Inductive Datatypes with Laws and Subtyping

—

A Relational Model

Colofon

Inductive Datatypes with Laws and Subtyping

—

A Relational Model

Proefschrift

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof. dr. M. Rem, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op woensdag 13 januari 1999 om 16.00 uur

door

Theodorus Sebastiaan Voermans

geboren te Eindhoven

Dit proefschrift is goedgekeurd door de promotoren:

prof. dr. R.C. Backhouse
en
prof. L.G.L.T. Meertens

Copromotor:
dr. J.C.S.P. van der Woude

Acknowledgements

Many people supported me during the preparation of this thesis. I would like to mention in particular:

- My Ph.D. supervisors Roland Backhouse, Lambert Meertens and Jaap van der Woude, for accepting me as their Ph.D. student, for their support and their patience.
- Graham Hutton and fellow Ph.D. students Henk Doornbos and Paul Hoogendijk, for many inspiring discussions.
- Chritiene Aarts, Netty van Gasteren, Grant Malcolm, Erik Meijer, Frans Rietman and the other members and former members of the Groningen, Eindhoven and Utrecht “mathematics of program construction clubs”.
- The external members of the Ph.D. committee: Oege de Moor and Frans Kruseman Aretz, for their helpful comments.
- Richard Verhoeven, for implementing Math/pad.
- My family and friends, who often wondered whether this thesis would ever be finished, for their support.

Contents

1 Introduction **1**

1.1 Overview . . . 3

2 Preliminaries **5**

2.1 Proof format and notation . . . 5

2.1.1 Proof format . . . 5

2.1.2 Quantifications . . . 6

2.2 Posets and complete lattices . . . 8

2.2.1 Fixed point theorems . . . 10

2.2.2 Galois connections . . . 11

3 The SPEC-calculus **15**

3.1 The lattice layer . . . 16

3.2 The reverse layer . . . 17

3.3 The composition layer . . . 18

3.4 Strengthening the axiomatisation . . . 21

3.4.1 Complements . . . 21

3.4.2 Desargues rule . . . 22

3.4.3 Cone Rule . . . 22

3.4.4 Extensionality . . . 23

3.5 A geometric model . . . 24

3.6 Conclusion . . . 25

4 Algebraic properties of the SPEC-calculus **27**

4.1 Factors . . . 27

4.2 Distribution of composition over cap . . . 30

4.3 Lattices and Galois connections 33
4.4 Representing sets 43
4.5 Extensionality 46
4.6 Classification rules 49

5 Pers as types **51**
5.1 The monotype system 51
5.2 Pers as types 52
5.3 Orderings on pers 53
5.3.1 Pers as sets of classes 53
5.3.2 Pers as quotients 58
5.3.3 Decomposing pers 61
5.3.4 The per lattice 64
5.3.5 Relational type-judgements and domain operators 70
5.4 Pointwise calculations in the SPEC-calculus 77
5.5 Difunctionals 79
5.6 Injectivity, surjectivity and totality 82
5.7 Type-deduction rules 85

6 The category of total difunctionals **87**
6.1 Short introduction to category theory 87
6.2 The category **Difun** 90
6.3 Relators 92
6.4 Natural transformations 101

7 Disjoint sum and Cartesian product relators **105**
7.1 Categorical construction 105
7.2 SPEC construction 107
7.3 Algebraic properties 113

8 Inductive types **119**
8.1 F-algebras 121
8.2 F-inductive algebras 126
8.2.1 Constructors 127

8.2.2 Simulations 129
8.2.3 F-inductive types 131
8.3 Parameterised inductive types 134
8.4 Other recursive programs 137
8.4.1 Injective F-inductive algebras 139
8.5 F-reductivity 141

9 Equational specification of datatypes 143
9.1 Constructing F-inductive algebras 143
9.2 Equations 146
9.2.1 F-inductive closures 151
9.2.2 F-inductive type construction 154
9.2.3 Example: Stacks 156
9.2.4 Solutions for structural induction 156
9.2.5 Parameterised types with equations 157
9.2.6 Example: the Boom-hierarchy 165
9.2.7 Example: Arrays 168
9.3 Transformers 170
9.4 Abstract Datatypes 171
9.5 Conclusion 173

10 Conclusions 175

References 179

Index 184

Samenvatting 189

Curriculum Vitae 193

Chapter 1

Introduction

Inductive datatypes, datatypes where elements of the type occur as “subcomponents” of other elements of the type, are an essential feature of all modern programming languages. Commonly used examples of such types are for example binary trees where, a tree can have other binary trees as subtrees, or cons-lists, where the tail of a cons-list is another cons-list. A standard mathematical method for reasoning about such datatypes and programs operating with these types was developed by Malcolm [42]. He constructed an elegant generic theory of free inductive datatypes using category theory based on the concepts of functors and initial algebras. By generic we mean parameterised by the shape of the datatype.

A limitation of this theory is that it only deals with free datatypes, types without rules specifying equality of elements or restrictions on the construction of elements. In practice there are many common datatypes that are not free. For example, join-lists have associativity laws on the join operation, and height-balanced trees can not be constructed using arbitrary subtrees. Fokkinga [24] extended Malcolm's theory to datatypes with laws, but was not able to handle restrictions on the construction of elements (subtyping).

Other, set-theoretical, theories [43] about inductive datatypes can handle both laws and subtyping but have as disadvantage that they treat laws and subtyping as dual concepts. This complicates reasoning about datatypes that combine both laws and subtyping. An example of a type combining both concepts is the AVL-tree, where different trees can be used to represent the same set of values (law), but where it is not allowed to join two arbitrary AVL-trees to construct a new valid AVL tree (restriction).

The goal of this thesis is to develop a theory about inductive datatypes that can handle laws and subtyping in a uniform way. The theory should predict when (recursively defined) operations are well-defined and when they are uniquely defined. The theory should also provide a sound basis for the construction and verification of generic programs.

The theory of inductive datatypes presented in this thesis was inspired by the category-theoretical approach but uses a point-free relational calculus to model both datatypes and programs. One of the main advantages of using the relational calculus is that it opens up the possibility of working with lattices where extreme solutions to equations are uniquely defined. Category theory always gives solutions “up to isomorphism” that are often less suitable for direct manipulation. The extreme solutions of lattice equations provide unique, canonical representations of the concepts that are being modelled. Datatypes and programs are usually specified as solutions to equations

Another advantage of the lattice structures that are available when working with relations is the abundant possibility for using Galois connections. Identifying Galois connections and using their calculational properties is a recurring theme throughout the thesis. We prefer a calculational style for constructing and presenting proofs and Galois connections are a great tool for this purpose.

We identify a special class of relations that can be used as representatives for datatypes. These datatypes are the elements of a complete lattice where the ordering represents (the combination of) subtyping and quotient formation. Combining these aspects in a single ordering allows us to find solutions for specifications involving both restrictions (subtyping) and laws (quotients). Combining these features is often difficult in other formalisms for datatypes. This ordering is a vital tool for achieving our goal of a uniform treatment of laws and subtyping.

Our datatype construction methods are inspired by categorical datatype theories and we will construct a category where objects and arrows are relations. Categorical notions like functors, natural transformations and F-algebras lead to relational constructions that are useful for the construction of datatypes and programs.

A variant of F-algebras is used for the introduction of inductive datatypes and structural recursion. An important aspect of datatype construction is simulation, implementing one datatype using another datatype. The notion of simulation can easily be formulated in our theory.

Inductive types that simulate each other form equivalence classes. We prove the remarkable result that every equivalence class contains one special representative. The special representatives form a complete lattice, using our special ordering of datatypes. The elements of the lattice represent all inductively defined datatypes for a given induction structure. Using this lattice, we can describe inductive datatypes with both laws and restrictions as an extreme fixpoint. We will give an equivalent characterization of the extreme fixpoint using a Galois connection. This Galois connection, which defines a closure operation, turns out to be very convenient for proving properties of inductive datatypes.

Laws and restrictions can be specified with equations, which can be combined to a single specification of the datatype. Not only are datatypes described as solutions of equations, but recursively defined operations on these inductive datatypes are also specified as solutions of equations. We will show that a large class of “recursion structure” equations for operations on inductive datatypes have at most one solution, so they are suitable as a specification.

Another subject investigated in this thesis is conditions under which parameterisation of inductive datatypes with laws and restrictions is possible. Here we demonstrate that, if the law and restriction equations satisfy certain naturality ("polymorphy") criteria, parameterisation is possible.

1.1 Overview

We continue with a short overview:

Chapter 2 contains some preliminaries for the rest of the thesis. Subjects are our proof-format and some standard results about orderings and lattices. The definition and uses of Galois connections at the end of the chapter are important for understanding the remainder of the thesis.

Chapter 3 gives an axiomatisation of the SPEC-calculus, a point-less relational calculus. Some easy consequences of the axioms are shown and alternatives for strengthening the axiomatisation are also presented.

Chapter 4 explores some important algebraic properties of the SPEC-calculus. Topics include factors (a kind of inverse of relational composition), distribution properties and Galois connections between lattices constructed with special classes of specs. Elements of these special classes can be used to represent subsets of the universe over which a relational calculus is constructed. The connection between the SPEC-calculus and relational calculus with points is studied by considering the consequences of adding an extensionality axiom to the SPEC-calculus.

Chapter 5 has as subject how Partial Equivalence Relations (pers) can be used to represent types and three new orderings between pers are discussed. One ordering corresponds to subset construction, one to quotient construction and the third ordering is a combination of the two previous orderings. The lattice properties of the orderings are studied and we prove that the pers form a complete lattice under the combined order. Functions between types represented by pers are modelled by a special class of relations called the difunctionals and a small calculus of type judgements is developed for typing difunctionals and other relations. Notions like functionality, injectivity, totality and surjectivity can be modelled using the judgements in the calculus.

Chapter 6 introduces some basics concepts from category theory and an interesting category called **Difun**, based on pers and difunctionals, is constructed. We construct operations and notions in the SPEC-calculus corresponding to functors and natural transformations in **Difun**.

Chapter 7 shows the construction of SPEC operations corresponding to Cartesian products and disjoint sum in **Difun**. Some natural transformations with Cartesian products and disjoint sums that will be useful later for program and datatype construction are also given in the chapter.

Chapter 8 studies the construction of inductive datatypes by generalising the standard categorical construction with F-algebras to datatypes with partial and/or non-injective constructors represented by F-inductive algebras. A notion of simulation and isomorphy of F-inductive algebras is developed and classes of isomorphic algebras are studied. It turns out that every isomorphy class can be represented by a special kind of F-inductive algebra called an F-inductive type. The F-inductive types form a lattice under the special ordering on pers developed in chapter 5. Another subject is the construction of recursively defined programs on inductive datatypes and how these programs can be specified as the unique solutions of equations.

Chapter 9 shows methods for the construction of inductive datatypes specified by equations. Two classes of equations are considered. The first class specifies laws on the datatype and examples like associativity, commutativity and unit laws are discussed for an interesting group of datatypes that is called the Boom-hierarchy. The second class of equations specifies restrictions on the arguments of constructors like height-balancedness of trees. An advantage of the theory presented in this thesis is that these two classes of equations can be combined without difficulty. An example of a datatype with both laws and restrictions presented in this chapter is the stack datatype. Another subject of this chapter is the existence of solutions to a class of specifications of recursively defined programs working on datatypes specified with equations. We can give a necessary and sufficient condition for programs performing structural induction on the inductive datatype. The chapter is concluded with a discussion of many-sorted datatypes.

Chapter 10 contains conclusions about the results presented in the thesis.

Chapter 2

Preliminaries

This chapter gives some preliminaries for the rest of the thesis. There is no new theory contained in this chapter and every result can be found in the literature. Readers familiar with the style of the "Eindhoven-school" will recognize most of the notations, definitions and conventions introduced in the first section. The second section contains results about posets and lattices that are used extensively in this thesis. This is also standard theory, but some notations and terminology about Galois connections are not standard. We assume that the reader is familiar with elementary predicate calculus and set theory.

2.1 Proof format and notation

2.1.1 Proof format

The proof format we adopt for this thesis is called "linear proofs" and was designed by Wim Feijen [17]. It consists of formulas connected by transitive relations, with hints why the relation is valid. A typical proof looks like:

$$
\begin{array}{ll}
 & P \\
\trianglelefteq & \qquad \{ \text{ Hint why } P \trianglelefteq Q \ \} \\
 & Q \\
\trianglelefteq & \qquad \{ \text{ Hint why } Q \trianglelefteq R \ \} \\
 & R
\end{array}
$$

Here we have proved that $P \trianglelefteq R$. Extra assumptions can be introduced by prefixing them with $\bullet$ in the hint. For example:

$$
\begin{array}{ll}
 & P \\
\trianglelefteq & \qquad \{ \ \bullet T; \text{ Hint why } P \trianglelefteq Q, \text{ assuming } T \ \} \\
 & Q
\end{array}
$$

$\trianglelefteq$ { Hint why $Q \trianglelefteq R$ }
R

Here we have proved that $P \trianglelefteq R \Leftarrow T$.

An often-used form of linear proof is called *cyclic inclusion*. In this case we have the same first and last formula and we have proved, if the connecting relation is anti-symmetric, that all formulas are equal to each other. The version of cyclic inclusion with as ordering the implication of boolean expressions is known as *cyclic implication*.

2.1.2 Quantifications

Quantifications play an important role in our calculations. All quantifications are written using the same standard notation:

$\mathbf{Q}(\mathit{dummies} \mid \mathit{predicate} \mid \mathit{range})$

Here $\mathbf{Q}$ is called the quantifier and determines the operation that is applied to the bag that is defined by $(\mathit{dummies} \mid \mathit{predicate} \mid \mathit{range})$. Examples are $\exists$ with operation $\vee$, $\forall$ with operation $\wedge$ and $\sum$ with operation $+$. For the quantification to be well-defined it is necessary that the operation be symmetric, associative and total on the elements of the *range* part. This is not always sufficient for quantifications over infinite bags. For example, taking the sum over all natural numbers is not well-defined.

The item *dummies* is a (possibly empty) list of variable names, separated by commas. They act as bound variables in the predicate and the range.

The *predicate* determines which values for the dummies are admissible in the *range*. The bag consists of all values of the range expression that are permitted by the predicate as instantiation for the dummies.

Example: $\sum(x \mid x \in \mathit{Int} \wedge -1 \leq x < 3 \mid x^2) = 1 + 0 + 1 + 4 = 6$

Dummy x can take values -1, 0, 1 and 2; so the bag consists of the values $(-1)^2$, 0^2, 1^2 and 2^2 and the value of the quantification is the sum, 6.

The empty bag will be denoted by $\emptyset$. A quantification over $\emptyset$ has as result the unit of the operation of the quantification (if it exists). So $\sum \emptyset = 0$, $\forall \emptyset = \mathit{true}$ and $\exists \emptyset = \mathit{false}$ etc.

For quantifications without dummies we define:

$\mathbf{Q}(\mid \mathit{true} \mid R) = R$
$\mathbf{Q}(\mid \mathit{false} \mid R) = \mathbf{Q}\emptyset$

The first one corresponds to a quantification over a one-element bag.

If the type of the dummies is clear from the context and there are no further restrictions on them then we will often omit the predicate and simply write $\mathbf{Q}(d \mid\mid R)$

Some often-used elementary rules for the quantifier calculus are given below. In the following formulas d, $d1$ and $d2$ stand for a (possibly empty) list of dummies, x for a single dummy and e, $e1$ and $e2$ for an expression. The operation corresponding to quantifier $\mathbf{Q}$ is $\otimes$.

Rule 2.1: (Range-Split)

$$\mathbf{Q}(d \mathrel{¦} P \mathrel{¦} R) \;=\; \mathbf{Q}(d \mathrel{¦} P \wedge S \mathrel{¦} R) \otimes \mathbf{Q}(d \mathrel{¦} P \wedge \neg S \mathrel{¦} R)$$

□

Rule 2.2: (Nesting)

$$\mathbf{Q}(d1 \mathrel{¦} P \mathrel{¦} \mathbf{Q}(d2 \mathrel{¦} R \mathrel{¦} S))$$
$$=$$
$$\mathbf{Q}(d1 \mathrel{¦¦} \mathbf{Q}(d2 \mathrel{¦} P \wedge R \mathrel{¦} S))$$
$$=$$
$$\mathbf{Q}(d1, d2 \mathrel{¦} P \wedge R \mathrel{¦} S)$$

The dummies $d2$ may not occur (free) in $d1$ or P.
□

In the following two rules $[x := e]P$ stands for P with all free occurrences of x replaced by e.

Rule 2.3: (One-Point Rule)

$$\begin{array}{ll} & \mathbf{Q}(d, x \mathrel{¦} P \mathrel{¦} R) \\ = & \qquad \{\ \bullet\ \forall(d, x \mathrel{¦} P \mathrel{¦} x = e),\ x \text{ not free in } e\ \ \} \\ & \mathbf{Q}(d \mathrel{¦} [x := e]P \mathrel{¦} [x := e]R) \end{array}$$

□

This rule is often used for the elimination of quantifications. In that case d is empty and the last formula is almost always immediately rewritten to $[x := e]R$ or $\mathbf{Q}\emptyset$ (if allowed). The rule may, of course, also be used in reverse order for the introduction of quantifications.

Rule 2.4: (Leibniz, Substitution)

$$\begin{array}{ll} & \mathbf{Q}(d \mathrel{¦} P \mathrel{¦} [x := e1]R) \\ = & \qquad \{\ \bullet\ \forall(d \mathrel{¦} P \mathrel{¦} e1 = e2),\ x \text{ not free in } e1 \text{ or } e2\ \ \} \\ & \mathbf{Q}(d \mathrel{¦} P \mathrel{¦} [x := e2]R) \end{array}$$

□

This rule is the bag equivalent of the Leibniz rule for functions.

The following rule is only valid for idempotent operation $\otimes$:

Rule 2.5: (Generalised Range-Disjunction)

$$\begin{array}{ll} & \mathbf{Q}(d1 \mathrel{¦} P \mathrel{¦} R) \\ = & \qquad \{\ \bullet\ \forall(d1 \mathrel{¦} P \mathrel{¦} \exists(d2 \mathrel{¦¦} S)),\ d2 \text{ not free in } d1,\ P \text{ or } R\ \ \} \\ & \mathbf{Q}(d1, d2 \mathrel{¦} P \wedge S \mathrel{¦} R) \end{array}$$

□

This rule is used for the introduction of new dummies in quantifications. It is usually followed by some applications of the one-point rule to remove old dummies.

A special form of quantification is set formation. It is defined by:

Definition 2.6: (Set Formation)

$$\{d \mid P \mid R\} = \bigcup(d \mid P \mid \{R\})$$

□

Another quantification-like notation is

$$\exists!(d \mid P \mid R)$$

This is not really a quantification, because we do not have a corresponding $\otimes$ operator. Its value is true if there exist unique values for the dummies such that both P and R are true and otherwise it is false. The $\exists!$-quantification does have all properties of normal quantifications mentioned above except for range-split and generalised range disjunction.

Our format for writing function spaces differs from the conventional method in that we write the range on the lhs and the domain on the rhs, reversing the direction of the arrow. Writing $f \in \mathcal{B} \longleftarrow \mathcal{A}$ means that f is a total function, mapping elements from $\mathcal{A}$ to elements of $\mathcal{B}$. One of the advantages of this convention is that determining the type of the composition of several functions is easier than with the usual convention.

We write function composition with a raised infix dot and function application with a lowered infix dot. The relation between composition and application is the usual $(f \cdot g).x = f.(g.x)$.

A convention that we sometimes use is overloading function application to subsets of the domain, for $f \in \mathcal{B} \longleftarrow \mathcal{A}$ and $\mathcal{A} \supseteq \mathcal{C}$ we write $f.\mathcal{C}$ for $\{x \mid x \in \mathcal{C} \mid f.x\}$. This is only used in situations where it is clear from the context which form of application is meant.

The priority of all pre- and postfix operators is equal and higher than the priority of any infix operator. Function application is the infix operator with the highest priority.

2.2 Posets and complete lattices

Orderings form an essential part of the theory that is presented in this thesis. This section introduces some basic notions about orderings, like partial orders, posets, complete lattices and Galois connections together with some elementary but often used theorems. For more background see e.g. Davey and Priestley [16] or Birkhoff [13].

We start by giving the definition of a poset:

Definition 2.7: (Poset)

For set $\mathcal{A}$ and binary relation $\preceq$ on $\mathcal{A}$, $(\mathcal{A}, \preceq)$ is a *poset* (partially ordered set) iff for all $x, y, z \in \mathcal{A}$:

(a) $x \preceq x$ (Reflexivity)
(b) $x \preceq y \wedge y \preceq z \;\Rightarrow\; x \preceq z$ (Transitivity)
(c) $x \preceq y \wedge y \preceq x \;\Rightarrow\; x = y$ (Anti-symmetry)

□

Given a poset we can construct new posets with functions to the original poset as elements and the "lifted" version of the original ordering as ordering:

Lemma 2.8: (Lifting)

For poset $(\mathcal{A}, \preceq)$ and set $\mathcal{B}$, $(\mathcal{A} \longleftarrow \mathcal{B}, \dot{\preceq})$ with $f \mathrel{\dot{\preceq}} g \triangleq \forall(x \mid x \in \mathcal{B} \mid f.x \preceq g.x)$ is also a poset.

□

An often-used method for proving the equality of two elements of a poset is formulated in the next lemma:

Lemma 2.9: (Indirect equality)

For poset $(\mathcal{A}, \preceq)$ and $X, Y \in \mathcal{A}$:

$$
\begin{array}{ll}
 & \forall(Z \mid Z \in \mathcal{A} \mid Z \preceq X \;\equiv\; Z \preceq Y) \\
\equiv & \\
 & X = Y \\
\equiv & \\
 & \forall(Z \mid Z \in \mathcal{A} \mid X \preceq Z \;\equiv\; Y \preceq Z)
\end{array}
$$

□

We continue with a list of definitions:

Definition 2.10: (Monotonicity)

For posets $(\mathcal{A}, \preceq)$ and $(\mathcal{B}, \sqsubseteq)$ and function $f \in \mathcal{B} \longleftarrow \mathcal{A}$, f is *monotonic* iff for all $x, y \in \mathcal{A}$:

$$x \preceq y \;\Rightarrow\; f.x \sqsubseteq f.y$$

Function f is *anti-monotonic* iff for all $x, y \in \mathcal{A}$:

$$x \preceq y \;\Rightarrow\; f.y \sqsubseteq f.x$$

□

Note that an anti-monotonic function from $(\mathcal{A}, \preceq)$ to $(\mathcal{B}, \sqsubseteq)$ is a monotonic function from $(\mathcal{A}, \preceq)$ to $(\mathcal{B}, \sqsupseteq)$. We use the notational convention that the mirrored version of an ordering denotes the reversed order, i.e. $x \succeq y \equiv y \preceq x$ and $x \sqsupseteq y \equiv y \sqsubseteq x$.

Definition 2.11: (Least Upper Bound)

For poset $(\mathcal{A}, \preceq)$ and $\mathcal{B} \subseteq \mathcal{A}$, $x \in \mathcal{A}$ is the *least upper bound* (lub) of $\mathcal{B}$ iff for all $y \in \mathcal{A}$:

$$\forall(z \mid z \in \mathcal{B} \mid z \preceq y) \;\equiv\; x \preceq y$$

□

Definition 2.12: (Greatest Lower Bound)

For poset $(\mathcal{A}, \preceq)$ and $\mathcal{B} \subseteq \mathcal{A}$, $x \in \mathcal{A}$ is the *greatest lower bound* (glb) of $\mathcal{B}$ iff for all $y \in \mathcal{A}$:

$$\forall(z \mid z \in \mathcal{B} \mid y \preceq z) \quad \equiv \quad y \preceq x$$

□

The uniqueness of these bounds is easy to prove. Existence is, however, not guaranteed.

Definition 2.13: (Bottom and Top)

For poset $(\mathcal{A}, \preceq)$, the *top* is defined as the least upper bound of $\mathcal{A}$ (if it exists) and the *bottom* is defined as the greatest lower bound of $\mathcal{A}$ (if it exists).
□

This thesis uses only one special kind of lattice, the complete lattice, so we omit the more general definitions of a lattice, semi-lattice etc.

Definition 2.14: (Complete lattice)

Poset $(\mathcal{A}, \preceq)$ is a *complete lattice* iff least upper bounds and greatest lower bounds exist for all subsets of $\mathcal{A}$
□

For proving that a poset is a complete lattice it is sufficient to show that just one of the two kinds of bounds exists as stated in the following lemma:

Lemma 2.15: (Complete lattice)

For poset $(\mathcal{A}, \preceq)$,

Greatest lower bounds exist for all subsets of $\mathcal{A}$

$\equiv$

Least upper bounds exist for all subsets of $\mathcal{A}$

□

2.2.1 Fixed point theorems

Extreme solutions to equations are often used as a means to define new notions and many of the properties of those notions follow from being an extreme solution. The most important theorem about fixed points for this thesis is the Knaster-Tarski theorem:

Theorem 2.16: (Knaster-Tarski) For complete lattice $(\mathcal{A}, \preceq)$ and monotonic function $f \in \mathcal{A} \leftarrow \mathcal{A}$, the equations

$$x = f.x$$
$$f.x \preceq x$$

have the same least solution denoted by μf. It is completely characterised by the following equations:

$$\begin{array}{lll} \text{(a)} & f.\mu f = \mu f & \text{(Calculation rule)} \\ \text{(b)} & \mu f \preceq x \;\Leftarrow\; f.x \preceq x, \text{ for all } x \in \mathcal{A} & \text{(Induction)} \end{array}$$

□

There is also the dual theorem for greatest solutions:

Theorem 2.17:(Knaster-Tarski) For complete lattice $(\mathcal{A}, \preceq)$ and monotonic function $f \in \mathcal{A} \leftarrow\!\dashv \mathcal{A}$, the equations

$$\begin{array}{l} x = f.x \\ x \preceq f.x \end{array}$$

have the same greatest solution denoted by νf. It is completely characterised by the following equations:

$$\begin{array}{lll} \text{(a)} & f.\nu f = \nu f & \text{(Calculation rule)} \\ \text{(b)} & x \preceq \nu f \;\Leftarrow\; x \preceq f.x, \text{ for all } x \in \mathcal{A} & \text{(Induction)} \end{array}$$

□

The original Knaster-Tarski theorem is somewhat stronger, stating that there is a complete lattice of fixed points, but the version given here contains all that is needed for this thesis. The rolling rule is a "folk theorem" from lattice theory that seems to be missing in the standard literature (although it is sometimes given as an exercise). It turns out to be quite useful for us:

Lemma 2.18:(Rolling)

For complete lattices $(\mathcal{A}, \preceq)$ and $(\mathcal{B}, \sqsubseteq)$ and monotonic functions $f \in \mathcal{B} \leftarrow\!\dashv \mathcal{A}$ and $g \in \mathcal{A} \leftarrow\!\dashv \mathcal{B}$:

$$\begin{array}{llll} \text{(a)} & \mu(f \cdot g) & = & f.\mu(g \cdot f) \\ \text{(b)} & \nu(f \cdot g) & = & f.\nu(g \cdot f) \end{array}$$

□

2.2.2 Galois connections

The use of Galois connections is another powerful method for defining new notions in a calculus with orders. It turns out that many definitions in this thesis can be expressed very compactly using Galois connections. We start by giving the definition:

Definition 2.19:(Galois Connection)

For posets $(\mathcal{A}, \preceq)$ and $(\mathcal{B}, \sqsubseteq)$ and functions $f \in \mathcal{B} \leftarrow\!\dashv \mathcal{A}$ and $g \in \mathcal{A} \leftarrow\!\dashv \mathcal{B}$, we say that (f, g) is a *Galois connection* iff for all $x \in \mathcal{A}$ and $y \in \mathcal{B}$ we have:

$$f.x \sqsubseteq y \;\equiv\; x \preceq g.y$$

□

We also use the terminology that f and g are Galois *adjoints*, where f is called the *lower adjoint* and g is called the *upper adjoint*. The definition given here is somewhat more restrictive than necessary, the poset requirement normally being weakened to pre-order. Some properties of Galois connections between posets are given in the following three lemmas:

Lemma 2.20: (Cancellation)

Given a Galois connection as defined in (2.19), we have:

$$\begin{array}{lrcl} \text{(a)} & x & \preceq & g.(f.x) \\ \text{(b)} & f.(g.y) & \sqsubseteq & y \\ \text{(c)} & f.x & = & f.(g.(f.x)) \\ \text{(d)} & g.(f.(g.y)) & = & g.y \end{array}$$

□

The functions in a Galois connection are monotonic:

Lemma 2.21: (Monotonicity)

Given a Galois connection as defined in (2.19), we have, for all $w \in \mathcal{A}$ and $z \in \mathcal{B}$:

$$\begin{array}{lrcl} \text{(a)} & x \preceq w & \Rightarrow & f.x \sqsubseteq f.w \\ \text{(b)} & z \sqsubseteq y & \Rightarrow & g.z \preceq g.y \end{array}$$

□

Galois adjoints are uniquely defined by one of their components: given the lower adjoint there is at most one upper adjoint and vice versa:

Lemma 2.22: (Unicity)

For posets $(\mathcal{A}, \preceq)$ and $(\mathcal{B}, \sqsubseteq)$ and functions $f \in \mathcal{B} \leftarrow \mathcal{A}$ and $g \in \mathcal{A} \leftarrow \mathcal{B}$,

(a) f has at most one upper adjoint.
(b) g has at most one lower adjoint.

□

We denote the upper adjoint of f by $f^\sharp$ (if it exists) and the lower adjoint of g by $g^\flat$ (if it exists). Note that $\sharp$ and $\flat$ are functions that are dependent on the posets $(\mathcal{A}, \preceq)$ and $(\mathcal{B}, \sqsubseteq)$.

We can say much more if we are working with complete lattices instead of posets. We use $\bigvee$ as the quantifier for the least upper bound and $\bigwedge$ as the quantifier for the greatest lower bound of complete lattice $(\mathcal{A}, \preceq)$; We use $\bigsqcup$ for the lub (least upper bound) and $\bigsqcap$ for the glb (greatest lower bound) of complete lattice $(\mathcal{B}, \sqsubseteq)$. The existence of Galois adjoints is closely connected to distributivity properties over the bounds of the lattices:

Lemma 2.23: (Distributivity)

For complete lattices $(\mathcal{A}, \preceq)$ and $(\mathcal{B}, \sqsubseteq)$ and function $f \in \mathcal{B} \leftarrow \mathcal{A}$

$$\begin{array}{ll} & f^\sharp \text{ exists} \\ \equiv & \\ & f.\bigvee\mathcal{C} = \bigsqcup(f.\mathcal{C}), \text{ for all subsets } \mathcal{C} \text{ of } \mathcal{A} \end{array}$$

□

Lemma 2.24: (Distributivity)

For complete lattices $(\mathcal{A}, \preceq)$ and $(\mathcal{B}, \sqsubseteq)$ and function $g \in \mathcal{A} \leftarrow \mathcal{B}$

$$\begin{array}{ll} & g^\flat \text{ exists} \\ \equiv & \\ & \bigwedge(g.\mathcal{C}) = g.\bigsqcap\mathcal{C}, \text{ for all subsets } \mathcal{C} \text{ of } \mathcal{B} \end{array}$$

□

It is also possible to express $f^\sharp$ in terms of f and $g^\flat$ in terms of g:

Lemma 2.25:

For complete lattices $(\mathcal{A}, \preceq)$ and $(\mathcal{B}, \sqsubseteq)$, functions $f \in \mathcal{B} \leftarrow \mathcal{A}$ and $g \in \mathcal{A} \leftarrow \mathcal{B}$ and for all $x \in \mathcal{A}$ and all $y \in \mathcal{B}$:

$$\begin{array}{ll} & (f, g) \text{ is a Galois connection} \\ \Rightarrow & \\ & g.y = \bigvee(z \mid f.z \sqsubseteq y \mid z) \;\wedge\; f.x = \bigsqcap(z \mid x \preceq g.z \mid z) \end{array}$$

□

Galois connections can be used for the construction of a complete lattice from a poset. The construction of a Galois connection between the poset and a complete lattice gives us a complete lattice structure for (part of) the poset:

Lemma 2.26:

For poset $(\mathcal{A}, \preceq)$, complete lattice $(\mathcal{B}, \sqsubseteq)$, functions $f \in \mathcal{B} \leftarrow \mathcal{A}$ and $g \in \mathcal{A} \leftarrow \mathcal{B}$:

$$\begin{array}{ll} & (f, g) \text{ is a Galois connection} \\ \Rightarrow & \\ & (g.\mathcal{B}, \preceq) \text{ is a complete lattice} \end{array}$$

□

We end this chapter with a very important lemma combining fixed points with Galois connections: fusion. The conditions of the lemma as they are given here are stronger than necessary, but weakening would introduce continuity conditions that we want to avoid here.

Lemma 2.27: (Fusion)

For complete lattices $(\mathcal{A}, \preceq)$ and $(\mathcal{B}, \sqsubseteq)$, monotonic functions $f \in \mathcal{A} \leftarrow \mathcal{A}$, $g \in \mathcal{B} \leftarrow \mathcal{B}$ and $h \in \mathcal{A} \leftarrow \mathcal{B}$:

$$\begin{array}{llcll} \text{(a)} & \mu f \preceq h.\mu g & \Leftarrow & f \cdot h \mathrel{\dot{\preceq}} h \cdot g & (\mu\text{-fusion}) \\ \text{(b)} & \mu f \succeq h.\mu g & \Leftarrow & f \cdot h \mathrel{\dot{\succeq}} h \cdot g \wedge h^\sharp \text{ exists} & (\mu\text{-fusion}) \\ \text{(c)} & \mu f = h.\mu g & \Leftarrow & f \cdot h = h \cdot g \wedge h^\sharp \text{ exists} & (\mu\text{-fusion}) \\ \text{(d)} & h.\nu g \preceq \nu f & \Leftarrow & h \cdot g \mathrel{\dot{\preceq}} f \cdot h & (\nu\text{-fusion}) \\ \text{(e)} & h.\nu g \succeq \nu f & \Leftarrow & h \cdot g \mathrel{\dot{\succeq}} f \cdot h \wedge h^\flat \text{ exists} & (\nu\text{-fusion}) \\ \text{(f)} & h.\nu g = \nu f & \Leftarrow & h \cdot g = f \cdot h \wedge h^\flat \text{ exists} & (\nu\text{-fusion}) \end{array}$$

□

Chapter 3

The SPEC-calculus

The SPEC-calculus is an (incomplete) point-free axiomatisation of the set-theoretic calculus of relations over a fixed set (universe). It is the basic calculational framework for this thesis. The calculus is presented using a layered approach. Every layer is introduced separately after which the interfaces with the previous layers are given. For every layer or interface we also give some often-used lemmas that follow from them. The axiomatisation is followed by a section describing possible strengthenings of the axiom system and the chapter finishes with an interesting non-relational model. The axiomatisation presented here was developed by the research group lead by R.C. Backhouse at Groningen University and Eindhoven University.

Axiomatisation of the relational calculus is an old subject in mathematics, it having been studied in the 19th century by, most prominently, De Morgan and Schröder. More important for us is the work of Tarski [56]. An overview of the history of the relational calculus can be found in Maddux [40]. More recent work is the generalisation of a relational calculus to an allegory (Freyd and Ščedrov [26]). Our axiom set is weaker than the classical axiomatisations because we do not assume the existence of complements, but is stronger than an allegory. A SPEC-calculus is a special case of a one-object allegory. An important new aspect of this axiom set is the frequent use of Galois connections allowing a very concise formulation of properties of the relational calculus that is amenable to formal manipulation. It turned out during the development of this calculus that the choice of axioms is very important for concise and easy-to-find proofs.

Definition 3.1: (SPEC-calculus)

A SPEC-calculus is a 5-tuple $(\mathcal{S}, \sqsubseteq, {\cup}, \circ, I)$ where $\mathcal{S}$ is a set, $\sqsubseteq$ is an ordering on $\mathcal{S}$, $\cup$ is a total endofunction on $\mathcal{S}$, $\circ$ is a total binary endofunction on $\mathcal{S}$ and $I \in \mathcal{S}$. It satisfies axioms (3.2), (3.3), (3.5), (3.10), (3.11), (3.14) and (3.16) given below.
□

The axioms for a SPEC-calculus are given in forthcoming sections of this chapter. These axioms have as one of their models the relations over some set $\mathcal{A}$ when we instantiate:

$$\begin{array}{rcll} \mathcal{S} & \Leftrightarrow & \mathcal{P}(\mathcal{A}\times\mathcal{A}) & \\ R \sqsubseteq S & \Leftrightarrow & R \subseteq S & \text{(relational inclusion)} \\ R\cup & \Leftrightarrow & \{x,y \mid (x,y)\in R \mid (y,x)\} & \text{(relational inverse)} \\ R\circ S & \Leftrightarrow & \{x,y,z \mid (x,y)\in R \ \wedge\ (y,z)\in S \mid (x,z)\} & \text{(relational composition)} \\ I & \Leftrightarrow & \{x \mid x\in\mathcal{A} \mid (x,x)\} & \text{(diagonal relation)} \end{array}$$

We will refer to this model as the *set-theoretic model* of a SPEC-calculus. The term *spec* is used for elements of $\mathcal{S}$ to avoid confusion with the relations from the set-theoretic model. Specs will be denoted by capital letters.

3.1 The lattice layer

The first layer in the axiomatisation imposes a lattice structure on the specs. The set-theoretic relations on universe $\mathcal{A}$ are the subsets of $\mathcal{A}\times\mathcal{A}$. This subset structure is axiomatised in the lattice layer.

Axiom 3.2: (SPEC-Lattice)

$(\mathcal{S},\sqsubseteq)$ is a complete lattice.
□

We denote the least upper bound with the $\bigsqcup$ quantification, greatest lower bound with the $\bigsqcap$ quantification, the lub of two specs R and S by $R \sqcup S$ and the glb by $R \sqcap S$, the bottom of the lattice by $\perp\!\!\perp$, the top by $\top\!\!\top$ and the reverse ordering by $\sqsupseteq$. The infix operators $\sqcap$ and $\sqcup$ have the same priority, lower than the priority of $\circ$. The lub operation is pronounced as cup and the glb operation as cap.

The correspondence with the set-theoretic model is as follows (with $\mathcal{T}$ a set of relations or specs):

$$\begin{array}{rcll} \bigsqcup\mathcal{T} & \Leftrightarrow & \bigcup\mathcal{T} & \text{(relational union)} \\ \bigsqcap\mathcal{T} & \Leftrightarrow & \bigcap\mathcal{T} & \text{(relational intersection)} \\ R\sqcup S & \Leftrightarrow & R\cup S & \text{(relational union)} \\ R\sqcap S & \Leftrightarrow & R\cap S & \text{(relational intersection)} \\ \perp\!\!\perp & \Leftrightarrow & \emptyset & \text{(empty relation)} \\ \top\!\!\top & \Leftrightarrow & \mathcal{A}\times\mathcal{A} & \text{(full relation)} \\ R\sqsubseteq S & \Leftrightarrow & R\subseteq S & \text{(relational inclusion)} \end{array}$$

Another aspect of the subset structure of the lattice of relations is the distribution of $\cup$ over $\cap$ and vice versa. These distribution properties do not follow automatically from the complete lattice structure, an extra axiom being required for this in the SPEC-calculus:

Axiom 3.3: (Distributivity)

For all specs R, $(R\sqcap)^\sharp$ and $(R\sqcup)^\flat$ exist.
□

The distributivity properties following from this axiom are used often but the adjoints themselves are not. This is the reason why we do not introduce a separate notation for the adjoints. The symmetry of the $\sqcup$ and $\sqcap$ gives us also the existence of the adjoints of the other sections: $(\sqcap R)^\sharp = (R\sqcap)^\sharp$ and $(\sqcup R)^\flat = (R\sqcup)^\flat$. We end this section with a corollary stating the axioms the way they are usually used in proofs:

Corollary 3.4: (Lattice layer)

For specs X and bags of specs $\mathcal{T}$,

$$
\begin{array}{llcll}
\text{(a)} & X \sqsupseteq \bigsqcup\mathcal{T} & \equiv & \forall(Z : Z \in \mathcal{T} : X \sqsupseteq Z) & \text{(Least Upper Bound)} \\
\text{(b)} & \bigsqcap\mathcal{T} \sqsupseteq X & \equiv & \forall(Z : Z \in \mathcal{T} : Z \sqsupseteq X) & \text{(Greatest Lower Bound)} \\
\text{(c)} & X & \sqsubseteq & \top\!\!\top & \text{(Top)} \\
\text{(d)} & \perp\!\!\perp & \sqsubseteq & X & \text{(Bottom)} \\
\text{(e)} & X \sqcap \bigsqcup\mathcal{T} & = & \bigsqcup(Z : Z \in \mathcal{T} : X \sqcap Z) & \text{(Distributivity)} \\
\text{(f)} & X \sqcup \bigsqcap\mathcal{T} & = & \bigsqcap(Z : Z \in \mathcal{T} : X \sqcup Z) & \text{(Distributivity)}
\end{array}
$$

□

3.2 The reverse layer

Set-theoretic relations are sets of pairs and the reverse layer axiomatises the fact that elements in a pair can be swapped. The postfix operator $^\cup$ corresponds with swapping all the pairs in a relation. We formulate the defining axiom with a Galois connection:

Axiom 3.5: (Reverse)

For all specs X and Y we have:

$$X^\cup \sqsubseteq Y \;\equiv\; X \sqsubseteq Y^\cup$$

□

We can instantiate the lemmas about Galois connections for this axiom. This yields the following properties: (Parts (a) and (b) need a little calculation. Note that (b) is an equivalence.)

Corollary 3.6: (Reverse)

For specs X and Y and bag of specs $\mathcal{T}$,

$$
\begin{array}{llcll}
\text{(a)} & X^{\cup\cup} & = & X & \text{(Cancellation)} \\
\text{(b)} & X^\cup \sqsupseteq Y^\cup & \equiv & X \sqsupseteq Y & \text{(Monotonicity)} \\
\text{(c)} & (\bigsqcup\mathcal{T})^\cup & = & \bigsqcup(Z : Z \in \mathcal{T} : Z^\cup) & \text{(Distributivity)} \\
\text{(d)} & (\bigsqcap\mathcal{T})^\cup & = & \bigsqcap(Z : Z \in \mathcal{T} : Z^\cup) & \text{(Distributivity)}
\end{array}
$$

□

A set-theoretic relation R is symmetric iff for x and y: $(x,y) \in R \equiv (y,x) \in R$. This notion can be expressed in the SPEC-calculus using the $^\cup$ operator:

Definition 3.7: (Symmetry)

Spec X is *symmetric* iff $X^\cup = X$.

□

We will sometimes have to prove that a spec is symmetric. This can be done by mutual inclusion, but proving one inclusion is sufficient:

Lemma 3.8: (Symmetry)

$$symmetric.X \;\equiv\; X^{\cup} \sqsubseteq X \;\vee\; X \sqsubseteq X^{\cup}$$

□

Proof:

$$\begin{array}{ll}
 & X^{\cup} = X \\
\equiv & \quad \{ \text{ anti-symmetry } \} \\
 & X^{\cup} \sqsubseteq X \;\wedge\; X \sqsubseteq X^{\cup} \\
\equiv & \quad \{ \text{ axiom 3.5 } \} \\
 & X^{\cup} \sqsubseteq X \;\vee\; X \sqsubseteq X^{\cup}
\end{array}$$

□

Instantiating this lemma with $\top\!\!\top$ and $\perp\!\!\perp$ results in:

Corollary 3.9: (Symmetry)

$$\begin{array}{llll}
\text{(a)} & \top\!\!\top^{\cup} = \top\!\!\top & \text{(Reverse)} \\
\text{(b)} & \perp\!\!\perp^{\cup} = \perp\!\!\perp & \text{(Reverse)}
\end{array}$$

□

3.3 The composition layer

The composition layer axiomatises the relational composition operation. Other relational calculi denote composition by juxtaposition, or use a semicolon as operator. We use the symbol $\circ$ because we view function composition as a special case of relational composition and want "functional" relational expressions to have the same appearance as their functional counterparts.

The "normal" method for writing a function as a relation is using the graph of the function defined by $graph(f) = \{x, y \mid y = f.x \mid (x, y)\}$. This has as disadvantage that functional and relational composition do not correspond to each other. Functional composition does correspond to relational composition in the set-theoretic model if we represent function f as the relation $\{x, y \mid y = f.x \mid (y, x)\}$:

$$\begin{array}{ll}
 & \{x, y \mid y = f.x \mid (y, x)\} \circ \{x, y \mid y = g.x \mid (y, x)\} \\
= & \quad \{ \text{ relational composition } \} \\
 & \{x, y, z \mid x = f.y \;\wedge\; y = g.z \mid (x, z)\} \\
= & \quad \{ \text{ one-point rule } \} \\
 & \{x, z \mid x = f.(g.z) \mid (x, z)\} \\
= & \quad \{ \text{ function composition } \} \\
 & \{x, z \mid x = (f \cdot g).z \mid (x, z)\}
\end{array}$$

□

This way of viewing functions is consistent with using the $\longleftarrow\!\!\!\!\dashv$ for function spaces and writing application with the function on the lhs and argument on the rhs. Function application becomes an instance of relational composition if we represent the argument as a singleton subset of the diagonal relation; the lhs of the resulting singleton relation is the function result.

The axiomatic structure of composition contains four axioms. There is one axiom about composition on its own, then two axioms for relating composition with the lattice and reverse structure, respectively. Finally there is an axiom relating all three layers of the axiomatisation.

The first composition axiom states properties of $\circ$ on its own:

Axiom 3.10: (Monoid structure)

$(\mathcal{S}, \circ, I)$ is a monoid. I.e., for all specs X,Y and Z:

$$\begin{array}{lrcccl} \text{(a)} & (X \circ Y) \circ Z & = & X \circ (Y \circ Z) & & \text{(Associativity)} \\ \text{(b)} & X \circ I & = & X & = I \circ X & \text{(Unit)} \end{array}$$

□

The next axiom relates composition to the lattice structure:

Axiom 3.11: (Factors)

For all specs X, $(X\circ)^\sharp$ and $(\circ X)^\sharp$ exist.

□

The importance of the adjoints of the sectioned compositions can be seen in the multitude of names and (re)discoveries of them in the literature. The oldest reference that we know about is by Dilworth [18] where they are called residuals. We use as name for the adjoints the factors as given by Conway [15] and use as notation the division notation from Hoare and He [33], but interchanging the left and right factor notation to make applicability of the cancellation laws easier to spot. The definition becomes:

Definition 3.12: (Factors)

For all specs X, Y and Z, the operations $\backslash$ and $/$ are defined by:

$$\begin{array}{lrclr} \text{(a)} & X \circ Y \sqsubseteq Z & \equiv & Y \sqsubseteq X\backslash Z & \text{(Right factor)} \\ \text{(b)} & Y \circ X \sqsubseteq Z & \equiv & Y \sqsubseteq Z/X & \text{(Left factor)} \end{array}$$

□

The infix operators $\backslash$ and $/$ have a higher priority than $\circ$. The interpretation of the factors in the set-theoretic model is the following:

$$\begin{array}{lcl} R\backslash S & \Leftrightarrow & \{x, y \mid \forall(z \mid (z, x) \in R \mid (z, y) \in S) \mid (x, y)\} \\ S/R & \Leftrightarrow & \{x, y \mid \forall(z \mid (y, z) \in R \mid (x, z) \in S) \mid (x, y)\} \end{array}$$

Instantiation of the properties of Galois connections yields the following corollary for the left factor (the right factor has dual properties):

Corollary 3.13: (Factors)

For all specs X,Y and Z and bags of specs $\mathcal{T}$,

$$
\begin{array}{llcll}
\text{(a)} & X & \sqsubseteq & (X \circ Y)/Y & \text{(Cancellation)} \\
\text{(b)} & X/Y \circ Y & \sqsubseteq & X & \text{(Cancellation)} \\
\text{(c)} & X \circ Y & = & (X \circ Y)/Y \circ Y & \text{(Cancellation)} \\
\text{(d)} & (X/Y \circ Y)/Y & = & X/Y & \text{(Cancellation)} \\
\text{(e)} & X \sqsubseteq Y & \Rightarrow & X \circ Z \sqsubseteq Y \circ Z & \text{(Monotonicity)} \\
\text{(f)} & X \sqsubseteq Y & \Rightarrow & X/Z \sqsubseteq Y/Z & \text{(Monotonicity)} \\
\text{(g)} & \bigsqcup \mathcal{T} \circ X & = & \bigsqcup (\mathcal{T} \circ X) & \text{(Distributivity)} \\
\text{(h)} & \bigsqcap \mathcal{T} / X & = & \bigsqcap (\mathcal{T} / X) & \text{(Distributivity)}
\end{array}
$$

□

Formula (3.13b) shows why X/Y is called a left factor of X.

The interface between composition and reverse is given in:

Axiom 3.14: (Contravariance)

$$(X \circ Y)^{\cup} = Y^{\cup} \circ X^{\cup}$$

□

As a consequence of this axiom we have:

(3.15) $I^{\cup} = I$

Proof:

$$
\begin{array}{ll}
 & I^{\cup} \\
= & \quad \{ \text{ unit, reverse } \} \\
 & I^{\cup\cup} \circ I^{\cup} \\
= & \quad \{ \text{ contravariance } \} \\
 & (I \circ I^{\cup})^{\cup} \\
= & \quad \{ \text{ unit } \} \\
 & I^{\cup\cup} \\
= & \quad \{ \text{ reverse } \} \\
 & I
\end{array}
$$

□

Axiom 3.14 together with $I^{\cup} = I$ is equivalent to saying that $^{\cup}$ is a contravariant monoid automorphism.

The last axiom for a SPEC-calculus is the Dedekind rule, a name given by Riguet [53]. The term modular identity is also used for this axiom, in particular by Freyd and Ščedrov [26]. The axiom is the only one involving all three layers of the SPEC-calculus. There are several equivalent forms for this axiom and we made an arbitrary choice, other versions being used as often as this one.

Axiom 3.16: (Dedekind, Modularity)

For all specs X, Y and Z,

$$X \circ Y \sqcap Z \sqsubseteq X \circ (Y \sqcap X^{\cup} \circ Z)$$

□

The alternative form for the Dedekind rule is the following corollary (it is actually equivalent to (3.16)):

Corollary 3.17:(Dedekind, Modularity)

For all specs X, Y and Z,

$$Y \circ X \ \sqcap \ Z \ \ \sqsubseteq \ \ (Y \ \sqcap \ Z \circ X^{\cup}) \circ X$$

□

The two forms can be derived from each other by substituting $X^{\cup}$ for X, $Y^{\cup}$ for Y and $Z^{\cup}$ for Z, applying (3.6b) followed by simplifying using the rules for reverse. The Dedekind rules are so important and so often used that it is not possible to give a short selection of consequences. The next chapter will show many examples of their use.

3.4 Strengthening the axiomatisation

The SPEC-calculus is a rather weak axiom system for the set-theoretic relations and stronger axioms are used in the other axiom systems mentioned in the beginning of this chapter. This section takes a closer look at four possible strengthenings of the SPEC-calculus.

3.4.1 Complements

The set-theoretic relations have a powerset structure, and this means that every relation on $\mathcal{A}$ can be complemented with respect to $\mathcal{A} \times \mathcal{A}$. For every relation R, there is a unique relation $\neg R$ satisfying

$$R \cap \neg R \ = \ \emptyset \ \wedge \ R \cup \neg R \ = \ \mathcal{A} \times \mathcal{A}$$

$\neg R$ is both the largest relation satisfying the first conjunct and the smallest relation satisfying the second conjunct. In the SPEC-calculus the largest spec satisfying the first conjunct is $(R\sqcap)^{\sharp}.\perp\!\!\perp$ and the smallest spec satisfying the second conjunct is $(R\sqcup)^{\flat}.\top\!\!\top$. The equality of these so-called *pseudo-complements* is not guaranteed by the axioms for a SPEC-calculus and we must add another axiom for the existence of complements:

Axiom 3.18:(Complements)

For all specs X, $(X\sqcap)^{\sharp}.\perp\!\!\perp \ = \ (X\sqcup)^{\flat}.\top\!\!\top$.

□

A SPEC-calculus satisfying (3.18) is called a *complemented SPEC-calculus*. Earlier work to which the author contributed assumed a complemented SPEC-calculus (see for example [2][4]), but this was abandoned because all important results could be achieved without resorting to complements. The discipline of not using complements in proofs often resulted in simpler and easier to understand proofs of properties earlier proved with complements.

3.4.2 Desargues rule

While complements are probably not a useful addition to the axiom system, the Desargues axiom looks more promising. This axiom is a stronger version of the Dedekind rule (Dedekind is a simple instantiation) whose only disadvantage seems to be that there are six variables in the formula:

Axiom 3.19: (Desargues)

For all specs P, Q, R, S, T and U:

$$\begin{array}{ll} & (P \circ T \ \sqcap \ Q \circ U^{\cup}) \circ (T^{\cup} \circ R \ \sqcap \ U \circ S) \ \sqsupseteq \ P \circ R \ \sqcap \ Q \circ S \\ \Leftarrow & \\ & T \circ U \ \sqsupseteq \ P^{\cup} \circ Q \ \sqcap \ R \circ S^{\cup} \end{array}$$

$\Box$

It is not difficult to prove this axiom in the set-theoretic model of relations. The Desargues axiom is mentioned in Freyd and Ščedrov [26] but looks absent from other axiom systems in the literature. Assuming this axiom can simplify the construction of Cartesian products (one axiom becomes a theorem) and is also useful for shortening proofs of properties about the distribution of $\circ$ over $\sqcap$ that are now proved using Dedekind.

The Dedekind rule can be obtained by a simple substitution in Desargues: $P := X$, $Q := I$, $R := Y$, $S := Z$, $T := I$ and $U := X^{\cup}$. So Desargues is clearly stronger then Dedekind but what about the converse? It turns out that there exist models for the SPEC-calculus where Desargues is not valid and one such model will be given in the section about models. So Desargues is strictly stronger then Dedekind. This axiom was recognised in a very late stage during the writing of this thesis, so the consequences of replacing Dedekind by Desargues in the definition of a SPEC-calculus could not be incorporated. It is an important subject for further study.

3.4.3 Cone Rule

Given two SPEC-calculi a new SPEC-calculus can be constructed by considering as specs all pairs with as first component a spec from one SPEC-calculus and as second component a spec from the other SPEC-calculus, all operations being defined coordinatewise. This method not only works with pairs but can of course also be done with triples etc. We call a SPEC-calculus constructed using this method a *product SPEC-calculus*.

A product SPEC-calculus is in general not isomorphic to relations in the set-theoretic model. Take as basis SPEC-calculi relations over a singleton set, i.e. a SPEC-calculus calculus with only two specs $\perp\!\!\perp$ and $I = \top\!\!\top$. The product SPEC-calculus has 4 different specs $((\perp\!\!\perp, \perp\!\!\perp), (\perp\!\!\perp, \top\!\!\top), (\top\!\!\top, \perp\!\!\perp), (\top\!\!\top, \top\!\!\top))$. This is clearly not isomorphic to relations over a set $\mathcal{A}$, since there are $2^{(|\mathcal{A}|^2)}$ relations in $\mathcal{P}(\mathcal{A} \times \mathcal{A})$.

The cone rule is an axiom that can be used to distinguish between product SPEC-calculi and "ordinary" SPEC-calculi. It is valid in the set-theoretic model but does not hold in product SPEC-calculi (an exception must be made for trivial one-spec calculi):

Axiom 3.20: (Cone Rule)

For all specs X,

$$\top\!\!\top \circ X \circ \top\!\!\top = \top\!\!\top \quad \equiv \quad X \neq \perp\!\!\perp$$

□

A SPEC-calculus satisfying the cone rule is called a *unary SPEC-calculus* . Most theory developed in this thesis is valid in all SPEC-calculi, but we sometimes need the stronger assumption of a unary SPEC-calculus. We use the term *binary SPEC-calculus* for the product of two unary SPEC-calculi, *ternary SPEC-calculus* for the product of three unary SPEC-calculi etc.

3.4.4 Extensionality

Set-theoretic relations can be written in a unique way as a union of singleton relations — relations consisting of exactly one pair. The extensionality axiom is used for modelling this property in a unary SPEC-calculus. To do this modelling we introduce the SPEC equivalent of a singleton relation, the singleton spec:

Definition 3.21: (Singleton)

A spec X is *singleton* iff it has the following four properties:

$$\begin{array}{lrcl} \text{(a)} & X & \neq & \perp\!\!\perp \\ \text{(b)} & I & \sqsupseteq & X \circ X\cup \\ \text{(c)} & I & \sqsupseteq & X\cup \circ X \\ \text{(d)} & X & = & X \circ \top\!\!\top \circ X \end{array}$$

□

It is not difficult to show that this corresponds to singleton relations in the set-theoretic model. We can now state the extensionality axiom:

Axiom 3.22: (Extensionality)

For all specs R,

$$R = \bigsqcup(X \mid singleton.X \;\wedge\; R \sqsupseteq X \mid X)$$

□

A SPEC-calculus satisfying this axiom is called an *extensional SPEC-calculus*. One of the most important properties of a unary extensional SPEC-calculus (proved in the next chapter) is that every spec can be written in a unique way as the cup of singleton specs. This property is essential for the following representation theorem:

Theorem 3.23: (Relational representation) A unary extensional SPEC-calculus is isomorphic to the set-theoretic relations over $\{X \mid singleton.X \mid X \circ X\cup\}$. Spec R corresponds to relation $\mathcal{R}$ via:

(a) $\mathcal{R} \ = \ \{X \mid singleton.X \ \wedge \ R \sqsupseteq X \mid (X \circ X^{\cup}, X^{\cup} \circ X)\}$
(b) $R \ = \ \bigsqcup(X, Y \mid (X, Y) \in \mathcal{R} \mid X \circ \top\!\!\top \circ Y)$
□

This theorem is also proved in the next chapter. The extensionality axiom is a very powerful tool allowing us to translate proofs in the set-theoretic relational calculus to proofs in the SPEC-calculus but, as with complements, the gain for the theory seems minimal. Proofs without extensionality are usually more elegant and only very few interesting properties really rely on extensionality. More about extensionality can be found in the work of Rietman [52] and in the next chapter of this thesis.

3.5 A geometric model

The SPEC-calculus is, as already mentioned, not a complete axiomatisation of the set-theoretic relations. Section 3.4.3 already exhibited a SPEC-calculus that is not isomorphic to any relational calculus (and does not have the cone rule). This section will show a family of models having the cone rule and where not every member satisfies Desargues.

Consider a projective geometry, that is a set $\mathcal{M}$ (called points) together with a set $\mathcal{L} \subseteq \mathcal{P}(\mathcal{M})$ (called lines) such that for every two different points there is precisely one line containing both points and for every two different lines there is precisely one point in the intersection of the lines. Add a new point ∞ (constructing a new set of points $\mathcal{M}'$) and add for every point in $\mathcal{M}$ a line with only ∞ and the point (constructing a new set of lines $\mathcal{L}'$). This is no longer a projective geometry but two different points are still on precisely one line. We can now construct a SPEC-calculus:

- Specs: subsets $X \subseteq \mathcal{M}'$ such that $X \ = \ \emptyset \ \vee \ \infty \in X$.
- Ordering: $\subseteq$.
- Reverse: identity function.
- Composition: $X \circ Y \ = \ \{p, q, r \mid p \in X \wedge q \in Y \wedge p \neq q \wedge \exists(l \mid l \in \mathcal{L}' \mid p, q, r \in l) \mid r\} \ \cup \ \{p, q \mid p \in X \ \wedge \ q \in Y \ \wedge \ p = q \mid p\}$.
- Identity: $\{\infty\}$.

The proof that this is indeed a SPEC-calculus is not difficult but a lot of work and is left to the interested reader. The interesting thing about this model is that many SPEC-calculus theorems turn out to be equivalent to geometric theorems. One "geometric translation" is the following:

Represent a point by the spec containing the point together with ∞, represent a line by the spec containing all points on that line together with ∞. The composition of two points is the line through both points if the points are different and the point itself if the points are the same. The cap of specs is the intersection. Doing this for the

Dedekind rule we see that it states that if Z is on the line through X and Y then Y is on a line through X and Z.

This also means that a counterexample to some geometric property gives a counterexample for the corresponding property in the SPEC-calculus. One interesting geometric theorem is Desargues theorem, which is a theorem for plane geometry that is not valid in all projective geometries. If we use the construction above starting with a projective geometry not satisfying Desargues theorem then the end result will be a SPEC-calculus not satisfying the Desargues rule (3.19), thereby proving that the Desargues rule is strictly stronger than the Dedekind rule.

3.6 Conclusion

The SPEC-calculus is a (weak) axiomatisation of the set-theoretic relations. The use of Galois connections helps in giving very compact axioms for the calculus. There are models for a SPEC-calculus that are not isomorphic to set-theoretic relations, but the addition of the cone rule and extensionality as extra axioms does give a structure isomorphic to set-theoretic relations. A point for further research is the question whether the Dedekind rule should be replaced by the Desargues rule.

Chapter 4

Algebraic properties of the SPEC-calculus

The aim of this chapter is to construct a toolkit of algebraic properties of the SPEC-calculus. The axioms of the SPEC-calculus are simple, low-level properties and proofs for the higher level results in other parts of the thesis would become very cumbersome without such a toolkit. Another objective of this chapter is to demonstrate various proof techniques that can be used in the SPEC-calculus. Almost all results in this chapter can be found in the literature, but the proofs given there often involve complements (which we do not assume) or are unnecessarily complicated.

The toolkit that is given here is not only intended to prepare for the subsequent chapters of this thesis, but also contains many general results that may be of interest for other applications of the SPEC-calculus that are not discussed here. The structure of our toolkit is as follows:

We start by examining how properties of composition carry over to properties of factors. This is followed by a section in which we discuss various forms of distribution of composition over cap. The third section shows how Galois connections between lattices of subsets of the specs can be used to derive many useful properties in the SPEC-calculus. The fourth section discusses the relationships between three different ways of representing a subset of the universe as a spec. The fifth section is about extensionality and the final section shows methods for the construction of certain useful classes of specs.

4.1 Factors

Galois-connected functions often "inherit" algebraic properties from each other and this is also the case for composition and factors. The axioms in the composition layer were all formulated using the $\circ$ operator because this operator is used more often in our calculations, but equivalent axiomatisations using factors are also possible. In this section we examine how the axioms about composition are formulated in terms of factors and give some lemmas following from such an axiomatisation.

The first axiom about composition is the monoid structure, stating that composition is associative and has as unit I. The associativity of composition leads to three versions for the corresponding axiom for the factors. These three versions are constructed by starting with the expression $R \ \sqsupseteq \ S \circ T \circ U$ and factoring two specs to the other side of the inclusion, leaving either S, T or U on the rhs. This factoring can be done in two different ways due to the associativity of composition, resulting in the following three equivalences:

$$\begin{array}{lcl} R/(T \circ U) \sqsupseteq S & \equiv & (R/U)/T \sqsupseteq S \\ (S\backslash R)/U \sqsupseteq T & \equiv & S\backslash(R/U) \sqsupseteq T \\ (S \circ T)\backslash R \sqsupseteq U & \equiv & T\backslash(S\backslash R) \sqsupseteq U \end{array}$$

These equivalences are universally quantified over R, S, T and U and each equivalence can be turned into an equality using the principle of "indirect equality" (2.9) . Thus we obtain the following three equalities for the equivalences above:

Lemma 4.1: (Factors)

$$\begin{array}{l} R/(T \circ U) = (R \,/\, U)/T \\ (S\backslash R)/U = S\backslash(R/U) \\ (S \circ T)\backslash R = T\backslash(S\backslash R) \end{array}$$

□

Each of these three equalities is equivalent to composition being associative (assuming the composition-factor Galois connection). The property that I is the unit of composition also has a corresponding property for factors. The equivalence form of this property is

$$I\backslash R \sqsupseteq S \quad \equiv \quad R \sqsupseteq S \quad \equiv \quad R/I \sqsupseteq S$$

Using indirect equality this becomes:

Lemma 4.2: (Factors)

$$I\backslash R = R = R/I$$

□

The contravariance of reverse over composition can be translated to a property of factors using the following equivalences:

$$\begin{array}{ll} & (R/S)^{\cup} \sqsupseteq T \\ \equiv & \qquad \{ \text{ reverse } \} \\ & R/S \sqsupseteq T^{\cup} \\ \equiv & \qquad \{ \text{ factors } \} \\ & R \sqsupseteq T^{\cup} \circ S \\ \equiv & \qquad \{ \text{ contravariance, reverse } \} \\ & R \sqsupseteq (S^{\cup} \circ T)^{\cup} \\ \equiv & \qquad \{ \text{ reverse } \} \end{array}$$

$$
\begin{array}{ll}
 & R^{\cup} \sqsupseteq S^{\cup} \circ T \\
\equiv & \qquad \{ \text{ factors } \} \\
 & S^{\cup} \setminus R^{\cup} \sqsupseteq T
\end{array}
$$

Using indirect equality this becomes the following lemma:

Lemma 4.3: (Factors)

$$(R/S)^{\cup} = S^{\cup} \setminus R^{\cup}$$

□

The Dedekind rule does not seem to have an elegant version using factors. The Galois connection gives us also some other often-used properties that were not mentioned before. The universal distribution of the adjoints over glb or lub means in particular that they also distribute over the glb or lub of the empty set of specs:

Lemma 4.4: (Zero)

$$
\begin{array}{ll}
\text{(a)} & \perp\!\!\perp \circ X = \perp\!\!\perp = X \circ \perp\!\!\perp \\
\text{(b)} & \top\!\!\top / X = \top\!\!\top = X \backslash \top\!\!\top
\end{array}
$$

□

The last properties of the factors that we discuss here follow from a simple rearrangement of the original Galois connection:

$$R/S \sqsupseteq T \qquad \equiv \qquad S \sqsubseteq T \backslash R$$

This shows a second Galois connection, this time between $R/$ and $\backslash R$. As usual with Galois connections we instantiate the cancellation, monotonicity and distributivity properties:

Corollary 4.5: (Factors)

For all specs X, Y and Z and bags of specs $\mathcal{T}$,

$$
\begin{array}{llcll}
\text{(a)} & X & \sqsubseteq & Y/(X\backslash Y) & \text{(Cancellation)} \\
\text{(b)} & X & \sqsubseteq & (Y/X)\backslash Y & \text{(Cancellation)} \\
\text{(c)} & X\backslash Y & = & (Y/(X\backslash Y))\backslash Y & \text{(Cancellation)} \\
\text{(d)} & Y/X & = & Y/((Y/X)\backslash Y) & \text{(Cancellation)} \\
\text{(e)} & X \sqsubseteq Y & \Rightarrow & Z/X \sqsupseteq Z/Y & \text{(Anti-Monotonicity)} \\
\text{(f)} & X \sqsubseteq Y & \Rightarrow & X\backslash Z \sqsupseteq Y\backslash Z & \text{(Anti-Monotonicity)} \\
\text{(g)} & Y \,/\, \bigsqcup\mathcal{T} & = & \bigsqcap(Z \mid Z\in\mathcal{T} \mid Y/Z) & \text{(Distributivity)} \\
\text{(h)} & \bigsqcup\mathcal{T} \setminus Y & = & \bigsqcap(Z \mid Z\in\mathcal{T} \mid Z\backslash Y) & \text{(Distributivity)}
\end{array}
$$

□

Note the reversal of the ordering in (4.5e) and (4.5f). This is explained by the fact that $R/$ and $\backslash R$ are Galois-connected functions when viewed between the posets $(\mathcal{S}, \sqsubseteq)$ and $(\mathcal{S}, \sqsupseteq)$.

Instantiating (4.5g) and (4.5h) with an empty bag of specs yields the last lemma of this section:

Lemma 4.6: (Factors)

$$Y/\bot\!\!\bot \;=\; \bot\!\!\bot\backslash Y \;=\; \top\!\!\top$$

□

4.2 Distribution of composition over cap

Composition is universally $\sqcup$-distributive, but distribution of composition over $\sqcap$ is in general not valid. In this section we investigate conditions under which distribution over a binary $\sqcap$ is allowed. The results can easily be generalised to distribution over arbitrary finite $\sqcap$.

Several forms of distribution will be examined, starting with the most straightforward form:

$$(R{\sqcap}S)\circ T \;=\; R\circ T\sqcap S\circ T$$

This is for distribution from the right; distribution from the left is valid under similar conditions (apply $^{\cup}$ to everything). Since $\circ T$ is monotonic we already know one inclusion, i.e.

$$(R{\sqcap}S)\circ T \;\sqsubseteq\; R\circ T\sqcap S\circ T$$

The main lemma about this form of distribution gives sufficient conditions for the other inclusion in situations where one or two of the variables are universally quantified:

Lemma 4.7: ($\circ$-$\sqcap$ Distribution)

$$\begin{array}{lrcl}
\text{(a)} & \forall(R \;\vdots\vdots\; (R{\sqcap}S)\circ T = R\circ T\sqcap S\circ T) & \Leftarrow & S \sqsupseteq S\circ T\circ T^{\cup} \\
\text{(b)} & \forall(R,S \;\vdots\vdots\; (R{\sqcap}S)\circ T = R\circ T\sqcap S\circ T) & \Leftarrow & I \sqsupseteq T\circ T^{\cup} \\
\text{(c)} & \forall(R,T \;\vdots\vdots\; (R{\sqcap}S)\circ T = R\circ T\sqcap S\circ T) & \Leftarrow & S = S\circ \top\!\!\top
\end{array}$$

□

Proof of (4.7a):

$$\begin{array}{ll}
 & (R{\sqcap}S)\circ T \\
\sqsupseteq & \qquad \{\ \bullet\ S \sqsupseteq S\circ T\circ T^{\cup};\ \text{monotonicity}\ \} \\
 & (R\ \sqcap\ S\circ T\circ T^{\cup})\circ T \\
\sqsupseteq & \qquad \{\ \text{Dedekind}\ \} \\
 & R\circ T\sqcap S\circ T
\end{array}$$

The other two parts follow immediately from this since $I \sqsupseteq T\circ T^{\cup}$ implies $S \sqsupseteq S\circ T\circ T^{\cup}$ (compose with S on both sides and use monotonicity) and $S = S\circ\top\!\!\top$ implies $S \sqsupseteq S\circ T\circ T^{\cup}$ ($\top\!\!\top \sqsupseteq T\circ T^{\cup}$ and monotonicity). This completes the proof of the lemma.

The conditions stated in the lemma are actually quite sharp because the implication is an equivalence in a complemented SPEC-calculus. The proofs in the $\Rightarrow$ direction in the complemented calculus are done by choosing appropriate instantiations for the quantified variables ($R := \neg S$, $S := I$, $T := \top\!\top$) followed by some simple manipulation to eliminate the negations in the inclusion.

The second form of distribution we investigate is the Dedekind rule. This axiom only gives an inclusion, but for many proofs an equality would be desirable. The distribution property that we are interested in is conditions for:

$$(R \sqcap S \circ T^\cup) \circ T = R \circ T \sqcap S$$

(conditions for distribution from the left will be dual). The lemma we have for the distribution is

Lemma 4.8: (∘-⊓ Distribution, Dedekind)

(a) $\forall(R ¦¦ (R \sqcap S \circ T^\cup) \circ T = R \circ T \sqcap S) \equiv S \sqsupseteq S \circ T^\cup \circ T$
(b) $\forall(R, S ¦¦ (R \sqcap S \circ T^\cup) \circ T = R \circ T \sqcap S) \equiv I \sqsupseteq T^\cup \circ T$
(c) $\forall(R, T ¦¦ (R \sqcap S \circ T^\cup) \circ T = R \circ T \sqcap S) \equiv S = S \circ \top\!\top$

□

By the Dedekind rule there is only one inclusion to prove in the $\Leftarrow$ direction:

$$\begin{array}{ll}
 & R \circ T \sqcap S \\
\sqsupseteq & \quad \{ \bullet\ S \sqsupseteq S \circ T^\cup \circ T \ \} \\
 & R \circ T \sqcap S \circ T^\cup \circ T \\
\sqsupseteq & \quad \{ \text{ glb, monotonicity } \} \\
 & (R \sqcap S \circ T^\cup) \circ T
\end{array}$$

This proves (4.8a); the other two components of the lemma follow immediately (same principle as used in the proof of lemma (4.7)). The proofs in the $\Rightarrow$ direction are done by instantiating (4.8a) in both cases with $S \circ T^\cup$ for R and then, in the case of (4.8b), I for S and, in the case of (4.8c), $\top\!\top$ for T. Some simple manipulation then leads to the desired expression on the rhs. Note that the equivalences in this lemma do not depend on the existence of complements and are valid in every SPEC-calculus.

The third distribution property examined here can be seen as a cross between the previous two:

$$(R \sqcap S) \circ T = R \circ T \sqcap S$$

The lemma we have for this distribution is

Lemma 4.9: (∘-⊓ Distribution)

(a) $\forall(R ¦¦ (R \sqcap S) \circ T = R \circ T \sqcap S) \Leftarrow S \sqsupseteq S \circ (T^\cup \sqcup T)$
(b) $\forall(R, S ¦¦ (R \sqcap S) \circ T = R \circ T \sqcap S) \equiv I \sqsupseteq T$
(c) $\forall(R, T ¦¦ (R \sqcap S) \circ T = R \circ T \sqcap S) \equiv S = S \circ \top\!\top$

□

For this property it is not the case that one inclusion is always valid so both inclusions have to be considered in the proof for the $\Leftarrow$ direction:

$$
\begin{array}{ll}
 & R \circ T \sqcap S \\
\sqsupseteq & \quad \{ \bullet\ S \sqsupseteq S \circ (T^{\cup} \sqcup T);\ S \circ (T^{\cup} \sqcup T) \sqsupseteq S \circ T \ \} \\
 & R \circ T \ \sqcap\ S \circ T \\
\sqsupseteq & \quad \{ \text{ monotonicity } \} \\
 & (R \sqcap S) \circ T \\
\sqsupseteq & \quad \{ \bullet\ S \sqsupseteq S \circ (T^{\cup} \sqcup T);\ S \circ (T^{\cup} \sqcup T) \sqsupseteq S \circ T^{\cup} \ \} \\
 & (R \ \sqcap\ S \circ T^{\cup}) \circ T \\
\sqsupseteq & \quad \{ \text{ Dedekind } \} \\
 & R \circ T \sqcap S
\end{array}
$$

The proof above is an example of the use of cyclic inclusion. All expressions are equal to each other, in particular the first and the third one. The remaining two parts of the lemma are simple consequences of this part. The proof for the $\Rightarrow$ direction of (4.9b) is done by instantiating (4.9a) with I for both R and S, the proof for (4.9c) by instantiating (4.9a) with S for R and $\top\!\!\top$ for T. The condition for (4.9a) is sharp because by instantiating R with $\neg S$ (in a complemented SPEC-calculus) and S we obtain conditions together equivalent to $S \sqsupseteq S \circ (T^{\cup} \sqcup T)$.

The distribution properties, with the exception of (4.9a), are used quite often and the classes of specs associated with these distribution properties occur so often that they deserve special names.

From (4.7b) we distinguish the class of all specs R satisfying $I \sqsupseteq R \circ R^{\cup}$. The interpretation in the set-theoretic model is that relation R satisfies:

$$\forall(x, y, z \mid (y, x){\in}R \wedge (z, x){\in}R \mid y = z)$$

This means, because we view relations as taking their argument from the rhs, that R is a (partial) endofunction on the universe. This class of specs is also used in the lhs-distribution version of (4.8b). This class of specs has been given different names in the literature, for example *simple* by Freyd and Ščedrov. Backhouse et al. [3, 4, 2] use the name *imp* (from implementation) and call the dual concept (see next paragraph) a *co-imp* . This suggests that one of them is more fundamental than the other one and to avoid this suggestion we use the terms left-imp and right-imp, showing the symmetry between the two concepts. The specs satisfying $I \sqsupseteq R \circ R^{\cup}$ are called *left-imps*, the corresponding predicate is *limp*.

In (4.8b) and the lhs-distribution version of (4.7b) we see the class of all specs R satisfying $I \sqsupseteq R^{\cup} \circ R$. Set-theoretically this means that relation R satisfies:

$$\forall(x, y, z \mid (y, x){\in}R \wedge (y, z){\in}R \mid x = z)$$

i.e. R is a (partial) injection on the universe. This class of specs is also known under several different names. We use the term *right-imp* with corresponding predicate *rimp*.

In (4.7c), (4.8c) and (4.9c) we identify the class of specs R satisfying $R = R \circ \top\!\!\top$, in the set-theoretic model $\forall(x, y, z \mid\mid (y, x){\in}R \equiv (y, z){\in}R)$. This means that an element of the universe on the lhs is either paired with all elements of the universe or not paired at all, partitioning the universe in two parts. This gives a method for representing subsets of the universe as specs using $(y, x){\in}R \equiv y{\in}\mathcal{S}$ to represent subset $\mathcal{S}$ of the universe as relation R. We call this class of specs the *left-conditions*, predicate *lc*. This name was suggested by Wim Feijen. Schmidt and Ströhlein [54] use the name *vectors* for this class of specs. From the lhs-distribution versions of the above mentioned properties we obtain the class of specs R satisfying $R = \top\!\!\top \circ R$, also a method for representing subsets. This class is called the *right-conditions*, the corresponding predicate *rc*.

In (4.9b) we see another class of specs: the class of specs R satisfying $I \sqsupseteq R$. Set-theoretically this becomes $\forall(x, y \mid (y, x) \in R \mid x = y)$. This means that relation R consists only of pairs with the same lhs and rhs. It gives a third method for representing subsets using $(x, x) \in R \equiv x \in \mathcal{S}$ to represent subset $\mathcal{S}$ of the universe as relation R. This class is called the *p*artial *id*entity relations, or pids for short. The corresponding predicate is *pid*. Backhouse et al. used the term *monotype*, whilst Freyd and Ščedrov refer to *coreflexives*.

Summing up, we introduce the following definitions:

Definition 4.10:(Classification)

(a)	$limp.R \triangleq I \sqsupseteq R \circ R^{\cup}$	(Left-imp)
(b)	$rimp.R \triangleq I \sqsupseteq R^{\cup} \circ R$	(Right-imp)
(c)	$lc.R \triangleq R = R \circ \top\!\!\top$	(Left-condition)
(d)	$rc.R \triangleq R = \top\!\!\top \circ R$	(Right-condition)
(e)	$pid.R \triangleq I \sqsupseteq R$	(Partial Identity Relation)

4.3 Lattices and Galois connections

The Galois connections we have seen thus far, (3.5) for reverse and (3.11) for composition, were all on the whole lattice of specs. In this section we will show that some interesting Galois connections can be constructed by working with a lattice of a subset of the specs.

Given a complete lattice $(\mathcal{S}, \sqsubseteq)$, we are first interested in conditions under which $(\mathcal{T}, \sqsubseteq)$ with $\mathcal{T} \subseteq \mathcal{S}$ forms a complete lattice. We know from (2.15) that it is sufficient to prove either that glbs exist for all subsets of $\mathcal{T}$ or lubs exist for all subsets of $\mathcal{T}$. A simple condition guaranteeing the existence of the bounds is that we can use either the glb or lub from the original lattice $(\mathcal{S}, \sqsubseteq)$ as glb or lub for $(\mathcal{T}, \sqsubseteq)$. This is the case if $\mathcal{T}$ is closed under glb or lub of $(\mathcal{S}, \sqsubseteq)$. If $\mathcal{T}$ is closed under both lub and glb then it is called a *sublattice*.

Five new lattices will be introduced in this section; the first one is $(\mathcal{T}, \sqsubseteq)$ with $\mathcal{T}$ defined by $R \in \mathcal{T} \equiv R \sqsubseteq T$ for some fixed spec T. The new lattice has the same lub as the original spec-lattice since $\bigsqcup \mathcal{U} \in \mathcal{T}$ holds for every $\mathcal{U} \subseteq \mathcal{T}$. The (rather trivial) proof of this fact goes as follows:

$$
\begin{array}{ll}
 & \bigsqcup \mathcal{U} \in \mathcal{T} \\
\equiv & \qquad \{\ R \in \mathcal{T} \ \equiv\ R \sqsubseteq T\ \}
\end{array}
$$

$$
\begin{array}{ll}
& \sqcup\mathcal{U} \sqsubseteq T \\
\equiv & \quad \{ \text{ lub } \} \\
& \forall(R \mid R\in\mathcal{U} \mid R \sqsubseteq T) \\
\equiv & \quad \{ R\in\mathcal{T} \;\equiv\; R \sqsubseteq T \} \\
& \forall(R \mid R\in\mathcal{U} \mid R\in\mathcal{T}) \\
\equiv & \quad \{ \mathcal{U} \subseteq \mathcal{T} \} \\
& \mathit{true}
\end{array}
$$

This lattice is used in the following Galois connection where $T \;=\; \top\!\!\top \circ f\; T$:
Lemma 4.11:

For all limps f, all specs X, and all specs Y satisfying $\top\!\!\top \circ f \;\sqsupseteq\; Y$:

$$Y \sqsubseteq X \circ f \quad\equiv\quad Y \circ f^{\cup} \sqsubseteq X$$

□

The lhs inclusion is in the lattice of specs at most $\top\!\!\top \circ f$ and the rhs inclusion is in the normal spec-lattice. The proof of this lemma goes as follows (using cyclic implication):

$$
\begin{array}{ll}
& Y \sqsubseteq X \circ f \\
\equiv & \quad \{ \top\!\!\top \circ f \sqsupseteq Y \} \\
& Y \sqcap \top\!\!\top \circ f \sqsubseteq X \circ f \\
\Leftarrow & \quad \{ \text{ Dedekind } \} \\
& (Y \circ f^{\cup} \sqcap \top\!\!\top) \circ f \sqsubseteq X \circ f \\
\Leftarrow & \quad \{ \text{ monotonicity, top } \} \\
& Y \circ f^{\cup} \sqsubseteq X \\
\Leftarrow & \quad \{ I \sqsupseteq f \circ f^{\cup} \} \\
& Y \circ f^{\cup} \sqsubseteq X \circ f \circ f^{\cup} \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
& Y \sqsubseteq X \circ f
\end{array}
$$

Instantiation of the standard Galois connection properties yields as most interesting result that $\circ f$ distributes over all glbs in the lattice of specs and, in so doing creates a glb in the lattice of specs at most $\top\!\!\top \circ f$. The glb in the latter lattice is the same as the cap of the spec-lattice except for the glb of an empty set which is $\top\!\!\top \circ f$ instead of $\top\!\!\top$. This means that $\circ f$ distributes over all non-empty caps:

Lemma 4.12: (Distributivity)

For all non-empty sets of specs $\mathcal{U}$ and all limps f:

$$\sqcap\mathcal{U} \circ f \;=\; \sqcap(\mathcal{U} \circ f)$$

□

The results above can be dualised by using a spec f satisfying $I \sqsupseteq f^{\cup} \circ f$ and working in the lattice of specs at most $f \circ \top\!\!\top$. In that case there is a Galois connection with the functions $f\circ$ and $f^{\cup}\circ$.

The second lattice that we are going to use is constructed by only working with the transitive specs. Transitivity of specs is defined by:

Definition 4.13: (Transitivity)

A spec R is *transitive* iff $R \circ R \sqsubseteq R$
□

The interpretation of transitivity in the set-theoretic model is that relation R satisfies $\forall(x, y, z \mid (x, y){\in}R \wedge (y, z){\in}R \mid (x, z){\in}R)$, the standard notion of transitivity of a relation. The definition for the lattice becomes $R{\in}\mathcal{T} \equiv R \circ R \sqsubseteq R$. The transitive specs are closed under the cap of the spec-lattice. That is, $\sqcap\mathcal{U} \in \mathcal{T}$ holds for all $\mathcal{U} \subseteq \mathcal{T}$:

$$
\begin{array}{ll}
 & \sqcap\mathcal{U} \in \mathcal{T} \\
\equiv & \qquad \{\ R{\in}\mathcal{T} \ \equiv\ R \circ R \sqsubseteq R\ \} \\
 & \sqcap\mathcal{U} \circ \sqcap\mathcal{U} \sqsubseteq \sqcap\mathcal{U} \\
\equiv & \qquad \{\ \text{glb}\ \} \\
 & \forall(R \mid R{\in}\mathcal{U} \mid \sqcap\mathcal{U} \circ \sqcap\mathcal{U} \sqsubseteq R) \\
\Leftarrow & \qquad \{\ R{\in}\mathcal{U} \ \Rightarrow\ \sqcap\mathcal{U} \sqsubseteq R\ \} \\
 & \forall(R \mid R{\in}\mathcal{U} \mid R \circ R \sqsubseteq R) \\
\equiv & \qquad \{\ R{\in}\mathcal{T} \ \equiv\ R \circ R \sqsubseteq R\ \} \\
 & \forall(R \mid R{\in}\mathcal{U} \mid R{\in}\mathcal{T}) \\
\equiv & \qquad \{\ \mathcal{U} \subseteq \mathcal{T}\ \} \\
 & \mathit{true}
\end{array}
$$

The Galois connection used comes from a very simple function: the identity (embedding) function. This function distributes universally over $\sqcap$ so it has a lower adjoint and we denote that adjoint with a postfix $^+$. Lower adjoints of embedding functions are called *closures*. An extensive treatment of closures can be found in [1]. The lower adjoint used here defines the transitive closure operation:

Definition 4.14: (Transitive closure)

The function $^+$ (called *transitive closure*) mapping specs to transitive specs is defined by the following Galois connection for all specs X and all transitive specs Y:

$$X \sqsubseteq Y \quad \equiv \quad X^+ \sqsubseteq Y$$

□

There is an invisible identity function applied to the first Y in the formula. Instantiation of the Galois connection cancellation properties leads to the following lemma:

Lemma 4.15: (Transitive closure)

For all specs X and all transitive specs *Y:*

(a) $X \sqsubseteq X^+$
(b) $X^{+\,+} = X^+$
(c) $Y^+ = Y$

□

The lub in the lattice of transitive specs does not coincide with cup in the spec-lattice because $\sqcup$ does not preserve transitivity, but calculating the lub is simple as is shown in the following calculation for all sets of transitive specs $\mathcal{U}$ and all transitive specs Y:

$$
\begin{array}{ll}
 & \forall(R \mid R{\in}\mathcal{U} \mid R \sqsubseteq Y) \\
\equiv & \qquad \{ \text{ spec-lub } \} \\
 & \sqcup\mathcal{U} \sqsubseteq Y \\
\equiv & \qquad \{ \text{ (4.14) } \} \\
 & (\sqcup\mathcal{U})^{+} \sqsubseteq Y
\end{array}
$$

The last step is necessary because the second inclusion is in the spec-lattice, not in the lattice of transitive specs. The result is formulated in:

Lemma 4.16: (Least Upper Bound)

The lub in the lattice of transitive specs of set $\mathcal{U}$ of transitive specs is $(\sqcup\mathcal{U})^{+}$.
□

The third lattice that we will exploit is the lattice of all specs that are both transitive and symmetric. The set-theoretic interpretation is that the relation is a *p*artial *e*quivalence *r*elation on the universe and that is the reason that we use the term *per* for such specs. As a definition:

Definition 4.17: (Per)

A spec R is a *per* iff R is transitive and R is symmetric.
□

The pers form a sublattice of the lattice of transitive specs, meaning that every per is transitive and that lubs and glbs of sets of pers calculated in the lattice of transitive specs are again pers. The transitivity part of the per-ness is trivial; we only have to prove that the result is symmetric. This is easy for the glb because reverse distributes universally over the spec glb and thus over the transitive glb. We need an extra lemma for the lub:

Lemma 4.18: (Transitive Closure, Reverse)

For all specs R: $R^{+\cup} = R^{\cup +}$
□

The proof uses that the reverse of a transitive spec is again transitive and is a nice example of the use of the principle of indirect equality. For all transitive specs Y:

$$
\begin{array}{ll}
 & R^{+\cup} \sqsubseteq Y \\
\equiv & \qquad \{ \text{ reverse } \} \\
 & R^{+} \sqsubseteq Y^{\cup} \\
\equiv & \qquad \{ \text{ transitive closure, } Y^{\cup} \text{ is transitive } \} \\
 & R \sqsubseteq Y^{\cup} \\
\equiv & \qquad \{ \text{ reverse } \} \\
 & R^{\cup} \sqsubseteq Y \\
\equiv & \qquad \{ \text{ transitive closure } \} \\
 & R^{\cup +} \sqsubseteq Y
\end{array}
$$

□

The fact that the lub of a set of pers in the lattice of transitive specs is symmetric is now trivial to prove. For set $\mathcal{U}$ of pers:

$$
\begin{array}{ll}
 & (\sqcup\mathcal{U})^{+\cup} \\
= & \qquad \{ \ (4.18) \ \} \\
 & (\sqcup\mathcal{U})^{\cup +} \\
= & \qquad \{ \ \text{reverse} \ \} \\
 & (\sqcup(\mathcal{U}^{\cup}))^{+} \\
= & \qquad \{ \ \text{pers are symmetric} \ \} \\
 & (\sqcup\mathcal{U})^{+}
\end{array}
$$

The pers play an essential role in this thesis as representations of types but treatment of this aspect is postponed until the next chapter. To distinguish pers from ordinary specs we use the following convention:

Convention 4.19: (Pers)

Pers are denoted by capitals early in the alphabet, i.e. A, B, etc. and from now on all specs denoted in this way will be pers, even if this is not explicitly declared.
□

The Galois connection that is shown in the next lemma defines what is called a (left) domain-operator:

Lemma 4.20: (Domains)

For all pers A and specs R:

$$R \circ \top\!\!\top \sqcap I \ \sqsubseteq \ A \qquad \equiv \qquad R \ \sqsubseteq \ A \circ \top\!\!\top$$

□

Proof:

$$
\begin{array}{ll}
 & R \circ \top\!\!\top \sqcap I \ \sqsubseteq \ A \\
\equiv & \qquad \{ \ \Leftarrow: \bullet\ A \circ \top\!\!\top \sqcap I \ \sqsubseteq \ A;\ \Rightarrow: \ A \ \sqsubseteq \ A \circ \top\!\!\top,\ \sqcap I \text{ on both sides } \} \\
 & R \circ \top\!\!\top \sqcap I \ \sqsubseteq \ A \circ \top\!\!\top \sqcap I \\
\equiv & \qquad \{ \ \Leftarrow: \sqcap I \text{ on both sides},\ \Rightarrow: \ A \circ \top\!\!\top \sqcap I \ \sqsubseteq \ A \circ \top\!\!\top \ \} \\
 & R \circ \top\!\!\top \sqcap I \ \sqsubseteq \ A \circ \top\!\!\top \\
\equiv & \qquad \{ \ \bullet \text{ trading } (4.21),\ lc.(A \circ \top\!\!\top) \ \} \\
 & R \ \sqcap \ I \circ \top\!\!\top^{\cup} \ \sqsubseteq \ A \circ \top\!\!\top \\
\equiv & \qquad \{ \ \text{unit, top} \ \} \\
 & R \ \sqsubseteq \ A \circ \top\!\!\top
\end{array}
$$

There are two unfulfilled proof obligations in the above: we used that $A \circ \top\!\!\top \sqcap I \ \sqsubseteq \ A$ for every per A and we used a new lemma called trading. These unfulfilled proof obligations are marked with a bullet. First the per property:

$$
\begin{array}{ll}
 & A \circ \top\!\!\top \sqcap I \\
\sqsubseteq & \qquad \{ \ \text{Dedekind} \ \}
\end{array}
$$

$$\begin{array}{ll} & A \circ (\top\!\!\top \ \sqcap \ A^{\cup} \circ I) \\ = & \quad \{ \text{ symmetry of pers, unit, top } \} \\ & A \circ A \\ \sqsubseteq & \quad \{ \text{ transitivity of pers } \} \\ & A \end{array}$$

The trading lemma used in the proof of (4.20) is the following:

Lemma 4.21:(Trading)

For all specs R, S, and T and all left-conditions L:

$$R \circ S \sqcap T \ \sqsubseteq \ L \quad \equiv \quad R \ \sqcap \ T \circ S^{\cup} \ \sqsubseteq \ L$$

□

Proving one implication is sufficient because the other implication can be derived by substituting $R{:=}T$, $S := S^{\cup}$ and $T{:=}R$.

Proof:

$$\begin{array}{ll} & R \circ S \sqcap T \ \sqsubseteq \ L \\ \Leftarrow & \quad \{ \text{ Dedekind } \} \\ & (R \ \sqcap \ T \circ S^{\cup}) \circ S \ \sqsubseteq \ L \\ \Leftarrow & \quad \{ \text{ top, } lc.L \ \} \\ & (R \ \sqcap \ T \circ S^{\cup}) \circ \top\!\!\top \ \sqsubseteq \ L \circ \top\!\!\top \\ \Leftarrow & \quad \{ \text{ monotonicity } \} \\ & R \ \sqcap \ T \circ S^{\cup} \ \sqsubseteq \ L \end{array}$$

□

The lower adjoint in the Galois connection, the function $X \ \mapsto \ X \circ \top\!\!\top \sqcap I$, is used so often that it deserves a special notation. Its dual $X \ \mapsto \ \top\!\!\top \circ X \sqcap I$ is also given a special notation here:

Definition 4.22:(Domains)

For all specs X, the postfix operators $<$ and $>$ are defined by:

$$X{<} \ \triangleq \ X \circ \top\!\!\top \sqcap I$$
$$X{>} \ \triangleq \ \top\!\!\top \circ X \sqcap I$$

□

The set-theoretic interpretation of $<$ is that $(y,x) \in R{<} \ \equiv \ x \ = \ y \wedge \exists(z \ \vdots\vdots \ (y,z){\in}R)$, i.e. $R{<}$ is the pid representation of the subset of the universe that occurs in the lhs of pairs in R. This is why $R{<}$ is called the left-domain of R and dually $R{>}$ is called the right-domain of R. Symmetric specs have the same left and right domains and we introduce a special domain notation for such specs to avoid choosing one domain operator over the other:

Definition 4.23:(Domains)

For all symmetric specs X the postfix operator $\times$ is defined by:

$$X{\times} \ \triangleq \ X{<}$$

or, equivalently,

$$X\!\times \;\; \triangleq \;\; X\!>$$

□

The $\times$ operator has the combined properties of the $<$ and $>$ operations.

An interesting property of the domain Galois connection is that this connection not only works between the specs and the pers but also between the specs and the lattice of pids. This lattice is an example of a lattice constructed by taking all specs included in a given spec, in this case I. Instantiating the Galois connection properties does not give many useful results, but two are worth mentioning: the monotonicity and lub-distributivity (for the connection with pids):

Lemma 4.24:(Domains)

For all specs R and S and bags of specs $\mathcal{T}$:

(a) $R \sqsubseteq S \;\Rightarrow\; R< \;\sqsubseteq\; S<$ (Monotonicity)

(b) $(\bigsqcup\mathcal{T})< \;=\; \bigsqcup(\mathcal{T}<)$ (Distributivity)

The domain operators can be used to give an alternative definition for pids:

Lemma 4.25:(Domains, pids)

For all specs R: $R = R< \;\;\equiv\;\; pid.R \;\;\equiv\;\; R = R>$

□

We only prove the first equivalence, the proof of the other equivalence being dual. The $\Rightarrow$ implication follows from $R = R \circ \top\!\top \sqcap I \sqsubseteq I$, for the $\Leftarrow$ implication we obtain $R \sqsubseteq R \circ \top\!\top \sqcap I$ from $R \sqsubseteq I$ and $R \sqsupseteq R \circ \top\!\top \sqcap I$ from the per-ness of pids and (4.20).

Many more interesting results about the domain operators can be obtained by using another Galois connection, this time between the lattice of pids and the lattice of left-conditions. The left-conditions form a lattice with same lub and glb as the normal spec-lattice, i.e. they form a sublattice. The Galois connection comes in two versions:

Lemma 4.26:

For all pids A and all left-conditions L:

(a) $A \circ \top\!\top \sqsubseteq L \;\;\equiv\;\; A \sqsubseteq L \sqcap I$

(b) $L \sqsubseteq A \circ \top\!\top \;\;\equiv\;\; L \sqcap I \sqsubseteq A$

(c) $L = A \circ \top\!\top \;\;\equiv\;\; L \sqcap I = A$

□

We only give the proof of the first version, the proof of (4.26b) being dual (replace all $\sqsubseteq$'s with $\sqsupseteq$'s):

$$\begin{array}{ll}
 & A \circ \top\!\top \sqsubseteq L \\
\equiv & \quad \{ \text{ unit, top } \} \\
 & A \circ \top\!\top \sqsubseteq L \sqcap I \circ \top\!\top \\
\equiv & \quad \{ \circ\text{-}\sqcap \text{ Distribution (4.9c)}, \; lc.L \; \} \\
 & A \circ \top\!\top \sqsubseteq (L \sqcap I) \circ \top\!\top
\end{array}$$

$$
\begin{array}{ll}
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
& A \sqsubseteq L\sqcap I \\
\equiv & \quad \{ A = A_<, \ pid.A \ \} \\
& A \circ \top\!\!\top \sqcap I \sqsubseteq L\sqcap I \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
& A \circ \top\!\!\top \sqsubseteq L
\end{array}
$$

□

There is a simpler proof of (4.26a), but that proof does not dualise to a proof of (4.26b).

Having a Galois connection with inclusions in both directions (as in (4.26a) and (4.26b)) means that all inclusions in the cancellation properties become equalities and that the connected functions are real inverses of each other. This is a very powerful property to have.

Lemma (4.26) is the last example in this section of a Galois connection using a lattice built with a subset of the specs, but we will show new examples in the next section. We finish this section by using the Galois connections and other lemmas above for proving some extra properties of the domain operators. The first lemma formulates the trading rule (4.21) using the left-domain operator:

Lemma 4.27: (Domain Trading)

For all specs R, S and T: $(R \circ S \sqcap T)_< \ = \ (R \ \sqcap \ T \circ S_\cup)_<$

□

The proof is done using indirect equality, for all pids A:

$$
\begin{array}{ll}
& (R \circ S \sqcap T)_< \sqsubseteq A \\
\equiv & \quad \{ \text{ domains } \} \\
& R \circ S \sqcap T \sqsubseteq A \circ \top\!\!\top \\
\equiv & \quad \{ (4.21), \ lc.(A \circ \top\!\!\top) \ \} \\
& R \ \sqcap \ T \circ S_\cup \ \sqsubseteq \ A \circ \top\!\!\top \\
\equiv & \quad \{ \text{ domains } \} \\
& (R \ \sqcap \ T \circ S_\cup)_< \sqsubseteq A
\end{array}
$$

□

A corollary of this lemma is the formula for the left-domain of an intersection:

Corollary 4.28: (Domains)

For all specs S and T: $(S\sqcap T)_< \ = \ I \ \sqcap \ T \circ S_\cup$

□

Proof:

$$
\begin{array}{ll}
& (S\sqcap T)_< \\
= & \quad \{ \text{ unit } \} \\
& (I \circ S \sqcap T)_<
\end{array}
$$

$$
\begin{array}{ll}
= & \quad \{ \text{ domain trading } \} \\
 & (I \sqcap T \circ S^{\cup})^{<} \\
= & \quad \{ \ (4.25) \ \} \\
 & I \sqcap T \circ S^{\cup}
\end{array}
$$

□

Two corollaries of the corollary are found by instantiation. The first one is an alternative definition of the domain-operator:

Corollary 4.29: (Domains)

For all specs R: $R^{<} \ = \ I \sqcap R \circ R^{\cup}$

□

Proof: instantiate (4.28) with R for both S and T.

□

The second corollary is an often-used property of pids:

Corollary 4.30: (Pids)

For all pids A and B: $A \sqcap B \ = \ A \circ B$

□

Proof: instantiate (4.28) with A for S and B for T and use that glb, composition and reverse of pids gives a pid as result.

□

The next lemma is an instantiation of the Galois connection (4.26). Instantiating A to $R^{<}$ and L to $R \circ \top\!\!\top$ in (4.26c) gives us:

Lemma 4.31: (Domains)

For all specs R: $R^{<} \circ \top\!\!\top \ = \ R \circ \top\!\!\top$

□

Lemma (4.26) is also essential in the proof of the following lemma:

Lemma 4.32: (Domains)

For all specs R and S: $R^{<} \sqsubseteq S^{<} \quad \equiv \quad R \circ \top\!\!\top \sqsubseteq S \circ \top\!\!\top$

□

Proof:

$$
\begin{array}{ll}
 & R^{<} \sqsubseteq S^{<} \\
\equiv & \quad \{ \text{ definition } < \ \} \\
 & R \circ \top\!\!\top \sqcap I \sqsubseteq S^{<} \\
\equiv & \quad \{ \ (4.26) \ \} \\
 & R \circ \top\!\!\top \sqsubseteq S^{<} \circ \top\!\!\top \\
\equiv & \quad \{ \ (4.31) \ \} \\
 & R \circ \top\!\!\top \sqsubseteq S \circ \top\!\!\top
\end{array}
$$

□

The previous lemma is used in the proof of the next lemma about the domain of a composition:

Lemma 4.33: (Domains)

For all specs R and S: $(R \circ S)_< \;=\; (R \circ S_<)_<$
□

Proof:

$$
\begin{array}{ll}
 & (R \circ S)_< \;=\; (R \circ S_<)_< \\
\equiv & \qquad \{\ (4.32)\ \} \\
 & R \circ S \circ \top\!\!\top \;=\; R \circ S_< \circ \top\!\!\top \\
\equiv & \qquad \{\ (4.31)\ \} \\
 & \mathit{true}
\end{array}
$$

□

The next lemma shows the relationship between composing with a domain and glb-ing with the corresponding left-condition:

Lemma 4.34: (Domains)

For all specs R and S: $S_< \circ R \;=\; R \sqcap S \circ \top\!\!\top$
□

Proof:

$$
\begin{array}{ll}
 & S_< \circ R \\
= & \qquad \{\ \text{definition } <\ \} \\
 & (I \sqcap S \circ \top\!\!\top) \circ R \\
= & \qquad \{\ \circ\text{-}\sqcap \text{ Distribution (4.9c)}, \ lc.(S \circ \top\!\!\top), \text{ unit}\ \} \\
 & R \sqcap S \circ \top\!\!\top
\end{array}
$$

□

A simple, but often used, corollary of this lemma is found by instantiating it with R for S:

Corollary 4.35: (Domains)

For all specs: $R_< \circ R \;=\; R$
□

Combining this corollary with (4.29) gives as result:

Lemma 4.36:

For all specs R: $R \;\sqsubseteq\; R \circ R_\cup \circ R$
□

Proof:

$$
\begin{array}{ll}
 & R \\
= & \qquad \{ \ (4.35) \ \} \\
 & R< \circ R \\
= & \qquad \{ \ (4.29) \ \} \\
 & (I \ \sqcap \ R \circ R\cup) \circ R \\
\sqsubseteq & \qquad \{ \ \text{monotonicity} \ \} \\
 & R \circ R\cup \circ R
\end{array}
$$

□

Lemma (4.34) is used in the proof of the final lemma of this section:

Lemma 4.37: (Domains)

For all specs R and pids A: $A \circ R \ = \ R \quad \equiv \quad R< \ \sqsubseteq \ A$

□

Proof:

$$
\begin{array}{ll}
 & A \circ R \ = \ R \\
\equiv & \qquad \{ \ A \ = \ A<, \ (4.34) \ \} \\
 & R \sqcap A \circ \top\!\top \ = \ R \\
\equiv & \qquad \{ \ \text{glb} \ \} \\
 & R \ \sqsubseteq \ A \circ \top\!\top \\
\equiv & \qquad \{ \ \text{domains} \ \} \\
 & R< \ \sqsubseteq \ A
\end{array}
$$

□

This section showed that Galois connections are a powerful tool for deriving algebraic properties. Another important aspect is that it is often helpful to restrict calculations to a limited part of the original lattice of discourse: none of the connections mentioned in this chapter are valid when working in the full lattice of specs. The lattices used in this section occur frequently in other parts of this thesis.

4.4 Representing sets

We have previously seen two methods for representing subsets of the universe as specs, the pids and conditions. We will also use a third method, called squares, in subsequent chapters. This section illustrates the relationships between the different methods of representation.

In the set-theoretic model we can represent subset $\mathcal{S}$ of the universe as a relation R in several different ways. Some methods that prove useful are:

$$\begin{array}{lrcl}
\text{Pids :} & (x,y)\in R & \equiv & x=y\wedge x\in\mathcal{S} \\
\text{Left}-\text{conditions :} & (x,y)\in R & \equiv & x\in\mathcal{S} \\
\text{Right}-\text{conditions :} & (x,y)\in R & \equiv & y\in\mathcal{S} \\
\text{Squares :} & (x,y)\in R & \equiv & x\in\mathcal{S}\wedge y\in\mathcal{S}
\end{array}$$

The SPEC characterisations of pids and conditions were given earlier in this chapter, the squares are defined by:

Definition 4.38: (Square)

A spec R is a *square* iff $R \;=\; R\circ\top\!\!\top\circ R^{\cup}$
□

It is not difficult to show the equivalence of the point-wise definition and the SPEC-definition in the set-theoretic model (hint: show first that all squares are symmetric). The squares form a complete lattice with the same glb as the normal spec-lattice. To prove this we first show that all squares are symmetric:

$$\begin{array}{ll}
 & R \\
= & \quad\{\ \text{definition square, reverse}\ \} \\
 & R^{\cup\cup}\circ\top\!\!\top^{\cup}\circ R^{\cup} \\
= & \quad\{\ \text{reverse}\ \} \\
 & (R\circ\top\!\!\top\circ R^{\cup})^{\cup} \\
= & \quad\{\ \text{definition square}\ \} \\
 & R^{\cup}
\end{array}$$

The claim that squares are closed under cap can now be proved; for all sets of squares $\mathcal{T}$:

$$\begin{array}{ll}
 & \sqcap\mathcal{T} \\
\sqsupseteq & \quad\{\ \text{definition square, monotonicity}\ \} \\
 & \sqcap\mathcal{T}\circ\top\!\!\top\circ\sqcap(\mathcal{T}^{\cup}) \\
= & \quad\{\ \text{reverse}\ \} \\
 & \sqcap\mathcal{T}\circ\top\!\!\top\circ(\sqcap\mathcal{T})^{\cup} \\
= & \quad\{\ \text{squares are symmetric, so cap of squares is also symmetric}\ \} \\
 & \sqcap\mathcal{T}\ \circ\ \top\!\!\top\ \circ\ \sqcap\mathcal{T} \\
\sqsupseteq & \quad\{\ \text{monotonicity}\ \} \\
 & \sqcap\mathcal{T}\circ(\sqcap\mathcal{T})^{\cup}\circ\sqcap\mathcal{T} \\
\sqsupseteq & \quad\{\ (4.36)\ \} \\
 & \sqcap\mathcal{T}
\end{array}$$

We express the relationships between pids, (left-)conditions and squares with Galois connections. The Galois connection between pids and left-conditions was given in the previous section but is repeated here for ease of comparison:

Lemma 4.26:

For all pids A and all left-conditions L:

(4.26a) $A \circ \top\!\top \sqsubseteq L \quad \equiv \quad A \sqsubseteq L \sqcap I$
(4.26b) $L \sqsubseteq A \circ \top\!\top \quad \equiv \quad L \sqcap I \sqsubseteq A$
(4.26c) $L = A \circ \top\!\top \quad \equiv \quad L \sqcap I = A$
□

The new lemmas expressing the relationships are:

Lemma 4.39:

For all left-conditions L and all squares S:
(a) $L \circ L^{\cup} \sqsubseteq S \quad \equiv \quad L \sqsubseteq S \circ \top\!\top$
(b) $S \sqsubseteq L \circ L^{\cup} \quad \equiv \quad S \circ \top\!\top \sqsubseteq L$
(c) $S = L \circ L^{\cup} \quad \equiv \quad S \circ \top\!\top = L$
□

Lemma 4.40:

For all pids A and all squares S:
(a) $A \circ \top\!\top \circ A \sqsubseteq S \quad \equiv \quad A \sqsubseteq S \sqcap I$
(b) $S \sqsubseteq A \circ \top\!\top \circ A \quad \equiv \quad S \sqcap I \sqsubseteq A$
(c) $S = A \circ \top\!\top \circ A \quad \equiv \quad S \sqcap I = A$
□

We prove (4.39a) in such a way that replacing all $\sqsubseteq$'s with $\sqsupseteq$'s gives a valid proof for (4.39b):

$$
\begin{array}{ll}
 & L \circ L^{\cup} \sqsubseteq S \\
\equiv & \quad \{ \text{ definition square, calculus } \} \\
 & L \circ L^{\cup} \sqsubseteq S \circ \top\!\top \circ \top\!\top \circ S^{\cup} \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
 & L \sqsubseteq S \circ \top\!\top \\
\equiv & \quad \{ \bullet\ L = L \circ L^{\cup} \circ \top\!\top, \text{ see below } \} \\
 & L \circ L^{\cup} \circ \top\!\top \sqsubseteq S \circ \top\!\top \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
 & L \circ L^{\cup} \sqsubseteq S
\end{array}
$$

There is still a small proof obligation:

$$
\begin{array}{ll}
 & L \\
\sqsubseteq & \quad \{ \text{ (4.36), top } \} \\
 & L \circ L^{\cup} \circ \top\!\top \\
\sqsubseteq & \quad \{ \ lc.L, \text{ top } \} \\
 & L
\end{array}
$$

We also give a proof for (4.40a) that is valid with both inclusions:

$$
\begin{array}{ll}
 & A \circ \top\!\top \circ A \sqsubseteq S \\
\equiv & \quad \{ \text{ top, symmetry pids } \}
\end{array}
$$

$$
\begin{array}{ll}
 & A \circ \top\!\top \circ \top\!\top \circ A^{\cup} \sqsubseteq S \\
\equiv & \quad \{ \ (4.39a),\ lc.(A \circ \top\!\top) \ \} \\
 & A \circ \top\!\top \sqsubseteq S \circ \top\!\top \\
\equiv & \quad \{ \ (4.26a),\ lc.(S \circ \top\!\top) \ \} \\
 & A \sqsubseteq S \circ \top\!\top \sqcap I \\
\equiv & \quad \{ \ \bullet\ S{\sqcap}I = S \circ \top\!\top \sqcap I, \text{ see below } \} \\
 & A \sqsubseteq S{\sqcap}I
\end{array}
$$

Again we have a proof obligation left over:

$$
\begin{array}{ll}
 & S{\sqcap}I \\
\sqsubseteq & \quad \{ \ \text{monotonicity, top} \ \} \\
 & S \circ \top\!\top \sqcap I \\
= & \quad \{ \ \text{domains} \ \} \\
 & S \circ S^{\cup} \sqcap I \\
\sqsubseteq & \quad \{ \ \text{monotonicity, top} \ \} \\
 & S \circ \top\!\top \circ S^{\cup} \sqcap I \\
= & \quad \{ \ \text{definition square} \ \} \\
 & S{\sqcap}I
\end{array}
$$

This completes the proofs of the two Galois connections. We will see the application of these connections in the next chapter.

4.5 Extensionality

Extensionality is a property that allows the translation of set-theoretic proofs about relations to proofs in the SPEC-calculus. The idea is that a unary extensional SPEC-calculus is isomorphic to a relational calculus with the singleton squares of the SPEC-calculus as universe, i.e. there is a one-to-one correspondence between specs and relations and the relational operators correspond to the SPEC operators. In the remainder of this section we assume that the SPEC-calculus that we are using is unary and extensional.

The extensionality axiom (3.22) is formulated using singleton specs and the elements of the relations are pairs of singleton squares. The relationship between singleton specs and pairs of singletons squares is formulated in the following lemma:

Lemma 4.41: (Singleton)

For all singleton squares X and Y and all singletons Z:

(a) $singleton.(X \circ \top\!\top \circ Y)$
(b) $singleton.(Z \circ Z^{\cup}) \quad \wedge \quad square.(Z \circ Z^{\cup})$
(c) $singleton.(Z^{\cup} \circ Z) \quad \wedge \quad square.(Z^{\cup} \circ Z)$
(d) $Z = X \circ \top\!\top \circ Y \quad \equiv \quad Z \circ Z^{\cup} = X \quad \wedge \quad Z^{\cup} \circ Z = Y$

□

The proofs of (4.41b), (4.41c) and (4.41d) are simple instantiations of the definitions with a little bit of rewriting using the cone rule and the fact that $R \circ \top\!\!\top \ = \ R \circ R^{\cup} \circ \top\!\!\top$ for all specs R. The proof of (4.41a) is similar, but there is one complication: the imp-ness conditions in the definition of singleton become $I \ \sqsupseteq \ X$ and $I \ \sqsupseteq \ Y$ after rewriting. We have to prove that singleton squares are pids.

$$
\begin{array}{ll}
 & I \\
\sqsupseteq & \qquad \{ \ \bullet \ \mathit{singleton}.X \ \} \\
 & X \circ X^{\cup} \\
= & \qquad \{ \ \bullet \ \mathit{square}.X, \text{ squares are symmetric } \} \\
 & X \circ \top\!\!\top \circ X^{\cup} \circ X \\
= & \qquad \{ \ \top\!\!\top \circ X \ = \ \top\!\!\top \circ X^{\cup} \circ X \ \} \\
 & X \circ \top\!\!\top \circ X \\
= & \qquad \{ \ \bullet \ \mathit{singleton}.X \ \} \\
 & X
\end{array}
$$

□

The correspondence between spec R and relation over singleton squares $\mathcal{R}$ is given by:

(4.42) $X \circ \top\!\!\top \circ Y \ \sqsubseteq \ R \quad \equiv \quad (X,Y) \in \mathcal{R}$, for all singleton squares X and Y.

Or equivalently (using (4.41d)):

(4.43) $Z \ \sqsubseteq \ R \quad \equiv \quad (Z \circ Z^{\cup}, Z^{\cup} \circ Z) \in \mathcal{R}$, for all singletons Z.

This correspondence is a one-to-one correspondence if every spec can be written in a unique way as the cup of singletons. The corresponding property for sets (every set is a unique union of singleton sets) is well-known from set-theory. The unique way of writing a spec as a cup of singletons in the SPEC-calculus is a consequence of the extensionality axiom (3.22): if we assume that two specs contain the same set of singletons then the extensionality axioms states that the two specs are equal (they are both equal to the cup of the singletons).

We have shown a one-to-one correspondence between specs and relations, but this does not yet prove that the SPEC-calculus is isomorphic to the relational calculus. We also have to show that the operations carry over. This is not difficult for cap and reverse (this is left as an exercise to the reader) but there are some complications for cup and composition.

The most difficult part in the proof for cup is that we have to prove that for singleton Z and bag of specs $\mathcal{T}$: $Z \ \sqsubseteq \ \bigsqcup \mathcal{T} \quad \equiv \quad \exists(R \ ¦ \ R \in \mathcal{T} \ ¦ \ Z \ \sqsubseteq \ R)$. Proof:

$$
\begin{array}{ll}
 & \exists(R \ ¦ \ R \in \mathcal{T} \ ¦ \ Z \ \sqsubseteq \ R) \\
\equiv & \qquad \{ \ (4.44), \ \mathit{singleton}.Z \ \} \\
 & \exists(R \ ¦ \ R \in \mathcal{T} \ ¦ \ R \sqcap Z \ \neq \ \perp\!\!\perp) \\
\equiv & \qquad \{ \ \text{cup, contrapositive } \} \\
 & \bigsqcup(R \ ¦ \ R \in \mathcal{T} \ ¦ \ R \sqcap Z) \ \neq \ \perp\!\!\perp \\
\equiv & \qquad \{ \ \text{distributivity } \}
\end{array}
$$

$$
\begin{array}{ll}
 & \sqcup\mathcal{T} \sqcap Z \neq \perp\!\!\perp \\
\equiv & \qquad \{ \ (4.44),\ singleton.Z \ \} \\
 & Z \sqsubseteq \sqcup\mathcal{T}
\end{array}
$$

The first and last steps in the above proof use an important property of singletons that is called atomicity:

Lemma 4.44:(Atomicity)

For all specs R and all singletons Z: $Z \sqsubseteq R \quad \equiv \quad R \sqcap Z \neq \perp\!\!\perp$
□

This property is called atomicity because it states that a singleton can only be completely contained in another spec. In other words, an atom is not divisible into two parts. Since singleton specs are non-empty, only the "⇐" part needs a proof:

$$
\begin{array}{ll}
 & R \\
\sqsupseteq & \qquad \{ \ singleton.Z, \text{ singletons are limps and rimps } \} \\
 & Z \circ Z^{\cup} \circ R \circ Z^{\cup} \circ Z \\
= & \qquad \{ \ singleton.Z, \text{ domains } \} \\
 & Z \circ Z{>} \circ \top\!\!\top \circ Z{<} \circ R \circ Z{>} \circ \top\!\!\top \circ Z{<} \circ Z \\
= & \qquad \{ \ \text{domains } \} \\
 & Z \circ \top\!\!\top \circ (Z \circ \top\!\!\top \sqcap R \sqcap \top\!\!\top \circ Z) \circ \top\!\!\top \circ Z \\
\sqsupseteq & \qquad \{ \ \text{monotonicity } \} \\
 & Z \circ \top\!\!\top \circ (R \sqcap Z) \circ \top\!\!\top \circ Z \\
= & \qquad \{ \ \bullet\ R \sqcap Z \neq \perp\!\!\perp; \text{ cone rule } \} \\
 & Z \circ \top\!\!\top \circ Z \\
= & \qquad \{ \ singleton.Z \ \} \\
 & Z
\end{array}
$$

□

The remainder of the proof of the correspondence of the cup operator on specs and the cup operation on relations is easy and left to the reader. The proof for the correspondence of the composition operators also has its difficult part. The difficult part here is proving for all specs R and S and all singleton squares X and Y that:

$$
\begin{array}{l}
X \circ \top\!\!\top \circ Y \sqsubseteq R \circ S \qquad \Rightarrow \\
\exists (V \ \vdots\ singleton.V \wedge square.V \ \vdots\ X \circ \top\!\!\top \circ V \sqsubseteq R \ \wedge \ V \circ \top\!\!\top \circ Y \sqsubseteq S)
\end{array}
$$

We are going to show that every singleton square V such that $V \sqsubseteq (X \circ R){>} \sqcap (S \circ Y){<}$ satisfies $X \circ \top\!\!\top \circ V \sqsubseteq R \ \wedge \ V \circ \top\!\!\top \circ Y \sqsubseteq S$ and that such a V exists. We only prove $X \circ \top\!\!\top \circ V \sqsubseteq R$, the proof for $V \circ \top\!\!\top \circ Y \sqsubseteq S$ being dual:

$$
\begin{array}{ll}
 & R \\
\sqsupseteq & \qquad \{ \ pid.X,\ X \text{ is singleton square (see above) } \} \\
 & X \circ R \\
= & \qquad \{ \ singleton.X \ \}
\end{array}
$$

$$
\begin{array}{ll}
& X \circ \top\!\!\top \circ X \circ R \\
\sqsupseteq & \qquad \{ \text{ domains, } V \sqsubseteq (X \circ R)> \ \} \\
& X \circ \top\!\!\top \circ V
\end{array}
$$

We prove the existence of such a V by proving that $(X \circ R)> \sqcap (S \circ Y)< \; \neq \; \perp\!\!\perp$. From extensionality it then follows that there is at least one singleton V such that $V \sqsubseteq (X \circ R)> \sqcap (S \circ Y)<$. Such a V is a pid, so it is symmetric and a symmetric singleton is a square.

$$
\begin{array}{ll}
& (X \circ R)> \sqcap (S \circ Y)< \; \neq \; \perp\!\!\perp \\
\equiv & \qquad \{ \ (4.30) \ \} \\
& (X \circ R)> \circ (S \circ Y)< \; \neq \; \perp\!\!\perp \\
\equiv & \qquad \{ \text{ cone rule, domains } \} \\
& \top\!\!\top \circ X \circ R \circ S \circ Y \circ \top\!\!\top \; = \; \top\!\!\top \\
\Leftarrow & \qquad \{ \ \bullet \, X \circ \top\!\!\top \circ Y \sqsubseteq R \circ S\text{; monotonicity, top } \} \\
& \top\!\!\top \circ X \circ X \circ \top\!\!\top \circ Y \circ Y \circ \top\!\!\top \; = \; \top\!\!\top \\
\equiv & \qquad \{ \ X \text{ and } Y \text{ are singleton squares } \} \\
& \mathit{true}
\end{array}
$$

The other parts of the proof of the correspondence of the composition operators are straightforward and left to the reader. Extensionality and its consequences are rarely used but were given here mainly for completeness sake. This is also the reason for not doing all proofs in full detail in this section; the interested reader can fill in the details that were omitted here. The main stumbling blocks in the proofs having been treated in this section.

4.6 Classification rules

The previous sections identified some interesting classes of specs but didn't give many methods for recognizing elements from these classes by examining the structure of the expression defining the spec. This section provides some of the missing methods, sometimes repeating previous results for completeness sake. The proofs of the lemmas in this section are trivial and left to the reader.

We start by giving conditions under which an expression is a limp. The conditions for expressions to be rimps are dual.

Lemma 4.45: (Limp)

$$
\begin{array}{lrcll}
\text{(a)} & \mathit{limp}.(f \sqcap R) & \Leftarrow & \mathit{limp}.f & \\
\text{(b)} & \mathit{limp}.f\cup & \equiv & \mathit{rimp}.f & \\
\text{(c)} & \mathit{limp}.(f \circ g) & \Leftarrow & \mathit{limp}.f & \wedge \quad \mathit{limp}.g \\
\text{(d)} & \mathit{limp}.f & \Leftarrow & \mathit{pid}.f &
\end{array}
$$

□

The conditions for right-conditions are dual to those for left-conditions:

Lemma 4.46: (Left-condition)

$$\begin{array}{lrcl}
\text{(a)} & lc.(R \sqcup S) & \Leftarrow & lc.R \quad \wedge \quad lc.S \\
\text{(b)} & lc.(R \sqcap S) & \Leftarrow & lc.R \quad \wedge \quad lc.S \\
\text{(c)} & lc.R^{\cup} & \equiv & rc.R \\
\text{(d)} & lc.(R \circ S) & \Leftarrow & lc.S \\
\text{(e)} & lc.(R/S) & \Leftarrow & rc.S \\
\text{(f)} & lc.(R\backslash S) & \Leftarrow & lc.S
\end{array}$$

□

Lemma 4.47: (Pid)

$$\begin{array}{lrcl}
\text{(a)} & pid.(A \sqcup B) & \equiv & pid.A \quad \wedge \quad pid.B \\
\text{(b)} & pid.(A \sqcap R) & \Leftarrow & pid.A \\
\text{(c)} & pid.A^{\cup} & \equiv & pid.A \\
\text{(d)} & pid.(A \circ B) & \Leftarrow & pid.A \quad \wedge \quad pid.B \\
\text{(e)} & pid.A & \equiv & per.A \quad \wedge \quad limp.A \\
\text{(f)} & pid.A & \equiv & per.A \quad \wedge \quad rimp.A
\end{array}$$

□

Lemma 4.48: (Square)

$$\begin{array}{lrcl}
\text{(a)} & square.(A \sqcap B) & \Leftarrow & square.A \quad \wedge \quad square.B \\
\text{(b)} & square.A^{\cup} & \equiv & square.A
\end{array}$$

□

Lemma 4.49: (Per)

$$\begin{array}{lrcl}
\text{(a)} & per.(A \sqcap B) & \Leftarrow & per.A \quad \wedge \quad per.B \\
\text{(b)} & per.A^{\cup} & \equiv & per.A \\
\text{(c)} & per.A & \Leftarrow & square.A \\
\text{(d)} & per.A & \Leftarrow & pid.A
\end{array}$$

□

These are the "typing-rules" for expressions built using the basic operations of the spec-calculus. We end this section with the base cases for the typing, the types of the constants $\perp\!\!\perp$, $\top\!\!\top$ and I:

Lemma 4.50: (Bottom, Top and Identity)

$$\begin{array}{ll}
\text{(a)} & square.\perp\!\!\perp \quad \wedge \quad pid.\perp\!\!\perp \quad \wedge \quad lc.\perp\!\!\perp \quad \wedge \quad rc.\perp\!\!\perp \\
\text{(b)} & square.\top\!\!\top \quad \wedge \quad lc.\top\!\!\top \quad \wedge \quad rc.\top\!\!\top \\
\text{(c)} & pid.I
\end{array}$$

□

The rules that are given above are usually sufficient for determining whether a spec belongs to one of the classes introduced in this chapter.

Chapter 5

Pers as types

The main goal of this thesis is to develop a theory of datatypes and programs using these datatypes *in* the SPEC-calculus. That means that types, programs and type-judgements are all expressed in the calculus and that we should be able to do all our calculations (datatype construction, program construction, type inference etc.) using the same calculus. In the previous chapters we introduced the SPEC-calculus and developed a toolkit for calculations within the calculus. In this chapter we will show how we can use a particular class of specs (the pers) to represent types and we will show how we can use them for type-judgements on specs.

An important result is that we can identify a special class of specs playing the role of functions on types represented as pers. This class, called the difunctionals, has properties similar to those of functions on sets.

5.1 The monotype system

The monotype system for typing specs, as defined by Backhouse et al in for example [1], can be seen as precursor of the type system that is used in this thesis. The basic idea is that a type is a subset of the universe and this subset is represented as a pid, a monotype in Backhouse's nomenclature.

Two type-judgements are defined in the monotype system, one for arbitrary specs and one especially for left-imps:

Definition 5.1: (Monotype judgements)

For all specs R and f and all pids A and B:

(a) $R \in A \sim B \;\triangleq\; A \circ R \;=\; R \;=\; R \circ B$

(b) $f \in A \leftarrow B \;\triangleq\; A \sqsupseteq f \circ f\cup \quad\wedge\quad f> \;=\; B$

□

Interpreting A and B as sets, $R \in A \sim B$ means that R relates elements of A with elements of B and $f \in A \leftarrow B$ that f is a total function mapping elements of B to elements of A. The domain operators give a "least" type for specs: For all specs R, all left-imps f and pids A and B:

$$(5.2) \quad R \in A \sim B \equiv R< \sqsubseteq A \wedge R> \sqsubseteq B$$
$$(5.3) \quad f \in A \leftarrow B \equiv f< \sqsubseteq A \wedge f> = B$$

The lattice ordering on pids corresponds to subtyping and pids can be used for domain restrictions on specs and left-imps. For pid C:

$$(5.4) \quad C \circ R \in A \sqcap C \sim B \Leftarrow R \in A \sim B$$
$$(5.5) \quad R \circ C \in A \sim B \sqcap C \Leftarrow R \in A \sim B$$
$$(5.6) \quad f \circ C \in A \leftarrow B \sqcap C \Leftarrow f \in A \leftarrow B$$

This can be useful to make type considerations explicit in calculations as is shown in for example [42].

It turns out that is possible to construct a surprisingly powerful theory about types, including initial inductive datatypes, by working with pids as representations of types as demonstrated in the work of Backhouse et al. [1, 2, 4]. But there are also some drawbacks and in our opinion the most important one is the problem of constructing types with laws. Types with laws are abundant in practical programming and a good type-theory must allow for the construction of and reasoning about these types.

5.2 Pers as types

The per-system was designed to capture two important methods for relating types to each other in a single system: imposing laws (creating equalities between elements) and imposing restrictions (subtyping). It can be seen as a generalization of the monotype system, which only allows subtyping: the type of a spec R is completely determined by $R<$ and $R>$ and the $\sqsubseteq$ order represents subtyping. In this section we will develop a method for typing a spec using per-valued domain operators and introduce an ordering on pers reflecting both laws and restrictions.

A common method for the introduction of a type is to give a base domain and then form a quotient by introducing equalities on the elements of the domain. A typical example of this approach is the way integers are introduced given pairs of natural numbers: $(a, b) \cong (c, d)$ iff $a + d = b + c$. This equality introduces equivalence classes on the pairs and every class corresponds to one integer. The mapping of an element of the base domain to its equivalence class is the quotient-map. In this case we have the mapping

$$(a, b) \mapsto \{(c, d) \mid a + d = b + c\}$$

as quotient-map. This simple example demonstrates that it is desirable to be able to express types constructed using quotients if we design a type-system. A second wish for such a type-system is that we are able to construct subtypes. For example, the even integers can be constructed from the integers by restriction to pairs with an even sum. In this chapter we will show that we can have both quotients and subtypes if we represent types with pers. The monotype system becomes a special case of the per system.

Representation of a quotient by a per is easy if we view the quotient map as a set-valued function and use the isomorphism between relations and set-valued functions. The resulting relation is transitive since every element of the domain is mapped to its equivalence class and this equivalence class is mapped to itself; the relation is symmetric because if element x is mapped to a set containing y then y is mapped to a set containing x (x and y are both in the same equivalence class). This means that the resulting relation is a partial equivalence relation, so it is a per in the SPEC-calculus.

Using pers to represent types is a method that is also used in other formalisms like the lambda calculus where pers over the natural numbers are used as a model (see for example [48]). Examples of using pers as types that are more closely related to the use in this thesis can be found in the work of Mili et al [47] and in the work of the Ruby group [38]. An important difference with these uses of pers is our emphasis on orderings and lattice structures on pers allowing us to define pers (= types) as extreme solutions of equations. This will be important for the construction of inductive types.

5.3 Orderings on pers

Orderings and lattices play an important role in our calculations and therefore we examine useful orderings on pers. The normal SPEC-ordering $\sqsubseteq$ has already been examined in the previous chapter (the pers form a complete lattice under $\sqsubseteq$) but $\sqsubseteq$ is quite meaningless as comparison of two pers viewed as representations of types. In this section we will investigate three new orderings on pers that do have a meaningful interpretation on pers viewed as types.

5.3.1 Pers as sets of classes

The elements of the quotient are the equivalence classes of the pers, so they are subsets of the universe. If we represent the quotient by a per then the natural representation of an element of the quotient is as a non-empty square because a per is the union of the squares representing the equivalence classes. This is demonstrated in the following definitions and lemmas.

Definition 5.7: (Class)

A spec R is a *class* iff R is a square and $R \neq \perp\!\!\perp$.
□

Definition 5.8: (Element)

For class X and per A: $\quad X \in A \quad \triangleq \quad X \sqsubseteq A \;\wedge\; X \circ A = X$
□

The interpretation of $X \in A$ is that X is an equivalence class of A, so X is a representation of an element of the quotient represented by A. The first conjunct in the definition of $X{\in}A$ states that X is a part of an equivalence class of A, the second conjunct states that X consists of complete equivalence classes of A. Together this means that X is a single equivalence class of A. The definition of $X{\in}A$ can be simplified for pids : $class.X \;\wedge\; pid.A \;\wedge\; X \sqsubseteq A \;\Rightarrow\; X \circ A = X$, so $X{\in}A \;\equiv\; X \sqsubseteq A$ for pid A. The trivial proof of this fact is left to the reader.

The statement that every per is the cup of its elements is an alternative way of stating the extensionality axiom:

Lemma 5.9: (Extensionality)

The following three statements are equivalent:

(a) $\forall(A \mid per.A \mid A = \bigsqcup(X \mid X{\in}A \mid X))$
(b) $\forall(A \mid pid.A \mid A = \bigsqcup(X \mid X{\in}A \mid X))$
(c) $\forall(R \mid\mid R = \bigsqcup(Z \mid singleton.Z \;\wedge\; Z \sqsubseteq R \mid Z))$

□

The last expression is the original definition of extensionality for a SPEC-calculus. We prove the equivalence of the three statements using cyclic implication. Because (5.9a) ⇒ (5.9b), since all pids are pers, it is sufficient to prove (5.9b) ⇒ (5.9c) and (5.9c) ⇒ (5.9a). The proof of (5.9b) ⇒ (5.9c) is the most difficult part:

$$
\begin{array}{ll}
 & R \\
\sqsupseteq & \quad \{\ \text{lub}\ \} \\
 & \bigsqcup(Z \mid singleton.Z \;\wedge\; Z \sqsubseteq R \mid Z) \\
\sqsupseteq & \quad \{\ \text{see below}\ \} \\
 & \bigsqcup(X, Y \mid X \in R_< \;\wedge\; Y \in (X \circ R)_> \mid X \circ \top\!\!\top \circ Y) \\
= & \quad \{\ \text{nesting}\ \} \\
 & \bigsqcup(X \mid X \in R_< \mid \bigsqcup(Y \mid Y \in (X \circ R)_> \mid X \circ \top\!\!\top \circ Y)) \\
= & \quad \{\ \text{distribution}\ \} \\
 & \bigsqcup(X \mid X \in R_< \mid X \circ \top\!\!\top \circ \bigsqcup(Y \mid Y \in (X \circ R)_> \mid Y)) \\
= & \quad \{\ (5.9b)\ \} \\
 & \bigsqcup(X \mid X \in R_< \mid X \circ \top\!\!\top \circ (X \circ R)_>) \\
= & \quad \{\ \text{domains, distribution}\ \} \\
 & \bigsqcup(X \mid X \in R_< \mid X \circ \top\!\!\top \circ X) \circ R \\
= & \quad \{\ \text{definition square, (5.9b)}\ \} \\
 & R_< \circ R \\
= & \quad \{\ \text{domains}\ \} \\
 & R
\end{array}
$$

The proof obligation for the second step above is:
$X \in R_< \;\wedge\; Y \in (X \circ R)_> \;\Rightarrow\; singleton.(X \circ \top\!\!\top \circ Y) \;\wedge\; X \circ \top\!\!\top \circ Y \sqsubseteq R$

The fact that $X \circ \top\!\!\top \circ Y$ is a singleton is a consequence of lemma (4.41a) and the fact that elements of pids are singletons. The other conjunct is proved as follows:

$$
\begin{array}{ll}
 & R \\
\sqsupseteq & \quad \{\ X \ \sqsubseteq\ R_< \ \} \\
 & X \circ R \\
= & \quad \{\ \mathit{square}.X \ \} \\
 & X \circ \top\!\!\top \circ X \circ R \\
= & \quad \{\ \text{domains} \ \} \\
 & X \circ \top\!\!\top \circ (X \circ R)_> \\
\sqsupseteq & \quad \{\ Y \ \sqsubseteq\ (X \circ R)_> \ \} \\
 & X \circ \top\!\!\top \circ Y
\end{array}
$$

This completes the proof of (5.9b) $\Rightarrow$ (5.9c). Now we prove (5.9c) $\Rightarrow$ (5.9a):

$$
\begin{array}{ll}
 & A \\
= & \quad \{\ \text{domains, } \mathit{per}.A \ \} \\
 & A \circ A_{\times} \circ A \\
= & \quad \{\ (5.9\text{c}) \ \} \\
 & A \circ \bigsqcup(Z \ ¦\ \mathit{singleton}.Z \quad \wedge \quad Z \ \sqsubseteq\ A_{\times} \ ¦\ Z) \circ A \\
= & \quad \{\ \text{distributivity} \ \} \\
 & \bigsqcup(Z \ ¦\ \mathit{singleton}.Z \quad \wedge \quad Z \ \sqsubseteq\ A_{\times} \ ¦\ A \circ Z \circ A) \\
\sqsubseteq & \quad \{\ \mathit{singleton}.Z \ \wedge\ Z \ \sqsubseteq\ A_{\times} \ \Rightarrow\ \mathit{square}.(A \circ Z \circ A), \\
 & \quad\ \ Z \ \sqsubseteq\ A_{\times} \ \Rightarrow\ A \circ Z \circ A \in A \ \} \\
 & \bigsqcup(X \ ¦\ X {\in} A \ ¦\ X) \\
\sqsubseteq & \quad \{\ X {\in} A \quad \Rightarrow \quad X \ \sqsubseteq\ A, \text{ lub} \ \} \\
 & A
\end{array}
$$

This completes the proof of lemma (5.9).

□

A fourth equivalent way of expressing extensionality provides a method for proving the equality of two specs in an extensional SPEC-calculus:

Lemma 5.10: (Extensionality)

A SPEC-calculus is extensional iff for all specs R and S and all pers A:

$$\forall(X \ ¦\ X {\in} A \ ¦\ X \circ R \ = \ X \circ S) \quad \equiv \quad A \circ R \ = \ A \circ S$$

□

Proof:

We first prove that the statement follows from extensionality as defined earlier:

$$
\begin{array}{ll}
 & \forall(X \ ¦\ X {\in} A \ ¦\ X \circ R \ = \ X \circ S) \\
\equiv & \quad \{\ X {\in} A \quad \Rightarrow \quad X \circ A \ = \ X \ \} \\
 & \forall(X \ ¦\ X {\in} A \ ¦\ X \circ A \circ R \ = \ X \circ A \circ S) \\
\Leftarrow & \quad \{\ \text{Leibniz} \ \} \\
 & A \circ R \ = \ A \circ S \\
\equiv & \quad \{\ \text{extensionality (5.9a)} \ \} \\
 & \bigsqcup(X \ ¦\ X {\in} A \ ¦\ X) \circ R \ = \ \bigsqcup(X \ ¦\ X {\in} A \ ¦\ X) \circ S
\end{array}
$$

$$
\begin{array}{ll}
\equiv & \quad \{ \text{ distributivity } \} \\
& \bigsqcup(X \mathrel{¦} X{\in}A \mathrel{¦} X \circ R) \;=\; \bigsqcup(X \mathrel{¦} X{\in}A \mathrel{¦} X \circ S) \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
& \forall(X \mathrel{¦} X{\in}A \mathrel{¦} X \circ R \;=\; X \circ S)
\end{array}
$$

For implying extensionality we prove (5.9a):

$$
\begin{array}{ll}
& \bigsqcup(X \mathrel{¦} X{\in}A \mathrel{¦} X) \;=\; A \\
\equiv & \quad \{ \; X{\in}A \quad \Rightarrow \quad A \circ X \;=\; X, \text{distributivity, unit } \} \\
& A \circ \bigsqcup(X \mathrel{¦} X{\in}A \mathrel{¦} X) \;=\; A \circ I \\
\equiv & \quad \{ \text{ assumption } \} \\
& \forall(Y \mathrel{¦} Y{\in}A \mathrel{¦} Y \circ \bigsqcup(X \mathrel{¦} X{\in}A \mathrel{¦} X) \;=\; Y) \\
\equiv & \quad \{ \text{ see below } \} \\
& \mathit{true}
\end{array}
$$

The last step needs some extra work:

$$
\begin{array}{ll}
& Y \\
= & \quad \{ \; Y{\in}A \; \} \\
& Y \circ A \\
\sqsupseteq & \quad \{ \; X{\in}A \quad \Rightarrow \quad X \;\sqsubseteq\; A \; \} \\
& Y \circ \bigsqcup(X \mathrel{¦} X{\in}A \mathrel{¦} X) \\
\sqsupseteq & \quad \{ \; Y{\in}A \; \} \\
& Y \circ Y \\
= & \quad \{ \; \mathit{per}.Y \; \} \\
& Y
\end{array}
$$

□

Having a notion of elements enables us to define a subset ordering on pers in the obvious way:

$$A \subseteq B \quad \triangleq \quad \forall(X \mathrel{¦} X{\in}A \mathrel{¦} X{\in}B)$$

But we don't use this as a definition because it has two big disadvantages. It contains a quantification and using this definition almost always involves extensionality. A definition that does not involve quantifications and is usable without assuming extensionality is the following:

Definition 5.11:(Subset)

For pers A and B, the subset relation is defined by:

$$A \subseteq B \quad \triangleq \quad A \;\sqsubseteq\; B \;\wedge\; A \circ B \;=\; A$$

□

We justify this definition of the subset ordering by proving that it is equivalent to the extensional definition:

Lemma 5.12: (Subset)

In an extensional SPEC-calculus:

$$A \subseteq B \quad \equiv \quad \forall(X \mathbin{\text{¦}} X{\in}A \mathbin{\text{¦}} X{\in}B)$$

□

Proof:

$$\begin{array}{ll}
 & \forall(X \mathbin{\text{¦}} X{\in}A \mathbin{\text{¦}} X{\in}B) \\
\equiv & \quad \{ \text{ definition } \in \ \} \\
 & \forall(X \mathbin{\text{¦}} X{\in}A \mathbin{\text{¦}} X \sqsubseteq B \ \wedge \ X \circ B = X) \\
\equiv & \quad \{ \text{ calculus } \} \\
 & \forall(X \mathbin{\text{¦}} X{\in}A \mathbin{\text{¦}} X \sqsubseteq B) \quad \wedge \quad \forall(X \mathbin{\text{¦}} X{\in}A \mathbin{\text{¦}} X \circ B = X) \\
\equiv & \quad \{ \text{ lub, unit } \} \\
 & \bigsqcup(X \mathbin{\text{¦}} X{\in}A \mathbin{\text{¦}} X) \sqsubseteq B \quad \wedge \quad \forall(X \mathbin{\text{¦}} X{\in}A \mathbin{\text{¦}} X \circ B = X \circ I) \\
\equiv & \quad \{ \text{ extensionality (5.9a) (5.10), unit } \} \\
 & A \sqsubseteq B \quad \wedge \quad A \circ B = A
\end{array}$$

□

Note that the definitions of $X{\in}A$ and $X \subseteq A$ are the same; a set with only one element is identified with its element. The $\subseteq$-ordering coincides with $\sqsubseteq$ on pids ($A \circ B = A{\sqcap}B$), so it is a true generalisation of the subset ordering of the monotype system. The pids of the monotype system form a complete lattice under the subset ordering but the pers do not form a complete lattice under the $\subseteq$-ordering: there is a bottom ($\perp\!\!\perp$) but no top. For example, both I and $\top\!\!\top$ are maximal under the $\subseteq$-ordering.

The subsets of a given per A do form a complete sublattice of the lattice of all specs contained in A. The lub in this lattice is the normal SPEC-cup, but is denoted by $\cup$, the glb is the normal SPEC-cap except for the glb of an empty collection which is A. The glb is denoted by $\cap$. A nice property that this lattice has in common with the pid lattice is that composition is also the glb:

Lemma 5.13:

For pers A, B and C such that $B \subseteq A$ and $C \subseteq A$:

$$B{\cap}C = B \circ C$$

□

Proof:

$$\begin{array}{ll}
 & B{\cap}C \\
= & \quad \{ \ B{\cap}C = B{\sqcap}C, \ C \subseteq A \ \} \\
 & B \sqcap A{\circ}C \\
= & \quad \{ \ (4.9\text{a}), \ C \subseteq A \Rightarrow C{\sqsubseteq}A, \ B \subseteq A \Rightarrow B = B \circ A \ \} \\
 & (B{\sqcap}A) \circ C \\
= & \quad \{ \ B \subseteq A \ \} \\
 & B \circ C
\end{array}$$

□

It is sometimes useful to have a definition of $\subseteq$ that has only one conjunct and the following lemma supplies such a definition:

Lemma 5.14: (Subset)

$$A \subseteq B \quad \equiv \quad A \;=\; A{\gtrless} \circ B$$

□

Proof:

$$\begin{array}{cl}
 & A \;\; \subseteq \;\; B \\
\equiv & \quad \{ \text{ definition } \subseteq \; \} \\
 & A \;\sqsubseteq\; B \quad \wedge \quad A \circ B \;=\; A \\
\equiv & \quad \{ \;\Rightarrow\text{: } A{\gtrless} \circ\text{, domains; } \Leftarrow\text{: } A{\gtrless} \;\sqsubseteq\; I\text{, unit; anti-symmetry } \} \\
 & A \;\sqsubseteq\; A{\gtrless} \circ B \quad \wedge \quad A \circ B \sqsubseteq A \quad \wedge \quad A \;\sqsubseteq\; A \circ B \\
\equiv & \quad \{ \;\Rightarrow\text{: } A{\gtrless} \;\sqsubseteq\; A\text{; } \Leftarrow\text{: } A \circ\text{, } A \circ A \;=\; A\text{; } A \sqsubseteq A{\gtrless} \circ B \;\Rightarrow\; A \sqsubseteq A \circ B \;\} \\
 & A \;\sqsubseteq\; A{\gtrless} \circ B \quad \wedge \quad A{\gtrless} \circ B \;\sqsubseteq\; A \\
\equiv & \quad \{ \text{ anti-symmetry } \} \\
 & A \;=\; A{\gtrless} \circ B
\end{array}$$

□

The fact that the $\subseteq$ -ordering does not give information about quotient-forming suggests that we need another ordering on pers. Another reason for considering a different ordering is that the construction of inductively defined types is linked to solving recursive equations on types. A complete lattice would simplify this construction. The $\sqsubseteq$ -ordering on pers does give a complete lattice but this ordering says nothing about the relationship of pers seen as the representation of types, making it unusable for our purposes.

5.3.2 Pers as quotients

Taking a quotient is equivalent to dividing collections of elements into classes. This corresponds in the per representation to joining elements (classes) together to form the new elements of the quotient. Formally we can express that per A is a quotient of per B as follows:

(5.15) $\forall(X \mid X{\in}B \mid \exists(Y \mid Y{\in}A \mid X \;\sqsubseteq\; Y))$
(5.16) $A{\gtrless} \;\sqsubseteq\; B{\gtrless}$

Part (5.15) states that every element of B has been put in one of the classes of A, (5.16) says that everything in A has been formed from parts of B. The definition given here is only a usable definition in an extensional SPEC-calculus because of the use of $\in$ and quantifications, so we would like to have an equivalent definition that is also usable in ordinary SPEC-calculi. We are going to prove that the conjunction of (5.15) and (5.16) is equivalent to

(5.17) $B \;\sqsubseteq\; A \quad \wedge \quad A \circ B \;=\; A$

in an extensional SPEC-calculus. This definition is also usable in a non-extensional SPEC-calculus. The equivalence of the two definitions is proved by mutual implication. First we prove that the conjunction of (5.15) and (5.16) implies (5.17)

$$
\begin{array}{ll}
 & A \\
= & \qquad \{ \text{ extensionality } \} \\
 & \bigsqcup(Y \mathrel{¦} Y{\in}A \mathrel{¦} Y) \\
\sqsupseteq & \qquad \{ \text{ monotonicity } \} \\
 & \bigsqcup(X, Y \mathrel{¦} X{\in}B \quad \wedge \quad Y{\in}A \quad \wedge \quad X \sqsubseteq Y \mathrel{¦} X) \\
= & \qquad \{ \bullet\ \forall(X \mathrel{¦} X{\in}B \mathrel{¦} \exists(Y \mathrel{¦} Y{\in}A \mathrel{¦} X \sqsubseteq Y));\ \text{gen. range disjunction } \} \\
 & \bigsqcup(X \mathrel{¦} X{\in}B \mathrel{¦} X) \\
= & \qquad \{ \text{ extensionality } \} \\
 & B
\end{array}
$$

$$
\begin{array}{ll}
 & A \circ B \\
\sqsupseteq & \qquad \{ \text{ domains } \} \\
 & A \circ B{\gtrless} \\
\sqsupseteq & \qquad \{ \bullet\ A{\gtrless} \sqsubseteq B{\gtrless};\ \text{domains } \} \\
 & A \\
\sqsupseteq & \qquad \{ B \sqsubseteq A,\ A = A \circ A \ \} \\
 & A \circ B
\end{array}
$$

Now we prove that (5.17) implies (5.15) and (5.16). Conjunct (5.15) is proved by:

$$
\begin{array}{ll}
 & \exists(Y \mathrel{¦} Y{\in}A \mathrel{¦} X \sqsubseteq Y) \\
\Leftarrow & \qquad \{ \ \mathit{class}.(A \circ X \circ A) \text{ (see below), calculus } \} \\
 & A \circ X \circ A \in A \quad \wedge \quad X \sqsubseteq A \circ X \circ A \\
\equiv & \qquad \{ \ \mathit{class}.(A \circ X \circ A) \text{ (see below), definition } \in \ \} \\
 & A \circ X \circ A \circ A = A \circ X \circ A \quad \wedge \quad A \circ X \circ A \sqsubseteq A \quad \wedge \quad X \sqsubseteq A \circ X \circ A \\
\Leftarrow & \qquad \{ \ A \circ A = A,\ \bullet\ A \circ B = A,\ B \sqsubseteq A \ \} \\
 & A \circ X \circ A \sqsubseteq A \circ B \circ A \quad \wedge \quad X \sqsubseteq B \circ X \circ B \\
\Leftarrow & \qquad \{ \text{ monotonicity, calculus } \} \\
 & X \sqsubseteq B \quad \wedge \quad X \circ B = X \\
\Leftarrow & \qquad \{ \text{ definition } \in \ \} \\
 & X{\in}B
\end{array}
$$

There is still something left to prove in the proof of (5.15): $\mathit{class}.(A \circ X \circ A) \Leftarrow X{\in}B$. First we prove that $A \circ X \circ A \neq \perp\!\!\perp$:

$$
\begin{array}{ll}
 & A \circ X \circ A \\
\sqsupseteq & \qquad \{ \bullet\ B \sqsubseteq A \ \} \\
 & B \circ X \circ B \\
= & \qquad \{ \bullet\ X{\in}B \ \} \\
 & X \\
\neq & \qquad \{ \bullet\ X{\in}B \ \} \\
 & \perp\!\!\perp
\end{array}
$$

Now the remaining proof obligation, omitting all $\cup$'s because all specs in the proof are symmetric:

$$
\begin{array}{ll}
 & A \circ X \circ A \circ \top\!\!\top \circ A \circ X \circ A \\
= & \qquad \{ \ \mathit{square}.X \ \} \\
 & A \circ X \circ \top\!\!\top \circ X \circ A \circ \top\!\!\top \circ A \circ X \circ \top\!\!\top \circ X \circ A \\
= & \qquad \{ \ \text{cone rule, } X \circ A \neq \bot\!\!\bot \quad \Leftarrow \quad A \circ X \circ A \neq \bot\!\!\bot \ \} \\
 & A \circ X \circ \top\!\!\top \circ X \circ A \\
= & \qquad \{ \ \mathit{square}.X \ \} \\
 & A \circ X \circ A
\end{array}
$$

□

Conjunct (5.16) is proved by:

$$
\begin{array}{ll}
 & A\lessgtr \\
= & \qquad \{ \ \bullet A \circ B \ = \ A, \text{ domains } \} \\
 & (A \circ B)> \\
\sqsubseteq & \qquad \{ \ \text{domains } \} \\
 & B\lessgtr
\end{array}
$$

From the calculations above we can conclude that $B \ \sqsubseteq \ A \quad \wedge \quad A \circ B \ = \ A$ expresses that per A represents a quotient of per B. We introduce a special notation for this relationship between pers:

Definition 5.18: (Quotient Order)

$$A \lhd\!\!\mid B \quad \triangleq \quad B \ \sqsubseteq \ A \quad \wedge \quad A \circ B \ = \ A$$

□

The quotient order is a partial order on pers, but the pers do not form a lattice under the quotient order. Lattices with the quotient order can be constructed if we restrict ourselves to pers with same domain. This follows from the following lemma:

Lemma 5.19: (Quotient Order)

$$A \lhd\!\!\mid B \quad \equiv \quad B \ \sqsubseteq \ A \quad \wedge \quad A\lessgtr \ = \ B\lessgtr$$

□

The quotient order coincides with the reversed SPEC-ordering for a collection of specs with the same domain. For proving (5.19) it is sufficient to prove
$B \ \sqsubseteq \ A \ \Rightarrow \ (A \circ B \ = \ A \quad \equiv \quad A\lessgtr \ = \ B\lessgtr)$:

$$
\begin{array}{ll}
 & A \circ B \ = \ A \\
\equiv & \qquad \{ \ \Leftarrow\!: \ \bullet \ B \ \sqsubseteq \ A, \text{ Hence } A \circ B \sqsubseteq A \circ A = A = A \circ B\lessgtr \ \sqsubseteq \ A \circ B; \\
 & \qquad\quad \Rightarrow\!: \ A \circ B \circ B\lessgtr = A \circ B \ \} \\
 & A \circ B\lessgtr \ = \ A \\
\equiv & \qquad \{ \ \Rightarrow\!: \ \bullet \ B \ \sqsubseteq \ A, \text{ Hence } B\lessgtr \ \sqsubseteq \ A\lessgtr \ \sqsubseteq \ (A \circ B\lessgtr)> \ \sqsubseteq \ B\lessgtr; \ \Leftarrow\!: \text{ domains } \} \\
 & A\lessgtr \ = \ B\lessgtr
\end{array}
$$

□

Lattices with the quotient ordering can be constructed using the following lemma:

Lemma 5.20:(Quotient Lattice)

For per A, $(\{B \mid B \lhd\!\mid A \mid B\}, \lhd\!\mid)$ and $(\{B \mid A \lhd\!\mid B \mid B\}, \lhd\!\mid)$ are complete lattices.

□

Because all pers in the two sets have the same domain ($A\ltimes$), the $\lhd\!\mid$ ordering is the same as the reversed SPEC-ordering $\sqsupseteq$. This means that we can use the glb and lub on pers from the previous chapter for these lattices. The lub is the glb from the pers (ordering is reversed), that is, the normal SPEC-cap except for the lub of an empty collection, which is A for $(\{B \mid B \lhd\!\mid A \mid B\}, \lhd\!\mid)$ and $A\ltimes$ for $(\{B \mid A \lhd\!\mid B \mid B\}, \lhd\!\mid)$. The glb is the lub from the pers, that is, the transitive closure of the SPEC-lub except for the glb of an empty collection, which is $A \circ \top\!\top \circ A$ for $(\{B \mid B \lhd\!\mid A \mid B\}, \lhd\!\mid)$ and A for $(\{B \mid A \lhd\!\mid B \mid B\}, \lhd\!\mid)$.

Another alternative for the definition of the quotient ordering that we will sometimes use is given in the following lemma:

Lemma 5.21:(Quotient Order)

$$A \lhd\!\mid B \quad \equiv \quad A \circ B \;=\; A \quad \wedge \quad A\ltimes \;=\; B\ltimes$$

□

We prove this by showing equivalence with (5.19). For this it is sufficient to prove $A\ltimes \;=\; B\ltimes \;\Rightarrow\; (A \circ B \;=\; A \quad \equiv \quad B \sqsubseteq A)$:

$$\begin{array}{ll}
 & A \circ B \;=\; A \\
\equiv & \qquad \{ \;\bullet\; A\ltimes \;=\; B\ltimes, \text{Hence } A \circ B \;\sqsupseteq\; A \circ B\ltimes \;=\; A \circ A\ltimes \;=\; A \;\} \\
 & A \circ B \sqsubseteq A \\
\equiv & \qquad \{ \;\Leftarrow\text{: } A\,\circ;\; \Rightarrow\text{: } A\ltimes \sqsubseteq A \;\} \\
 & A\ltimes \circ B \sqsubseteq A \\
\equiv & \qquad \{ \;\bullet\; A\ltimes \;=\; B\ltimes, \text{domains} \;\} \\
 & B \sqsubseteq A
\end{array}$$

□

We remarked at the end of section 5.3.1 that the subset ordering does not result in a complete lattice on the set of all pers. The quotient lattices have the same problem, namely too many pers are incomparable, and the set of all pers does not form a complete lattice under the $\lhd\!\mid$ order. The $\lhd\!\mid$ order only gives information about quotient formation and is not usable in situations where subsets are constructed.

We need an ordering on pers that encompasses both subset and quotient construction and gives us a complete lattice. An ordering satisfying these criteria will be constructed in the remainder of this section and it turns out that this new ordering can be viewed as a combination of the subset and quotient ordering.

5.3.3 Decomposing pers

Calculations with pers can often be simplified if we write the pers as the composition $A \circ E$ ($:=\; E \circ A$) of two components:

- A: the domain that it is operating on, i.e. a subset of the universe. This can be represented by a pid.

- E: the quotient map extended to the whole universe, i.e. a partitioning of the universe. This can be represented by a *partition*, a per containing I.

We extend the quotient map to the universe because this allows us to restrict our attention to total quotient maps, and thereby simplifies the calculations. The idea of decomposing a per into a partition and a pid was originally suggested by Jaap van de Woude and some of the results and proofs are his. We first formalize the notions of partition and extension in the SPEC-calculus:

Definition 5.22: (Partition)

A spec E is a *partition* iff $per.E \;\wedge\; I \sqsubseteq E$.
□

Since partitions are also pers we will use capitals early in the alphabet to denote partitions, preferably E and D. This is just a preference, not a notational convention.

Definition 5.23: (Extension)

A partition E is an *extension* of per A iff $A \subseteq E$.
□

Using (5.14) we see that the definition of E being an extension is equivalent to

(5.24) $A \;=\; A_{\ltimes} \circ E$

and that per A is written as the composition of a pid and a partition.

The decomposition of pers is not unique since the extension of the quotient-map is not unique. Examining the two extreme extensions we find that the smallest extension under the $\sqsubseteq$-ordering is $A \sqcup I$, but this extension doesn't seem to be useful. The extension that we will use is the largest extension; one class containing everything outside the domain is added. We introduce a new operator for this extension:

Definition 5.25: (Greatest Extension)

For $per.A$ we define

$$A_{\rtimes} \;\triangleq\; A/A \sqcap (A/A)_{\cup}$$

□

The fact that $A_{\rtimes}$ is the greatest extension can be proved by instantiating the next lemma:

Lemma 5.26: (Extension)

For all pers A and symmetric specs X:

$$X \circ A_{\ltimes} \;=\; A \quad\equiv\quad A \sqsubseteq X \quad\wedge\quad X \sqsubseteq A_{\rtimes}$$

□

Proof:

$$
\begin{array}{ll}
 & X \circ A^{\times} \ = \ A \\
\equiv & \qquad \{ \ \text{anti-symmetry} \ \} \\
 & X \circ A^{\times} \ \sqsubseteq \ A \quad \wedge \quad A \ \sqsubseteq \ X \circ A^{\times} \\
\equiv & \qquad \{ \ \Rightarrow\!: \circ A,\ A^{\times} \ \sqsubseteq \ I;\ \Leftarrow\!:\ A^{\times} \ \sqsubseteq \ A, \circ A^{\times}; \text{domains} \ \} \\
 & X \circ A \ \sqsubseteq \ A \quad \wedge \quad A \ \sqsubseteq \ X \\
\equiv & \qquad \{ \ \text{factors} \ \} \\
 & X \ \sqsubseteq \ A/A \quad \wedge \quad A \ \sqsubseteq \ X \\
\equiv & \qquad \{ \ \bullet\, X \ = \ X^{\cup}\,, \text{reverse, definition } {\searrow\!\!\!\swarrow} \ \} \\
 & X \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow} \quad \wedge \quad A \ \sqsubseteq \ X
\end{array}
$$

□

The result that every extension of A is contained in $A^{\searrow\!\!\!\swarrow}$ follows immediately from this lemma. We prove that $A^{\searrow\!\!\!\swarrow}$ is itself an extension by choosing appropriate instantiations for X. For $X \ = \ A \sqcup I$ we obtain as result

(5.27) $A \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow} \quad \wedge \quad I \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow}$

Proof:

$$
\begin{array}{ll}
 & A \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow} \quad \wedge \quad I \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow} \\
\equiv & \qquad \{ \ \text{lub} \ \} \\
 & A \ \sqsubseteq \ A \sqcup I \quad \wedge \quad A \sqcup I \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow} \\
\equiv & \qquad \{ \ A \sqcup I \text{ is symmetric, (5.26)} \ \} \\
 & (A \sqcup I) \circ A^{\times} \ = \ A \\
\equiv & \qquad \{ \ \text{distribution, domains} \ \} \\
 & A \sqcup A^{\times} \ = \ A \\
\equiv & \qquad \{ \ A^{\times} \ \sqsubseteq \ A \ \} \\
 & \mathit{true}
\end{array}
$$

□

Instantiating (5.26) with $X \ = \ A^{\searrow\!\!\!\swarrow}$ gives us (using $A \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow}$):

(5.28) $A^{\searrow\!\!\!\swarrow} \circ A^{\times} \ = \ A$

Instantiation with $X \ = \ A^{\searrow\!\!\!\swarrow} \circ A^{\searrow\!\!\!\swarrow}$ is used in proving the transitivity of $A^{\searrow\!\!\!\swarrow}$:

(5.29) $A^{\searrow\!\!\!\swarrow} \circ A^{\searrow\!\!\!\swarrow} \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow}$

Proof:

$$
\begin{array}{ll}
 & A^{\searrow\!\!\!\swarrow} \circ A^{\searrow\!\!\!\swarrow} \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow} \\
\equiv & \qquad \{ \ A \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow},\ A \ = \ A \circ A \ \} \\
 & A \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow} \circ A^{\searrow\!\!\!\swarrow} \quad \wedge \quad A^{\searrow\!\!\!\swarrow} \circ A^{\searrow\!\!\!\swarrow} \ \sqsubseteq \ A^{\searrow\!\!\!\swarrow} \\
\equiv & \qquad \{ \ (5.26) \ \} \\
 & A^{\searrow\!\!\!\swarrow} \circ A^{\searrow\!\!\!\swarrow} \circ A^{\times} \ = \ A \\
\equiv & \qquad \{ \ A^{\searrow\!\!\!\swarrow} \circ A^{\times} \ = \ A \ = \ A^{\times} \circ A \ \}
\end{array}
$$

$$
\begin{array}{ll}
 & A{\swarrow\!\!\!\!\!\searrow} \circ A{\gtrless} \circ A \ = \ A \\
\equiv & \qquad \{ \ A{\swarrow\!\!\!\!\!\searrow} \circ A{\gtrless} \ = \ A, A \ = \ A \circ A \ \} \\
 & \mathit{true}
\end{array}
$$

$\square$

The definition of $A{\swarrow\!\!\!\!\!\searrow}$ guarantees symmetry and combined with transitivity (5.29) and its containing I (5.27) we have proved that $A{\swarrow\!\!\!\!\!\searrow}$ is a partition.

Not every composition of a pid and a partition corresponds to a per. The composition of a pid and a partition is always transitive but not always symmetric. A necessary and sufficient condition is given in

Lemma 5.30:

For $pid.A$ and $partition.E$ we have

$$per.(A \circ E) \quad \equiv \quad A \circ E \ \sqsubseteq \ \top\!\!\top \circ A$$

$\square$

Proof:

$$
\begin{array}{ll}
 & per.(A \circ E) \\
\equiv & \qquad \{ \ transitive.(A \circ E), \text{ reverse, symmetry of pids and partitions } \ \} \\
 & A \circ E \ = \ E \circ A \\
\equiv & \qquad \{ \ \text{anti-symmetry, reverse, symmetry of pids and partitions } \ \} \\
 & A \circ E \ \sqsubseteq \ E \circ A \\
\equiv & \qquad \{ \ \Leftarrow: A \ \sqsubseteq \ I; \Rightarrow: A \ \circ, A \circ A \ = \ A \ \} \\
 & A \circ E \ \sqsubseteq \ A \circ E \circ A \\
\equiv & \qquad \{ \ \text{domains} \ \} \\
 & A \circ E \ \sqsubseteq \ A \circ E \sqcap \top\!\!\top \circ A \\
\equiv & \qquad \{ \ \text{cap} \ \} \\
 & A \circ E \ \sqsubseteq \ \top\!\!\top \circ A
\end{array}
$$

$\square$

5.3.4 The per lattice

Let's take a closer look at pids and partitions. We have shown in the previous chapter that the pids form a complete lattice under the $\sqsubseteq$ -order, and from lemma (5.20) it follows that the partitions, the pers $\lhd I$, also form a complete lattice. The ordering on the partitions is the reversed SPEC-ordering $\sqsupseteq$.

Given the orderings on pids and partitions we can construct combined orderings on pers decomposed into pids and partitions. In principle there are two possible combined orderings, for pers A and B:

$$
\begin{array}{l}
A{\gtrless} \ \sqsubseteq \ B{\gtrless} \quad \wedge \quad A{\swarrow\!\!\!\!\!\searrow} \ \sqsubseteq \ B{\swarrow\!\!\!\!\!\searrow} \\
A{\gtrless} \ \sqsubseteq \ B{\gtrless} \quad \wedge \quad A{\swarrow\!\!\!\!\!\searrow} \ \sqsupseteq \ B{\swarrow\!\!\!\!\!\searrow}
\end{array}
$$

The first of these two orderings is not very interesting because we can prove the following result in a complemented SPEC-calculus:

$$A{\ltimes} \sqsubseteq B{\ltimes} \quad \wedge \quad A{\rtimes} \sqsubseteq B{\rtimes} \qquad \Rightarrow \qquad B{\ltimes} = I \quad \vee \quad A{\ltimes} = B{\ltimes}$$

This severely limits the applicability of the first ordering. The other ordering does have desirable algebraic properties, most importantly that we have a complete lattice. We first introduce a notation for our ordering:

Definition 5.31:(Per Order)

$$A \lhd B \quad \triangleq \quad A{\ltimes} \sqsubseteq B{\ltimes} \wedge B{\rtimes} \sqsubseteq A{\rtimes}$$

□

This ordering has a very meaningful interpretation if we view pers as types with classes as elements. The two conjuncts of (5.31) have two different effects: $B{\rtimes} \sqsubseteq A{\rtimes}$ means that A has at least the same equalities on the elements of the universe as B, but might have more. In the example of the integers and pairs of natural numbers in section 5.2 we see that we have introduced extra equalities. Parts of the universe that were in the domain of B but are no longer in the domain of A are added to the extra class in $B{\rtimes}$ that is outside B. The other conjunct $A{\ltimes} \sqsubseteq B{\ltimes}$ states that A is only defined on (a part of) the domain of B, parts outside the domain of B cannot be added. In the example of the even integers we see that the integers are restricted to those pairs of naturals with an even sum. The per order corresponds to combining two important methods of type construction: adding laws and imposing restrictions. $A \lhd B$ means that A can be formed from B using these two methods.

The definition of the ordering given above has advantages for investigating lattice properties but for most other uses we prefer the following equivalent definition:

Definition 5.32:(Per Order)

$$A \lhd B \quad \equiv \quad A \circ B = A$$

□

The equivalence of (5.31) and (5.32) is proved by showing that the definition of (5.31) implies that of (5.32), followed by proving that the definition of (5.32) implies that of (5.31).

$$
\begin{array}{ll}
 & A \circ B \\
= & \qquad \{ \ (5.28) \ \} \\
 & A \circ B{\ltimes} \circ B{\rtimes} \\
= & \qquad \{ \ \bullet\ A{\ltimes} \sqsubseteq B{\ltimes}, \text{domains} \ \} \\
 & A \circ B{\rtimes} \\
\sqsubseteq & \qquad \{ \ \bullet\ B{\rtimes} \sqsubseteq A{\rtimes} \ \} \\
 & A \circ A{\rtimes} \\
= & \qquad \{ \ A \circ A{\ltimes} = A, (5.28) \ \} \\
 & A \\
= & \qquad \{ \ \bullet\ A{\ltimes} \sqsubseteq B{\ltimes}, \text{domains} \ \} \\
 & A \circ B{\ltimes} \\
\sqsubseteq & \qquad \{ \ B{\ltimes} \sqsubseteq B \ \} \\
 & A \circ B
\end{array}
$$

$$
\begin{array}{ll}
 & A{>} \\
= & \quad \{ \ \bullet\ A \circ B \ = \ A \ \} \\
 & (A \circ B){>} \\
\sqsubseteq & \quad \{ \ \text{domains} \ \} \\
 & B{>}
\end{array}
$$

$$
\begin{array}{ll}
 & B{<} \ \sqsubseteq \ A{<} \\
\equiv & \quad \{ \ (5.25), \text{symmetry of } B{<} \ \} \\
 & B{<} \circ A \ \sqsubseteq \ A \\
\equiv & \quad \{ \ \bullet\ A \circ B \ = \ A, \text{Hence } A{>} \ \sqsubseteq \ B{>} \ \} \\
 & B{<} \circ B{>} \circ A \ \sqsubseteq \ A \\
\equiv & \quad \{ \ \bullet\ A \circ B \ = \ A,\ B{<} \circ B{>} \ = \ B \ \} \\
 & \mathit{true}
\end{array}
$$

□

We prove that the pers form a complete lattice under the $\lhd$ -ordering by first showing that the decompositions of pers form a complete lattice followed by establishing a Galois connection between this lattice and the poset of pers under the $\lhd$ -ordering. We can then conclude using lemma (2.26) that the pers form a complete lattice under the $\lhd$ -ordering.

The first proof obligation for the proof sketched above is:

$$
\begin{array}{l}
(\{A, E \mid \mathit{pid}.A \quad \wedge \quad \mathit{partition}.E \quad \wedge \quad \mathit{per}.(A \circ E) \mid (A, E)\}, (\sqsubseteq, \sqsupseteq)) \\
\text{is a complete lattice}
\end{array}
$$

Proof: The lub of the lattice is the pairing of the lub of the pid-lattice under the $\sqsubseteq$ -ordering and the lub of the partition-lattice under the $\sqsupseteq$ -ordering (glb under the $\sqsubseteq$ -ordering). For proving this it is sufficient to show that the set is closed under the combined lub operation, for $\mathcal{S} \ \subseteq \ \{A, E \mid \mathit{pid}.A \ \wedge \ \mathit{partition}.E \ \wedge \ \mathit{per}.(A \circ E) \mid (A, E)\}$, $\mathcal{A} \ = \ \{A, E \mid (A, E) {\in} \mathcal{S} \mid A\}$, $\mathcal{E} \ = \ \{A, E \mid (A, E) {\in} \mathcal{S} \mid E\}$:

$$
\begin{array}{ll}
 & \bigsqcup \mathcal{A} \circ \bigsqcap \mathcal{E} \\
= & \quad \{ \ \text{distributivity, definition } \mathcal{A} \ \} \\
 & \bigsqcup (A, E \mid (A, E) {\in} \mathcal{S} \mid A \circ \bigsqcap \mathcal{E}) \\
\sqsubseteq & \quad \{ \ \text{definition } \mathcal{E}, \text{cap} \ \} \\
 & \bigsqcup (A, E \mid (A, E) {\in} \mathcal{S} \mid A \circ E) \\
\sqsubseteq & \quad \{ \ \mathit{per}.(A \circ E),\ (5.30) \ \} \\
 & \bigsqcup (A, E \mid (A, E) {\in} \mathcal{S} \mid \top\!\!\top \circ A) \\
= & \quad \{ \ \text{distributivity} \ \} \\
 & \top\!\!\top \circ \bigsqcup \mathcal{A}
\end{array}
$$

Using (5.30) we conclude that $\bigsqcup \mathcal{A} \circ \bigsqcap \mathcal{E}$ is a per and that the decompositions form a complete lattice. The next step in proving that the pers form a complete lattice under the $\lhd$ -ordering is the construction of a Galois connection between the pers and the decompositions: for pid A, per B and partition E such that $\mathit{per}.(A \circ E)$

$$
\begin{array}{ll}
& (B{\times}, B{\rtimes})(\sqsubseteq, \sqsupseteq)(A, E) \\
\equiv & \qquad \{\ \text{pairing}\ \} \\
& B{\times} \sqsubseteq A \quad \wedge \quad E \sqsubseteq B{\rtimes} \\
\equiv & \qquad \{\ \text{definition } \rtimes\text{, cap, reverse, symmetry of } E\text{, factors}\ \} \\
& B{\times} \sqsubseteq A \quad \wedge \quad E \circ B \sqsubseteq B \\
\equiv & \qquad \{\ \text{domains, } I \sqsubseteq E\text{, symmetry of } B \text{ and } E\ \} \\
& B \circ A = B \quad \wedge \quad B \circ E = B \\
\equiv & \qquad \{\ \Leftarrow: B \circ E \circ A \circ A = B \circ E \circ A,\ B \circ A \circ E \circ E = B \circ A \circ E; \\
& \qquad\ \ \Rightarrow: \text{substitution}\ \} \\
& B \circ E \circ A = B \quad \wedge \quad B \circ A \circ E = B \\
\equiv & \qquad \{\ per.(A \circ E)\text{, definition } \lhd\ \} \\
& B \lhd A \circ E
\end{array}
$$

From the fact that every per can be written as the composition of a pid and a partition it follows now, using (2.26), that the pers form a complete lattice. Application of lemma (2.26) also has as condition that the pers with $\lhd$ form a poset but this is trivial to establish and left to the reader. As conclusion of the proofs above we have proved the following theorem:

Theorem 5.33: (Per-Lattice)

The pers form a complete lattice under the $\lhd$ -ordering
□

The bottom of the lattice is $\perp\!\!\!\perp$ and the top is I. The lub and glb of the lattice can be expressed using other constructs from the SPEC-calculus, but as these explicit formulations are not suitable for manipulation and are also not needed in the remainder of this thesis, we omit such formulations here.

The glb operator of the lattice plays an important role in our theory about inductive datatypes with laws and therefore we introduce a special notation for it:

Definition 5.34: (Per-glb)

The infix operator $\wedge$ is defined by, for all pers A, B and C:

$$A \lhd B{\wedge}C \quad \equiv \quad A \lhd B \quad \wedge \quad A \lhd C$$

□

We do not know an explicit formula for $A{\wedge}B$ that is convenient to work with for all A's and B's, but for some A's and B's we can give a practical formula. Two "practical" formulas are given in the next two lemma's:

Lemma 5.35: (Per-glb)

For pers A and B such that $A \circ B = B \circ A$ we have $A{\wedge}B = A \circ B$
□

It is easy to see that $A \circ B = B \circ A$ implies that $A \circ B$ is a per (left to the reader). The fact that $A{\wedge}B = A \circ B$ is proved using indirect equality:

$$
\begin{array}{ll}
 & X \ \lhd\ A_{\wedge}B \\
\equiv & \qquad \{\ \text{definition } {}_{\wedge}\ \} \\
 & X \lhd A \quad \wedge \quad X \lhd B \\
\equiv & \qquad \{\ \text{definition } \lhd\ \} \\
 & X \circ A \ = \ X \quad \wedge \quad X \circ B \ = \ X \\
\equiv & \qquad \{\ \Leftarrow\!:\ X \circ B \circ A \circ A \ = \ X \circ B \circ A,\ X \circ A \circ B \circ B \ = \ X \circ A \circ B; \\
 & \qquad\quad \Rightarrow\!:\ \text{substitution}\ \} \\
 & X \circ B \circ A \ = \ X \quad \wedge \quad X \circ A \circ B \ = \ X \\
\equiv & \qquad \{\ \bullet\ A \circ B \ = \ B \circ A\ \} \\
 & X \circ A \circ B \ = \ X \\
\equiv & \qquad \{\ \text{definition } \lhd\ \} \\
 & X \ \lhd\ A \circ B
\end{array}
$$

□

Lemma 5.36:(Per-glb)

For pers A and B such that $A\between \ = \ B\between$ we have $A_{\wedge}B \ = \ (A \sqcup B)^{+}$ and $(A_{\wedge}B)\between \ = \ A\between$

□

This is also proved by indirect equality, but this time with a per X satisfying $X\between \ = \ A\between$:

$$
\begin{array}{ll}
 & X \ \lhd\ A_{\wedge}B \\
\equiv & \qquad \{\ \text{definition } {}_{\wedge}\ \} \\
 & X \lhd A \quad \wedge \quad X \lhd B \\
\equiv & \qquad \{\ \bullet\ X\between \ = \ A\between,\ \bullet\ A\between \ = \ B\between;\ (5.21)\ \ \} \\
 & X \lhd\!|\, A \quad \wedge \quad X \lhd\!|\, B \\
\equiv & \qquad \{\ \text{quotient lattice}\ \} \\
 & X \ \lhd\!|\ (A \sqcup B)^{+} \\
\equiv & \qquad \{\ \bullet\ X\between \ = \ A\between,\ \bullet\ A\between = B\between;\ (5.21),\ A\between = B\between \ \Rightarrow\ (A \sqcup B)^{+}\between = A\between\ \ \} \\
 & X \ \lhd\ (A \sqcup B)^{+}
\end{array}
$$

The calculation above is only a valid proof of $A_{\wedge}B \ = \ (A \sqcup B)^{+}$ if $(A_{\wedge}B)\between \ = \ A\between$ and $(A \sqcup B)^{+}\between \ = \ A\between$. The latter is easy to prove and left to the reader. The former follows from $A\between \circ \top\!\!\top \circ A\between \ \lhd\ A_{\wedge}B \ \lhd\ A$ and some simple domain properties. The fact that $A_{\wedge}B \ \lhd\ A$ follows directly from the definition of a glb whereas $A\between \circ \top\!\!\top \circ A\between \ \lhd\ A_{\wedge}B$ is proved by:

$$
\begin{array}{ll}
 & A\between \circ \top\!\!\top \circ A\between \ \lhd\ A_{\wedge}B \\
\equiv & \qquad \{\ \text{definition } {}_{\wedge}\ \} \\
 & A\between \circ \top\!\!\top \circ A\between \ \lhd\ A \quad \wedge \quad A\between \circ \top\!\!\top \circ A\between \ \lhd\ B \\
\equiv & \qquad \{\ \text{definition } \lhd\ \} \\
 & A\between\circ\top\!\!\top\circ A\between \circ A \ = \ A\between\circ\top\!\!\top\circ A\between \quad \wedge \quad A\between\circ\top\!\!\top\circ A\between \circ B \ = \ A\between\circ\top\!\!\top\circ A\between \\
\Leftarrow & \qquad \{\ \text{domains},\ \bullet\ A\between \ = \ B\between,\ \text{Leibniz}\ \} \\
 & \top\!\!\top \circ B\between \ \circ\ B \ = \ \top\!\!\top \circ A\between \\
\equiv & \qquad \{\ \bullet\ A\between \ = \ B\between,\ \text{domains}\ \}
\end{array}
$$

true

□

Earlier we mentioned that the per order could be seen as a combination of the subset order and the quotient order. This claim is substantiated by the following lemma:

Lemma 5.37:(Per Order)

(a) $A \lhd B \;\equiv\; \exists(C \;\vdots\vdots\; A \subseteq C \;\wedge\; C \lhd\!| \, B)$

(b) $A \lhd B \;\equiv\; \exists(C \;\vdots\vdots\; A \lhd\!| \, C \;\wedge\; C \subseteq B)$

□

The proofs of this lemma in the $\Leftarrow$ direction are simple because both the subset and quotient ordering imply the per order and the lhs follows then from the transitivity of the per order. For the proofs in the $\Rightarrow$ direction we provide instantiations for C. For (5.37a) we choose $A \sqcup B$; for (5.37b) there is no choice, the only possible instantiation for C being $A \sqcap B$. The subset and quotient order definitions have two conjuncts, a per order part and a SPEC-lattice order part. The choices made above fulfill the requirements for the SPEC-lattice order parts immediately. The per order parts proof obligation is to prove, assuming $A \lhd B$, that $A \lhd A \sqcup B \lhd B$ and $A \lhd A \sqcap B \lhd B$. This can be done by distribution over cup or cap followed by using $A \circ B = A$.

We end this section with some miscellaneous properties of the orderings on pers that are needed later in this thesis:

Lemma 5.38:(Per Orderings)

(a) $A \lhd B \;\Rightarrow\; A{\times} \lhd B{\times}$

(b) $A \lhd B \;\wedge\; C \lhd D \;\Rightarrow\; A \circ C \lhd B \circ D$

(c) $A \subseteq B \;\equiv\; B \circ A{\times} \lhd A \lhd B$

(d) $A \lhd C \;\wedge\; B \subseteq C \;\Rightarrow\; A_{\wedge} B \subseteq A$

(e) $A \lhd\!| B \;\equiv\; A \lhd B \;\wedge\; A{\times} = B{\times}$

(f) $A \lhd\!| B \;\equiv\; B \circ \top\!\!\top \circ B \lhd A \lhd B$

(g) $A \lhd\!| B \lhd\!| C \;\equiv\; A \lhd B \lhd C \;\wedge\; A \lhd\!| C$

□

We give only a sketch of the proofs needed for this lemma. The proofs are easy and the details are left as an exercise. For (5.38a) we use that the $\lhd$ ordering coincides with $\sqsubseteq$ on pids. For (5.38b) we use that pers are symmetric. Note that (5.38b) only makes sense if $A \circ C$ and $B \circ D$ are pers. This fact (in particular $A \circ C$ is symmetric) is used in its proof. The proof of (5.38c) uses lemma (5.14). Note that $A \lhd B \;\Rightarrow\; per.(B \circ A{\times})$. The proof of (5.38d) is somewhat more complicated than the other proofs. The essential idea is proving $A_{\wedge} B \;=\; (A_{\wedge} B){\times} \circ A$ by mutual inclusion in the per lattice. Part (5.38e) is proved by proving $A \sqsupseteq B \;\Leftarrow\; A \lhd B \;\wedge\; A{\times} = B{\times}$. The proof of (5.38f) uses $B \circ \top\!\!\top \circ B \;=\; A \circ \top\!\!\top \circ A$ for the "$\Rightarrow$" direction and (5.38a) for the other direction. The proof of (5.38g) also uses (5.38a).

5.3.5 Relational type-judgements and domain operators

In the previous section we have shown how we can use pers to represent types and we introduced some partial orders on pers. We will show in this section how we can use pers to type specs and introduce some operators for this purpose.

The elements of a per are its equivalence classes and stating that a per is a domain of a spec should mean that the spec respects the equivalences in the per. This means in the set-theoretic model that elements of the universe that are equivalent in the per should be handled equivalently by the spec. For right domains this can be expressed as follows, where S is a spec and x is an equivalence class represented as a class:

$$\forall(p, q, r \mid (q, r) \in x \mid (p, q) \in S \equiv (p, r) \in S)$$

In the SPEC-calculus this becomes $S \sqsupseteq S \circ x$. The equivalence between the formula in the model and the spec-formula is easy to see: because x is symmetric we can replace the $\equiv$ by $\Rightarrow$ and then we shunt $(p, q) \in S$ to the domain of the quantification. We call a class satisfying the condition a right equivalence class of S and of course we also have the dual notion of a left equivalence class. Our definition is thus:

Definition 5.39:(Left and Right Equivalence Class)

For spec R and class x we define:

$$\begin{array}{rcl} x \textit{ is a left equivalence class of } R & \triangleq & R \sqsupseteq x \circ R \\ x \textit{ is a right equivalence class of } R & \triangleq & R \sqsupseteq R \circ x \end{array}$$
□

An example: consider the relation between pairs of natural numbers defined by $((i, j), (m, n)) \in T \equiv i + 2n = j + 2m$. This represents the doubling function on integers. The classes of the integers are both left and right equivalence classes of T. Note that, although T only relates even integers on the lhs with integers on the rhs, all the integers are still left equivalence classes. For a class x representing an odd integer we simply have $x \circ T = \perp\!\!\perp$, so this is certainly included in T. On the other hand the even integers do not completely cover the rhs of T, all integers are needed to completely cover the rhs.

We want to capture this kind of situation in a definition for domains. An appropriate definition should ensure that both the integers and the even integers are left domains of T, but that only the integers form a right-domain. The considerations above lead to a definition like: for per A (this represents a type, a collection of classes) and spec R we have that A is a right-domain iff

$$\forall(x \mid x \in A \mid x \textit{ is a right equivalence class of } R) \wedge A \sqsupseteq R{>}$$

The second conjunct ensures that A completely covers the rhs of R. The formula above is not very practical because it contains a quantification, but removing this quantification is simple and we express the notion of a domain as a type-judgement:

Definition 5.40:(Domains)

$$R\colon A\text{—} \;\triangleq\; A \circ R = R$$

$$R\colon —A \;\triangleq\; R \circ A = R$$

□

We pronounce $R\colon A—$ as "A is a *left domain* of R" or "R respects A on the left hand side" and $R\colon —A$ as "A is a *right domain* of R" or "R respects A on the right hand side". We can combine judgements about left and right domains in one judgement: $R\colon A—$ and $R\colon —B$ combine to $R\colon A—B$. Another way of seeing a judgement with only one domain is simply assuming that the missing domain is I, and this will be our convention for type-judgements: all judgements are binary and missing pers are assumed to be I.

We only give one type-deduction rule in this section, other deduction rules can be found in the last section of this chapter. The proof of the following lemma is trivial and left to the reader.

Lemma 5.41: (Domains)

$$\begin{array}{lcl} R\colon A— & \Leftarrow & R\colon B— \;\wedge\; B \triangleleft A \\ R\colon —A & \Leftarrow & R\colon —B \;\wedge\; B \triangleleft A \end{array}$$

□

Now we have defined domains we look for characterizations of the domains belonging to a given spec. We will show that for a given spec there exist both least and greatest domains under both the $\sqsupseteq$ and $\triangleleft$ order and that the collection of all domains is completely characterized by the least domain under the $\triangleleft$ order. We will do our calculations only for right domains, but everything can be dualized to left domains.

In the previous chapter we introduced the $>$ operator using a Galois correspondence. An alternative definition can be given using the following lemma:

Lemma 5.42: (Least Domains)

$$R> \;=\; \sqcap(A \mid R \circ A = R \mid A)$$

□

This lemma clearly shows the reason why we call $R>$ the least right domain of R. One inclusion is for free since $R \circ R> = R$. We prove the other inclusion:

$$\begin{array}{ll} & \sqcap(A \mid R \circ A = R \mid A) \;\sqsupseteq\; R> \\ \equiv & \qquad \{ \text{ glb } \} \\ & \forall(A \mid R \circ A = R \mid A \sqsupseteq R>) \\ \equiv & \qquad \{ \text{ domains, (4.20) } \} \\ & \forall(A \mid R \circ A = R \mid \top\!\top \circ A \sqsupseteq R) \\ \equiv & \qquad \{ \text{ top, monotonicity } \} \\ & \mathit{true} \end{array}$$

□

We define the greatest domain in a similar way:

Definition 5.43:(Greatest Domains)

$$R\searrow \;\triangleq\; \bigsqcup(A \;\vdots\; R\circ A = R \;\vdots\; A)$$

□

We pronounce $R\searrow$ as "greatest right domain of R". There are two problems with this definition: it is not obvious that $\bigsqcup(...)$ is a per and this definition is in terms of a quantification and we would prefer a simple closed formula. A closed formula is given in the following lemma and in the proof of this lemma we also prove that definition 5.43 indeed defines a per-valued operator.

Lemma 5.44:(Greatest Domains)

$$R\searrow \;=\; R\backslash R \sqcap (R\backslash R)^{\cup}$$

□

$R\searrow$ is clearly a per since it is the intersection of transitive specs and it is its own reverse. We fulfill our proof obligations by first showing that $R\searrow$ is a right domain of R, giving us the $\sqsubseteq$ inclusion:

$$
\begin{array}{ll}
 & R \circ (R\backslash R \sqcap (R\backslash R)^{\cup}) \\
\sqsupseteq & \quad \{ \; R\backslash R \sqsupseteq I \; \} \\
 & R \\
\sqsupseteq & \quad \{ \; \text{cancellation} \; \} \\
 & R \circ R\backslash R \\
\sqsupseteq & \quad \{ \; \text{calculus} \; \} \\
 & R \circ (R\backslash R \sqcap (R\backslash R)^{\cup})
\end{array}
$$

Proof "$\sqsupseteq$":

$$
\begin{array}{ll}
 & R\backslash R \sqcap (R\backslash R)^{\cup} \sqsupseteq \sqcup(A \;\vdots\; R\circ A = R \;\vdots\; A) \\
\equiv & \quad \{ \; \text{symmetry} \; \} \\
 & R\backslash R \sqsupseteq \sqcup(A \;\vdots\; R\circ A = R \;\vdots\; A) \\
\equiv & \quad \{ \; \text{factors, distribution} \; \} \\
 & R \sqsupseteq \sqcup(A \;\vdots\; R\circ A = R \;\vdots\; R\circ A) \\
\equiv & \quad \{ \; \text{substitution} \; \} \\
 & R \sqsupseteq \sqcup(A \;\vdots\; R\circ A = R \;\vdots\; R) \\
\equiv & \quad \{ \; \text{calculus} \; \} \\
 & true
\end{array}
$$

□

The greatest domain is a special case of what is called the symmetric quotient in for example [54]. There the relation $syq(R,S)$ is defined as the largest symmetric relation X such that $R\circ X \sqsubseteq S$ and a closed formula using complements is given. Our greatest right domain of R is equal to $syq(R,R)$.

The domains of a given spec are completely determined by the least and the greatest domain of that spec. We have the following lemma:

Lemma 5.45:

$$R \circ A = R \;\equiv\; R{\searrow} \sqsupseteq A \sqsupseteq R{>}$$

□

The proof of this lemma is trivial. In the $\Rightarrow$ direction we use (5.42) and (5.43). For the $\Leftarrow$ direction we calculate: $R = R \circ R{\searrow} \sqsupseteq R \circ A \sqsupseteq R \circ R{>} = R$. Interpretation of the $\searrow$ operator in terms of classes shows that $R{\searrow}$ consists of the maximal right equivalence classes of R, which means that we cannot enlarge any class $x \in R{\searrow}$ without falsifying $R \sqsupseteq R \circ x$. $R{\searrow}$ is a partition (because I is always a right domain and thus is contained in $R{\searrow}$) and contains a single class for the elements of the universe where R is not defined on the rhs. In terms of our operators we can formulate that as:

Lemma 5.46:

$$R{\searrow} \sqsupseteq R{>}{\swarrow\!\!\!\searrow}$$

□

The proof of this lemma is easy and left to the reader (hint: use (5.44) and $R{>} = R{>}{\not\asymp} \circ R{>}{\swarrow\!\!\!\searrow} = R{>} \circ R{>}{\swarrow\!\!\!\searrow}$).

The definitions and properties above are only for right domains, but we have of course also left domains:

Definition 5.47:(Greatest Domains)

$$R{\swarrow} \;\triangleq\; R/R \sqcap (R/R)^{\cup}$$

□

All properties of right domains can be dualized to properties of left domains.

These calculations were about the least and greatest domains under the $\sqsupseteq$ order. We now continue with the $\lhd$ order. The greatest domain under the $\lhd$ order is quite trivial: I is always a domain and I is the top of the per lattice, so the greatest domain in the per-lattice is simply I. The least domain is more interesting:

Definition 5.48:(Equivalence Domains)

$$R{\succ} \;\triangleq\; \bigwedge(A \;\text{¦}\; R \circ A = R \;\text{¦}\; A)$$

□

As above, this is not the kind of definition that we want to use in our calculations. Although it is clear that we are defining a per and there are also no problems with the minimality, it is not obvious that we are defining a domain and the quantification is also undesirable. Both problems are solved in the following lemma:

Lemma 5.49:(Equivalence Domains)

$$R{\succ} \;=\; R{>} \circ R{\searrow}$$

□

The first part of the proof of this lemma shows that $R{>} \circ R{\searrow}$ is indeed a per. We can use lemma 5.30 for this:

$$\begin{array}{ll}
 & \top\!\!\top \circ R{>} \\
\sqsupseteq & \qquad \{ \text{ least domains, top } \} \\
 & R^{\cup} \circ R \\
\sqsupseteq & \qquad \{ \text{ cancellation } \} \\
 & R^{\cup} \circ R \circ R \backslash R \\
\sqsupseteq & \qquad \{ \text{ least and greatest domains } \} \\
 & R{>} \circ R{\searrow}
\end{array}$$

Because it is the composition of right domains of R, we immediately see that $R{>} \circ R{\searrow}$ is also a right domain of R. This give us one inclusion: $R{\succ} \lhd R{>} \circ R{\searrow}$. We prove the other inclusion:

$$\begin{array}{ll}
 & R{>} \circ R{\searrow} \lhd R{\succ} \\
\equiv & \qquad \{ \text{ glb } \} \\
 & \forall(A \mathrel{¦} R \circ A = R \mathrel{¦} R{>} \circ R{\searrow} \lhd A)
\end{array}$$

We prove this last formula by mutual inclusion after rewriting the $\lhd$ inclusion to $R{>} \circ R{\searrow} \circ A = R{>} \circ R{\searrow}$:

$$\begin{array}{ll}
 & R{>} \circ R{\searrow} \\
= & \qquad \{ \ \mathit{per}.R{\searrow} \ \} \\
 & R{>} \circ R{\searrow} \circ R{\searrow} \\
\sqsupseteq & \qquad \{ \ \bullet\ R \circ A = R;\ \text{lemma 5.45} \ \} \\
 & R{>} \circ R{\searrow} \circ A \\
\sqsupseteq & \qquad \{ \ \bullet\ R \circ A = R;\ \text{lemma 5.45} \ \} \\
 & R{>} \circ R{\searrow} \circ R{>} \\
= & \qquad \{ \ \mathit{per}.R{>},\ R{>} \circ R{\searrow} \ = \ R{\searrow} \circ R{>} \ \} \\
 & R{>} \circ R{\searrow}
\end{array}$$

This completes the proof of lemma 5.49. From the calculation above and lemma 5.41 we can immediately conclude

Lemma 5.50:(Equivalence Domains)

$$R \circ A = R \ \equiv \ R{\succ} \lhd A$$

□

This means that the collection of domains is completely characterized by the least domain under the $\lhd$ order. The interpretation of the per $R{\succ}$ is about the same as that of $R{\searrow}$, except that the class consisting of everything outside $R{>}$ is eliminated by the composition with $R{>}$. The ${\succ}$ operator constructs a right domain; the corresponding left domain operator is defined in

Definition 5.51:(Equivalence Domains)

$$R{\prec} \ = \ R{<} \circ R{\swarrow}$$

□

All properties of the right domain can be dualized to properties of the left domain.

The equivalence domain is an important operator that will be used quite often in the remainder of this thesis and therefore we prove some useful properties of the operator.

Lemma 5.52:

For spec R, per A and $\underline{\square} \in \{\sqsupseteq, =, \sqsubseteq\}$ we have

$$\begin{array}{lrcl}
\text{(a)} & A{\succ} & = & A \\
\text{(b)} & R^{\cup} \circ R & \sqsupseteq & R{\succ} \\
\text{(c)} & R \circ A \;\underline{\square}\; R & \equiv & R{\succ} \circ A \;\underline{\square}\; R{\succ} \\
\text{(d)} & R{\succ}{\times} & = & R{>}
\end{array}$$

$\square$

Proof (5.52a):

$$\begin{array}{ll}
 & A{\succ} \\
= & \quad \{ \; 5.49 \; \} \\
 & A{>} \circ A{\searrow} \\
= & \quad \{ \; per.A \; \} \\
 & A{\times} \circ A{\times} \\
= & \quad \{ \; \text{extension} \; \} \\
 & A
\end{array}$$

Proof (5.52b):

$$\begin{array}{ll}
 & R^{\cup} \circ R \\
\sqsupseteq & \quad \{ \; \text{cancellation} \; \} \\
 & R^{\cup} \circ R \circ R\backslash R \\
\sqsupseteq & \quad \{ \; \text{least and greatest domains} \; \} \\
 & R{>} \circ R{\searrow} \\
= & \quad \{ \; 5.49 \; \} \\
 & R{\succ}
\end{array}$$

Proof (5.52c): $\Leftarrow$ is easy. $\Rightarrow$ proceeds as follows. First instantiating the rhs with $\sqsubseteq$ for $\underline{\square}$.

$$\begin{array}{ll}
 & R{\succ} \sqsupseteq R{\succ} \circ A \\
\equiv & \quad \{ \; \text{def} \searrow\text{, glb, reverse} \; \} \\
 & R\backslash R \sqsupseteq R{\succ} \circ A \;\wedge\; R\backslash R \sqsupseteq A \circ R{\succ} \\
\equiv & \quad \{ \; \text{factors} \; \} \\
 & R \sqsupseteq R \circ R{\succ} \circ A \;\wedge\; R \sqsupseteq R \circ A \circ R{\succ} \\
\equiv & \quad \{ \; \bullet\; R \sqsupseteq R \circ A;\; R \circ R{\succ} = R \; \} \\
 & \mathit{true}
\end{array}$$

Now with $\sqsupseteq$ for $\trianglelefteq$.

$$\begin{array}{ll}
 & R{\succ} \circ A \\
\sqsupseteq & \quad \{ \; \text{least domains} \; \} \\
 & R{\succ} \circ (R \circ A){>} \\
\sqsupseteq & \quad \{ \; \bullet\; R \circ A \sqsupseteq R \; \} \\
 & R{\succ} \circ R{>}
\end{array}$$

$$
\begin{array}{ll}
= & \quad \{ \ (5.52\text{d}) \ \} \\
& R\succ
\end{array}
$$

Proof (5.52d):

$$
\begin{array}{ll}
& \top\!\top \circ R\succ \\
= & \quad \{ \ 5.49 \ \} \\
& \top\!\top \circ R\searrow \circ R> \\
= & \quad \{ \ R\searrow \sqsupseteq I \ \} \\
& \top\!\top \circ R>
\end{array}
$$

□

We conclude this section with a small lemma giving a method for the construction of subtypes of a per:

Lemma 5.53:(Subtype construction)

$$R\colon A \text{---} A \quad \Rightarrow \quad R\sqcap A \quad \subseteq \quad A$$

□

There is only one problematic part in the proof of this lemma. One has to establish that $R\sqcap A$ is a per. Transitivity is easy to prove, but symmetry is more difficult:

$$
\begin{array}{ll}
& R\sqcap A \\
= & \quad \{ \ \text{reverse} \ \} \\
& (R\sqcap A)\cup\cup \\
= & \quad \{ \ R\sqcap A \ \sqsupseteq \ ((R\sqcap A)\cup)< \ \text{(see below)}, \ (R\sqcap A)\cup\colon A \text{---}, \ R\sqcap A \ \sqsubseteq \ A \ \} \\
& ((R\sqcap A) \circ (R\sqcap A)\cup)\cup \\
= & \quad \{ \ \text{reverse} \ \} \\
& (R\sqcap A) \circ (R\sqcap A)\cup \\
= & \quad \{ \ R\sqcap A \ \sqsupseteq \ ((R\sqcap A)\cup)< \ \text{(see below)}, \ (R\sqcap A)\cup\colon A \text{---}, \ R\sqcap A \ \sqsubseteq \ A \ \} \\
& (R\sqcap A)\cup
\end{array}
$$

We still have to prove $R\sqcap A \ \sqsupseteq \ ((R\sqcap A)\cup)<$, which is equivalent to $R\sqcap A \ \sqsupseteq \ (R\sqcap A)>$.

$$
\begin{array}{ll}
& R\sqcap A \\
= & \quad \{ \ \bullet\, R\colon A \text{---} \ \} \\
& A \circ (R\sqcap A) \\
\sqsupseteq & \quad \{ \ \text{monotonicity}, \ per.A \ \} \\
& (R\sqcap A)\cup \circ (R\sqcap A) \\
\sqsupseteq & \quad \{ \ \text{domains} \ \} \\
& (R\sqcap A)>
\end{array}
$$

The remainder of the proof of (5.53) is easy and left to the reader.

□

5.4 Pointwise calculations in the SPEC-calculus

Until now we have avoided pointwise calculations as much as possible, because they are only valid in the model of set-theoretic relations. This sometimes complicates proofs of theorems that are quite trivial pointwise, but require a lot a manipulation in the axiomatic system. In this section we develop some theory that allows us to do pointwise calculations *inside* the axiomatic system. We will assume extensionality throughout this section.

The role of the "points" in our calculations is played by the classes and we start by examining under which conditions we can meaningfully write the predicate $x\langle R\rangle y$, with intended meaning that class x is related to class y by spec R. It is clear that we can only do this if x is an element of a left-domain of R and y is an element of a right-domain of R. This leads to the condition:

$$\exists(A, B \mathrel{¦} R\colon A—B \mathrel{¦} x \in A \wedge y \in B)$$

We can simplify this expression to

(5.54) $R \sqsupseteq x \circ R \ \wedge\ R \sqsupseteq R \circ y$

The equivalence of the two expressions is easy to see: the implication in the "$\Rightarrow$" direction follows immediately from the definitions of — and $\in$, for the "$\Leftarrow$" direction we instantiate with $x \sqcup I$ for A and $y \sqcup I$ for B.

The expression $x \circ R \circ y$ has only two possible values: expanding the definition of squares we get $x \circ \top\!\!\top \circ x \circ R \circ y \circ \top\!\!\top \circ y$ and we know from the cone-rule that this has as possible values $\perp\!\!\perp$ and $x \circ \top\!\!\top \circ y$. This leads to the following definition:

Definition 5.55: (Relates)

For classes x and y and spec R satisfying (5.54) we define

$$x\langle R\rangle y \quad \triangleq \quad x \circ R \circ y = x \circ \top\!\!\top \circ y \quad (\equiv\ x \circ R \circ y \neq \perp\!\!\perp)$$

□

This is pronounced as "R relates x to y". The $\langle \_ \rangle$ predicate has almost the same properties as its counterpart in the set-theoretic relations. Some useful properties are given in

Lemma 5.56:

Assuming $R, S\colon A—B$, $T\colon B—C$, $w, x \in A$, $y \in B$ and $z \in C$ we have:

(a)	$y\langle R^{\cup}\rangle x$	$\equiv$	$x\langle R\rangle y$	(Reverse)
(b)	$x\langle R \sqcap S\rangle y$	$\equiv$	$x\langle R\rangle y \wedge x\langle S\rangle y$	(Cap)
(c)	$x\langle R \sqcup S\rangle y$	$\equiv$	$x\langle R\rangle y \vee x\langle S\rangle y$	(Cup)
(d)	$x\langle S \circ T\rangle z$	$\equiv$	$\exists(y \mathrel{¦} y \in B \mathrel{¦} x\langle S\rangle y \wedge y\langle T\rangle z)$	(Compose)
(e)	$R \sqsubseteq S$	$\equiv$	$\forall(x, y \mathrel{¦} x \in A \wedge y \in B \wedge x\langle R\rangle y \mathrel{¦} x\langle S\rangle y)$	(Extensionality)
(f)	$w\langle A\rangle x$	$\equiv$	$w = x$	(Equality)

□

Proof:
Part (5.56a) is trivial, (5.56b) is proved using definition $x\langle R\rangle y \ \equiv\ x \circ R \circ y = x \circ \top\!\!\top \circ y$

and lemma (4.7a) , (5.56c) uses definition $x\langle R\rangle y \equiv x \circ R \circ y \neq \perp\!\!\perp$. We prove the remaining three parts below:

Part (5.56d):

$$
\begin{array}{ll}
 & x\langle S \circ T\rangle z \\
\equiv & \quad \{ \text{ relates } \} \\
 & x \circ S \circ T \circ z \neq \perp\!\!\perp \\
\equiv & \quad \{ \bullet\ S\colon\ —B \ \} \\
 & x \circ S \circ B \circ T \circ z \neq \perp\!\!\perp \\
\equiv & \quad \{ \text{ extensionality } \} \\
 & x \circ S \circ \sqcup(y \mid y \in B \mid y) \circ T \circ z \neq \perp\!\!\perp \\
\equiv & \quad \{ \text{ distribution } \} \\
 & \sqcup(y \mid y \in B \mid x \circ S \circ y \circ T \circ z) \neq \perp\!\!\perp \\
\equiv & \quad \{ \text{ bottom } \} \\
 & \exists(y \mid y \in B \mid x \circ S \circ y \circ T \circ z \neq \perp\!\!\perp) \\
\equiv & \quad \{ \ square.y \ \} \\
 & \exists(y \mid y \in B \mid x \circ S \circ y \circ \top\!\!\top \circ y \circ T \circ z \neq \perp\!\!\perp) \\
\equiv & \quad \{ \text{ calculus } \} \\
 & \exists(y \mid y \in B \mid x \circ S \circ y \neq \perp\!\!\perp \ \wedge \ y \circ T \circ z \neq \perp\!\!\perp) \\
\equiv & \quad \{ \text{ relates } \} \\
 & \exists(y \mid y \in B \mid x\langle S\rangle y \wedge y\langle T\rangle z)
\end{array}
$$

Part (5.56e), "⇐" direction (the other direction is left to the reader):

$$
\begin{array}{ll}
 & S \\
= & \quad \{ \bullet\ S\colon A—B \ \} \\
 & A \circ S \circ B \\
= & \quad \{ \text{ extensionality } \} \\
 & \sqcup(x \mid x \in A \mid x) \circ S \circ \sqcup(y \mid y \in B \mid y) \\
= & \quad \{ \text{ distribution } \} \\
 & \sqcup(x, y \mid x \in A \wedge y \in B \mid x \circ S \circ y) \\
= & \quad \{ \text{ relates, unit } \} \\
 & \sqcup(x, y \mid x \in A \wedge y \in B \wedge x\langle S\rangle y \mid x \circ \top\!\!\top \circ y) \\
\sqsupseteq & \quad \{ \bullet\ \forall(x, y \mid x \in A \wedge y \in B \wedge x\langle R\rangle y \mid x\langle S\rangle y) \ \} \\
 & \sqcup(x, y \mid x \in A \wedge y \in B \wedge x\langle R\rangle y \mid x \circ \top\!\!\top \circ y) \\
= & \quad \{ \text{ relates, unit } \} \\
 & \sqcup(x, y \mid x \in A \wedge y \in B \mid x \circ R \circ y) \\
= & \quad \{ \text{ distribution } \} \\
 & \sqcup(x \mid x \in A \mid x) \circ R \circ \sqcup(y \mid y \in B \mid y) \\
= & \quad \{ \text{ extensionality } \} \\
 & A \circ R \circ B \\
= & \quad \{ \bullet\ R\colon A—B \ \} \\
 & R
\end{array}
$$

Part (5.56f), "⇒" direction (the other direction is left to the reader).
We prove $w = w \cap x$. Then, by symmetry, it follows that $x = w \cap x$ so $w = x$:

$$
\begin{array}{ll}
 & w \\
= & \quad \{\ square.w,\ x \neq \perp\!\!\perp,\ \text{cone rule}\ \} \\
 & w \circ \top\!\!\top \circ x \circ x \circ \top\!\!\top \circ w \\
= & \quad \{\ \bullet\ w\langle A\rangle x\ \} \\
 & w \circ A \circ x \circ x \circ A \circ w \\
= & \quad \{\ x \in A\ \} \\
 & w \circ x \circ x \circ w \\
= & \quad \{\ (5.13),\ w,\ x{\subseteq}A\ \} \\
 & w \cap x
\end{array}
$$

□

5.5 Difunctionals

In this section we examine a special class of specs that can be viewed as "functional", in the sense that classes on the rhs of a spec have a unique image on the lhs.

Note that we see functions as having their arguments on the rhs and their results on the lhs, this being consistent with writing the application of function f to element x as $f.x$.

For a domain (per) A we want to give a spec-formalization of spec R being "functional to A". Of course we want R to respect the classes of A so the first part is $R\colon A{-\!\!-}$. Assuming this we can write the property that R relates classes on the rhs to unique elements of A as:

$$\forall(x,y,z \ ¦\ y,z \in A \wedge y\langle R\rangle x \wedge z\langle R\rangle x \ ¦\ y = z)$$

We want to get rid of the quantification and obtain an expression only depending on R and A. This is done in the following calculation:

$$
\begin{array}{ll}
 & \forall(x,y,z \ ¦\ y,z \in A \wedge y\langle R\rangle x \wedge z\langle R\rangle x \ ¦\ y = z) \\
\equiv & \quad \{\ \text{generalised range disjunction, reverse, equality}\ \} \\
 & \forall(y,z \ ¦\ y,z \in A \wedge \exists(x \ ¦¦\ y\langle R\rangle x \wedge x\langle R^{\cup}\rangle z) \ ¦\ y\langle A\rangle z) \\
\equiv & \quad \{\ \text{composition}\ \} \\
 & \forall(y,z \ ¦\ y,z \in A \wedge y\langle R \circ R^{\cup}\rangle z \ ¦\ y\langle A\rangle z) \\
\equiv & \quad \{\ \text{extensionality},\ R\colon A{-\!\!-}\ \} \\
 & R \circ R^{\cup} \sqsubseteq A
\end{array}
$$

This leads to the following definition:

Definition 5.57:(Functionality)

$$R\colon A{\longleftarrow} \quad \triangleq \quad R\colon A{-\!\!-} \ \wedge\ A \sqsupseteq R \circ R^{\cup}$$

□

The $\longleftarrow$ is the second of our type judgements. In the next section we will see three other judgements for totality, injectivity and surjectivity. We can combine all these judgements by superposition of symbols, for example $R\colon A{\longleftarrow}B \ \equiv\ R\colon A{\longleftarrow} \wedge R\colon {-\!\!-}B$.

We have the same convention for omitted types in judgements as with $\longleftarrow$: they are always I, so $R\colon \longleftarrow$ is the same as $limp.R$

The class of functional specs is quite large. Almost all specs that we have encountered so far are functional to some domain. Characterizing the functional specs is easy:

Lemma 5.58:

$$\exists(A \mathrel{¦¦} R\colon A\longleftarrow) \;\equiv\; R \sqsupseteq R\circ R^{\cup}\circ R$$

□

If there is an A such that $R\colon A \longleftarrow$ then $R = A\circ R \sqsupseteq R\circ R^{\cup}\circ R$, and if $R \sqsupseteq R\circ R^{\cup}\circ R$ then we have that $R\colon R\circ R^{\cup}\longleftarrow$. Limps, rimps, left conditions, right conditions, pers and squares are all examples of specs that satisfy the condition above. The class as a whole is known under several different names in the literature: for example pseudo-invertible relations [37], regular relations [6] and difunctional relations [36, 54]. We call this class the difunctional specs or for short the difunctionals:

Definition 5.59: (Difunctionality)

A spec R is *difunctional* iff $R \sqsupseteq R\circ R^{\cup}\circ R$.
□

Note that since we have $R\circ R^{\cup}\circ R \sqsupseteq R$ for every spec R we could also have used

(5.60) R is *difunctional* $\triangleq$ $R = R\circ R^{\cup}\circ R$

as definition for difunctionality. We use the following convention:

Convention 5.61: (Difunctionals)

Difunctionals are denoted by lowercase characters f, g, h, etc. and from now on all specs denoted in this way will be difunctionals, even if this is not explicitly declared.
□

We introduced functionality using the uniqueness of the image of a right-equivalence class. We can use this for the definition of function application:

Definition 5.62: (Application)

For $f: A\longleftarrow B$, $y \in A$ and $x \in B$ we define $f.x$ by

$$y = f.x \;\equiv\; y\langle f\rangle x$$

□

This is a good characterization, but sometimes we need a closed formula. A suitable expression is given in the following lemma:

Lemma 5.63: (Application)

For difunctional f and class x such that $f \sqsupseteq f\circ x \neq \perp\!\!\perp$ we have

$$f.x \;=\; f\circ x\circ f^{\cup}$$

□

Proof: we instantiate definition 5.62 with $f\circ x\circ f^{\cup}$ for y and $x\sqcup I$ for B. Before we can use the definition we have to check the preconditions: $f: \longleftarrow x\sqcup I$, $x \in x\sqcup I$ and that for all A such that $f : A\leftarrow$ we have $f \circ x \circ f^{\cup} \in A$. The first follows directly from $f \sqsupseteq f \circ x$, the second is true for all classes and for the third precondition we use both $f \sqsupseteq f \circ x$ and $f \circ x \neq \perp\!\!\perp$. Also, $f \circ x \circ f^{\cup}$ is a class. We complete the proof of the lemma:

$$
\begin{array}{ll}
 & f \circ x \circ f^{\cup} = f.x \\
\equiv & \quad \{ \text{ application } \} \\
 & f \circ x \circ f^{\cup} \langle f \rangle\ x \\
\equiv & \quad \{ \text{ relates } \} \\
 & f \circ x \circ f^{\cup} \circ f \circ x \neq \perp\!\!\perp \\
\equiv & \quad \{ \ \mathit{per}.x, \text{ reverse } \} \\
 & f \circ x \circ (f \circ x)^{\cup} \circ f \circ x \neq \perp\!\!\perp \\
\equiv & \quad \{ \ \bullet\ f \circ x \neq \perp\!\!\perp \ \} \\
 & \mathit{true}
\end{array}
$$

□

We introduced difunctionals using “unique images” but this is not the only way. We could also have taken the path of pseudo-invertibles (see [37]). We define pseudo-invertibility as follows:

Definition 5.64:(Pseudo-Invertible)

A spec R is *pseudo-invertible* iff

$$\exists(A, B, S \mid R\colon A\text{—}B \ \wedge\ S : B\text{—}A \mid R \circ S = A \ \wedge\ S \circ R = B)$$

□

The spec S can be viewed as an inverse of R with respect to A and B. We show that there is only one possible candidate for S, namely $R^{\cup}$. Substitution gives us then that R is difunctional. Vice versa, for a difunctional R we can take $R \circ R^{\cup}$, $R^{\cup} \circ R$ and R for A, B and S to make the rhs of the definition true. From this we can conclude that difunctionality and pseudo-invertibility are the same. We still have to prove that $R^{\cup}$ is the only candidate for S: we only prove one inclusion, this being sufficient because of symmetry.

$$
\begin{array}{ll}
 & R^{\cup} \\
= & \quad \{ \ \bullet\ R\colon A\text{—} \ \} \\
 & R^{\cup} \circ A \\
= & \quad \{ \ \bullet\ R \circ S = A \ \} \\
 & R^{\cup} \circ R \circ S \\
\sqsupseteq & \quad \{ \text{ least domains } \} \\
 & R> \circ S \\
\sqsupseteq & \quad \{ \ \bullet\ S \circ R = B \ \ (\Rightarrow R> \sqsupseteq B\times) \ \} \\
 & B\times \circ S \\
= & \quad \{ \ \bullet\ S : B\text{—} \ \} \\
 & S
\end{array}
$$

The equivalence domain gives us another method for the introduction of the difunctionals. We have already seen that $R\succ$ is at most $R^{\cup} \circ R$. This upper bound is achieved iff R is difunctional:

Lemma 5.65:

$$difunctional.R \quad \equiv \quad R{\succ} = R{\cup} \circ R$$

□

Proof:

$$
\begin{array}{ll}
 & R{\succ} = R{\cup} \circ R \\
\equiv & \quad \{ \text{ equivalence domains } \} \\
 & R{\succ} \sqsupseteq R{\cup} \circ R \\
\equiv & \quad \{ \text{ equivalence domains } \} \\
 & R{\searrow} \circ R{>} \sqsupseteq R{\cup} \circ R \\
\equiv & \quad \{ \text{ least domains } \} \\
 & R{\searrow} \sqcap \top\!\!\top \circ R \sqsupseteq R{\cup} \circ R \\
\equiv & \quad \{ \text{ monotonicity, top } \} \\
 & R{\searrow} \sqsupseteq R{\cup} \circ R \\
\equiv & \quad \{ \text{ greatest domains, } symmetric.(R{\cup} \circ R) \} \\
 & R \backslash R \sqsupseteq R{\cup} \circ R \\
\equiv & \quad \{ \text{ factors } \} \\
 & R \sqsupseteq R \circ R{\cup} \circ R \\
\equiv & \quad \{ \text{ difunctional } \} \\
 & difunctional.R
\end{array}
$$

□

As usual we can dualize the lemma: $difunctional.R \equiv R{\prec} = R \circ R{\cup}$.

A pointwise interpretation of the lemma (combined with its dual) is that a difunctional R is a bijection between the maximal left-equivalence classes and the maximal right-equivalence classes of R.

We end this section with some algebraic properties of difunctionals:

Lemma 5.66: *(Difunctionals)*

$$
\begin{array}{lrcl}
\text{(a)} & per.(f \circ f{\cup}) & & \\
\text{(b)} & per.(f{\cup} \circ f) & & \\
\text{(c)} & difunctional.f{\cup} & & \\
\text{(d)} & difunctional.(f \sqcap g) & & \\
\text{(e)} & difunctional.(f \sqcup g) & \Leftarrow & f \circ g{\cup} = \perp\!\!\perp \quad \wedge \quad f{\cup} \circ g = \perp\!\!\perp \\
\text{(f)} & difunctional.(f \circ g) & \equiv & transitive.(f{\succ} \circ g{\prec})
\end{array}
$$

□

The proof of this lemma is straightforward and left to the reader.

5.6 Injectivity, surjectivity and totality

Functionality is not the only property of a spec that we are interested in; we show in this section how we can define injectivity, surjectivity and totality (the -ity properties) for a spec. We use the same approach that we followed with functionality, namely starting with the familiar pointwise definitions of the concepts we derive a spec-formalization.

We begin with injectivity; for a spec $R\colon \text{—}A$ we want to derive a spec-formulation for the predicate that R is injective on A. A pointwise formulation for this is:

$$\forall(x, y, z \mid x, y \in A \wedge z\langle R\rangle x \wedge z\langle R\rangle y \mid x = y)$$

This is the same expression that we had for functionality, if we replace R with $R^{\cup}$. So we can substitute $R^{\cup}$ for R in the formula for $A\!\leftarrow$ and obtain the expression for injectivity. We use the symbol $\rightarrowtail$ to indicate that we exchanged left and right in the expression for functionality. So we define:

Definition 5.67:(Injectivity)

$$f\colon \rightarrowtail A \quad \triangleq \quad f\colon \text{—}A \wedge A \sqsupseteq f^{\cup} \circ f$$

□

We used an f in this definition because every injective spec is also difunctional. We have the same convention for leaving out the argument that we had for functional specs: $\rightarrowtail$ on its own means $\rightarrowtail I$. This is the class of the right-imps. Injectivity can also be combined with other type judgements, like functionality, by fusing the symbols together

We continue with totality and surjectivity (sometimes called right-totality and left-totality). We only derive the expression for totality, the derivation for surjectivity being dual. For a spec $R\colon \text{—}A$ we have the following pointwise formula for R being total on A:

$$\forall(x \mid x \in A \mid \exists(y \mid\mid y\langle R\rangle x))$$

There are many equivalent expressions for totality, the one that is calculated below being chosen for its duality with the definition of injectivity:

$$\begin{array}{ll}
 & \forall(x \mid x \in A \mid \exists(y \mid\mid y\langle R\rangle x)) \\
\equiv & \qquad \{ \text{ reverse, calculus } \} \\
 & \forall(x \mid x \in A \mid \exists(y \mid\mid x\langle R^{\cup}\rangle y \wedge y\langle R\rangle x)) \\
\equiv & \qquad \{ \text{ composition } \} \\
 & \forall(x \mid x \in A \mid x\langle R^{\cup} \circ R\rangle x) \\
\equiv & \qquad \{ \text{ generalised range disjunction } \} \\
 & \forall(x, y \mid x, y \in A \wedge x = y \mid x\langle R^{\cup} \circ R\rangle x) \\
\equiv & \qquad \{ \text{ equality, substitution } \} \\
 & \forall(x, y \mid x, y \in A \wedge x\langle A\rangle y \mid x\langle R^{\cup} \circ R\rangle y) \\
\equiv & \qquad \{ \text{ extensionality, } R\colon \text{—}A \ \} \\
 & A \sqsubseteq R^{\cup} \circ R
\end{array}$$

This gives us the following definitions for totality and surjectivity:

Definition 5.68:(Totality)

$$R\colon \text{—}\!\!\!\!\dashv A \quad \triangleq \quad R\colon \text{—}A \wedge R^{\cup} \circ R \sqsupseteq A$$

□

Definition 5.69:(Surjectivity)

$$R\colon A{\vdash\!\!\relbar\!\!\relbar} \quad \triangleq \quad R\colon A{\relbar\!\!\relbar} \ \wedge R \circ R^{\cup} \sqsupseteq A$$

□

Combined judgements are expressed with fused symbols; for example combining functionality, surjectivity and respecting a right domain uses the symbol $|\!\!\longleftarrow$, where $f : A|\!\!\longleftarrow B$ is equivalent to $f : A{\relbar\!\!\relbar}B \ \wedge \ A = f \circ f^{\cup}$.

We mentioned that the expression above is only one of the possible definitions of totality. The following two lemmas give some other possibilities for definitions of totality and surjectivity:

Lemma 5.70:(Totality)

The following three expressions are all equivalent to $R\colon {\relbar\!\!\relbar\!\!\dashv}A$:

(a) $R\colon {\relbar\!\!\relbar}A \ \wedge \ \top\!\!\top \circ R \sqsupseteq A$
(b) $R\colon {\relbar\!\!\relbar}A \ \wedge \ R^{>} = A^{\times}$
(c) $R^{\succ} \mathrel{\lhd\!|} A$

□

Lemma 5.71:(Surjectivity)

The following three expressions are all equivalent to $R\colon A{\vdash\!\!\relbar\!\!\relbar}$:

(a) $R\colon A{\relbar\!\!\relbar} \ \wedge \ R \circ \top\!\!\top \sqsupseteq A$
(b) $R\colon A{\relbar\!\!\relbar} \ \wedge \ R^{<} = A^{\times}$
(c) $R^{\prec} \mathrel{\lhd\!|} A$

□

We only prove lemma 5.70, the proof of lemma 5.71 being dual. Since the definition of ${\relbar\!\!\relbar\!\!\dashv}A$ and the three expressions of the lemma have as common part $R\colon {\relbar\!\!\relbar}A$ (for (5.70c) this is $R^{\succ} \lhd A$) we have to prove the equivalence of $R^{\cup} \circ R \sqsupseteq A$, $\top\!\!\top \circ R \sqsupseteq A$, $R^{>} \sqsupseteq A^{>}$ and $R^{\succ} \sqsupseteq A$ under the assumption $R\colon {\relbar\!\!\relbar}A$.

Proof:

$$
\begin{array}{ll}
 & R^{\cup} \circ R \sqsupseteq A \\
\Rightarrow & \qquad \{ \ \text{monotonicity} \ \} \\
 & \top\!\!\top \circ R \sqsupseteq A \\
\equiv & \qquad \{ \ \text{least domains} \ \} \\
 & R^{>} \sqsupseteq A^{\times} \\
\Rightarrow & \qquad \{ \ \text{monotonicity, least domains} \ \} \\
 & R^{>} \circ A \sqsupseteq A \\
\Rightarrow & \qquad \{ \ \bullet\ R\colon {\relbar\!\!\relbar}A, \text{ greatest domain} \ \} \\
 & R^{>} \circ R^{\searrow} \sqsupseteq A \\
\equiv & \qquad \{ \ \text{def } \succ \ \} \\
 & R^{\succ} \sqsupseteq A \\
\Rightarrow & \qquad \{ \ \text{domains} \ \} \\
 & R^{\cup} \circ R \sqsupseteq A
\end{array}
$$

□

5.7 Type-deduction rules

In the previous sections we defined five basic type-judgements: respecting a domain, functionality to a domain, surjectivity to a domain, injectivity on a domain and totality on a domain. Respecting a domain is implied by the other four notions but these other notions can be combined independently, giving us a total of 16 different type-judgements. We treat the components of a type-judgement separately to avoid a combinatorial explosion. Since injectivity and functionality have basically the same properties (swap left and right) we only need to give rules for one of them. In this section we treat functionality, the properties of injectivity can be derived immediately from these. We have the same duality for totality and surjectivity, and we only give properties of totality.

The rest of this section gives a systematic overview of the properties respecting a domain on the left, functionality and totality when applied to specs that are constructed using the basic operations $^{\cup}$, $\sqcap$, $\sqcup$ and $\circ$. The proofs of the lemmas in this section are easy and are left to the reader.

Lemma 5.72:(Type Reverse)

(a) $R^{\cup} : A\relbar\joinrel\relbar \quad\equiv\quad R : \relbar\joinrel\relbar A$
(b) $R^{\cup} : A\longleftarrow \quad\equiv\quad R : \longrightarrow A$
(c) $R^{\cup} : \relbar\joinrel\dashv A \quad\equiv\quad R : A\vdash\joinrel\relbar$

□

Lemma 5.73:(Type Cap)

(a) $R \sqcap S : (A \sqcap B)\relbar\joinrel\relbar \quad\Leftarrow\quad R : A\relbar\joinrel\relbar \ \wedge\ S : B\relbar\joinrel\relbar$
(b) $R \sqcap S : (A \sqcap B)\longleftarrow \quad\Leftarrow\quad R : A\longleftarrow \ \wedge\ S : B\longleftarrow$

□

Lemma 5.74:(Type Cup)

(a) $R \sqcup S : A\relbar\joinrel\relbar \quad\Leftarrow\quad R : A\relbar\joinrel\relbar \ \wedge\ S : A\relbar\joinrel\relbar$
(b) $R \sqcup S : \relbar\joinrel\dashv A \quad\Leftarrow\quad R : \relbar\joinrel\dashv A \ \wedge\ S : \relbar\joinrel\dashv A$

□

Lemma 5.75:(Type Composition)

(a) $R \circ S : A\relbar\joinrel\relbar \quad\Leftarrow\quad R : A\relbar\joinrel\relbar$
(b) $R \circ S : A\longleftarrow \quad\Leftarrow\quad R : A \longleftarrow B \ \wedge\ S : C\longleftarrow \ \wedge\ (B \sqsupseteq C \vee B \lhd C)$
(c) $R \circ S : \relbar\joinrel\dashv A \quad\Leftarrow\quad R : \relbar\joinrel\dashv C \ \wedge\ S : B \relbar\joinrel\dashv A \ \wedge\ (C \sqsupseteq B \vee B \lhd C)$

□

We can simplify (5.75b) and (5.75c) if we have C equal to I. This corresponds to limpness of S or totality of R. This gives us the following lemma:

Lemma 5.76:

(a) $R \circ S : A\longleftarrow \quad\Leftarrow\quad R : A \longleftarrow \ \wedge\ S : \longleftarrow$
(b) $R \circ S : \relbar\joinrel\dashv A \quad\Leftarrow\quad R : \relbar\joinrel\dashv \ \wedge\ S : \relbar\joinrel\dashv A$

□

The final lemma of this section is the generalization of lemma 5.41 to the functionality and totality judgements:

Lemma 5.77:

$$
\begin{array}{lllll}
\text{(a)} & R\colon A{\leftarrow} & \Leftarrow & R\colon B{\leftarrow} \;\wedge\; A \supseteq B \\
\text{(b)} & R\colon {\longrightarrow\!\!\!\dashv} A & \Leftarrow & R\colon {\longrightarrow\!\!\!\dashv} B \;\wedge\; B \mathrel{\triangleleft\!|} A
\end{array}
$$

□

Chapter 6

The category of total difunctionals

Category theory has become popular in computing science as a method for describing and defining datatypes and operations on these datatypes. Some examples of the use of category theory for this purpose can be found in for example [23, 24, 31, 42, 49, 55]. This chapter gives a short introduction to some basic notions from category theory, followed by the construction of an interesting category called **Difun** based on a SPEC-calculus. Then we show how to find constructions in the SPEC-calculus corresponding to categorical constructions in **Difun**. The algebraic properties of the new constructions are also investigated.

6.1 Short introduction to category theory

This section gives a short introduction to some basic concepts of category theory. The definitions that are given here have been copied from Pierce [50] (with some small adaptations in notation) and the reader is referred to this book or other introductions to category theory like [7, 39] for a more extensive introduction to category theory. Only those parts of category theory that are essential for the understanding of the remainder of this thesis are presented here.

Category theory is a framework that allows the expression of a common structure in many mathematical theories. The primitive concept in category theory is the arrow (this can often be interpreted as a function) and the source and targets of these arrows are called the objects of the category. The action of the arrows on their source objects are not described by an internal structure of arrows and objects but by the properties under composition with other arrows. A formal definition:

Definition 6.1: (Category)

A category $\mathcal{C}$ consists of a collection of objects and a collection of arrows with functions dom, cod, $\circ$ and id such that:

1. For every arrow f we have objects $dom.f$ and $cod.f$, the domain and codomain of f. respectively. Writing $f \in B \leftarrow A$ means $A = dom.f \ \wedge \ B = cod.f$. The collection of all arrows with domain A and codomain B is denoted by $\mathcal{C}(A, B)$.

2. For all arrows f and g such that $dom.g = cod.f$ there is a composite arrow $g \circ f$ with $g \circ f \; \in \; cod.g \leftarrow dom.f$. The $\circ$ operator is associative:

   $$h \circ (g \circ f) = (h \circ g) \circ f$$

   for all f, g, h such that $cod.f = dom.g$ and $cod.g = dom.h$.

3. For every object A we have an identity arrow $id.A$ with $id.A \in A \leftarrow A$. The identity arrow is an identity of $\circ$: for $f \in B \leftarrow A$ we have

   $$id.B \circ f = f \circ id.A = f$$

□

Pierce uses the notation $f{:}A{\to}B$ for $f \in B \leftarrow A$, but this could cause confusion with the relational type judgements and is therefore not used here. Note how impractical the rule for the typing of compositions becomes had we used Pierce's notation. This is a consequence of using the function composition symbol in combination with the $\to$ for typing functions. Other authors (e.g. [24]) use the reverse composition $f; g = g \circ f$ to circumvent this problem, but this shifts the problem to function application: $(f; g).x = g.(f.x)$. Also note that the composition operator is partial. The domain and codomain have to match exactly, otherwise the composition is not defined.

A simple example of a category is the category **Set** with as objects sets where $f{\in}B{\leftarrow}A$ if f is a total function with domain set A and codomain set B. In **Set** the codomain is not the same as the range of a function. For example, the squaring function from reals to reals is not the same as the squaring function from reals to non-negative reals.

Our first categorical concept is the notion of isomorphy:

Definition 6.2: (Isomorphy)

Objects A and B are *isomorphic* iff there exist arrows f and g such that

$$f \circ g = id.A \;\wedge\; g \circ f = id.B.$$

□

The arrows f and g are called *isomorphisms*. It is easy to prove that given one of the isomorphisms the other is uniquely determined. Proving this is left as an exercise to the reader. In the example **Set** we see that the notion of isomorphy of sets in set theory is the same as in category theory and that the two arrows are each other's inverse.

The concepts of initiality and terminality play a central role in our use of category theory:

Definition 6.3: (Initiality)

Object A is an *initial object* iff for every object B there is exactly one arrow f with $f \in B \leftarrow A$

□

Definition 6.4: (Terminality)

Object A is a *terminal object* iff for every object B there is exactly one arrow f with

$f \in A \leftarrow B$
□

Not every category has initial or terminal objects, but if they exist then they are unique up to isomorphism. Arrows from a terminal object to an object A are called *global elements* or *constants* of A. The category **Set** has a single initial object, the empty set, and all one-element sets are terminal objects. Constants from a set A are functions mapping the single element of a terminal object to an element of A. Every element of a set can be picked out by a constant.

Isomorphisms, initiality and terminality are all inside a single category, but we are often interested in mappings between two (not necessarily different) categories. Functors are structure-preserving mappings between categories and they are the arrows in the category of categories where the objects themselves are (small) categories.

Definition 6.5: (Functor)

For categories $\mathcal{C}$ and $\mathcal{D}$ a *functor* F such that $F \in \mathcal{D} \leftarrow \mathcal{C}$ maps objects of $\mathcal{C}$ to objects of $\mathcal{D}$ and arrows $f \in A \leftarrow B$ of $\mathcal{C}$ to arrows $F.f \ \in \ F.A \leftarrow F.B$ of $\mathcal{D}$. This mapping has to satisfy the following two conditions:

1. $F.(id.A) = id.(F.A)$, for all objects A of $\mathcal{C}$
2. $F.(g \circ f) = F.g \circ F.f$, for all arrows f and g of $\mathcal{C}$ such that $dom.g = cod.f$

□

A functor with the same source and target category is called an *endofunctor*. It is easy to see that functors preserve isomorphisms, but initial and terminal objects are not necessarily preserved. Some simple examples of functors are the identity functor, an endofunctor that does nothing, and constant functors that map all objects to a fixed object A of the target category and all arrows to $id.A$.

For a non-trivial example of a functor, consider for given categories $\mathcal{C}$ and $\mathcal{D}$ the *product category* $\mathcal{C}\times\mathcal{D}$. The objects in this category are pairs of objects of $\mathcal{C}$ and $\mathcal{D}$ and the arrows are pairs of arrows from $\mathcal{C}$ and $\mathcal{D}$, i.e. if $f \in A \leftarrow B$ in $\mathcal{C}$ and $g \in E \leftarrow G$ in $\mathcal{D}$ then there is $(f, g){\in}(A, E) \leftarrow (B, G)$ in $\mathcal{C}\times\mathcal{D}$. Compositions and identities are defined coordinatewise.

Several functors can be defined using product categories. For every category $\mathcal{C}$ there is the doubling functor δ where $\delta \in \mathcal{C}\times\mathcal{C} \leftarrow \mathcal{C}$. This functor maps object A to object (A, A) and arrow f to arrow (f, f). For a product category $\mathcal{C} \times \mathcal{D}$ there are two projection functors, $\ll \ \in \mathcal{C} \leftarrow \mathcal{C}\times\mathcal{D}$ and $\gg \ \in \mathcal{D} \leftarrow \mathcal{C}\times\mathcal{D}$ defined by $\ll .(A, B) = A$, $\ll .(f, g) = f$, $\gg .(A, B) = B$ and $\gg .(f, g) = g$ for objects A and B and arrows f and g. Functors with as source a product category are called binary functors .

Functors often play the role of type-constructors in category-theory-based type theories. A simple example of this is list construction. Given a set A we can construct the set $List.A$ of all finite lists over A and given a total function f from A to B we can construct a list-homomorphism $List.f$ from finite lists over A to finite lists over B mapping the list $[x_0, x_1, .., x_{n-1}]$ to $[f.x_0, f.x_1, .., f.x_{n-1}]$. *List* is a functor.

A functor is a structure-preserving map between categories, but the abstraction can be taken one step further. We are often interested in structure-preserving maps between functors, an arrow in a category where the objects are functors. This is done with the so-called natural transformations:

Definition 6.6: (Natural Transformation)

For categories $\mathcal{C}$ and $\mathcal{D}$ and functors $F,G \in \mathcal{D} \leftarrow \mathcal{C}$ a function τ mapping every $\mathcal{C}$-object A to a $\mathcal{D}$-arrow $\tau.A \in G.A \leftarrow F.A$ is a *natural transformation* from F to G, notation $\tau \in G \stackrel{\cdot}{\leftarrow} F$, iff

$$G.f \circ \tau.A \ = \ \tau.B \circ F.f$$

for every $\mathcal{C}$-arrow $f \in B \leftarrow A$

□

We use a dot over the arrow because of the lifting that is going on. An example of a natural transformation is the singleton-list constructor. For every set A let $\tau.A$ be the function from A to $List.A$ mapping an element of A to the singleton list containing that element. Now τ is a natural transformation from the identity functor to the list functor. That is, for all $f \in B \leftarrow A$ we have $List.f \circ \tau.A = \tau.B \circ f$.

This example shows how identity structures (the structure of **Set**) are preserved by mapping to singleton-lists. Another example of a natural transformation is the reverse function on lists, which is a natural transformation from $List$ to $List$.

There are many other constructions in category theory that are of interest for the SPEC-calculus, but the constructions of this section are the most important ones for us and the remainder of this chapter shows how to build a category using the SPEC-calculus and develop SPEC-constructions that can play the roles of functors and natural transformations. Later chapters examine how particular constructions of functors from category-theory can be performed in our category and examine the algebraic properties of the corresponding SPEC-operations.

6.2 The category Difun

The type judgements developed in the previous chapter can be used to construct a number of categories. The objects of those categories are the pers. We could for example for $(R, B, A) \in B \leftarrow A$ use the judgement $R\colon B{-\!\!-}A$. This would construct a category that is similar (but not isomorphic for cardinality reasons) to the category **Rel** of binary relations between sets. The arrow is chosen to be a triple because spec R usually does not have a unique type judgement and every arrow must have a unique domain and codomain. Another possibility is using the judgement $R\colon B{\leftarrow\!\!-}A$, giving something similar (not isomorphic, again for cardinality reasons) to the category **Pfn** (the category of partial functions between sets).

The category **Set** has proven to be very useful for the description of the construction of datatypes and (functional) programs using these types (see for example Malcolm

[42]) and the category that is constructed here is in many aspects similar to **Set**. We assume that the underlying SPEC-calculus is unary.

Definition 6.7: (Difun)

The category **Difun** is defined by

- Objects: pers
- Arrows: **Difun**(A, B) is the collection of all triples (f, B, A) such that $f : B \longleftarrow\!\!\!\dashv A$
- Domains: $dom.(f, A, B) = B$ and $cod.(f, A, B) = A$
- Composition: $(f, A, B) \circ (g, B, C) = (f \circ g, A, C)$
- Identity: $id.A = (A, A, A)$

□

The arrows are triples because a difunctional f can have more than one $\longleftarrow\!\!\!\dashv$ type judgement and an arrow must have unique domains. The reader can check for himself that the definition above indeed satisfies all the requirements of a category.

Two pers A and B are isomorphic in **Difun** iff there exist f and g such that $f : A \longleftarrow\!\!\!\dashv B$, $g : B \longleftarrow\!\!\!\dashv A$, $f \circ g = A$ and $g \circ f = B$. From the pseudo-invertibility characterisation of difunctionals it follows then that $g = f^{\cup}$ and that $f : A \leftarrow\!\!\!\dashv\!\!\!\rightarrow B$. Formulated as a lemma this becomes:

Lemma 6.8: (Isomorphy)

Pers A and B are isomorphic iff there exists a difunctional f such that $f \circ f^{\cup} = A$ and $f^{\cup} \circ f = B$
□

This means that f is a one-to-one mapping between elements of A and B in the interpretation of pers as types with equivalence classes as elements. This corresponds to the normal notion of isomorphy of sets. **Difun** is in many aspects similar to **Set** and the other witnesses of this similarity can be found in the initial and terminal objects:

Lemma 6.9:

Difun has a unique initial object $\perp\!\!\!\perp$ and has as terminal objects the classes.
□

Viewing pers as sets of elements, $\perp\!\!\!\perp$ corresponds to the empty set (the initial object of **Set**) and the classes correspond to one-element sets (the terminal objects of **Set**). Since all terminal objects are isomorphic we can choose a representative, and the obvious candidate for this role is of course $\top\!\!\!\top$ because this is the only class whose existence follows directly from the SPEC-axioms. The proof of the facts that the classes are the pers isomorphic to $\top\!\!\!\top$ and that $\perp\!\!\!\perp$ is only isomorphic to itself are straightforward and left to the reader.

Now we only have to prove that $\perp\!\!\perp$ is an initial object and that $\top\!\!\top$ is a terminal object. The initiality of $\perp\!\!\perp$ is proved by showing the existence and uniqueness of arrows with codomain B for all pers B:

$$
\begin{array}{ll}
 & f : B \leftarrow\!\!\dashv \perp\!\!\perp \\
\equiv & \qquad \{ \text{ definition } \leftarrow\!\!\dashv, \perp\!\!\perp \ \sqsubseteq\ f^{\cup} \circ f \ \} \\
 & f \circ f^{\cup} \sqsubseteq B \quad \wedge \quad B \circ f \ = \ f \quad \wedge \quad f \circ \perp\!\!\perp \ = \ f \\
\equiv & \qquad \{ \text{ bottom, zero } \} \\
 & f \ = \ \perp\!\!\perp
\end{array}
$$

The terminality of $\top\!\!\top$ is proved by showing the existence and uniqueness of arrows with domain B for all pers B:

$$
\begin{array}{ll}
 & f : \top\!\!\top \leftarrow\!\!\dashv B \\
\equiv & \qquad \{ \text{ definition } \leftarrow\!\!\dashv, f \circ f^{\cup} \ \sqsubseteq\ \top\!\!\top \ \} \\
 & \top\!\!\top \circ f \ = \ f \quad \wedge \quad f \circ B \ = \ f \quad \wedge \quad \top\!\!\top \circ f \ = \ \top\!\!\top \circ B \\
\equiv & \qquad \{ \text{ calculus } \} \\
 & f \ = \ \top\!\!\top \circ B
\end{array}
$$

□

There is one-to-one correspondence between constants of a per A (arrows from a terminal object to A) and elements of A, since for $f : A \leftarrow\!\!\dashv \top\!\!\top$ and $x \in A$ we have the following Galois connection:

$$x \ = \ f \circ f^{\cup} \quad \equiv \quad x \circ \top\!\!\top \ = \ f$$

The proof of this connection is trivial and left to the reader. The one-to-one correspondence follows from the use of the equality as ordering.

6.3 Relators

In this section we investigate functors for **Difun**. Since our goal is to perform all calculations in the SPEC-calculus we are looking for SPEC-operators that can play the role of functors. We do our calculations first for endo-functors and generalise this later to products of **Difun**.

A functor consists of two parts, one operating on objects and one operating on arrows, but it is sufficient for **Difun** to only define a single operation on specs. Such an operator will be called a *relator* and a relator F used as functor maps object A to object $F.A$ and arrow (f, A, B) to arrow $(F.f, F.A, F.B)$. Although F is only applied to difunctionals in **Difun**, our aim is to develop a total operation on specs. This is done because it makes calculations in the SPEC-calculus much easier. One doesn't have to check that the argument of a relator is a difunctional and it allows generalisations of functional categorical notions to notions about relations.

Examining the definition of a functor gives us three conditions that a relator has to satisfy:

1. $per.A \;\Rightarrow\; per.(F.A)$
2. $f : A {\leftarrow\!\!\dashv} B \;\Rightarrow\; F.f : F.A {\leftarrow\!\!\dashv} F.B$
3. $F.(f \circ g) \;=\; F.f \circ F.g$

The remaining condition, the preservation of identity arrows, follows from the first condition. Condition 3 gives us the first property desired for a relator:

$$F.(R \circ S) \;=\; F.R \circ F.S$$

Condition 1 can be rewritten to $F.A = F.A \circ (F.A)^{\cup}$ for per A. Assuming condition 3 this follows if we require that F commutes with $^{\cup}$. This gives us a second property for relators:

$$(F.R)^{\cup} = F.(R^{\cup})$$

This allows us to write $F.R^{\cup}$ without having to specify precedence. The two previous properties already gave us $F.f : F.A \text{---} F.B$ (assuming $f : A {\leftarrow\!\!\dashv} B$). A third property of relators is needed to complete the obligations for a functor: $F.A \sqsupseteq F.f \circ F.f^{\cup}$ and $F.f^{\cup} \circ F.f \sqsupseteq F.B$. A simple and sufficient condition is to demand monotonicity:

$$F.R \sqsupseteq F.S \;\Leftarrow\; R \sqsupseteq S$$

Combining gives us the following definition for relators:

Definition 6.10: (Relators)

A total function F from specs to specs is a *relator* iff for all specs R and S:

$$\begin{array}{lrcl} \text{(a)} & F.(R \circ S) & = & F.R \circ F.S \\ \text{(b)} & (F.R)^{\cup} & = & F.(R^{\cup}) \\ \text{(c)} & F.R \sqsupseteq F.S & \Leftarrow & R \sqsupseteq S \end{array}$$

□

Relators were originally defined for the monotype system by Backhouse and made public at a summerschool on Ameland in 1989. The definition was published in Malcolm's thesis [42]. Their definition is the same as the one given here with one extra condition, $F.I \;\sqsubseteq\; I$, that is essential for the preservation of monotypes (pids). Bird and de Moor [10] don't have (6.10b) as an axiom. Instead they show that (6.10b) and (6.10c) are equivalent assuming a "tabularity" axiom on all specs.

The definition given above is for unary relators; they map one spec to one spec. In category theory we have product categories and functors going from and to these product categories; The corresponding notions in the SPEC-calculus are built using binary SPEC-calculi. The category **Difun** × **Difun** has as objects pairs of pers (A, B) and as arrows pairs of triples $((f, A, C), (g, B, D))$ such that $f : A {\leftarrow\!\!\dashv} C$ and $g : B {\leftarrow\!\!\dashv} D$. Functors from **Difun** × **Difun** to **Difun** then correspond to relators from binary specs to specs satisfying:

Definition 6.11: (Binary Relator)

A total binary function $\otimes$ from pairs of specs to specs is a *binary relator* iff for all specs R, S, T and U:

$$\begin{array}{llcl}
\text{(a)} & (R \otimes S) \circ (T \otimes U) & = & (R \circ T) \otimes (S \circ U) \\
\text{(b)} & (R \otimes S)^{\cup} & = & R^{\cup} \otimes S^{\cup} \\
\text{(c)} & R \otimes S \sqsupseteq T \otimes U & \Leftarrow & R \sqsupseteq T \ \wedge \ S \sqsupseteq U
\end{array}$$

□

Realizing that there are also $\circ$, $\cup$ and $\sqsupseteq$ operations in binary SPEC-calculi shows us that this really is the same definition as with unary relators. Tagging the operations in the binary SPEC-calculus with a 2 and those in the unary with a 1 and switching to prefix notation for the relator, (6.11a) to (6.11c) can be rewritten to:

$$\begin{array}{llcl}
\text{(d)} & \otimes.(R,S) \circ_1 \otimes.(T,U) & =_1 & \otimes.((R,S) \circ_2 (T,U)) \\
\text{(e)} & (\otimes.(R,S))^{\cup_1} & =_1 & \otimes.((R,S)^{\cup_2}) \\
\text{(f)} & \otimes.(R,S) \sqsupseteq_1 \otimes (T,U) & \Leftarrow & (R,S) \sqsupseteq_2 (T,U)
\end{array}$$

The generalisation from unary and binary relators to relators between SPEC-calculi with arbitrary arities is easy and left as an exercise to the reader.

All functors mentioned in the section about category theory have a corresponding relator in the SPEC-calculus. In particular:

- The identity relator: $\mathcal{I}.R = R$
- The constant relators: for all pers A, $A^{\bullet}.R = A$
- The doubling relator: $\delta.R = (R, R)$
- The projection relators: $R \ll S = R$, $R \gg S = S$

A construction for the *List* relator will be given in the chapter on inductive types. There are many methods for the construction of relators from other relators and four of them are given in this chapter. Other methods can be found in following chapters. The two most common methods for the construction of relators from other relators is relator composition and tupling:

Definition 6.12: (Relator Composition)

For relators F and G such that source of F is the target of G the relator FG is defined by

$$FG.X \ = \ F.(G.X)$$

□

Definition 6.13: (Relator Tupling)

For relators F and G the relator (F, G) is defined by:

$$(F,G).(X,Y) \ = \ (F.X, G.Y)$$

□

This definition can of course be generalised to non-binary tuples also. An example combining relator composition and tupling: for per A and binary relator $\otimes$ the sectioned relator $A\otimes$ can be defined as $\otimes(A^{\bullet}, \mathcal{I})\delta$. A binary relator with one of its arguments fixed to a per is a unary relator. The last construction method of this section is:

Definition 6.14:(Isomorphic Relators)

For relator F and spec $\gamma\colon F.I|\!\!\longleftarrow$, define for all specs R:

$$F^{\gamma}.R \triangleq \gamma^{\cup} \circ F.R \circ \gamma$$

□

Relator F^{γ} is called isomorphic to F because $F.A$ is isomorphic to $F^{\gamma}.A$ for all pers A. The isomorphisms are $F.A \circ \gamma$ and $\gamma^{\cup} \circ F.A$.

Relators preserve all orders and type judgements encountered so far:

Lemma 6.15:(Relators)

$$
\begin{array}{rcl}
F.R \sqsubseteq F.S & \Leftarrow & R \sqsubseteq S \\
F.R \lhd F.S & \Leftarrow & R \lhd S \\
F.R \subseteq F.S & \Leftarrow & R \subseteq S \\
F.R \lhd\!| F.S & \Leftarrow & R \lhd\!| S \\
F.R\colon F.A\!\longleftarrow\!\!\!\!-\!\!\!- & \Leftarrow & R\colon A\!\relbar\!\!\relbar \\
F.R\colon F.A\!\longleftarrow & \Leftarrow & R\colon A\!\longleftarrow \\
F.R\colon F.A\!\vdash\!\!\!\relbar\!\!\relbar & \Leftarrow & R\colon A\!\vdash\!\!\!\relbar\!\!\relbar \\
F.R\colon \relbar\!\!\relbar F.A & \Leftarrow & R\colon \relbar\!\!\relbar A \\
F.R\colon \longrightarrow F.A & \Leftarrow & R\colon \longrightarrow A \\
F.R\colon \relbar\!\!\!\dashv F.A & \Leftarrow & R\colon \relbar\!\!\!\dashv A
\end{array}
$$

□

The proof of this lemma is trivial and omitted. Having distribution over $\circ$, it is natural to look at the distribution over the factors. It turns out that there is only an inclusion:

Lemma 6.16:(Relators)

$$
\begin{array}{l}
F.R\backslash F.S \sqsupseteq F.(R\backslash S) \\
F.R/F.S \sqsupseteq F.(R/S)
\end{array}
$$

□

The proof is again trivial; use the Galois correspondence, distribute the relator and use cancellation and monotonicity. Lemma 6.15 showed that relators preserve domains, i.e. if A is a right-domain of R then $F.A$ is a right-domain of $F.R$, but this does not necessarily mean that relators distribute over the domain operators. There are only inclusions for least and greatest domains, but it is possible to achieve equality for least domains by adding an extra condition. Distribution over the equivalence domain is still an open problem: inclusion can be proved for every relator, but it turns out that there is equality for almost every relator. The question whether distribution is valid in general is still open.

Only distribution over right-domains is considered in this section, but every result can be dualized to left-domains. Relators distribute over greatest domains with an inclusion:

Lemma 6.17:(Greatest Domains)

$$(F.R){\searrow} \sqsupseteq F.(R{\searrow})$$

□

Proof:

$$
\begin{array}{ll}
 & (F.R)\searrow \;\sqsupseteq\; F.(R\searrow) \\
\equiv & \quad \{ \text{ symmetry, def } \searrow \text{ } \} \\
 & F.R \backslash F.R \;\sqsupseteq\; F.(R\searrow) \\
\equiv & \quad \{ \text{ factors } \} \\
 & F.R \;\sqsupseteq\; F.R \circ F.(R\searrow) \\
\equiv & \quad \{ \text{ relators, greatest domains } \} \\
 & true
\end{array}
$$

□

Equality is in general not true; consider for example the relator $\perp\!\!\!\perp^{\bullet}$.

Relators distribute over least domains also only with inclusions, but an extra condition on the relator allows equality.

Lemma 6.18:(Least Domains)

$$F.(R>) \sqsupseteq (F.R)>$$

□

The proof is an instantiation of the Galois-correspondence for least domains (4.20):

$$
\begin{array}{ll}
 & \top\!\!\top \circ F.(R>) \\
\sqsupseteq & \quad \{ \text{ top } \} \\
 & F.\top\!\!\top \circ F.(R>) \\
= & \quad \{ \text{ relators, least domains } \} \\
 & F.(\top\!\!\top \circ R) \\
\sqsupseteq & \quad \{ \text{ relators } \} \\
 & F.R
\end{array}
$$

□

In practice we often use another lemma that gives us an equality instead of an inclusion:

Lemma 6.19:(Least Domains)

$$F.(R>) = F.I \circ (F.R)>$$

□

Proof:

$$
\begin{array}{ll}
 & F.(R>) \\
= & \quad \{ \text{ relators } \} \\
 & F.I \circ F.(R>) \\
\sqsupseteq & \quad \{ \text{ (6.18) } \} \\
 & F.I \circ (F.R)> \\
= & \quad \{ \text{ least domains } \} \\
 & F.I \sqcap \top\!\!\top \circ F.R \\
\sqsupseteq & \quad \{ \text{ top, relators } \} \\
 & F.I \sqcap F.(\top\!\!\top \circ R)
\end{array}
$$

$$
\begin{array}{ll}
\sqsupseteq & \quad \{ \text{ monotonicity, least domains } \} \\
& F.(R{>})
\end{array}
$$

□

There is also an inclusion with the distribution using the per-order:

Lemma 6.20: (Least Domains)

$$F.(R{>}) \lhd (F.R){>}$$

□

Proof:

$$
\begin{array}{ll}
& F.(R{>}) \circ (F.R){>} \\
= & \quad \{ \text{ relators } \} \\
& F.(R{>}) \circ F.I \circ (F.R){>} \\
= & \quad \{ \text{ (6.19) } \} \\
& F.(R{>}) \circ F.(R{>}) \\
= & \quad \{ \text{ pers } \} \\
& F.(R{>})
\end{array}
$$

□

Equality instead of inclusions can be obtained by imposing an extra condition on the relator:

Lemma 6.21: (Least Domains)

$$\forall (R \mathrel{\vdots\vdots} F.(R{>}) = (F.R){>}) \quad \equiv \quad I \sqsupseteq F.I$$

□

Relators satisfying $I \sqsupseteq F.I$ are called *strong relators* and are used in the monotype-system because they preserve monotypes (in the monotype-system they are simply called relators there and our relators are called weak relators).

The proof of the lemma is trivial: in the "$\Rightarrow$" direction, instantiate the lhs with I for R giving $F.I = (F.I){>} \sqsubseteq I$; in the "$\Leftarrow$" direction we calculate:

$$
\begin{array}{ll}
& (F.R){>} \\
\sqsupseteq & \quad \{ \bullet\ I \sqsupseteq F.I \ \} \\
& F.I \circ (F.R){>} \\
= & \quad \{ \text{ (6.19) } \} \\
& F.(R{>}) \\
\sqsupseteq & \quad \{ \text{ (6.18) } \} \\
& (F.R){>}
\end{array}
$$

Distribution of relators over the equivalence domain also gives inclusions for both SPEC and per order:

Lemma 6.22: (Equivalence domains)

$$
\begin{array}{l}
(F.R){\succ} \sqsupseteq F.(R{\succ}) \\
(F.R){\succ} \lhd F.(R{\succ})
\end{array}
$$

□

Proof:

$$
\begin{array}{ll}
 & (F.R)_{\succ} \\
= & \quad \{ \text{ def } \succ \ \} \\
 & (F.R)_{\searrow} \circ (F.R)_{>} \\
\sqsupseteq & \quad \{ \text{ greatests domains } \} \\
 & F.(R_{\searrow}) \circ (F.R)_{>} \\
= & \quad \{ \text{ relators } \} \\
 & F.(R_{\searrow}) \circ F.I \circ (F.R)_{>} \\
= & \quad \{ \text{ least domains } \} \\
 & F.(R_{\searrow}) \circ F.(R_{>}) \\
= & \quad \{ \text{ relators, def } \succ \ \} \\
 & F.(R_{\succ})
\end{array}
$$

$$
\begin{array}{ll}
 & (F.R)_{\succ} \lhd F.(R_{\succ}) \\
\equiv & \quad \{ \text{ equivalence domains } \} \\
 & F.R \circ F.(R_{\succ}) = F.R \\
\equiv & \quad \{ \text{ relators, equivalence domains } \} \\
 & true
\end{array}
$$

□

These inclusions are valid for all relators, but it turns out that there is almost always an equality. For difunctionals there is always $(F.f)_{\succ} = F.f \circ F.f_{\cup} = F.(f \circ f_{\cup}) = F.(f_{\succ})$ and we will show that there is distribution for almost every relator that we can construct. There are no counterexamples for distribution at the moment, although Oege de Moor discovered that the finite set functor from the category **Set** (related to the *List*-functor, but constructs finite sets instead of lists) would provide a counterexample when it is extended to relations. Unfortunately, we have not been able to prove that there exists a finite set relator in the SPEC-calculus. We come back to the problems with this relator in the chapter about types with equations.

So, for the moment, it is not clear whether distribution is valid for all specs and all relators, but all relators encountered so far do distribute. The reader can prove this for himself for the identity, constant, doubling and selection relators. The three methods for the construction of relators from other relators preserve the distribution properties:

Lemma 6.23:

If $(F.R)_{\succ} = F.(R_{\succ})$ and $(G.R)_{\succ} = G.(R_{\succ})$ for all specs R then:

$$
\begin{array}{lrcl}
\text{(a)} & FG.(R_{\succ}) & = & (FG.R)_{\succ} \\
\text{(b)} & (F,G).((R,S)_{\succ}) & = & ((F,G).(R,S))_{\succ} \\
\text{(c)} & F^{\gamma}.(R_{\succ}) & = & (F^{\gamma}.R)_{\succ}
\end{array}
$$

□

The proofs for FG and (F,G) are simple and left to the reader (for (F,G) use that $(R,S)_{\succ} = (R_{\succ}, S_{\succ})$), but the proof for F^{γ} is not easy and is given here:

The first step is to show that $\gamma \circ (F^{\gamma}.R)_{\succ} \circ \gamma_{\cup}$ is a per:

$$
\begin{array}{ll}
 & \gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup \circ (\gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup)_\cup \\
= & \qquad \{ \text{ reverse } \} \\
 & \gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup \circ \gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup \\
= & \qquad \{ \text{ def } F^\gamma \ \} \\
 & \gamma \circ (F^\gamma.R)_\succ \circ F^\gamma.I \circ (F^\gamma.R)_\succ \circ \gamma_\cup \\
= & \qquad \{ \ F^\gamma.R \circ F^\gamma.I = F^\gamma.R, \text{ (5.50) } \} \\
 & \gamma \circ (F^\gamma.R)_\succ \circ (F^\gamma.R)_\succ \circ \gamma_\cup \\
= & \qquad \{ \text{ pers } \} \\
 & \gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup
\end{array}
$$

This is used in the following calculation:

$$
\begin{array}{ll}
 & (F.R)_\succ = (F.R)_\succ \circ \gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup \\
\equiv & \qquad \{ \ per.(\gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup), \ \text{(5.50)} \ \} \\
 & F.R = F.R \circ \gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup \\
\equiv & \qquad \{ \text{ def } F^\gamma \ \} \\
 & F.R = \gamma \circ F^\gamma.R \circ (F^\gamma.R)_\succ \circ \gamma_\cup \\
\equiv & \qquad \{ \text{ domains } \} \\
 & F.R = \gamma \circ F^\gamma.R \circ \gamma_\cup \\
\equiv & \qquad \{ \text{ def } F^\gamma \ \} \\
 & true
\end{array}
$$

Now it's time for the proof of the distribution, assuming $(F.R)_\succ = F.(R_\succ)$:

$$
\begin{array}{ll}
 & F^\gamma.(R_\succ) \\
= & \qquad \{ \text{ def } F^\gamma \ \} \\
 & \gamma_\cup \circ F.(R_\succ) \circ \gamma \\
= & \qquad \{ \ \bullet \ F.(R_\succ) = (F.R)_\succ \ \} \\
 & \gamma_\cup \circ (F.R)_\succ \circ \gamma \\
= & \qquad \{ \text{ see above } \} \\
 & \gamma_\cup \circ (F.R)_\succ \circ \gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup \circ \gamma \\
= & \qquad \{ \ \bullet \ F.(R_\succ) = (F.R)_\succ \ \} \\
 & \gamma_\cup \circ F.(R_\succ) \circ \gamma \circ (F^\gamma.R)_\succ \circ \gamma_\cup \circ \gamma \\
= & \qquad \{ \text{ def } F^\gamma \ \} \\
 & F^\gamma.(R_\succ) \circ (F^\gamma.R)_\succ \circ F^\gamma.I \\
= & \qquad \{ \ F^\gamma.R \circ F^\gamma.I = F^\gamma.R, \text{ (5.50) } \} \\
 & F^\gamma.(R_\succ) \circ (F^\gamma.R)_\succ \\
= & \qquad \{ \text{ (6.22) } \} \\
 & (F^\gamma.R)_\succ
\end{array}
$$

□

A remarkable property of relators with as domain a unary extensional SPEC-calculus is that they are either constant, or order-isomorphisms (and thus injective). This is expressed in the following lemma:

Lemma 6.24: (Relators) For all non-constant relators F and specs R, S from a unary extensional SPEC-calculus:

$$F.R \sqsupseteq F.S \;\equiv\; R \sqsupseteq S$$

□

This lemma has to be rewritten a bit before it can be proved. Constant relators are characterized by $F.\perp\!\!\perp = F.\top\!\!\top$ and the "$\equiv$" can be replaced by a "$\Rightarrow$" since the other implication follows from monotonicity. This gives us:

$$F.\perp\!\!\perp \neq F.\top\!\!\top \;\wedge\; F.R \sqsupseteq F.S \;\Rightarrow\; R \sqsupseteq S.$$

Shunting twice delivers the version of the lemma that is proved here:

$$\neg(R \sqsupseteq S) \;\wedge\; F.R \sqsupseteq F.S \;\Rightarrow\; F.\perp\!\!\perp = F.\top\!\!\top$$

Rewriting $\neg(R \sqsupseteq S)$ gives us

$$
\begin{array}{ll}
 & \neg(R \sqsupseteq S) \\
\equiv & \quad \{ \text{ extensionality } \} \\
 & \neg\forall(Z \mathrel{¦} singleton.Z \;\;\wedge\;\; S \sqsupseteq Z \mathrel{¦} R \sqsupseteq Z) \\
\equiv & \quad \{ \text{ De Morgan } \} \\
 & \exists(Z \mathrel{¦} singleton.Z \;\;\wedge\;\; S \sqsupseteq Z \mathrel{¦} \neg(R \sqsupseteq Z)) \\
\equiv & \quad \{ \text{ atomicity } \} \\
 & \exists(Z \mathrel{¦} singleton.Z \;\;\wedge\;\; S \sqsupseteq Z \mathrel{¦} R \sqcap Z = \perp\!\!\perp)
\end{array}
$$

We assume that singleton a satisfies $S \sqsupseteq a$ and $R \sqcap a = \perp\!\!\perp$ in the following proof:

$$
\begin{array}{ll}
 & F.\perp\!\!\perp \\
= & \quad \{ \;\bullet\; R \sqcap a = \perp\!\!\perp\text{; zero } \} \\
 & F.(\top\!\!\top \circ a^{\cup} \circ (R \sqcap a) \circ a^{\cup} \circ \top\!\!\top) \\
= & \quad \{ \text{ distribution, singletons are difunctional } \} \\
 & F.(\top\!\!\top \circ (a^{\cup} \circ R \circ a^{\cup} \;\sqcap\; a^{\cup} \circ a \circ a^{\cup}) \circ \top\!\!\top) \\
= & \quad \{ \; a^{\cup} \circ a \circ a^{\cup} = a^{\cup} = a^{\cup} \circ \top\!\!\top \circ a^{\cup} \text{, monotonicity } \} \\
 & F.(\top\!\!\top \circ a^{\cup} \circ R \circ a^{\cup} \circ \top\!\!\top) \\
\sqsupseteq & \quad \{ \;\bullet\; F.R \sqsupseteq F.S\text{; relators } \} \\
 & F.(\top\!\!\top \circ a^{\cup} \circ S \circ a^{\cup} \circ \top\!\!\top) \\
\sqsupseteq & \quad \{ \;\bullet\; S \sqsupseteq a \; \} \\
 & F.(\top\!\!\top \circ a^{\cup} \circ a \circ a^{\cup} \circ \top\!\!\top) \\
= & \quad \{ \text{ singletons are difunctional } \} \\
 & F.(\top\!\!\top \circ a^{\cup} \circ \top\!\!\top) \\
= & \quad \{ \text{ singletons are non-empty, cone rule } \} \\
 & F.\top\!\!\top
\end{array}
$$

Note that this proof does not use that F commutes with $^{\cup}$. Every non-constant monotonic function that distributes over $\circ$ is injective.

6.4 Natural transformations

Natural transformations are introduced following the same strategy as used with functors; construct a notion in the SPEC-calculus that can be used as a natural transformation in **Difun**. For relators F and G a natural transformation τ from F to G can be constructed by defining

$$\tau.A \;=\; (G.A \circ \gamma \circ F.A, G.A, F.A)$$

if γ satisfies the following conditions: for all difunctionals f,

$$G.f \circ \gamma \sqsupseteq \gamma \circ F.f$$
$$\gamma : G.I \longleftarrow\!\!\!\dashv F.I$$

The type judgement for γ is used for proving that $\tau.A$ is an arrow between the correct objects. The proof is easy and left to the reader. Now we can prove the other part of the definition of a natural transformation:

$$G.(f, A, B) \circ \tau.B \;=\; \tau.A \circ F.(f, A, B)$$

The proof of this equality is trivial for the domain parts; the difunctional part is proved by cyclic inclusion:

$$
\begin{array}{ll}
 & G.f \circ G.B \circ \gamma \circ F.B \\
= & \qquad \{ \;\bullet\; f : A \longleftarrow\!\!\!\dashv B;\ \text{relators} \;\} \\
 & G.A \circ G.f \circ \gamma \circ F.B \\
\sqsupseteq & \qquad \{ \;\bullet\; G.f \circ \gamma \;\sqsupseteq\; \gamma \circ F.f \;\} \\
 & G.A \circ \gamma \circ F.f \circ F.B \\
= & \qquad \{ \;\bullet\; f : A \longleftarrow\!\!\!\dashv B;\ \text{relators} \;\} \\
 & G.A \circ \gamma \circ F.A \circ F.f \\
\sqsupseteq & \qquad \{ \;\bullet\; f : A \longleftarrow\!\!\!\dashv B;\ \text{relators} \;\} \\
 & G.f \circ G.f^{\cup} \circ \gamma \circ F.f \\
\sqsupseteq & \qquad \{ \;\bullet\; G.f^{\cup} \circ \gamma \;\sqsupseteq\; \gamma \circ F.f^{\cup};\ \text{relators} \;\} \\
 & G.f \circ \gamma \circ F.f^{\cup} \circ F.f \\
\sqsupseteq & \qquad \{ \;\bullet\; f : A \longleftarrow\!\!\!\dashv B;\ \text{relators} \;\} \\
 & G.f \circ G.B \circ \gamma \circ F.B
\end{array}
$$

The desired equality is between the first and the fourth line. The proof given here was inspired by a proof constructed by Paul Hoogendijk. As with relators, we don't want difunctionality conditions in our definitions and this leads to the following definition for a natural transformation in the SPEC-calculus:

Definition 6.25:(Natural Transformation)

For relators F and G, a spec γ is a *natural transformation* from F to G (notation $\gamma : G \overset{\cdot}{\leftsquigarrow} F$) iff for all specs R:

$$G.R \circ \gamma \sqsupseteq \gamma \circ F.R$$
$$\gamma : G.I \text{---} F.I$$

□

Note that not every natural transformation according to this definition also gives a categorical natural transformation in **Difun**. Functionality and totality of γ are also necessary for this. Also note the type judgement notation for naturality; the naturality type of a spec gives us important information about the algebraic properties of the spec and naturality properties can often be derived in a way similar to the way conventional type judgements can be derived. A collection of naturality type inference rules is given later in this section.

Sometimes there is a need for other notions of naturality in our calculations. We can use the reverse inclusion from the SPEC-calculus or strengthen even further to equality. This leads to the following definitions:

Definition 6.26: (Natural Transformation)

For relators F and G, spec γ, naturality-type judgements $\stackrel{.}{\rightsquigarrow}$, $\stackrel{.}{\leftsquigarrow}$ and $\stackrel{.}{\leftrightsquigarrow}$ are defined by:

$$\begin{array}{rcl} \gamma : G \stackrel{.}{\rightsquigarrow} F & \triangleq & \gamma : G.I \text{—} F.I \wedge \forall(R \,\vdots\vdots\, G.R \circ \gamma \sqsubseteq \gamma \circ F.R) \\ \gamma : G \stackrel{.}{\leftsquigarrow} F & \triangleq & \gamma : G.I \text{—} F.I \wedge \forall(R \,\vdots\vdots\, G.R \circ \gamma \sqsupseteq \gamma \circ F.R) \\ \gamma : G \stackrel{.}{\leftrightsquigarrow} F & \triangleq & \gamma : G.I \text{—} F.I \wedge \forall(R \,\vdots\vdots\, G.R \circ \gamma = \gamma \circ F.R) \end{array}$$

□

A simple example of a natural transformation is found in the so-called isomorphic relators:

$$\gamma : F \stackrel{.}{\leftrightsquigarrow} F^{\gamma}$$

The easy proof of this claim is left to the reader. An informal interpretation of natural transformations is that they are transformations between relators and do not "look into" the arguments of these relators. This can be seen more formally in the strong link that exists between naturality properties and polymorphism of functions. See Reynolds [51], Wadler [58] or De Bruin [14] for more information about this subject.

One of the uses of natural transformations is in the construction of relators from other relators:

Lemma 6.27: (Subrelators)

For a difunctional spec $\gamma : F \stackrel{.}{\leftrightsquigarrow} F$, define the relator F_{γ} by

$$F_{\gamma}.R \triangleq \gamma^{\cup} \circ F.R \circ \gamma$$

□

This is the same formula that was used for isomorphic relators, but the condition on γ is different. The proof that the subrelator construction indeed constructs a relator is trivial and left to the reader. This method of construction is called subrelator construction because for every per A:

$$F_{\gamma}.A \lhd F.A$$

Extra conditions on γ allow even stronger results:

$$\begin{array}{l} \gamma : \rightarrow F.I \;\Rightarrow\; F_{\gamma}.A \subseteq F.A \\ \gamma : \longrightarrow\!\!\!\!\mid F.I \;\Rightarrow\; F_{\gamma}.A \lhd\!\mid F.A \end{array}$$

We already saw that relational type judgements are not unique. A spec can satisfy more than one judgement. The same goes for naturality type judgements and subrelators provide us with an example of this. For a difunctional γ with $\gamma : F \overset{\cdot}{\leftrightsquigarrow} F$ there are three other valid judgements:

$$\begin{array}{l} \gamma : F \overset{\cdot}{\leftrightsquigarrow} F_{\gamma} \\ \gamma : F_{\gamma^{\cup}} \overset{\cdot}{\leftrightsquigarrow} F \\ \gamma : F_{\gamma^{\cup}} \overset{\cdot}{\leftrightsquigarrow} F_{\gamma} \end{array}$$

There is not only a natural transformation between $F_{\gamma^{\cup}}$ and F_{γ}, but even a *natural isomorphism*. This is the combination of having a natural transformation between the relators and the relators being isomorphic to each other. In this example we have for every per A that $F_{\gamma^{\cup}}.A$ is isomorphic to $F_{\gamma}.A$ with isomorphisms $F.A \circ \gamma$ and $\gamma^{\cup} \circ F.A$.

Although the subrelator construction looks very similar to the construction of isomorphic relators, they are clearly different in their algebraic properties and their use. Isomorphic relators are mainly used in the proofs of properties of other constructions, while subrelators are important as a method for the design of datatypes. An open problem for subrelators is whether they distribute over the equivalence domain operator. All other methods (in this and following chapters) for the construction of relators preserve or guarantee distribution but distribution of subrelators is still an open problem.

This section is concluded with a collection of naturality-type judgement rules. Only rules for the $\overset{\cdot}{\leftsquigarrow}$ judgements are given. The reader can derive the rules for the other judgements using $\gamma : F \overset{\cdot}{\rightsquigarrow} G \equiv \gamma^{\cup} : G \overset{\cdot}{\leftsquigarrow} F$ and $\gamma : F \overset{\cdot}{\leftrightsquigarrow} G \equiv \gamma : F \overset{\cdot}{\leftsquigarrow} G \wedge \gamma : F \overset{\cdot}{\rightsquigarrow} G$. The proof of the lemma is easy and left to the reader.

Lemma 6.28:(Natural Transformation)

$$\begin{array}{rcl} \alpha \circ \beta : F \overset{\cdot}{\leftsquigarrow} G & \Leftarrow & \alpha : F \overset{\cdot}{\leftsquigarrow} H \wedge \beta : H \overset{\cdot}{\leftsquigarrow} G \\ H.\gamma : HF \overset{\cdot}{\leftsquigarrow} HG & \Leftarrow & \gamma : F \overset{\cdot}{\leftsquigarrow} G \\ FH.I \circ \gamma \circ GH.I : FH \overset{\cdot}{\leftsquigarrow} GH & \Leftarrow & \gamma : F \overset{\cdot}{\leftsquigarrow} G \\ \bigsqcup(\gamma \mid \gamma \in \Gamma \mid \gamma) : F \overset{\cdot}{\leftsquigarrow} G & \Leftarrow & \forall(\gamma \mid \gamma \in \Gamma \mid \gamma : F \overset{\cdot}{\leftsquigarrow} G) \end{array}$$

□

Chapter 7

Disjoint sum and Cartesian product relators

Disjoint sum and Cartesian product are probably the most common methods used for the construction of datatypes. This chapter starts by giving a categorical definition of disjoint sum and Cartesian product functors, followed by a construction of the corresponding relators. The SPEC properties of these relators are examined in the final section of this chapter. The SPEC axiomatization as developed in this chapter is based on the work presented in [1].

7.1 Categorical construction

The method that is used for the introduction of disjoint sum and Cartesian product in this section is not the only possible one. Other conventional introductions use for example categorical notions like adjunctions, but it is our intention to restrict the number of category theory notions to a minimum, and it is possible to do without these notions. The construction in this chapter is equivalent to using (co-)products. The terms disjoint sums and Cartesian product are normally only used for the (co-) product notions in the category **Set**, but the interpretation and properties are very similar in **Difun** and therefore we borrow these names.

The approach of this thesis is to use the already given notions of initial and terminal objects for the definition. The constructions for disjoint sum and Cartesian product have a similar structure and we will only do the construction of a disjoint sum in full detail. The construction of the Cartesian product is only sketched.

Given a category $\mathcal{C}$ we can construct for every pair of objects A, B of $\mathcal{C}$ a category $\Sigma_{A,B}$ defined by:

- Objects: pairs (f, g) with f and g arrows of $\mathcal{C}$ such that $dom.f = A$, $dom.g = B$ and $cod.f = cod.g$.

- Arrows: pairs $(h,(f,g))$ with h an arrow of $\mathcal{C}$ and (f,g) an object of $\Sigma_{A,B}$ such that $dom.h = cod.f = cod.g$.
- Domains: $dom.(h,(f,g)) = (f,g)$ and $cod.(h,(f,g)) = (h \circ f, h \circ g)$.
- Composition: $(m,(k,l)) \circ (h,(f,g)) = (m \circ h,(f,g))$.
- Identity: $id.(f,g) = (id.(cod.f),(f,g)) = (id.(cod.g),(f,g))$.

The reader can check for himself that this indeed defines a category. Now category $\mathcal{C}$ has (binary) disjoint sums iff for every pair of objects A, B from $\mathcal{C}$ there exists an initial object $(\hookrightarrow_{A,B}, \hookleftarrow_{A,B})$ in $\Sigma_{A,B}$.

The fact that an initial object is an object gives us three properties for the so called *injections*:

$$\begin{array}{l} dom.{\hookrightarrow_{A,B}} = A \\ dom.{\hookleftarrow_{A,B}} = B \\ cod.{\hookrightarrow_{A,B}} = cod.{\hookleftarrow_{A,B}} \end{array}$$

From the initiality follows that for every pair (f,g) of arrows of $\mathcal{C}$ with $cod.f = cod.g$ there is a unique arrow $(h,(\hookrightarrow_{A,B}, \hookleftarrow_{A,B}))$ from $(\hookrightarrow_{A,B}, \hookleftarrow_{A,B})$ to (f,g) in $\Sigma_{A,B}$ where $A = dom.f$ and $B = dom.g$. The h in this unique arrow is denoted by $f \triangledown_{A,B} g$. Using the definition of the codomain and the existence and uniqueness of the arrow we can see that $f \triangledown_{A,B} g$ is characterized by:

$$h = f \triangledown_{A,B} g \ \equiv \ h \circ {\hookrightarrow_{A,B}} = f \ \wedge \ h \circ {\hookleftarrow_{A,B}} = g$$

Using the $\triangledown$ as defined above we can construct a binary functor $+$ from $\mathcal{C} \times \mathcal{C}$ to $\mathcal{C}$. This is done as follows:

- For objects A and B from $\mathcal{C}$: $A + B = cod.{\hookrightarrow_{A,B}} = cod.{\hookleftarrow_{A,B}}$.
- For arrows f and g from $\mathcal{C}$: $f + g = (\hookrightarrow_{A,B} \circ f) \triangledown_{dom.f, dom.g} (\hookleftarrow_{A,B} \circ g)$ where $A = cod.f$ and $B = cod.g$

The proof that $+$ is a functor is simple and left to the reader. Interpretation of the constructions above in **Set** shows that we are constructing disjoint sums. For two sets A and B there are two injection functions mapping them to disjoint parts ("left" and "right") of set $A + B$. $f \triangledown g$ is a kind of case statement: when applied to an element of a disjoint sum it applies f if the argument comes from the "left" part and g if the argument comes from the "right" part.

The construction of the Cartesian product is categorically dual to the disjoint sum construction. For every pair of objects A and B from $\mathcal{C}$ construct the category $\Pi_{A,B}$ defined by:

- Objects: pairs (f,g) with f and g arrows of $\mathcal{C}$ such that $cod.f = A$, $cod.g = B$ and $dom.f = dom.g$.

- Arrows: pairs $(h,(f,g))$ with h an arrow of $\mathcal{C}$ and (f,g) an object of $\Pi_{A,B}$ such that $cod.h = dom.f = dom.g$.
- Domains: $cod.(h,(f,g)) = (f,g)$ and $dom.(h,(f,g)) = (f \circ h, g \circ h)$.
- Composition: $(m,(k,l)) \circ (h,(f,g)) = (m \circ h,(k,l))$.
- Identity: $id.(f,g) = (id.(dom.f),(f,g)) = (id.(dom.g),(f,g))$.

Category $\mathcal{C}$ has (binary) Cartesian products iff for every pair of objects A and B of $\mathcal{C}$ there exists a terminal object $(\ll_{A,B}, \gg_{A,B})$ in $\Pi_{A,B}$. The so called *projections* satisfy:

$$\begin{aligned} cod.\ll_{A,B} &= A \\ cod.\gg_{A,B} &= B \\ dom.\ll_{A,B} &= dom.\gg_{A,B} \end{aligned}$$

The unique arrow from (f,g) to the corresponding terminal object has as first component $f \vartriangle_{A,B} g$, which is characterized by:

$$h = f \vartriangle_{A,B} g \;\equiv\; \ll_{A,B} \circ h = f \;\wedge\; \gg_{A,B} \circ h = g$$

The functor $\times$ is defined as follows:

- For objects A and B from $\mathcal{C}$: $A \times B = dom.\ll_{A,B} = dom.\gg_{A,B}$.
- For arrows f and g from $\mathcal{C}$: $f \times g = (f \circ \ll_{A,B}) \vartriangle_{cod.f,cod.g} (g \circ \gg_{A,B})$ where $A = dom.f$ and $B = dom.g$

Interpretation in **Set**: for two sets A and B we can interpret $A \times B$ as the set of pairs (x,y) with $x \in A$ and $y \in B$. The function $f \vartriangle g$ maps a value x to the pair $(f.x, g.x)$

7.2 SPEC construction

The goal of this section is to establish sufficient conditions under which there exist disjoint sums and Cartesian products in the category **Difun**. This is done by constructing $\hookrightarrow, \hookleftarrow, \triangledown, \ll, \gg$ and $\vartriangle$ satisfying the conditions in the previous section and ensuring that the $+$ and $\times$ definitions construct relators.

The injections and projections are families of specs, but we can use the same approach as with natural transformations: take one spec and the family is constructed by composing with the appropriate domain:

$$\begin{aligned} \hookrightarrow_{A,B} &= A{+}B \circ \hookrightarrow \\ \hookleftarrow_{A,B} &= A{+}B \circ \hookleftarrow \\ \ll_{A,B} &= \ll \circ A{\times}B \\ \gg_{A,B} &= \gg \circ A{\times}B \end{aligned}$$

The $+$ and $\times$ in these formulas are relators that we still have to construct. From the *dom* and *cod* properties of the projections and injections the following judgements are required:

$$\begin{array}{l}\hookrightarrow_{A,B} : A{+}B \leftarrow\!\!\dashv A \\ \hookleftarrow_{A,B} : A{+}B \leftarrow\!\!\dashv B \\ \ll_{A,B} : A \leftarrow\!\!\dashv A{\times}B \\ \gg_{A,B} : B \leftarrow\!\!\dashv A{\times}B\end{array}$$

And the type judgements for $\triangledown$ and $\vartriangle$:

$$\begin{array}{l}f \triangledown_{A,B} g : C \leftarrow\!\!\dashv A{+}B \;\Leftarrow\; f : C \leftarrow\!\!\dashv A \wedge g : C \leftarrow\!\!\dashv B \\ f \vartriangle_{A,B} g : A{\times}B \leftarrow\!\!\dashv C \;\Leftarrow\; f : A \leftarrow\!\!\dashv C \wedge g : B \leftarrow\!\!\dashv C\end{array}$$

Now we can derive SPEC-formulas for $\triangledown, \vartriangle, +$ and $\times$. For f and g such that $f : C \leftarrow\!\!\dashv A$ and $g : C \leftarrow\!\!\dashv B$ we calculate:

$$\begin{array}{ll} & f \triangledown_{A,B} g \\ = & \quad \{ \text{ domains } \} \\ & f \triangledown_{A,B} g \circ A{+}B \\ \sqsupseteq & \quad \{ \text{ type } \hookrightarrow_{A,B} \} \\ & f \triangledown_{A,B} g \circ \hookrightarrow_{A,B} \circ \hookrightarrow_{A,B}{}^{\cup} \\ = & \quad \{ \text{ characterization } \triangledown \} \\ & f \circ \hookrightarrow_{A,B}{}^{\cup}\end{array}$$

Using this method one can also derive $f \triangledown_{A,B} g \sqsupseteq g \circ \hookleftarrow_{A,B}{}^{\cup}$. Combining the two results gives us:

$$f \triangledown_{A,B} g \sqsupseteq f \circ \hookrightarrow_{A,B}{}^{\cup} \sqcup g \circ \hookleftarrow_{A,B}{}^{\cup}$$

This suggests the following definition for $\triangledown$:

Definition 7.1: (Junc)

$$R \triangledown S \;\triangleq\; R \circ \hookrightarrow^{\cup} \sqcup S \circ \hookleftarrow^{\cup}$$

□

Applying the definition of the +-functor from the previous section gives us a definition for the + relator:

Definition 7.2: (Disjoint Sum)

$$R{+}S \;\triangleq\; (\hookrightarrow \circ R) \triangledown (\hookleftarrow \circ S) \;=\; \hookrightarrow \circ R \circ \hookrightarrow^{\cup} \sqcup \hookleftarrow \circ S \circ \hookleftarrow^{\cup}$$

□

A similar calculation for the Cartesian product leads to the following definitions:

Definition 7.3: (Split)

$$R \vartriangle S \;\triangleq\; \ll^{\cup} \circ R \sqcap \gg^{\cup} \circ S$$

□

Definition 7.4: (Cartesian Product)

$$R{\times}S \;\triangleq\; (R \circ \ll) \vartriangle (S \circ \gg) \;=\; \ll^{\cup} \circ R \circ \ll \sqcap \gg^{\cup} \circ S \circ \gg$$

□

These definitions are not enough to guarantee that we have constructed disjoint sums and Cartesian products in the category Difun. The first thing to check is whether

$+$ and $\times$ are indeed relators. The fact that the monotonicity and commuting with $^{\cup}$ conditions are satisfied follows directly from the shape of the defining expressions, but extra axioms are necessary for the distribution over composition. We could add as axioms that $+$ and $\times$ defined as above are relators, but these axioms would be rather weak. The conditions of the categories also state properties of the composition of junc and split with injections or projections and about domains of junc or split, and only demanding distribution over composition does not give enough information. We postulate the following axioms:

Axiom 7.5: (Disjoint Sum)

$$(R \circ \hookrightarrow^{\cup} \sqcup S \circ \hookleftarrow^{\cup}) \circ (\hookrightarrow \circ T \sqcup \hookleftarrow \circ U) \;=\; R \circ T \sqcup S \circ U$$

□

Axiom 7.6: (Cartesian Product)

$$(R \circ \ll \sqcap S \circ \gg) \circ (\ll^{\cup} \circ T \sqcap \gg^{\cup} \circ U) \;=\; R \circ T \sqcap S \circ U$$

□

A simple expansion of the definitions shows that these axioms guarantee that $+$ and $\times$ distribute over composition, so the remaining condition for relators is fulfilled.

The next step of the construction of disjoint sum and Cartesian product in **Difun** is making sure that the types of the injections, projections, junc and split are correct. This is done by first determining the types of $\hookrightarrow$, $\hookleftarrow$, $\ll$ and $\gg$ according to the SPEC-judgements and then deriving from these the types of injections and projections.

The types of $\hookrightarrow$ and $\hookleftarrow$ are simple to determine: instantiating (7.2) with I for R and $\perp\!\!\perp$ for S gives

$$\hookrightarrow \circ \hookrightarrow^{\cup} \;=\; I + \perp\!\!\perp$$

And instantiating (7.5) with I for R, $\perp\!\!\perp$ for S, I for T and $\perp\!\!\perp$ for U gives:

$$\hookrightarrow^{\cup} \circ \hookrightarrow \;=\; I$$

From this calculation (and a dual one for $\hookleftarrow$) we conclude:

Lemma 7.7: (Disjoint Sum)

$$\hookrightarrow : I + \perp\!\!\perp \mathrel{|\!\!\leftrightarrow\!\!|} I$$
$$\hookleftarrow : \perp\!\!\perp + I \mathrel{|\!\!\leftrightarrow\!\!|} I$$

□

Until now we had complete duality between disjoint sum and Cartesian product, but this duality is broken if we want to use the axioms given above to determine the types of $\ll$ and $\gg$. Dualizing (7.7) we desire

$$\ll^{\cup} \circ \ll \;=\; I \times \top\!\!\top$$
$$\ll \circ \ll^{\cup} \;=\; I$$

but this can not be derived from (7.4) and (7.6) alone. Instantiation of the axioms in the previous formulas leads to following equalities:

$$\ll^{\cup} \circ \ll \;=\; \ll^{\cup} \circ \ll \sqcap \gg^{\cup} \circ \top\!\!\top \circ \gg$$
$$\ll \circ \ll^{\cup} \;=\; (\ll \sqcap \top\!\!\top \circ \gg) \circ (\ll^{\cup} \sqcap \gg^{\cup} \circ \top\!\!\top)$$

These equalities can be established by adding an extra axiom for the projections:

Axiom 7.8: (Cartesian Product)

$$\top\!\!\top \circ \ll \;=\; \top\!\!\top \circ \gg$$

□

The corresponding axiom for disjoint sum would be $\hookrightarrow \circ \perp\!\!\perp = \hookleftarrow \circ \perp\!\!\perp$, but this already follows from the normal SPEC-axioms. Using the extra axiom we can dualize the calculations for the type of the injections and find as type for the projections:

Lemma 7.9: (Cartesian Product)

$$\ll : I \leftarrow\!\!\!\!\mapsto\!\!\!\!\twoheadrightarrow I \times \top\!\!\top$$
$$\gg : I \leftarrow\!\!\!\!\mapsto\!\!\!\!\twoheadrightarrow \top\!\!\top \times I$$

□

Other useful properties of injections and projections follow by instantiating (7.5) and (7.6) with I for R and U, and (disjoint sums) $\perp\!\!\perp$ or (Cartesian product) $\top\!\!\top$ for S and T:

$$\hookrightarrow^{\cup} \circ \hookleftarrow \;=\; \perp\!\!\perp$$
$$\ll \circ \gg^{\cup} \;=\; \top\!\!\top$$

Now it's time to calculate the type of the categorical injections and projections. The complete calculation for $\ll_{A,B}$ is given as an example and afterwards we generalize the results to the other projections and injections.

$$
\begin{array}{ll}
 & \ll_{A,B}{}^{\cup} \circ \ll_{A,B} \\
= & \qquad \{ \text{ def } \ll_{A,B} \text{ and } \gg_{A,B} \ \} \\
 & A\times B \circ \ll^{\cup} \circ \ll \circ A\times B \\
= & \qquad \{ \ \ll : \twoheadrightarrow I \times \top\!\!\top \ \} \\
 & A\times B \circ I \times \top\!\!\top \circ A\times B \\
= & \qquad \{ \text{ relators } \} \\
 & A \times\ (B \circ \top\!\!\top \circ B)
\end{array}
$$

$$
\begin{array}{ll}
 & \ll_{A,B} \circ \ll_{A,B}{}^{\cup} \\
= & \qquad \{ \text{ def } \ll_{A,B} \text{ and } \gg_{A,B} \ \} \\
 & \ll \circ A\times B \circ \ll^{\cup} \\
= & \qquad \{ \text{ Cartesian product } \} \\
 & \ll \circ (\ll^{\cup} \circ A \circ \ll \sqcap \gg^{\cup} \circ B \circ \gg) \circ \ll^{\cup} \\
= & \qquad \{ \text{ distribution } \} \\
 & \ll \circ \ll^{\cup} \circ A \circ \ll \circ \ll^{\cup} \sqcap \ll \circ \gg^{\cup} \circ B \circ \gg \circ \ll^{\cup} \\
= & \qquad \{ \ \ll : I \leftarrow\!\!\!\!\mapsto,\ \ll \circ \gg^{\cup} = \top\!\!\top \ \} \\
 & A \sqcap \top\!\!\top \circ B \circ \top\!\!\top
\end{array}
$$

Dualizing these calculations to $\hookrightarrow_{A,B}$ gives as result:

$$\hookrightarrow_{A,B} \circ \hookrightarrow_{A,B}{}^{\cup} \;=\; A{+}(A \circ \perp\!\!\perp \circ A) \;=\; A{+}\perp\!\!\perp$$
$$\hookrightarrow_{A,B}{}^{\cup} \circ \hookrightarrow_{A,B} \;=\; A \sqcup \perp\!\!\perp \circ B \circ \perp\!\!\perp \;=\; A$$

Combining the results above with the difunctionality of categorical injections and projections (easy proof, left to the reader) and using lemma 5.77 with $A \supseteq A \sqcap \top\!\!\top \circ B \circ \top\!\!\top$, $A\times\,(B \circ \top\!\!\top \circ B) \lhd\!\mid A{\times}B$ and $A{+}B \supseteq A{+}\bot\!\!\bot$ we arrive at the following judgements:

$$\begin{array}{l}
\hookrightarrow_{A,B} : A{+}B \mathrel{\leftrightarrow\!\!|} A \\
\hookleftarrow_{A,B} : A{+}B \mathrel{\leftrightarrow\!\!|} B \\
\ll_{A,B} : A \mathrel{\longleftarrow\!\!|} A{\times}B \\
\gg_{A,B} : B \mathrel{\longleftarrow\!\!|} A{\times}B
\end{array}$$

The results for $\hookleftarrow_{A,B}$ and $\gg_{A,B}$ were obtained by dualizing the derivations for $\hookrightarrow_{A,B}$ and $\ll_{A,B}$, respectively. Note that we derived a stronger result than necessary for $\hookleftarrow_{A,B}$ and $\hookrightarrow_{A,B}$. The injectivity of the injections was not stated in our original specifications but was found by calculation.

The final step in the proof that we constructed disjoint sums and Cartesian products is establishing the existence of the unique arrows. This has two components: the arrowness and the uniqueness. The arrowness is an instance of the relational type judgements for $\triangledown$ and $\vartriangle$:

Lemma 7.10:

$$\begin{array}{llll}
(a) & R{\vartriangle}S : A{\times}B\text{—} & \Leftarrow & R : A\text{—} \;\wedge\; S : B\text{—} \\
(b) & R{\vartriangle}S : A{\times}B\leftarrow & \Leftarrow & R : A\leftarrow \;\wedge\; S : B\leftarrow \\
(c) & R{\vartriangle}S : A{\times}B\vdash\!\text{—} & \Leftarrow & R : A\vdash\!\text{—} \;\wedge\; S : B\vdash\!\text{—} \;\wedge\; R \circ S^{\cup} = R \circ \top\!\!\top \circ S^{\cup} \\
(d) & R{\vartriangle}S : \text{—}A & \Leftarrow & R : \text{—}A \;\wedge\; S : \text{—}A \\
(e) & R{\vartriangle}S : \rightarrow A & \Leftarrow & (R : \rightarrow A \wedge S : \text{—}A) \;\vee\; (R : \text{—}A \wedge S : \rightarrow A) \\
(f) & R{\vartriangle}S : \text{—}\!\dashv A & \Leftarrow & R : \text{—}\!\dashv A \;\wedge\; S : \text{—}\!\dashv A \\
(g) & R{\triangledown}S : A\text{—} & \Leftarrow & R : A\text{—} \;\wedge\; S : A\text{—} \\
(h) & R{\triangledown}S : A\leftarrow & \Leftarrow & R : A\leftarrow \;\wedge\; S : A\leftarrow \\
(i) & R{\triangledown}S : A\vdash\!\text{—} & \Leftarrow & (R : A\vdash\!\text{—} \wedge S : A\text{—}) \;\vee\; (R : A\text{—} \wedge S : A\vdash\!\text{—}) \\
(j) & R{\triangledown}S : \text{—}A{+}B & \Leftarrow & R : \text{—}A \;\wedge\; S : \text{—}B \\
(k) & R{\triangledown}S : \rightarrow A{+}B & \Leftarrow & R : \rightarrow A \;\wedge\; S : \rightarrow B \;\wedge\; R^{\cup} \circ S = \bot\!\!\bot \\
(l) & R{\triangledown}S : \text{—}\!\dashv A{+}B & \Leftarrow & R : \text{—}\!\dashv A \;\wedge\; S : \text{—}\!\dashv B
\end{array}$$

□

The proof of this lemma is not difficult, except for part (7.10c). The proof of this part requires some more advanced manipulation with domain and distribution properties. The proofs of the other parts of the lemma are simple expansions of the definitions. Proving the lemma is left to the reader.

Parts (7.10b), (7.10f), (7.10h) and (7.10l) prove that we are indeed constructing arrows in **Difun**. Before we can continue with the uniqueness proof we need one more lemma about equality of arrows. In the category there is only equality between arrows, but in the SPEC-calculus we also have inclusions and we exploit this to prove equality of arrows in **Difun**. Equality can be proved by combining type judgements with a single inclusion:

Lemma 7.11:

$$f = g \;\Leftarrow\; f : A \mathrel{\text{—}\!\!\dashv} B \;\wedge\; g : A \leftarrow B \;\wedge\; g \sqsupseteq f$$

□

Proof:

$$
\begin{array}{ll}
 & f \\
\sqsupseteq & \quad \{ \ \bullet\ f : A\!\leftarrow\!\!-,\ g : A\!\leftarrow \ \} \\
 & g \circ g^{\cup} \circ f \\
\sqsupseteq & \quad \{ \ \bullet\ g \sqsupseteq f \ \} \\
 & g \circ f^{\cup} \circ f \\
\sqsupseteq & \quad \{ \ \bullet\ f : \,-\!\!\dashv B,\ g : -B \ \} \\
 & g
\end{array}
$$

□

Only the uniqueness of the arrow for Cartesian product is proved below, but this proof can be dualized to disjoint sums as well. We assume for f and g that:

$$f : A \leftarrow\!\!\dashv C \ \wedge \ g : B \leftarrow\!\!\dashv C$$

and have to prove that

$$h = f \triangle_{A,B} g \ \equiv \ \ll_{A,B} \circ h = f \ \wedge \ \gg_{A,B} \circ h = g$$

$f \triangle_{A,B} g$ is defined as $f \triangle g$ and both the lhs and the rhs of the equivalence above imply that $h : A \times B \leftarrow\!\!\dashv C$. The equivalence is proved by mutual implication. The proof obligation for the "⇒" direction is:

$$\ll_{A,B} \circ f \triangle g = f \ \wedge \ \gg_{A,B} \circ f \triangle g = g$$

and the proof obligation for the "⇐" direction is:

$$h = (\ll_{A,B} \circ h) \triangle (\gg_{A,B} \circ h)$$

We only prove the first conjunct for the first proof obligation, the proof for the second conjunct being similar. From the types we can deduce, using lemma 7.11, that it is sufficient to prove only an inclusion for the second proof obligation, we prove the "$\sqsubseteq$" inclusion.

Proof "⇒":

$$
\begin{array}{ll}
 & \ll_{A,B} \circ f \triangle g \\
= & \quad \{ \ \text{def} \ll_{A,B} \ \} \\
 & \ll \circ A \times B \circ f \triangle g \\
= & \quad \{ \ \bullet\ f : A-,\ g : B-;\ (7.10\text{a}) \ \} \\
 & \ll \circ f \triangle g \\
= & \quad \{ \ (7.8) \ \} \\
 & (\ll \sqcap \top\!\top \circ \gg) \circ f \triangle g \\
= & \quad \{ \ (7.6) \ \} \\
 & f \sqcap \top\!\top \circ g \\
= & \quad \{ \ \bullet\ g : \,-\!\!\dashv C,\ f : -C \ \} \\
 & f
\end{array}
$$

Proof "⇐":

$$
\begin{array}{ll}
 & (\ll_{A,B} \circ h) \triangle (\gg_{A,B} \circ h) \\
= & \quad \{ \ \text{def} \ll_{A,B},\ \gg_{A,B},\ \triangle \ \}
\end{array}
$$

$$
\begin{array}{ll}
 & \ll^{\cup} \circ \ll \circ A{\times}B \circ h \sqcap \gg^{\cup} \circ \gg \circ A{\times}B \circ h \\
= & \qquad \{ \bullet\, h : A{\times}B\text{—} \} \\
 & \ll^{\cup} \circ \ll \circ h \sqcap \gg^{\cup} \circ \gg \circ h \\
\sqsupseteq & \qquad \{ \text{def } \times,\ \text{monotonicity} \} \\
 & I{\times}I \circ h \\
= & \qquad \{ \bullet\, h : A{\times}B\text{—};\ A{\times}B \lhd I{\times}I \} \\
 & h
\end{array}
$$

This completes the proof that given specs $\hookrightarrow$, $\hookleftarrow$, $\ll$ and $\gg$ satisfying the axioms of this section we can construct disjoint sums and Cartesian products in **Difun**.

7.3 Algebraic properties

This chapter is concluded by a further investigation of the algebraic properties of disjoint sum and Cartesian product. While in the previous sections we were mainly concerned with properties that were necessary for **Difun**, in this section we will look at properties with respect to arbitrary specs and we investigate the naturality properties. Finally we will give some examples of natural transformations that can be used for the description of the notions of symmetry and associativity without points.

We start with some useful results about split and junc with respect to composition. The proofs of these properties are simple and left to the reader:

Lemma 7.12:

$$
\begin{array}{lrcll}
(a) & R\triangledown S \circ T{+}U & = & (R\circ T)\triangledown(S\circ U) & (Fusion) \\
(b) & T{\times}U \circ R\vartriangle S & = & (T\circ R)\vartriangle(U\circ S) & (Fusion) \\
(c) & R \circ S\triangledown T & = & (R\circ S)\triangledown(R\circ T) & (Distribution) \\
(d) & S\vartriangle T \circ R & \sqsubseteq & (S\circ R)\vartriangle(T\circ R) & (Distribution) \\
(e) & R\triangledown S \circ \hookrightarrow & = & R & (Calculation\ rule) \\
(f) & R\triangledown S \circ \hookleftarrow & = & S & (Calculation\ rule) \\
(g) & \ll \circ R\vartriangle S & = & R \circ S{>} & (Calculation\ rule) \\
(h) & \gg \circ R\vartriangle S & = & S \circ R{>} & (Calculation\ rule)
\end{array}
$$

□

The inclusion in (7.12d) comes from the fact that composition does not distribute over cap with equality. The theory from chapter 4 can be applied here for finding conditions under which distribution is allowed. The lemmas 4.7a, 4.7b and 4.7c can be instantiated for split giving the following result:

Lemma 7.13:(Distribution)

$$
\begin{array}{ll}
 & S\vartriangle T \circ R = (S\circ R)\vartriangle(T\circ R) \\
\Leftarrow & \\
 & S \sqsupseteq S\circ R\circ R^{\cup} \\
\vee & I \sqsupseteq R\circ R^{\cup} \\
\vee & S = S\circ \top\!\!\top
\end{array}
$$

□

The next lemma gives some domain properties. It is hard to say something useful about the least right domain of a junc or the least left domain of a split, but there are simple formulas for the other least domains. In the previous chapter we suggested that the equivalence domain commutes with almost every relator, and disjoint sum and Cartesian product are two relators that commute with the equivalence domain:

Lemma 7.14: (Domains)

$$
\begin{array}{lrcl}
\text{(a)} & (R \triangledown S)_< & = & R_< \sqcup S_< \\
\text{(b)} & (R \vartriangle S)_> & = & R_> \sqcap S_> \\
\text{(c)} & (R+S)_\succ & = & R_\succ + S_\succ \\
\text{(d)} & (R\times S)_\succ & = & R_\succ \times S_\succ
\end{array}
$$

□

The proofs of (7.14a) and (7.14b) are straightforward and left to the reader, but the proofs of the rest of the lemmas are quite complicated and are given in full detail here. We start off with a small lemma about equivalence domains:

Lemma 7.15:

$$R_\succ \lhd (R \sqcup S)_\succ \;\Leftarrow\; R_\cup \circ S = \perp\!\!\!\perp$$

□

Proof:

$$
\begin{array}{ll}
 & R_\succ \lhd (R \sqcup S)_\succ \\
\equiv & \qquad \{\ \text{domains}\ \} \\
 & R = R \circ (R \sqcup S)_\succ \\
\equiv & \qquad \{\ \bullet\ R_\cup \circ S = \perp\!\!\!\perp;\ R_< \circ S = \perp\!\!\!\perp\ \} \\
 & R = R_< \circ (R \sqcup S) \circ (R \sqcup S)_\succ \\
\equiv & \qquad \{\ \text{domains}\ \} \\
 & R = R_< \circ (R \sqcup S) \\
\equiv & \qquad \{\ \bullet\ R_\cup \circ S = \perp\!\!\!\perp;\ R_< \circ S = \perp\!\!\!\perp\ \} \\
 & \mathit{true}
\end{array}
$$

□

This lemma is the kernel of the proof of the distribution. Because $+$ is a relator we already know that $(R+S)_\succ \lhd R_\succ + S_\succ$. The other inclusion is proved below. The fact that the relator $+\perp\!\!\!\perp$ is equal to $\mathcal{I}^{\hookrightarrow\cup}$ combined with lemma 6.23c gives us $(R+\perp\!\!\!\perp)_\succ = R_\succ + \perp\!\!\!\perp$. The dual $(\perp\!\!\!\perp+S)_\succ = \perp\!\!\!\perp + S_\succ$ is proved dually.

$$
\begin{array}{ll}
 & (R_\succ + S_\succ) \circ (R+S)_\succ \\
= & \qquad \{\ \text{disjoint sum}\ \} \\
 & (R_\succ + \perp\!\!\!\perp \sqcup \perp\!\!\!\perp + S_\succ) \circ (R+S)_\succ \\
= & \qquad \{\ \text{distribution}\ \} \\
 & R_\succ + \perp\!\!\!\perp \circ (R+S)_\succ \sqcup \perp\!\!\!\perp + S_\succ \circ (R+S)_\succ \\
= & \qquad \{\ \text{see above}\ \} \\
 & (R+\perp\!\!\!\perp)_\succ \circ (R+S)_\succ \sqcup (\perp\!\!\!\perp + S)_\succ \circ (R+S)_\succ \\
= & \qquad \{\ \text{disjoint sum, (7.15)},\ (R+\perp\!\!\!\perp)_\cup \circ \perp\!\!\!\perp + S = \perp\!\!\!\perp,\ (\perp\!\!\!\perp + S)_\cup \circ R + \perp\!\!\!\perp = \perp\!\!\!\perp\ \}
\end{array}
$$

$$
\begin{array}{ll}
 & (R+\perp\!\!\perp){\succ} \sqcup (\perp\!\!\perp+S){\succ} \\
= & \quad \{ \text{ see above } \} \\
 & R{\succ}+\perp\!\!\perp \ \sqcup \ \perp\!\!\perp+S{\succ} \\
= & \quad \{ \text{ disjoint sum } \} \\
 & R{\succ}+S{\succ}
\end{array}
$$

□

The proof of the distribution of ${\succ}$ over $\times$ also uses an extra lemma:

Lemma 7.16:

For every binary relator $\otimes$: $(R\otimes\top\!\!\top){\succ} \sqsupseteq (R\otimes S){\succ}$

□

Proof:

$$
\begin{array}{ll}
 & (R\otimes\top\!\!\top){\succ} \sqsupseteq (R\otimes S){\succ} \\
\equiv & \quad \{ \text{ domains } \} \\
 & (R\otimes\top\!\!\top){>} \circ (R\otimes\top\!\!\top){\searrow} \sqsupseteq (R\otimes S){\succ} \\
\equiv & \quad \{ \ (R\otimes\top\!\!\top){>} \sqsupseteq (R\otimes S){>} \ \} \\
 & (R\otimes\top\!\!\top){\searrow} \sqsupseteq (R\otimes S){\succ} \\
\equiv & \quad \{ \text{ def } {\searrow}, \ \mathit{symmetric}.(R\otimes S){\succ} \ \} \\
 & R\otimes\top\!\!\top \sqsupseteq R\otimes\top\!\!\top \circ (R\otimes S){\succ} \\
\equiv & \quad \{ \text{ relator } \otimes, \text{ domains } \} \\
 & R\otimes\top\!\!\top \sqsupseteq R\otimes\top\!\!\top \circ R{\succ}\otimes S{\succ} \circ (R\otimes S){\succ} \\
\equiv & \quad \{ \text{ domains, relators } \} \\
 & R\otimes\top\!\!\top \sqsupseteq I\otimes\top\!\!\top \circ R\otimes S \circ (R\otimes S){\succ} \\
\equiv & \quad \{ \text{ domains } \} \\
 & R\otimes\top\!\!\top \sqsupseteq I\otimes\top\!\!\top \circ R\otimes S \\
\equiv & \quad \{ \text{ relators } \} \\
 & \mathit{true}
\end{array}
$$

□

We prove only one inclusion of the distribution here; the other one ($(R\times S){\succ} \sqsupseteq R{\succ}\times S{\succ}$) follows from $\times$ being a relator. We also use that $\times\top\!\!\top$ is the same as $\mathcal{I}^{\ll}$, implying that $(R\times\top\!\!\top){\succ} = R{\succ}\times\top\!\!\top$ (and of course the dual theorem for $\top\!\!\top\times$). The proof of the distribution is now very simple:

$$
\begin{array}{ll}
 & R{\succ}\times S{\succ} \\
= & \quad \{ \text{ Cartesian product } \} \\
 & R{\succ}\times\top\!\!\top \sqcap \top\!\!\top\times S{\succ} \\
= & \quad \{ \text{ see above } \} \\
 & (R\times\top\!\!\top){\succ} \sqcap (\top\!\!\top\times S){\succ} \\
\sqsupseteq & \quad \{ \ (7.16) \ \} \\
 & (R\times S){\succ}
\end{array}
$$

□

Our exploration of the algebraic properties of disjoint sum and Cartesian product is continued with an investigation of the naturality properties of the injections, projections, junc and split. The following lemma gives the results, the proofs are simple and left to the reader.

Lemma 7.17:(Naturality)

$$
\begin{array}{lrcl}
\text{(a)} & \hookrightarrow : + \dot{\leftrightsquigarrow} \ll & & \\
\text{(b)} & \hookleftarrow : + \dot{\leftrightsquigarrow} \gg & & \\
\text{(c)} & \ll : \ \ll \dot{\leftsquigarrow} \times & & \\
\text{(d)} & \gg : \ \gg \dot{\leftsquigarrow} \times & & \\
\text{(e)} & R \triangle S : H \dot{\leftsquigarrow} F \dot{+} G & \Leftarrow & R : H \dot{\leftsquigarrow} F \ \wedge \ S : H \dot{\leftsquigarrow} G \\
\text{(f)} & R \triangle S : H \dot{\rightsquigarrow} F \dot{+} G & \Leftarrow & R : H \dot{\rightsquigarrow} F \ \wedge \ S : H \dot{\rightsquigarrow} G \\
\text{(g)} & R \triangledown S : F \dot{\times} G \dot{\leftsquigarrow} H & \Leftarrow & R : F \dot{\leftsquigarrow} H \ \wedge \ S : G \dot{\leftsquigarrow} H
\end{array}
$$

□

For binary function $\otimes$ the function $F \dot{\otimes} G$ is defined by $(F \dot{\otimes} G).x = (F.x) \otimes (G.x)$. There is only one naturality property for split because compose does not distribute over cap in general. This chapter is concluded with two examples of natural transformations that can be used to express symmetry and associativity of binary specs.

The first one is a spec that swaps its argument pair, i.e. $\mathsf{swap}.(x, y) = (y, x)$, the second one is specified by $\mathsf{assoc}.(x, (y, z)) = ((x, y), z)$. Definitions in SPEC-terms are given in:

Definition 7.18:(Swap & Assoc)

$$
\begin{array}{lrcl}
\text{(a)} & \mathsf{swap} & \triangleq & \gg \triangle \ll \\
\text{(b)} & \mathsf{assoc} & \triangleq & (\ll \triangle (\ll \circ \gg)) \triangle (\gg \circ \gg)
\end{array}
$$

□

That swap and assoc satisfy the specifications above is exemplified in the following lemma:

Lemma 7.19:

For all specs R, S and T:

$$
\begin{array}{lrcl}
\text{(a)} & \mathsf{swap} \circ R \triangle S & = & S \triangle R \\
\text{(b)} & \mathsf{assoc} \circ R \triangle (S \triangle T) & = & (R \triangle S) \triangle T
\end{array}
$$

□

The proof of this lemma is straightforward, except for the distribution over $\triangle$ in the first step. The correctness of the distribution follows from lemma 7.13, third disjunct. The lemma is used in the proof of the following naturality properties:

Lemma 7.20:

For all specs R, S and T:

$$
\begin{array}{lrcl}
\text{(a)} & R \times S \circ \mathsf{swap} & = & \mathsf{swap} \circ S \times R \\
\text{(b)} & (R \times S) \times T \circ \mathsf{assoc} & = & \mathsf{assoc} \circ R \times (S \times T)
\end{array}
$$

□

The proof is an instantiation of lemma 7.19 using $S \times R = (S \circ \ll) \triangle (R \circ \gg)$ and $R \times (S \times T) = (R \circ \ll) \triangle ((S \circ \ll \circ \gg) \triangle (T \circ \gg \circ \gg))$ followed by $\times - \triangle$ fusion.

swap and assoc are used in the definitions of symmetry and associativity for specs. First we define the notion of a binary spec. This is simply a spec operating on a pair:

Definition 7.21:(Binary Spec)

A spec X is *binary* iff $X \circ I{\times}I = X$.
□

And for binary specs we define symmetry and associativity by:

Definition 7.22:(Symmetry & Associativity)

A binary spec X is *symmetric* iff $X \circ \mathsf{swap} = X$.
A binary spec X is *associative* iff $X \circ X{\times}I \circ \mathsf{assoc} = X \circ I{\times}X$.
□

A pointwise interpretation of these formulas shows that we are indeed formalising symmetry and associativity.

Chapter 8

Inductive types

The datatypes (pers) considered thus far have a very simple structure. We can construct types with zero elements ($\perp\!\!\perp$) or exactly one element ($\top\!\!\top$), we have a universal type I and we can construct Cartesian products (records) and disjoint sums (unions) of given datatypes. This is not enough for normal programming practice. We need to be able to construct recursively-defined types and operations working on those types. Examples of commonly used, recursively defined types include lists, trees and the natural numbers.

A typical example of a function defined on the natural numbers is the function $2\times$ specified by:

$$\begin{array}{lll} 2\times.0 & = & 0 \\ 2\times.(n+1) & = & 2\times.n+2 \end{array}$$

This defines $2\times$ by structural induction. Elements of the natural numbers are either constructed by the constant function 0 or by applying the function $+\,1$ to another natural number. These functions are called the *constructors* of the natural numbers.

Notice that on the rhs of the definitions we applied $2\times$ only to the argument of the constructor. This guarantees (with a suitable induction principle) that the function is uniquely defined. Had we defined

$$2\times.(n+1) \quad = \quad 2\times.(n+2)-2$$

(this is also valid for multiplying by 2) then the function $f.n$ which is 0 for $n = 0$ and $2n+2$ otherwise would also have been a solution.

An induction principle is also important. Consider the type of the natural numbers extended with two new numbers ∞ and $-\infty$ satisfying $\infty+1 = \infty$ and $-\infty+1 = -\infty$. Then the function defined as $2\times$ on the standard natural numbers and $2\times.\infty = \infty$ satisfies the specification, but it is not the only solution: $2\times.\infty = -\infty$ also gives a solution. There is now more than one solution because $2\times.\infty$ can not be calculated by a terminating recursion. This is the consequence of an induction principle: the fact that every natural number is constructed by a finite number of applications of the constructors guarantees that the recursion in the definition of $2\times$ will terminate. A precise definition of this induction principle will be given in another section.

The combination of specifying a function on all constructors of a type, applying the function only to the arguments of the constructor on the rhs and an induction principle gives us the unicity of a solution to the specification. The existence of a solution is a separate issue that will be discussed in another section. For the moment it is sufficient to know that totality and injectivity of the constructors guarantees the existence of a solution. The constructors of the natural numbers are total and injective.

The natural number type is the simplest non-trivial example of an inductive datatype, but it illustrates all important characteristics: constructor(s), and definition of functions on the type by structural induction. The next step is to investigate how constructors and structural induction work for general inductive datatypes. A natural approach is to view a datatype as a set and a constructor as a function with as range (a subset of) the datatype. The domain of the constructor is a set that may depend on the datatype again. One way of doing this is by fixing the domain to be the result of the application of an endofunctor of **Set** to the datatype. So in general constructors τ of datatype A satisfy:

$$\tau \in A \leftarrow F.A$$

where F is a functor from **Set** . This also works for constants of the type, because we can take $F \;=\; \mathbb{1}^\bullet$, where $\mathbb{1}$ is a set containing a single anonymous element and use the isomorphy between elements and functions from a one-element domain. For the natural numbers we have :

$$\begin{array}{l} 0 \in \mathbb{N} \;\leftarrow\; \mathbb{1}^\bullet.\mathbb{N} \\ +1 \in \mathbb{N} \leftarrow \mathcal{I}.\mathbb{N} \end{array}$$

Using the functor approach we can also give a general form for the specifying equations of a function defined by structural induction. If constructor $\tau : A \leftarrow F.A$ and we want to specify $X : B \leftarrow A$ then the equation has as form:

$$X \circ \tau \;=\; f \circ F.X$$

where $f : B \leftarrow F.B$. In the $2\times$ example we can rewrite the specification to:

$$\begin{array}{lcl} 2\times \circ 0 & = & 0 \;\circ\; \mathbb{1}^\bullet.2\times \\ 2\times \circ +1 & = & (+\,2) \circ \mathcal{I}.2\times \end{array}$$

This shape guarantees that the recursion is only performed on the argument of the constructor, but it is quite restrictive. For example the equation for the factorial function $fac.(n+1) \;=\; (n+1) \;\times\; fac.n$ cannot be transformed to this shape, because the n can only be used in the recursive call and not afterwards. Later on we will generalise to less restrictive forms of structural induction, but for the moment we restrict our attention to this form of equation.

Sets of equations can be transformed to a single equation by combining them with $\triangledown$. Using the distribution of $\circ$ over $\triangledown$ and $\triangledown$ - $+$ fusion the set of equations becomes a single equation of the correct shape. In the $2\times$ example we get:

$$2\times \;\circ\; (0\triangledown{+1}) \;=\; 0\triangledown(+2) \;\circ\; \mathbb{1}^\bullet.2\times + \mathcal{I}.2\times$$

We can view $0\triangledown{+}\mathit{1} \in \mathbb{N} \leftarrow \mathbb{1}^{\bullet}.\mathbb{N} + \mathcal{I}.\mathbb{N}$ as a single constructor for the natural numbers, and this constructor is surjective. The original constructors can be retrieved from this single constructor by composition with the injections.

8.1 F-algebras

In this section we will show a categorical approach to the definition of inductive datatypes, based on the approach sketched in the previous section.

A standard method for the construction of an inductive type using category theory [32, 42, 43] is defining it via the category of F-algebras (in fact defining it to be the initial object). This is a categorical statement of the initial algebra approach that can be found, for example, in the work of Goguen et al [29, 30].

Given a category $\mathcal{C}$ with an endofunctor F we define the category of F-algebras over $\mathcal{C}$ by :

Definition 8.1: (F-algebra)

The category of F-algebras over $\mathcal{C}$ is defined by:

- Objects: arrows $\tau \in \mathit{cod}.\tau \leftarrow F.(\mathit{cod}.\tau)$ from $\mathcal{C}$ (F-algebras).
- Arrows: triples $(f, \sigma, \tau) \in \sigma \leftarrow \tau$ where $f \in \mathit{cod}.\sigma \leftarrow \mathit{cod}.\tau$ is an arrow from $\mathcal{C}$ and σ and τ are F-algebras such that $f \circ \tau \ = \ \sigma \circ F.f$ in $\mathcal{C}$.
- Domains: $\mathit{dom}.(f, \sigma, \tau) \ = \ \tau$ and $\mathit{cod}.(f, \sigma, \tau) \ = \ \sigma$.
- Composition: $(f, \sigma, \tau) \circ (g, \tau, \rho) \ = \ (f \circ g, \sigma, \rho)$.
- Identity: $\mathit{id}.\tau \ = \ (\mathit{id}.(\mathit{cod}.\tau), \tau, \tau)$ where id in the rhs is taken from $\mathcal{C}$.

□

The objects of this category are called *F-algebras* and the codomain of an F-algebra is called the *carrier* of that algebra. The arrows are called *F-homomorphisms.*

Let's consider as an example the construction of the natural numbers in **Set**. The functor corresponding to the constructors is $F.X \ \triangleq \ \mathbb{1}{+}X$. A function f on the natural numbers with as range A is defined by structural induction by supplying a value for $f.0$ ($g \in A \leftarrow \mathbb{1}$, $f.0 \ = \ g.x$ where x is the unique element in $\mathbb{1}$) and method for calculating the value of $f.(n+1)$ from $f.n$ ($h \in A \leftarrow A$, $f.(n+1) \ = \ h.(f.n)$). The function f is completely determined by g and h. Both the junc of the constructors of the natural numbers ($0^{\bullet} \triangledown (+\,1)$) and $g \triangledown h$ are F-algebras and f can be defined as the first component of an arrow in $g \triangledown h \ \leftarrow \ 0^{\bullet} \triangledown (+\,1)$. Expanding the definition of an arrow in the category of F-algebras and using some elementary properties of disjoint sum we see that f is the unique function satisfying

$$f \circ 0^{\bullet} = g \quad \wedge \quad f \circ (+1) = h \circ f$$

This is a variable-free version of the definition of f by structural induction. The F-algebra $0^{\bullet} \triangledown (+\,1)$ is an initial object because a unique function f exists for every F-algebra $g \triangledown h$.

Our main interest is in the initial object of the category of F-algebras and not the properties of the category itself. This is the reason for introducing the following definition:

Definition 8.2: (Initial F-algebra)

For endofunctor F, arrow $\tau \;\in\; B \longleftarrow F.B$ is an *initial F-algebra* iff τ is an initial object in the category of F-algebras.
□

The constructor of the datatype corresponding to functor F can be defined as an initial F-algebra. An initial F-algebra in **Set** can be interpreted as the constructor function of the free type corresponding to the functor F. The natural numbers are the free type for functor $F.X \triangleq 1\!\!1 + X$. Some other examples of free types in **Set** are cons-lists with values from set A (functor $F.X \triangleq 1\!\!1 + A \times X$) or binary trees with values from A in the leaves (functor $F.X \triangleq A + X \times X$).

A property of an initial F-algebra that is easy to prove is that it has an inverse. For an initial F-algebra $\tau \in B \longleftarrow F.B$, consider the unique arrow $(\gamma, F.\tau, \tau) \;\in\; F.\tau \longleftarrow \tau$ ($F.\tau$ is also an F-algebra). Now $\tau \circ \gamma \;=\; id.B$ and $\gamma \circ \tau \;=\; id.(F.B)$. The former is proved by showing that $(\tau \circ \gamma, \tau, \tau) \in \tau \longleftarrow \tau$ and is therefore equal to $id.\tau = (id.B, \tau, \tau)$, the latter follows from the fact that $(\gamma, F.\tau, \tau)$ is an arrow so $\gamma \circ \tau \;=\; F.\tau \circ F.\gamma \;=\; F.(id.B)$. In **Set** this means that an initial F-algebra is not only functional and total, but also surjective and injective.

A problem with the initial-object approach is the question of the existence of initial objects in the category. In **Set** the existence is not guaranteed for every endofunctor, but for the functors that we usually consider (the polynomial functors) there is always an initial object. The situation is simpler in **Difun**; we now prove that for every endorelator F there exists an initial F-algebra in **Difun**:

Lemma 8.3: (Initial F-algebra)

For endorelator F, $(\mu F, \mu F, F.\mu F)$ is an initial F-algebra in **Difun**.
□

We wrote μF here but didn't specify which lattice is meant. This is not a problem because the least fixpoint in the SPEC-lattice and the least fixpoint in the per-lattice coincide. Writing μF for the least fixpoint in the SPEC-lattice we calculate:

$$\begin{array}{ll}
 & \mu F \circ A \;=\; \mu F \\
\Leftarrow & \qquad \{\ \mu\text{-fusion}\ \} \\
 & \forall (X \;\vdots\vdots\; F.X \circ A \;=\; F.(X \circ A)) \\
\Leftarrow & \qquad \{\ \text{relators}\ \} \\
 & A \;=\; F.A
\end{array}$$

Instantiating this calculation with $\mu F^{\cup}$ for A gives us the proof that μF is a per. If A is a per that is a fixpoint of F then we obtain $\mu F \lhd A$, i.e. μF is also the least fixpoint in the per-lattice.

The next step in the proof of lemma (8.3) is showing that $(\mu F, \mu F, F.\mu F)$ is an F-algebra in **Difun**. This is equivalent, using $\mu F = F.\mu F$, to proving $\mu F : \mu F \leftarrow\!\!\dashv \mu F$. Expanding the definition of $\leftarrow\!\!\dashv$ has as result that the proof obligation reduces to showing that μF is a per, and that has already been established.

Our next task is to establish the existence of a unique arrow from μF to arbitrary F-algebra σ in the category of F-algebras over **Difun**. We prove that $((\![F;\sigma]\!), \sigma, \mu F)$ is the unique arrow, where $(\![F;\sigma]\!)$ is defined by:

Definition 8.4: (Catamorphism)

For relator F and spec R the F-catamorphism of R is defined by:

$$(\![F;R]\!) \quad \triangleq \quad \mu(X \mapsto R \circ F.X)$$

□

We usually omit the relator "F" in $(\![F;\sigma]\!)$ if it is clear from the context which relator is meant.

There are two proof obligations: we have to prove that $(\![\sigma]\!)$ is an arrow in **Difun** with the correct type and we have to prove that $(\![\sigma]\!)$ is the only solution of

$$X{::} \qquad X \circ \mu F \;=\; \sigma \circ F.X$$

i.e. there is only one arrow between μF and σ. The first obligation is fulfilled by proving the following lemma about the typing rules for the catamorphism:

Lemma 8.5: (Type catamorphism)

$$\begin{array}{lrcl}
\text{(a)} & R : A \,— & \Rightarrow & (\![R]\!) : A \,— \,\mu F \\
\text{(b)} & R : A \leftarrow F.A & \Rightarrow & (\![R]\!) : A \leftarrow \mu F \\
\text{(c)} & R : A \rightarrow F.A & \Rightarrow & (\![R]\!) : A \rightarrow \mu F \\
\text{(d)} & R : A \longrightarrow\!\!\!\!\dashv F.A & \Rightarrow & (\![R]\!) : A \longrightarrow\!\!\!\!\dashv \mu F
\end{array}$$

□

The proof of (8.5a) is trivial for the left-domain part (only the calculation rule for fixpoints is used). The right-domain is proved by:

$$\begin{array}{ll}
 & (\![R]\!) \circ \mu F \;=\; (\![R]\!) \\
\Leftarrow & \qquad \{\ \mu\text{-fusion, definition catamorphism}\ \} \\
 & \forall(X :: R \circ F.X \circ \mu F \;=\; R \circ F.(X \circ \mu F)) \\
\equiv & \qquad \{\ \text{relators, fixpoints}\ \} \\
 & \mathit{true}
\end{array}$$

For (8.5b) we need a lemma about how the composition of a catamorphism and a reversed catamorphism can be written as a single least fixed point. Such a combination is called a hylomorphism and later we will see that large classes of recursive programs can be written as hylomorphisms.

Definition 8.6: (Hylomorphism)

For relator F and specs R and S the F-hylomorphism of R and S is defined as:

$$[\![F; R, S]\!] \quad \triangleq \quad \mu(X \mapsto R \circ F.X \circ S)$$

□

Again we usually omit the relator if it is clear from the context. The lemma about the composition of a catamorphism and a reversed catamorphism is the following:

Lemma 8.7: (Hylomorphism)

$$[\![R, S]\!] \quad = \quad (\![R]\!) \circ (\![S^{\cup}]\!)^{\cup}$$

□

Proof:

$$\begin{array}{ll}
 & [\![R, S]\!] \quad = \quad (\![R]\!) \circ (\![S^{\cup}]\!)^{\cup} \\
\Leftarrow & \qquad \{ \ \mu\text{-fusion, definitions hylomorphism and catamorphism } \} \\
 & \forall(X :: R \circ F.(X \circ (\![S^{\cup}]\!)^{\cup}) \circ S \quad = \quad R \circ F.X \circ (\![S^{\cup}]\!)^{\cup}) \\
\equiv & \qquad \{ \ \text{relators, calculation rule, definition catamorphism, reverse } \} \\
 & \mathit{true}
\end{array}$$

The proof of the preservation of functionality as is stated in (8.5b) is simple now:

$$\begin{array}{ll}
 & A \quad \sqsupseteq \quad (\![R]\!) \circ (\![R]\!)^{\cup} \\
\equiv & \qquad \{ \ (8.7) \ \} \\
 & A \quad \sqsupseteq \quad [\![R, R^{\cup}]\!] \\
\Leftarrow & \qquad \{ \ \text{induction, definition hylo } \} \\
 & A \quad \sqsupseteq \quad R \circ F.A \circ R^{\cup} \\
\equiv & \qquad \{ \ \bullet \ R : A \leftarrow F.A \ \} \\
 & \mathit{true}
\end{array}$$

Parts (8.5c) and (8.5d) are proved with a single proof. The proof of (8.5d) is the same as that of (8.5c), only the inclusion is in the opposite direction.

$$\begin{array}{ll}
 & \mu F \quad \sqsupseteq \quad (\![R]\!)^{\cup} \circ (\![R]\!) \\
\Leftarrow & \qquad \{ \ \mu\text{-fusion, definition cata } \} \\
 & \forall(X :: F.((\![R]\!)^{\cup} \circ X) \quad \sqsupseteq \quad (\![R]\!)^{\cup} \circ R \circ F.X) \\
\Leftarrow & \qquad \{ \ \text{relators, monotonicity } \} \\
 & F \,.\, (\![R]\!)^{\cup} \quad \sqsupseteq \quad (\![R]\!)^{\cup} \circ R \\
\equiv & \qquad \{ \ \text{assumption, 8.5a, relators, calculation rule } \} \\
 & F \,.\, (\![R]\!)^{\cup} \circ F.A \quad \sqsupseteq \quad F \,.\, (\![R]\!)^{\cup} \circ R^{\cup} \circ R \\
\Leftarrow & \qquad \{ \ \text{monotonicity } \} \\
 & F.A \quad \sqsupseteq \quad R^{\cup} \circ R \\
\equiv & \qquad \{ \ \text{assumption } \} \\
 & \mathit{true}
\end{array}$$

The assumption used in the last step is $R\colon \longrightarrow F.A$ for the proof of (8.5c) and would be $R\colon \longrightarrow\!\!\!\!\dashv F.A$ for the proof of (8.5d). So if $\sigma : A \longleftarrow\!\!\!\!\dashv F.A$ then by (8.5b) and (8.5d) we have $(\!|\sigma|\!) : A \longleftarrow\!\!\!\!\dashv \mu F$ and this is the desired type. The fact that $(\!|\sigma|\!)$ is a solution to the equation $X \circ \mu F \;=\; \sigma \circ F.X$ is trivial because μF can be absorbed and the equality is established by the calculation rule. The uniqueness of the arrow is the last part of the proof of lemma (8.3). We show that $(\!|\sigma|\!)$ is the only $X : A \longleftarrow\!\!\!\!\dashv \mu F$ such that $X \circ \mu F \;=\; \sigma \circ F.X$ by instantiating theorem (8.8) with R equal to σ, thereby completing the proof of lemma (8.3).

The unique extension property can be seen as the SPEC-version of stating the initiality for an initial F-algebra:

Theorem 8.8: (Unique Extension Property)

$$X \;=\; (\!|R|\!) \qquad \equiv \qquad X \circ \mu F \;=\; X \;\;\wedge\;\; X \;=\; R \circ F.X$$

□

The proof for the implication in the $\Rightarrow$ direction is an application of lemma (8.5) and the calculation rule for fixpoints. The implication in the $\Leftarrow$ direction is proved by:

$$\begin{array}{ll}
 & X \;=\; (\!|R|\!) \\
\equiv & \qquad \{\ \bullet\ X = X \circ \mu F\ \} \\
 & X \circ \mu F \;=\; (\!|R|\!) \\
\Leftarrow & \qquad \{\ \mu\text{-fusion, definition cata}\ \} \\
 & \forall (Y \;\vdots\vdots\; X \circ F.Y \;=\; R \circ F.(X \circ Y)) \\
\Leftarrow & \qquad \{\ \text{relators, monotonicity}\ \} \\
 & X \;=\; R \circ F.X
\end{array}$$

□

The rhs of the unique extension theorem can be combined to a single equation:

$$\begin{array}{ll}
 & X \circ \mu F \;=\; X \;\;\wedge\;\; X \;=\; R \circ F.X \\
\equiv & \qquad \{\ \text{substitution}\ \} \\
 & X \circ \mu F \;=\; X \;\;\wedge\;\; X \;=\; R \circ F.(X \circ \mu F) \\
\equiv & \qquad \{\ \text{relators, fixpoints}\ \} \\
 & X \circ \mu F \;=\; X \;\;\wedge\;\; X \;=\; R \circ F.X \circ \mu F \\
\equiv & \qquad \{\ \text{substitution}\ \} \\
 & R \circ F.X \circ \mu F \circ \mu F \;=\; R \circ F.X \circ \mu F \;\;\wedge\;\; X \;=\; R \circ F.X \circ \mu F \\
\equiv & \qquad \{\ \text{per}\ \mu F\ \} \\
 & X \;=\; R \circ F.X \circ \mu F
\end{array}$$

So, phrased as a lemma, we obtain:

Lemma 8.9: (Unique Extension Property)

$$X \;=\; (\!|R|\!) \qquad \equiv \qquad X \;=\; R \circ F.X \circ \mu F$$

□

8.2 F-inductive algebras

Working with initial F-algebras gives quite a powerful theory of inductive datatypes, see for example [9] for a demonstration of the development of algorithms using relational catamorphisms, but many real-life inductive datatypes don't fit this framework. The problem is that all constructors constructed from initial F-algebras over **Set** or **Difun** are total and injective. For many practical datatypes this is not the case. The restrictiveness of injectivity is exemplified by join-lists: we have that $[1,2]{+\!\!+}[3] = [1]{+\!\!+}[2,3]$, so the $+\!\!+$ constructor is not injective. The restrictiveness of totality is exemplified by height-balanced trees: we cannot combine a tree of height 4 with a tree of height 7, so the constructor is not total.

Switching to the category of F-algebras over a category of partial functions or partial difunctionals does not solve the totality problem because initial objects in general don't exist for the functors that we are interested in. The initiality gave us two properties: the existence and the uniqueness of solutions for specifications. The important one here is the uniqueness: the specifying equations can have at most one solution. We keep the functionality and surjectivity requirements for the constructor of the datatype and investigate the conditions under which the specification has at most one solution. The surjectivity and functionality requirements are sometimes called "no junk" and "no confusion" (see Meseguer and Goguen [46]).

We saw in the previous section that the proof of the unique extension property of the catamorphism operator does not depend on properties of the argument of the catamorphism. This gives us hope that we don't need to introduce extra properties for the more general form of inductive datatype introduced in this section. The specification of our problem is now as follows: given a functional and surjective spec $\tau : A \twoheadleftarrow F.A$, under what condition does the pair of equations:

$$
\begin{array}{lll}
(8.10)\ X \circ A & = & X \\
(8.11)\ X \circ \tau & = & R \circ F.X
\end{array}
$$

have at most one solution. Equation (8.10) is necessary for a unique solution because if X solves (8.11) then $X \circ A$ also solves it. We only want to impose conditions on τ, not on R.

Applying $\circ\, \tau^{\cup}$ on both sides of (8.11) gives, in combination with (8.10), that X solves the equation for $[\![R, \tau^{\cup}]\!]$ (X:: $X \;=\; R \circ F.X \circ \tau^{\cup}$), so this gives us a candidate for the unique solution. μ-Fusion is the standard technique for proving that something is equal to a least fixpoint and (8.11) provides the key for this application:

$$
\begin{array}{ll}
 & X \circ [\![\tau, \tau^{\cup}]\!] \;=\; [\![R, \tau^{\cup}]\!] \\
\Leftarrow & \quad \{\ \mu\text{-fusion, definition hylo}\ \} \\
 & \forall (Y \;\vdots\vdots\; X \circ \tau \circ F.Y \circ \tau^{\cup} \;=\; R \circ F.(X \circ Y) \circ \tau^{\cup}) \\
\equiv & \quad \{\ \text{relators, (8.11)}\ \} \\
 & \mathit{true}
\end{array}
$$

From this we conclude that $[\![\tau, \tau^{\cup}]\!] \;=\; A$ is a sufficient condition for having at most one solution for the combination of (8.10) and (8.11). Note that the functionality or surjectivity of τ play no role in this proof, it being only used for guessing a candidate for the unique solution. The condition $[\![\tau, \tau^\cup]\!] \;=\; A$ turns out to be so important that it deserves a special name:

Definition 8.12:(*F*-inductive algebra)

For endorelator F, spec τ is an *F-inductive algebra* with carrier A iff $\tau : A \;—\; F.A$ and

$$[\![F; \tau, \tau^\cup]\!] \quad = \quad A$$

□

This definition can be interpreted, assuming continuity properties, as stating that every element of A can be constructed using a finite number of applications of the constructor τ. Using 0 applications one can only construct the elements of the empty type $\perp\!\!\!\perp$, with 1 application the elements of type $\tau \circ F.\perp\!\!\!\perp \circ \tau^\cup$, with 2 applications the elements of $\tau \circ F.(\tau \circ F.\perp\!\!\!\perp \circ \tau^\cup) \circ \tau^\cup$ etc. The limit of this process is $[\![F; \tau, \tau^\cup]\!]$.

Summing up the preceding calculations results in the following theorem:

Theorem 8.13:(Unique Extension Property)

For F-inductive algebra $\tau : A \;—\; F.A$:

$$X \circ A \;=\; X \quad \wedge \quad X \circ \tau \;=\; R \circ F.X \qquad \Rightarrow \qquad X \;=\; [\![R, \tau^\cup]\!]$$

□

Our aim was to construct an inductive datatype with a constructor, but definition (8.12) does not mention functionality or surjectivity of τ. This is not a problem because functionality and surjectivity follow immediately from the definition:

Lemma 8.14:(*F*-inductive algebra)

For F-inductive algebra $\tau : A \;—\; F.A$ we have $\tau : A \twoheadleftarrow$

□

Proof:

$$\begin{array}{ll}
 & \tau \circ \tau^\cup \\
= & \qquad \{ \;\bullet\; \tau : A \;—\; F.A \;\} \\
 & \tau \circ F.A \circ \tau^\cup \\
= & \qquad \{ \;\bullet\; [\![\tau, \tau^\cup]\!] \;=\; A \;\} \\
 & \tau \circ F.[\![\tau, \tau^\cup]\!] \circ \tau^\cup \\
= & \qquad \{ \text{ calculation rule, definition hylo } \} \\
 & [\![\tau, \tau^\cup]\!] \\
= & \qquad \{ \;\bullet\; [\![\tau, \tau^\cup]\!] \;=\; A \;\} \\
 & A
\end{array}$$

8.2.1 Constructors

Using the techniques described in this chapter we can build algebras for many common datatypes, like the natural numbers: $\mu(X \mapsto \top\!\!\!\top + X)$ ($\top\!\!\!\top$ corresponds to a 1-element set), and cons-lists over a type A: $\mu(X \;\mapsto\; \top\!\!\!\top \,+\, A \times X)$ (singleton and cons-constructor). The separate constructors for these datatypes can easily be extracted from the least fixpoint; if F is a sum relator then we can write μF as the junc of two constructors. We define a sum relator by:

Definition 8.15:(Sum Relator)

Relator F is a sum relator iff $F.I \lhd I{+}I$
□

We give the extraction of the constructors for all F-inductive algebras, not just μF. We assume $\tau : A$ — $F.A$ is an F-inductive algebra and F is a sum relator.

$$
\begin{array}{ll}
 & \tau \\
= & \{ \ \bullet\, \tau : \text{—}\ F.A,\ F \text{ sum relator} \Rightarrow \quad F.A \lhd F.I \lhd I{+}I \ \} \\
 & \tau \circ I{+}I \\
= & \{ \ \bullet \text{ disjoint sum } \} \\
 & \tau \ \circ \ \hookrightarrow \triangledown \hookleftarrow \\
= & \{ \ \text{disjoint sum } \} \\
 & (\tau\circ \hookrightarrow) \triangledown (\tau\circ \hookleftarrow)
\end{array}
$$

A small example using constructors: the length function on cons-lists. For $F.X \ = \ \top\!\!\top{+}X$ and $G.X \ = \ \top\!\!\top + A{\times}X$ we can construct μF with constructors $0^\bullet \ \triangleq \ \mu F\circ \hookrightarrow$ and $+1 \ \triangleq \ \mu F\circ \hookleftarrow$ (the constructors of the natural numbers) and μG with constructors $[]^\bullet \ \triangleq \ \mu G\circ \hookrightarrow$ and $\rightarrowtail \ \triangleq \ \mu G\circ \hookleftarrow$ (the constructors of the cons-lists). The set of equations:

$$
\begin{array}{lll}
\mathit{length} \circ []^\bullet & = & 0^\bullet \\
\mathit{length} \circ \rightarrowtail & = & +1 \circ \gg \circ A{\times}\mathit{length}
\end{array}
$$

has a unique solution: $\mathit{length} \ = \ (\!|G; 0^\bullet \triangledown (+1 \circ \gg)|\!)$.

Typically a sum relator F is of the form $F.X \ = \ G.X + H.X$, for some relators G and H. For such relators we can also give type judgements for the constructors:

$$
\begin{array}{l}
\tau\circ \hookrightarrow \ : A \leftarrow G.A \\
\tau\circ \hookleftarrow \ : A \leftarrow H.A
\end{array}
$$

We prove this claim for $\tau\circ \hookrightarrow$. The proof for the other constructor is dual.

$$
\begin{array}{ll}
 & A \\
= & \{ \ \tau : A \text{ — } F.A,\ (8.14) \ \} \\
 & \tau\circ\tau\cup \\
= & \{ \ \bullet\, \tau : \text{—}\ F.A,\ F \text{ sum relator} \Rightarrow \quad F.A \lhd F.I \lhd I{+}I \ \} \\
 & \tau \circ I{+}I \circ \tau\cup \\
\sqsupseteq & \{ \ \text{disjoint sum } \} \\
 & \tau\circ \hookrightarrow \ \circ (\tau\circ \hookrightarrow)\cup
\end{array}
$$

$$
\begin{array}{ll}
 & \tau\circ \hookrightarrow \ \circ G.A \\
= & \{ \ \text{disjoint sum } \} \\
 & \tau \ \circ \ G.A + H.A \ \circ \ \hookrightarrow
\end{array}
$$

$=$ $\{\ \bullet\ \tau : \text{—}\ F.A\ \}$
$\tau\circ \hookrightarrow$
$=$ $\{\ \bullet\ \tau : A\ \text{—}\ \}$
$A \circ \tau\circ \hookrightarrow$

8.2.2 Simulations

The notion of simulation of a datatype by another datatype plays an important role in programming. A simple example of simulation is the datatype of booleans, constructed with constructors *false*$^\bullet$ and *not*. This is an F-inductive algebra with the same relator as for the algebra of the natural numbers. We can simulate the booleans by the natural numbers by mapping every even number to false and every odd number to true, the constant 0 constructor simulates the constant false constructor and the successor constructor simulates the not constructor. The simulation mapping is functional (every number corresponds to at most one boolean) and surjective (every boolean is simulated). Formally we define simulation of F-algebras as follows:

Definition 8.16: (Simulation)

For F-inductive algebras $\tau : A\ \text{—}\ F.A$ and $\sigma : B\ \text{—}\ F.B$ we say that $\tau \gtrsim \sigma$ (pronunciation: τ *simulates* σ) iff there exists an $\alpha : B\ \text{—}\ A$ such that $\alpha \circ \tau\ =\ \sigma \circ F.\alpha$.
□

The equation $\alpha \circ \tau\ =\ \sigma \circ F.\alpha$ guarantees that elements of B can be built in the same way as corresponding elements of A, i.e. if an element of A in the right domain of α can be constructed with a certain combination of constructors then the corresponding construction in σ constructs the α-image of that element. For a simulation as defined above, type B is called the *abstract* type, type A is called the *concrete* type and function α is called the *abstraction function* or *simulation* . Note that the F-inductivity of the algebras guarantees that there is at most one simulation between two algebras:
$\alpha\ =\ [\![\sigma, \tau\cup]\!]$.
The fact that simulations are functional and surjective is expressed in the following lemma:

Lemma 8.17: (Simulation)

For F-inductive algebras $\tau : A\ \text{—}\ F.A$ and $\sigma : B\ \text{—}\ F.B$ such that $\tau \gtrsim \sigma$ with simulation α ($=\ [\![\sigma, \tau\cup]\!]$) we have $\alpha : B \leftarrowtail$
□

Proof:

$\alpha \circ \alpha\cup\ =\ B$
$=$ $\{\ \alpha\ =\ [\![\sigma, \tau\cup]\!]$, σ is an F-inductive algebra $\}$
$[\![\sigma, \tau\cup]\!] \circ \alpha\cup\ =\ [\![\sigma, \sigma\cup]\!]$
$\Leftarrow$ $\{\ \mu$-fusion, definition hylo $\}$
$\forall(Y\ ::\ \sigma \circ F.Y \circ \tau\cup \circ \alpha\cup\ =\ \sigma \circ F.(Y \circ \alpha\cup) \circ \sigma\cup)$
$\Leftarrow$ $\{$ relators $\}$

$$
\begin{array}{ll}
 & \tau^{\cup} \circ \alpha^{\cup} \;=\; F.\alpha^{\cup} \circ \sigma^{\cup} \\
\equiv & \quad \{ \text{ reverse, } \alpha \circ \tau \;=\; \sigma \circ F.\alpha \;\} \\
 & \mathit{true}
\end{array}
$$

□

The $\gtrsim$ relation between algebras is transitive and reflexive, so it is a preorder. The proof of this claim is fairly trivial. For transitivity: if $\tau \gtrsim \sigma$ with simulation α and $\sigma \gtrsim \rho$ with simulation β then $\tau \gtrsim \rho$ with simulation $\beta \circ \alpha$. For reflexivity: $\tau : A \longrightarrow F.A$ simulates itself with simulation A. The $\gtrsim$ relation is not a partial order because it is not anti-symmetric. Consider for example the F-inductive algebra $[]^{\bullet} \triangledown (\rightarrowtail \circ \gg^{\cup})$ for relator $F.X = \top\!\!\top + X$ with as carrier the cons-lists over $\top\!\!\top$. The carrier has as elements lists over a one-element set, so there is a unique list for each length. The algebra simulates the algebra of the natural numbers $0^{\bullet} \triangledown (+\,1)$ by representing number n with the list of length n. The natural numbers simulate the algebra by representing a list by its length. The algebras $[]^{\bullet} \triangledown (\rightarrowtail \circ \gg^{\cup})$ and $0^{\bullet} \triangledown (+\,1)$ simulate each other but are not equal.

Because the $\gtrsim$ relation is a preorder we can construct a category of F-inductive algebras based on the $\gtrsim$ relation:

Definition 8.18:(Category of F-inductive algebras)

The category of F-inductive algebras is defined by:

- Objects: F-inductive algebras
- Arrows: $(\sigma, \tau) \in \sigma \longleftarrow \tau$ iff $\tau \gtrsim \sigma$
- Domains: $\mathit{dom}.(\sigma, \tau) \;=\; \tau$ and $\mathit{cod}.(\sigma, \tau) \;=\; \sigma$
- Composition: $(\sigma, \tau) \circ (\tau, \rho) \;=\; (\sigma, \rho)$
- Identity: $\mathit{id}.\tau \;=\; (\tau, \tau)$

□

This is the standard construction for making a category from a preorder. We could have added the simulation to the arrow, but this is not necessary because there is at most one simulation between two given F-inductive algebras. The initial objects of the category of F-inductive algebras are the initial F-algebras from the category **Difun** and there is a unique terminal object $\perp\!\!\perp : \perp\!\!\perp \longrightarrow F.\perp\!\!\perp$. The proof of this claim is easy and left as an exercise to the reader. Two F-inductive algebras that simulate each other are automatically isomorphic objects in this category because there is at most one arrow between any two objects. Meseguer and Goguen [46] define the notion of an *abstract data type* as an isomorphism class of algebras.

8.2.3 F-inductive types

The simulation preorder can be used to make a partial order by selecting one element of every class of mutually simulating algebras and restricting the pre-order to those elements. Something similar was done with the initial F-algebras over **Difun**, where there can be many initial algebras, all isomorphic, but a special one, μF, was selected because it had nice properties. We start by giving a definition for isomorphy of F-inductive algebras:

Definition 8.19:(Isomorphic F-inductive algebras)

F-inductive algebras $\tau : A \longleftarrow F.A$ and $\sigma : B \longleftarrow F.B$ are isomorphic (notation $\tau \simeq \sigma$) iff there exists an $\alpha : B \leftarrowtail\!\!\twoheadrightarrow A$ such that $\alpha \circ \tau \ = \ \sigma \circ F.\alpha$

□

The α in (8.19) is called the *isomorphism*. An alternative definition for $\tau \simeq \sigma$ would have been that τ and σ simulate each other, but the definition given in (8.19) is more convenient for further manipulation. We will prove the equivalence of the definition in (8.19) with

$$\tau \simeq \sigma \quad \triangleq \quad \tau \gtrsim \sigma \quad \wedge \quad \sigma \gtrsim \tau$$

now. It is clear from the definition of simulation that (8.19) implies $\tau \gtrsim \sigma$ with simulation α. The fact that $\sigma \gtrsim \tau$ with simulation $\alpha^{\cup}$ follows from:

$$
\begin{array}{ll}
 & \alpha^{\cup} \circ \sigma \\
= & \qquad \{ \ \bullet \ \sigma : \longleftarrow F.B, \ \alpha : B \leftarrowtail\!\longleftarrow \ \} \\
 & \alpha^{\cup} \ \circ \ \sigma \ \circ \ F.\alpha \ \circ \ F.\alpha^{\cup} \\
= & \qquad \{ \ \bullet \ \alpha \circ \tau \ = \ \sigma \circ F.\alpha \ \} \\
 & \alpha^{\cup} \ \circ \ \alpha \ \circ \ \tau \ \circ \ F.\alpha^{\cup} \\
= & \qquad \{ \ \bullet \ \tau : A \longleftarrow, \ \alpha : \longrightarrow\!\!\!\!\rightarrowtail A \ \} \\
 & \tau \ \circ \ F.\alpha^{\cup}
\end{array}
$$

The proof that mutual simulation implies (8.19) is also not difficult; If $\tau \gtrsim \sigma$ then $\alpha = [\![\sigma, \tau^{\cup}]\!]$ satisfies $\alpha : B \leftarrowtail\!\longleftarrow A$ and $\alpha \circ \tau \ = \ \sigma \circ F.\alpha$, so the only thing that remains to be proved is $\alpha : B \longrightarrow\!\!\!\!\rightarrowtail A$. From $\sigma \gtrsim \tau$ follows that $[\![\tau, \sigma^{\cup}]\!]$ satisfies $[\![\tau, \sigma^{\cup}]\!] : A \leftarrowtail\!\longleftarrow B$, which is equivalent to $[\![\tau, \sigma^{\cup}]\!]^{\cup} : B \longrightarrow\!\!\!\!\rightarrowtail A$. The proof is completed by proving that $[\![\tau, \sigma^{\cup}]\!]^{\cup} \ = \ [\![\sigma, \tau^{\cup}]\!] \ = \ \alpha$:

$$
\begin{array}{ll}
 & [\![\tau, \sigma^{\cup}]\!]^{\cup} \ = \ [\![\sigma, \tau^{\cup}]\!] \\
\Leftarrow & \qquad \{ \ \mu\text{-fusion, definition hylo} \ \} \\
 & \forall (Y \ \vdots\vdots \ (\tau \circ F.Y \circ \sigma^{\cup})^{\cup} \ = \ \sigma \ \circ \ F.Y^{\cup} \ \circ \ \tau^{\cup}) \\
\equiv & \qquad \{ \ \text{reverse, relators} \ \} \\
 & \mathit{true}
\end{array}
$$

□

The initial F-algebra μF has the important property that the constructor is the same as the carrier and it would be nice if we can find an algebra with carrier identical to constructor in every class of isomorphic F-inductive algebras. The definition of F-inductive algebra can be simplified under the assumption that the carrier is equal to the constructor.

Lemma 8.20: **(F-inductive algebra)**

Per A is an F-inductive algebra with itself as carrier iff $A \lhd F.A \quad \wedge \quad A = A \circ \mu F \circ A$
□

Instantiating (8.12) gives us as definition that per A has to satisfy $A : A$ — $F.A$ and $A = [\![A, A]\!]$. The first condition can be simplified to $A \lhd F.A$, the second one to $A = A \circ \mu F \circ A$ because $A \lhd F.A \Rightarrow A \circ \mu F \circ A = [\![A, A]\!]$:

$$
\begin{array}{ll}
 & A \circ \mu F \circ A \; = \; [\![A, A]\!] \\
\Leftarrow & \qquad \{ \;\; \mu\text{-fusion, definition hylo} \;\; \} \\
 & \forall (Y \;\vdots\vdots\; A \circ F.Y \circ A \; = \; A \circ F.(A \circ Y \circ A) \circ A) \\
\equiv & \qquad \{ \;\; \text{relators} \;\; \} \\
 & \forall (Y \;\vdots\vdots\; A \circ F.Y \circ A \; = \; A \circ F.A \circ F.Y \circ F.A \circ A) \\
\equiv & \qquad \{ \;\; \bullet \; A \lhd F.A \;\; \} \\
 & \mathit{true}
\end{array}
$$

□

The pers satisfying $A \lhd F.A \quad \wedge \quad A = A \circ \mu F \circ A$ form an interesting class, but we will restrict our attention to a subclass called the F-inductive types that is defined by:

Definition 8.21: **(F-inductive type)**

A per A is an *F-inductive type* iff $A \lhd F.A \quad \wedge \quad A \lhd \mu F$
□

Every F-inductive type is an F-inductive algebra because $A \lhd \mu F \;\Rightarrow\; A = A \circ \mu F \circ A$, but not every per that is an F-inductive algebra is also an F-inductive type. For a relator F such that $\mu F \;\neq\; \perp\!\!\!\perp$ and $(\mu F)_{\ltimes} \;\neq\; (\nu F)_{\ltimes}$ it can be proved that per $\nu F \circ \top\!\!\!\top \circ \nu F$ is an F-inductive algebra but not an F-inductive type. The proof of this claim is left as an exercise to the reader.

One of the most important reasons for working with F-inductive types instead of using F-inductive algebras is that they form a complete lattice under the $\lhd$-order, allowing us to use extreme solutions of equations as definitions of datatypes. Examples will be given in the next chapter.

Lemma 8.22: **(Lattice of F-inductive types)**

(F-inductive types, $\lhd$) is a complete lattice with the same least upper bound as the per-lattice.
□

The pers A such that $A \lhd \mu F$ form an initial segment of the per-lattice and are therefore closed under the per-lub. Relator F is a monotonic function on the per-lattice, and this means that the post-fixpoints of F, the pers A such that $A \lhd F.A$, are closed under the per-lub. The combination gives us that the F-inductive types are closed under the per-lub, so they form a complete lattice with as lub the per-lub. The bottom of the lattice is $\perp\!\!\!\perp$ and the top is μF.

Earlier we mentioned that we want to choose one representative from every class of isomorphic F-inductive algebras. The main result about F-inductive types allowing such a choice is that every class of isomorphic F-inductive algebras contains exactly one F-inductive type, giving a one-to-one correspondence between F-inductive types and classes of isomorphic F-inductive algebras. We have the following theorem:

Theorem 8.23: (Algebra - Type correspondence)

For F-inductive type A and F-inductive algebra $\sigma\colon B \;\text{—}\; F.B$ we have the following Galois connections:

$$
\begin{array}{lrcl}
\text{(a)} & \mathcal{I}.A \gtrsim \sigma & \equiv & A \triangleright \llparenthesis \sigma \rrparenthesis_{\succ} \\
\text{(b)} & \sigma \gtrsim \mathcal{I}.A & \equiv & \llparenthesis \sigma \rrparenthesis_{\succ} \triangleright A \\
\text{(c)} & \mathcal{I}.A \simeq \sigma & \equiv & A = \llparenthesis \sigma \rrparenthesis_{\succ}
\end{array}
$$

□

The Galois connections are between the pre-order of the F-inductive algebras and the lattice of the F-inductive types. One of the adjoint functions is the identity function $\mathcal{I}$ embedding the F-inductive types into the F-inductive algebras. The other adjoint $R \mapsto \llparenthesis R \rrparenthesis_{\succ}$ should have the F-inductive types as range. This is proved with the following lemma:

Lemma 8.24: (Catamorphisms)

$\llparenthesis R \rrparenthesis_{\succ}$ is an F-inductive type for every spec R.

□

The fact that $\llparenthesis R \rrparenthesis_{\succ} \triangleleft \mu F$ follows immediately from the domain properties of catamorphisms. So it remains to prove:

$$
\begin{array}{ll}
 & \llparenthesis R \rrparenthesis_{\succ} \triangleleft F.\llparenthesis R \rrparenthesis_{\succ} \\
\equiv & \qquad \{ \text{ domains } \} \\
 & \llparenthesis R \rrparenthesis \circ F.\llparenthesis R \rrparenthesis_{\succ} = \llparenthesis R \rrparenthesis \\
\equiv & \qquad \{ \text{ calculation rule, definition catamorphism, relators } \} \\
 & R \circ F.(\llparenthesis R \rrparenthesis \circ \llparenthesis R \rrparenthesis_{\succ}) = \llparenthesis R \rrparenthesis \\
\equiv & \qquad \{ \text{ domains, calculation rule, definition catamorphism } \} \\
 & \mathit{true}
\end{array}
$$

□

Now we can prove Galois connection (8.23a):

$$
\begin{array}{ll}
 & A \gtrsim \sigma \\
\equiv & \qquad \{ \text{ definition } \gtrsim \} \\
 & \exists(\alpha \mathrel{¦} \alpha\colon B \;\text{—}\; A \mathrel{¦} \alpha \circ A = \sigma \circ F.\alpha) \\
\equiv & \qquad \{ \text{ domains, } A \triangleleft \mu F \} \\
 & \exists(\alpha \mathrel{¦} \alpha\colon B \;\text{—}\; A \mathrel{¦} \alpha = \sigma \circ F.\alpha \;\wedge\; \alpha \circ \mu F = \alpha) \\
\equiv & \qquad \{ \text{ one-point-rule, unique extension property } \}
\end{array}
$$

$$
\begin{array}{ll}
 & (\!|\sigma|\!) : B \leftarrow A \\
\equiv & \quad \{ \text{ domain catamorphisms, domains } \} \\
 & A \rhd (\!|\sigma|\!)_{\succ}
\end{array}
$$

□

Instantiating the proof above with $(\!|\sigma|\!)_{\succ}$ for A yields $(\!|\sigma|\!)_{\succ} \gtrsim \sigma$. We prove $\sigma \gtrsim (\!|\sigma|\!)_{\succ}$ by showing $(\!|\sigma|\!)_{\cup}$ to be the simulation:

$$
\begin{array}{ll}
 & (\!|\sigma|\!)_{\cup} \circ \sigma \\
= & \quad \{ \text{ domains, } \sigma \text{ } F\text{-inductive algebra, (8.12) } \} \\
 & (\!|\sigma|\!)_{\cup} \circ \sigma \circ F.((\!|\sigma|\!) \circ (\!|\sigma|\!)_{\cup}) \\
= & \quad \{ \text{ relators, calculation rule, definition catamorphism } \} \\
 & (\!|\sigma|\!)_{\cup} \circ (\!|\sigma|\!) \circ F.(\!|\sigma|\!)_{\cup} \\
= & \quad \{ \text{ } (\!|\sigma|\!) \text{ difunctional, domains } \} \\
 & (\!|\sigma|\!)_{\succ} \circ F.(\!|\sigma|\!)_{\cup}
\end{array}
$$

Now we can prove (8.23b):

$$
\begin{array}{ll}
 & \sigma \gtrsim A \\
\equiv & \quad \{ \text{ } (\!|\sigma|\!)_{\succ} \gtrsim \sigma \text{ and } \sigma \gtrsim (\!|\sigma|\!)_{\succ}\text{, see above } \} \\
 & (\!|\sigma|\!)_{\succ} \gtrsim A \\
\equiv & \quad \{ \text{ (8.23a) } \} \\
 & (\!|\sigma|\!)_{\succ} \rhd (\!|A|\!)_{\succ} \\
\equiv & \quad \{ \text{ } A \text{ } F\text{-inductive type } \Rightarrow (\!|A|\!) = A \Rightarrow (\!|A|\!)_{\succ} = A \text{ } \} \\
 & (\!|\sigma|\!)_{\succ} \rhd A
\end{array}
$$

Galois connection (8.23c) is the combination of (8.23a) and (8.23b) using the definition of $\simeq$ and the anti-symmetry of $\rhd$.

□

Knowing that an F-inductive type is an F-inductive algebra, we can take another look at the unique solution of (8.10) and (8.11) for the special case that τ is an F-inductive type. Substituting A for τ the equations become:

(8.25) $X \circ A \ = \ X$
(8.26) $X \circ A \ = \ R \circ F.X$

From (8.25) and $A \lhd \mu F$ it follows that $X \circ \mu F \ = \ X$ and from the combination of (8.25) and (8.26) it follows that $X \ = \ R \circ F.X$. Together this implies, by the unique extension property for catamorphisms, that the unique candidate for the solution is simply $(\!|R|\!)$, independent of which particular F-inductive type is used.

8.3 Parameterised inductive types

Many modern programming languages allow the construction of inductive datatypes that can be parameterised by another datatype. A standard example is the cons-list construction. One can define a type class of cons-list that can be instantiated to lists of booleans, list of numbers etc. We want to have the same capability in the SPEC-calculus. So, for example, we want to be able to define an operator that, given as argument a type (a per), produces as result the type of cons-lists over the argument type.

At the end of section 8.1 we saw that we can construct cons-lists over a type A by $\mu(X \mapsto \top\!\!\top + A{\times}X)$. The endorelator $X \mapsto \top\!\!\top + A{\times}X$ can also be seen as the binary relator $(Y,X) \mapsto \top\!\!\top + Y{\times}X$, namely by fixing the first argument to per A. Partially applying a binary relator to a per results in a unary relator and the corresponding inductive type is produced by simply taking the least fixpoint. This is the general principle for producing a parameterised inductive type for the cases with total and injective constructors. Given a binary relator $\otimes$, construct $\mu(A\otimes)$ as the inductive type parameterised with A.

From now on we will abbreviate the function $X \mapsto \mu(X\otimes)$ to the prefix-operator ϖ. The fact that ϖ maps a per to per is a consequence of the more general theorem:

Theorem 8.27:(Map)

For binary relator $\otimes$, ϖ is a relator.
□

Proof:

$$
\begin{array}{ll}
 & \varpi R \ \sqsupseteq\ \varpi S \\
\Leftarrow & \qquad \{\ \mu\text{-fusion using identity function}\ \} \\
 & \forall(X \mathrel{::} R{\otimes}X \ \sqsupseteq\ S{\otimes}X) \\
\Leftarrow & \qquad \{\ \text{relators}\ \} \\
 & R \ \sqsupseteq\ S
\end{array}
$$

$$
\begin{array}{ll}
 & (\varpi R)\cup \ =\ \varpi(R\cup) \\
\Leftarrow & \qquad \{\ \mu\text{-fusion}\ \} \\
 & \forall(X \mathrel{::} (R{\otimes}X)\cup \ =\ R\cup \otimes X\cup) \\
\equiv & \qquad \{\ \text{relators}\ \} \\
 & \mathit{true}
\end{array}
$$

$$
\begin{array}{ll}
 & \varpi R \circ \varpi S \ =\ \varpi(R \circ S) \\
\Leftarrow & \qquad \{\ \mu\text{-fusion}\ \} \\
 & \forall(X \mathrel{::} R{\otimes}X \circ \varpi S \ =\ (R \circ S){\otimes}(X \circ \varpi S)) \\
\equiv & \qquad \{\ \text{relators}\ \} \\
 & \forall(X \mathrel{::} R{\otimes}X \circ \varpi S \ =\ R{\otimes}X \ \circ\ S \otimes \varpi S) \\
\equiv & \qquad \{\ \text{calculation rule}\ \} \\
 & \mathit{true}
\end{array}
$$

The interpretation of ϖ as a type constructor on pers is clear. Let's now consider the interpretation on arbitrary specs. Writing τ for ϖI, we can derive the following equation for ϖR using the calculation rule and relator properties:

$$\varpi R \circ \tau \quad = \quad \tau \circ R{\otimes}I \circ I \otimes \varpi R$$

Interpreting this equation pointwise we see that it means that R is applied to all parameter components of the argument, but that the structure does not change. Such an operation is called a map operation in functional programming, and therefore we call the relators that are constructed using this method *map relators*.

One property that the map relators share with the polynomial relators is, given that the $\succ$ operation distributes over the underlying relator, it also distributes over the map.

$$
\begin{array}{ll}
 & (\varpi R)\succ \;=\; \varpi(R\succ) \\
\equiv & \qquad \{ \text{ domains, } \varpi R \circ \varpi I \;=\; \varpi R,\ \varpi I \text{ is a per } \} \\
 & (\varpi R)\succ \circ\, \varpi I \;=\; \varpi(R\succ) \\
\Leftarrow & \qquad \{ \ \mu\text{-fusion, definition } \varpi \ \} \\
 & \forall(Y \;\vdots\vdots\; (\varpi R)\succ \circ I{\otimes}Y \;=\; R\succ \otimes ((\varpi R)\succ \circ Y)) \\
\equiv & \qquad \{ \text{ relators } \} \\
 & \forall(Y \;\vdots\vdots\; (\varpi R)\succ \circ I{\otimes}Y \;=\; R\succ \otimes (\varpi R)\succ \circ I{\otimes}Y) \\
\Leftarrow & \qquad \{ \text{ calculation rule, monotonicity } \} \\
 & (R \otimes \varpi R)\succ \;=\; R\succ \otimes (\varpi R)\succ \\
\equiv & \qquad \{ \ \bullet\ \forall(P, Q \;\vdots\vdots\; (P{\otimes}Q)\succ \;=\; P\succ \otimes Q\succ) \ \} \\
 & \mathit{true}
\end{array}
$$

□

The map relator construction is the least fixpoint of a relator if the argument is a per, so then the result is an inductive datatype with total and injective constructors (an initial algebra). But we also want to be able to parameterise inductive types with non-total or non-injective constructors like join-list or height-balanced tree. This is possible if the type construction is completely polymorphic, i.e. does not "look inside" the parameter type. A datatype like sorted list over type A can only be constructed if one has an ordering on the elements of type A, so a general construction for every parameter type is impossible. Another problematic type is the finite set construction where an equality test is necessary for the construction.

The problem is, given a type-constructor φ (a total function from pers to pers), to construct a map-like relator that agrees with φ on pers. Our solution is to construct the relator as a subrelator of the map relator:

(8.28) $\varphi R \;\triangleq\; \varpi_{\varphi I} R$

There are some conditions that have to be imposed on φ to make this a good choice. The first is of course the naturality condition needed for subrelator construction:

$$\varphi I : \varpi \;\dot{\leftrightsquigarrow}\; \varpi$$

Secondly. we need that the subrelator agrees with φ when applied to a per. This means that $\varphi A \;=\; \varpi_{\varphi I} A$ should hold for all pers A. This holds iff

$$\varphi A \;=\; \varphi I \circ \varpi A, \text{ for all pers } A.$$

The naturality condition guarantees that the relator is polymorphic and does not look inside the parameter. In the following chapter we will show how parameterised types like join-lists and finite bags can be constructed using this method.

8.4 Other recursive programs

At the beginning of this chapter we mentioned the factorial function as an example of a recursively defined relation on an inductive datatype that cannot be expressed as a catamorphism. The catamorphism recursion scheme turns out to be too restrictive for many practical purposes. In this section we will examine more general recursion schemes allowing the specification of many "standard" programs.

The factorial function was given as an example of a function that can not be expressed as a catamorphism. From this example we will generalise a new shape for specifications having at most one solution and the catamorphism is another instance of this shape. The factorial function is defined by (after removing variables):

$$\begin{array}{lcl} fac \circ 0^\bullet & = & 1^\bullet \circ \top\!\!\top \\ fac \circ +1 & = & \times \circ (+1)\times I \circ \mathbb{N}\times fac \circ I\vartriangle I \end{array}$$

After juncing together this becomes:

$$fac \circ (0^\bullet \triangledown +1) \;=\; 1^\bullet \triangledown (\times \circ (+1)\times I) \circ \top\!\!\top+(I\times fac) \circ \top\!\!\top+(\mathbb{N}\vartriangle I)$$

Abstracting from this concrete example and adding the domain condition we obtain the following set of equations where $\tau : A \;—\; F.A$ is an F-inductive algebra:

$$\begin{array}{llcl} (8.29) & X \circ A & = & X \\ (8.30) & X \circ \tau & = & R \circ G.X \circ \gamma \end{array}$$

This form of specification is much more general than the equations for the catamorphism but is still not general enough. One kind of specification that can not be brought into this shape is the specification of a relation that has not just a single element of an inductive type as argument, but has other arguments as well. A simple example is the addition of two natural numbers specified by:

$$\begin{array}{lcl} add \circ \mathbb{N}\times 0^\bullet & = & \ll \\ add \circ \mathbb{N}\times(+1) & = & +1 \circ add \end{array}$$

Combining into a single equation leads after some calculation to the following equation:

$$add \circ \mathbb{N}\times(0\triangledown +1) \;=\; \ll\triangledown +1 \circ (\mathbb{N}\times\top\!\!\top)+(\mathbb{N}\times add) \circ ((I\times\hookrightarrow)\triangledown(I\times\hookleftarrow))^\cup$$

and abstraction from this example leads to the specification, for F-inductive algebra $\tau : A \;—\; F.A$:

$$\begin{array}{llcl} (8.31) & X \circ H.A & = & X \\ (8.32) & X \circ H.\tau & = & R \circ G.X \circ \gamma \end{array}$$

We will derive conditions under which this specification has at most one solution. Expressed as a theorem we obtain the following result:

Theorem 8.33:(Unique Extension Property)

For F-inductive algebra $\tau : A \;—\; F.A$, relator G, relator H with an upper Galois adjoint on lattice $(\{B \mid\mid B\subseteq A\}, \sqsubseteq)$ and spec γ with $\forall(B \mid B \subseteq A \mid \gamma \circ HF.B \sqsubseteq GH.B \circ \gamma)$:

$$X \circ H.A = X \;\wedge\; X \circ H.\tau = R \circ G.X \circ \gamma \quad\Rightarrow\quad X = [\![G; R, \gamma\circ H.\tau^\cup]\!]$$

□

The condition on γ looks like the naturality property $\gamma : GH \stackrel{\cdot}{\leftarrow} HF$ but is truly weaker. Instantiating the theorem with $F.X = \top\!\!\top + X$, $G.X = \top\!\!\top + (I \times X)$, $H.X = X$, $\gamma = \top\!\!\top + (\mathbb{N} \triangle I)$ and $A = \mathbb{N}$ as used in the factorial example demonstrates a γ that satisfies the condition in the theorem but that does not have the naturality property.

The proof of the theorem uses the fact that the definition of F-inductive algebra (which uses a hylomorphism that is defined as a least fixpoint in the SPEC-lattice), can be expressed using another hylomorphism defined as a least fixpoint in the lattice $(\{B \mid\mid B \subseteq A\}, \sqsubseteq)$. We want to use

$$[\![F; \tau, \tau^\cup]\!] \;=\; \mu(B \;\mapsto\; \tau \circ F.B \circ \tau^\cup)$$

where the rhs least fixpoint is taken in the lattice $(\{B \mid\mid B \subseteq A\}, \sqsubseteq)$. We need to check two conditions for proving the equality of the fixpoints using mutual inclusion and the induction rule for least fixpoints. The first condition to check is that $B \;\mapsto\; \tau \circ F.B \circ \tau^\cup$ is a monotonic endofunction on $(\{B \mid\mid B \subseteq A\}, \sqsubseteq)$, ensuring that the least fixpoint is well-defined and thereby allowing us to use induction for proving the $\sqsubseteq$ inclusion. The fact that $B \;\mapsto\; \tau \circ F.B \circ \tau^\cup$ is an endofunction follows from $\tau : A \longleftarrow F.A$, which follows from the assumption that τ is an F-inductive algebra using (8.14). The second condition to check is that $[\![F; \tau, \tau^\cup]\!] \subseteq A$, allowing us to use induction for proving the $\sqsupseteq$ inclusion. This follows immediately from the F-inductivity of τ.

We prove theorem (8.33) by mutual inclusion:

$$
\begin{array}{ll}
 & [\![G; R, \gamma \circ H.\tau^\cup]\!] \;\sqsubseteq\; X \\
\Leftarrow & \qquad \{ \text{ induction, definition hylomorphism } \} \\
 & R \circ G.X \circ \gamma \circ H.\tau^\cup \;\sqsubseteq\; X \\
\equiv & \qquad \{ \text{ (8.32), (8.31) } \} \\
 & X \circ H.\tau \circ H.\tau^\cup \;\sqsubseteq\; X \circ H.A \\
\equiv & \qquad \{ \;\bullet\; \tau \text{ } F\text{-inductive algebra; (8.14), relators } \} \\
 & \mathit{true}
\end{array}
$$

$$
\begin{array}{ll}
 & X \;\sqsubseteq\; [\![G; R, \gamma \circ H.\tau^\cup]\!] \\
\equiv & \qquad \{ \;\bullet\; \tau \text{ } F\text{-inductive algebra; (8.31) } \} \\
 & X \circ H.[\![F; \tau, \tau^\cup]\!] \;\sqsubseteq\; [\![G; R, \gamma \circ H.\tau^\cup]\!] \\
\Leftarrow & \qquad \{ \;\bullet\; H \text{ has an upper adjoint; } \mu\text{-fusion (lattice } (\{B \mid\mid B \subseteq A\}, \sqsubseteq)) \} \\
 & \forall(B \mid B \subseteq A \mid X \circ H.(\tau \circ F.B \circ \tau^\cup) \;\sqsubseteq\; R \circ G.(X \circ H.B) \circ \gamma \circ H.\tau^\cup) \\
\equiv & \qquad \{ \text{ relators } \} \\
 & \forall(B \mid B \subseteq A \mid X \circ H.\tau \circ HF.B \circ H.\tau^\cup \;\sqsubseteq\; R \circ G.X \circ GH.B \circ \gamma \circ H.\tau^\cup) \\
\equiv & \qquad \{ \text{ (8.32) } \} \\
 & \forall(B \mid B \subseteq A \mid R \circ G.X \circ \gamma \circ HF.B \circ H.\tau^\cup \;\sqsubseteq\; R \circ G.X \circ GH.B \circ \gamma \circ H.\tau^\cup) \\
\Leftarrow & \qquad \{ \text{ monotonicity } \} \\
 & \forall(B \mid B \subseteq A \mid \gamma \circ HF.B \;\sqsubseteq\; GH.B \circ \gamma)
\end{array}
$$

□

Note that H only has to have an upper adjoint for $B \subseteq A$, not for all specs. Several special cases of (8.33) have been studied in the literature. The factorial function can be written as an instance of a so-called *paramorphism* [45], which is defined as the solution of the equations:

$$X \circ A \;=\; X \quad \wedge \quad X \circ \tau \;=\; R \circ F.(I{\times}X) \circ F.(A{\vartriangle}I)$$

This shape of specification corresponds to primitive recursion if τ is the algebra of the natural numbers (relator $F.X \;=\; \top\!\!\top{+}X$). If we it with $m^\bullet \triangledown g$ for R where $m{\in}\mathbb{N}$ and g has type $\mathbb{N} \longleftarrow \mathbb{N}{\times}\mathbb{N}$ then the solution of the equation (called f here) satisfies

$$f.0 \;=\; m \quad \wedge \quad f.(n{+}1) \;=\; g.n.(f.n)$$

which is the standard way of defining a function by primitive recursion.

Another special case of (8.33) is the *mutumorphism* [24] where two specs are defined in a mutually recursive way:

$$\begin{array}{lllll} X \circ A & = & X & \wedge & X \circ \tau \;=\; R \circ F.(X{\vartriangle}Y) \\ Y \circ A & = & Y & \wedge & Y \circ \tau \;=\; S \circ F.(X{\vartriangle}Y) \end{array}$$

These equations can be combined by working in a binary SPEC-calculus:

$$(X,Y) \circ \delta.A \;=\; (X,Y) \quad \wedge \quad (X,Y) \circ \delta.\tau \;=\; (R,S) \circ \delta F{\times}.(X,Y) \circ \delta F.(I{\vartriangle}I)$$

This has the correct shape for (8.33). The doubling relator δ has as upper adjoint the $\sqcap$ operation.

A third special case is the *zygomorphism* [42], where also two specs are defined but one of these specs only depends on itself:

$$\begin{array}{lllll} X \circ A & = & X & \wedge & X \circ \tau \;=\; R \circ F.X \\ Y \circ A & = & Y & \wedge & Y \circ \tau \;=\; S \circ F.(X{\vartriangle}Y) \end{array}$$

These equations can also be transformed to the correct shape using a binary SPEC-calculus, resulting in the equation:

$$\begin{array}{l} (X,Y) \circ \delta.A \;=\; (X,Y) \\ (X,Y) \circ \delta.\tau \;=\; (R,S) \circ (F \ll, F{\times})\delta.(X,Y) \circ (F.I, F.(I{\vartriangle}I)) \end{array}$$

The shape of theorem 8.33 is general enough for almost all practical structural induction specifications, although it is sometimes quite a lot of work to transform a given specification to this shape. Combining equations for each constructor to a single equation means that one often has to introduce extra natural transformations. A small calculus of natural transformations between common relators is very useful for this purpose. A good basis for such a calculus (for the monotype system, but adaptable to our situation) can be be found in [1].

8.4.1 Injective F-inductive algebras

Theorem (8.33) guarantees the uniqueness of solutions to a large class of equations. But, a problem with the F-inductive algebra approach for structural induction is the *existence* of solutions to the specifying equations:

(8.34) $X \circ H.A \;=\; X$

(8.35) $X \circ H.\tau \;=\; R \circ G.X \circ \gamma$

We will show that solutions always exist, independent of R, if the F-inductive algebra τ is injective and γ satisfies some domain properties. The theorem that we are going to prove is the following:

Theorem 8.36:(Unique Extension Property)

For F-inductive algebra $\tau : A \rightarrowtail F.A$, relator G, relator H with an upper Galois adjoint on lattice $(\{B \ ¦¦\ B \subseteq A\}, \sqsubseteq)$ and spec $\gamma : \text{—}\ HF.A$ satisfying $\gamma{>} \sqsubseteq (H.\tau){>}$ and $\forall(B \ ¦\ B \subseteq A \ ¦\ \gamma \circ HF.B \sqsubseteq GH.B \circ \gamma)$:

$$X \circ H.A = X \ \wedge \ X \circ H.\tau = R \circ G.X \circ \gamma \quad \equiv \quad X = [\![G; R, \gamma \circ H.\tau{\cup}]\!]$$

□

The $\Rightarrow$ implication follows from theorem (8.33) and the fact that $[\![G; R, \gamma \circ H.\tau{\cup}]\!]$ satisfies (8.34) is trivial to prove. We show that $[\![G; R, \gamma \circ H.\tau{\cup}]\!]$ satisfies (8.35):

$$\begin{array}{ll}
 & [\![G; R, \gamma \circ H.\tau{\cup}]\!] \circ H.\tau \\
= & \qquad \{ \text{ calculation rule, definition hylo } \} \\
 & R \circ G.[\![G; R, \gamma \circ H.\tau{\cup}]\!] \circ \gamma \circ H.\tau{\cup} \circ H.\tau \\
= & \qquad \{ \ \tau{\cup} \circ \tau = \tau{>} \circ F.A \text{ (see below), relators } \} \\
 & R \circ G.[\![G; R, \gamma \circ H.\tau{\cup}]\!] \circ \gamma \circ H.\tau{>} \circ HF.A \\
= & \qquad \{ \text{ relators, domains } \} \\
 & R \circ G.[\![G; R, \gamma \circ H.\tau{\cup}]\!] \circ \gamma \circ (H.\tau){>} \circ H.I \circ HF.A \\
= & \qquad \{ \text{ relators, } \bullet\ \gamma{>} \sqsubseteq (H.\tau){>},\ \gamma : \text{—}\ HF.A \ \} \\
 & R \circ G.[\![G; R, \gamma \circ H.\tau{\cup}]\!] \circ \gamma
\end{array}$$

We prove $\tau{\cup} \circ \tau = \tau{>} \circ F.A$ by

$$\begin{array}{ll}
 & \tau{\cup} \circ \tau \\
= & \qquad \{ \ \bullet\ \tau : \text{—}\ F.A \ \} \\
 & \tau{\cup} \circ \tau \circ F.A \\
\sqsupseteq & \qquad \{ \text{ domains } \} \\
 & \tau{>} \circ F.A \\
\sqsupseteq & \qquad \{ \ \bullet\ \tau : \rightarrowtail F.A \ \} \\
 & \tau{>} \circ \tau{\cup} \circ \tau \\
= & \qquad \{ \text{ domains } \} \\
 & \tau{\cup} \circ \tau
\end{array}$$

□

The theorem shows that unique solutions for the structural recursion equations exist if the inductive datatype only has restrictions. In the next chapter we will investigate conditions for the existence of solutions when there are laws. We will demonstrate there that conditions have to be imposed on R for guaranteeing the existence of solutions.

The extra domain conditions that were imposed on γ are rather weak. If we start off with a spec γ that only satisfies the condition $\forall(B \ ¦\ B \subseteq A \ ¦\ \gamma \circ HF.B \sqsubseteq GH.B \circ \gamma)$, then we can safely restrict it to

$$\gamma \circ H.\tau{>} \circ HF.A$$

without changing the solution to (8.34) and (8.35) because

$$[\![G; R, \gamma \circ H.\tau\cup]\!] = [\![G; R, \gamma \circ H.\tau{>} \circ HF.A \circ H.\tau\cup]\!]$$

Adding the restriction preserves the "naturality" property, so there is still at most one solution. We prove this preservation property by:

$$
\begin{array}{ll}
 & GH.B \circ \gamma \circ H.\tau{>} \circ HF.A \\
\sqsupseteq & \quad \{ \ \bullet \forall(B \mid B \subseteq A \mid \gamma \circ HF.B \sqsubseteq GH.B \circ \gamma) \ \} \\
 & \gamma \circ HF.B \circ H.\tau{>} \circ HF.A \\
\sqsupseteq & \quad \{ \ \text{relators, } F.B \circ \tau{>} \sqsupseteq \tau{>} \circ F.B \text{ (see below) } \} \\
 & \gamma \circ H.\tau{>} \circ HF.B \circ HF.A \\
= & \quad \{ \ \text{relators, } \bullet B \subseteq A \ \} \\
 & \gamma \circ H.\tau{>} \circ HF.A \circ HF.B
\end{array}
$$

$$
\begin{array}{ll}
 & F.B \circ \tau{>} \\
= & \quad \{ \ \text{domains} \ \} \\
 & F.B \sqcap \top\!\!\top \circ \tau \\
= & \quad \{ \ \tau : \text{---} F.A \ \} \\
 & F.B \sqcap \top\!\!\top \circ \tau \circ F.A \\
\sqsupseteq & \quad \{ \ \bullet B \subseteq A; \tau{>} \sqsubseteq I \ \} \\
 & \tau{>} \circ F.B \sqcap \top\!\!\top \circ \tau \circ F.B \\
= & \quad \{ \ \tau{>} \sqsubseteq \top\!\!\top \circ \tau \ \} \\
 & \tau{>} \circ F.B
\end{array}
$$

□

8.5 F-reductivity

We chose F-inductive algebras as generalisation of initial F-algebras, but this is not the only possible choice. Another approach is based on the fact that the set of equations

$$
\begin{aligned}
X \circ A &= X \\
X \circ \tau &= R \circ F.X
\end{aligned}
$$

can be rewritten to the single equation

$$(8.37) \quad X = R \circ F.X \circ \tau\cup$$

if τ satisfies $\tau : A \leftarrowtail\!\!\twoheadrightarrow F.A$, which is the case for an initial F-algebra. The equation (8.37) has as advantage over the original specification that solutions always exist, in particular the least solution $[\![R, \tau\cup]\!]$. The notion of F-reductivity was invented by Doornbos and Backhouse [19, 20, 21] for the investigation of conditions under which (8.37) has at most one solution. Doornbos also describes a notion called F-inductivity in his thesis [19], but this is not the same as our notion.

The definition of F-reductivity uses a new operator $\searrow$ called the *monotype factor* that is defined by the following Galois connection (both orderings in the pid-lattice):

$$(R \circ A)< \;\sqsubseteq\; B \quad\equiv\quad A \;\sqsubseteq\; R\backslash B$$

The notion of F-reductivity is defined by

Definition 8.38: (F-reductivity)

A spec R is F*-reductive* iff $\mu(A \;\mapsto\; R\backslash F.A) \;=\; I$.
□

The least fixpoint in this definition is in the pid-lattice. We have the following theorem about the unique solution of (8.37):

Theorem 8.39: (F-reductivity)

For F-reductive spec S we have the following unique extension property:

$$X \;=\; R \circ F.X \circ S \quad\equiv\quad X \;=\; [\![R, S]\!]$$

□

The proof of this theorem can be found in [5].

Working with F-reductivity has as advantage that one is no longer restricted to algebras with their functionality for determining whether there exists a unique solution for the equation we are interested in. An important disadvantage of F-reductivity is that the equation (8.37) does not have a unique solution if we are working with types with laws, like for example join-lists. The reverse of a constructor of a type with laws is in general not F-reductive, so theorem (8.39) can not be used. This precludes using the reductivity theory for many common datatypes with laws.

There seems to be a close correspondence between F-inductivity and F-reductivity. For almost any result about F-inductivity there exists a similar result about F-reductivity and vice versa. For example, our theorem (8.33) was inspired by theorem 35 in [5]:

Let R be an F-reductive spec, $\alpha : HG \;\dot{\leftsquigarrow}\; GF$ and G a relator that has an upper adjoint for pids. Then $\alpha \circ G.R$ is H-reductive.

Having the similarity in results one would expect that the proofs of the results would also be similar, but this is not the case. Further research into the relationship of F-inductivity and F-reductivity is necessary.

Chapter 9

Equational specification of datatypes

The previous chapter introduced a general theory about inductive datatypes and recursively defined programs on these datatypes, but did not give methods for the construction of instances of these inductive types (except for the trivial cases of μF and $\perp\!\!\perp$). The main subject of this chapter is to give methods for the construction of inductive types with properties expressed by equations.

An important theme in this chapter is the shape of equations expressing properties of inductive datatypes. We would like to solve equations using the Knaster-Tarski theorem, but this requires monotonicity of the specifying equations. The "obvious" shape of equations expressing laws or restrictions is not suitable for Knaster-Tarski and we will show how and when these equations can be transformed to a suitable form.

A problem in the previous chapter was that we defined programs by sets of equations with at most one solution, but were not able to provide conditions under which a solution exists. For inductive types augmented with equations specifying laws or restrictions we can derive conditions under which the catamorphism equation does have a solution. This is demonstrated by the Boom-hierarchy, an important class of parameterised inductive types. Finally, we examine many-sorted algebras and show that the techniques developed for inductive types can also be used for the construction of many-sorted algebras.

9.1 Constructing F-inductive algebras

This section will show a method for the construction of F-inductive algebras based on imposing laws and restrictions on the carrier of the algebra. The process we have in mind is as follows. Beginning with a free algebra isomorphic to μF, we want to refine the type by imposing a succession of laws and/or restrictions. At each stage, the algebra constructed should be F-inductive. Thus, the general step is, given F-inductive algebra $\tau : A — F.A$, to construct a new F-inductive algebra $\delta : B — F.B$ where B

is formed by imposing laws (quotients) and restrictions (subtypes) on A. This means (see (5.37)) that $B \lhd A$. The obvious candidate for δ is then $B \circ \tau$ and our first task is to find a necessary and sufficient condition on B making δ an F-inductive algebra. Such a condition is expressed in the following lemma:

Lemma 9.1: (F-inductive algebras)

For F-inductive algebra $\tau : A \sim F.A$ and $B \lhd A$, $B{\circ}\tau$ is an F-inductive algebra iff $B \circ \tau \ = \ B \circ \tau \circ F.B$.
□

For $B{\circ}\tau$ to be an F-inductive algebra we need to have a per C such that $B{\circ}\tau : C \sim F.C$ and $C \ = \ [\![B{\circ}\tau, (B{\circ}\tau)\cup]\!]$. By (8.14) this implies $B{\circ}\tau : C \leftarrowtail$, so $C \ = \ (B{\circ}\tau) \circ (B{\circ}\tau)\cup$ which gives us, using $\tau : A \leftarrowtail$ and $B \lhd A$, that $C \ = \ B$. Substituting C by B has as result that the condition for F-inductivity becomes $B{\circ}\tau : B \sim F.B$ and $B \ = \ [\![B{\circ}\tau, (B{\circ}\tau)\cup]\!]$. Instantiation of the definition of the type judgement and using $B \circ B \ = \ B$ simplifies $B{\circ}\tau : B \sim F.B$ to $B{\circ}\tau \ = \ B{\circ}\tau{\circ}F.B$. The other condition, $B \ = \ [\![B{\circ}\tau, (B{\circ}\tau)\cup]\!]$, follows from $B{\circ}\tau \ = \ B{\circ}\tau{\circ}F.B$:

$$
\begin{array}{ll}
 & B \ = \ [\![B{\circ}\tau, (B{\circ}\tau)\cup]\!] \\
\equiv & \qquad \{ \ B \lhd A \ \} \\
 & B \circ A \circ B \ = \ [\![B{\circ}\tau, (B{\circ}\tau)\cup]\!] \\
\equiv & \qquad \{ \ \tau \ F\text{-inductive algebra} \ \} \\
 & B \circ [\![\tau, \tau\cup]\!] \circ B \ = \ [\![B{\circ}\tau, (B{\circ}\tau)\cup]\!] \\
\Leftarrow & \qquad \{ \ \mu\text{-fusion, definition hylo} \ \} \\
 & \forall (X \ :: \ B \circ \tau \circ F.X \circ \tau\cup \circ B \ = \ B \circ \tau \circ F.(B \circ X \circ B) \circ (B{\circ}\tau)\cup) \\
\Leftarrow & \qquad \{ \ \text{relators, reverse, monotonicity} \ \} \\
 & B \circ \tau \ = \ B \circ \tau \circ F.B
\end{array}
$$
□

For the special case where the algebra is an F-inductive type $A : A \sim F.A$, the condition in (9.1) specialises to $B \circ A \ = \ B \circ A \circ F.B$. This condition can be simplified, using $B \lhd A$, to $B \ \lhd \ F.B$ and from $A \ \lhd \ \mu F$ and $B \lhd A$ it follows that $B \ \lhd \ \mu F$. This means that B is an F-inductive type such that $B \lhd A$.

An important problem is the existence of solutions of the structural induction specification:

$$(9.2) \quad X \circ B \ = \ X \quad \wedge \quad X \circ \delta \ = \ R \circ F.X \circ \delta{>}$$

and the relationship with the existence of solutions to the structural induction specification for the original algebra τ:

$$(9.3) \quad X \circ A \ = \ X \quad \wedge \quad X \circ \tau \ = \ R \circ F.X \circ \tau{>}$$

The domain restrictions are added because we are working with possibly non-total algebras (see section 8.4.1 for more details about the domain restriction). We investigate two special cases for B, imposing restrictions ($B \subseteq A$) and imposing laws ($B {\lhd\!\!|} A$). Imposing restrictions on the carrier ($B \subseteq A$) does not influence the existence of solutions, as is formulated in the following lemma:

Lemma 9.4: (Restricted F-inductive algebra)

For F-inductive algebra $\tau : A \!—\! F.A$, $B \subseteq A$ such that $B{\circ}\tau = B{\circ}\tau{\circ}F.B$ and $\delta = B{\circ}\tau$:

$$\begin{array}{ll} & X \circ A \;=\; X \quad \wedge \quad X \circ \tau \;=\; R \circ F.X \circ \tau_{>} \\ \Rightarrow & \\ & (X \circ B) \circ B \;=\; X \circ B \quad \wedge \quad (X \circ B) \circ \delta \;=\; R \circ F.(X \circ B) \circ \delta_{>} \end{array}$$

□

The proof of the first conjunct is trivial, the second conjunct is proved by

$$\begin{array}{ll} & X \circ B \circ \delta \\ = & \qquad \{\ \delta : B \,— \ \} \\ & X \circ \delta \\ = & \qquad \{\ \delta \;=\; \tau \circ F.B \circ \delta_{>} \text{ (see below) } \} \\ & X \circ \tau \circ F.B \circ \delta_{>} \\ = & \qquad \{\ \bullet\ X \circ \tau \;=\; R \circ F.X \circ \tau_{>} \ \} \\ & R \circ F.X \circ \tau_{>} \circ F.B \circ \delta_{>} \\ = & \qquad \{\ F.B \circ \delta_{>} \;=\; \delta_{>} \circ F.B \;\Leftarrow\; \delta : \,—\, F.B,\ \tau_{>} \;\sqsupseteq\; (B{\circ}\tau)_{>} \;=\; \delta_{>} \ \} \\ & R \circ F.X \circ F.B \circ \delta_{>} \\ = & \qquad \{\ \text{relators} \ \} \\ & R \circ F.(X \circ B) \circ \delta_{>} \end{array}$$

$$\begin{array}{ll} & \delta \\ = & \qquad \{\ \delta : B \,—,\ B \lhd A, \text{ domains}, \ \tau : A \leftarrowtail \ \} \\ & \tau \circ \tau^{\cup} \circ \delta \\ = & \qquad \{\ \delta : B \,—,\ B{\circ}\tau \;=\; \delta,\ \delta : \,—\, F.B \ \} \\ & \tau \circ F.B \circ \delta^{\cup} \circ \delta \\ \sqsupseteq & \qquad \{\ \text{domains} \ \} \\ & \tau \circ F.B \circ \delta_{>} \\ \sqsupseteq & \qquad \{\ \tau : A \,—,\ B \;\sqsubseteq\; A \ \} \\ & B \circ \tau \circ F.B \circ \delta_{>} \\ = & \qquad \{\ B \circ \tau \;=\; \delta,\ \delta : \,—\, F.B, \text{ domains} \ \} \\ & \delta \end{array}$$

□

The interpretation of this lemma is that if the carrier of F-inductive algebra τ is restricted to some per B, and if (9.3) has a solution, then (9.2) also has a solution, being the solution of (9.3) restricted to B. For imposing laws we obtain the following result:

Lemma 9.5: (Lawful F-inductive algebra)

For F-inductive algebra $\tau : A \!—\! F.A$, $B \trianglelefteq A$ such that $B{\circ}\tau = B{\circ}\tau{\circ}F.B$ and $\delta = B{\circ}\tau$:

$$
\begin{array}{ll}
 & X \circ A \;=\; X \quad \wedge \quad X \circ \tau \;=\; R \circ F.X \circ \tau{>} \quad \wedge \quad X \circ B \;=\; X \\
\equiv & \\
 & X \circ B \;=\; X \quad \wedge \quad X \circ \delta \;=\; R \circ F.X \circ \delta{>}
\end{array}
$$

□

From $B \lhd A$ we have $X \circ B \;=\; X \quad \Rightarrow \quad X \circ A \;=\; X$, reducing the proof obligation to:

$$X \circ B \;=\; X \quad \Rightarrow \quad (X \circ \tau \;=\; R \circ F.X \circ \tau{>} \quad \equiv \quad X \circ \delta \;=\; R \circ F.X \circ \delta{>})$$

This is proved by:

$$
\begin{array}{ll}
 & X \circ \delta \;=\; X \circ \tau \\
\equiv & \qquad \{ \;\bullet\, X \circ B \;=\; X \;\} \\
 & X \circ \delta \;=\; X \circ B \circ \tau \\
\equiv & \qquad \{ \;\delta \;=\; B{\circ}\tau \;\} \\
 & \mathit{true}
\end{array}
$$

$$
\begin{array}{ll}
 & R \circ F.X \circ \tau{>} \;=\; R \circ F.X \circ \delta{>} \\
\Leftarrow & \qquad \{ \;\text{monotonicity}, \delta \;=\; B{\circ}\tau \;\} \\
 & \tau{>} \;=\; (B{\circ}\tau){>} \\
\equiv & \qquad \{ \;\tau : A \longleftarrow, \text{domains} \;\} \\
 & (A{>} \circ \tau){>} \;=\; (B{>} \circ \tau){>} \\
\equiv & \qquad \{ \;A{>} \;=\; B{>} \quad \Leftarrow \quad B \lhd\!| A \;\} \\
 & \mathit{true}
\end{array}
$$

□

The interpretation of lemma (9.5) is that imposing a law does not change the solution of the structural induction specification but can invalidate it. The specification with the law has a solution if the original specification has a solution and this solution also has the new carrier as a right domain.

9.2 Equations

Join-lists and height-balanced trees have been mentioned as examples of inductive types, but a method for the construction of these and other types with possibly non-injective or non-total constructors was not given. This section will show a method for the construction of inductive types with properties specified by equations. We will consider two types of equations, first equations that specify a quotient type construction (laws) and then equations for subtype construction (restrictions). The calculations in this section are done with F-inductive types instead of with F-inductive algebras because the lattice properties of F-inductive types are essential for definitions of types as extreme solutions of equations.

The non-empty join-lists form an example of a type with non-injective constructors, because for join-lists a, b and c we have the following equality:

$$(a +\!\!+\, b) +\!\!+\, c \quad = \quad a +\!\!+\, (b +\!\!+\, c)$$

Such an equation specifies a law, an equality between two elements of the inductive type that are constructed in different ways. Written without variables this equation is transformed into:

$$+\!\!+ \circ +\!\!+\!\times I \circ \mathsf{assoc} \quad = \quad +\!\!+ \circ I \times +\!\!+$$

The relator corresponding to non-empty join-lists over A is $F.X = A + X \times X$ and we start the construction of the lists with as base type the non-empty join-trees μF. This F-inductive type has two constructors, $\mathit{leaf} : \mu F \leftarrow A$ and $\mathit{join} : \mu F \leftarrow \mu F \times \mu F$. Following the type-with-law construction in the previous section, using that μF is an F-inductive type, we want to construct an F-inductive type $JL \lhd\!\!\shortmid\, \mu F$, the non-empty join-lists. The constructors of this new type are $\tau : JL \leftarrow A$ and $+\!\!+ : JL \leftarrow JL \times JL$ and they can be written as the composition of JL and the corresponding constructor of μF:

$$
\begin{array}{ll}
 & \tau \\
= & \qquad \{ \text{ definition constructor } \} \\
 & JL \circ \hookrightarrow \\
= & \qquad \{ \ JL \lhd \mu F \ \} \\
 & JL \circ \mu F \circ \hookrightarrow \\
= & \qquad \{ \text{ definition constructor } \} \\
 & JL \circ \mathit{leaf}
\end{array}
$$

Similarly we have $+\!\!+ \ = \ JL \circ \mathit{join}$.

The law equation is given using the constructors from JL but can be rewritten to a form using the constructors of μF. We demonstrate this on the lhs of the equation, the calculation for the rhs being similar:

$$
\begin{array}{ll}
 & +\!\!+ \circ +\!\!+\!\times I \circ \mathsf{assoc} \\
= & \qquad \{ \ +\!\!+ \ = \ JL \circ \mathit{join} \ \} \\
 & +\!\!+ \circ (JL \circ \mathit{join}) \times I \circ \mathsf{assoc} \\
= & \qquad \{ \text{ relators } \} \\
 & +\!\!+ \circ JL \times I \circ \mathit{join} \times I \circ \mathsf{assoc} \\
= & \qquad \{ \ +\!\!+ : \text{---}\ JL \times JL \ \} \\
 & +\!\!+ \circ \mathit{join} \times I \circ \mathsf{assoc} \\
= & \qquad \{ \ +\!\!+ \ = \ JL \circ \mathit{join} \ \} \\
 & JL \circ \mathit{join} \circ \mathit{join} \times I \circ \mathsf{assoc}
\end{array}
$$

The rewritten version of the law equation becomes:

$$JL \circ \mathit{join} \circ \mathit{join} \times I \circ \mathsf{assoc} \quad = \quad JL \circ \mathit{join} \circ I \times \mathit{join}$$

We want JL to be as large as possible under the $\lhd$ order, equating as few elements of μF as possible. Generalising from this particular example we have some F-inductive type B (μF in the example) and want to construct another F-inductive type X, $X \lhd\!\!\shortmid B$, X as large as possible under the $\lhd$-order, such that

(9.6) $X \circ f \quad = \quad X \circ g$

where f and g have type $B \longleftarrow C$ for some type C ($\mu F \times (\mu F \times \mu F)$ in the example). We want a largest solution in the per-lattice, but the equations that X has to satisfy are not in the correct shape for Knaster-Tarski, so it is not immediately clear that a largest solution does exist. We will show later on in lemma (9.8) that equations for laws of the form (9.6) can be rewritten in a form suitable for Knaster-Tarski, but first we examine another type of equation corresponding to restrictions (or non-total constructors).

Height-balanced trees are an example of an inductive type with a non-total constructor. The aim is to construct a subtype of the non-empty join-trees where every tree is height-balanced, i.e. the height of the left and the right subtree differ by at most one at every level of the tree. One way of doing this is by constructing functions that return the height of the left or the right subtree and only keeping those trees for which the difference is at most one. The standard function calculating the height of a subtree is partial because singleton trees don't have subtrees, but we do want singleton trees to be height-balanced. This can be achieved by using 0 for the height of subtrees of singleton trees and adding 1 to the height of subtrees of trees constructed with the *join* constructor. We specify the function *lheight* for the height of the left subtree by:

$$\begin{aligned} \mathit{lheight} \circ \mathit{leaf} &= 0^\bullet \\ \mathit{lheight} \circ \mathit{join} &= \mathit{1+} \circ \mathit{height} \circ \ll \circ \mu F \times \mu F \end{aligned}$$

The two equations can be combined to a single definition with the junc operation using $\mathit{leaf} \triangledown \mathit{join} = \mu F$ and $\top\!\top + \mu F \times \mu F = \mu F$:

$$\mathit{lheight} \triangleq 0^\bullet \triangledown (\mathit{1+} \circ \mathit{height} \circ \ll) \circ \mu F$$

The function *height* is the function calculating the height of a tree by structural induction and can be defined as an F-catamorphism:

$$\mathit{height} \triangleq (\!|0^\bullet \triangledown (1+ \circ \mathit{max})|\!)$$

with *max* the maximum function on natural numbers. The function calculating the height of the right subtree can be defined by:

$$\mathit{rheight} \triangleq 0^\bullet \triangledown (\mathit{1+} \circ \mathit{height} \circ \gg) \circ \mu F$$

The relation between natural numbers expressing that their difference is at most one is $(1+) \sqcup \mathbb{N} \sqcup (1+)^\cup$.

There are several ways of combining the preceding notions for expressing the restriction of height-balancedness. The form chosen here has as advantage that it can be transformed to the correct shape for using Knaster-Tarski (see lemma (9.9)). The specification for height-balanced trees is to construct an F-inductive type $X \subseteq \mu F$ satisfying:

$$((1+) \sqcup \mathbb{N} \sqcup (1+)^\cup) \circ \mathit{lheight} \circ X \quad \sqsupseteq \quad \mathit{rheight} \circ X$$

We are looking for the $\lhd$-largest X because we want to keep as many trees as possible.

Generalisation of the specification above yields the following shape of equations for subtyping: given an F-inductive type B, find the per-lattice greatest F-inductive type $X \subseteq B$ such that:

(9.7) $R \circ X \quad \sqsupseteq \quad f \circ X$

where $R\colon A \longleftarrow B$ and $f : A \longleftarrow\!\dashv B$ for some type A. Again we have an equation that does not have the correct shape for Knaster-Tarski, but we will show that both the equation for laws and the equation for subtyping are special instances of a more general equation that does have the correct shape for Knaster-Tarski.

In both equations we have an F-inductive type B and want to construct the largest X in the per-lattice satisfying:

$$X \lhd B$$
$$X \lhd F.X$$

and an extra condition ((9.6) or (9.7)) that is not of the form $X \lhd \varphi.X$ as is required for Knaster-Tarski. Additionally we require in the case of a law that the greatest solution satisfies $X \lhd\!\!| B$ and in the case of a restriction satisfies $X \subseteq B$. We will show that (9.6) and (9.7) can be rewritten in the form $X \lhd P$ where P is a constant per depending on the desired property for the type under construction. We start with the equation for laws. The per P is formulated in the following lemma, assuming f,g have type $B \longleftarrow\!\dashv C$ for some per C and B is an F-inductive type:

Lemma 9.8: (Law equation)

For spec X with $X \circ B \;=\; X$ and per $P \;\triangleq\; \mu F \sqcup (f \circ g^{\cup} \;\sqcup\; g \circ f^{\cup})^{+}$:

$$X \circ P \;=\; X \quad\equiv\quad X \circ f \;=\; X \circ g$$

□

Proof:

$$
\begin{array}{ll}
 & X \circ P \;=\; X \\
\equiv & \qquad \{\ \text{definition } P,\ X \circ B \;=\; X \quad\wedge\quad B \lhd \mu F \quad\Rightarrow\quad X \circ \mu F \;=\; X\ \} \\
 & X \circ (f \circ g^{\cup} \;\sqcup\; g \circ f^{\cup})^{+} \;\sqsubseteq\; X \\
\equiv & \qquad \{\ \text{factors}\ \} \\
 & (f \circ g^{\cup} \;\sqcup\; g \circ f^{\cup})^{+} \;\sqsubseteq\; X \setminus X \\
\equiv & \qquad \{\ X \setminus X \text{ transitive, transitive closure}\ \} \\
 & f \circ g^{\cup} \;\sqcup\; g \circ f^{\cup} \;\sqsubseteq\; X \setminus X \\
\equiv & \qquad \{\ \text{factors, cup}\ \} \\
 & X \circ f \circ g^{\cup} \;\sqsubseteq\; X \quad\wedge\quad X \circ g \circ f^{\cup} \;\sqsubseteq\; X \\
\equiv & \qquad \{\ \text{Galois connection (see below)},\ \top\!\!\top \circ f \;=\; \top\!\!\top \circ g\ \} \\
 & X \circ f \sqsubseteq X \circ g \quad\wedge\quad X \circ g \sqsubseteq X \circ f \\
\equiv & \qquad \{\ \text{anti-symmetry}\ \} \\
 & X \circ f \;=\; X \circ g
\end{array}
$$

The penultimate step uses a generalisation of (4.11). Specifically, for $f : A \longleftarrow$ and specs X and Y satisfying $X \circ A \;=\; X$ and $Y \;\sqsubseteq\; \top\!\!\top \circ f$ the following Galois connection can be proved (straightforward generalisation of the proof for (4.11)):

$$Y \sqsubseteq X \circ f \quad\equiv\quad Y \circ f^{\cup} \sqsubseteq X$$

Lemma (4.11) is an instantiation with $A \;=\; I$

□

Note that the per P that is defined in (9.8) satisfies

$$P \lhd\!\!| \ \mu F$$

We will show later that this helps to guarantee that the constructed type is an equivalence relation on B. Another useful property of P is that it satisfies the equation, i.e. $P \circ f \ = \ P \circ g$. This result is obtained by instantiating (9.8) with P for X, using that P is a per. The proof above doesn't show or use that P is a per; the perness of P follows from $(f \circ g^{\cup} \sqcup g \circ f^{\cup})^{+} \lhd \mu F$ using (5.37), which follows from $f{\circ}g^{\cup} \sqcup g{\circ}f^{\cup} : \,—\, \mu F$, which follows from $f,g : B \leftarrow$ and $B \lhd \mu F$.

For the equation corresponding to restrictions we have another lemma. We assume here that $R\colon A \,—\, B$, $f : A \hookleftarrow B$ for some type A and that B is an F-inductive type:

Lemma 9.9: (Restriction equation)

For spec X with $X : B \leftarrow$ and per $P \ \triangleq \ \mu F \sqcap f^{\cup} \circ R$:

$$P \circ X \ = \ X \quad \equiv \quad R \circ X \ \sqsupseteq \ f \circ X$$

□

Proof:

$$\begin{array}{ll}
 & P \circ X \ = \ X \\
\equiv & \quad \{ \text{ definition } P\text{, distribution, } R\colon — B,\ X : B \leftarrow \ \} \\
 & \mu F \circ X \ \sqcap \ f^{\cup} \circ R \circ X \ = \ X \\
\equiv & \quad \{ \ B \lhd \mu F \ \wedge \ X : B \leftarrow \ \Rightarrow \ X : \mu F \,—\text{, cap } \} \\
 & X \ \sqsubseteq \ f^{\cup} \circ R \circ X \\
\equiv & \quad \{ \text{ Galois connection (see proof (9.8)), } A \circ R \ = \ R, \\
 & \quad\quad X \ \sqsubseteq \ B \circ \top\!\!\top = f^{\cup} \circ \top\!\!\top \ \} \\
 & f \circ X \ \sqsubseteq \ R \circ X
\end{array}$$

□

The fact that P as defined above is a per and satisfies $P \subseteq \mu F$ follows using (5.53) from $f^{\cup} \circ R\colon \mu F \,—\, \mu F$ which follows from $R\colon — B$, $f : \hookleftarrow B$ and $B \lhd \mu F$. The result that P satisfies

$$P \subseteq \mu F$$

guarantees, as we will show later, that the constructed type is a subset of B. Per P also satisfies its equation $(R \circ P \ \sqsupseteq \ f \circ P)$. We would like to prove this by instantiating (9.9) with P for X but this is not allowed because $P\colon B \,—$ does not hold in general. A separate proof is necessary :

$$\begin{array}{ll}
 & f \circ (\mu F \ \sqcap \ f^{\cup} \circ R) \\
= & \quad \{ \text{ pers } \} \\
 & f \circ (\mu F \ \sqcap \ f^{\cup} \circ R) \circ (\mu F \ \sqcap \ f^{\cup} \circ R) \\
\sqsubseteq & \quad \{ \text{ monotonicity } \} \\
 & f \circ f^{\cup} \circ R \circ (\mu F \ \sqcap \ f^{\cup} \circ R) \\
\sqsubseteq & \quad \{ \text{ type } f \text{ and } R \ \} \\
 & R \circ (\mu F \ \sqcap \ f^{\cup} \circ R)
\end{array}$$

9.2.1 F-inductive closures

The inductive type that we want to construct by adding the equation(s) corresponding to P to base type B is the greatest solution of the equation:

$$X \;\lhd\; F.X \wedge P \wedge B$$

The conjunct $F.X$ is added to make this largest solution F-inductive. The result that the equations expressing laws and expressing restrictions are both special cases of the *same* monotonic equation in the per-lattice seems to be new and is a considerable simplification compared with, for example, categorical approaches to datatype construction with equations. Also important for manipulation is the fact that both types with laws and types with restrictions are greatest solutions. In contrast, set-theoretic approaches often define types with laws as least solutions and types with restrictions as greatest solutions, making it difficult to combine laws and restrictions.

The type to be constructed can be defined as the greatest solution of an equation and this normally means that many properties will be proven using induction. In this special case we can do something else because it can also be expressed as an upper adjoint in a Galois connection. This has two major advantages: first, the standard properties of Galois connections can be instantiated giving "free" results about the type, and secondly, proofs using the Galois connection are usually proofs consisting of equivalences while proofs using greatest fixpoints and induction use (cyclic) implication and are in general longer and more complicated.

In lemma (8.22) we saw that the lattice of F-inductive types is the lattice of post-fixed points of F taken from the lattice of pers below μF and that the lub is the same as the lub of the per-lattice. This last statement is another way of saying that the identity function from F-inductive types to pers below μF distributes over lubs. This means that the identity function has an upper adjoint, mapping pers below μF to F-inductive types. We use this to define a new operator:

Definition 9.10:(F-Inductive Closure)

The F*-inductive closure* operator $\lfloor \_ \rfloor$ is defined by the following Galois connection: for all F-inductive types A and all pers B, $B \lhd \mu F$:

$$A \lhd B \quad\equiv\quad A \lhd \lfloor B \rfloor_F$$

The lhs inclusion is in the lattice of pers below μF, the rhs inclusion is in the lattice of F-inductive types. The F subscript is usually omitted if the relator is clear from the context.
□

This is the second example of a closure operator defined using a Galois connection. The transitive closure is defined as a lower adjoint of the (identity) embedding of transitive specs in the standard spec-lattice. Closure operators that are defined as an upper adjoint of an embedding are sometimes called kernels [28]. The type with equations construction can now be expressed using the F-inductive closure:

Lemma 9.11:

For per P, $P \lhd \mu F$, and F-inductive type B:

$$\nu(X \mapsto F.X \wedge P \wedge B) \;=\; \lfloor B \wedge P \rfloor$$

□

The greatest fixed point is calculated in the lattice of pers below μF. The equality in the lemma is proved using indirect equality. Both $\nu(X \mapsto F.X \wedge P \wedge B)$ and $\lfloor B \wedge P \rfloor$ are F-inductive types and we assume that the X used below is also F-inductive to allow the substitution that is necessary for a proof by indirect equality:

$$\begin{array}{ll}
 & X \lhd \lfloor B \wedge P \rfloor \\
\equiv & \qquad \{ \ (9.10) \ \} \\
 & X \lhd B \wedge P \\
\Leftarrow & \qquad \{ \ \text{monotonicity} \ \} \\
 & X \lhd F.\nu(X \mapsto F.X \wedge P \wedge B) \wedge B \wedge P \\
\equiv & \qquad \{ \ \text{calculation rule} \ \} \\
 & X \lhd \nu(X \mapsto F.X \wedge P \wedge B) \\
\Leftarrow & \qquad \{ \ \text{induction} \ \} \\
 & X \lhd F.X \wedge B \wedge P \\
\equiv & \qquad \{ \ X \text{ is } F\text{-inductive} \ \} \\
 & X \lhd B \wedge P
\end{array}$$

□

We continue with the investigation of the algebraic properties of the $\lfloor \_ \rfloor$ operation:

Lemma 9.12: (F**-Inductive Closure**)

$$\begin{array}{lrcl}
\text{(a)} & \lfloor A \rfloor & \lhd & F.\lfloor A \rfloor \\
\text{(b)} & \lfloor A \rfloor & \lhd & \mu F \\
\text{(c)} & \lfloor A \rfloor & \lhd & A \\
\text{(d)} & \lfloor A \rfloor \lhd \lfloor B \rfloor & \Leftarrow & A \lhd B \\
\text{(e)} & \lfloor A \rfloor = A & \Leftarrow & A \lhd F.A \quad \wedge \quad A \lhd \mu F \\
\text{(f)} & \lfloor A \rfloor \subseteq \lfloor B \rfloor & \Leftarrow & A \subseteq B \\
\text{(g)} & \lfloor A \rfloor \mathrel{\lhd|} \lfloor B \rfloor \mathrel{\lhd|} \mu F & \Leftarrow & A \mathrel{\lhd|} B \mathrel{\lhd|} \mu F
\end{array}$$

□

Parts (9.12a) and (9.12b) state that the range of $\lfloor \_ \rfloor$ operator is the lattice of F-inductive types. Parts (9.12c) and (9.12d) are instantiations of cancellation and monotonicity properties for Galois connections. Part (9.12e) is the combination of (9.12c) with the other cancellation property. We prove the remaining parts, starting with (9.12f):

$$\begin{array}{ll}
 & \lfloor A \rfloor \subseteq \lfloor B \rfloor \\
\equiv & \qquad \{ \ (5.38c) \ \} \\
 & \lfloor B \rfloor \circ \lfloor A \rfloor^{\between} \lhd \lfloor A \rfloor \lhd \lfloor B \rfloor \\
\equiv & \qquad \{ \ \bullet\, A \subseteq B;\ A \subseteq B \ \Rightarrow \ A \lhd B,\ (9.12d) \ \} \\
 & \lfloor B \rfloor \circ \lfloor A \rfloor^{\between} \lhd \lfloor A \rfloor
\end{array}$$

$$
\begin{array}{ll}
\equiv & \qquad \{ \ \lfloor B\rfloor \circ \lfloor A\rfloor^{\cup} \text{ is an } F\text{-inductive type (see below), definition } \lfloor\_\rfloor \ \} \\
& \lfloor B\rfloor \circ \lfloor A\rfloor^{\cup} \lhd A \\
\equiv & \qquad \{ \ \bullet\ A \subseteq B;\ (5.14), \ \} \\
& \lfloor B\rfloor \circ \lfloor A\rfloor^{\cup} \ \lhd \ B \circ A^{\cup} \\
\Leftarrow & \qquad \{ \ (5.38b) \ \} \\
& \lfloor B\rfloor \lhd B \quad \wedge \quad \lfloor A\rfloor^{\cup} \lhd A^{\cup} \\
\equiv & \qquad \{ \ (9.12c), (5.38a) \ \} \\
& \mathit{true}
\end{array}
$$

We still have something to prove for the third step above:

$$
\begin{array}{ll}
& \lfloor B\rfloor \circ \lfloor A\rfloor^{\cup} \ \lhd \ F.(\lfloor B\rfloor \circ \lfloor A\rfloor^{\cup}) \quad \wedge \quad \lfloor B\rfloor \circ \lfloor A\rfloor^{\cup} \ \lhd \ \mu F \\
\equiv & \qquad \{ \ \text{relators, domains, } \lfloor B\rfloor \lhd \mu F,\ (5.38b) \ \} \\
& \lfloor B\rfloor \circ \lfloor A\rfloor^{\cup} \ \lhd \ F.\lfloor B\rfloor \circ F.I \circ (F.\lfloor A\rfloor)^{\cup} \\
\Leftarrow & \qquad \{ \ \text{relators, unit, } (5.38b) \ \} \\
& \lfloor B\rfloor \ \lhd \ F.\lfloor B\rfloor \quad \wedge \quad \lfloor A\rfloor^{\cup} \lhd (F.\lfloor A\rfloor)^{\cup} \\
\equiv & \qquad \{ \ (9.12a), (5.38a) \ \} \\
& \mathit{true}
\end{array}
$$

We finish with the proof of (9.12g):

$$
\begin{array}{ll}
& \lfloor A\rfloor \mathrel{\lhd\!\mid} \lfloor B\rfloor \mathrel{\lhd\!\mid} \mu F \\
\equiv & \qquad \{ \ (5.38g) \ \} \\
& \lfloor A\rfloor \ \lhd \ \lfloor B\rfloor \ \lhd \ \mu F \quad \wedge \quad \lfloor A\rfloor \mathrel{\lhd\!\mid} \mu F \\
\equiv & \qquad \{ \ (9.12e), (5.38f) \ \} \\
& \lfloor A\rfloor \lhd \lfloor B\rfloor \lhd \lfloor \mu F\rfloor \quad \wedge \quad \mu F \circ \top\!\!\top \circ \mu F \ \lhd \ \lfloor A\rfloor \ \lhd \ \mu F \\
\equiv & \qquad \{ \ \bullet\ A \lhd B \lhd \mu F;\ (9.12d), (9.12b) \ \} \\
& \mu F \circ \top\!\!\top \circ \mu F \ \lhd \ \lfloor A\rfloor \\
\equiv & \qquad \{ \ \mu F \circ \top\!\!\top \circ \mu F \text{ is an } F\text{-inductive type (see below), definition } \lfloor\_\rfloor \ \} \\
& \mu F \circ \top\!\!\top \circ \mu F \ \lhd \ A \\
\equiv & \qquad \{ \ \bullet\ A \mathrel{\lhd\!\mid} \mu F,\ (5.38f) \ \} \\
& \mathit{true}
\end{array}
$$

The penultimate step used that $\mu F \circ \top\!\!\top \circ \mu F$ is an F-inductive type. One part of the proof obligation for this claim, $\mu F \circ \top\!\!\top \circ \mu F \ \lhd \ \mu F$, is trivial and the other part, $\mu F \circ \top\!\!\top \circ \mu F \ \lhd \ F.(\mu F \circ \top\!\!\top \circ \mu F)$, is proved by:

$$
\begin{array}{ll}
& \mu F \circ \top\!\!\top \circ \mu F \\
\sqsupseteq & \qquad \{ \ \text{top, calculation rule} \ \} \\
& \mu F \ \circ \ \top\!\!\top \ \circ \ \mu F \ \circ \ F\,.\,\mu F \ \circ \ F.\top\!\!\top \ \circ \ F\,.\,\mu F \\
= & \qquad \{ \ \text{relators} \ \} \\
& \mu F \circ \top\!\!\top \circ \mu F \circ F.(\mu F \circ \top\!\!\top \circ \mu F) \\
\sqsupseteq & \qquad \{ \ \text{top, calculation rule, relators} \ \} \\
& \mu F \circ \top\!\!\top \circ F.(\mu F \circ \mu F \circ \mu F \circ \mu F) \\
\equiv & \qquad \{ \ \text{per } \mu F, \text{ calculation rule} \ \} \\
& \mu F \circ \top\!\!\top \circ \mu F
\end{array}
$$

□

9.2.2 F-inductive type construction

We continue with the investigation of the use of the F-inductive closure for the construction of F-inductive types. The first property that we prove is that adding laws is really a quotient-type construction. A per P constructed from law equations satisfies $P \lhd\!| \mu F$ and we would like, given an F-inductive type B, the type constructed by adding the laws to B, $\lfloor B_\wedge P \rfloor$, to form a quotient of B, i.e. $\lfloor B_\wedge P \rfloor \lhd\!| B$

Unfortunately this depends not only on the laws but also on the base type B. Consider the law represented by $P \;=\; \mu F \circ \top\!\!\top \circ \mu F$ (for a non-empty μF) which can be interpreted as imposing that all elements of the type are equal to each other. From $\lfloor B_\wedge P \rfloor \lhd B_\wedge P$ we have that $\lfloor B_\wedge P \rfloor \lhd P$, but P is a class and has only a single element. This means that $\lfloor B_\wedge P \rfloor \;=\; \perp\!\!\perp$ or $\lfloor B_\wedge P \rfloor \;=\; P$. Combined with $\lfloor B_\wedge P \rfloor \lhd\!| B$ we obtain $\perp\!\!\perp \lhd\!| B$ or $P \lhd\!| B$ which can be simplified to $B = \perp\!\!\perp$ or $B \lhd\!| \mu F$. The former, imposing laws on an empty type, is not very useful so we see that it is necessary to demand $B \lhd\!| \mu F$ if we want that imposing a law on B constructs a quotient of B. The interpretation of $B \lhd\!| \mu F$ is that B is a non-restricted type and all constructors of B are total.

The condition $B \lhd\!| \mu F$ is not only necessary but is also sufficient as is shown in the following lemma:

Lemma 9.13:(Quotient Construction)

For F-inductive type B:

$$P \lhd\!| \mu F \quad \wedge \quad B \lhd\!| \mu F \qquad \Rightarrow \qquad \lfloor B_\wedge P \rfloor \lhd\!| B$$

□

$$
\begin{array}{ll}
 & \lfloor B_\wedge P \rfloor \lhd\!| B \\
\equiv & \quad \{\ \bullet\ B \lhd\!| \mu F; B \text{ is an } F\text{-inductive type; (9.12e)}\ \} \\
 & \lfloor B_\wedge P \rfloor \lhd\!| \lfloor B \rfloor \lhd\!| \mu F \\
\Leftarrow & \quad \{\ \text{(9.12g)}\ \} \\
 & B_\wedge P \lhd\!| B \lhd\!| \mu F \\
\equiv & \quad \{\ \ B_\wedge P \lhd B \lhd \mu F,\ \text{(5.38g)}\ \} \\
 & B_\wedge P \lhd\!| \mu F \\
\equiv & \quad \{\ B_\wedge P \lhd \mu F,\ \text{(5.38e)}\ \} \\
 & (B_\wedge P)_{\ltimes} \;=\; (\mu F)_{\ltimes} \\
\Leftarrow & \quad \{\ \text{(5.36)}\ \} \\
 & B_{\ltimes} \;=\; P_{\ltimes} \;=\; (\mu F)_{\ltimes} \\
\equiv & \quad \{\ \bullet\ P \lhd\!| \mu F \quad \wedge \quad B \lhd\!| \mu F;\ \text{(5.38e)}\ \} \\
 & \mathit{true}
\end{array}
$$

□

This quotient construction is known in the literature as constructing a congruence relation. Inductive types $X \lhd\!| \mu F$ satisfy $X \;=\; X \circ F.X \;\sqsupseteq\; X_{\ltimes} \circ F.X \;=\; F.X$. Such a type X is sometimes referred to as an F-congruence.

There are no extra constraints on the base type for subtype construction:

Lemma 9.14:(Subtype Construction)

For F-inductive type B:

$$P \subseteq \mu F \;\Rightarrow\; \lfloor B_{\wedge} P \rfloor \subseteq B$$

□

$$
\begin{array}{ll}
 & \lfloor B_{\wedge} P \rfloor \;\subseteq\; B \\
\equiv & \qquad \{\; B \text{ is an } F\text{-inductive type, (9.12e)} \;\} \\
 & \lfloor B_{\wedge} P \rfloor \;\subseteq\; \lfloor B \rfloor \\
\Leftarrow & \qquad \{\; \text{(9.12f)} \;\} \\
 & B_{\wedge} P \;\subseteq\; B \\
\Leftarrow & \qquad \{\; B \lhd \mu F, \text{(5.38d)} \;\} \\
 & P \;\subseteq\; \mu F
\end{array}
$$

□

The construction of an F-inductive type will in practice often start with μF, the F-inductive type without any laws or restrictions, followed by adding the desired properties represented by pers. One important question is then whether the order in which the properties are added influences the end result. It would be desirable that the order is not important and this is indeed the case. We have the following lemma:

Lemma 9.15:(F-Inductive Closure)

$$\lfloor \lfloor A \rfloor_{\wedge} B \rfloor \;=\; \lfloor A_{\wedge} B \rfloor$$

□

Proof by indirect equality with F-inductive type X:

$$
\begin{array}{ll}
 & X \lhd \lfloor \lfloor A \rfloor_{\wedge} B \rfloor \\
\equiv & \qquad \{\; \text{definition } \lfloor \_ \rfloor \;\} \\
 & X \;\lhd\; \lfloor A \rfloor_{\wedge} B \\
\equiv & \qquad \{\; \text{glb} \;\} \\
 & X \lhd \lfloor A \rfloor \quad \wedge \quad X \lhd B \\
\equiv & \qquad \{\; \text{definition } \lfloor \_ \rfloor \;\} \\
 & X \lhd A \quad \wedge \quad X \lhd B \\
\equiv & \qquad \{\; \text{glb} \;\} \\
 & X \;\lhd\; A_{\wedge} B \\
\equiv & \qquad \{\; \text{definition } \lfloor \_ \rfloor \;\} \\
 & X \lhd \lfloor A_{\wedge} B \rfloor
\end{array}
$$

□

A consequence of this lemma is that a type constructed by sequentially adding properties, which would normally result in nested F-inductive closures, can be written as the F-inductive closure of the per-glb of the pers representing the properties. The per-glb is associative and commutative, so the order in which the properties are added does not influence the end result.

9.2.3 Example: Stacks

A well-known example of a type with a law and a restriction is the stack. We want to construct stacks over a given type A and have three constructors:

$$\begin{aligned} &\mathit{empty} : \mathit{Stack} \leftarrow\!\!\dashv \top\!\!\top \\ &\mathit{push} : \mathit{Stack} \leftarrow\!\!\dashv A\times\mathit{Stack} \\ &\mathit{pop} : \mathit{Stack} \leftarrow \mathit{Stack} \end{aligned}$$

There is one law on stacks:

$$\mathit{pop} \circ \mathit{push} \;=\; \gg \circ A\times\mathit{Stack}$$

This law states that popping an element after pushing an element leaves the stack unchanged. There is also a restriction:

$$\top\!\!\top \circ \mathit{push}\cup \;\sqsupseteq\; \top\!\!\top \circ \mathit{pop}$$

This restriction states that popping is only allowed on stacks that were constructed by pushing (non-empty stacks). The relator corresponding to *Stack* is

$$F.X \;=\; \top\!\!\top+(A\times X + X)$$

The equations above satisfy the conditions for law and restriction equations and the *Stack* type can be constructed using the F-inductive closure.

9.2.4 Solutions for structural induction

In this section we will combine the results of section (9.1) with the construction of F-inductive types via closures to find conditions under which there exists a solution X for the structural induction specification:

(9.16) $X \circ \lfloor A_\wedge B\rfloor \;=\; X \quad\wedge\quad X \circ \lfloor A_\wedge B\rfloor \;=\; R \circ F.X \circ \lfloor A_\wedge B\rfloor{>}$

We assume that $A \lhd \mu F$ (A encodes laws) and $B \subseteq \mu F$ (B encodes restrictions). From (9.4) and (9.14), $\lfloor A_\wedge B\rfloor \subseteq \lfloor A\rfloor$, it follows that we can use the solution to

(9.17) $X \circ \lfloor A\rfloor \;=\; X \quad\wedge\quad X \circ \lfloor A\rfloor \;=\; R \circ F.X \circ \lfloor A\rfloor{>}$,

restricted to $\lfloor A_\wedge B\rfloor$. The next question is then of course whether (9.17) has a solution. Here we can use lemma (9.5) because from (9.12g) we know $\lfloor A\rfloor \lhd \mu F$. Specification (9.17) is equivalent to:

(9.18) $X \circ \mu F \;=\; X \quad\wedge\quad X \circ \mu F \;=\; R \circ F.X \circ (\mu F){>} \quad\wedge\quad X \circ \lfloor A\rfloor \;=\; X$

The first two conjuncts have a unique solution, $(\!|R|\!)$, so the complete specification (9.18) has a solution iff $(\!|R|\!) \circ \lfloor A\rfloor \;=\; (\!|R|\!)$. This last condition can be simplified because

(9.19) $(\!|R|\!) \circ \lfloor A\rfloor \;=\; (\!|R|\!) \quad\equiv\quad (\!|R|\!) \circ A \;=\; (\!|R|\!)$

which is proved by:

$$\begin{aligned} & (\!|R|\!) \circ \lfloor A\rfloor \;=\; (\!|R|\!) \\ \equiv\quad & \qquad\{\ \text{domains, } \lfloor A\rfloor \text{ is a per}\ \} \end{aligned}$$

$$
\begin{array}{ll}
 & (\![R]\!)_{\succ} \ \lhd \ \lfloor A \rfloor \\
\equiv & \qquad \{ \ (\![R]\!)_{\succ} \text{ is an } F\text{-inductive type (8.24), definition } \lfloor \_ \rfloor \ \} \\
 & (\![R]\!)_{\succ} \ \lhd \ A \\
\equiv & \qquad \{ \ \text{domains, } \lfloor A \rfloor \text{ is a per } \} \\
 & (\![R]\!) \circ A \ = \ (\![R]\!)
\end{array}
$$

Summing up, the calculations above proved the following lemma:

Lemma 9.20:(Structural Induction)

For pers $A \lhd\!\!\cdot\ \mu F$ and $B \ \subseteq \ \mu F$, specification (9.16) has as solution $(\![R]\!) \circ \lfloor A_\wedge B \rfloor$ if $(\![R]\!) \circ A \ = \ (\![R]\!)$.
□

Per A is the per-glb of the pers corresponding to the laws that are used for the construction of $\lfloor A_\wedge B \rfloor$. Using

$$S \circ (C_\wedge D) \ = \ S \quad \equiv \quad S \circ C \ = \ S \quad \wedge \quad S \circ D \ = \ S,$$

the condition $(\![R]\!) \circ A \ = \ (\![R]\!)$ is equivalent to demanding that all constituents of A are right domains of $(\![R]\!)$. These constituents are normally constructed using the method described in (9.8) with B instantiated to μF. If A is constructed from functions $f,g : \mu F \longleftarrow C$ for some per C then, applying (9.8), we obtain

(9.21) $(\![R]\!) \circ A \ = \ (\![R]\!) \quad \equiv \quad (\![R]\!) \circ f \ = \ (\![R]\!) \circ g,$

i.e. the catamorphism has to satisfy the law equation. The condition for having a solution to (9.16) has been simplified to $(\![R]\!)$ satisfying the equations for the law part of $\lfloor A_\wedge B \rfloor$ but this is still a condition expressed in terms of $(\![R]\!)$ and not directly in R itself. We will show later that a simple condition in terms of R is possible for some common laws like commutativity, associativity and unit laws.

9.2.5 Parameterised types with equations

At the end of section (8.3) a condition was given under which an inductive type could be parameterised. In this section we investigate conditions under which an inductive type constructed using equations can be parameterised. As before we assume that we have a binary relator $\otimes$ and abbreviate $\mu(R\otimes)$ to ϖR.

For a given parameter type A we can construct an $A\otimes$-inductive type $\lfloor P.A \rfloor_{A\otimes}$ (the subscript denotes the relator that is used for the closure) and we would like to extend this type construction to a relator. We write $P.A$ here, instead of a fixed per, because the per representing the equations can also depend on the parameter type A. The relator construction is possible if the following two conditions are satisfied:

(9.22) $\lfloor P.I \rfloor_{I\otimes} : \varpi \ \dot{\leftrightsquigarrow} \ \varpi$
(9.23) $\lfloor P.A \rfloor_{A\otimes} \ = \ \lfloor P.I \rfloor_{I\otimes} \circ \varpi A$, for all pers A

The constructed relator is the subrelator $\varpi_{\lfloor P.I \rfloor_{I\otimes}}$ and $\varpi_{\lfloor P.I \rfloor_{I\otimes}} A \ = \ \lfloor P.A \rfloor_{A\otimes}$ for all pers A. Using lemma (5.35) and assuming (9.22) we have that

$\lfloor P.I \rfloor_{I\otimes} \circ \varpi A \;=\; \lfloor P.I \rfloor_{I\otimes} \wedge \varpi A$. This is used in the following proof by indirect equality of (9.23) where X is an $A\otimes$-inductive type:

$$
\begin{array}{ll}
 & X \lhd \lfloor P.A \rfloor_{A\otimes} \\
\equiv & \quad \{ \; X \; A\otimes\text{-inductive type, definition } \lfloor \_ \rfloor \; \} \\
 & X \;\lhd\; P.A \\
\equiv & \quad \{ \; \bullet \; P.A \;=\; P.I \wedge \varpi A \; \} \\
 & X \;\;\lhd\;\; P.I \wedge \varpi A \\
\equiv & \quad \{ \; \text{glb, } X \; A\otimes\text{-inductive type} \Rightarrow X \; I\otimes\text{-inductive type (see below),} \\
 & \qquad \text{definition } \lfloor \_ \rfloor \; \} \\
 & X \;\;\lhd\;\; \lfloor P.I \rfloor_{I\otimes} \wedge \varpi A
\end{array}
$$

The fact that X is an $A\otimes$-inductive type $\Rightarrow$ X is an $I\otimes$-inductive type follows from $X \lhd \varpi A \lhd \varpi I$ and $X \lhd A\otimes X \lhd I\otimes X$. We also have to check that $\lfloor P.I \rfloor_{I\otimes} \wedge \varpi A$ is an $A\otimes$-inductive type. Part $\lfloor P.I \rfloor_{I\otimes} \wedge \varpi A \;\;\lhd\;\; \varpi A$ of this proof obligation is trivial. We prove $\lfloor P.I \rfloor_{I\otimes} \circ \varpi A \;\;\lhd\;\; A\otimes(\lfloor P.I \rfloor_{I\otimes} \circ \varpi A)$:

$$
\begin{array}{ll}
 & A\otimes(\lfloor P.I \rfloor_{I\otimes} \circ \varpi A) \\
= & \quad \{ \; \text{relators} \; \} \\
 & I\otimes\lfloor P.I \rfloor_{I\otimes} \;\circ\; A \otimes \varpi A \\
\rhd & \quad \{ \; (5.38b), \; \lfloor P.I \rfloor_{I\otimes} \lhd I\otimes\lfloor P.I \rfloor_{I\otimes}, \; A \otimes \varpi A \;=\; \varpi A \; \} \\
 & \lfloor P.I \rfloor_{I\otimes} \circ \varpi A
\end{array}
$$

The calculations above introduced a new condition:

(9.24) $P.A \;=\; P.I \wedge \varpi A$, for all pers A

and we still have to prove (9.22):

$$
\begin{array}{ll}
 & \forall (R \;\vdots\vdots\; \varpi R \circ \lfloor P.I \rfloor_{I\otimes} \;=\; \lfloor P.I \rfloor_{I\otimes} \circ \varpi R) \\
\equiv & \quad \{ \; \text{transitivity pers} \; \} \\
 & \forall (R \;\vdots\vdots\; \varpi R \circ \lfloor P.I \rfloor_{I\otimes} \;=\; \lfloor P.I \rfloor_{I\otimes} \circ \varpi R \circ \lfloor P.I \rfloor_{I\otimes} \;=\; \lfloor P.I \rfloor_{I\otimes} \circ \varpi R) \\
\equiv & \quad \{ \; \text{reverse} \; \} \\
 & \forall (R \;\vdots\vdots\; \lfloor P.I \rfloor_{I\otimes} \circ \varpi R \circ \lfloor P.I \rfloor_{I\otimes} \;=\; \lfloor P.I \rfloor_{I\otimes} \circ \varpi R) \\
\equiv & \quad \{ \; \lfloor P.I \rfloor_{I\otimes} \circ \varpi R \;=\; (\!|\lfloor P.I \rfloor_{I\otimes} \circ R\otimes I|\!) \text{ (see below), (9.19)} \; \} \\
 & \forall (R \;\vdots\vdots\; \lfloor P.I \rfloor_{I\otimes} \circ \varpi R \circ P.I \;=\; \lfloor P.I \rfloor_{I\otimes} \circ \varpi R) \\
\equiv & \quad \{ \; \bullet \; P.I : \varpi \;\stackrel{\cdot}{\leadsto}\; \varpi \; \} \\
 & \forall (R \;\vdots\vdots\; \lfloor P.I \rfloor_{I\otimes} \circ P.I \circ \varpi R \;=\; \lfloor P.I \rfloor_{I\otimes} \circ \varpi R) \\
\equiv & \quad \{ \; \lfloor P.I \rfloor_{I\otimes} \lhd P.I \; \} \\
 & \mathit{true}
\end{array}
$$

The third step in the proof above used that $\lfloor P.I \rfloor_{I\otimes} \circ \varpi R$ can be written as a catamorphism. This is proved by:

$$
\begin{array}{ll}
 & \lfloor P.I \rfloor_{I\otimes} \circ \varpi R \;=\; (\!|\lfloor P.I \rfloor_{I\otimes} \circ R\otimes I|\!) \\
\equiv & \quad \{ \; \text{UEP catamorphism} \; \}
\end{array}
$$

$$
\begin{array}{ll}
 & \lfloor P.I \rfloor_{I\otimes} \circ \varpi R \;=\; \lfloor P.I \rfloor_{I\otimes} \circ R{\otimes}I \circ I{\otimes}(\lfloor P.I \rfloor_{I\otimes} \circ \varpi R) \circ \varpi I \\
\equiv & \qquad \{ \text{ relators } \} \\
 & \lfloor P.I \rfloor_{I\otimes} \circ \varpi R \circ \varpi I \;=\; \lfloor P.I \rfloor_{I\otimes} \circ I{\otimes}\lfloor P.I \rfloor_{I\otimes} \circ R \otimes \varpi R \circ \varpi I \\
\equiv & \qquad \{ \lfloor P.I \rfloor_{I\otimes} \triangleleft I{\otimes}\lfloor P.I \rfloor_{I\otimes},\, R \otimes \varpi R \;=\; \varpi R \} \\
 & \mathit{true}
\end{array}
$$

The conditions for the construction of a relator have been reduced to conditions on P, condition (9.24) and an extra condition introduced in the last proof, $P.I : \varpi \stackrel{\cdot}{\leftrightsquigarrow} \varpi$. We proved the following lemma:

Lemma 9.25:

If $P.A \;=\; P.I \wedge \varpi A$ for all pers A and $P.I : \varpi \stackrel{\cdot}{\leftrightsquigarrow} \varpi$ then $\lfloor P.A \rfloor_{A\otimes} \;=\; \varpi_{\lfloor P.I \rfloor_{I\otimes}} A$ for all pers A.

□

The next question is how to establish these conditions for a P corresponding to a collection of equations for laws and restrictions. Such a P is normally the per-glb of the pers corresponding to the different equations. The conditions on P in (9.25) are compositional, i.e. if the components of P satisfy the condition, then P also satisfies the condition. We prove this for a P that is the per-glb of two components but this proof can easily be generalised to more components. For the first condition in (9.25) we prove, supposing $P.A \;=\; Q.A \wedge R.A$:

$$
\begin{array}{ll}
 & P.A \\
= & \qquad \{ \text{ assumption } \} \\
 & Q.A \wedge R.A \\
= & \qquad \{ \bullet\, Q.A \;=\; Q.I \wedge \varpi A,\, R.A \;=\; R.I \wedge \varpi A \} \\
 & Q.I \wedge \varpi A \wedge R.I \wedge \varpi A \\
= & \qquad \{ \text{ glb } \} \\
 & Q.I \wedge R.I \wedge \varpi A \\
= & \qquad \{ \text{ assumption } \} \\
 & P.I \wedge \varpi A
\end{array}
$$

The second condition of (9.25) is a naturality condition. The preservation of the naturality properties of the components of a per-glb is a consequence of the following lemma:

Lemma 9.26: (Naturality)

$$
A,B : F \stackrel{\cdot}{\leftrightsquigarrow} F \quad \Rightarrow \quad A{\wedge}B : F \stackrel{\cdot}{\leftrightsquigarrow} F
$$

□

Proof:

$$
\begin{array}{ll}
 & \forall(R :: F.R \circ (A{\wedge}B) \;=\; (A{\wedge}B) \circ F.R) \\
\equiv & \qquad \{ \text{ transitivity pers } \} \\
 & \forall(R :: F.R \circ (A{\wedge}B) \;=\; (A{\wedge}B) \circ F.R \circ (A{\wedge}B) \;=\; (A{\wedge}B) \circ F.R)
\end{array}
$$

$$
\begin{array}{ll}
\equiv & \quad \{ \text{ reverse } \} \\
& \forall(R \mathrel{¦¦} (A \wedge B) \circ F.R \circ (A \wedge B) \ = \ (A \wedge B) \circ F.R) \\
\equiv & \quad \{ \text{ domains, per-glb } \} \\
& \forall(R \mathrel{¦¦} (A \wedge B) \circ F.R \circ A = (A \wedge B) \circ F.R \ \wedge \ (A \wedge B) \circ F.R \circ B = (A \wedge B) \circ F.R) \\
\equiv & \quad \{ \ \bullet \ A,B : F \ \dot{\leftrightsquigarrow} \ F \ \} \\
& \forall(R \mathrel{¦¦} (A \wedge B) \circ A \circ F.R = (A \wedge B) \circ F.R \ \wedge \ (A \wedge B) \circ B \circ F.R = (A \wedge B) \circ F.R) \\
\equiv & \quad \{ \ A \wedge B \ \triangleleft \ A, \ A \wedge B \ \triangleleft \ B \ \} \\
& \mathit{true}
\end{array}
$$

□

The next task is to find conditions on the equations such that the constituent per P satisfies $P.A \ = \ P.I \wedge \varpi A$ for all pers A and $P.I : \varpi \ \dot{\leftrightsquigarrow} \ \varpi$. We start with the per corresponding to the law equation.

The functions f and g in the equation depend on the parameter type and this is made explicit by writing $f.A$ and $g.A$. From $P.I : \varpi \ \dot{\leftrightsquigarrow} \ \varpi$ it follows using (5.35) that $P.I \wedge \varpi A \ = \ \varpi A \circ P.I$ and we are looking for conditions on $f.A$ and $g.A$ ensuring that $P.A \ = \ \varpi A \circ P.I$. A sufficient condition is given in the following lemma:

Lemma 9.27:

For all pers A:

$$
\begin{array}{ll}
& f.A \ = \ \varpi A \circ f.I \quad \wedge \quad g.A \ = \ \varpi A \circ g.I \quad \wedge \quad f.I \circ (g.I)^{\cup} : \varpi \ \dot{\leftrightsquigarrow} \ \varpi \\
\Rightarrow & \\
& \varpi A \sqcup (f.A \circ (g.A)^{\cup} \ \sqcup \ g.A \circ (f.A)^{\cup})^{+} = \\
& \varpi A \circ (\varpi I \sqcup (f.I \circ (g.I)^{\cup} \ \sqcup \ g.I \circ (f.I)^{\cup})^{+})
\end{array}
$$

□

$$
\begin{array}{ll}
& \varpi A \sqcup (f.A \circ (g.A)^{\cup} \ \sqcup \ g.A \circ (f.A)^{\cup})^{+} \\
= & \quad \{ \ \bullet \ f.A \ = \ \varpi A \circ f.I \quad \wedge \quad g.A \ = \ \varpi A \circ g.I \ \} \\
& \varpi A \sqcup (\varpi A \circ f.I \circ (g.I)^{\cup} \circ \varpi A \ \sqcup \ \varpi A \circ g.I \circ (f.I)^{\cup} \circ \varpi A)^{+} \\
= & \quad \{ \text{ distributivity } \} \\
& \varpi A \sqcup (\varpi A \circ (f.I \circ (g.I)^{\cup} \ \sqcup \ g.I \circ (f.I)^{\cup}) \circ \varpi A)^{+} \\
= & \quad \{ \ \bullet \ f.I \circ (g.I)^{\cup} : \varpi \ \dot{\leftrightsquigarrow} \ \varpi, \ \varpi A \circ \varpi A \ = \ \varpi A \ \} \\
& \varpi A \sqcup (\varpi A \circ (f.I \circ (g.I)^{\cup} \ \sqcup \ g.I \circ (f.I)^{\cup}))^{+} \\
= & \quad \{ \ \bullet \ f.I \circ (g.I)^{\cup} : \varpi \ \dot{\leftrightsquigarrow} \ \varpi, \ \varpi A \circ \varpi A \ = \ \varpi A; \ (9.28) \ \} \\
& \varpi A \ \sqcup \ \varpi A \circ (f.I \circ (g.I)^{\cup} \ \sqcup \ g.I \circ (f.I)^{\cup})^{+} \\
= & \quad \{ \text{ distribution, relators } \} \\
& \varpi A \circ (\varpi I \sqcup (f.I \circ (g.I)^{\cup} \ \sqcup \ g.I \circ (f.I)^{\cup})^{+})
\end{array}
$$

□

The proof above used the following lemma about transitive closures:

Lemma 9.28:

$$A \circ R \ = \ A \circ R \circ A \quad \Rightarrow \quad A \circ R^{+} \ = \ (A \circ R)^{+}$$

□

The transitive closure of a spec is defined using a Galois connection in this thesis but an equivalent definition can be given using a least fixpoint: $S^+ = \mu(X \mapsto S \sqcup S \circ X)$. The equivalence of both definitions is not difficult to prove and left to the interested reader. We prove the lemma by:

$$
\begin{array}{ll}
 & A \circ R^+ = (A \circ R)^+ \\
\Leftarrow & \quad \{ \ \mu\text{-fusion, } S^+ = \mu(X \mapsto S \sqcup S \circ X) \ \} \\
 & \forall(X \;\vdots\vdots\; A \circ (R \sqcup R \circ X) = A \circ R \sqcup A \circ R \circ A \circ X) \\
\Leftarrow & \quad \{ \ \text{distributivity} \ \} \\
 & A \circ R = A \circ R \circ A
\end{array}
$$

□

We still have to prove the naturality condition $P.I : \varpi \stackrel{\cdot}{\leftrightsquigarrow} \varpi$. This is done in the following calculation:

$$
\begin{array}{ll}
 & \varpi I \sqcup (f.I \circ (g.I)^{\cup} \sqcup g.I \circ (f.I)^{\cup})^+ : \varpi \stackrel{\cdot}{\leftrightsquigarrow} \varpi \\
\Leftarrow & \quad \{ \ \text{naturality} \ \} \\
 & \varpi I : \varpi \stackrel{\cdot}{\leftrightsquigarrow} \varpi \quad \wedge \quad (f.I \circ (g.I)^{\cup} \sqcup g.I \circ (f.I)^{\cup})^+ : \varpi \stackrel{\cdot}{\leftrightsquigarrow} \varpi \\
\Leftarrow & \quad \{ \ \text{relators, naturality (see below)} \ \} \\
 & f.I \circ (g.I)^{\cup} \sqcup g.I \circ (f.I)^{\cup} : \varpi \stackrel{\cdot}{\leftrightsquigarrow} \varpi \\
\Leftarrow & \quad \{ \ \text{naturality, reverse} \ \} \\
 & f.I \circ (g.I)^{\cup} : \varpi \stackrel{\cdot}{\leftrightsquigarrow} \varpi
\end{array}
$$

The condition here was already a condition in (9.27). The second step of the proof above uses

$$R : F \stackrel{\cdot}{\leftrightsquigarrow} F \quad \Rightarrow \quad R^+ : F \stackrel{\cdot}{\leftrightsquigarrow} F$$

This result is the combination of the following two naturality properties:

$$
\begin{array}{lcl}
R : F \stackrel{\cdot}{\leftsquigarrow} F & \Rightarrow & R^+ : F \stackrel{\cdot}{\leftsquigarrow} F \\
R : F \stackrel{\cdot}{\rightsquigarrow} F & \Rightarrow & R^+ : F \stackrel{\cdot}{\rightsquigarrow} F
\end{array}
$$

We only prove the first one, the second one follows by symmetry:

$$
\begin{array}{ll}
 & F.S \circ R^+ \sqsupseteq R^+ \circ F.S \\
\equiv & \quad \{ \ \text{factors} \ \} \\
 & (F.S \circ R^+) \,/\, F.S \sqsupseteq R^+ \\
\equiv & \quad \{ \ \text{transitive closure, } (F.S \circ R^+) \,/\, F.S \text{ is transitive (see below)} \ \} \\
 & (F.S \circ R^+) \,/\, F.S \sqsupseteq R \\
\equiv & \quad \{ \ \text{factors} \ \} \\
 & F.S \circ R^+ \sqsupseteq R \circ F.S \\
\Leftarrow & \quad \{ \ R^+ \sqsupseteq R \ \} \\
 & F.S \circ R \sqsupseteq R \circ F.S
\end{array}
$$

The transitivity of $(F.S \circ R^+) \,/\, F.S$ is proved by

$$
\begin{array}{ll}
 & (F.S \circ R^+) \;/\; F.S \;\;\sqsupseteq\;\; (F.S \circ R^+) \;/\; F.S \;\circ\; (F.S \circ R^+) \;/\; F.S \\
\Leftarrow & \quad \{ \text{ factors, cancellation } \} \\
 & F.S \circ R^+ \;\;\sqsupseteq\;\; (F.S \circ R^+) \;/\; F.S \;\circ\; F.S \;\circ\; R^+ \\
\Leftarrow & \quad \{ \text{ cancellation } \} \\
 & F.S \circ R^+ \;\sqsupseteq\; F.S \circ R^+ \circ R^+ \\
\equiv & \quad \{ \; R^+ \;\sqsupseteq\; R^+ \circ R^+ \; \} \\
 & \mathit{true}
\end{array}
$$

□

Summing up, we obtained as conditions for the parameterisability of types constructed by a law equation specified with functions $f.A$ and $g.A$ that it is sufficient that:

$$f.A \;=\; \varpi A \circ f.I \quad \wedge \quad g.A \;=\; \varpi A \circ g.I \quad \wedge \quad f.I \circ (g.I)^\cup : \varpi \;\dot{\leftrightarrow}\; \varpi$$

The last condition is usually the result of separate naturality properties of $f.I$ and $g.I$:

(9.29) $f.A \;=\; \varpi A \circ f.I \quad \wedge \quad g.A \;=\; \varpi A \circ g.I \quad \wedge \quad f.I, g.I : \varpi \;\dot{\leftrightarrow}\; F$, for some relator F.

The associativity law for non-empty join-lists is an example of a law where the equation satisfies condition (9.29). The relator for this example is $A{\otimes}X \;=\; A + X{\times}X$ and ϖA has as constructors $\mathit{leaf}.A : \varpi A \leftarrow A$ and $\mathit{join}.A : \varpi A \;\leftarrow\; \varpi A \times \varpi A$. The functions for the associativity equation are:

$$f.A \;\;=\;\; \mathit{join}.A \;\circ\; \mathit{join}.A \times I \;\circ\; \mathsf{assoc} \quad \wedge \quad g.A \;\;=\;\; \mathit{join}.A \;\circ\; I \times \mathit{join}.A$$

We only check the conditions in (9.29) for $f.A$, the proofs for $g.A$ being similar. The constructor $\mathit{join}.A$ can be written as $\varpi A \circ \hookleftarrow$. This is used in the following proof of $f.A \;=\; \varpi A \circ f.I$:

$$
\begin{array}{ll}
 & \mathit{join}.A \;\circ\; \mathit{join}.A \times I \;\circ\; \mathsf{assoc} \\
= & \quad \{ \; \mathit{join}.A \;=\; \varpi A \circ \hookleftarrow \; \} \\
 & \varpi A \circ \hookleftarrow \circ (\varpi A \circ \hookleftarrow){\times}I \circ \mathsf{assoc} \\
= & \quad \{ \; \varpi A \;=\; \varpi A \circ \varpi A, \text{ calculation rule, definition } \varpi \; \} \\
 & \varpi A \;\circ\; A \,+\, \varpi A \times \varpi A \;\circ\; \hookleftarrow \;\circ\; (\varpi A \circ \hookleftarrow){\times}I \;\circ\; \mathsf{assoc} \\
= & \quad \{ \text{ naturality } \hookleftarrow \; \} \\
 & \varpi A \;\circ\; \hookleftarrow \;\circ\; \varpi A \times \varpi A \;\circ\; (\varpi A \circ \hookleftarrow){\times}I \;\circ\; \mathsf{assoc} \\
= & \quad \{ \text{ relators, } \varpi A \circ \varpi A \;=\; \varpi A \circ \varpi I \; \} \\
 & \varpi A \;\circ\; \hookleftarrow \;\circ\; \varpi A \times \varpi A \;\circ\; (\varpi I \circ \hookleftarrow){\times}I \;\circ\; \mathsf{assoc} \\
= & \quad \{ \text{ naturality } \hookleftarrow \; \} \\
 & \varpi A \;\circ\; A \,+\, \varpi A \times \varpi A \;\circ\; \hookleftarrow \;\circ\; (\varpi I \circ \hookleftarrow){\times}I \;\circ\; \mathsf{assoc} \\
= & \quad \{ \text{ calculation rule, definition } \varpi, \; \varpi A \circ \varpi A \;=\; \varpi A \circ \varpi I \; \} \\
 & \varpi A \circ \varpi I \circ \hookleftarrow \circ (\varpi I \circ \hookleftarrow){\times}I \circ \mathsf{assoc} \\
= & \quad \{ \; \mathit{join}.I \;=\; \varpi I \circ \hookleftarrow \; \} \\
 & \varpi A \;\circ\; \mathit{join}.I \;\circ\; \mathit{join}.I \times I \;\circ\; \mathsf{assoc}
\end{array}
$$

The other condition, $f.I : \varpi \;\dot{\leftrightarrow}\; F$, is proved for relator $F.R \;=\; \varpi R \times (\varpi R \times \varpi R)$:

$$
\begin{array}{ll}
 & \varpi R \circ \varpi I \circ \hookleftarrow \circ (\varpi I \circ \hookleftarrow) \times I \circ \mathsf{assoc} \\
= & \qquad \{ \text{ relators, calculation rule } \} \\
 & \varpi I \ \circ \ R \ + \ \varpi R \times \varpi R \ \circ \ \hookleftarrow \ \circ \ (\varpi I \circ \hookleftarrow) \times I \ \circ \ \mathsf{assoc} \\
= & \qquad \{ \text{ naturality } \hookleftarrow \ \} \\
 & \varpi I \ \circ \ \hookleftarrow \ \circ \ \varpi R \times \varpi R \ \circ \ (\varpi I \circ \hookleftarrow) \times I \ \circ \ \mathsf{assoc} \\
= & \qquad \{ \text{ relators, calculation rule } \} \\
 & \varpi I \ \circ \ \hookleftarrow \ \circ \ \varpi I \times I \ \circ \ (R \ + \ \varpi R \times \varpi R \ \circ \ \hookleftarrow) \times \varpi R \ \circ \ \mathsf{assoc} \\
= & \qquad \{ \text{ naturality } \hookleftarrow\text{, relators } \} \\
 & \varpi I \ \circ \ \hookleftarrow \ \circ \ (\varpi I \circ \hookleftarrow) \times I \ \circ \ (\varpi R \times \varpi R) \times \varpi R \ \circ \ \mathsf{assoc} \\
= & \qquad \{ \text{ naturality assoc } \} \\
 & \varpi I \ \circ \ \hookleftarrow \ \circ \ (\varpi I \circ \hookleftarrow) \times I \ \circ \ \mathsf{assoc} \ \circ \ \varpi R \times (\varpi R \times \varpi R)
\end{array}
$$

So we can conclude that the type of non-empty join-lists is indeed parameterisable.

The next problem that we want to solve is finding conditions on the restriction equation such that the corresponding per

$$P.A \ = \ \varpi A \ \sqcap \ (f.A)^{\cup} \circ R.A$$

satisfies $P.A \ = \ P.I \wedge \varpi A$ for all pers A and $P.I : \varpi \leftrightsquigarrow \varpi$. The following lemma can be used for the naturality requirement (instantiate with ϖ for F and $(f.I)^{\cup} \circ R.I$ for S):

Lemma 9.30:(Naturality)

$$S : F.\top\!\!\top \ — \ F.\top\!\!\top \qquad \Rightarrow \qquad F.I \sqcap S : F \leftrightsquigarrow F$$

□

Proof:

$$
\begin{array}{ll}
 & (F.I \sqcap S) \circ F.X \\
= & \qquad \{ \ S : F.\top\!\!\top \ — \ F.\top\!\!\top \text{, distribution, relators } \} \\
 & F.X \ \sqcap \ F.\top\!\!\top \circ S \circ F.(\top\!\!\top \circ X) \\
= & \qquad \{ \text{ domains } \} \\
 & F.X \ \sqcap \ F.\top\!\!\top \circ S \circ F.\top\!\!\top \circ (F.X)> \\
= & \qquad \{ \text{ domains } \} \\
 & F.X \ \sqcap \ (F.X)< \circ F.\top\!\!\top \circ S \circ F.\top\!\!\top \\
= & \qquad \{ \text{ domains } \} \\
 & F.X \ \sqcap \ F.(X \circ \top\!\!\top) \circ S \circ F.\top\!\!\top \\
= & \qquad \{ \ S : F.\top\!\!\top \ — \ F.\top\!\!\top \text{, distribution, relators } \} \\
 & F.X \circ (F.I \sqcap S)
\end{array}
$$

□

So we obtain as condition $(f.I)^{\cup} \circ R.I : \varpi\top\!\!\top \ — \ \varpi\top\!\!\top$. From $P.I : \varpi \leftrightsquigarrow \varpi$ and the fact that ϖA is a per follows that the condition $P.A \ = \ P.I \wedge \varpi A$ is equivalent to $P.A \ = \ \varpi A \circ P.I \circ \varpi A$. We calculate:

$$
\begin{array}{ll}
 & \varpi A \circ (\varpi I \ \sqcap \ (f.I)^{\cup} \circ R.I) \circ \varpi A \\
= & \quad \{ \text{ distribution, } (f.I)^{\cup} \circ R.I : \varpi\top\!\!\top \leftarrow \varpi\top\!\!\top,\ \varpi\top\!\!\top \ \sqsupseteq \ \varpi\top\!\!\top \circ \varpi A \ \} \\
 & \varpi A \circ \varpi I \circ \varpi A \ \sqcap \ \varpi A \circ (f.I)^{\cup} \circ R.I \circ \varpi A \\
= & \quad \{ \text{ relators, per } A \ \} \\
 & \varpi A \ \sqcap \ \varpi A \circ (f.I)^{\cup} \circ R.I \circ \varpi A \\
= & \quad \{ \ \bullet\ f.A \ = \ f.I \circ \varpi A,\ R.A \ = \ R.I \circ \varpi A \ \} \\
 & \varpi A \ \sqcap \ (f.A)^{\cup} \circ R.A
\end{array}
$$

The condition $(f.I)^{\cup} \circ R.I : \varpi\top\!\!\top \leftarrow \varpi\top\!\!\top$ is normally obtained using type deduction from $R.I, f.I : \leftarrow \varpi\top\!\!\top$. Combining the calculations above, we have derived as condition for the parameterisability of the subset type construction:

(9.31) $f.A \ = \ f.I \circ \varpi A \quad \wedge \quad R.A \ = \ R.I \circ \varpi A \quad \wedge \quad R.I, f.I : \leftarrow \varpi\top\!\!\top$

Height-balanced trees provide an example of a type where the equation satisfies these conditions. The specs in the equation are

$$
\begin{array}{lll}
f.A & = & 0^{\bullet} \triangledown (1+ \circ \mathit{height}.A \circ \ll) \ \circ \ \varpi A \text{ and} \\
R.A & = & ((1+) \sqcup \mathbb{N} \sqcup (1+)^{\cup}) \ \circ \ 0^{\bullet} \triangledown (1+ \circ \mathit{height}.A \circ \gg) \ \circ \ \varpi A,
\end{array}
$$

where *height* is defined by

$$\mathit{height}.A \quad \triangleq \quad (\!| A\otimes ; 0^{\bullet} \ \triangledown \ (1+ \circ \mathit{max}) |\!).$$

The first step in the proof that the conditions in (9.31) are satisfied is showing that

$$\mathit{height}.A \ = \ \mathit{height}.I \circ \varpi A:$$

$$
\begin{array}{ll}
 & (\!| A\otimes ; 0^{\bullet} \ \triangledown \ (1+ \circ \mathit{max}) |\!) \ = \ (\!| I\otimes ; 0^{\bullet} \ \triangledown \ (1+ \circ \mathit{max}) |\!) \circ \varpi A \\
\Leftarrow & \quad \{ \ \mu\text{-fusion, definition catamorphism } \ \} \\
 & \forall (X \ \because \ 0^{\bullet} \triangledown (1+ \circ \mathit{max}) \circ A\otimes (X \circ \varpi A) \ = \ 0^{\bullet} \triangledown (1+ \circ \mathit{max}) \circ I\otimes X \circ \varpi A) \\
\Leftarrow & \quad \{ \text{ monotonicity, relators } \} \\
 & \forall (X \ \because \ I\otimes X \ \circ \ A \otimes \varpi A \ = \ I\otimes X \circ \varpi A) \\
\equiv & \quad \{ \text{ calculation rule, definition } \varpi \ \} \\
 & \mathit{true}
\end{array}
$$

We only prove $f.A \ = \ f.I \circ \varpi A$, the proof of $R.A \ = \ R.I \circ \varpi A$ being similar:

$$
\begin{array}{ll}
 & 0^{\bullet} \triangledown (1+ \circ \mathit{height}.A \circ \ll) \ \circ \ \varpi A \\
= & \quad \{ \ \mathit{height}.A \ = \ \mathit{height}.I \circ \varpi A \ \} \\
 & 0^{\bullet} \triangledown (1+ \circ \mathit{height}.I \circ \varpi A \circ \ll) \ \circ \ \varpi A \\
= & \quad \{ \text{ products } \} \\
 & 0^{\bullet} \triangledown (1+ \ \circ \ \mathit{height}.I \ \circ \ \ll \ \circ \ \varpi A \times I) \ \circ \ \varpi A \\
= & \quad \{ \text{ fusion } \} \\
 & 0^{\bullet} \triangledown (1+ \circ \mathit{height}.I \circ \ll) \ \circ \ I \ + \ \varpi A \times I \ \circ \ \varpi A \\
= & \quad \{ \ \varpi A \ = \ A \ + \ \varpi A \times \varpi A,\ \varpi A \text{ is a per, relators } \} \\
 & 0^{\bullet} \triangledown (1+ \ \circ \ \mathit{height}.I \ \circ \ \ll) \ \circ \ \varpi I \ \circ \ \varpi A
\end{array}
$$

The proofs of $f.I : \leftarrow \varpi\top\!\!\top$ and $R.I : \leftarrow \varpi\top\!\!\top$ are also similar and we only do the proof for $f.I$:

$$
\begin{array}{ll}
& 0^\bullet \triangledown (1+ \circ \; height.I \; \circ \; \ll) \; \circ \; \varpi I \; \circ \; \varpi \top\!\!\top \\
= & \qquad \{ \text{ relators, calculation rule } \} \\
& 0^\bullet \triangledown (1+ \circ \; height.I \; \circ \; \ll) \; \circ \; \top\!\!\top \; + \; \varpi \top\!\!\top \times \varpi \top\!\!\top \; \circ \; \varpi I \\
= & \qquad \{ \text{ fusion, } 0^\bullet : \!-\!\!\!- \top\!\!\top \; \} \\
& 0^\bullet \triangledown (1+ \circ \; height.I \; \circ \; \ll \; \circ \; \varpi \top\!\!\top \times \varpi \top\!\!\top) \; \circ \; \varpi I \\
= & \qquad \{ \text{ products, relators } \} \\
& 0^\bullet \triangledown (1+ \circ \; height.I \; \circ \; \varpi \top\!\!\top \; \circ \; \ll \; \circ \; \varpi I \times \varpi I) \; \circ \; \varpi I \\
= & \qquad \{ \text{ fusion } \} \\
& 0^\bullet \triangledown (1+ \circ \; height.I \; \circ \; \varpi \top\!\!\top \; \circ \; \ll) \; \circ \; I + (\varpi I \times \varpi I) \; \circ \; \varpi I \\
= & \qquad \{ \text{ calculation rule, relators, } height.I \circ \varpi \top\!\!\top = height.I \text{ (see below) } \} \\
& 0^\bullet \triangledown (1+ \circ \; height.I \; \circ \; \ll) \; \circ \; \varpi I
\end{array}
$$

The last step used $height.I \;\circ\; \varpi \top\!\!\top \;=\; height.I$ which is proved by:

$$
\begin{array}{ll}
& (\![0^\bullet \triangledown (1+ \circ \, max)]\!) \;=\; (\![0^\bullet \triangledown (1+ \circ \, max)]\!) \circ \varpi \top\!\!\top \\
\Leftarrow & \qquad \{ \; \mu\text{-fusion, definition catamorphism } \} \\
& \forall (X \;\vdots\vdots\; 0^\bullet \triangledown (1+ \circ \, max) \circ I \otimes (X \circ \varpi \top\!\!\top) \;=\; 0^\bullet \triangledown (1+ \circ \, max) \circ I \otimes X \circ \varpi \top\!\!\top) \\
\equiv & \qquad \{ \text{ calculation rule, relators } \} \\
& \forall (X \;\vdots\vdots\; 0^\bullet \triangledown (1+ \circ \, max) \circ I \otimes (X \circ \varpi \top\!\!\top) = 0^\bullet \triangledown (1+ \circ \, max) \circ \top\!\!\top \otimes (X \circ \varpi \top\!\!\top)) \\
\Leftarrow & \qquad \{ \text{ definition } \otimes \text{, fusion, monotonicity } \} \\
& 0^\bullet \circ I \;=\; 0^\bullet \circ \top\!\!\top \\
\equiv & \qquad \{ \; 0^\bullet : \!-\!\!\!- \top\!\!\top \; \} \\
& true
\end{array}
$$

We can conclude that the type of height-balanced trees is indeed parameterisable.

9.2.6 Example: the Boom-hierarchy

The Boom-hierarchy is a hierarchy of types with laws that was first described by Hendrik Boom and later became prominent by its use as the basis types of the Bird-Meertens formalism [11, 12, 34, 35]. The types in the Boom-hierarchy are parameterised (we use A for the parameter and B for the carrier) and have three constructors:

- $[] : B \longleftarrow \top\!\!\top$, constructing an "empty" element.
- $\tau : B \longleftarrow A$, constructing a singleton element from an element of the parameter type.
- $+\!\!+ : B \longleftarrow B \times B$, joining two elements together.

The relator corresponding to these constructors is defined by

$$A \otimes X \quad \triangleq \quad \top\!\!\top + (A + X \times X)$$

We denote $\mu(R\otimes)$ by ϖR. For the base type of the hierarchy, the type without any laws, the constructors are defined as follows:

$$
\begin{array}{lcl}
[] & = & \varpi A \circ \hookrightarrow \\
\tau & = & \varpi A \circ \hookleftarrow \circ \hookrightarrow \\
+\!\!+ & = & \varpi A \circ \hookleftarrow \circ \hookleftarrow
\end{array}
$$

We will show that all types in the hierarchy except the last, finite sets, can be parameterised. The first type in the hierarchy is the free type. Every subsequent type is constructed by adding laws to the foregoing type of the hierarchy. The constructors of the types in the hierarchy are constructed by pre-composing the constructor of the free type with the new carrier.

The type *Tree* is made from the free type by adding laws making [] both a left and a right unit of $+\!\!+$:

$$
\begin{array}{lcl}
\mathit{Tree} \circ +\!\!+ \circ []\triangle I & = & \mathit{Tree} \circ \varpi A \\
\mathit{Tree} \circ +\!\!+ \circ I\triangle [] & = & \mathit{Tree} \circ \varpi A
\end{array}
$$

The functions $+\!\!+ \circ []\triangle I$, ϖA and $+\!\!+ \circ I\triangle[]$ all have type $\varpi A \longleftarrow \varpi A$ and, for $A = I$, naturality type $\varpi \overset{\cdot}{\leftrightarrow} \varpi$, so this type can be parameterised. We have

$$\mathit{Tree} = \lfloor \mathit{lunit} \wedge \mathit{runit} \rfloor_{A\otimes}$$

where per *lunit* is defined as in (9.8) from the functions $+\!\!+ \circ []\triangle I$ and ϖA and per *runit* is constructed from $+\!\!+ \circ I\triangle[]$ and ϖA

The third type in the hierarchy is made from the second by adding a law making the $+\!\!+$ operator associative. This is similar to the example with non-empty join-lists that was used in the beginning of this chapter, trees with an associative join and units are known as join-lists, here denoted by *List*. The equation for associativity law is the following:

$$\mathit{List} \circ +\!\!+ \circ +\!\!+ \times I \circ \mathsf{assoc} = \mathit{List} \circ +\!\!+ \circ I \times (+\!\!+)$$

Both $+\!\!+ \circ +\!\!+ \times I \circ \mathsf{assoc}$ and $+\!\!+ \circ I \times (+\!\!+)$ have type $\varpi A \longleftarrow \varpi A \times (\varpi A \times \varpi A)$ and, for $A = I$, naturality type $\varpi \overset{\cdot}{\leftrightarrow} \varpi \times (\varpi \times \varpi)$, so this type can also be parameterised. We have

$$\mathit{List} = \lfloor \mathit{lunit} \wedge \mathit{runit} \wedge \mathit{asso} \rfloor_{A\otimes}$$

where per *asso* is constructed from $+\!\!+ \circ +\!\!+ \times I \circ \mathsf{assoc}$ and $+\!\!+ \circ I \times (+\!\!+)$.

The fourth type in the hierarchy is the type of bags, constructed by adding a law making $+\!\!+$ commutative. This type is denoted by *Bag* and the equation for the law is

$$\mathit{Bag} \circ +\!\!+ \circ \mathsf{swap} = \mathit{Bag} \circ +\!\!+$$

Both $+\!\!+ \circ \mathsf{swap}$ and $+\!\!+$ have type $\varpi A \longleftarrow \varpi A \times \varpi A$ and, for $A = I$, naturality type $\varpi \overset{\cdot}{\leftrightarrow} \varpi \times \varpi$, so again we can parameterise. We obtain

$$\mathit{Bag} = \lfloor \mathit{lunit} \wedge \mathit{runit} \wedge \mathit{asso} \wedge \mathit{comm} \rfloor_{A\otimes}$$

where per *comm* is constructed from $+\!\!+ \circ \mathsf{swap}$ and $+\!\!+$.

The last type in the hierarchy is the type of finite sets, constructed by adding a law making $+\!\!+$ idempotent. We denote this type by *Set*. The equation is :

$$\mathit{Set} \circ +\!\!+ \circ I\triangle I = \mathit{Set} \circ \varpi A$$

The per-typing for $+\!\!+ \circ I \triangle I$ and ϖA is all right, both having type $\varpi A \leftarrow\!\dashv \varpi A$, but the naturality is a problem. We only have $\varpi \ \dot{\leftarrow\!\sim}\ \varpi$ for $+\!\!+ \circ I \triangle I$, so parameterising is not possible with the techniques that have been developed here. We still have

$$Set \;=\; \lfloor lunit \wedge runit \wedge asso \wedge comm \wedge idem \rfloor_{A\otimes}$$

where per *idem* is constructed from $+\!\!+ \circ I \triangle I$ and ϖA but it is not clear whether there exists a *Set* relator.

We obtain interesting results for conditions under which the types from the Boom-hierarchy are domains of catamorphisms. It turns out that if the arguments of the catamorphism obey equations similar to the ones for the constructors then the types with laws are domains. An $A\otimes$-catamorphism has as general shape $(\![R \triangledown (S \triangledown T)]\!)$ where $R : \text{—}\ \top\!\!\top$ gives the action to be taken on empty elements, $S : \text{—}\ A$ the action for singletons and $T : \text{—}\ I{\times}I$ the action for the join of two elements. We abbreviate $(\![R \triangledown (S \triangledown T)]\!)$ to $(\![\,]\!)$ to keep the expressions manageable. We start, using (9.21) for the left unit law:

$$
\begin{array}{cl}
 & (\![\,]\!) \circ +\!\!+ \circ [\,] \triangle I \;=\; (\![\,]\!) \circ \varpi A \\
\equiv & \qquad \{ \text{ catamorphisms, constructors } \} \\
 & T \circ (\![\,]\!) {\times} (\![\,]\!) \circ [\,] \triangle I \;=\; (\![\,]\!) \\
\equiv & \qquad \{ \text{ catamorphisms, constructors, Cartesian product } \} \\
 & T \circ (R \circ \top\!\!\top) \triangle (\![\,]\!) \;=\; (\![\,]\!) \\
\equiv & \qquad \{ \text{ distribution, } R : \text{—}\ \top\!\!\top \ \} \\
 & T \circ R \triangle I \circ (\![\,]\!) \;=\; (\![\,]\!) \\
\Leftarrow & \qquad \{ \text{ domains } \} \\
 & T \circ R \triangle I \circ (\![\,]\!)< \;=\; (\![\,]\!)<
\end{array}
$$

This means that the value of R must be a left unit of T for the range of the catamorphism. A dual calculation for the right unit law leads to the condition:

$$T \circ I \triangle R \circ (\![\,]\!)< \;=\; (\![\,]\!)<$$

The results of the calculation for the associativity law is the condition:

$$T \circ T{\times}I \circ \mathsf{assoc} \circ (\![\,]\!)< \times ((\![\,]\!)< \times (\![\,]\!)<) \;=\; T \circ I{\times}T \circ (\![\,]\!)< \times ((\![\,]\!)< \times (\![\,]\!)<)$$

The interpretation of this condition is that T is associative on the range of the catamorphism. For the symmetry law we obtain as condition:

$$T \circ \mathsf{swap} \circ (\![\,]\!)< {\times} (\![\,]\!)< \;=\; T \circ (\![\,]\!)< {\times} (\![\,]\!)<$$

So T must be symmetric on the range of the catamorphism. For *Set* we have a more complicated situation. We are not able to remove the catamorphism itself from the condition:

$$T \circ (\![\,]\!) \triangle (\![\,]\!) \;=\; (\![\,]\!)$$

A more complete treatment of the Boom-hierarchy as defined here can be found in the work of Hoogendijk [34, 35], including for example how filter and reduce operations can be defined.

9.2.7 Example: Arrays

The datatype of Arrays was discussed by Bird in [12]. This is a parameterised type with both laws and restrictions (we use A for the parameter and B for the carrier) and has three constructors:

- $\tau : B \hookleftarrow A$, constructing a 1×1 array from an element of the parameter type.
- $above : B \leftarrow B{\times}B$, constructing a new array by putting one array above another array.
- $beside : B \leftarrow B{\times}B$, constructing a new array by putting one array beside another array.

The relator corresponding to these constructors is defined by

$$A{\otimes}X \quad \triangleq \quad A{+}(X{\times}X{+}X{\times}X)$$

We denote $\mu(R\otimes)$ by ϖR. For the free type corresponding to this relator, the type without any laws and restrictions, the constructors are defined as follows:

$$\begin{array}{lcl} \tau & = & \varpi A \circ \hookrightarrow \\ above & = & \varpi A \circ \hookleftarrow \circ \hookrightarrow \\ beside & = & \varpi A \circ \hookleftarrow \circ \hookleftarrow \end{array}$$

Arrays satisfy three laws:

1. The *above* operation is associative
2. The *beside* operation is associative
3. The *above* and *beside* operations "abide" with each other: for arrays P, Q, R and S, the following equality holds:

   $$(P \; beside \; Q) \;\; above \;\; (R \; beside \; S) \quad = \quad (P \; above \; R) \;\; beside \;\; (Q \; above \; S)$$

   Or, in a more graphical form:

   $$\left[\begin{array}{c|c} P & Q \\ \hline R & S \end{array}\right] = \left[\begin{array}{c|c} \frac{P}{R} & \frac{Q}{S} \end{array}\right]$$

The restrictions for arrays are that only arrays with the same width can be put above each other and that only arrays with the same height can be put beside each other.

Bird has severe problems handling the partiality of the *above* and *beside* constructors and often has to add qualifications like "provided that (some expression) is defined" to definitions. Our theory gives a formal justification for the assumptions that he makes.

The first two law equations can be expressed in the form of equation 9.6 as:

$$Array \circ above \circ above{\times}I \circ \mathsf{assoc} \quad = \quad Array \circ above \circ I{\times}above$$

and

$$Array \circ beside \circ beside{\times}I \circ \mathsf{assoc} \quad = \quad Array \circ beside \circ I{\times}beside$$

We can transform the third law equation to a point-free form by assuming that the lhs and rhs of the following equality are both applied to $((P,S),(Q,R))$:

$$\begin{array}{ll} & Array \circ above \circ beside{\times}beside \circ (\ll \times \ll)\triangle((\gg \times \gg) \circ \mathsf{swap}) \\ = & \\ & Array \circ beside \circ above{\times}above \circ (\ll \times \gg)\triangle(\ll \times \gg \ \circ \mathsf{swap}) \end{array}$$

This equation also has the correct shape (equation 9.6) that our formalism can handle.

We introduce four extra functions to express the restrictions on the construction of arrays. We define catamorphisms that calculate the "lengths" of the 4 sides of an array:

$$\begin{array}{lll} toplength & \triangleq & (\!|1^{\bullet} \triangledown (\ll \triangledown add)|\!) \\ bottomlength & \triangleq & (\!|1^{\bullet} \triangledown (\gg \triangledown add)|\!) \\ leftlength & \triangleq & (\!|1^{\bullet} \triangledown (add \triangledown \ll)|\!) \\ rightlength & \triangleq & (\!|1^{\bullet} \triangledown (add \triangledown \gg)|\!) \end{array}$$

The function *add* above is the addition of natural numbers. We can express that only two arrays with the same width can be placed above each other by demanding that

$$toplength \circ Array \ = \ bottomlength \circ Array$$

This equation holds on all "levels" of the array. If two arrays are placed above each other, then the new top length can only be equal to the new bottom length if the top length (= bottom length) of the upper array equals the bottom length (= top length) of the lower array.

The equation can easily be transformed to the desired shape (formula 9.7) for restriction equations:

$$\begin{array}{l} toplength \circ Array \ \sqsupseteq \ bottomlength \circ Array \\ bottomlength \circ Array \ \sqsupseteq \ toplength \circ Array \end{array}$$

The restriction that only arrays with the same height can be put beside each other is expressed similarly:

$$\begin{array}{l} leftlength \circ Array \ \sqsupseteq \ rightlength \circ Array \\ rightlength \circ Array \ \sqsupseteq \ leftlength \circ Array \end{array}$$

Parameterising the *Array* construction, extending the construction above to a relator is possible. The equations given above do have the naturality properties, (9.29) for laws and (9.31) for subtyping, that are sufficient for parameterisability.

Checking the naturality properties does require some rather long, but not difficult calculations. The calculations for the law equations are similar to the calculations used for the associativity law for non-empty join lists and the calculation for the subtyping equations is similar to that for height-balanced trees. The details of the calculations are left to the reader.

9.3 Transformers

The calculations with catamorphisms in section 9.2.6 show an interesting pattern. For the left-unit law we calculated (with some substitutions and application of the rules for disjoint sum):

$$
\begin{array}{l}
\llparenthesis R\triangledown(S\triangledown T)\rrparenthesis \circ \varpi A \circ \hookleftarrow \circ \hookleftarrow \circ (\varpi A \circ \hookrightarrow \circ \top\!\!\top)\vartriangle I \;=\\
R\triangledown(S\triangledown T) \circ \hookleftarrow \circ \hookleftarrow \circ (R\triangledown(S\triangledown T) \circ \hookrightarrow \circ \top\!\!\top)\vartriangle I \circ \llparenthesis R\triangledown(S\triangledown T)\rrparenthesis
\end{array}
$$

The subexpressions with ϖA and $R\triangledown(S\triangledown T)$ have a similar structure:

$$\llparenthesis R\triangledown(S\triangledown T)\rrparenthesis \circ \Phi.\varpi A \;=\; \Phi.(R\triangledown(S\triangledown T)) \circ \llparenthesis R\triangledown(S\triangledown T)\rrparenthesis$$

where $\Phi.X \;=\; X \circ \hookleftarrow \circ \hookleftarrow \circ (X \circ \hookrightarrow \circ \top\!\!\top)\vartriangle I$.

The proof of the equality only uses the following property of $\llparenthesis R\triangledown(S\triangledown T)\rrparenthesis$, $R\triangledown(S\triangledown T)$ and ϖA:

$$\llparenthesis R\triangledown(S\triangledown T)\rrparenthesis \circ \varpi A \;=\; R\triangledown(S\triangledown T) \circ A\otimes\llparenthesis R\triangledown(S\triangledown T)\rrparenthesis$$

The function Φ satisfies the following condition for all R, S and T:

$$R \circ S \;=\; T \circ A\otimes R \quad\Rightarrow\quad R \circ \Phi.S \;=\; \Phi.T \circ R$$

The function Φ is an example of a *transformer* as introduced in Fokkinga's Ph.D. thesis [24], here translated to a SPEC-calculus definition:

Definition 9.32: (Transformer)

For relators F, G, H and J, the function Φ is a *transformer* of type $(F,G) \longrightarrow (H,J)$ if for all R, S and T:

$$G.R \circ S \;=\; T \circ F.R \quad\Rightarrow\quad J.R \circ \Phi.S \;=\; \Phi.T \circ H.R$$

□

The categorical definition in [24] has functors instead of relators and R, S and T are arrows with appropriate types. Many laws on an F-inductive type can be specified using two transformers of type $(F,\mathcal{I}) \longrightarrow (H,\mathcal{I})$ for some relator H. For transformers Φ and Ψ the equation in (9.8) becomes:

$$X \circ \Phi.\mu F \;=\; X \circ \Psi.\mu F$$

The condition under which a catamorphism has the type with law as a domain can also be simplified for laws constructed with transformers:

$$
\begin{array}{ll}
 & \llparenthesis R\rrparenthesis \circ \Phi.\mu F \;=\; \llparenthesis R\rrparenthesis \circ \Psi.\mu F \\
\equiv & \qquad \{\ \Phi \text{ and } \Psi \text{ transformers } (F,\mathcal{I}) \longrightarrow (H,\mathcal{I}),\ \llparenthesis R\rrparenthesis \circ \mu F \;=\; R \circ F.\llparenthesis R\rrparenthesis\ \} \\
 & \Phi.R \circ H.\llparenthesis R\rrparenthesis \;=\; \Psi.R \circ H.\llparenthesis R\rrparenthesis \\
\Leftarrow & \qquad \{\ \text{domains}\ \} \\
 & \Phi.R \circ (H.\llparenthesis R\rrparenthesis)< \;=\; \Psi.R \circ (H.\llparenthesis R\rrparenthesis)<
\end{array}
$$

A special case arises for the equation $X \circ \Phi.\mu F \;=\; X \circ \mu F$, the unit laws being an example of this kind of equation. In general there is no transformer Ψ with type $(F,\mathcal{I}) \longrightarrow (\mathcal{I},\mathcal{I})$ such that $\Psi.\mu F \;=\; \mu F$, so specification with two transformers is not possible. For this situation we calculate:

$$
\begin{array}{ll}
 & (\!|R|\!) \circ \Phi.\mu F \;=\; (\!|R|\!) \circ \mu F \\
\Leftarrow & \qquad \{\ \Phi \text{ transformer of type } (F,\mathcal{I}) \longrightarrow (\mathcal{I},\mathcal{I}),\ (\!|R|\!) \circ \mu F \;=\; R \circ F.(\!|R|\!)\ \} \\
 & \Phi.R \circ (\!|R|\!) \;=\; (\!|R|\!) \\
\Leftarrow & \qquad \{\ \text{domains}\ \} \\
 & \Phi.R \circ (\!|R|\!)_< \;=\; (\!|R|\!)_<
\end{array}
$$

All laws in the Boom-hierarchy, except *Set*, can be specified using the transformer methods described above. The unit laws have one transformer and μF, the associativity and commutativity laws have two transformers. For *Set* we have on one side μF but we are unable to find a transformer for the other side. This is a problem of the SPEC-approach because there is a candidate, $\Phi.X \;=\; X \circ \;\hookleftarrow\; \circ \;\hookleftarrow\; \circ I \triangle I$, that would work in a functional setting but this Φ only satisfies the condition in the definition of transformer for R's that are left-imps.

Fokkinga uses transformers in his thesis to define algebras with laws. Given two transformers Φ and Ψ he considers algebras τ such that $\Phi.\tau \;=\; \Psi.\tau$. These are the lawful algebras and they also form a category. Initial objects exist under suitable conditions in this category of lawful algebras. This initial object corresponds to our type with law constructed using the method described in this chapter. His method is potentially more powerful than our method in the sense that he also has the possibility to specify laws on F-algebras with transformers of type $(F,\mathcal{I}) \longrightarrow (H,J)$ where J is not an identity functor while our equational definition method corresponds to using $(F,\mathcal{I}) \longrightarrow (H,\mathcal{I})$ transformers. However, this freedom is not exploited in his thesis. A later article by Fokkinga [25] uses equations similar to ours as the basis for a transformer approach to datatypes with laws. The transformer method is an elegant way of introducing laws in a categorical theory of algebra because of its simplicity, avoiding the complications of the more traditional methods using signatures (see for example [22, 57]). A weakness of the transformer method is that there does not seem to be a simple way of generalising the construction method to subtypes like height-balanced trees.

There is still an important open question about the relationship between the transformer method and our method. It is unknown whether every equation that is allowed in our system can also be expressed using transformers. We don't know any counterexamples but we also have no idea how such a result could be proved.

9.4 Abstract Datatypes

The theory developed for constructing a single inductively-defined type can also be used for the construction of mutually recursive inductively-defined types. This is done by working in a binary SPEC-calculus if we are defining two types, in a ternary SPEC-calculus for three types etc..

A simple example of this technique is the construction of the natural numbers as two types, the even and the odd numbers. We have three constructors:

$0^{\bullet} : even \leftarrow\!\dashv\, \top\!\!\top$,
$\sigma o : even \leftarrow\!\dashv\, odd$ and
$\sigma e : odd \leftarrow\!\dashv\, even$.

The corresponding relator is defined by

$$F.(A,B) \quad \triangleq \quad (\top\!\!\top + B, A)$$

The least fixed point of F is a per consisting of two components, the even numbers and the odd numbers. The constructors can be defined in the usual way. Adding laws and restrictions to inductive types in non-unary SPEC-calculi is performed using the same techniques that we developed for unary SPEC-calculi.

Abstract datatypes can not only have multiple carrier types but can also have functions with as range a type that is not being constructed by the constructors. A good example is the abstract datatype of *stacks*. Although there is only one type being constructed, *Stack*, normal specifications also define two functions that do not have *Stack* as range. Constructing the constructors and extra functions of an abstract datatype is not always possible using our methodology, but we show that, if the extra functions are specified using equations of an appropriate shape, we can still construct solutions for equations. We demonstrate the construction method using stacks as example.

The constructors for stacks over a given type A are:

$empty : Stack \leftarrow\!\dashv\, \top\!\!\top$
$push : Stack \leftarrow\!\dashv\, A \times Stack$
$pop : Stack \leftarrow Stack$

with equations

$$pop \circ push \quad = \quad \gg \circ\, A \times Stack$$
$$\top\!\!\top \circ push^{\cup} \;\sqsupseteq\; \top\!\!\top \circ pop$$

Stacks also have two extra functions:

$top : A \leftarrow Stack$
$isempty : \mathbb{B} \leftarrow\!\dashv\, Stack$

These functions are specified by the following equations:

$$top \circ push \quad = \quad \ll \circ\, A \times Stack$$
$$isempty \circ empty \quad = \quad true^{\bullet}$$
$$isempty \circ push \quad = \quad false^{\bullet} \circ A \times Stack$$

Every equation has the same shape; the function f that is being defined satisfies

$$f \circ g \;=\; h$$

where g and h are functions independent of f and the type judgements of all functions match. Applying $\circ\, g^{\cup}$ to both sides yields:

$$f \circ g \circ g^{\cup} \;=\; h \circ g^{\cup}$$

Because g is functional we can conclude

$$f \;\sqsupseteq\; h \circ g^{\cup}$$

and f can be defined as $h \circ g^\cup$ on the range of g. For the functions of the stack datatype we get as candidates:

$$top = \ll \circ push^\cup$$
$$isempty = true^\bullet \circ empty^\cup \sqcup false^\bullet \circ push^\cup$$

These are partial definitions; the functions are not defined outside the domain that was specified by the equations. Larger solutions to the equations may exist. The particular shape of the equations allows us to find candidates for a solution to the specification, but there are of course also other shapes of equations for which solutions exist.

9.5 Conclusion

We showed that many equational definitions of properties of inductive datatypes can be captured in a per-lattice based theory of datatypes. The most important result of this chapter is that many common forms of equations in definitions of abstract datatypes are instances of one general equation in the per lattice and that the greatest solution of this equation is the datatype that is to be constructed.

Chapter 10

Conclusions

This thesis is intended to contribute to a long tradition in the application of the calculational method to the mathematics of program construction. The tradition, begun by Dijkstra and Feijen, has been exposed in a number of theses of which we mention particularly that of A.J.M. van Gasteren [27] on the presentation of mathematical proofs.

The calculational method has as goal to shift the balance between inspiration and perspiration. Inspiration is identifying the right concepts and finding the right formalisms in the process of managing complex problems. Transpiration is the hard work of calculating a solution to the problem, once the inspirational process is complete. Inspiration is "hard" because it requires intellectual effort, whereas transpiration is "hard" in the sense of requiring much toil. An optimum balance is achieved when the final calculation is guided by the shape of the mathematical formulation of the problem.

The topic of this thesis is the use of a point-free relational calculus for modelling inductive datatypes with laws and restrictions. We chose to explore the use of point-free relational calculus rather than a pointwise calculus in order to achieve a greater degree of conciseness. Proofs using the point-free calculus are in general much easier to construct than pointwise proofs since the latter tend to involve complicated quantifications with a myriad of dummies. Working with a point-free calculus has not only proved to be an advantage for the calculus used in this thesis, but also in other formalisms like category theory or the Bird-Meertens formalism.

The choice of axioms and mathematical tools turned out to be very important for the usability of the calculus. Some important aspects of our methodology that turned out to be critical are:

- The use of complete lattices. Every monotonic endofunction on a complete lattice has least and greatest fixedpoints, giving a simple and powerful method for the definition of new constructions.

- Using Galois connections. This is also an important tool for the definition of new constructions. Their use in proofs is important too. It turns out that many algebraic properties of the constructions we are interested in follow immediately from general Galois connection properties. The use of Galois connections has also simplified many proofs.

- Fusion properties. Fusion properties arise from the combination of fixpoints and Galois connections. Elsewhere [9, 41, 44] they have proved to be indispensable in program transformation. Here they have been invaluable in condensing proofs and suggesting proof strategies.

- Not using complements. This is to some extent a matter of style because a relational calculus with complements would have been equally usable for the purposes of this thesis. Limiting the possibilities of the calculus proved useful in finding elegant proofs for properties that were previously proved with "ugly" proofs involving complements.

The work in this thesis has been much influenced by earlier theories of datatypes based on category theory, although we have not needed to use category theory directly. A particular influence has been the work of Malcolm [42] which made the categorical notion of an initial F-algebra accessible to a much wider audience and laid the basis for a generic theory of free datatypes. Also relevant has been Fokkinga's [24] fine-tuning of Malcolm's work and his extensions to types with laws using transformers. One of the reasons why we did not use category theory more heavily is that a combination of laws and restrictions seems difficult to formulate in category theory.

Characteristic of our approach is that types and programs are both relations and can be mixed in calculations. In our calculus a relator is a single mapping from relations to relations, whereas in category theory a functor is a pair of mappings, one defined on objects (types) and one defined on arrows (programs). In addition, a natural transformation in our system is a single relation as opposed to a family of arrows. The internalisation of these notions in a single calculus simplifies calculations by reducing the formal overhead.

The relational calculus is a rich calculus and there are several possibilities for modelling types. The method used in this thesis, using partial equivalence relations (pers), led to the discovery of a new complete lattice on pers where the ordering corresponds to subtype and quotient type formation.

Inductive types, where every element can be constructed using a finite number of applications of a constructor, are defined using greatest fixed points of monotonic functions in the per-lattice. The collection of inductive types with a given recursion structure, specified by a relator, forms a complete lattice with the same ordering as the per lattice. Inductive types satisfying law and restriction equations are constructed using a Galois connection between the per lattice and the lattice of the inductive types. An important result is that law and restriction equations are both instances of a more general class of equations and can be combined without problems. This contrasts with other formalisms where laws and restrictions are dual notions and combination is difficult.

Many recursively defined programs on inductive datatypes can be defined as unique solutions of equations. Doornbos proposes the notion of F-reductivity [19] in order to model termination properties of structural recursion. In essence, F-reductivity formalises the idea that elements of an inductive datatype *must* be reduced within a finite number of steps to their basic components. Doornbos considers a special class of equations for which the existence of solutions is trivial, but the uniqueness of solutions makes essential use of the reductivity property. The disadvantage is that the theory does not appear to extend to types with laws. The main distinction between our theory of F-inductivity and the theory of F-reductivity can be formulated as that F-inductivity formalizes that every element of an inductive datatype *can* be constructed with a finite number of steps. The shape of equations that we consider for specifying structural recursion is such that the uniqueness problem can be resolved by using F-inductivity, but the existence of solutions remains an issue when types with laws are combined with complicated patterns of structural recursion.

The programs resulting from the constructions above are relations and can be nondeterministic. Functionality requirements are not necessary for having unique solutions to the equations used for specifying recursive programs on inductive datatypes. The relevance of this to practical program development is not discussed here but is evident in many publications. See for example Bird and De Moor [10] or Berghammer and von Karger [8].

Another topic for further research is infinite datatypes, types containing elements that can not be constructed with a finite number of applications of constructors. A typical example of such a datatype is infinite lists. The definition of an inductive datatype that is given in this thesis has two components, one specifying the recursion structure and one specifying the finiteness of construction of elements. It seems that dropping the second component from the definition would allow infinite datatypes but the consequences of this choice are not clear yet.

A small open problem that needs further research is the existence of a finite set relator. It is not difficult to define such an operator on pers, but extending the operation to all specs in the standard way results in a function for which we are unable to prove that it is a relator.

There is also a more fundamental problem with the calculus that is used in this thesis. A datatype constructor that is commonly used in functional programming is the function space constructor. This construction can not be modelled in our calculus because of cardinality problems. Functions having function (or relation) parameters, like the fold in Miranda/Haskell/Gofer, are not objects of the calculus and have to be defined in other ways, for example as solutions of parameterised equations. How this problem can be solved is not clear yet. One approach might be weakening the axioms of the calculus but this will probably invalidate many of the results obtained thus far.

The relational calculus is a small calculus but proved to be a surprisingly strong tool for modelling (inductive) datatypes and programs. The simplicity of the characterisations in the calculus of many important notions about datatypes and programs and the ease of calculation with these notions shows that the relational calculus is an excellent tool for reasoning about these subjects.

Bibliography

[1] C.J. Aarts, R.C. Backhouse, P. Hoogendijk, T.S. Voermans, and J. van der Woude. A relational theory of datatypes. Available via anonymous ftp from `ftp.win.tue.nl` in directory `pub/math.prog.construction`, September 1992.

[2] R.C. Backhouse, P. de Bruin, P. Hoogendijk, G. Malcolm, T.S. Voermans, and J. van der Woude. Polynomial relators. In M. Nivat, C.S. Rattray, T. Rus, and G. Scollo, editors, *Proceedings of the 2nd Conference on Algebraic Methodology and Software Technology, AMAST'91*, pages 303–362. Springer-Verlag, Workshops in Computing, 1992.

[3] R.C. Backhouse, P. de Bruin, G. Malcolm, T.S. Voermans, and J. van der Woude. A relational theory of datatypes. Eindhoven University of Technology and University of Groningen, September 1990.

[4] R.C. Backhouse, P. de Bruin, G. Malcolm, T.S. Voermans, and J. van der Woude. Relational catamorphisms. In Möller B., editor, *Proceedings of the IFIP TC2/WG2.1 Working Conference on Constructing Programs*, pages 287–318. Elsevier Science Publishers B.V., 1991.

[5] R.C. Backhouse and H. Doornbos. Induction and recursion on datatypes. Department of Computing Science, Eindhoven University of Technology. Available via anonymous ftp from `ftp.win.tue.nl` in directory `pub/math.prog.construction`, 1993.

[6] R. Banach. Regular relations and bicartesian squares. *Theoretical Computer Science*, (129), 1994.

[7] Michael Barr and Charles Wells. *Category Theory for Computing Science*. Prentice Hall International Series in Computer Science. Prentice Hall, 1990.

[8] Rudolf Berghammer and Burghard von Karger. *Algorithms from Relational Specifications*, chapter 9, pages 131–149. Advances in Computing. Springer-Verlag, Wien, New York, 1997.

[9] Richard S. Bird and Oege de Moor. Solving optimisation problems with catamorphisms. In R.S. Bird, C.C. Morgan, and J. C. P. Woodcock, editors, *2nd International Conference on the Mathematics of Program Construction*, pages 45–66. Springer Verlag LNCS, June/July.

[10] Richard S. Bird and Oege de Moor. *Algebra of Programming.* Prentice-Hall International, 1996.

[11] R.S. Bird. An introduction to the theory of lists. In M. Broy, editor, *Logic of Programming and Calculi of Discrete Design.* Springer-Verlag, 1987. NATO ASI Series, vol. F36.

[12] R.S. Bird. Lectures on constructive functional programming. In M. Broy, editor, *Constructive Methods in Computing Science*, pages 151–216. Springer-Verlag, 1989. NATO ASI Series, vol. F55.

[13] Garrett Birkhoff. *Lattice Theory*, volume 25 of *American Mathematical Society Colloquium Publications.* American Mathematical Society, Providence, Rhode Island, 3rd edition, 1967.

[14] P.J. de Bruin. Naturalness of polymorphism. Technical Report CS8916, Department of Mathematics and Computing Science, University of Groningen, 1989.

[15] J.H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, London, 1971.

[16] B. A. Davey and H. A. Priestly. *Introduction to Lattices and Order.* Cambridge Mathematical Textbooks. Cambridge University Press, first edition, 1990.

[17] E.W. Dijkstra and W.H.J. Feijen. *Een Methode van Programmeren.* Academic Service, Den Haag, 1984. Also available as *A Method of Programming*, Addison-Wesley, Reading, Mass., 1988.

[18] R.P. Dilworth. Non-commutative residuated lattices. *Transactions of the American Mathematical Society*, 46:426–444, 1939.

[19] H. Doornbos. *Reductivity arguments and program construction.* PhD thesis, Eindhoven University of Technology, Department of Mathematics and Computing Science, June 1996.

[20] Henk Doornbos and Roland Backhouse. Induction and recursion on datatypes. In B. Möller, editor, *Mathematics of Program Construction, 3rd International Conference*, volume 947 of *LNCS*, pages 242–256. Springer-Verlag, July 1995.

[21] Henk Doornbos and Roland Backhouse. Reductivity. *Science of Computer Programming*, 26(1–3):217–236, 1996.

[22] H. Ehrig, H.-J. Kreowski, B. Mahr, and P. Padawitz. Algebraic implementation of abstract data types. *Theoretical Computer Science*, 20:209–263, 1982.

[23] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag Berlin, 1985.

[24] Maarten M. Fokkinga. *Law and Order in Algorithmics.* PhD thesis, Universiteit Twente, The Netherlands, 1992.

[25] Maarten M. Fokkinga. Datatype laws without signatures. *Mathematical Structures in Computer Science*, (6):1–32, 1996.

[26] P.J. Freyd and A. Ščedrov. *Categories, Allegories*. North-Holland, 1990.

[27] A.J.M. van Gasteren. *On the Shape of Mathematical Arguments*. PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1988.

[28] G. Gierz, K. H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, and D. S. Scott. *A Compendium of Continuous Lattices*. Springer-Verlag, 1980.

[29] J.A. Goguen and J.W. Thatcher. Initial algebra semantics. In *Switching and automata theory conference*. IEEE, 1974.

[30] J.A. Goguen, J.W. Thatcher, E. Wagner, and J. Wright. Abstract datatypes as initial algebras and the correctness of datarepresentations. In *Computer graphics, pattern recognition and data structure*, pages 89–93. IEEE, 1975.

[31] J.A. Goguen, J.W. Thatcher, and E.G. Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In R.T. Yeh, editor, *Current Trends in Programming Methodology, Volume 4: Data Structuring*, pages 80–149. Prentice-Hall, 1978.

[32] T. Hagino. A typed lambda calculus with categorical type constructors. In D.H. Pitt, A. Poigne, and D.E. Rydeheard, editors, *Category Theory and Computer Science*, pages 140–57. Springer-Verlag Lecture Notes in Computer Science 283, 1988.

[33] C.A.R. Hoare and Jifeng He. The weakest prespecification. *Fundamenta Informaticae*, 9:51–84, 217–252, 1986.

[34] Paul Hoogendijk and Roland C. Backhouse. Relational programming laws in the tree, list, bag, set hierarchy. *Science of Computer Programming*, 22(1–2):67–105, 1994.

[35] P.F. Hoogendijk. (Relational) Programming laws in the Boom hierarchy of types. In R.S. Bird, C.C. Morgan, and J.C.P. Woodcock, editors, *Mathematics of Program Construction. 2nd International Conference, June/July 1992*, volume 669 of *Lecture Notes in Computer Science*, pages 163–190. Springer Verlag, 1993.

[36] Ali Jaoua and Martin Beaudry. Difunctional relations: A formal tool for program design. Technical report, Université de Sherbrooke, Faculté des Sciences, Département de mathématiques et d'informatique, July 1989.

[37] Ali Jaoua, N. Boudriga, J.L. Durieux, and A. Mili. Pseudo invertibility, a measure of regularity of relations. *Theoretical Computer Science*, 77, November 1990.

[38] Geraint Jones and Mary Sheeran. Designing arithmetic circuits by refinement in Ruby. Technical Reports TR-1-92, Oxford University Computing Laboratory, Programming Research Group, Januari 1992.

[39] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, 1971.

[40] Roger D. Maddux. The origin of relation algebras in the development and axiomatization of the calculus of relations. Department of Mathematics, Iowa State University, Ames, IOWA 50011, USA, May 1990.

[41] G. Malcolm. Homomorphisms and promotability. In J.L.A. van de Snepscheut, editor, *Conference on the Mathematics of Program Construction*, pages 335–347. Springer-Verlag LNCS 375, 1989.

[42] G. Malcolm. *Algebraic data types and program transformation.* PhD thesis, Groningen University, 1990.

[43] E.G. Manes and M.A. Arbib. *Algebraic Approaches to Program Semantics.* Texts and Monographs in Computer Science. Springer-Verlag, Berlin, 1986.

[44] L. Meertens. Algorithmics – towards programming as a mathematical activity. In *Proceedings of the CWI Symposium on Mathematics and Computer Science*, pages 289–334. North-Holland, 1986.

[45] L. Meertens. Paramorphisms. *Formal Aspects of Computing*, 4(5):413–424, 1992.

[46] J. Meseguer and J.A. Goguen. Initiality, induction and computability. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 459–542. Cambridge University Press, 1985.

[47] Ali Mili, Jules Desharnais, and Fatma Mili. *Computer program construction.* Oxford University Press, 1994.

[48] J.C. Mitchell. A type-inference approach to reduction properties and semantics of polymorphic expressions. In *ACM Conference on Lisp and Functional Programming*, 1986.

[49] O. de Moor. *Categories, Relations and Dynamic Programming.* PhD thesis, Oxford University Laboratory, Programming Research Group, April 1992.

[50] Benjamin C. Pierce. *Basic category theory for computer scientists.* MIT Press, London, 1991.

[51] J.C. Reynolds. Types, abstraction and parametric polymorphism. In R.E. Mason, editor, *IFIP '83*, pages 513–523. Elsevier Science Publishers, 1983.

[52] F.J. Rietman. A note on extensionality. In J. van Leeuwen, editor, *Proceedings Computer Science in the Netherlands 91*, pages 468–483, 1991.

[53] J. Riguet. Relations binaires, fermetures, correspondances de Galois. *Bulletin de la Société Mathématique de France*, 76:114–155, 1948.

[54] G. Schmidt and T. Ströhlein. *Relations and Graphs.* Springer-Verlag, 1991.

[55] M. Spivey. A categorical approach to the theory of lists. In J.L.A. van de Snepscheut, editor, *Conference on the Mathematics of Program Construction*, pages 399–408. Springer-Verlag, LNCS 375, 1989.

[56] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6(3):73–89, 1941.

[57] N. Verwer. *Astract Data Conversion.* PhD thesis, Rijks Universiteit Utrecht, 1993.

[58] P. Wadler. Theorems for free! In *4'th Symposium on Functional Programming Languages and Computer Architecture, ACM, London*, September 1989.

Index

., 8, 80
/, 19
F-algebra, 121
  initial, 122
F-homomorphism, 121
F-inductive algebra, 127, 132
  lawful, 145
  restricted, 145
F-inductive algebras, 144
  category of, 130
  isomorphic, 131
F-inductive closure, 151, 152, 155
F-inductive type, 132
F-inductive types
  lattice of, 132
F-reductivity, 141, 142
I, 15
$\scriptstyle<$, 38
$\scriptstyle\prec$, 74
$\swarrow$, 73
$\scriptstyle>$, 38
$\scriptstyle\succ$, 73
$\searrow$, 72
$\backslash$, 19
$+$, 106, 108
$\bigsqcap$, 16
$\bigsqcup$, 16
$\bullet$, 94
$\relbar\!\!\relbar$, 70
$\longrightarrow$, 83
$\relbar\!\!\relbar\!\dashv$, 83
$\cdot$, 8
$\circ$, 15
$\circ$-$\sqcap$ distribution, 30, 31
δ, 94
$[\![F;R,S]\!]$, 124
$\emptyset$, 6
$\leftarrow$, 79
$\leftarrow\!\dashv$, 8
$\flat$, 12, 13
$(\![F;R]\!)$, 123
$\langle \_ \rangle$, 77
$\ll$, 94
μ, 11
μ-fusion, 13
$\prec\!\dot\sim$, 102
$\prec\!\dot\sim\!\succ$, 102
$\dot\sim\!\succ$, 102
ν, 11
ν-fusion, 13
$\swarrow\!\!\!\searrow$, 62
$\scriptstyle\rtimes$, 38
$\perp\!\!\!\perp$, 16
$\top\!\!\!\top$, 16
$\scriptstyle\cup$, 15
$\backslash\!\!\!\blacktriangleleft$, 141
$\gg$, 94
$\vdash\!\!\relbar$, 84
$\sharp$, 12, 13
$\sqcap$, 16
$\sqcup$, 16
$\sqsubseteq$, 15
$\sqsupseteq$, 16
$\triangledown$, 108
$\vartriangle$, 108
$\lfloor \_ \rfloor$, 151, 152
$\times$, 107
$\triangleleft$, 65
Set, 88
$\mathcal{I}$, 94
$\cap$, 57
$\cup$, 57
$\triangleleft\!\mid$, 60
$\in$, 53

cod, 87
dom, 87
id, 87
$\simeq$, 131
$\gtrsim$, 129
$\subseteq$, 56
$\times$, 108
assoc, 116
$\circ$, 87
swap, 116
Bag, 166
List, 166
Set, 166
Tree, 166

abstract data type, 130
adjoint, 11
    lower, 11
    upper, 11
anti-monotonicity, 9, 29
anti-symmetry, 9
application, 80
associativity, 19, 117, 166
atomicity, 48

binary functor, 89
binary relator, 93
binary spec, 117
Boom-hierarchy, 165
bottom, 10, 17, 50
bound
    greatest lower, 10, 17
    least upper, 9, 17

calculation rule, 11, 113
cancellation, 12, 17, 20, 29
cap, 16, 77
carrier, 121
Cartesian product, 106, 108–110
catamorphism, 123, 133
category, 87
class, 53
closure, 35
closure operator, 151
codomain, 87
commutativity, 166
complement, 21
compose, 77
cone rule, 23
constant, 89
constant functor, 89
constructor, 119, 128
contravariance, 20
cup, 16, 77
cyclic implication, 6
cyclic inclusion, 6

Dedekind, 20, 21, 31
Desargues, 22
Difun, 91
difunctionality, 80
difunctionals, 82
disjoint sum, 106, 108, 109
distribution, 113
distributivity, 12, 13, 16, 17, 20, 29, 34
domain, 87
domain trading, 40
domains, 37–43, 70, 71, 114
doubling functor, 89

element, 53
endofunctor, 89
equality, 77
equivalence class
    left, 70
    right, 70
equivalence domains, 73, 74, 97
extension, 62
    greatest, 62
extensionality, 23, 46, 54, 55, 77

factor
    left, 19
    right, 19
factors, 19, 28–30
free type, 122
functionality, 79
functor, 89
fusion, 13, 113

Galois connection, 11
generalized range-disjunction, 7

glb, 10
global element, 89
greatest domains, 72, 73, 95

hylomorphism, 124

idempotent, 166
identity, 50
identity arrow, 88
identity functor, 89
indirect equality, 9
induction, 11
initial F-algebra, 122
initial object, 88
initiality, 88
injection, 106
injectivity, 83
isomorphic relators, 95
isomorphism, 88, 131
isomorphy, 88, 91

junc, 108

kernel, 151
Knaster-Tarski, 10, 11

lattice
    complete, 10
law, 149
law equation, 149
lc, 33
least domains, 71, 96, 97
least upper bound, 36
left-condition, 33, 50
left-domain, 38
left-imp, 32
Leibnitz, 7
lifting, 9
limp, 32, 49
linear proof, 5
lub, 9, 36

modularity, 20, 21
monoid, 19
monotonicity, 9, 12, 17, 20
monotype factor, 141
monotype judgements, 51
mutumorphism, 139

natural isomorphism, 103
natural transformation, 90, 101–103
naturality, 116, 163
nesting, 7

one-point rule, 7

paramorphism, 139
partial identity relation, 33
partition, 62
per, 36, 50
per order, 65, 69
per-glb, 67, 68
per-lattice, 67
Pfn, 90
pid, 33, 39, 41, 50
polymorphism, 102
poset, 8
product category, 89
projection, 107
projection functors, 89
pseudo-complement, 21
pseudo-invertible, 81

quantifications, 6
quotient construction, 154
quotient lattice, 61
quotient order, 60, 61

range-split, 7
rc, 33
reflexivity, 9
Rel, 90
relates, 77
relational representation, 23
relations
    pseudo-invertible, 80
    regular, 80
relator, 93
    constant, 94
    doubling, 94
    identity, 94
    projection, 94
relator composition, 94
relator tupling, 94

relators, 95, 100
restriction equation, 150
reverse, 17, 36, 77
right-condition, 33
right-domain, 38
right-imp, 32
rimp, 32
rolling, 11

set
  partially ordered, 8
set formation, 8
set-theoretic model, 16
simulation, 129
singleton, 23, 46
spec, 16
SPEC-calculus, 15
  binary, 23
  complemented, 21
  extensional, 23
  product, 22
  ternary, 23
  unary, 23
SPEC-lattice, 16
split, 108
square, 44, 50
stack, 156, 172
strong relator, 97
structural induction, 120
sublattice, 33
subrelator, 102
subset, 56–58
substitution, 7
subtype construction, 76, 155
sum relator, 128
surjectivity, 84
symmetry, 17, 18, 117

terminal object, 88
terminality, 88
top, 10, 17, 50
totality, 83, 84
trading, 38
transformer, 170
transitive closure, 35, 36
transitivity, 9, 35

type
  $\circ$, 85
  $(\!|..|\!)$, 123
  $\cup$, 85
  $\sqcap$, 85
  $\sqcup$, 85
  catamorphism, 123

uep, 125, 127, 140
unicity, 12
unique extension property, 125, 127, 140
unit, 19, 166
  left, 166
  right, 166

zero, 29
zygomorphism, 139

Samenvatting

Introductie

Het onderwerp van dit proefschrift is de wiskundige modellering van inductieve datatypes en recursief gedefiniëerde programma's op deze datatypes. Inductieve datatypes zijn types waarbij de elementen van het type weer subcomponenten van het type kunnen bevatten. Enkele veel gebruikte voorbeelden van inductieve datatypes zijn o.a. binaire bomen, waarbij bomen kunnen worden geconstrueerd door het samennemen van twee subbomen, en cons-lijsten, waarbij een nieuwe lijst kan worden geconstrueerd door het op kop toevoegen van een waarde aan een bestaande cons-lijst.

De twee bovengenoemde types zijn voorbeelden van zgn. vrije types. Dit zijn types zonder wetten en restricties die de constructie van elementen beïnvloeden. In de praktijk zijn er echter ook veel datatypes waarbij wel wetten en restricties gelden. Wetten definiëren gelijkheden tussen elementen, zoals b.v. de commutativiteit van de verenigingsoperator op twee bags bij het datatype bag: $A \cup B = B \cup A$ voor alle bags A en B. Restricties leggen beperkingen op bij de constructie van elementen, b.v. bij hoogtegebalanceerde binaire bomen mogen de hoogtes van de linker en rechter subboom van een element ten hoogste 1 verschillen.

Grant Malcolm ontwikkelde een elegante wiskundige theorie voor vrije datatypes en recursieve programma's op deze types, gebruikmakend van zgn. F-algebra's uit de categorie theorie. Maarten Fokkinga breidde deze theorie later uit naar types met wetten. Hun theoriën behandelden echter geen types met restricties. Andere theoriën van datatypes kunnen wel wetten en restricties modelleren, maar behandelen deze vaak als duale concepten, wat het combineren van wetten en restricties in één datatype bemoeilijkt.

In dit proefschrift wordt een wiskundig model voor inductieve datatypes ontwikkeld waarbij wetten en restricties op een uniforme manier worden behandeld en probleemloos te combineren zijn. Wetten en restricties kunnen via vergelijkingen worden gedefiniëerd. Recursief gedefiniëerde programma's op de inductieve datatypes kunnen worden gespecificeerd als unieke oplossingen van vergelijkingen en de theorie uit dit proefschrift geeft condities waaronder oplossingen van deze vergelijkingen bestaan.

Categorie-theoretische begrippen zijn een bron van inspiratie voor de theorie uit dit proefschrift, maar de modellering die hier gebruikt wordt wijkt duidelijk af van wat gebruikelijk is. Als basis wordt een puntvrije relationele calculus gebruikt. In tegenstelling tot categorie theorie, waar alles strict getypeerd is, is deze relationele calculus ongetypeerd. Relaties worden gebruikt om zowel datatypes als programma's te modelleren. Typering wordt *intern*, in de calculus, gedefiniëerd. Het proefschrift bevat een uigebreide introductie op de relationele calculus (SPEC-calculus) die als basis van de modellering wordt gebruikt.

Naast de relationele calculus spelen volledige tralies een belangrijke wiskundige rol in dit proefschrift. Een volledig tralie heeft als belangrijke eigenschap dat monotone functies altijd unieke extreme fixpoints hebben. de zgn. Knaster-Tarski stelling. Dit biedt de mogelijkheid om canonieke representaties te definiëren voor begrippen waarbij categorie-theoretisch alleen een klasse van "isomorfe oplossingen" gedefiniëerd wordt. In de categorie theorie moet vaak ook nog veel gedaan worden om het bestaan van "oplossingen" aan te tonen.

Een andere wiskundige techniek die in dit proefschrift veel gebruikt wordt is het definiëren van nieuwe begrippen met behulp van Galois connecties. De combinatie van volledige tralies, fixpoints en Galois connecties maakt het vaak mogelijk om eenvoudige en compacte bewijzen van stellingen in een calculationele stijl te construeren. Voor bewijzen en afleidingen wordt in het hele proefschrift zoveel mogelijk deze stijl gebruikt. Dit sluit ook goed aan bij de keuze voor een puntvrije relationele calculus.

Constructie van datatypes en programma's

Datatypes en programma's worden beide gemodelleerd als relaties. De partiële equivalentie relaties (*per*s) uit de relationele calculus worden gebruikt voor de representatie van datatypes. Typering van relaties kan dan worden uitgedrukt in de relationele calculus zelf. Ook standaard begrippen als functionaliteit, injectiviteit, surjectiviteit en totaliteit kunnen relationeel worden gedefiniëerd.

In dit proefschrift wordt een nieuwe ordening op *per*s geïntroduceerd waarbij de *per*s een volledig tralie vormen. De interpretatie van deze ordening is dat een type "kleiner" is dan een ander type als het eerste type kan worden geconstrueerd uit het tweede type door het toevoegen van extra gelijkheden (wetten) en het weglaten van elementen (restricties). Het blijkt dat deze ordening de verbanden tussen types veel beter kan weergeven dan de standaard relationele ordening op *per*s.

De categorie-theoretische aanpak van Malcolm, Fokkinga e.a. is gebaseerd op categoriën waarbij de objecten datatypes zijn en de pijlen functies. In dit proefschrift wordt een categorie (**Difun**) gedefiniëerd die in vele opzichten vergelijkbaar is met de door hen gebruikte categorieën. De objecten zijn *per*s en de pijlen zijn de zgn. difunctionele relaties.

De categorische constructie van datatypes vormt de inspiratie om de relationele calculus zodanig uit te breiden dat dezelfde constructies ook in **Difun** mogelijk zijn. Voor categorische begrippen zoals functors e.d. kan relatief eenvoudig een corresponderend begrip in de relationele calculus geconstrueerd worden. Ook de constructie van basistypen, zoals Cartesisch product en disjoint sum, is na een kleine uitbreiding van de relationele calculus in **Difun** mogelijk.

Een categorie van F-algebra's, die categorisch de basis vormt van de modellering van vrije datatypes, kan eveneens met behulp van **Difun** worden gedefinieerd. Een bijzondere klasse van F-algebra's, de F-inductieve algebra's, speelt hierbij een belangrijke rol. Deze klasse representeert die inductieve datatypes waarbij ieder element met een eindig aantal applicaties van constructoren te construeren is. De F-inductieve algebra's blijken te kunnen worden gepartitioneerd in klasses van onderling simulerende datatypes. Iedere klasse van elkaar simulerende datatypes blijkt één bijzonder element te bevatten dat kan worden gebruikt als canonieke representant van de klasse. Deze representanten (F-inductieve types) vormen samen weer een volledig tralie met de eerdergenoemde ordening op *per*s als ordening. Het lege type vormt de bottom van het tralie, en het vrije type de top.

Voor de vergelijkingen die wetten en restricties definiëren wordt als oplossing een F-inductief type gezocht waarbij het kleinste aantal elementen uit het vrije type zijn samengevoegd of verwijderd. Dit betekent dat de grootste oplossing in het tralie van de F-inductieve types wordt gezocht. Veelgebruikte vormen van vergelijkingen voor wetten en restricties kunnen worden getransformeerd naar een vorm waarbij een grootste fixpoint van een monotone functie op het tralie wordt gevraagd. Omdat wetten en restricties allebei grootste oplossingen zijn, zijn ze eenvoudig met elkaar te combineren.

In het proefschrift wordt de constructie van een aantal bekende datatypes gedemonstreerd. Voorbeelden die behandeld worden zijn o.a.: natuurlijke getallen, cons-lijsten, hoogte-gebalanceerde bomen, de types van de Boom-hierarchie (Tree, List, Bag en Set), stacks en arrays. Het blijkt dat de vergelijkingen van de wetten en restricties (indien aanwezig) van deze types de juiste vorm hebben om de constructie van het datatype met behulp van de theorie uit dit proefschrift mogelijk te maken.

Een ander onderwerp dat besproken wordt is de constructie van geparametriseerde inductieve datatypes. Hierbij wordt, gegeven één of meer argumenttypes, een inductief datatype geconstrueerd. De cons-lijst constructie kan bijvoorbeeld geparametriseerd worden met een argument voor het type van de lijstelementen. Als als argument booleans gebruikt worden, zal de typeconstructie als resultaat cons-lijsten over de booleans hebben etc.. Parametriseerbaarheid is geen probleem bij vrije types, maar het blijkt dat de vergelijkingen voor wetten en restricties aan een aantal extra eisen moeten voldoen om een nette parameteriseerbare type constructie te krijgen. De extra eisen op de vergelijkingen zijn zogenaamde "natuurlijkheids"-eigenschappen die specificeren dat de typeconstructie niet "in" het argumentdatatype mag kijken bij de wetten en restricties.

Recursieve programma's op de inductieve types kunnen ook met behulp van vergelijkingen gespecificeerd worden. Bij een grote klasse van recursiestructuren blijkt dat de inductie eigenschappen van het datatype garanderen dat er ten hoogste één oplossing van de specificerende vergelijkingen bestaat. Of er werkelijk oplossingen bestaan blijkt af te hangen van de vergelijkingen voor het datatype en de operaties in het recursieve programma. Restricties blijken geen invloed te hebben op het bestaan van oplossingen, maar wetten kunnen het bestaan van oplossingen onmogelijk maken.

In het proefschrift worden ook nog een aantal onderwerpen voor toekomstig onderzoek besproken. Interessante open problemen bestaan er o.a. bij datatypes met oneindige elementen (b.v. oneindige lijsten zoals die gebruikt worden bij functioneel programmeren) en bij het bestaan van oplossingen voor de vergelijkingen van recursieve programma's als de recursie structuur ingewikkeld is en er wetten zijn voor het datatype.

Curriculum Vitae

18 februari 1965	Geboren te Eindhoven
mei 1983	Diploma Atheneum B Hertog Jan College Valkenswaard
januari 1988	Doctoraal examen Technische Informatica Technische Universiteit Eindhoven
februari 1988 - augustus 1990	Assistent in opleiding Rijks Universiteit Groningen
september 1990 - augustus 1992	Assistent in opleiding Technische Universiteit Eindhoven
november 1992 - december 1993	Vervangende dienstplicht Universiteit Utrecht
augustus 1994 - heden	Werkzaam bij Algorithmic Research B.V. Nuenen

Titles in the IPA Dissertation Series

The State Operator in Process Algebra
J. O. Blanco
Faculty of Mathematics and Computing Science, TUE, 1996-1

Transformational Development of Data-Parallel Algorithms
A. M. Geerling
Faculty of Mathematics and Computer Science, KUN, 1996-2

Interactive Functional Programs: Models, Methods, and Implementation
P. M. Achten
Faculty of Mathematics and Computer Science, KUN, 1996-3

Parallel Local Search
M. G. A. Verhoeven
Faculty of Mathematics and Computing Science, TUE, 1996-4

The Implementation of Functional Languages on Parallel Machines with Distrib. Memory
M. H. G. K. Kesseler
Faculty of Mathematics and Computer Science, KUN, 1996-5

Distributed Algorithms for Hard Real-Time Systems
D. Alstein
Faculty of Mathematics and Computing Science, TUE, 1996-6

Communication, Synchronization, and Fault-Tolerance
J. H. Hoepman
Faculty of Mathematics and Computer Science, UvA, 1996-7

Reductivity Arguments and Program Construction
H. Doornbos
Faculty of Mathematics and Computing Science, TUE, 1996-8

Functorial Operational Semantics and its Denotational Dual
D. Turi
Faculty of Mathematics and Computer Science, VUA, 1996-9

Single-Rail Handshake Circuits
A. M. G. Peeters
Faculty of Mathematics and Computing Science, TUE, 1996-10

A Systems Engineering Specification Formalism
N. W. A. Arends
Faculty of Mechanical Engineering, TUE, 1996-11

Normalisation in Lambda Calculus and its Relation to Type Inference
P. Severi de Santiago
Faculty of Mathematics and Computing Science, TUE, 1996-12

Abstract Interpretation and Partition Refinement for Model Checking
D. R. Dams
Faculty of Mathematics and Computing Science, TUE, 1996-13

Topological Dualities in Semantics
M. M. Bonsangue
Faculty of Mathematics and Computer Science, VUA, 1996-14

Algorithms for Graphs of Small Treewidth
B. L. E. de Fluiter
Faculty of Mathematics and Computer Science, UU, 1997-01

Process-algebraic Transformations in Context
W. T. M. Kars
Faculty of Computer Science, UT, 1997-02

A Generic Theory of Data Types
P. F. Hoogendijk
Faculty of Mathematics and Computing Science, TUE, 1997-03

The Evolution of Type Theory in Logic and Mathematics
T. D. L. Laan
Faculty of Mathematics and Computing Science, TUE, 1997-04

Preservation of Termination for Explicit Substitution
C. J. Bloo
Faculty of Mathematics and Computing Science, TUE, 1997-05

Discrete-Time Process Algebra
J. J. Vereijken
Faculty of Mathematics and Computing Science, TUE, 1997-06

A Functional Approach to Syntax and Typing
F. A. M. van den Beuken
Faculty of Mathematics and Informatics, KUN, 1997-07

Ins and Outs in Refusal Testing
A.W. Heerink
Faculty of Computer Science, UT, 1998-01

A Discrete-Event Simulator for Systems Engineering
G. Naumoski and W. Alberts
Faculty of Mechanical Engineering, TUE, 1998-02

Scheduling with Communication for Multiprocessor Computation
J. Verriet
Faculty of Mathematics and Computer Science, UU, 1998-03

An Asynchronous Low-Power 80C51 Microcontroller
J. S. H. van Gageldonk
Faculty of Mathematics and Computing Science, TUE, 1998-04

In Terms of Nets: System Design with Petri Nets and Process Algebra
A. A. Basten
Faculty of Mathematics and Computing Science, TUE, 1998-05

Inductive Datatypes with Laws and Subtyping – A Relational Model
E. Voermans
Faculty of Mathematics and Computing Science, TUE, 1999-01