightarrow C$ be a function.

Recall (5.7.3 and 5.7.18), that

- (1) for every $d \in D$,

$$f(d)$$

is the f -value of d or f -image of d

- (2) for every $D' \subseteq D$,

$$f[D'] := \{f(d) \mid d \in D'\}$$

is the f -image class of D'

We will now generalize these notions as follows:

- (3) For every class A and $a \in A$ we define

$$f[[a]] := \begin{cases} f(a) & \text{if } a \in D \\ a & \text{if } a \notin D \end{cases}$$

(4) For every class A we define

$$f[[A]] := \{f[[a]] \mid a \in A\}$$

Note, that,

(5) $f[[a]]$ and $f[[A]]$ are proper generalizations of $f(a)$ and $f[A]$, respectively, in the sense that: for every $f : D \rightarrow C$ and if $a \in A \subseteq D$, then $f[[a]] = f(a)$ and $f[[A]] = f[A]$. So we could have used the same notation again.

(6) For every $f : D \rightarrow C$ and each class A we have

$$f[[A]] = (D \setminus A) \cup f[D \cap A]$$

22.2.2 Remark _____ records and pair classes _____

Recall, that functions can either be given as prescriptions (how to map given things to another thing) or as left-unique pair classes (i.e. argument-value pairs). We repeat this relation here again, but especially for records.

(1) Let A, B be two classes and $\Pi \subseteq A \times B$. We put

$$L := \{a \mid \langle a, b \rangle \in \Pi\}$$

$$R := \{b \mid \langle a, b \rangle \in \Pi\}$$

and we say that Π is

♣ right unique iff

for each $a \in L$ there is exactly one $b \in R$ with $\langle a, b \rangle \in \Pi$

♣ left unique iff

for each $b \in R$ there is exactly one $a \in L$ with $\langle a, b \rangle \in \Pi$

And in case Π is indeed right unique, we define

$$\mathbf{rec}(\Pi) := [a \mapsto \mathbf{the } b \in R \text{ with } \langle a, b \rangle \in \Pi \mid a \in L]$$

the record of Π .

(2) On the other hand, let $\xi = [\xi_i \mid i \in I]$ is a record, then

$$\mathbf{gr}(\xi) := \{\langle i, \xi_i \rangle \mid i \in I\}$$

is the graph of ξ .

Obviously, $\mathbf{gr}(\xi)$ is a right-unique pair class.

Right-unique pair classes and records can thus be seen as two representations of the same idea in the sense that

(3) for every right-unique pair class Π holds

$$\mathbf{gr}(\mathbf{rec}(\Pi)) = \Pi$$

(4) for every record ξ holds

$$\mathbf{rec}(\mathbf{gr}(\xi)) = \xi$$

In case of finite examples, this is somehow simply another way of writing:

$$\begin{array}{ccc} \begin{bmatrix} i_1 \mapsto \xi_1 \\ i_2 \mapsto \xi_2 \\ \vdots \\ i_n \mapsto \xi_n \end{bmatrix} & \begin{array}{c} \xrightarrow{\mathbf{gr}} \\ \xleftarrow{\mathbf{rec}} \end{array} & \begin{Bmatrix} \langle i_1, \xi_1 \rangle \\ \langle i_2, \xi_2 \rangle \\ \vdots \\ \langle i_n, \xi_n \rangle \end{Bmatrix} \\ \text{record} & & \text{right-unique} \\ & & \text{pair class} \end{array}$$

22.2.3 Definition _____

Let τ and $\xi = [\xi_i \mid i \in I]$ be two records. We define

$$\mathbf{tr}(\xi, \tau) := \mathbf{rec}(\Xi) \quad \text{where} \quad \Xi := \{\langle \tau([i]), \xi_i \rangle \mid i \in I\}$$

is the (attribute) translation of ξ by τ
or the τ -translation of ξ .

Note, that $\mathbf{tr}(\xi, \tau)$ is well-defined iff Ξ is right-unique. In that case, we say that ξ is translatable by τ (or τ -translatable).

22.2.4 Example _____

Let us investigate some examples similar to the situation in 22.1.1 and 22.1.2.

(1) Given

$$\xi = \begin{bmatrix} \text{child} \mapsto c \\ \text{father} \mapsto f \\ \text{mother} \mapsto m \end{bmatrix} \quad \text{and} \quad \tau = \begin{bmatrix} \text{child} \mapsto \text{kind} \\ \text{father} \mapsto \text{vater} \\ \text{mother} \mapsto \text{mutter} \end{bmatrix}$$

we put

$$\Xi := \left\{ \begin{array}{l} \langle \tau([\text{child}]), \xi([\text{child}]) \rangle \\ \langle \tau([\text{father}]), \xi([\text{father}]) \rangle \\ \langle \tau([\text{mother}]), \xi([\text{mother}]) \rangle \end{array} \right\} = \left\{ \begin{array}{l} \langle \text{kind}, c \rangle \\ \langle \text{vater}, f \rangle \\ \langle \text{mutter}, m \rangle \end{array} \right\}$$

and obtain

$$\mathbf{tr}(\xi, \tau) = \mathbf{rec}(\Xi) = \begin{bmatrix} \text{kind} \mapsto c \\ \text{vater} \mapsto f \\ \text{mutter} \mapsto m \end{bmatrix}$$

(2) Given

$$\xi = \begin{bmatrix} \text{child} \mapsto c \\ \text{father} \mapsto f \\ \text{mother} \mapsto m \end{bmatrix} \quad \text{and} \quad \tau = \begin{bmatrix} \text{father} \mapsto \text{parent} \\ \text{mother} \mapsto \text{parent} \end{bmatrix}$$

we put

$$\Xi := \left\{ \begin{array}{l} \langle \tau([\text{child}]), \xi([\text{child}]) \rangle \\ \langle \tau([\text{father}]), \xi([\text{father}]) \rangle \\ \langle \tau([\text{mother}]), \xi([\text{mother}]) \rangle \end{array} \right\} = \left\{ \begin{array}{l} \langle \text{child}, c \rangle \\ \langle \text{parent}, f \rangle \\ \langle \text{parent}, m \rangle \end{array} \right\}$$

Ξ is not right-unique, $\mathbf{tr}(\xi, \tau)$ is undefined.

(3) If we take the same τ and the schema X of the example relation R from 22.1.1, i.e.

$$X = \begin{bmatrix} \text{child} \mapsto H \\ \text{father} \mapsto H \\ \text{mother} \mapsto H \end{bmatrix} \quad \text{and} \quad \tau = \begin{bmatrix} \text{father} \mapsto \text{parent} \\ \text{mother} \mapsto \text{parent} \end{bmatrix}$$

We put

$$\Xi := \left\{ \begin{array}{l} \langle \tau([\text{child}]), X([\text{child}]) \rangle \\ \langle \tau([\text{father}]), X([\text{father}]) \rangle \\ \langle \tau([\text{mother}]), X([\text{mother}]) \rangle \end{array} \right\} = \left\{ \begin{array}{l} \langle \text{child}, H \rangle \\ \langle \text{parent}, H \rangle \\ \langle \text{parent}, H \rangle \end{array} \right\}$$

This time, Ξ is right-unique and we obtain

$$\mathbf{tr}(X, \tau) = \mathbf{rec}(\Xi) = \begin{bmatrix} \text{child} \mapsto H \\ \text{parent} \mapsto H \end{bmatrix}$$

22.2.5 Example _____

Recall from 9.2.1, that a tuple is a special kind of record. We can use tuples as translators. Let us take $\tau = \langle 3, 2, 1, 4, 4, 6 \rangle$ for example. We generate some τ -translations of records ξ that are tuples as well.

(1) For $\xi = \langle \xi_1, \xi_2, \xi_3 \rangle$ we obtain

$$\begin{aligned} \mathbf{tr}(\xi, \tau) &= \mathbf{rec} \left(\left\{ \begin{array}{l} \langle \tau([1]), \xi_1 \rangle \\ \langle \tau([2]), \xi_2 \rangle \\ \langle \tau([3]), \xi_3 \rangle \end{array} \right\} \right) = \mathbf{rec} \left(\left\{ \begin{array}{l} \langle 3, \xi_1 \rangle \\ \langle 2, \xi_2 \rangle \\ \langle 1, \xi_3 \rangle \end{array} \right\} \right) \\ &= \begin{bmatrix} 1 \mapsto \xi_3 \\ 2 \mapsto \xi_2 \\ 3 \mapsto \xi_1 \end{bmatrix} = \langle \xi_3, \xi_2, \xi_1 \rangle \end{aligned}$$

In other words, for a ternary tuple ξ , the τ -translation is a

reversion of its components.

(2) For $\xi = \langle \xi_1, \xi_2, \xi_3, \xi_4, \xi_5 \rangle$ we obtain

$$\mathbf{tr}(\xi, \tau) = \mathbf{rec} \left(\left\{ \begin{array}{l} \langle 3, \xi_1 \rangle \\ \langle 2, \xi_2 \rangle \\ \langle 1, \xi_3 \rangle \\ \langle 4, \xi_4 \rangle \\ \langle 4, \xi_5 \rangle \end{array} \right\} \right)$$

The right hand of this equation is well-defined only, if $\xi_4 = \xi_5$. If that is indeed the case, we obtain

$$\mathbf{tr}(\xi, \tau) = \langle \xi_3, \xi_2, \xi_1, c, c \rangle \quad \text{with} \quad c := \xi_4 = \xi_5$$

In other words, for tuples with 5 (or more) components, the τ -translation is defined only, if the fourth and fifth component are equal.

(3) For $\xi = \langle a, b, c, d, d, e \rangle$, the constraint $\xi(4) = \xi(5)$ is satisfied. We obtain

$$\mathbf{tr}(\xi, \tau) = \mathbf{rec} \left(\left\{ \begin{array}{l} \langle 3, a \rangle \\ \langle 2, b \rangle \\ \langle 1, c \rangle \\ \langle 4, d \rangle \\ \langle 4, d \rangle \\ \langle 6, e \rangle \end{array} \right\} \right) = \begin{array}{l} 1 \mapsto c \\ 2 \mapsto b \\ 3 \mapsto a \\ 4 \mapsto d \\ 6 \mapsto e \end{array}$$

The result is well-defined, but not a tuple anymore. And that happens to all tuples ξ with more than five components.

22.2.6 Lemma

Let τ and $\xi = [\xi_i | i \in I]$ be two records. If $\xi' := \mathbf{tr}(\xi, \tau)$ is well-defined, then

- (1) $\mathbf{dom}(\xi') = \tau[I]$
- (2) $\mathbf{cod}(\xi') \subseteq \mathbf{cod}(\xi)$

22.2.7 Proof of 22.2.6

(1) follows from definition 22.2.3, 22.2, and 22.2.2. (2) is due to the fact, that the values don't change during translation.

22.3 Translations in $\mathbf{Proj}(\xi)$

22.3.1 Remark

Recall 9.2.4, that for every two classes I and C ,

$$[C|I] := [i \mapsto C | i \in I]$$

is the *univalent* schema with index class I and value C . It maps each $i \in I$ to the same value: the class C .

Accordingly and for every given class I

$$\otimes[I|I] = \{\tau \in \mathbf{Rec} \mid \mathbf{dom}(\tau) \subseteq I, \mathbf{cod}(\tau) \subseteq I\}$$

That is the class of translations that stay inside I .

22.3.2 Remark and example

For future purposes we need an answer to the following question: Given a record $\xi = [\xi_i | i \in I]$. Which $\tau \in \otimes[I|I]$ translates ξ into one of its projections, i.e. $\mathbf{tr}(\xi, \tau) \in \mathbf{Proj}(\xi)$, i.e. $\mathbf{tr}(\xi, \tau) \leq \xi$?

For example, let

$$I = \{a, b, c, p, q, x, y\} \quad \text{and} \quad \xi := \begin{array}{l} a \mapsto 2 \\ b \mapsto 2 \\ c \mapsto 2 \\ p \mapsto 4 \\ q \mapsto 4 \\ x \mapsto 6 \\ y \mapsto 6 \end{array}$$

Three members of $\otimes[I|I]$ are given by

$$\tau_1 := \begin{array}{l} a \mapsto c \\ b \mapsto c \\ c \mapsto c \\ p \mapsto q \end{array} \quad \tau_2 := \begin{array}{l} a \mapsto p \\ b \mapsto p \\ c \mapsto p \\ p \mapsto a \\ q \mapsto a \end{array} \quad \tau_3 := \begin{array}{l} a \mapsto p \\ b \mapsto p \\ c \mapsto p \end{array}$$

The according translations are

$$\mathbf{tr}(\xi, \tau_1) = \mathbf{rec} \left(\left\{ \begin{array}{l} \langle c, 2 \rangle \\ \langle c, 2 \rangle \\ \langle c, 2 \rangle \\ \langle q, 4 \rangle \\ \langle q, 4 \rangle \\ \langle x, 6 \rangle \\ \langle y, 6 \rangle \end{array} \right\} \right) = \begin{array}{l} c \mapsto 2 \\ q \mapsto 4 \\ x \mapsto 6 \\ y \mapsto 6 \end{array} \in \mathbf{Proj}(\xi)$$

$$\mathbf{tr}(\xi, \tau_2) = \mathbf{rec} \left(\left\{ \begin{array}{l} \langle p, 2 \rangle \\ \langle p, 2 \rangle \\ \langle p, 2 \rangle \\ \langle a, 4 \rangle \\ \langle a, 4 \rangle \\ \langle x, 6 \rangle \\ \langle y, 6 \rangle \end{array} \right\} \right) = \begin{array}{l} a \mapsto 4 \\ p \mapsto 2 \\ x \mapsto 6 \\ y \mapsto 6 \end{array} \notin \mathbf{Proj}(\xi)$$

$$\mathbf{tr}(\xi, \tau_3) = \mathbf{rec} \left(\left\{ \begin{array}{l} \langle p, 2 \rangle \\ \langle p, 2 \rangle \\ \langle p, 2 \rangle \\ \langle q, 4 \rangle \\ \langle q, 4 \rangle \\ \langle x, 6 \rangle \\ \langle y, 6 \rangle \end{array} \right\} \right) = \text{undefined}$$

A general answer to the question is the following

22.3.3 Definition

If $\xi = [\xi_i | i \in I]$ is a record and $k \in I$, then

$$[k]_\xi := \{j \in I \mid \xi_j = \xi_k\}$$

the the ξ -equivalence class of k

22.3.4 Lemma

For every record $\xi = [\xi_i | i \in I]$ and $\tau \in \otimes[I|I]$, the following statements are equivalent:

- (1) The τ -translation of ξ is well-defined with $\mathbf{tr}(\xi, \tau) \in \mathbf{Proj}(\xi)$
- (2) $\xi_{\tau(i)} = \xi_i$ for every $i \in \mathbf{dom}(\tau)$
- (3) $\tau \in \otimes [[i]_\xi \mid i \in I]$
- (4) For every $v \in \mathbf{Proj}(\xi)$ the τ -translation of v is well-defined with $\mathbf{tr}(v, \tau) \in \mathbf{Proj}(\xi)$.

22.3.5 Proof of 22.3.4

(1) \Leftrightarrow (2) Suppose, $\xi' := \mathbf{tr}(\xi, \tau)$ is a well-defined member of $\mathbf{Proj}(\xi)$. That means, $\xi' = [\xi_j | j \in J]$ for some $J \subseteq I$. For each $i \in I$, $\xi'_{\tau(i)}$ is defined as ξ_i , so $\xi_{\tau(i)} = \xi_i$. In case

$i \notin \text{dom}(\tau)$, $\xi'_{\tau([i])} = \xi'_i = \xi_i$ anyway. So $\text{tr}(\xi, \tau) \in \text{Proj}(\xi)$ iff $\xi_{\tau([i])} = \xi_i$ for all $i \in \text{dom}(\tau)$.

(2) \Leftrightarrow (3) Due to the definition of $[i]_\xi$, we have $\xi_{\tau([i])} = \xi_i$ iff $\tau([i]) \in [i]_\xi$. Thus $\xi_{\tau([i])} = \xi_i$ for all $i \in I$ iff $\tau \in \otimes [[i]_\xi \mid i \in I]$.

(1) \Leftrightarrow (4) $v \in \text{Proj}(\xi)$ is saying that $v \leq \xi$. An immediate consequence of definition 22.2.3 is the fact that the existence of $\text{tr}(\xi, \tau)$ and $v \leq \xi$ implies the existence of $\text{tr}(v, \tau)$ with $\text{tr}(v, \tau) \leq \text{tr}(\xi, \tau)$. So (1) implies (4). And with $\xi \leq \xi$, the other direction holds as well.

$$\tau := \begin{bmatrix} \text{child} \mapsto \text{kind} \\ \text{father} \mapsto \text{vater} \\ \text{mother} \mapsto \text{mutter} \end{bmatrix}$$

The τ -translation of the schema exists

$$\text{tr} \left(\begin{bmatrix} \text{child} \mapsto H \\ \text{father} \mapsto H \\ \text{mother} \mapsto H \end{bmatrix}, \tau \right) = \begin{bmatrix} \text{kind} \mapsto H \\ \text{vater} \mapsto H \\ \text{mutter} \mapsto H \end{bmatrix}$$

and for each

$$\begin{bmatrix} \text{child} \mapsto c_i \\ \text{father} \mapsto f_i \\ \text{mother} \mapsto m_i \end{bmatrix} \in R$$

we have

$$\text{tr} \left(\begin{bmatrix} \text{child} \mapsto c_i \\ \text{father} \mapsto f_i \\ \text{mother} \mapsto m_i \end{bmatrix}, \tau \right) = \begin{bmatrix} \text{kind} \mapsto c_i \\ \text{vater} \mapsto f_i \\ \text{mutter} \mapsto m_i \end{bmatrix}$$

so that

$$\text{tr}(R, \tau) = \left\{ \begin{array}{l} \begin{bmatrix} \text{kind} \mapsto H \\ \text{vater} \mapsto H \\ \text{mutter} \mapsto H \end{bmatrix} \\ \left\{ \begin{array}{l} \begin{bmatrix} \text{kind} \mapsto c_1 \\ \text{vater} \mapsto f_1 \\ \text{mutter} \mapsto m_1 \end{bmatrix}, \begin{bmatrix} \text{kind} \mapsto c_2 \\ \text{vater} \mapsto f_2 \\ \text{mutter} \mapsto m_2 \end{bmatrix} \\ \begin{bmatrix} \text{kind} \mapsto c_3 \\ \text{vater} \mapsto f_3 \\ \text{mutter} \mapsto m_3 \end{bmatrix} \end{array} \right\} \end{array} \right\}$$

kind : H	vater : H	mutter : H
c_1	f_1	m_1
c_2	f_2	m_2
c_3	f_3	m_3

as expected.

22.3.6 Example

Applied to the previous example ξ from ??: The class of all $\tau \in \otimes [I|I]$ such that $\text{tr}(\xi, \tau)$ is a (well-defined) element of $\text{Proj}(\xi)$ is given by

$$\otimes [[i]_\xi \mid i \in I] = \otimes \begin{bmatrix} a \mapsto \{a, b, c\} \\ b \mapsto \{a, b, c\} \\ c \mapsto \{a, b, c\} \\ p \mapsto \{p, q\} \\ q \mapsto \{p, q\} \\ x \mapsto \{x, y\} \\ y \mapsto \{x, y\} \end{bmatrix}$$

22.4 Translation of relations

22.4.1 Definition (attribute) translation of relations

Let $R = [X, \Gamma]$ be a relation and τ a record. If $\text{tr}(X, \tau)$ is well-defined, we say that R is τ -translatable and we define

$$\text{tr}(R, \tau) := \begin{bmatrix} \text{tr}(X, \tau) \\ \{\text{tr}(x, \tau) \mid x \in \Gamma \text{ is } \tau\text{-translatable}\} \end{bmatrix}$$

the (attribute) translation of R by τ .

22.4.2 Example

Consider example 22.1.1 again with

$$R := \begin{array}{|c|c|c|} \hline \text{child} : H & \text{father} : H & \text{mother} : H \\ \hline c_1 & f_1 & m_1 \\ \hline c_2 & f_2 & m_2 \\ \hline c_3 & f_3 & m_3 \\ \hline \end{array}$$

$$= \left\{ \begin{array}{l} \begin{bmatrix} \text{child} \mapsto H \\ \text{father} \mapsto H \\ \text{mother} \mapsto H \end{bmatrix} \\ \left\{ \begin{array}{l} \begin{bmatrix} \text{child} \mapsto c_1 \\ \text{father} \mapsto f_1 \\ \text{mother} \mapsto m_1 \end{bmatrix}, \begin{bmatrix} \text{child} \mapsto c_2 \\ \text{father} \mapsto f_2 \\ \text{mother} \mapsto m_2 \end{bmatrix} \\ \begin{bmatrix} \text{child} \mapsto c_3 \\ \text{father} \mapsto f_3 \\ \text{mother} \mapsto m_3 \end{bmatrix} \end{array} \right\} \end{array} \right\}$$

and

22.4.3 Example

Consider the common linear order relation on the set of integers $\leq : \mathbb{Z} \leftrightarrow \mathbb{Z}$. In example 22.2.5 we already used tuples as translators for tuples. Let us now take $\tau := \langle 2, 1 \rangle$ as a translator for R .

In general, the τ -translation of any pair $\xi = \langle \xi_1, \xi_2 \rangle$ is

$$\begin{aligned} \text{tr}(\xi, \tau) &= \text{rec} \left(\left\{ \left\{ \langle \langle 2, 1 \rangle ([1]), \langle \xi_1, \xi_2 \rangle ([1]) \rangle \right\} \right\} \right) \\ &= \text{rec} \left(\left\{ \langle \langle 2, \xi_1 \rangle \rangle \right\} \right) = \begin{bmatrix} 2 \mapsto \xi_1 \\ 1 \mapsto \xi_2 \end{bmatrix} = \langle \xi_2, \xi_1 \rangle \end{aligned}$$

We can use this result now first for the τ -translation of the schema $\langle \mathbb{Z}, \mathbb{Z} \rangle$ of R and obtain

$$\text{tr}(\langle \mathbb{Z}, \mathbb{Z} \rangle, \tau) = \langle \mathbb{Z}, \mathbb{Z} \rangle$$

This is a well-defined tuple, so R is τ -translatable with

$$\text{tr}(\leq, \tau) = \begin{bmatrix} \langle \mathbb{Z}, \mathbb{Z} \rangle \\ \{\langle b, a \rangle \mid a \leq b\} \end{bmatrix}$$

and that is \geq , the greater-or-equal relation on \mathbb{Z} .

22.4.4 Remark ——— translations in mathematics

The example in 22.1.1 suggests that (attribute) translations are important operations in relational database applications. But they are also very useful in traditional mathematics. They provide the means to assign meanings to formulas of predicate logic.

A typical situation is a given expression like “ $R(x, y)$ ” together with an interpretation \mathcal{I} , that turns the relation symbol R into a proper binary endorelation $R^{\mathcal{I}}$ on a certain class $C^{\mathcal{I}}$. By means of attribute translations, this expression “ $R(x, y)$ ” has an own proper meaning: $\mathbf{tr}(R^{\mathcal{I}}, \langle x, y \rangle)$. In more detail:

- ♣ $R : C \rightsquigarrow C$ is the given signature, it says that R is the symbol of a binary endorelation, defined on on a class symbolized by C .
- ♣ $R^{\mathcal{I}} : C^{\mathcal{I}} \rightsquigarrow C^{\mathcal{I}}$ is a concrete binary endorelation on $C^{\mathcal{I}}$, given by say

$$R^{\mathcal{I}} = \left[\begin{array}{c} C^{\mathcal{I}} \rightsquigarrow C^{\mathcal{I}} \\ \langle a, b \rangle \rightsquigarrow \varphi(a, b) \end{array} \right]$$

- ♣ If the schema translation $\mathbf{tr}(\langle C^{\mathcal{I}}, C^{\mathcal{I}} \rangle, \langle x, y \rangle)$ is well-defined then

$$\mathbf{tr}(R^{\mathcal{I}}, \langle x, y \rangle) = \left[\begin{array}{c} C^{\mathcal{I}} \rightsquigarrow C^{\mathcal{I}} \\ \left[\begin{array}{c} x \mapsto a \\ y \mapsto b \end{array} \right] \rightsquigarrow \langle a, b \rangle \in R^{\mathcal{I}} \end{array} \right]$$

In traditional predicate logic, interpretations assign meanings to formulas only, if the formulas are closed. And then, their meaning is either true (when the interpretation is a model for the formula) or false. Atomic formulas like $R(x, y)$ are not closed, x and y are free variables. But with attribute translations of this kind, we provide a semantics for $R(x, y)$. And together with the other operations $\sqcap, \sqcup, \neg, \dots$, we now have a denotational semantics for all formulas, not only the closed ones.

22.5 Translations in $\mathbf{Prel}(X)$

22.5.1 Remark ——— introduction

We saw, that a translation of a given relation may or may not exist. But sometimes it is important to know in advance if the outcome is well-defined. When we consider a structure or algebra based on a carrier class \mathcal{R} of relations, we want to know the class \mathcal{T} of all the suitable relations for \mathcal{R} . “Suitable” means, that \mathcal{R} is the “ \mathcal{T} -closure” of \mathcal{R} in the sense that, for every $\tau \in \mathcal{T}$ and $R \in \mathcal{R}$, the τ -translation of R is well-defined and again a member of \mathcal{R} .

There is one relation class \mathcal{R} which is important here, namely the class $\mathbf{Prel}(X)$, for a given schema X . We use $\mathbf{Trans}(X)$ to denote the “suitable” translation class for $\mathbf{Prel}(X)$. Given $X = [X_i | i \in I]$, it makes sense to only include translations τ

with arguments and values from I , i.e. $\tau \in \otimes[I|I]$.

22.5.2 Definition ——— translator classes

Let $X = [X_i | i \in I]$ be a proper schema. We define

$$\mathbf{Trans}(X) := \left\{ \tau \in \otimes[I|I] \mid \begin{array}{l} \mathbf{tr}(R, \tau) \in \mathbf{Prel}(X) \\ \text{for each } R \in \mathbf{Prel}(X) \end{array} \right\}$$

the translator class on X

22.5.3 Remark

Note, that in definition 22.5.2 the characterization “ $\mathbf{tr}(R, \tau) \in \mathbf{Prel}(X)$ for each $R \in \mathbf{Prel}(X)$ ” implies, that $\mathbf{tr}(R, \tau)$ is actually well-defined for all R .

22.5.4 Lemma

If $X = [X_i | i \in I]$ is a proper schema, then

$$\mathbf{Trans}(X) = \otimes [[i]_X \mid i \in I]$$

22.5.5 Proof of 22.5.4

So let $X = [X_i | i \in I]$ be given.

Again (see 22.3.3), $[i]_X := \{j \in I \mid X_j = X_i\}$ is the X -equivalence class of i , for each $i \in I$. $\mathbf{Prel}(X)$ is (see 17.8.1) defined to be the class of all relations with a schema $Y \in \mathbf{Proj}(X)$. According to 22.4.1, $\mathbf{tr}(R, \tau)$ is defined iff $\mathbf{tr}(Y, \tau)$, the τ -translation of its schema, is defined. So

$$\mathbf{Trans}(X) = \left\{ \tau \in \otimes[I|I] \mid \begin{array}{l} \mathbf{tr}(Y, \tau) \in \mathbf{Proj}(X) \\ \text{for each } Y \in \mathbf{Proj}(X) \end{array} \right\}$$

Due to 22.3.4, the right hand side of this equation is equal to $\otimes [[i]_X \mid i \in I]$.

22.5.6 Lemma

For every proper schema $X = [X_i | i \in I]$ the following statements are equivalent:

- (1) X is univalent (i.e. $X_i = X_j$ for all $i, j \in I$)
- (2) $\mathbf{Trans}(X) = \otimes[I|I]$

22.5.7 Proof of 22.5.6

$$\mathbf{Trans}(X) = \otimes[I|I]$$

$$\Leftrightarrow \mathbf{tr}(R, \tau) \in \mathbf{Prel}(X) \text{ for all } R \in \mathbf{Prel}(X) \text{ and } \tau \in \otimes[I|I]$$

$$\Leftrightarrow \mathbf{tr}(Y, \tau) \in \mathbf{Proj}(X) \text{ for all } Y \in \mathbf{Proj}(X) \text{ and } \tau \in \otimes[I|I]$$

$$\Leftrightarrow X_{\tau(i)} = X_i \text{ for each } i \in I \text{ and } \tau \in \otimes[I|I]$$

due to 22.3.4

$$\Leftrightarrow X_j = X_k \text{ for all } j, k \in I$$

Part IX

Theory algebras of relations

23 Theory algebras on $\mathbf{Prel}(X)$

23.0.8 [introduction and overview](#)

In this section we concentrate on the effect of the two relations \sqsubseteq and \trianglelefteq on a given relation class $\mathbf{Prel}(X)$. We already established that each one is a quasi-order relation (19.1.6 and 20.4.12). In fact, and that will be the theme here, each one turns $\mathbf{Prel}(X)$ into a *complete quasi-boolean lattice*.

Recall (see chapter III) the difference between a *quasi-boolean lattice* in general and a *boolean lattice* in particular: for a real *boolean lattice*, the typical boolean operations (the join of two elements, the complement, the bottom element etc.) are unique, a boolean lattice is a boolean algebra. For a *quasi-boolean lattice* however, these operations do exist (that is the very meaning of the term), but they are not necessarily unique. Take the example (see 6.4) of the formulas $\mathbf{Form}(\mathbb{A})$ on some \mathbb{A} , together with the quasi-boolean order \Rightarrow . There is a bottom element $\mathbf{0}$, but there are different other bottom elements as well: $[a \wedge \neg a]$, $[\mathbf{0} \wedge \mathbf{1}]$, etc.

The structures imposed by \sqsubseteq and \trianglelefteq on $\mathbf{Prel}(X)$ are indeed “quasi” in this sense, at least if X is not trivial. We will prove that they are quasi-boolean lattices, but we also want two quasi-boolean algebras with an actual and well-defined set of boolean operators to operate with. The selection of various operators we defined so far does provide all we need for the structure on \sqsubseteq : we have a meet \sqcap and join \sqcup , a negator \neg and two extremes \perp and \top .

For the structure on \trianglelefteq , the situation is more complicated. We have a couple of possibilities available to operate on the relation schemas: $\parallel, \Downarrow, \Uparrow, \uparrow, \text{pr}, \text{cpj}$ and some others. But none of them is defined on relations only, they all have two different types of arguments. For example, \Downarrow is defined on $\textcircled{1} \in \mathbf{Prel}(X)$ and $\textcircled{2} \in \mathbf{Proj}(Z)$. For our search for proper boolean operations we can either modify the domain of the available operations, which is quite easy: e.g. for $R, S \in \mathbf{Prel}(X)$ we could declare $R \Downarrow S := R \Downarrow \mathbf{x}(S)$. Or we could create an entire new set of operations. In the sequel, we will do both. And in the end, we have a couple of quasi-boolean algebras defined on $\langle \mathbf{Prel}(X), \trianglelefteq \rangle$.³⁵

The whole idea of a theory algebra as a double quasi-ordered structure is the core idea of this whole text. We have added a section on the theory algebra of bit value relations, where we take this simple and important relations and discuss and display their algebra in detail. In these examples, the important properties of all the more abstract cases are reflected already and a contemplation on the order diagrams displayed there may help to grasp the whole idea.

The structure on \trianglelefteq is called *syntactic*, the structure on \sqsubseteq is called *semantic*, the specific combination of the two is the *theory structure* (see 3.4.1 for the motivation of this terminology).

23.1 The semantic structures on $\mathbf{Prel}(X)$

23.1.1 Definition

If X is a proper schema, then

$$\mathfrak{Prel}^{\sqsubseteq}(X) := \langle \mathbf{Prel}(X), \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \prod, \coprod, \neg \rangle$$

is the semantic algebra (on $\mathbf{Prel}(X)$) (of X)

23.1.2 Lemma

$\mathfrak{Prel}^{\sqsubseteq}(X)$ is a complete quasi-boolean algebra, for every proper schema X .

23.1.3 Proof of 23.1.2

Complete quasi-boolean algebras (see definition 7.6.2(5)), and $\mathfrak{Prel}^{\sqsubseteq}(X)$ in particular, must have the following properties:

- (i) $\langle \mathbf{Prel}(X), \sqsubseteq, \equiv \rangle$ is a quasi-ordered class.
That is true according to 20.4.12.

And in this quasi-ordered structure $\langle \mathbf{Prel}(X), \sqsubseteq, \equiv \rangle$,

- (ii) \perp must be a least element and \top a greatest element.
That is true according to 20.4.21(17) and (18).
- (iii) \prod is a big conjunctor and \coprod is a big disjunctive.
In other words, $\prod \mathcal{R}$ is a greatest lower and $\coprod \mathcal{R}$ a least upper bound, for every $\mathcal{R} \subseteq \mathbf{Prel}(X)$. And that is true according to 20.4.19(3) and (4).
- (iv) \sqcap is a meet and \sqcup is a join operator.
In other words, $R \sqcap S$ is a greatest lower and $R \sqcup S$ is a least upper bound of R and S , for all $R, S \in \mathbf{Prel}(X)$. And that means, that $R \sqcap S \equiv \prod \{R, S\}$ and $R \sqcup S \equiv \coprod \{R, S\}$, which is true according to 20.4.21(9) and (10).
- (v) The structure is distributive.
 $R \sqcap (S \sqcup T) \equiv (R \sqcap S) \sqcup (R \sqcap T)$ is true, due to 20.4.21(11),
 $R \sqcup (S \sqcap T) \equiv (R \sqcup S) \sqcap (R \sqcup T)$ is true, due to 20.4.21(12), for all $R, S, T \in \mathbf{Prel}(X)$.
- (vi) \neg is a complement function (7.5.2).
 $R \sqcap \neg R \equiv \perp$ and $R \sqcup \neg R \equiv \top$ is true for all $R \in \mathbf{Prel}(X)$, according to 20.4.21(21) and (22).

23.1.4 Lemma

$\langle \mathbf{Prel}(X), \sqsubseteq \rangle$ is a complete quasi-boolean lattice, for every proper schema X .

23.1.5 Proof of 23.1.4

This is an immediate consequence of 23.1.2, according to definition 7.6.2(5).

³⁵The practical applications for theory algebras will probably rather use the original mixed domain operators. For example, in logic, the quantor $\forall \textcircled{1} . \textcircled{2}$ is very much the infimum eliminator $\textcircled{2} \uparrow \textcircled{1}$. But from a more theoretical point of view, it is much more elegant to just have one sort of arguments only.

23.2 Selective syntactic structures on $\mathbf{Prel}(X)$

23.2.1 Definition

Let X be a proper schema. A syntactic selector on X is a function $\varsigma : \mathbf{Proj}(X) \rightarrow \mathbf{Prel}(X)$ such that $\mathbf{x}(\varsigma(Y)) = Y$, for all $Y \in \mathbf{Proj}(X)$.

23.2.2 Example

We already have defined some syntactic selectors: by restricting \perp_{\oplus} and \top_{\oplus} from 19.3.1 and \mathbf{Id}_{\oplus} from 19.2.1 to $\mathbf{Prel}(X)$ for a given proper schema X we obtain three syntactic selectors on X :

$$(1) \perp_{\oplus} = \left[\begin{array}{c} \mathbf{Proj}(X) \longrightarrow \mathbf{Prel}(X) \\ Y \mapsto \left[\begin{array}{c} Y \\ \emptyset \end{array} \right] \end{array} \right]$$

$$(2) \top_{\oplus} = \left[\begin{array}{c} \mathbf{Proj}(X) \longrightarrow \mathbf{Prel}(X) \\ Y \mapsto \left[\begin{array}{c} Y \\ \otimes Y \end{array} \right] \end{array} \right]$$

$$(3) \mathbf{Id}_{\oplus} = \left[\begin{array}{c} \mathbf{Proj}(X) \longrightarrow \mathbf{Prel}(X) \\ Y \mapsto \left[\begin{array}{c} Y \\ v \rightsquigarrow \forall i, j \in \mathbf{dom}(Y) . v_j = v_k \end{array} \right] \end{array} \right]$$

23.2.3 Definition

Let X be a proper schema and ς a syntactic selector on X . We define

$$\mathfrak{Prel}_{\varsigma}^{\triangleleft}(X) := \left\langle \mathbf{Prel}(X), \triangleleft, \triangle, \emptyset_{\varsigma}, \mathbf{1}_{\varsigma}, \wedge_{\varsigma}, \vee_{\varsigma}, \bigwedge_{\varsigma}, \bigvee_{\varsigma}, \neg_{\varsigma} \right\rangle$$

the selective syntactic structure of ς over X

where

$$\emptyset_{\varsigma} := \varsigma(\langle \rangle) \quad \mathbf{1}_{\varsigma} := \varsigma(X)$$

$$\textcircled{1} \wedge_{\varsigma} \textcircled{2} := \left[\begin{array}{c} \mathbf{Prel}(X) \times \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ \langle R, S \rangle \mapsto \varsigma(\mathbf{x}(R) \wedge \mathbf{x}(S)) \end{array} \right]$$

$$\textcircled{1} \vee_{\varsigma} \textcircled{2} := \left[\begin{array}{c} \mathbf{Prel}(X) \times \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ \langle R, S \rangle \mapsto \varsigma(\mathbf{x}(R) \vee \mathbf{x}(S)) \end{array} \right]$$

$$\bigwedge_{\varsigma} := \left[\begin{array}{c} \mathbf{P}(\mathbf{Prel}(X)) \longrightarrow \mathbf{Prel}(X) \\ \mathcal{R} \mapsto \begin{cases} \varsigma\left(\bigwedge_{R \in \mathcal{R}} \mathbf{x}(R)\right) & \text{if } \mathcal{R} \neq \emptyset \\ \varsigma(X) & \text{if } \mathcal{R} = \emptyset \end{cases} \end{array} \right]$$

$$\bigvee_{\varsigma} := \left[\begin{array}{c} \mathbf{P}(\mathbf{Prel}(X)) \longrightarrow \mathbf{Prel}(X) \\ \mathcal{R} \mapsto \varsigma\left(\bigvee_{R \in \mathcal{R}} \mathbf{x}(R)\right) \end{array} \right]$$

$$\neg_{\varsigma} := \left[\begin{array}{c} \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ R \mapsto \varsigma(X \setminus \mathbf{x}(R)) \end{array} \right]$$

23.2.4 Lemma

$\mathfrak{Prel}_{\varsigma}^{\triangleleft}(X)$ is a complete quasi-boolean algebra, for every proper schema X and syntactic selector ς on X .

23.2.5 Proof of 23.2.4

Let $X = [X_i | i \in I]$ be the given proper schema. Recall, that for all $R, S \in \mathbf{Prel}(X)$,

$$R \triangleleft S \quad \text{iff} \quad \mathbf{x}(R) \leq \mathbf{x}(S) \quad \text{iff} \quad @ (R) \subseteq @ (S)$$

We proceed with the same steps as in the proof of 23.1.2.

- (i) $\langle \mathbf{Prel}(X), \triangleleft, \triangle \rangle$ is a quasi-ordered class.
That is true, due to 19.1.6.

And in this quasi-ordered class we have:

- (ii) \emptyset_{ς} is a least and $\mathbf{1}_{\varsigma}$ is a top element.
For every $R \in \mathbf{Prel}(X)$ holds $\emptyset_{\varsigma} \triangleleft R$ iff $@(\emptyset_{\varsigma}) \subseteq @(R)$ iff $\emptyset \subseteq @(R)$, and that is always true. On the other hand, $R \triangleleft \mathbf{1}_{\varsigma}$ iff $\mathbf{x}(R) \leq X$, which is true because $R \in \mathbf{Prel}(X)$.
- (iii) \bigwedge_{ς} is a big conjunctor and \bigvee_{ς} is a big disjunctor.
We proof this for \bigwedge_{ς} , a proof for \bigvee_{ς} is similar. \bigwedge_{ς} is a big conjunctor iff $\bigwedge_{\varsigma} \mathcal{R}$ is the greatest lower bound (g.l.b.) of \mathcal{R} in $\langle \mathbf{Prel}(X), \triangleleft \rangle$, for every $\mathcal{R} \subseteq \mathbf{Prel}(X)$. That is the case exactly when $\mathbf{x}(\bigwedge_{\varsigma} \mathcal{R})$ is the g.l.b. of $\{\mathbf{x}(R) | R \in \mathcal{R}\}$ in $\langle \mathbf{Proj}(X), \leq \rangle$, i.e. iff $\bigwedge \{\mathbf{x}(R) | R \in \mathcal{R}\}$ is the g.l.b. of $\{\mathbf{x}(R) | R \in \mathcal{R}\}$ in $\langle \mathbf{Proj}(X), \leq \rangle$. Which is true indeed, for every \mathcal{R} (including $\mathcal{R} = \emptyset$ in particular).

- (iv) $R \wedge_{\varsigma} S \triangleq \bigwedge_{\varsigma} \{R, S\}$ and $R \vee_{\varsigma} S \triangleq \bigvee_{\varsigma} \{R, S\}$.
For $R, S \in \mathbf{Prel}(X)$ we have $R \wedge_{\varsigma} S = \varsigma(\mathbf{x}(R) \wedge \mathbf{x}(S)) =$

$\varsigma(\bigwedge \{\mathbf{x}(R), \mathbf{x}(S)\}) = \bigwedge_{\varsigma} \{R, S\}$. And similarly, $R \vee_{\varsigma} S \equiv \bigvee_{\varsigma} \{R, S\}$.

(v) The distributive laws are satisfied.

If $R, S, T \in \mathbf{Prel}(X)$ with $A = @ (R)$, $B = @ (S)$, $C = @ (T)$, then

$$R \wedge_{\varsigma} (S \vee_{\varsigma} T) \triangleq (R \wedge_{\varsigma} S) \vee_{\varsigma} (R \wedge_{\varsigma} T)$$

$$\text{iff } A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

and the latter is a general truth for all classes A, B, C . Similarly, we can see that $R \vee_{\varsigma} (S \wedge_{\varsigma} T) \triangleq (R \vee_{\varsigma} S) \wedge_{\varsigma} (R \vee_{\varsigma} T)$.

(vi) $-\varsigma$ is a complement function (7.5.2).

If $R = [Y, \Gamma] \in \mathbf{Prel}(X)$, then $-\varsigma R = \varsigma(X \setminus Y)$. We have $R \wedge_{\varsigma} (-\varsigma R) = \varsigma(Y \wedge (X \setminus Y)) = \emptyset_{\varsigma}$ and $R \vee_{\varsigma} (-\varsigma R) = \varsigma(Y \vee (X \setminus Y)) = \mathbf{1}_{\varsigma}$. Thus $-\varsigma$ is indeed a complement function.

23.2.6 Lemma

$\langle \mathbf{Prel}(X), \trianglelefteq, \trianglelefteq \rangle$ is a complete quasi-boolean lattice, for every proper schema X .

23.2.7 Proof of 23.2.6

Immediate consequence of 23.2.4 by means of 7.6.2(5).

23.3 The 3-carrier syntactic structures on $\mathbf{Prel}(X)$

23.3.1 Repetition

(1) Recall 6.2.2, that the power structure of a given class I

$$\mathfrak{P}(I) = \langle \mathbf{P}(I), \subseteq, \emptyset, \mathbf{1}, \cap, \cup, \bigcap, \bigcup, \mathcal{C} \rangle$$

is the usual complete boolean algebra with

$$\mathbf{1} := I \quad \text{and} \quad \mathcal{C} \oplus := \left[\begin{array}{c} \mathbf{P}(I) \longrightarrow \mathbf{P}(I) \\ A \mapsto C \setminus A \end{array} \right]$$

(2) Recall 11.9.1, that

$$\mathfrak{Proj}(X) = \langle \mathbf{Proj}(X), \leq, \langle \rangle, X, \wedge, \vee, \bigwedge, \bigvee, - \rangle$$

is the complete boolean algebra of projections of a record or schema X , where

$$\bigwedge \emptyset := X \quad \text{and} \quad - \oplus := \left[\begin{array}{c} \mathbf{Proj}(X) \longrightarrow \mathbf{Proj}(X) \\ Y \mapsto X \setminus Y \end{array} \right]$$

(3) Recall 5.8.6, that $\mathfrak{S} \uparrow \mathfrak{S}'$ is the *combination* of two structures \mathfrak{S} and \mathfrak{S}' . The result basically is the union of the carrier classes and the operations.

23.3.2 Definition

For every proper schema $X = [X_i | i \in I]$ we define

$$\mathfrak{Prel}_3^{\trianglelefteq}(X) := \left\langle \mathbf{Prel}(X), \mathbf{Proj}(X), \mathbf{P}(I), \trianglelefteq, \trianglelefteq, \perp_{\oplus}, \top_{\oplus}, @, \mathbf{x}, \parallel, \downarrow, \uparrow, \downarrow, \uparrow \right\rangle$$

the 3-carrier syntactic structure of X

$$\mathfrak{Prel}_3^{\trianglelefteq}(X) := \mathfrak{Prel}_3^{\trianglelefteq}(X) \uparrow \mathfrak{Proj}(X) \uparrow \mathfrak{P}(I)$$

the 3-structure syntactic structure of X

23.3.3 Remark the 3-carrier syntactic structure

The 3-carrier syntactic structure of a given proper schema $X = [X_i | i \in I]$ is the structure made of 3 carrier classes: relations $\mathbf{Prel}(X)$, schemas $\mathbf{Proj}(X)$ and attribute (or identifier) classes $\mathbf{P}(I)$. And a couple of operations, all defined in earlier chapters. Next to \trianglelefteq and \trianglelefteq , there are:

A couple of functions that serve as converters between the three carrier classes:

- $\perp_{\oplus}, \top_{\oplus} : \mathbf{Proj}(X) \longrightarrow \mathbf{Prel}(X)$ the *empty* and *full relation* function (19.3.1) with $\perp_{\langle \rangle} = \perp$ and $\top_{\langle \rangle} = \top$,
- $@ : \mathbf{Prel}(X) \longrightarrow \mathbf{P}(I)$ the *attribute class* function (??)
- $\mathbf{x} : \mathbf{Prel}(X) \longrightarrow \mathbf{Proj}(X)$ the *schema* function (??)

And operations to modify the schema of relations:

- $\parallel, \downarrow, \uparrow : \mathbf{Prel}(X) \times \mathbf{Proj}(X) \longrightarrow \mathbf{Prel}(X)$ the *expander* and the *supremum* and *infimum reductor* (20.1.1 and 21.6.1)
- $\downarrow, \uparrow : \mathbf{Prel}(X) \times \mathbf{P}(I) \longrightarrow \mathbf{Prel}(X)$ the *supremum* and *infimum eliminators* (21.4.1)

All these functions are well-defined on their entire domains, because both $\mathbf{Prel}(X)$ and $\mathbf{Proj}(X)$ are (pairwise) compatible.

23.3.4 Remark more or less operations

The selection of operations in $\mathfrak{Prel}_3^{\trianglelefteq}$ is somewhat arbitrary. Earlier on we defined a couple of other operations that could equally well added to the definition of this structure. Most of these additional operations are definable in terms of the chosen ones.

Additional schema comparing relations would have been:

- $\checkmark : \mathbf{Prel}(X) \rightsquigarrow \mathbf{Prel}(X)$ the *distinctness* relation (19.1.2(3)), which can be derived by $R \checkmark S \Leftrightarrow (@R \cap @S = \emptyset)$
- $\smile : \mathbf{Prel}(X) \rightsquigarrow \mathbf{Prel}(X)$ the *compatibility* relation; which is superfluous in this context, because $R \smile S$ holds anyway for all $R, S \in \mathbf{Prel}(X)$ (19.1.10(4)).

More carrier class conversions are:

- $\text{Id}_{\oplus} : \mathbf{Proj}(X) \longrightarrow \mathbf{Prel}(X)$ the *identity relation* constructor (19.2.1). This is indeed a converter between two of the carrier classes in $\mathfrak{Prel}_3^{\trianglelefteq}$, one that cannot be derived from it in general and adds a whole new dimension. But we exclude it here for several reasons and treat it as an extra additional operation.
- $\text{dom} : \mathbf{Proj}(X) \longrightarrow \mathbf{P}(I)$ is also a converter between two of the carrier classes. But neither its source nor its target is the relation class. And besides, it can be derived by $\text{dom}(Y) = @(\perp_Y) = @(\top_Y)$.

More relation schema modifiers are:

- $\text{pr}, \text{cpj} : \mathbf{Prel}(X) \times \mathbf{P}(I) \dashrightarrow \mathbf{Prel}(X)$ are only defined as partial functions in 21.2.2. And for their defined arguments, they can be derived in $\mathfrak{Prel}_3^{\trianglelefteq}(X)$ by

$$\text{pr}(R, J) = R \downarrow (@(R) \setminus J) \quad \text{cpj}(R, J) = R \uparrow (@(R) \setminus J)$$

- $\updownarrow : \mathbf{Prel}(X) \times \mathbf{Proj}(X) \dashrightarrow \mathbf{Prel}(X)$ the *equivalent reductor* (21.6.1) is only a partial function. And the restriction to its defined arguments coincides with both \downarrow and \uparrow , i.e. if $R \updownarrow Y$ is defined, then $R \updownarrow Y = R \uparrow Y = R \downarrow Y$, according to 21.6.6(3).

23.3.5 Remark the 3-structure syntactic structure

The 3-structure syntactic structure is fully given by

$$\mathfrak{Prel}_3^{\triangleleft}(X) = \left\langle \begin{array}{l} \mathbf{Prel}(X), \mathbf{Proj}(X), \mathbf{P}(I), \\ \triangleleft, \underline{\triangle}, \perp_{\oplus}, \top_{\oplus}, @, \mathbf{x}, \parallel, \Downarrow, \Uparrow, \downarrow, \uparrow, \\ \leq, \langle \rangle, X, \wedge, \vee, \hat{\wedge}, \hat{\vee}, \hat{\neg}, \hat{0}, \hat{1}, \cap, \cup, \bigcap, \bigcup, \mathcal{C} \end{array} \right\rangle$$

The addition of the operations of $\mathfrak{Proj}(X)$ and $\mathfrak{P}(I)$ enables us to simulate the full set of boolean operations on $\langle \mathbf{Prel}(X), \triangleleft \rangle$.

More precisely, we e.g. obtain a complete quasi-boolean algebra if we fix some constant $T \in \{\perp, \top\}$ and define

$$\mathfrak{Prel}_T^{\triangleleft}(X) := \left\langle \mathbf{Prel}(X), \triangleleft, \underline{\triangle}, \hat{0}, \hat{1}, \hat{\wedge}, \hat{\vee}, \hat{\neg}, \hat{\Lambda}, \hat{\Sigma} \right\rangle$$

and define

$$\begin{aligned} \hat{0} &:= T \\ \hat{1} &:= T \parallel X \\ R \hat{\wedge} S &:= T \parallel (\mathbf{x}(R) \wedge \mathbf{x}(S)) \\ R \hat{\vee} S &:= T \parallel (\mathbf{x}(R) \vee \mathbf{x}(S)) \\ \hat{\Lambda} \mathcal{R} &:= T \parallel \bigwedge_{R \in \mathcal{R}} \mathbf{x}(R) \\ \hat{\Sigma} \mathcal{R} &:= T \parallel \bigwedge_{R \in \mathcal{R}} \mathbf{x}(R) \\ \hat{\neg} R &:= T \parallel \neg \mathbf{x}(R) \end{aligned}$$

for all $R, S \in \mathbf{Prel}(X)$ and $\mathcal{R} \subseteq \mathbf{Prel}(X)$.

In fact, it is an easy exercise to confirm that

$$\mathfrak{Prel}_T^{\triangleleft}(X) = \mathfrak{Prel}_{\varsigma}^{\triangleleft}(X) \quad \text{with} \quad \varsigma = \begin{cases} \perp_{\oplus} & \text{if } T = \perp \\ \top_{\oplus} & \text{if } T = \top \end{cases}$$

23.4 The 2-carrier syntactic structures on $\mathbf{Prel}(X)$

23.4.1 Remark

The 3-structure syntactic structure $\mathfrak{Prel}_3^{\triangleleft}(X)$ is somehow redundant, because the two structure $\mathfrak{Proj}(X)$ and $\mathfrak{P}(I)$ are isomorph (11.9.3). Since X is fixed in the context of this structure, we could represent each $Y = [Y_j | j \in J] \in \mathbf{Proj}(X)$ just by its domain $J \in \mathbf{P}(I)$, a reconstruction of Y is always given by $Y = \mathbf{pr}(X, J)$. This way, we could replace the whole structure $\mathfrak{Proj}(X)$ by its isomorph counterpart $\mathfrak{P}(I)$. We just need to replace the operations like $\Downarrow: \mathbf{Prel}(X) \times \mathbf{Proj}(X) \rightarrow \mathbf{Prel}(X)$ by a its according mutation $\Downarrow: \mathbf{Prel}(X) \times \mathbf{P}(I) \rightarrow \mathbf{Prel}(X)$.

We use the two dots on \Downarrow to explicitly distinguish this “2-carrier version” from the original \Downarrow . But in practice it should be save to write \Downarrow again.

23.4.2 Definition

Let $X = [X_i | i \in I]$ be a proper schema. We define

$$\begin{aligned} \Downarrow_{\oplus} &:= \left[\begin{array}{l} \mathbf{P}(X) \rightarrow \mathbf{Prel}(X) \\ J \mapsto \perp_{\mathbf{pr}(X, J)} \end{array} \right] \\ \Uparrow_{\oplus} &:= \left[\begin{array}{l} \mathbf{P}(X) \rightarrow \mathbf{Prel}(X) \\ J \mapsto \top_{\mathbf{pr}(X, J)} \end{array} \right] \\ \Downarrow_{\parallel} &:= \left[\begin{array}{l} \mathbf{Prel}(X) \times \mathbf{P}(I) \rightarrow \mathbf{Prel}(X) \\ \langle R, J \rangle \mapsto R \parallel \mathbf{pr}(X, J) \end{array} \right] \\ \Downarrow_{\downarrow} &:= \left[\begin{array}{l} \mathbf{Prel}(X) \times \mathbf{P}(I) \rightarrow \mathbf{Prel}(X) \\ \langle R, J \rangle \mapsto R \Downarrow \mathbf{pr}(X, J) \end{array} \right] \\ \Downarrow_{\uparrow} &:= \left[\begin{array}{l} \mathbf{Prel}(X) \times \mathbf{P}(I) \rightarrow \mathbf{Prel}(X) \\ \langle R, J \rangle \mapsto R \Uparrow \mathbf{pr}(X, J) \end{array} \right] \end{aligned}$$

23.4.3 Definition

For every proper schema $X = [X_i | i \in I]$ we define

$$\begin{aligned} \mathfrak{Prel}_2^{\triangleleft}(X) &:= \left\langle \begin{array}{l} \mathbf{Prel}(X), \mathbf{P}(I), \\ \triangleleft, \underline{\triangle}, \perp_{\oplus}, \top_{\oplus}, @, \mathbf{x}, \parallel, \Downarrow, \Uparrow, \downarrow, \uparrow \end{array} \right\rangle \\ &\quad \text{the 2-carrier syntactic structure of } X \\ \mathfrak{Prel}_2^{\triangleleft}(X) &:= \mathfrak{Prel}_2^{\triangleleft}(X) \uparrow \mathfrak{P}(I) \\ &\quad \text{the 2-structure syntactic structure of } X \end{aligned}$$

23.5 The 1-carrier syntactic structures on $\mathbf{Prel}(X)$

23.5.1 Remark

As yet another variation of the syntactic structure we could reduce the carrier classes even further and only keep the relation class $\mathbf{Prel}(X)$ itself. This way, we derive a “1-carrier syntactic structure” from the 2-carrier version by e.g. replacing $R \downarrow J$ for $R \in \mathbf{Prel}(X)$ and $J \in \mathbf{P}(I)$ by $R \downarrow @ (S)$ where $R, S \in \mathbf{Prel}(X)$.

From the \triangleleft -ordered point of view, there is a great resemblance between the 1-, 2-, and 3-carrier syntactic structures and it is easy to change from one to the other, as we demonstrated in 23.3.5, for example. In other words, operations from one structure can be simulated by operations of the other up to $\underline{\triangle}$ -equivalence.³⁶ With one exception: the 1-carrier syntactic structure $\mathfrak{Prel}_1^{\triangleleft}(X)$ is not able to quasi-simulate a complete quasi-boolean algebra on $\langle \mathbf{Prel}(X), \triangleleft \rangle$, but only the weaker generalized quasi-boolean algebra. In general, it might not have a top element and no complementation, but only the relative complementation (23.5.4).

Many mathematical surveys, universal algebra for example,

³⁶Yet another criterion for this “equal power” of the structures is to say that, for every given class \mathcal{R} of relations, its *closure*, i.e. the class of all the relations that can be generated from \mathcal{R} with the given operations, is quasi-equal for each structure. We introduce subalgebras and closures properly at a later point.

concentrate on algebras as structures with just one carrier class. With the 1-carrier structures on $\mathbf{Prel}(X)$, we can compare and integrate theory algebras in this broader contexts. This allows us to formulate universal properties like the associativity

$$(R \parallel S) \parallel T = R \parallel (S \parallel T)$$

(which actually happens to hold for \parallel , but not for \Downarrow and \Uparrow), etc.

Again we distinguish the new version, say \parallel , of the original operation \parallel by an additional dot on top. And this notational distinction again should be superfluous in most practical circumstances.

23.5.2 Definition

Let $X = [X_i | i \in I]$ be a proper schema. We define

$$\perp_{\oplus} := \left[\begin{array}{l} \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ R \mapsto \perp_{\mathbf{x}(R)} \end{array} \right]$$

$$\dagger_{\oplus} := \left[\begin{array}{l} \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ R \mapsto \top_{\mathbf{x}(R)} \end{array} \right]$$

$$\textcircled{1} \parallel \textcircled{2} := \left[\begin{array}{l} \mathbf{Prel}(X) \times \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ \langle R, S \rangle \mapsto R \parallel \mathbf{x}(S) \end{array} \right]$$

$$\textcircled{1} \Downarrow \textcircled{2} := \left[\begin{array}{l} \mathbf{Prel}(X) \times \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ \langle R, S \rangle \mapsto R \Downarrow \mathbf{x}(S) \end{array} \right]$$

$$\textcircled{1} \Uparrow \textcircled{2} := \left[\begin{array}{l} \mathbf{Prel}(X) \times \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ \langle R, S \rangle \mapsto R \Uparrow \mathbf{x}(S) \end{array} \right]$$

$$\textcircled{1} \dot{\parallel} \textcircled{2} := \left[\begin{array}{l} \mathbf{Prel}(X) \times \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ \langle R, S \rangle \mapsto R \dot{\parallel} \textcircled{2} \end{array} \right]$$

$$\textcircled{1} \dot{\Uparrow} \textcircled{2} := \left[\begin{array}{l} \mathbf{Prel}(X) \times \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ \langle R, S \rangle \mapsto R \dot{\Uparrow} \textcircled{2} \end{array} \right]$$

23.5.3 Definition

For every proper schema $X = [X_i | i \in I]$ we define

$$\mathfrak{Prel}_1^{\triangleleft}(X) := \left\langle \mathbf{Prel}(X), \triangleleft, \triangle, \perp_{\oplus}, \dagger_{\oplus}, \parallel, \Downarrow, \Uparrow, \dot{\parallel}, \dot{\Uparrow} \right\rangle$$

the 1-carrier syntactic structure of X

23.5.4 Lemma

Given a proper schema $X = [X_i | i \in I]$ and a syntactic selector ζ on X . As described in 7.6.16 we construct

$$\textcircled{1} -_{\zeta} \textcircled{2} := \left[\begin{array}{l} \mathbf{Prel}(X) \times \mathbf{Prel}(X) \longrightarrow \mathbf{Prel}(X) \\ \langle R, S \rangle \mapsto R \wedge_{\zeta} (-_{\zeta} S) \end{array} \right]$$

the *relative complementor* in $\mathfrak{Prel}_{\zeta}^{\triangleleft}(X)$. Now for all $R, S \in \mathbf{Prel}(X)$ the following statements are true:

(1) If $\varepsilon \in \{\dot{\parallel}, \dot{\Uparrow}\}$ and $\rho \in \{\Downarrow, \Uparrow\}$ then

- (a) $\emptyset_{\zeta} \triangleq R \varepsilon R$
- (b) $R -_{\zeta} S \triangleq R \varepsilon S$
- (c) $R \wedge_{\zeta} S \triangleq R \varepsilon (R \varepsilon S)$
- (d) $R \vee_{\zeta} S \triangleq R \dot{\parallel} S \triangleq S \dot{\parallel} R$

(2) On the other hand

- (a) $\perp_R \triangleq R$
- (b) $\dagger_R \triangleq R$
- (c) $R \dot{\parallel} S \triangleq R \vee_{\zeta} S$
- (d) $R \Downarrow S \triangleq S$
- (e) $R \Uparrow S \triangleq S$
- (f) $R \dot{\parallel} S \triangleq R -_{\zeta} S$
- (g) $R \dot{\Uparrow} S \triangleq R -_{\zeta} S$

23.5.5 Proof of 23.5.4

We need to prove the truth of statements of the form $\sigma_1 \triangleq \sigma_2$. And obviously, the σ_1, σ_2 are elements of $\mathbf{Prel}(X)$ in each case, no matter what ε and ρ we choose. Therefore our proof is given by a demonstration of $\textcircled{1}(\sigma_1) = \textcircled{1}(\sigma_2)$. We repeat, that

- (i) $\textcircled{1}(R \parallel S) = \textcircled{1}(R) \cup \textcircled{1}(S)$
- (ii) $\textcircled{1}(R \varepsilon S) = \textcircled{1}(R) \setminus \textcircled{1}(R)$ for each $\varepsilon \in \{\dot{\parallel}, \dot{\Uparrow}\}$
- (iii) $\textcircled{1}(R \rho S) = \textcircled{1}(S)$ for each $\rho \in \{\Downarrow, \Uparrow\}$

Applying this we derive

$$(1)(a) \quad \textcircled{1}(\emptyset_{\zeta}) = \text{dom}(\langle \rangle) = \emptyset = \textcircled{1}(R) \setminus \textcircled{1}(R) = \textcircled{1}(R \varepsilon R)$$

$$(1)(b) \quad \begin{aligned} \textcircled{1}(R -_{\zeta} S) &= \textcircled{1}(R \wedge_{\zeta} (-_{\zeta} S)) \\ &= \textcircled{1}(\zeta(\mathbf{x}(R) \wedge \zeta(\mathbf{x}(X \setminus \mathbf{x}(S)))))) \\ &= \textcircled{1}(\zeta(\mathbf{x}(R) \wedge (X \setminus \mathbf{x}(S)))) \\ &= \textcircled{1}(\zeta(\mathbf{x}(R) \setminus \mathbf{x}(S))) = \textcircled{1}(R) \setminus \textcircled{1}(S) = \textcircled{1}(R \varepsilon S) \end{aligned}$$

$$(1)(c) \quad \begin{aligned} \textcircled{1}(R \wedge_{\zeta} S) &= \textcircled{1}(\zeta(\mathbf{x}(R) \wedge \mathbf{x}(S))) \\ &= \text{dom}(\mathbf{x}(R) \wedge \mathbf{x}(S)) = \textcircled{1}(R) \cap \textcircled{1}(S) \\ &= \textcircled{1}(R) \setminus (\textcircled{1}(R) \setminus \textcircled{1}(S)) = \textcircled{1}(R \varepsilon (R \varepsilon S)) \end{aligned}$$

$$(1)(d) \quad \begin{aligned} \textcircled{1}(R \vee_{\zeta} S) &= \textcircled{1}(\zeta(\mathbf{x}(R) \vee \mathbf{x}(S))) \\ &= \text{dom}(\mathbf{x}(R) \vee \mathbf{x}(S)) = \textcircled{1}(R) \cup \textcircled{1}(S) \\ &= \textcircled{1}(R \dot{\parallel} S) = \textcircled{1}(S \dot{\parallel} R) \end{aligned}$$

$$(2)(a) \quad \textcircled{1}(\perp_R) = \textcircled{1}(\perp_{\mathbf{x}(R)}) = \text{dom}(\mathbf{x}(R)) = \textcircled{1}(R)$$

(2)(b) similar to (a).

(2)(c) see (1)(d).

$$(2)(d) \quad \textcircled{1}(R \Downarrow S) = \textcircled{1}(R \Downarrow \mathbf{x}(S)) = \text{dom}(\mathbf{x}(S)) = \textcircled{1}(S)$$

(2)(e) similar to (d).

(2)(f) see (1)(b).

(2)(g) again see (1)(b).

23.6 Theory structures on $\mathbf{Prel}(X)$

23.6.1 Remark

We now combine the semantic and syntactic structure on $\mathbf{Prel}(X)$ into a single “*theory structure*”. Formally speaking, the general idea here is to introduce

$$\mathfrak{Prel}(X) := \mathfrak{Prel}^{\sqsubseteq}(X) \dagger \mathfrak{Prel}^{\sqsupseteq}(X)$$

But since we have several versions of syntactic structures, we also have several of these theoretic structures as well.

23.6.2 Definition

For every proper schema $X = [X_i | i \in I]$ we define:

$$\mathfrak{Prel}_1(X) := \mathfrak{Prel}^{\sqsubseteq}(X) \dagger \mathfrak{Prel}_1^{\sqsupseteq}(X)$$

the single-sort projection relation structure of X

$$\mathfrak{Prel}_2(X) := \mathfrak{Prel}^{\sqsubseteq}(X) \dagger \mathfrak{Prel}_2^{\sqsupseteq}(X)$$

the two-sort projection relation structure of X

$$\mathfrak{Prel}_3(X) := \mathfrak{Prel}^{\sqsubseteq}(X) \dagger \mathfrak{Prel}_3^{\sqsupseteq}(X)$$

the three-sort projection relation structure of X

And if ζ is a syntactic selector on X , then

$$\mathfrak{Prel}_{\zeta}(X) := \mathfrak{Prel}^{\sqsubseteq}(X) \dagger \mathfrak{Prel}_{\zeta}^{\sqsupseteq}(X)$$

the ζ -selector projection relation structure of X

23.6.3 Remark

Let us repeat these new structures in more detail, assuming that some proper schema $X = [X_i | i \in I]$ is given. In all cases, we first list the carrier classes, then the syntactic and finally the semantic operations, where double occurring operations are only mentioned once.

(1) The single-sort projection relation structure of X is

$$\mathfrak{Prel}_1(X) = \left\langle \begin{array}{l} \mathbf{Prel}(X), \\ \sqsubseteq, \sqsupseteq, \perp, \top, \sqcap, \sqcup, \prod, \coprod, \neg, \\ \leq, \triangleq, \perp_{\oplus}, \top_{\oplus}, \parallel, \psi, \uparrow, \downarrow, \dot{\perp} \end{array} \right\rangle$$

(2) The two-sort projection relation structure of X is

$$\mathfrak{Prel}_2(X) = \left\langle \begin{array}{l} \mathbf{Prel}(X), \mathbf{P}(I), \sqsubseteq, \sqsupseteq, \perp, \top, \sqcap, \sqcup, \prod, \coprod, \neg, \\ \leq, \triangleq, \perp_{\oplus}, \top_{\oplus}, @, \parallel, \psi, \uparrow, \downarrow, \subseteq, \emptyset, I, \cap, \cup, \cap, \cup, \mathcal{C} \end{array} \right\rangle$$

(3) The three-sort projection relation structure of X is

$$\mathfrak{Prel}_3(X) = \left\langle \begin{array}{l} \mathbf{Prel}(X), \mathbf{Proj}(X), \mathbf{P}(I), \sqsubseteq, \sqsupseteq, \perp, \top, \sqcap, \sqcup, \prod, \coprod, \neg, \\ \leq, \triangleq, \perp_{\oplus}, \top_{\oplus}, @, \mathbf{x}, \parallel, \psi, \uparrow, \downarrow, \\ \leq, \langle \rangle, X, \wedge, \vee, \bigwedge, \bigvee, -, \subseteq, \emptyset, I, \cap, \cup, \cap, \cup, \mathcal{C} \end{array} \right\rangle$$

(4) For every syntactic selector ζ on X , the selective projection relation structure of X is

$$\mathfrak{Prel}_{\zeta}(X) = \left\langle \begin{array}{l} \mathbf{Prel}(X), \\ \sqsubseteq, \sqsupseteq, \perp, \top, \sqcap, \sqcup, \prod, \coprod, \neg, \\ \leq, \triangleq, \theta_{\zeta}, \mathbf{1}_{\zeta}, \wedge_{\zeta}, \vee_{\zeta}, \bigwedge_{\zeta}, \bigvee_{\zeta}, \neg_{\zeta} \end{array} \right\rangle$$

A Summary of symbols

Chapter I Introduction

Chapter II The language of mathematics

$o(\xi_1, \dots, \xi_n)$ (5.2.4)

default notation for the application of an operator o on arguments ξ_1, \dots, ξ_n

①, ②, ③, (5.2.4 and 5.2.5)

placeholders used in the definition of symbols

$\xi[x/v]$ (5.2.8)

substitution of the expression v for the identifier x in the expression ξ

$\xi := \omega$ (5.3.5)

assignment

$\xi : \tau$ (5.3.5)

type expression

$\xi := \begin{bmatrix} \tau \\ \omega \end{bmatrix}$ (5.3.5)

default notation for an operator definition

let ① **in** ② (5.3.7)

local definition with a let expression

where ① (5.3.7)

local definition with a where expression

$\langle x_1, \dots, x_n \rangle$ (5.4.1 and 9.2.1)

n -tuple

$\langle \rangle$ (5.4.1, 5.7.6 and 9.2.2)

empty tuple, empty function or empty record

lg (x) (5.4.1)

the length of a tuple x

$x \uparrow y$ (5.4.1 and 10.5.1)

the concatenation of two tuples x and y

if φ **then** θ_1 **else** θ_2 (5.4.3)

conditional expression

$\begin{cases} \theta_1 & \text{if } \varphi_1 \\ \theta_2 & \text{if } \varphi_2 \\ \vdots & \vdots \\ \vdots & \vdots \end{cases}$ (5.4.3)

default notation for nested conditional expressions

the ① **with** ② (5.4.4)

a description

sing (C) (5.4.4 and 5.6.13)

the unique element of a singleton class C

① = ② (5.5.1)

equation

false or **0** (5.5.2)

zero bit or false value

true or **1** (5.5.2)

unit bit or true value

\mathbb{B} (5.5.2 and 17.2.8)

bit value class or boolean value class; also the empty schema relation class

$\neg\varphi$ (5.5.3)

not φ or the negation of φ

$\varphi_1 \wedge \dots \wedge \varphi_n$ (5.5.3 and 5.5.4)

φ_1 and ... and φ_n or conjunction

$\varphi_1 \vee \dots \vee \varphi_n$ (5.5.3 and 5.5.4)

φ_1 or ... or φ_n or disjunction

$\varphi_1 \rightarrow \dots \rightarrow \varphi_n$ (5.5.3 and 5.5.4)

the subjunction of $\varphi_1, \dots, \varphi_n$

$\varphi_1 \leftrightarrow \dots \leftrightarrow \varphi_n$ (5.5.3 and 5.5.4)

the equijunction of $\varphi_1, \dots, \varphi_n$

$\exists x . \varphi$ or $\exists x : C . \varphi$ or $\exists x \in C . \varphi$ (5.5.5)

the existentialization

$\forall x . \varphi$ or $\forall x : C . \varphi$ or $\forall x \in C . \varphi$ (5.5.5)

the generalization

$\varphi_1 \Rightarrow \varphi_2$ (5.5.7)

means φ_1 entails φ_2 ; the subvalence, entailment or consequence relation

$\varphi_1 \Leftrightarrow \varphi_2$ (5.5.7)

the equivalence or φ_1 and φ_2

$\{x \mid \varphi\}$ or $\{x \in C \mid \varphi(x)\}$ etc. (5.6.2)

the usual class expressions

\emptyset (5.6.2)

empty class

$e \in C$ or $e \notin C$ (5.6.3)

the usual membership or its negation, respectively

$C_1 \subseteq C_2$, $C_1 \not\subseteq C_2$, $C_1 \subset C_2$, $C_1 \supseteq C_2$ etc. (5.6.13)

the usual inclusion relation on classes with its standard variations

$C_1 \cap C_2$ (5.6.13)

the intersection of classes C_1 and C_2

$C_1 \cup C_2$ (5.6.13)

the union of classes C_1 and C_2

$C_1 \uplus C_2$ (5.6.13)

the disjunct union of two (disjunct) classes C_1 and C_2

$\bigcap \mathcal{K}$ (5.6.13)

the (big) intersection of \mathcal{K}

$\bigcup \mathcal{K}$ (5.6.13)

the (big) union of \mathcal{K}

$C_1 \setminus C_2$ (5.6.13)

the difference of two classes C_1 and C_2

$C_1 \nabla \dots \nabla C_2$ (5.6.6)

the opposition or symmetric difference of classes C_1, \dots, C_n

$\nabla \mathcal{K}$ (5.6.6)

the (big) opposition of a class

$\mathbf{P}(C)$ (5.6.13)

the power class of a class C

Fin (C) (5.6.13)

the finite (sub)classes of a class C

Sg (C) (5.6.13)

the singleton class of a class C

$f : D \dashrightarrow C$ (5.6.8 and 5.7.15)

f is a partial function from D to C

$f : D \longrightarrow C$ (5.6.8 and 5.7.3)

f is a function from D to C

$R : D_1 \rightsquigarrow \dots \rightsquigarrow D_n$ (5.6.8 and 5.7.10 and 17.3.1)

R is a n -ary or ordinary relation on D_1, \dots, D_n

$R : \mathbf{Pty}(D)$ (5.6.8)

R is a unary relation or property on D

$R : \mathbb{B}$	(5.6.8 and 5.5.2)
R is a <u>nullary relation</u> , which is one of two values	
$C_1 \times \dots \times C_n$	(5.6.9 and 12.2.2)
the <u>n-ary cartesian product</u> of classes C_1, \dots, C_n	
C^n	(5.6.9 and 12.5.1)
the <u>n-th (cartesian) power</u> or <u>n-tuple class</u> of a class C	
C^*	(5.6.9 and 12.5.1)
the <u>Kleene closure</u> or <u>tuple class</u> of C	
$\text{Char} = \{ '0', \dots, '9', 'A', \dots, 'a', \dots \}$	(5.6.10)
some <u>character class</u>	
String	(5.6.10)
the class of <u>strings</u> or <u>character tuples</u>	
"Hallo world"	(5.6.10)
an <u>example string</u>	
$\mathbb{N} := \{0, 1, \dots\}$	(5.6.11)
the <u>natural number class</u>	
\mathbb{Z}	(5.6.11)
the class of <u>integers</u>	
\mathbb{Q}	(5.6.11)
the class of <u>rational numbers</u>	
\mathbb{R}	(??)
the class of <u>real numbers</u>	
$0, 1, +, -, \cdot, /, \leq, <, \min, \max, \dots$	(5.6.12)
the <u>usual operations</u> on numbers	
card (C)	(5.6.13)
the <u>cardinality</u> or <u>cardinal number</u> of C	
$x \mapsto \theta$	(5.7.1)
<u>map(ping) expression</u>	
$(x \mapsto \theta)(\sigma)$	(5.7.1)
the <u>application</u> of a map on a term σ	
$\langle x_1, \dots, x_n \rangle \mapsto \theta$	(5.7.1)
<u>n-ary map(ping) expression</u>	
$\begin{bmatrix} \xi_1 \mapsto \theta_1 \\ \vdots \\ \xi_n \mapsto \theta_n \end{bmatrix}$	(5.7.1 and 9.1.5)
conditional map expression or <u>finite record expression</u>	
$\begin{bmatrix} D \rightarrow C \\ x \mapsto \theta \end{bmatrix}$	(5.7.3)
default form of a <u>(total) function</u>	
$\begin{bmatrix} D \dashrightarrow C \\ x \mapsto \theta \end{bmatrix}$	(5.7.15)
default form of a <u>partial function</u>	
dom (f)	(5.7.3 and 5.7.15)
the <u>domain</u> of a (partial) function f	
cod (f)	(5.7.3 and 5.7.15)
the <u>codomain</u> of a (partial) function f	
def (f)	(5.7.15)
the <u>defined domain</u> of a partial function f	
$f \upharpoonright Z$	(5.7.6)
the <u>(domain) restriction</u> of a function f onto Z	
$g \circ f$	(5.7.6 and 5.7.17)
the <u>composition</u> of two (partial) functions	
id $_C$	(5.7.6)
the <u>identity function</u> of a class C	
f^{-1}	(5.7.17)
the <u>inverse</u> of a (partial) function f	
$f[A]$	(5.7.18)
the <u>image class</u> of A for a (partial) function f	

$f^{-1}[A]$	(5.7.18)
the <u>coimage class</u> of A for a (partial) function f	
$x \rightsquigarrow \varphi$	(5.7.8)
a <u>predicator expression</u>	
$(x \rightsquigarrow \varphi)(\theta)$	(5.7.8)
the <u>application</u> of a predicator on a term θ	
$\langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi$	(5.7.8)
a <u>n-ary predicator expression</u>	
$\begin{bmatrix} \xi_1 \rightsquigarrow \varphi_1 \\ \vdots \\ \xi_n \rightsquigarrow \varphi_n \end{bmatrix}$	(5.7.8)
conditional predicator expression	
$\begin{bmatrix} D_1 \leftrightarrow \dots \leftrightarrow D_n \\ \langle x_1, \dots, x_n \rangle \rightsquigarrow \varphi \end{bmatrix}$	(5.7.13)
the default form of a <u>n-ary relation</u>	
dom (R)	(5.7.10 and 17.2.2)
the <u>domain</u> of a relation R	
gr (R)	(5.7.10 and 17.2.2)
the <u>graph</u> of a relation R	
$x \in R$ (or $x \notin R$)	(5.7.10 and 17.2.5)
tuple or record x is (not) a <u>member</u> of the relation R , or R <u>holds</u> for x (or not)	
$R[Z]$	(5.7.13)
the <u>image class</u> of Z for a binary relation R	
$R^{-1}[Z]$	(5.7.13)
the <u>coimage class</u> of Z for a binary relation R	
$\mathfrak{S} = \langle \mathcal{K}, \mathcal{O} \rangle$ and $\mathfrak{S} = \langle C_1, \dots, c_1, \dots, f_1, \dots, R_1, \dots \rangle$	(5.8.1)
general forms for <u>(ordinary) structures</u>	
$\mathfrak{B} = \langle \mathbb{B}, \leq, 0, 1, -, \wedge, \vee \rangle$	(5.8.4)
the <u>boolean (or bit) value algebra</u>	
$\mathfrak{I} = \langle \mathbb{Z}, 0, 1, +, -, \cdot, /, \leq \rangle$	(5.8.5)
the linearly ordered <u>integer ring</u>	
$\mathfrak{Q} = \langle \mathbb{Q}, 0, 1, +, -, \cdot, /, \leq \rangle$	(5.8.5)
the linearly ordered <u>field of rational numbers</u>	
$\mathfrak{S} \uparrow \mathfrak{S}'$	(5.8.6)
the <u>combination</u> of two structures \mathfrak{S} and \mathfrak{S}'	
$\mathfrak{h} : \mathfrak{S} \rightarrow \mathfrak{S}'$	(5.8.9)
a <u>homomorphism</u> from \mathfrak{S} into \mathfrak{S}'	
$\mathfrak{h} : \mathfrak{S} \cong \mathfrak{S}'$	(5.8.9)
an <u>isomorphism</u> from \mathfrak{S} into \mathfrak{S}'	
$\mathfrak{S} \cong \mathfrak{S}'$	(5.8.9)
\mathfrak{S} and \mathfrak{S}' are <u>isomorph</u>	
Chapter III Quasi-hierarchies	
$\mathfrak{P}(C) := \langle \mathbf{P}(C), \subseteq, \emptyset, 1, \cap, \cup, \bigcap, \bigcup, \emptyset \rangle$	(6.2.2)
the <u>power algebra</u> or <u>subclass algebra</u> of C	
$\mathfrak{Fin}(C) := \langle \mathbf{Fin}(C), \subseteq, \emptyset, \cap, \cup, \setminus \rangle$	(6.2.2)
the <u>finite power algebra</u> of finite subclass algebra of C	
$\langle C^*, \subseteq_*, \equiv_*, \setminus, \cap_*, \cup_*, \setminus_* \rangle$	(6.3.1)
the generalized quasi-boolean algebra on tuples over C	
Form (A)	(6.4.1)
the <u>propositional formula class</u> on A	
$\mathfrak{Form}(A) = \langle \mathbf{Form}(A), \Rightarrow, \Leftrightarrow, \mathbf{f}, \mathbf{t}, \cap, \cup, \neg \rangle$	(6.4.3)
the <u>standard quasi-boolean algebra</u> on Form (A)	
$\mathfrak{a}(\varphi)$	(6.4.4)
the <u>atom class</u> of a given formula	
$\varphi \preceq \psi$	(6.4.5)

the <u>subatomic relation</u> on formulas	
$\varphi \triangleq \psi$	(6.4.5)
the <u>equiatomic relation</u> on formulas	
$\varphi \parallel \langle \alpha_1, \dots, \alpha_n \rangle$	(6.4.7)
the <u>atomic expansion</u> of a formula φ by atoms $\alpha_1, \dots, \alpha_n$	
$\varphi \uparrow \langle \alpha_1, \dots, \alpha_n \rangle$	(6.4.8)
the <u>infimum reduction</u> of a formula φ onto $\alpha_1, \dots, \alpha_n$	
$\varphi \downarrow \langle \alpha_1, \dots, \alpha_n \rangle$	(6.4.8)
the <u>supremum reduction</u> of a formula φ onto $\alpha_1, \dots, \alpha_n$	
$\langle \mathbb{Z}^\infty, \leq, -\infty, \infty, \min, \max \rangle$	(6.5.2)
the expansion of the linearly ordered integers to a bounded lattice	
$m \mid n$	(6.5.4)
m divides n	
$\langle \mathbb{F}, \succsim, \approx, \mathbf{0}, \mathbf{1}, +, -, \cdot, \div \rangle$	(6.6.1)
the <u>quasi-ordered field of fractions</u>	
$\rho \upharpoonright_C$	(7.2.1)
the <u>restriction</u> of a binary endorelation ρ onto C	
$\langle C, \sqsubseteq, \equiv \rangle$	(7.3.2)
one representation of a quasi-ordered class	
\perp	(7.4.1)
standard symbol for a <u>bottom, zero</u> or <u>least</u> element	
\top	(7.4.1)
standard symbol for a <u>top, unit</u> or <u>greatest</u> element	
$\odot \sqcap \odot$	(7.4.3)
standard symbol for a <u>meet</u> function or <u>conjunctor</u>	
$\odot \sqcup \odot$	(7.4.3)
standard symbol for a <u>join</u> function or <u>disjunctor</u>	
min	(7.4.4)
<u>minimum function</u>	
max	(7.4.4)
<u>maximum function</u>	
$\prod \odot$	(7.4.7)
standard symbol for a <u>infimum</u> function or <u>big conjunctor</u>	
$\prod \odot$	(7.4.7)
standard symbol for a <u>supremum</u> function or <u>big disjunctor</u>	
$\neg \odot$	(7.5.2)
standard symbol for a <u>negator</u> or <u>complement function</u>	
$\odot - \odot$	(7.5.5)
standard symbol for a <u>relative complement function</u>	
$\langle Q, \sqsubseteq, \equiv, \sqcap, \sqcup \rangle$	(7.6.2)
standard form of a <u>quasi-lattice algebra</u>	
$\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup \rangle$	(7.6.2)
standard form of a <u>bounded quasi-lattice algebra</u>	
$\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \prod, \prod \rangle$	(7.6.2)
standard form of a <u>complete quasi-lattice algebra</u>	
$\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \neg \rangle$	(7.6.2)
standard form of a <u>quasi-boolean algebra</u>	
$\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \prod, \prod, \neg \rangle$	(7.6.2)
standard form of a <u>complete quasi-boolean algebra</u>	
$\langle Q, \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, - \rangle$	(7.6.2)
standard form of a <u>generalized quasi-boolean algebra</u>	
(\rightarrow) or $(\rightarrow x)$ or $x_1 \rightarrow \dots \rightarrow x_n$ or $\prod_{i=1}^n x_i$	(8.2.2 and 8.2.3)
<u>subjunctions</u>	
(\leftrightarrow) or $(\leftrightarrow x)$ or $x_1 \leftrightarrow \dots \leftrightarrow x_n$ or $\prod_{i=1}^n x_i$	(8.2.2 and 8.2.3)
<u>equijunctions</u>	

Chapter IV Records

$[i \mapsto \xi_i \mid i \in I]$ or $\left[\begin{array}{c} i \mapsto \xi_i \\ i \in I \end{array} \right]$	(9.1.2 and 9.1.3)
usual form of a <u>record</u>	
$[\xi_i \mid i \in I]$ or $\left[\begin{array}{c} \xi \\ i \in I \end{array} \right]$	(9.1.2 and ??)
also a usual form of a <u>record</u>	
dom (ξ)	(9.1.2)
the <u>domain</u> or <u>index class</u> or <u>attribute class</u> of a record ξ	
REC	(9.1.4)
the overall class of records	
$\left[\begin{array}{c} i_1 \mapsto \xi_1 \\ \vdots \\ i_n \mapsto \xi_n \end{array} \right]$	(9.1.5)
representation for a finite record	
$\langle \xi_1, \dots, \xi_n \rangle$	(9.2.1)
n -ary tuple	
$[c \mid I]$	(9.2.4)
representation for a <u>univalent</u> record, which has the same value c for each index in I	
$\langle c \mid n \rangle$	(9.2.6)
representation for the univalent n -ary tuple where every component is c	
$[X_i \mid i \in I]$	(9.3.1)
a <u>schema</u> is a record, where each value X_i is a class	
$\xi(i)$	(10.1.1)
the <u>value</u> at i or i -th component of a record ξ	
dom (ξ)	(10.1.1)
the <u>domain</u> or <u>index class</u> or <u>attribute class</u> of a record ξ	
cod (ξ)	(10.1.1)
the <u>codomain</u> or <u>value class</u> of a record ξ	
pr (ξ, J)	(10.2.1)
the <u>projection</u> of a record ξ onto J	
Proj (ξ)	(10.2.6)
the class of all projections of a given record ξ	
$\xi \dot{\neq} v$ (or $\xi \dot{\neq} v$)	(10.3.2 and 10.3.3)
the records ξ and v are <u>distinct</u> (or not distinct)	
$\xi \sim v$ (or $\xi \not\sim v$)	(10.3.2 and 10.3.3)
the records ξ and v are <u>compatible</u> (or not compatible)	
$\xi \leq v$ (or $\xi \not\leq v$)	(10.3.2 and 10.3.3)
ξ is <u>subrecord</u> or <u>smaller</u> than v (or not)	
$\xi < v$ (or $\xi \not< v$)	(10.3.2 and 10.3.3)
ξ is a <u>proper subrecord</u> or <u>strictly smaller</u> than v (or not)	
$\xi \dot{\vee} v$	(10.4.1)
the <u>distinct join</u> of distinct records ξ and v	
$\dot{\vee} \Xi$	(10.4.1)
the <u>distinct join</u> of a (pairwise) distinct record class Ξ	
$\xi_1 \dagger \dots \dagger \xi_n$ or $\prod_{i=1}^n \xi_i$	(10.5.1)
the <u>concatenation</u> of tuples ξ_1, \dots, ξ_n	
Rec (I, C)	(11.2.2)
the class of all the records with indices from I and values from C	
DΞ	(11.3.2)
the (total) <u>domain</u> of a record class Ξ	
D$\Xi^{\not\sim}$	(11.3.2)
the <u>incompatible domain</u> of a record class Ξ	
DΞ^{\sim}	(11.3.2)
the <u>compatible domain</u> of a record class ξ	
$\xi \setminus v$	(11.4.3)
the <u>subtraction</u> of records ξ and v , or ξ <u>without</u> v	

$\xi \wedge v$ (11.4.3)
	the <u>meet</u> of records ξ and v , or ξ <u>and</u> v
$\xi \times v$ (11.4.3)
	the <u>rejection</u> of records ξ and v , or ξ <u>rejected by</u> v
$\xi \nabla v$ (11.4.3)
	the <u>opposition</u> of records ξ and v , or ξ <u>opposed to</u> v
$\xi \vee v$ (11.4.3)
	the <u>join</u> of records ξ and v , or ξ <u>or</u> v
$\xi \surd v$ (11.4.3)
	the <u>update</u> of records ξ and v , or ξ <u>updated with</u> v
$\Xi(k)$ (11.7.2)
	the value of a record class Ξ at k
$\bigvee \Xi$ (11.7.2)
	the <u>supremum</u> or (<u>big</u>) <u>disjunction</u> of a record class Ξ
$\bigwedge \Xi$ (11.7.2)
	the <u>infimum</u> or (<u>big</u>) <u>conjunction</u> of a record class Ξ
$\bigtriangledown \Xi$ (11.7.2)
	the (<u>big</u>) <u>opposition</u> of a record class Ξ
$\mathfrak{R} \in \mathfrak{C}$ (11.8.1)
	the (<u>general</u>) record structure
$\mathfrak{Pr}(\eta)$ (11.9.1)
	the <u>record projection structure</u> of a given record η

Chapter V Schemas and their various products

$\otimes X$ (12.1.2)
	the <u>star product</u> or the (<u>general</u>) record class of a schema X
$\boxtimes X$ (12.1.2)
	the (<u>cartesian</u>) product or the <u>expanded record class</u> of a schema X
$\oplus X$ (12.1.2)
	the <u>coproduct</u> or <u>disjunct union</u> or <u>singular record class</u> or <u>literal record class</u> of a schema X
$C_1 * \dots * C_n$ or $\prod_{i=1}^n C_i$ (12.2.2)
	the (<u>ordinal</u>) star product of classes C_1, \dots, C_n
$C_1 \times \dots \times C_n$ or $\prod_{i=1}^n C_i$ (12.2.2)
	the (<u>ordinal</u> or <u>cartesian</u>) star product of classes C_1, \dots, C_n
$C_1 + \dots + C_n$ or $\bigoplus_{i=1}^n C_i$ (12.2.2)
	the (<u>ordinal</u>) coproduct or <u>disjunct union</u> of classes C_1, \dots, C_n
$L_1 \dagger \dots \dagger L_n$ or $\biguplus_{i=1}^n L_i$ (12.3.2)
	the <u>concatenation</u> of tuple classes L_1, \dots, L_n
C^I (12.5.1)
	the I -th <u>power (product)</u> of C , where I and C are classes
C^n (12.5.1)
	the n -th <u>power (product)</u> or the <u>n-tuple class</u> of C , where C is a class and $n \in \mathbb{N}$
C^* (12.5.1)
	the <u>Kleene closure</u> or the <u>tuple class</u> of a given class C
$\subseteq \supseteq \cap \cup \setminus \bigcap \bigcup$ (14.1.1)
	<u>inclusion</u> , <u>intersection</u> , <u>union</u> and <u>difference</u> defined on schemas with identical domains
$\mathfrak{Incl}(X) = \langle \mathfrak{Incl}(X), \subseteq, \{\emptyset\}, X, \cap, \cup, \bigcap, \bigcup, \mathfrak{C} \rangle$ (14.3.1)
	the <u>inclusion algebra</u> of a given schema X

Chapter VI Graphs and their distinct product

$\textcircled{\Xi}$ (15.1.1)
	the <u>attribute class</u> of a given record class Ξ
$\text{dom}_\alpha(\Xi)$ (15.1.1)
	the α - <u>domain</u> of a given record class Ξ and attribute α of Ξ

$\mathbf{x}(\Xi)$ (15.1.1)
	the <u>schema</u> of a given record class Ξ
$\text{doms}(\rho)$ (16.2.1)
	the <u>domain-schema</u> or <u>index schema</u> of a record-record ρ
$\Gamma_1 \dot{\cap} \dots \dot{\cap} \Gamma_n$ (16.3.1)
	the graphs (record classes) Ξ_1, \dots, Ξ_n are <u>relatively distinct</u>
$\odot \Gamma$ or $\bigcirc_{k \in K} \Gamma$ (16.4.1)
	the <u>distinct product</u> or a relatively distinct graph record Γ
$\Gamma_1 \odot \dots \odot \Gamma_n$ or $\bigcirc_{i=1}^n \Gamma_i$ (16.4.1)
	the <u>distinct product</u> or relatively distinct graphs $\Gamma_1, \dots, \Gamma_n$
π_Γ (16.4.7)
	the bijection from $\otimes \Gamma$ into $\odot \Gamma$ for a relatively distinct graph record Γ
$\text{pr}(\Gamma, J)$ (16.5.2)
	the <u>projection</u> of a graph onto a class J

Chapter VII Relations

$\left[\begin{array}{c} X \\ \Gamma \end{array} \right]$ or $[X, \Gamma]$ (17.2.1 and 17.3.5)
	(the <u>schema-graph form</u> of) a <u>relation</u> with schema X and graph Γ
$\left[\begin{array}{c} X \\ x \mapsto \varphi \end{array} \right]$ (17.3.5)
	(the <u>schema-predicator form</u> of) a <u>relation</u>
$\left[\begin{array}{c} \mathbf{Rel}(X) \\ \Gamma \end{array} \right]$ (17.3.5)
	(the <u>typed-graph form</u> of) a <u>relation</u> with schema X and graph Γ
$\left[\begin{array}{c} \mathbf{Rel}(X) \\ x \mapsto \varphi \end{array} \right]$ (17.3.5)
	(the <u>typed-predicator form</u> of) a <u>relation</u>
$\left[\begin{array}{c} D_1 \leftrightarrow \dots \leftrightarrow D_n \\ (x_1, \dots, x_n) \rightsquigarrow \varphi \end{array} \right]$ (17.3.1)
	(the <u>typed-predicator form</u> of) an <u>ordinary relation</u>
$\left[\begin{array}{c} \langle D_1, \dots, D_n \rangle \\ \{(x_1, \dots, x_n) \in D_1 \times \dots \times D_n \mid \varphi\} \end{array} \right]$ (17.3.1)
	(the <u>schema-graph form</u> of) an <u>ordinary relation</u>
χ_R (17.4.2)
	<u>characteristic function</u> of a relation R
$\mathbf{rel}(\chi)$ (17.4.2)
	the relation of a given characteristic function χ
\textcircled{R} (17.2.2)
	the <u>attribute class</u> or <u>index class</u> of a relation R
$\mathbf{x}(R)$ (17.2.2)
	the <u>schema</u> of a relation R
$\mathbf{gr}(R)$ (17.2.2)
	the <u>graph</u> of a relation R
$\text{dom}(R)$ (17.2.2)
	the <u>domain</u> of a relation R
$\text{dom}_i(R)$ (17.2.2)
	the i - <u>domain</u> of a relation R and each attribute i of R
$x \in R$ or $x \notin R$ (17.2.5)
	a record x is or is not a <u>member</u> of a relation R
\mathbf{REL} (17.2.6)
	the overall class of relations
$\mathbf{Rel}(X)$ (17.2.6)
	the class of all relations with schema X
$\text{rng}_i(R)$ (17.2.11)
	the i -th <u>range</u> of a relation R
$\text{rng}(R)$ (17.2.11)
	the <u>range</u> of a relation R

\mathbb{B} (17.2.8)
	the empty schema relation class ($\mathbb{B} = \mathbf{Rel}(\langle \rangle)$), but also the bit value class (5.5.2); both are made of just two members
$\mathbf{Tab}(X)$ (17.5.3)
	the <u>table</u> class of a given schema X
$\mathbf{Prel}(X)$ (17.8.1)
	the <u>projection relation class</u> of a given schema X
$\mathbf{Brel}(A)$ (17.8.4)
	the <u>bit value relation</u> class of a given class A

$\begin{bmatrix} X \\ \Gamma \end{bmatrix}$ or $[X, \Gamma]$ (18.1.2)
--	----------------

a quasi-relation with schema X and quasi-graph $\Gamma \subseteq \textcircled{X}$

$x \parallel X$ (18.2.3)
-----------------	----------------

the expansion of x by a schema X , where $x \in \textcircled{X}$

$\mathbf{rel}(Q)$ (18.2.6)
-------------------	----------------

the relation of a given quasi-relation Q

$R \parallel Y$ (18.3.3 and 20.1.1)
-----------------	---------------------------

the expansion of a relation R by a schema Y

Chapter VIII Operations on relations

$R \trianglelefteq S$ (or $R \not\trianglelefteq S$) (19.1.2)
---	----------------

relation R is sub-schematic to relation S (or not)

$R \triangleleft S$ (or $R \not\triangleleft S$) (19.1.2)
---	----------------

relation R is equi-schematic to relation S (or not)

$R \sim S$ (or $R \not\sim S$) (19.1.2)
---------------------------------	----------------

relations R and S are compatible (or not)

$R \not\sim S$ (or $R \not\sim S$) (19.1.2)
-------------------------------------	----------------

relations R and S are (attribute or schema) distinct (or not)

\mathbf{Id}_X (19.2.1)
-----------------	----------------

identity relation of X

\perp_X and \top_X (19.3.1)
------------------------	----------------

the empty/bottom/zero and full/top/unit relation of a given schema X

\perp and \top (19.3.1)
--------------------	----------------

the empty/bottom/zero and full/top/unit relation

$\neg R$ or \overline{R} (19.4.1)
----------------------------	----------------

the complement or negation of a relation R

$R \subseteq S$ (or $R \not\subseteq S$) (19.5.1 and 19.5.2)
---	---------------------------

relation R is included by relation S (or not)

$R \subset S$ (or $R \not\subset S$) (19.5.1 and 19.5.2)
---------------------------------------	---------------------------

relation R is properly included by relation S (or not)

$R \cap S$ (19.5.1)
------------	----------------

the intersection of equi-schematic relations R and S

$R \cup S$ (19.5.1)
------------	----------------

the union of equi-schematic relations R and S

$\bigcap \mathcal{R}$ or $\bigcap_{R \in \mathcal{R}} R$ (19.5.3)
--	----------------

the (big) intersection of a class \mathcal{R} of equischematic relations

$\bigcup \mathcal{R}$ or $\bigcup_{R \in \mathcal{R}} R$ (19.5.3)
--	----------------

the (big) union of a class \mathcal{R} of equischematic relations

$\mathfrak{Rel}(X) = \langle \mathbf{Rel}(X), \subseteq, \perp_X, \top_X, \cap, \cup, \bigcap, \bigcup, \neg \rangle$ (19.5.9)
---	----------------

the equi-schematic relation algebra over a given schema X

$R \odot S$ (19.6.2)
-------------	----------------

the distinct (cartesian) product of two distinct relations R and S

$\odot \mathcal{R}$ or $\bigodot_{k \in K} \mathcal{R}_k$ (19.6.2)
---	----------------

the distinct (cartesian) product of a (pairwise) distinct relation class \mathcal{R}

$R \dagger S$ (19.7.2)
---------------	----------------

the concatenation of two ordinary relations R and S

$R \parallel Y$ (20.1.1)
-----------------	----------------

the expansion of a relation R by a schema Y

$R \sqsubseteq S$ (20.2.2)
-------------------	----------------

the subvalence of R under S , for compatible relations R and S

$R \equiv S$ (20.2.2)
--------------	----------------

the equivalence of R and S , for compatible relations R and S

$R \sqsubseteq S, R \supseteq S, R \sqsubset S, R \sqsupset S, R \not\equiv S, \dots$ (20.2.3)
---	----------------

the usual derived operations of \sqsubseteq and \equiv

$R_1 \sqcap \dots \sqcap R_n$ (20.2.2 and 20.4.23)
-------------------------------	----------------------------

the conjunction of compatible relations R_1, \dots, R_n

$R_1 \sqcup \dots \sqcup R_n$ (20.2.2 and 20.4.23)
-------------------------------	----------------------------

the disjunction of compatible relations R_1, \dots, R_n

$\prod \mathcal{R}$ or $\prod_{R \in \mathcal{R}} R$ (20.2.4)
--	----------------

the (big) conjunction of a compatible relation class \mathcal{R}

$\prod \mathcal{R}$ or $\prod_{R \in \mathcal{R}} R$ (20.2.4)
--	----------------

the (big) disjunction of a compatible relation class \mathcal{R}

$R - S$ (20.5.2)
---------	----------------

the subtraction of R by S , for compatible relations R and S

$R \rightarrow S$ (20.5.2)
-------------------	----------------

the subjunctions of R and S , for compatible relations R and S

$R \leftrightarrow S$ (20.5.2)
-----------------------	----------------

the equijunctions of R and S , for compatible relations R and S

$\mathbf{pr}(R, J)$ (21.2.2)
---------------------	----------------

the projection of a relation R onto a class J

$\mathbf{cpj}(R, J)$ (21.2.2)
----------------------	----------------

the coprojection of a relation R onto a class J

$R \downarrow J$ (21.4.1)
------------------	----------------

the upper R without J or the supremum elimination of a class J from a relation R

$R \downarrow j$ (21.4.1)
------------------	----------------

the upper R without j or the supremum elimination of a single j from a relation R

$R \uparrow J$ (21.4.1)
----------------	----------------

the lower R without J or the infimum elimination of a class J from a relation R

$R \uparrow j$ (21.4.1)
----------------	----------------

the lower R without j or the infimum elimination of a single j from a relation R

$R \Downarrow Y$ (21.1.2 and 21.6.1)
------------------	---------------------------

the equivalent reduction of a relation R by a schema Y

$R \Downarrow Y$ (21.6.1)
------------------	----------------

the supremum reduction of a relation R by a schema Y

$R \Uparrow Y$ (21.6.1)
----------------	----------------

the infimum reduction of a relation R by a schema Y

$\mathbf{redAt}(R)$ (21.8.2)
---------------------	----------------

the redundant attribute class of a relation R

$\mathbf{irrAt}(R)$ (21.8.2)
---------------------	----------------

the irredundant attribute class of a relation R

$f([a])$ (22.2.1)
----------	----------------

the generalized f -value of a , for every function f and each a

$f[[A]]$ (22.2.1)
----------	----------------

the generalized f -image class of A , for every function f and each class A

$\mathbf{rec}(\Pi)$ (22.2.2)
---------------------	----------------

the record of a pair class $\Pi \subseteq A \times B$

$\mathbf{gr}(\xi)$ (22.2.2)
--------------------	----------------

the graph of a record ξ

$\mathbf{tr}(\xi, \tau)$ (22.2.3)
--------------------------	----------------

the (attribute) translation of a record ξ by a record τ

$$[k]\xi \dots\dots\dots (22.3.3)$$

the ξ -equivalence class of k , for a record ξ and attribute k

$$\text{tr}(R, \tau) \dots\dots\dots (22.4.1)$$

the (attribute) translation of a relation R by a record τ

$$\text{Trans}(X) \dots\dots\dots (22.5.2)$$

the translator class of a proper schema X

Chapter IX Theory algebras of relations

$$\text{PreI}^{\sqsubseteq}(X) = \langle \text{Prel}(X), \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \Pi, \coprod, \neg \rangle \dots\dots (23.1.1)$$

the semantic algebra on $\text{Prel}(X)$

$$\text{PreI}_{\zeta}^{\triangleleft}(X) = \langle \text{Prel}(X), \triangleleft, \hat{\triangleleft}, \theta_{\zeta}, \mathbf{1}_{\zeta}, \wedge_{\zeta}, \vee_{\zeta}, \wedge_{\zeta}, \vee_{\zeta}, \neg_{\zeta} \rangle \dots\dots (23.2.3)$$

the selective syntactic structure of ζ over X

$$\text{PreI}_{\mathfrak{B}}^{\triangleleft}(X) = \langle \text{Prel}(X), \text{Proj}(X), \mathbf{P}(I), \triangleleft, \hat{\triangleleft}, \perp_{\mathfrak{B}}, \top_{\mathfrak{B}}, \mathfrak{B}, \mathbf{x}, \parallel, \psi, \hat{\psi}, \uparrow, \downarrow \rangle \dots\dots (23.3.2)$$

the 3-carrier syntactic structure of X

$$\text{PreI}_{\mathfrak{B}}^{\triangleleft}(X) = \langle \text{Prel}(X), \text{Proj}(X), \mathbf{P}(I), \triangleleft, \hat{\triangleleft}, \perp_{\mathfrak{B}}, \top_{\mathfrak{B}}, \mathfrak{B}, \mathbf{x}, \parallel, \psi, \hat{\psi}, \uparrow, \downarrow, \leq, \langle \rangle, X, \wedge, \vee, \wedge, \vee, -, \subseteq, \emptyset, \mathbf{1}, \cap, \cup, \cup, \mathbf{U}, \mathbf{0} \rangle \dots\dots (23.3.2)$$

the 3-structure syntactic structure of X

$$\text{PreI}_{\mathfrak{B}}^{\triangleleft}(X) = \langle \text{Prel}(X), \mathbf{P}(I), \triangleleft, \hat{\triangleleft}, \perp_{\mathfrak{B}}, \top_{\mathfrak{B}}, \mathfrak{B}, \parallel, \psi, \hat{\psi}, \uparrow, \downarrow \rangle \dots\dots (23.4.3)$$

the 2-carrier syntactic structure of X

$$\text{PreI}_{\mathfrak{B}}^{\triangleleft}(X) = \langle \text{Prel}(X), \mathbf{P}(I), \triangleleft, \hat{\triangleleft}, \perp_{\mathfrak{B}}, \top_{\mathfrak{B}}, \mathfrak{B}, \parallel, \psi, \hat{\psi}, \uparrow, \downarrow \rangle \dots\dots (23.4.3)$$

the 2-structure syntactic structure of X

$$\text{PreI}_{\mathfrak{B}}^{\triangleleft}(X) = \langle \text{Prel}(X), \triangleleft, \hat{\triangleleft}, \perp_{\mathfrak{B}}, \top_{\mathfrak{B}}, \parallel, \psi, \hat{\psi}, \uparrow, \downarrow \rangle \dots\dots (23.5.3)$$

the 1-carrier syntactic structure of X

$$\text{PreI}_{\mathfrak{B}}(X) = \langle \text{Prel}(X), \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \Pi, \coprod, \neg \rangle \dots\dots (23.6.2)$$

the single-sort projection relation structure of X

$$\text{PreI}_{\mathfrak{B}}(X) = \langle \text{Prel}(X), \mathbf{P}(I), \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \Pi, \coprod, \neg, \triangleleft, \hat{\triangleleft}, \perp_{\mathfrak{B}}, \top_{\mathfrak{B}}, \mathfrak{B}, \parallel, \psi, \hat{\psi}, \uparrow, \downarrow \rangle \dots\dots (23.6.2)$$

the two-sort projection relation structure of X

$$\text{PreI}_{\mathfrak{B}}(X) = \langle \text{Prel}(X), \text{Proj}(X), \mathbf{P}(I), \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \Pi, \coprod, \neg, \triangleleft, \hat{\triangleleft}, \perp_{\mathfrak{B}}, \top_{\mathfrak{B}}, \mathfrak{B}, \mathbf{x}, \parallel, \psi, \hat{\psi}, \uparrow, \downarrow, \leq, \langle \rangle, X, \wedge, \vee, \wedge, \vee, -, \subseteq, \emptyset, I, \cap, \cup, \cup, \mathbf{U}, \mathbf{0} \rangle \dots\dots (23.6.2)$$

the three-sort projection relation structure of X

$$\text{PreI}_{\zeta}(X) = \langle \text{Prel}(X), \sqsubseteq, \equiv, \perp, \top, \sqcap, \sqcup, \Pi, \coprod, \neg, \triangleleft, \hat{\triangleleft}, \theta_{\zeta}, \mathbf{1}_{\zeta}, \wedge_{\zeta}, \vee_{\zeta}, \wedge_{\zeta}, \vee_{\zeta}, \neg_{\zeta} \rangle \dots\dots (23.6.2)$$

the selective projection relation structure of X

Index

- R -coimage class of Z , 28
- R -image class of Z , 28
- \sim -compatible, 44
- ζ -selector projection relation structure, 168
- ξ -equivalence class of k , 159
- f of a , 28
- f -coimage class of A , 29
- f -image class of A , 29
- i -, 54
- i -domain, 99
- i -th component, 21
- i -th component of ξ , 56
- i -th domain, 73
- i -th range, 101
- i -th domain, 54
- n -ary, 14, 26, 27, 53
- n -ary relation, 27
- n -th (cartesian) power, 25
- n -tuple, 21, 54
- n -tuple class, 79
- (atomic) expander, 35
- (attribute or schema) distinct, 113
- (attribute) translation, 158, 160
- (big) conjunction, 68, 124
- (big) disjunction, 68, 124
- (big) intersection, 84, 117
- (big) opposition, 68
- (big) union, 84, 117
- (binary) (cartesian) product construction, 81
- (binary) (endo-)relation, 38
- (binary) coproduct construction, 81
- (cartesian) product, 73
- (co)projection, 135
- (concrete) (ordinary) structure, 29
- (distributive) bounded quasi-lattice algebra, 41
- (distributive) complete quasi-lattice algebra, 41
- (distributive) quasi-lattice algebra, 41
- (domain) restriction, 27
- (general) infimum elimination, 145
- (general) record class, 73
- (general) record structure, 69
- (general) supremum elimination, 145
- (ordered) pairs, 21
- (ordinary) relation class, 25
- (pairwise) compatible, 57, 113
- (pairwise) distinct, 57, 90, 113, 120
- (pairwise) equi-schematic, 113
- (partial) order, 38
- (partially) ordered class, 38
- (record) partition of, 91
- (schema) compatible, 113
- (singular) infimum elimination, 145
- (singular) supremum elimination, 145
- (small) intersection, 117
- (small) union, 117
- (total) domain, 61
- (total) function, 26
- (untyped) description, 21
- (well-) defined, 19
- (well-) instantiated, 19
- (well-) typed, 19
- 1-carrier syntactic structure, 167
- 2-carrier syntactic structure, 166
- 2-structure syntactic structure, 166
- 3-carrier syntactic structure, 165
- 3-structure syntactic structure, 165

- algebra, 29
- and, 22, 63
- anti-chain, 38
- antisymmetric, 38
- application, 17, 18, 26, 27
- arity, 53
- assignment, 19
- asymmetric, 38
- atom class function, 35
- atoms, 34
- attribute, 99
- attribute class, 53, 56, 87
- attributes, 53

- belongs to, 24
- big conjunctive, 40
- big disjunctive, 40
- big intersection, 24
- big union, 24
- bijection, 26
- bijective, 26
- binary, 53
- bit, 22
- bit value, 102
- bit value relation, 107
- boolean (or bit) value algebra, 29
- boolean lattice, 41
- boolean quasi-lattice, 41
- boolean table (diagram), 103
- boolean value, 22, 102
- bottom, 39, 115
- bounded, 40

- canonic class, 38
- canonic quasi-ordered class, 38

- cardinal numbers, 25
- cardinality, 25, 105
- carrier, 29
- cartesian, 75
- cartesian graph, 88
- cartesian product, 15, 25
- chain, 38
- chapter, 13
- characteristic function, 28, 102
- characters, 25
- Class, 13
- class, 23
- class expression, 20, 23
- class family, 23
- class partition (record), 90
- class partition of, 90
- classes, 20
- closed, 19
- codomain, 20, 26, 28, 53, 56
- cograph, 88
- combination, 30
- compatibility, 57
- compatible domain, 61
- complement, 32, 84, 116
- complement function, 40
- complemented, 40
- complete lattice, 40
- complete partial order, 60
- complete quasi-boolean algebra, 41
- complete quasi-lattice, 40
- completely finite, 55, 73, 103
- components, 54
- composition, 27, 28
- concatenation, 21, 25, 59, 76, 121
- conditional expression, 21
- congruence relation, 44
- conjunction, 22, 124
- conjunctive, 39
- connex, 38
- consequence, 22
- constant, 29
- constant (value), 19
- context, 19
- coproduct, 73, 75
- coprojection, 135, 137
- countable, 25
- cpo, 60

- default, 108
- default n -ary function expression, 14
- default n -ary relation expression, 14
- default coproduct construction, 81
- default product construction, 81
- defined domain, 28
- degenerated, 120
- derived, 22
- difference, 24, 84, 134
- dimension, 53, 105
- disjunctive union, 24, 73, 75
- disjunction, 124
- disjunctive, 39
- distinct, 90, 120
- distinct (cartesian) product, 120
- distinct join, 58, 90, 120
- distinct product, 93, 120
- distinctness, 57
- distributive, 40
- does not hold, 27
- domain, 20, 26-28, 53, 56, 87, 99
- domain form, 102
- domain schema, 90
- double tables, 104

- element, 24
- elementary, 105
- elements, 23
- elimination, 135
- eliminations, 135
- empty, 105, 115, 116
- empty class, 23
- empty function, 27, 54
- empty record, 54
- empty schema, 100
- empty tuple, 21, 27, 54
- endorelation, 105
- entailment, 22, 34
- entails, 22, 34
- enumerable, 25
- environment, 19
- equal, 21
- equality, 38
- equation, 21
- equi-schematic, 113
- equi-schematic relation algebra, 118
- equiatomic relation, 35
- equijunction, 22, 134
- equijunctive, 45
- equivalence, 22, 111, 124
- equivalence class, 43
- equivalence relation, 38, 39
- equivalent, 22, 34
- equivalent reduction, 35, 135, 149
- existentialization, 22

- expanded graph, 88
- expanded record class, 73
- expands, 35
- expansion, 35, 109–111, 122
- expression, 17

- false, 22
- field of rational numbers, 30
- finite, 25, 53, 55, 73, 103
- finite class expression, 23
- finite classes, 24
- finite power algebra, 32
- finite subclass algebra, 32
- finite table, 103
- follows (or derives) from, 22
- formulas, 20
- fractions, 36
- full, 105, 115, 116
- full class, 32
- function class, 25

- general graph, 88
- generalization, 22
- generalized boolean lattice, 41
- generalized boolean quasi-lattice, 41
- generalized quasi-boolean algebra, 42
- graph, 15, 27, 88, 90, 99
- graph form, 102
- graph table (diagram), 100, 103
- greatest, 39
- greatest lower bound, 39, 40

- Hasse diagrams, 39
- hierarchy, 38
- holds, 27
- homomorphism, 30

- identical, 21
- Identifier, 13
- identifier, 17, 18
- identifiers, 53
- identity (relation), 114
- identity function, 27
- identity relation, 38
- if, 22
- iff, 22
- implies, 22
- in, 100, 155
- included schema class, 84
- inclusion, 24, 84, 117
- inclusion algebra, 84
- inclusions, 84
- incompatible domain, 61
- index, 53, 56, 99
- index schema, 90
- indices, 53
- infimum, 40, 68
- infimum elimination, 135
- infimum reduction, 135, 136, 149
- infimum reductor, 35
- injection, 26
- injective, 26
- instantiation, 19
- integer ring, 30
- integers, 25
- intersection, 24, 84
- inverse, 28
- irredundant attribute class, 155
- irredundant for, 155
- irreflexive, 38
- is a table, 103
- is in, 24
- is not in, 24
- isomorph, 30
- isomorphism, 30

- join, 39, 63

- Kleene closure, 25, 79
- Kleene graph, 88
- Kuratowski product construction, 81

- lambda expression, 26
- lambda expressions, 14
- lattice, 40
- least, 39
- least upper bound, 39, 40
- left unique, 158
- length, 21
- linear quasi-order, 38
- linear order, 38
- linearly ordered class, 38
- linearly ordered field, 37
- literal, 53, 105
- local definition, 20
- locally finite, 55
- lower, 145, 173

- map(ping) (expression), 26
- map(ping) expression, 14
- mapping, 14
- maximum, 40
- meet, 39, 63
- meet semi-lattice, 60
- member, 100
- members, 23
- membership, 24
- minimum, 40
- minus, 41

- natural numbers, 25
- negation, 22, 116
- negator, 40
- new word, 17
- not, 22, 116
- not in, 100
- null value, 108
- nullary, 53, 54
- nullary product, 25

- of, 26
- open, 19
- Operation, 13
- operator expression, 20
- operators, 20
- opposed to, 63
- opposition, 24, 63
- or, 22, 63
- order diagrams, 39
- ordinal, 75
- ordinary, 14, 15, 26, 27
- ordinary map(ping) expression, 26
- ordinary predicator (expression), 27
- overloaded, 19

- P-record, 53
- paragraph, 13
- partial function, 28
- partial function class, 25
- partial functions, 28
- partial table, 108
- partition, 38
- pattern matching, 20
- poiclass, 38
- power (product) of, 79
- power algebra, 32
- power class, 24
- power class algebra, 118
- predicator, 14
- predicator (expression), 27
- predicator expression, 14
- predicator form, 102
- product, 25, 75
- projection, 56, 95, 135, 137
- projection class, 56
- projection relation class, 106
- proper, 55, 73, 90–92, 100
- proper incl., 24
- proper subrecord, 57
- property, 25
- propositional formulas, 34

- quadruples, 21
- quasi-algebras, 41
- quasi-boolean algebra, 41
- quasi-boolean lattice, 41
- quasi-chain, 38
- quasi-class, 38
- quasi-graph, 108
- quasi-hierarchy, 38
- quasi-lattice, 40
- quasi-linear order, 38
- quasi-linearly ordered class, 38
- quasi-order, 38, 128
- quasi-order diagrams, 39
- quasi-ordered class, 38, 128
- quasi-ordered field of fractions, 37
- quasi-relation, 108
- quasi-relation graph, 88
- quotient class, 43
- quotient function, 44
- quotient relation, 44
- quotient structure, 44

- range, 101
- rational numbers, 25, 37
- real numbers, 25
- record, 15, 53
- record class, 60
- record of Π , 158
- record partition, 91
- record projection structure, 70
- record table, 73
- record-record, 90
- reduces, 135
- reduction, 135
- redundant, 136, 155
- redundant attribute class, 155
- redundant for, 155
- reflexive, 38
- rejected by, 63
- rejector, 63
- relation, 99, 102, 110
- relation class, 100
- relation graph, 88
- relative complement, 41
- relative complement function, 41
- relatively distinct, 92, 120

restriction, 38
right unique, 158

schema, 15, 54, 73, 87, 99
schema form, 102
schema partition, 91
schema record, 90
schemas, 53, 54
schematic, 15
section, 13
selective syntactic structure, 164
semantic algebra, 163
semantics, 102
set, 23
similar, 30
single-sort projection relation structure, 168
singleton, 25
singleton class, 24
singular, 53, 105
singular (or literal) record class, 73
singular finite, 55
singular graph, 88
smaller, 57
standard form of an ordinal function definition, 26
standard quasi-boolean algebra, 34
star graph, 88
star product, 73, 75
strict smaller, 57
strings, 25
structured class, 38
sub-schematic, 113
subatomic, 35
subclass algebra, 32
subjunction, 22, 134
subjunctive, 45
subrecord, 57
subsection, 13
substitution, 18
substructure, 30
subtraction, 63, 134
subvalence, 22, 34, 124
sum, 25
supremum, 40, 68
supremum elimination, 135
supremum reduction, 135, 136, 149
supremum reductor, 35
surjection, 26
surjective, 26
symbol, 17, 18
symmetric, 38
symmetric difference, 24
syntactic selector, 164
syntax, 102

table class, 103
terms, 20
the, 115
the algebra (on), 118
the equivalence (relation) of, 128
the graph of ξ , 158
then, 22
three-sort projection relation structure, 168
token, 18
top, 39, 115
total, 38, 105
total in i , 105
transitive, 38
translatable, 158, 160
translation, 158
translator class, 161
triples, 21
true, 22
truth, 22
tuple, 15
tuple class, 25, 79
tuples, 53
two-sort projection relation structure, 168
type, 19
type expression, 14, 19, 100
typed description, 21
typed form, 102

unary, 53
union, 24, 84
unit, 39, 115
unit bit, 22
univalent, 54
updated with, 63
updater, 63
upper, 145, 173

value, 19, 56, 68
value class, 53, 56
values, 53
variable (value), 19

without, 41, 63, 145, 173

zero, 39, 115
zero bit, 22