Algebraic Foundations of the Unifying Theories of Programming

Dissertation

zur Erlangung des Doktorgrades Dr. rer. nat.
der Fakultät für Ingenieurwissenschaften und Informatik
der Universität Ulm

Walter Guttmann

aus Großwardein

Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik
Institut für Programmiermethodik und Compilerbau
Direktor: Prof. Dr. Helmuth A. Partsch

2007

Amtierender Dekan:	Prof. Dr. Helmuth A. Partsch
Erster Gutachter:	Prof. Dr. Helmuth A. Partsch
Zweiter Gutachter:	Prof. Dr. Friedrich W. von Henke
Dritter Gutachter:	Prof. Dr. Bernhard Möller
Tag der Promotion:	5. Dezember 2007

Abstract

Hoare and He's *Unifying Theories of Programming* take a relational view on semantics. The meaning of a non-deterministic, imperative program is described by 'designs' composed of two relations. They represent terminating states and relate the initial and final values of the observable variables, respectively. Several 'healthiness conditions' are imposed by the theory to obtain properties found in practice.

This work determines the structure of designs and modifies the theory to support non-strict computations. It achieves these goals by identifying healthiness conditions and related axioms that involve unnecessary restrictions and subsequently removing them. The outcome provides a clear account of the algebraic foundations of the Unifying Theories of Programming.

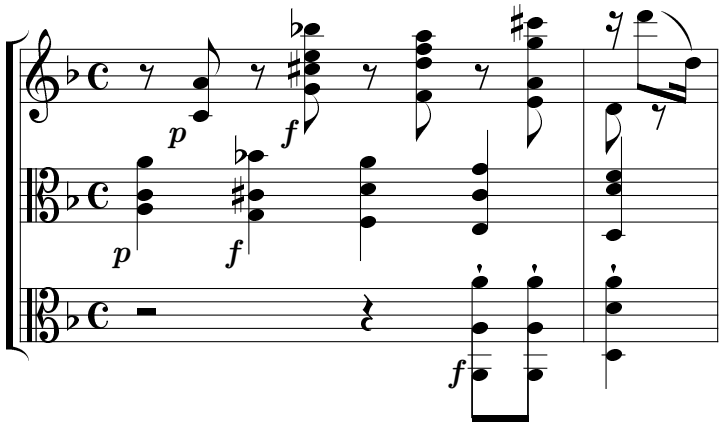
One of the results is a generalisation of designs by constructing them on semirings with ideals, structures having fewer axioms than relations. This clarifies the essential algebraic structure of designs, allows the reuse of existing mathematical theory and connects to further semantical approaches. The framework is extended by algebraic formulations of finite and infinite iteration, domain, pre-image, determinacy, invariants and convergence. Calculations and reasoning become simpler by the new, more abstract representation as is shown by applying the theory to investigate linear recursions.

Another result is an extension of the Unifying Theories of Programming to deal with undefined values irrespective of non-termination. Besides being closer to practice, it forms the basis of a new theory of relations representing non-strict computations. They satisfy additional healthiness conditions that model dependence in computations in an elegant algebraic form. Programs can then be executed according to the principle of lazy evaluation, otherwise known from functional programming languages.

Acknowledgements

Several individuals played various roles in the creation of this work. Here they are, in chronological order. I thank Wolfram Schulte and Ton Vullinghs who attracted me to (and then unfortunately left from) the department 'Programmiermethodik und Compilerbau', the place where this work has been produced. I thank Prof. Dr. Helmuth Partsch who supported my work by providing a very unconstrained environment at that place, and who brought me into contact with the next person. I thank Bernhard Möller whose delightful collaboration resulted in key works [GM06a, GM06b] for Chapter 2 that cannot be imagined without his and his collaborators' prior research. I thank Prof. Dr. Friedrich von Henke who agreed to act as the second referee of this work. I thank Marc Meister and Christian Ehrhardt for helpful remarks on the text.

Walter Guttmann


p
f
p
f
f

Contents

Chapter 1

Introduction

The *Unifying Theories of Programming* (UTP) developed in [HH98] provide a framework to describe and compare the semantics of specifications and programs of various paradigms using a common formalism. Its core is a relational, state-based model of non-deterministic, imperative programs. They are represented as special relations, so-called 'designs', combining transition and termination information. Although designs serve as a solid foundation for further theories, they impose unnecessary constraints and thus restrict the range of application. The topic of this work is to investigate what happens if these limits are removed.

We study two kinds of restrictions. Both have to do with the internal and external structure of designs. The internal structure is given by the components of designs, namely two relations modelling the terminating states and the transition behaviour of a computation. The external structure is given by the so-called 'healthiness conditions', laws that models of computations should satisfy to be useful.

Our first contribution generalises the component relations by replacing them with elements of semirings with ideals; the latter are constrained by fewer axioms. As a consequence, we obtain a more general structure also for designs, which nevertheless suffices to formulate and prove the healthiness conditions and other traditional and new properties. This clarifies the algebraic structure of designs, considerably simplifies reasoning about and calculating with designs, and allows the application of existing theories to UTP. Particular results we obtain are stated at the outset of Chapter 2 that contains this development.

Our second contribution extends UTP by the ability to represent variables and expressions with undefined values independently of non-termination. Such a finer distinction is necessary, for example, to reason about finite and infinite failure and to safely handle computations with undefined results. It also enables us to transfer lazy evaluation, known from functional programming languages, to the imperative context of UTP. However, we have to revoke one of the characterising healthiness conditions of designs and develop a new relational approach. This is carried out in Chapter 3 whose beginning provides a more detailed account.

Chapters 2 and 3 can be read and understood independently of each other. The reasoning in the former chapter is purely algebraic, hence abstract, but the intended model of imperative, non-deterministic programs is well-known. On the other hand, the latter chapter is more concrete by using relations, but the modelled non-strict computations might be less familiar. It is helpful for Chapter 2 and necessary for Chapter 3 to keep in mind the relational interpretation of programs as described in Sections 1.1 and 3.1.2. In particular, the reader is encouraged to *visualise* the occurring relations and their abstract counterparts, for example, as (infinite) matrices with Boolean entries.

Preceding the main development, we give an overview of those parts of UTP necessary to understand the context of this work. This concerns the non-deterministic, imperative core but not its extension to further paradigms such as concurrent, logic and object-oriented programming; that is treated, for example, in [HH01, LHLL04, HLL06, OCW07] and again [HH98]. The latter book also contains general motivation for the UTP approach.

1.1 Unifying theories of programming

The core of UTP deals with imperative programs, that is, those characterised by having state and statements. UTP defines a formal semantics of such programs by modelling them in mathematical terms. It is then possible, for example, to prove statements about programs and to calculate with programs just as with other mathematical objects.

The state of an imperative program is given by the values of its variables. A statement transforms a given state into a new state. We therefore distinguish between the values of the variables before and after execution of the statement. Consider a variable x: Its initial value is denoted just by x and its final value by x'. A statement or computation relates these values. Speaking mathematically we obtain a relation, that is, a subset of a Cartesian product. For example, the assignment $x := 3x + 1$ is such a relation, namely $\{(x, x') \mid x' = 3x + 1\}$. If x ranges over the natural numbers, this is just $\{(0, 1), (1, 4), (2, 7), (3, 10), (4, 13), \ldots\}$. This relation is in fact a mapping, because every x is related to exactly one x'.

Usually a program's state comprises more than one variable, say $x_1, x_2, \ldots, x_m$ which we abbreviate as $\vec{x}_{1..m}$ or $\vec{x}$. If e denotes an expression that may depend on the initial values $\vec{x}$, the assignment of e to x_1 is given by $(x_1 := e) =_{\text{def}} \{(\vec{x}_{1..m}, \vec{x}'_{1..m}) \mid x'_1 = e \wedge \vec{x}'_{2..m} = \vec{x}_{2..m}\}$. Simultaneous assignments such as $x_1, x_2 := x_2, x_1$ are treated similarly. A new variable x_{m+1} is introduced by the statement **var** $x_{m+1} =_{\text{def}} \{(\vec{x}_{1..m}, \vec{x}'_{1..m+1}) \mid \vec{x}'_{1..m} = \vec{x}_{1..m}\}$. Thus the state space is extended, but the new variable is not initialised. The converse **end** x_{m+1} removes that variable.

Assignment, variable declaration and undeclaration are basic statements; compound statements include the sequential composition, the conditional and the non-deterministic choice. They are described next. To state that Q is executed after P, we form the relational composition $P \,;\, Q =_{\text{def}} \{(\vec{x}, \vec{x}') \mid \exists \vec{y} : (\vec{x}, \vec{y}) \in P \wedge (\vec{y}, \vec{x}') \in Q\}$. The intermediate state $\vec{y}$ is hidden from outside observation. Sequential composition has a neutral element, namely the identity relation $\mathbb{I} =_{\text{def}} \{(\vec{x}, \vec{x}') \mid \vec{x}' = \vec{x}\}$ called *skip* denoting the empty statement or no-operation.

The conditional statement 'if b then P else Q' with relations b, P and Q is represented by the relation $(P \lhd b \rhd Q) =_{\text{def}} (b \cap P) \cup (\overline{b} \cap Q)$. Note the use of the set-theoretic operations $\cap$ and $\cup$ since relations are, in particular, sets. Observe furthermore the condition b which in contrast to P and Q is a special kind of relation, called *vector* in [SS89], that only depends on the initial values $\vec{x}$. It models the set of states that satisfy the condition b by relating every such state $\vec{x}$ to all available values for $\vec{x}'$, for example, $b = \{(\vec{x}, \vec{x}') \mid x_1 < 5\}$. Another way to model a set of states is given by *tests* in Section 2.2.1 but not used in original UTP.

The non-deterministic choice between statements P and Q is their union $P \cup Q$. For example, $(x := 3x + 1) \cup (x := x \div 2) = \{(x, x') \mid x' = 3x + 1 \vee x' = x \div 2\}$ models a program that yields either $3x + 1$ or $x \div 2$ for a given input x; this relation is not a mapping. Choice models underspecification by describing the possible observations that can be made about the execution of a program. Intuitively, $(\vec{x}, \vec{x}') \in P$ states that starting the program P with input $\vec{x}$ *may* produce output $\vec{x}'$. An implementation P satisfies a specification Q iff $P \subseteq Q$. This means that every observation one can make about P is admitted by Q. Since $\subseteq$ is transitive, this enables development by stepwise refinement.

Closing the gap to a general-purpose programming language, we finally add recursion. Two concrete examples are given in Section 2.2. As usual, recursion is described by fixpoints of isotone functions mapping relations to relations. These fixpoints exist by Tarski's theorem [Tar55] since relations form a complete lattice with respect to the subset order. Actually, the set of all fixpoints is a complete lattice itself and thus has a least and a greatest element. By choosing the *greatest* fixpoint, UTP models demonic non-determinism or total correctness. It follows that a non-terminating program, such as an endless loop, is modelled by the universal relation $\top\!\!\top =_{\text{def}} \{(\vec{x}, \vec{x}') \mid \mathit{true}\}$.

The theory as described so far is a bit too simplistic, since certain laws observed in practice for programs are not satisfied by arbitrary relations. For example, $\top\!\!\top \,;\, P = \top\!\!\top$ intuitively states that executing P after an endless loop has no effect. While this makes

sense, the equation is not satisfied by all relations; for a counterexample, take the empty relation $P = \bot\!\!\!\bot$. To rectify the situation one might, for example, redefine the sequential composition operator, but we appreciate a solution that retains its meaning as relational composition. The class of relations employed to describe programs is instead restricted by healthiness conditions; the obtained relations are called 'designs'. The latter is a technical term of UTP that must not be mistaken for the general notion used in software engineering.

1.2 Designs and healthiness conditions

A *design* is a relation $P \vdash Q$ composed of two relations P and Q with the help of two auxiliary Boolean variables ok and ok'. Intuitively, the *assumption* P represents the initial states where execution of the program is guaranteed to terminate, the *commitment* Q describes the possible transitions from such states, and ok and ok' model that the program has been started and has terminated, respectively. Thus Q corresponds to the relations introduced in Section 1.1, while P, ok and ok' extend that model. Formally, a design is

$$P \vdash Q =_{\text{def}} \{((ok, \vec{x}), (ok', \vec{x}')) \mid (ok \wedge (\vec{x}, \vec{x}') \in P) \Rightarrow (ok' \wedge (\vec{x}, \vec{x}') \in Q)\}.$$

It can be interpreted as follows: If the program starts and the assumption P holds, then the program terminates so that the commitment Q holds. The relation P thus acts as a precondition for the transition Q. In the general case, UTP allows P to involve both the initial and final values of the program variables, but normally P is a condition depending only on the initial values, see H3 below. Such a two-component description of the semantics of programming constructs also exists for applicative languages, see the *definedness* predicate and *breadth* function of [BGW79].

Two particular designs deserve to be mentioned. First, the identity $\mathbb{I}_{\text{D}} =_{\text{def}} \top \vdash \mathbb{I}$ has assumption $\top$, hence always terminates, and commitment $\mathbb{I}$, hence does not change the state. It is the skip or no-operation of designs. Second, the *loop* $\top_{\text{D}} =_{\text{def}} \bot\!\!\!\bot \vdash \bot\!\!\!\bot$ has assumption $\bot\!\!\!\bot$, hence never terminates; therefore its commitment actually does not matter. Of course, the other operations introduced in Section 1.1 must be lifted to designs as well. For example, the assignment design is $(x := e)_{\text{D}} =_{\text{def}} \mathcal{D}e \vdash (x := e)$ with the assumption $\mathcal{D}e$ that checks definedness of e. If e is undefined, this is just the same as non-termination; this issue is treated in Chapter 3 in more detail. The conditional is lifted similarly, while sequential composition, non-deterministic choice and recursion retain their meaning as relational operations, now on designs.

The reason why UTP considers designs is because they satisfy a number of properties expected from the modelled programs, as we will demonstrate below. However, the restriction to designs does not yield all of the desired algebraic laws. These are enforced by four *healthiness conditions* H1–H4 imposed on a relation R. Using the original predicate notation of [HH98, page 82] they are:

H1. $R = (ok \Rightarrow R)$

H2. $\big[R[\mathit{false}/ok'] \Rightarrow R[\mathit{true}/ok']\big]$

H3. $R = R \,;\, \mathbb{I}_{\text{D}}$

H4. $\top_{\text{D}} = R \,;\, \top_{\text{D}}$

The idea is to restrict the discourse to those relations that fulfil some or all of H1–H4. We shall not elaborate on the underlying intuition, which is done in [HH98], but quote three facts from that book.

* A relation R satisfies H1 iff it satisfies $\mathbb{I}_{\text{D}} \,;\, R = R$ and $\top_{\text{D}} \,;\, R = \top_{\text{D}}$.
* Designs are precisely the relations that satisfy H1 and H2.
* A design $P \vdash Q$ satisfies H3 iff P is a condition (or vector), that is, $P = P \,;\, \top$.

Therefore designs that satisfy H3 are also called *normal designs*, using terminology introduced by [Dun01]. The latter work also remarks that only H2 lacks a convincing algebraic

formulation. In the meantime this problem has been solved by observing that H3 actually implies H2; see, for example, [Gut06]. This gives another good reason to concentrate on normal designs.

Considering these facts, we are left with the following four, algebraically elegant restrictions with respect to sequential composition:

$$\begin{array}{ll} \text{H1a. } \mathbb{I}_{\mathrm{D}} \mathbin{;} R = R & \text{H3. } R = R \mathbin{;} \mathbb{I}_{\mathrm{D}} \\ \text{H1b. } \mathbb{T}_{\mathrm{D}} \mathbin{;} R = \mathbb{T}_{\mathrm{D}} & \text{H4. } \mathbb{T}_{\mathrm{D}} = R \mathbin{;} \mathbb{T}_{\mathrm{D}} \end{array}$$

Skip should be left- and right-neutral and loop should be left- and right-absorbing. Imposing such laws of course raises the question whether they are all required. Having a neutral element with respect to composition is useful, for example, to define a one-sided conditional and to terminate iterations. The law H4 is useful for implementation purposes, see also Section 3.3.4. Thus the question remains if H1b is needed; it is investigated in Chapter 3. There is another question implicit in these axioms: It is tacitly assumed that the variable R ranges over relations. The topic of Chapter 2 is to relax this requirement. We resume these questions in our conclusion in Chapter 4.

Chapter 2

The structure of designs

Our goal is to clarify the algebraic structure of UTP designs. This is achieved through generalisation, by defining designs as matrices over semirings with ideals. We apply our framework to investigate symmetric linear recursion and its relation to tail-recursion.

Preceding work [GM06a, Möl06] establishes a general algebraic treatment of designs and of the more liberal predicates known as *prescriptions* [Dun01] that reflect a general correctness view. This chapter is devoted to a simpler algebraic framework, that of *ideal semirings*, tailored to the particular case of (normal) designs. While still strictly more general than the original, purely relational UTP semantics, it exhibits the algebraic structure of designs more clearly and allows a much simpler derivation of the basic properties.

Particularly important is that the generalised normal designs again form an ideal semiring and even a weak Kleene and omega algebra. Moreover we show that they can be made into a test algebra and enriched by the modal operators diamond and box. For termination analysis we deal with the convergence and divergence operators that characterise the program states from which either no or at least one infinite computation is possible.

The benefits of our algebraic approach are also demonstrated by a number of investigations on special linear but non-tail recursions, hitherto not performed in UTP. Since these are carried out at the more abstract level of Kleene and omega algebras, they are not only valid for UTP but also for many other models. Furthermore algebraic formulations of determinacy, domain and invariants can be applied during this task.

In Section 2.1 we propose axioms for (normal) designs and show that they form a weak Kleene and omega algebra and an ideal semiring. In contrast to previous axiomatisations, the new axioms are based on the established ring-theoretic concept of ideals and perfectly suited for the calculation with designs using their matrix representation. The axioms together with the matrix representation allow a concise derivation of the Kleene star and omega operations for normal designs. We then generalise [HH98, Theorem 3.1.6] to our setting and use it to extend star and omega to general designs.

After the foundations have been laid, in Section 2.2 we show how to apply the framework of Kleene algebra and omega algebra to relate tail-recursion to linear recursion. By using the model of Section 2.1, our results are considerably more general than in plain UTP. By replacing conditions with tests, our calculations also become more simple. We moreover deal with both the least and the greatest fixpoints.

In Section 2.3 we apply our algebraic techniques to symmetric linear recursion. Our general framework, hence also UTP, is extended by algebraic notions of domain, pre-image, determinacy, invariants and convergence. We show how to implement both the recursion's least and greatest fixpoints by two consecutive while-loops each. We also derive that progressive boundedness and progressive finiteness coincide for deterministic elements.

This chapter is a revised and enhanced edition of those parts of [GM06b] that have been substantially developed by the present author, who acknowledges collaboration with Bernhard Möller.

2.1 Star and omega designs

In this section we derive the Kleene star and omega operations for generalised designs. While the result already appears in [GM06a], the present generalisation is based on a modified set of axioms, and the new derivation is considerably shortened by using the matrix representation of [Möl06]. The star and omega operations are important since they enable the application of general results derived algebraically to UTP, as shown in Sections 2.2 and 2.3.

Condition semirings have been introduced in [GM06a] to model the essence of normal designs. They are more general than the predicates or relations used in [HH98], that is, they impose fewer axioms. In Section 2.1.1 we propose a modified set of axioms that reflects the traditional nomenclature from ring theory, and investigate the connection to the former definition. We then define designs and normal designs as matrices of elements governed by our new axioms and point out why the new axiomatisation is adequate for this purpose.

Since it is well-known how to lift the Kleene star operation to matrix algebras [Koz94], the matrix model also lends itself to showing that normal designs form a weak Kleene algebra in Section 2.1.3. Using a similar lifting in Section 2.1.4, we show that normal designs also form a weak omega algebra. The Kleene star and omega operations are given explicitly. In Section 2.1.6 we generalise [HH98, Theorem 3.1.6] that describes the least and greatest fixpoints of functions on (not necessarily normal) designs. Our result is finally applied to derive the star and omega operations for these designs.

2.1.1 Axioms for conditions

In [GM06a] normal designs are modelled as commands over condition semirings, adapting the axioms of commands over test semirings studied in [MS06]. In the following, we base our axiomatisation on the established ring-theoretic concept of ideals, generalised to semirings as in [HW93].

Definition 1.

1. A *weak semiring* is a structure $(S, +, 0, \cdot, 1)$ such that
   - $(S, +, 0)$ is a commutative monoid,
   - $(S, \cdot, 1)$ is a monoid,
   - the operation $\cdot$ distributes over $+$ in both arguments and
   - 0 is a left annihilator, that is, $0 \cdot x = 0$.
2. A weak semiring is *idempotent* if $+$ is, that is, if $x + x = x$.
3. In an idempotent weak semiring the relation $x \leq y \Leftrightarrow_{\text{def}} x + y = y$ is a partial order, called the *natural order* on S.
4. A *semiring* is a weak semiring in which 0 is also a right annihilator, that is, $x \cdot 0 = 0$.

The $\cdot$ operation is extended elementwise to sets $A, B \subseteq S$ by $A \cdot B =_{\text{def}} \{a \cdot b \mid a \in A \wedge b \in B\}$ and $A \cdot b =_{\text{def}} A \cdot \{b\}$ for $b \in S$. We frequently omit the $\cdot$ notation and abbreviate $a \cdot b$ to ab.

In an idempotent weak semiring $+$ can be interpreted as (angelic) choice where 0 models the program with no transitions, and $\cdot$ as sequential composition where 1 models the program skip. The natural order has 0 as its least element. Moreover $+$ and $\cdot$ are isotone with respect to $\leq$ and $a + b$ is the least upper bound or *join* of a and b with respect to $\leq$. A prominent example of an idempotent semiring is $(A \leftrightarrow B, \cup, \perp\!\!\!\perp, ;, \mathbb{I})$ where $A \leftrightarrow B =_{\text{def}} \mathscr{P}(A \times B)$ are the relations between A and B, see also Section 3.1.2.2. Another example is $(A^*, \cup, \emptyset, \cdot, \{\varepsilon\})$, the formal languages over A with language concatenation $\cdot$ and empty word ε.

Because of the relational interpretation, we can model the commitment parts of designs by semiring elements. To model their assumption parts, we need special semiring elements that play the role of conditions. To this end, let us list some properties that are typical of conditions represented as vectors in the relational calculus:

1. The sequential composition of an arbitrary relation and a condition yields a condition again.
2. An arbitrary relation is input-restricted by conjoining it, that is, forming its *meet* or greatest lower bound, with a condition.
3. This restriction distributes over union in both arguments.
4. Restriction by the universal condition is no restriction at all.
5. Conditions can be used for Boolean reasoning, since they form a Boolean algebra.
6. Conditions are invariant under post-composition with the universal relation.

It turns out that these properties are sufficient for an abstract axiomatisation of conditions. Since for many results already properties 1–5 suffice, we first deal only with these; property 6 will be added later. Property 1 can be rephrased by saying that the set of all conditions is a left ideal of the semiring under consideration. This motivates the following definition.

Definition 2. A structure $(S, T, +, 0, \cdot, 1, \sqcap, \top, \overline{\ })$ is an *ideal semiring* iff the following properties hold.

- $(S, +, 0, \cdot, 1)$ is an idempotent weak semiring having a greatest element $\top$.
- T is a left ideal of S, that is,
  - $(T, +, 0)$ is a submonoid of $(S, +, 0)$ and
  - $S \cdot T \subseteq T$.
- The *restriction operation* $\sqcap : T \times S \to S$ distributes over $+$, that is,
  - $\forall a \in S : \forall t, u \in T : (t + u) \sqcap a = (t \sqcap a) + (u \sqcap a)$ and
  - $\forall a, b \in S : \forall t \in T : t \sqcap (a + b) = (t \sqcap a) + (t \sqcap b)$.
- $\forall a \in S : \top \sqcap a = a$.
- $(T, +, 0, \sqcap, \top, \overline{\ })$ is a Boolean algebra.

In the remainder we abbreviate ideal semiring structures to (S, T). An ideal semiring is *strict* if the underlying weak semiring is a semiring, that is, if 0 is both a left and a right annihilator.

Over an ideal semiring, the assumption part of a design will be taken from the set T and its commitment part from the set S. This makes sense because of the following instance: Let A and B be sets and $S =_{\text{def}} A \leftrightarrow B$, then the relations $(S, S \cdot \mathbb{T}, \cup, \perp\!\!\!\perp, ;, \mathbb{I}, \cap, \mathbb{T}, \overline{\ })$ form a (strict) ideal semiring with the vectors $S \cdot \mathbb{T}$ as left ideal. Another example of an ideal semiring is given in Corollary 9 by the normal designs. Both examples are even ideal condition semirings as defined below.

Consider an ideal semiring (S, T). Since T is a left ideal of S, it follows that T is also a subsemiring (without 1) of S. But T has another semiring structure, by virtue of being a Boolean algebra, with $+$ as addition and $\sqcap$ as composition. We can therefore relate T to the concept of a module from ring theory [HW93]. The following lemma also gives a few properties of restriction and shows that $t \sqcap a$ is the meet of t and a.

Lemma 3. *In an ideal semiring (S, T),*

1. *$\sqcap$ is isotone in both arguments,*
2. *$\sqcap$ is the greatest lower bound operation on $T \times S$,*
3. *the shunting rule $t \sqcap a \leq b \Leftrightarrow a \leq \overline{t} + b$ holds for all $t \in T$ and $a, b \in S$ and*
4. *S is a unitary left T-semimodule with scalar multiplication $\sqcap$.*

Proof. Let $a, b \in S$ and $t, u \in T$.

1. The claim follows immediately from the distributivity axioms of $\sqcap$.

2. First, $t \sqcap a \leq t \sqcap \top = t$ by part 1 and Boolean algebra. Second, $t \sqcap a \leq \top \sqcap a = a$ by part 1 and an axiom. Third, if $b \leq t$ and $b \leq a$, then $b = \top \sqcap b = (\overline{t} + t) \sqcap b = \overline{t} \sqcap b + t \sqcap b \leq \overline{t} \sqcap t + t \sqcap a = t \sqcap a$ by axioms, Boolean algebra and part 1.

3. The part ($\Rightarrow$) follows by $a = (\overline{t} + t) \sqcap a = \overline{t} \sqcap a + t \sqcap a \leq \overline{t} + b$ using part 2. The part ($\Leftarrow$) follows by $t \sqcap a \leq t \sqcap (\overline{t} + b) = t \sqcap \overline{t} + t \sqcap b \leq b$ using parts 1 and 2.

4. It remains to show the associative law $(t \sqcap u) \sqcap a = t \sqcap (u \sqcap a)$ since the distributive and unitary laws are already axioms. The calculations use parts 1 and 2 several times.

   * $(t \sqcap u) \sqcap a \leq t \sqcap u \leq t$ and $(t \sqcap u) \sqcap a \leq u \sqcap a$, hence $(t \sqcap u) \sqcap a \leq t \sqcap (u \sqcap a)$.
   * $t \sqcap (u \sqcap a) \leq t \sqcap u$ and $t \sqcap (u \sqcap a) \leq u \sqcap a \leq a$, hence $t \sqcap (u \sqcap a) \leq (t \sqcap u) \sqcap a$. $\square$

Thus the elements T of an ideal semiring (S, T) can be seen as acting on the elements of the semiring S. At the same time the associative law is typical of a restriction operation. As a consequence of Lemma 3.4, (S, T) may be characterised by stating that T is a Boolean algebra and a left ideal of S, and S is a unitary left T-semimodule, all with respect to the appropriate operations.

Ideal semirings are sufficient for representing designs, whereas for normal designs we need the subclass of ideal condition semirings. As stated above, in the relational calculus any condition t is invariant under post-composition with the universal element $\top$, that is, $t \cdot \top = t$. As will be shown below, it suffices to require that $t = u \cdot \top$ for some $u \in T$. This motivates the following definition.

Definition 4. An *ideal condition semiring* is an ideal semiring (S, T) where additionally $T \subseteq T \cdot \top$ holds. In this case the elements of T are called *conditions*. A semiring S with greatest element $\top$ is *replete* iff $S \cdot \top$ is a Boolean algebra.

Observe that the set $S \cdot \top \subseteq T$ consists of all elements that are invariant under right composition with $\top$. If $S \cdot \top$ is a Boolean algebra, it could be a proper subset of T and certainly is another candidate for the condition set of an ideal semiring over S. In the context of Kleene algebra with tests, one similarly considers taking all elements ≤ 1 or just a Boolean subalgebra of them as tests [Koz97]. We show below that such a question does not arise here since $S \cdot \top$ coincides with the full set T of conditions, hence the name *replete*. In [GM06a] the now inappropriate term 'ideal-closed' is used.

Lemma 5. *Let (S, T) be an ideal condition semiring.*

1. *$S \cdot \top = T$, hence S is replete.*

2. *$\forall t \in T : t \cdot \top = t$.*

3. *$\forall t \in T : \forall a, b \in S : (t \sqcap a) \cdot b = t \sqcap (a \cdot b)$.*

Proof. Let $a, b \in S$ and $t \in T$.

1. $S \cdot \top \subseteq S \cdot T \subseteq T \subseteq T \cdot \top \subseteq S \cdot \top$, hence $S \cdot \top = T$ is a Boolean algebra.

2. Since $T \subseteq T \cdot \top$, there is a $u \in T$ such that $t = u \cdot \top$. Thus $t \cdot \top = u \cdot \top \cdot \top = u \cdot \top = t$, using $\top \cdot \top \geq \top \cdot 1 = \top$.

3. The part ($\leq$) follows by isotony since $(t \sqcap a) \cdot b \leq a \cdot b$ and $(t \sqcap a) \cdot b \leq t \cdot b \leq t \cdot \top = t$ using Lemma 3.2 and part 2. As a consequence, the part ($\geq$) follows by Boolean algebra since $t \sqcap (a \cdot b) = t \sqcap ((t \sqcap a) \cdot b + (\overline{t} \sqcap a) \cdot b) \leq t \sqcap ((t \sqcap a) \cdot b + \overline{t} \sqcap (a \cdot b)) = t \sqcap ((t \sqcap a) \cdot b) \leq (t \sqcap a) \cdot b$. $\square$

Therefore, and in contrast to the condition semirings of [GM06a], every ideal condition semiring is already replete. The cause for this restriction is the axiom $S \cdot T \subseteq T$, since the other prerequisites of Lemma 5 also hold in condition semirings. As we will point out in Section 2.1.2, however, this axiom is necessary for the representation of normal designs as matrices and, subsequently, to obtain a Kleene and omega algebra. Nevertheless, the new axioms are less restrictive than those in [Möl06] since they do not require S to be a Boolean semiring.

2.1.2 Designs and normal designs as matrices

We define (normal) designs as 2×2 matrices over a weak semiring, similarly to [Möl06]. The difference is that we do not demand a Boolean semiring but take the elements from an ideal semiring. Our goal is to lift the semiring structure to designs which is a prerequisite for further structures introduced in the following sections.

The matrix representation of designs eliminates the auxiliary variables ok and ok' that are introduced to deal with non-termination in [HH98]. It is motivated by the following observation. Let $\mathbb{B} =_{\text{def}} \{\mathit{false}, \mathit{true}\}$ denote the Boolean values, which ok and ok' can take. A design R is a relation $R : \mathbb{B} \times X \leftrightarrow \mathbb{B} \times X'$, where the sets X and X' represent the possible states before and after the execution of the computation modelled by R. Let

$$R(ok, ok') =_{\text{def}} \{(x, x') \mid ((ok, x), (ok', x')) \in R\}$$

be the residual relation with respect to $ok, ok' \in \mathbb{B}$. This corresponds to a restriction or selection followed by projection in the formalism of [Cod70]. Hence the design R may be represented by four residual relations, one for each combination of values of ok and ok', arranged in a matrix:

$$R = \begin{pmatrix} R(\mathit{false}, \mathit{false}) & R(\mathit{false}, \mathit{true}) \\ R(\mathit{true}, \mathit{false}) & R(\mathit{true}, \mathit{true}) \end{pmatrix}.$$

Operations on designs thus reduce to standard matrix operations; reasoning is completely component-free, without ok and ok' and variable substitutions. The matrix representation is further elaborated in [Möl06].

Designs are characterised in [HH98] by the healthiness conditions H1 and H2. They translate to the matrix representation as follows. A relation R satisfies H1 iff both residuals in the top row of R's matrix are the universal relation. It satisfies H2 iff in both rows of its matrix the left residual is a subset of the right one. This motivates our definition of designs in the abstract setting, where we use 2×2 matrices with elements from an ideal semiring S as entries.

Definition 6. Let (S, T) be an ideal semiring. The set of *designs* over (S, T) is

$$\mathrm{D}(S,T) =_{\text{def}} \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in S^{2\times 2} \;\middle|\; a = b = \top \;\wedge\; c \in T \;\wedge\; c \leq d \right\}.$$

For $t \in T$ and $a \in S$, we define the *design*

$$t \vdash a =_{\text{def}} \begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} + a \end{pmatrix}.$$

If, additionally, (S, T) is an ideal condition semiring, we call its designs *normal* and set $\mathrm{ND}(S,T) =_{\text{def}} \mathrm{D}(S,T)$.

From these definitions it follows that *every* design can be denoted in abbreviated form, since

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{D}(S,T) \Rightarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \overline{c} \vdash d.$$

Using matrix addition and multiplication, we can lift the semiring structure to normal designs as the following theorem shows. While this is clear for the matrix semiring $S^{2\times 2}$, we have to check the restrictions imposed by designs. Observe that the left ideal property is crucial for the totality of $\cdot$ and that a strict ideal condition semiring is needed for the right unit law.

Theorem 7. *Let (S,T) be a strict ideal condition semiring. Then the structure of normal designs $(\mathrm{ND}(S,T),+,\top\vdash 0,\cdot,\top\vdash 1)$ is an idempotent weak semiring.*

Proof. Let $a,b\in S$ and $t,u\in T$ such that $t\le a$ and $u\le b$.

* $+$ is total since $t+u\in T$ and $t+u\le a+b$ and
$$\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}+\begin{pmatrix}\top & \top\\ u & b\end{pmatrix}=\begin{pmatrix}\top & \top\\ t+u & a+b\end{pmatrix}.$$
* $+$ is associative, commutative and idempotent since it is defined componentwise.
* $\top\vdash 0$ is neutral with respect to $+$ since
$$\begin{pmatrix}\top & \top\\ 0 & 0\end{pmatrix}+\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}=\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}=\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}+\begin{pmatrix}\top & \top\\ 0 & 0\end{pmatrix}.$$
* $\cdot$ is total since $t\top+au=t+au\in T$ by Lemma 5.2 and $t+au\le t+ab$ and
$$\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}\cdot\begin{pmatrix}\top & \top\\ u & b\end{pmatrix}=\begin{pmatrix}\top\top+\top u & \top\top+\top b\\ t\top+au & t\top+ab\end{pmatrix}=\begin{pmatrix}\top & \top\\ t+au & t+ab\end{pmatrix}.$$
* $\cdot$ is associative since matrix multiplication is associative.
* $\top\vdash 1$ is neutral with respect to $\cdot$ since
$$\begin{pmatrix}\top & \top\\ 0 & 1\end{pmatrix}\cdot\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}=\begin{pmatrix}\top & \top\\ 0+1t & 0+1a\end{pmatrix}=\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}$$
and, using $a0=0$,
$$\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}\cdot\begin{pmatrix}\top & \top\\ 0 & 1\end{pmatrix}=\begin{pmatrix}\top & \top\\ t+a0 & t+a1\end{pmatrix}=\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}.$$
* $\cdot$ distributes over $+$ since it does so for matrices.
* $\top\vdash 0$ is a left annihilator of $\cdot$ since
$$\begin{pmatrix}\top & \top\\ 0 & 0\end{pmatrix}\cdot\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}=\begin{pmatrix}\top & \top\\ 0+0t & 0+0a\end{pmatrix}=\begin{pmatrix}\top & \top\\ 0 & 0\end{pmatrix}.$$ □

Remark. With two modifications, Theorem 7 generalises to designs over ideal semirings: The composition of designs is the more verbose
$$\begin{pmatrix}\top & \top\\ t & a\end{pmatrix}\cdot\begin{pmatrix}\top & \top\\ u & b\end{pmatrix}=\begin{pmatrix}\top & \top\\ t\top+au & t\top+ab\end{pmatrix},$$
and the right unit law fails. (End of remark)

This theorem and its following corollary shows that normal designs behave just as expected from [HH98], also in their abbreviated forms. Additionally, the natural order of normal designs also makes sense for designs. Observe that it reflects the implication order and not the refinement order of UTP, which is the reverse.

Corollary 8. *The natural order of (normal) designs is*

$$t \vdash a \leq u \vdash b \quad \Leftrightarrow \quad u \leq t \ \wedge \ u \sqcap a \leq b \quad \Leftrightarrow \quad u \leq t \ \wedge \ a \leq \overline{u} + b.$$

Moreover,

$$t \vdash a = u \vdash b \quad \Leftrightarrow \quad t = u \ \wedge \ t \sqcap a = u \sqcap b$$

and

$$t \vdash a \ = \ t \vdash \overline{t} + a \ = \ t \vdash t \sqcap a.$$

The composition of designs is $(t \vdash a) \cdot (u \vdash b) = \overline{\overline{t}\top + a\overline{u}} \vdash ab$ *which simplifies to* $t \sqcap \overline{a\overline{u}} \vdash ab$ *for normal designs. The sum of designs is* $(t \vdash a) + (u \vdash b) = t \sqcap u \vdash a + b$.

Proof. Let $a, b \in S$ and $t, u \in T$.

- By the componentwise matrix order and the shunting rule of Lemma 3.3,

$$\begin{aligned} t \vdash a \leq u \vdash b \quad &\Leftrightarrow \quad \begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} + a \end{pmatrix} \leq \begin{pmatrix} \top & \top \\ \overline{u} & \overline{u} + b \end{pmatrix} \\ &\Leftrightarrow \quad \overline{t} \leq \overline{u} \ \wedge \ \overline{t} + a \leq \overline{u} + b \\ &\Leftrightarrow \quad \overline{t} \leq \overline{u} \ \wedge \ \overline{t} \leq \overline{u} + b \ \wedge \ a \leq \overline{u} + b \\ &\Leftrightarrow \quad u \leq t \ \wedge \ u \sqcap a \leq b. \end{aligned}$$

  Therefore,

$$\begin{aligned} t \vdash a = u \vdash b \quad &\Leftrightarrow \quad t \vdash a \leq u \vdash b \ \wedge \ u \vdash b \leq t \vdash a \\ &\Leftrightarrow \quad u \leq t \ \wedge \ u \sqcap a \leq b \ \wedge \ t \leq u \ \wedge \ t \sqcap b \leq a \\ &\Leftrightarrow \quad t = u \ \wedge \ t \sqcap a = u \sqcap b. \end{aligned}$$

  From this, $t \vdash a \ = \ t \vdash \overline{t} + a \ = \ t \vdash t \sqcap a$ follows immediately.

- The sum of designs is given by

$$\begin{aligned} (t \vdash a) + (u \vdash b) \ &= \ \begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} + a \end{pmatrix} + \begin{pmatrix} \top & \top \\ \overline{u} & \overline{u} + b \end{pmatrix} \ = \ \begin{pmatrix} \top & \top \\ \overline{t} + \overline{u} & \overline{t} + \overline{u} + a + b \end{pmatrix} \\ &= \ \overline{\overline{t} + \overline{u}} \vdash a + b \ = \ t \sqcap u \vdash a + b. \end{aligned}$$

- The composition of designs is given by

$$\begin{aligned} (t \vdash a) \cdot (u \vdash b) \ &= \ \begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} + a \end{pmatrix} \cdot \begin{pmatrix} \top & \top \\ \overline{u} & \overline{u} + b \end{pmatrix} \\ &= \ \begin{pmatrix} \top & \top \\ \overline{t}\top + (\overline{t} + a)\overline{u} & \overline{t}\top + (\overline{t} + a)(\overline{u} + b) \end{pmatrix} \\ &= \ \begin{pmatrix} \top & \top \\ \overline{t}\top + a\overline{u} & \overline{t}\top + a\overline{u} + ab \end{pmatrix} \ = \ \overline{\overline{t}\top + a\overline{u}} \vdash ab, \end{aligned}$$

  and $\overline{\overline{t}\top + a\overline{u}} = t \sqcap \overline{a\overline{u}}$ for normal designs. □

For the remainder of this section and the three following ones, we restrict our attention to normal designs, assuming a strict ideal condition semiring. In Section 2.1.6 we return to the more general case of designs over an ideal semiring.

We can also lift the ideal condition semiring structure to normal designs, which will be useful to represent UTP-conditions as tests in Section 2.2. The $\sqcap$ operation is lifted componentwise and the lifted $\overline{\ }$ acts on the assumption only, as detailed in the proof.

Corollary 9. *Let* (S, T) *be a strict ideal condition semiring and*

$$C =_{\text{def}} \{t \vdash 0 \mid t \in T\} = \left\{ \begin{pmatrix} \top & \top \\ t & t \end{pmatrix} \;\middle|\; t \in T \right\}.$$

Then the structure $(\mathrm{ND}(S, T), C, +, \top \vdash 0, \cdot, \top \vdash 1, \sqcap, 0 \vdash 0, \overline{\ })$ *is an ideal condition semiring.*

Proof. Theorem 7 shows that $\mathrm{ND}(S,T)$ forms an idempotent weak semiring. Its greatest element is $0 \vdash 0$ by Corollary 8. It is easily calculated that C is a submonoid of $\mathrm{ND}(S,T)$, and the left ideal property follows since

$$(t \vdash a) \cdot (u \vdash 0) = t \sqcap \overline{a\overline{u}} \vdash a0 = t \sqcap \overline{a\overline{u}} \vdash 0.$$

As a special case, we obtain the condition property $(t \vdash 0) \cdot (0 \vdash 0) = t \vdash 0$. We define the restriction $\sqcap$ as the componentwise restriction on the matrix representation:

$$\begin{pmatrix} \top & \top \\ t & t \end{pmatrix} \sqcap \begin{pmatrix} \top & \top \\ u & b \end{pmatrix} =_{\mathrm{def}} \begin{pmatrix} \top & \top \\ t \sqcap u & t \sqcap b \end{pmatrix}.$$

Immediate consequences are distributivity over $+$ and neutrality of the universal condition $0 \vdash 0$. The Boolean algebra structure also follows using the complement

$$\overline{\overline{t} \vdash 0} = \overline{\begin{pmatrix} \top & \top \\ t & t \end{pmatrix}} =_{\mathrm{def}} \begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} \end{pmatrix} = t \vdash 0. \qquad \square$$

2.1.3 Star designs

We show that normal designs form a weak Kleene algebra. The Kleene star operation is useful to give closed representations of finite iterations and simplifies calculations involving such iterations. It will be used in Sections 2.2 and 2.3 for this purpose. That normal designs have a star entails that the results of these sections are applicable to UTP.

Definition 10. A *Kleene algebra* is a structure $(S, {}^*)$ such that S is an idempotent semiring and the operation star * satisfies the unfold and induction laws

$$\begin{array}{ll} 1 + a \cdot a^* \leq a^* & \qquad b + a \cdot c \leq c \Rightarrow a^* \cdot b \leq c \\ 1 + a^* \cdot a \leq a^* & \qquad b + c \cdot a \leq c \Rightarrow b \cdot a^* \leq c \end{array}$$

for $a, b, c \in S$ [Koz94]. In a *weak* Kleene algebra, S is only required to be an idempotent weak semiring.

It follows from these axioms that a^*b is the least fixpoint of the function $\lambda x.ax + b$ and that the operation * is isotone with respect to the natural order. Moreover the unfold law can be strengthened to an equality $1 + aa^* = a^*$. Further consequences are $a^*a^* = a^*$ and $a(ba)^* = (ab)^*a$. Examples of Kleene algebras are again the relations where R^* is the reflexive transitive closure of R, and the formal languages where $L^* = \bigcup_{n \in \mathbb{N}} L^n$ is the finitely iterated language concatenation.

Remark. A related example is formed by path sets in a directed graph where $\cdot$ is path concatenation. The relation, language and path algebras can be used to derive graph and pointer algorithms from high-level specifications [Möl93]. In the context of graph theory also other Kleene algebras arise, for example, to calculate the shortest paths between all pairs of nodes using the Floyd-Warshall algorithm. The underlying instance is $(\mathbb{R} \cup \{\pm\infty\}, \min, \infty, +, 0, {}^*)$ where a^* is $-\infty$ for negative a and 0 otherwise. Further examples and a framework for such problems are given in [CLR90] using a variant called *closed semiring*; see [Koz90] for the slight difference to Kleene algebra. (End of remark)

There is also a generic way to build Kleene algebras from given ones, namely to form matrices with elements as entries. The star operation can be lifted to matrices by this standard construction. We take the version presented in [ÉL05], similar definitions appear in [Con71, Koz94]. By iterative application these constructions can handle $n \times n$ matrices, but we only need the case $n = 2$.

Definition 11. The *Kleene star* of a 2×2 matrix is given by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^* =_{\mathrm{def}} \begin{pmatrix} f^* & f^*bd^* \\ e^*ca^* & e^* \end{pmatrix},$$

where $f = a + bd^*c$ and $e = d + ca^*b$.

The Kleene star of a normal design hence is

$$\begin{pmatrix} \top & \top \\ t & a \end{pmatrix}^* = \begin{pmatrix} \top^* & \top a^* \\ a^*t\top & a^* \end{pmatrix} = \begin{pmatrix} \top & \top \\ a^*t & a^* \end{pmatrix},$$

since $f = \top + \top a^* t = \top$ and $e = a + t\top^*\top = a + t = a$ and $\top a^* \geq \top 1 = \top$. The result is a normal design since $t \in T \Rightarrow a^*t \in T$ and $t \leq a \Rightarrow a^*t \leq a^*a \leq a^*$. Observe that the left ideal property is crucial again. We therefore obtain the following result.

Theorem 12. *Let (S, T) be an ideal condition semiring such that S is a Kleene algebra. Then the structure $(\mathrm{ND}(S,T), +, \top \vdash 0, \cdot, \top \vdash 1, ^*)$ is a weak Kleene algebra.*

Proof. After Theorem 7, it remains to show that the star unfold and induction axioms are satisfied. But this follows, since they are valid in the encompassing full matrix algebra and $\mathrm{ND}(S,T)$ is closed under star. □

The benefit of deriving the star operation via the matrix construction is manifest when compared to the previous approach in [GM06a] where the operation had to be 'guessed' and verified by an extensive proof. We finally derive the Kleene star of a normal design in the abbreviated representation used by [HH98]. As a prerequisite we prove the following distribution lemma.

Lemma 13. *Consider an ideal condition semiring (S, T) such that S is a Kleene algebra and let $a \in S$ and $t \in T$.*

1. $(t + a)^* = a^*t + a^*$.
2. $(t + a)^*t = a^*t$.

Proof.

1. First, we have $1 + (t + a)(a^*t + a^*) \leq 1 + t\top + a(a^*t + a^*) = t + aa^*t + a^* = a^*t + a^*$, and therefore $(t+a)^* \leq a^*t+a^*$ by star induction. Second, by isotony and star unfold, $a^*t + a^* \leq (t + a)^*(t + a) + (t + a)^* \leq (t + a)^*$.
2. By part 1, we have $(t + a)^*t = (a^*t + a^*)t \leq a^*t\top + a^*t = a^*t$. The other inequality $a^*t \leq (t + a)^*t$ follows by isotony of star. □

Corollary 14. *Let (S, T) be an ideal condition semiring such that S is a Kleene algebra. Let $t \vdash a$ be a normal design over (S, T). Then $(t \vdash a)^* = (\overline{a^*\bar{t}} \vdash a^*)$.*

Proof. By Theorem 12 and Lemma 13,

$$(t \vdash a)^* = \begin{pmatrix} \top & \top \\ \bar{t} & \bar{t} + a \end{pmatrix}^* = \begin{pmatrix} \top & \top \\ (\bar{t} + a)^*\bar{t} & (\bar{t} + a)^* \end{pmatrix} = \begin{pmatrix} \top & \top \\ a^*\bar{t} & a^*\bar{t} + a^* \end{pmatrix} = \overline{a^*\bar{t}} \vdash a^*. \qquad \square$$

2.1.4 Omega designs

We show that normal designs form a weak omega algebra. The omega operation together with the Kleene star allows a closed representation of infinite iterations and of certain greatest fixpoints. Corresponding results of Sections 2.2 and 2.3 are applicable to UTP due to the fact that normal designs have an omega.

Definition 15. An *omega algebra* is a structure $(S, ^\omega)$ such that S is a Kleene algebra and the operation omega $^\omega$ satisfies the unfold and co-induction laws

$$a^\omega \leq a \cdot a^\omega \qquad\qquad c \leq a \cdot c + b \Rightarrow c \leq a^\omega + a^* \cdot b$$

for $a, b, c \in S$ [Coh00]. In a *weak* omega algebra, S is only required to be a weak Kleene algebra, but the unfold law is strengthened to $a^\omega = a \cdot a^\omega$ since the inequality $(\geq)$ need not hold in absence of the right annihilation axiom [Möl04].

It follows from these axioms that $a^\omega + a^*b$ is the greatest fixpoint of the function $\lambda x.ax+b$ and that $^\omega$ is isotone. Moreover $a^*a^\omega = a^\omega$ holds. For the special case $b = 0$ we therefore obtain $c \leq ac \Rightarrow c \leq a^\omega + a^*0 = a^*a^\omega + a^*0 = a^*a^\omega = a^\omega$, hence a^ω is the greatest fixpoint of $\lambda x.ax$. Furthermore there exists a greatest element $\top = 1^\omega$ with $a^\omega = a^\omega\top$ for each $a \in S$.

An example of an omega algebra are again the relations where the vector R^ω characterises those points from which an infinite path of transitions under R emerges. The formal languages also form an omega algebra, but L^ω is either the set of all elements or empty, depending on whether $\varepsilon \in L$ or not. This becomes more useful if one includes as elements the infinite words over the base set.

As with Kleene algebras, we can form matrices with elements of omega algebras as entries. The omega operation can be lifted to matrices by another construction, presented in [MD06]. It also can handle $n \times n$ matrices by iterative application, but again we only need the case $n = 2$.

Definition 16. The *omega* of a 2×2 matrix is given by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^\omega =_{\text{def}} \begin{pmatrix} f^\omega + a^*be^\omega & f^\omega + a^*be^\omega \\ d^*cf^\omega + e^\omega & d^*cf^\omega + e^\omega \end{pmatrix},$$

where $f = a + bd^*c$ and $e = d + ca^*b$ as in Definition 11.

The omega of a normal design hence is

$$\begin{pmatrix} \top & \top \\ t & a \end{pmatrix}^\omega = \begin{pmatrix} \top^\omega + \top a^\omega & \top^\omega + \top a^\omega \\ a^*t\top^\omega + a^\omega & a^*t\top^\omega + a^\omega \end{pmatrix} = \begin{pmatrix} \top & \top \\ a^\omega + a^*t & a^\omega + a^*t \end{pmatrix},$$

since, as before, $f = \top$ and $e = a$. The result is a normal design since $t \in T \Rightarrow a^\omega + a^*t = a^\omega\top + a^*t \in T$. Observe that the left ideal property is crucial again. We therefore obtain the following result.

Theorem 17. *Let (S, T) be an ideal condition semiring such that S is an omega algebra. Then the structure $(\mathrm{ND}(S, T), +, \top \vdash 0, \cdot, \top \vdash 1, ^*, ^\omega)$ is a weak omega algebra.*

Proof. After Theorem 12, it remains to show that the omega co-induction and unfold axioms are satisfied. But this follows, since they are valid in the encompassing full matrix algebra and $\mathrm{ND}(S, T)$ is closed under omega. □

Again this may be compared to the previous approach to see the advantage, as pointed out in Section 2.1.3. We finally derive the omega of a normal design in the abbreviated representation. As a prerequisite we prove the following distribution lemma.

Lemma 18. *Consider an ideal condition semiring (S, T) such that S is an omega algebra and let $a \in S$ and $t \in T$. Then $(t + a)^\omega + (t + a)^*t = a^\omega + a^*t$.*

Proof. The part $(\geq)$ is immediate by isotony. For $(\leq)$, after application of Lemma 13.2 it suffices to show $(t + a)^\omega \leq a^\omega + a^*t$. But this follows by omega co-induction from $(t + a)^\omega \leq (t + a)(t + a)^\omega \leq t\top + a(t + a)^\omega = t + a(t + a)^\omega$. □

Corollary 19. *Let (S, T) be an ideal condition semiring such that S is an omega algebra. Let $t \vdash a$ be a normal design over (S, T). Then $(t \vdash a)^\omega = (\overline{a^\omega + a^*\bar{t}} \vdash 0)$.*

Proof. By Theorem 17 and Lemma 18,

$$\begin{pmatrix} \top & \top \\ \bar{t} & \bar{t} + a \end{pmatrix}^\omega = \begin{pmatrix} \top & \top \\ (\bar{t} + a)^\omega + (\bar{t} + a)^*\bar{t} & (\bar{t} + a)^\omega + (\bar{t} + a)^*\bar{t} \end{pmatrix} = \begin{pmatrix} \top & \top \\ a^\omega + a^*\bar{t} & a^\omega + a^*\bar{t} \end{pmatrix}.$$ □

2.1.5 UTP algebras

As we have seen in the previous sections, normal designs form an idempotent weak semiring, a weak Kleene algebra and a weak omega algebra. The qualifier 'weak' means that the right annihilation law $\forall x : x \cdot 0 = 0$ is not required to hold. In a semiring with greatest element $\top$ this axiom may be restated as $\top \cdot 0 = 0$ by isotony.

As stated in Corollary 9, (normal) designs have the greatest element $\top_{\mathrm{D}} =_{\mathrm{def}} 0 \vdash 0$ that corresponds to the relation $\mathbb{T}_{\mathrm{D}}$ of Section 1.2 in the instance of UTP. The least element $0_{\mathrm{D}} =_{\mathrm{def}} \top \vdash 0$ is not a right annihilator of designs; indeed $\top_{\mathrm{D}} \cdot 0_{\mathrm{D}} = \top_{\mathrm{D}}$ since

$$(0 \vdash 0) \cdot (\top \vdash 0) \;=\; \overline{\overline{0} \cdot \top + 0 \cdot \overline{\top}} \vdash 0 \cdot 0 \;=\; 0 \vdash 0.$$

Omitting the right annihilation law gives us the freedom to impose this left annihilation law instead. A theory without a left absorbing greatest element is investigated in Chapter 3.

Definition 20. A *UTP semiring/Kleene algebra/omega algebra* is a weak semiring/Kleene algebra/omega algebra with greatest element $\top$ such that $\top \cdot 0 = \top$ holds or, equivalently, $\forall x : \top \cdot x = \top$.

An immediate consequence is that normal designs form a UTP omega algebra. The axiom $\top \cdot 0 = \top$ is typical of total correctness frameworks, not only of UTP. For example, it also holds in the demonic refinement algebras of [Wri04]. Actually the latter are equivalent to UTP omega algebras as is shown by [HMS06] under the name of *top-left-strict weak omega algebras*. Once $\top$ is a left annihilator, further elements are, too. In the case of normal designs they include the conditions C of Corollary 9.

Lemma 21.

1. *Let a be an element of a UTP omega algebra, then a^ω is a left annihilator.*
2. *Let (S, T) be a strict ideal condition semiring and $a \in S$ and $t \in T$ such that $a \leq \overline{t}$. Then $t \vdash a$ is a left annihilator. In particular, $t \vdash \overline{t}$ and $t \vdash 0$ are left annihilators.*

Proof.

1. $a^\omega x = a^\omega \top x = a^\omega \top = a^\omega$.
2. By Corollary 8, $t \vdash a = t \vdash t \sqcap a = t \vdash 0$ and $(t \vdash 0) \cdot (u \vdash b) = t \sqcap \overline{0\overline{u}} \vdash 0b = t \vdash 0$. □

Remark. In UTP the axiom $\top \cdot x = \top$ is a consequence of the healthiness condition H1, called H1b in Section 1.2. Together with this axiom, Theorem 7 shows that our generalised normal designs actually satisfy H1–H3. A simple calculation shows that the healthiness condition H4 requiring totality reduces to $t \leq a\top$ for a normal design $t \vdash a$. On the one hand, the subset of total designs is interesting for computation purposes, see also Section 3.3.4. On the other hand, it is possible to set up a correspondence between total normal designs and the elements of the underlying semiring S [GM06a]. This connection abstractly expresses a previous result of [Gut06] and is used to derive demonic programming operators and a demonic refinement order on S that reflects total correctness. Using the star and omega operations of Sections 2.1.3 and 2.1.4, closed representations of the least and greatest fixpoints of the demonic while-loop on S are calculated, too. (End of remark)

2.1.6 Fixpoints and designs

In Sections 2.1.3 and 2.1.4 we have shown how to calculate the least and the greatest fixpoints of the iteration function $\lambda x.ax + b$ on normal designs. We now use our algebraic techniques to extend this by considering all designs instead of just normal designs, and by investigating fixpoints of the more general function

$$H(P \vdash Q) =_{\mathrm{def}} F(P \vdash Q) \vdash G(P \vdash Q)$$

on designs. Its greatest fixpoint is described in [HH98, Theorem 3.1.6]. We additionally deal with the least fixpoint and generalise that result to our setting, that is, we do not assume that P and Q are relations. Instead, we only require that $P \vdash Q$ is a design over an ideal semiring and that certain fixpoints exist. Our treatment is based on the following axioms for fixpoints, see [DMT06].

Definition 22. Let f be a function on a partial order. An element a is a *fixpoint* of f iff it satisfies the fixpoint law $f(a) = a$. The element μf is the *least prefixpoint* of f iff the following unfold and induction laws hold:

$$f(\mu f) \leq \mu f \qquad \forall x : f(x) \leq x \Rightarrow \mu f \leq x$$

The element νf is the *greatest postfixpoint* of f iff the following unfold and co-induction laws hold:

$$\nu f \leq f(\nu f) \qquad \forall x : x \leq f(x) \Rightarrow x \leq \nu f$$

We abbreviate $\mu(\lambda x.f(x))$ by $\mu x.f(x)$ and $\nu(\lambda x.f(x))$ by $\nu x.f(x)$.

In this chapter, by writing μf and νf we assume that these elements exist. If f is isotone and the underlying partial order is complete, this is a consequence of Tarski's fixpoint theorem [Tar55]. We furthermore recall the following facts about fixpoints from [DMT06]. If f is isotone, then μf is the least fixpoint of f, and νf the greatest. If f and g are isotone and $f \leq g$ it follows that $\mu f \leq \mu g$ and $\nu f \leq \nu g$. If the partial order is a Boolean algebra with complement $\neg$ it follows that $\mu f = \neg\nu x.\neg f(\neg x)$ and $\nu f = \neg\mu x.\neg f(\neg x)$.

2.1.6.1 The greatest fixpoint

In the remainder of this section, let $H(t \vdash a) =_{\text{def}} F(t \vdash a) \vdash G(t \vdash a)$ be an isotone function of designs over an ideal semiring (S, T) such that $F(t \vdash a) \in T$ for all $t \in T$ and $a \in S$. The following definitions of P, R and Q are based on [HH98]. Note that μ and ν are swapped relative to the original definitions, since we use the implication order and not the refinement order.

Definition 23.

1. Define $P : S \to T$ by $P(a) =_{\text{def}} \mu t.F(t \vdash a)$.
2. Define $R : S \to S$ by $R(a) =_{\text{def}} \overline{P(a)} + G(P(a) \vdash a)$.
3. Define $Q =_{\text{def}} \nu R$.

We first prove a few isotony statements supporting the existence of the used fixpoints. If T is complete, then $P(a)$ and hence $R(a)$ exist by Lemma 24.1. If additionally S is complete, then Q exists by Lemma 24.3. Afterwards we generalise [HH98, Theorem 3.1.6] to our setting.

Lemma 24.

1. *Let $a, b \in S$ and $t, u \in T$ such that $a \leq b$ and $u \leq t$. Then $F(u \vdash b) \leq F(t \vdash a)$ and $F(u \vdash b) \sqcap G(t \vdash a) \leq G(u \vdash b)$. In particular, $\lambda t.F(t \vdash a)$ is isotone and $\lambda a.F(t \vdash a)$ is antitone.*
2. *P is antitone.*
3. *R is isotone.*

Proof.

1. Since $u \leq t$ and $u \sqcap a \leq a \leq b$ by Lemma 3.2 we have $t \vdash a \leq u \vdash b$ by Corollary 8. Since H is isotone we conclude $F(t \vdash a) \vdash G(t \vdash a) \leq F(u \vdash b) \vdash G(u \vdash b)$ which, again by Corollary 8, is equivalent to the claim.

2. Assume $a \leq b$. By part 1 we have $\lambda t.F(t \vdash a) \geq \lambda t.F(t \vdash b)$, from which we obtain $\mu t.F(t \vdash a) \geq \mu t.F(t \vdash b)$ by isotony of μ.

3. Let $a, b \in S$ such that $a \leq b$. By part 2 we have $P(b) \leq P(a)$. Now part 1 shows $F(P(b) \vdash b) \sqcap G(P(a) \vdash a) \leq G(P(b) \vdash b)$. By Definition 23.1, $F(P(b) \vdash b) = P(b)$ and shunting shows $G(P(a) \vdash a) \leq \overline{P(b)} + G(P(b) \vdash b) = R(b)$. Since $\overline{P(a)} \leq \overline{P(b)} \leq R(b)$, we have $R(a) \leq R(b)$ by the join property. □

Theorem 25. $\nu H = P(Q) \vdash Q$.

Proof. First, we prove that $P(Q) \vdash Q$ is a fixpoint of H. By Definition 23.1 we have $P(Q) = F(P(Q) \vdash Q)$. Hence, by Corollary 8,

$$\begin{aligned} H(P(Q) \vdash Q) &= F(P(Q) \vdash Q) \vdash G(P(Q) \vdash Q) = P(Q) \vdash G(P(Q) \vdash Q) \\ &= P(Q) \vdash \overline{P(Q)} + G(P(Q) \vdash Q) = P(Q) \vdash R(Q) = P(Q) \vdash Q. \end{aligned}$$

Second, we prove that $P(Q) \vdash Q$ is the greatest postfixpoint of H. Assume $t \vdash a \leq H(t \vdash a)$ which by Corollary 8 is equivalent to

$$F(t \vdash a) \leq t \text{ and } F(t \vdash a) \sqcap a \leq G(t \vdash a).$$

Hence $P(a) \leq t$ by Definition 23.1, and therefore also $P(a) = F(P(a) \vdash a) \leq F(t \vdash a)$ by Lemma 24.1. As a consequence $P(a) \sqcap a \leq G(t \vdash a)$, and therefore

$$P(a) \sqcap a \leq F(P(a) \vdash a) \sqcap G(t \vdash a) \leq G(P(a) \vdash a)$$

by Lemma 24.1. By shunting we obtain $a \leq \overline{P(a)} + G(P(a) \vdash a) = R(a)$, hence $a \leq Q$ by Definition 23.3. Therefore $P(Q) \sqcap a \leq Q$, and $P(Q) \leq P(a) \leq t$ by Lemma 24.2. Altogether $t \vdash a \leq P(Q) \vdash Q$ by Corollary 8. □

In Section 2.1.4 we have derived the omega operation on normal designs. As an example of using Theorem 25, let us now characterise the omega operation on designs over an ideal semiring (S, T) such that S is a weak omega algebra. Recall that a^ω is the greatest fixpoint of the function $\lambda x.ax$; this motivates the definition of H in the following result.

Corollary 26. *Let $t \vdash a$ be a design and define H by setting $H(u \vdash b) =_{\text{def}} (t \vdash a)(u \vdash b)$. Then $\nu H = \overline{a^\omega + a^* \bar{t} \top} \vdash 0$.*

Proof. Observe that H is isotone and, by Corollary 8,

$$H(u \vdash b) = (t \vdash a)(u \vdash b) = \overline{\bar{t}\top + a\overline{u}} \vdash ab = F(u \vdash b) \vdash G(u \vdash b),$$

where $F(u \vdash b) =_{\text{def}} \overline{\bar{t}\top + a\overline{u}}$ and $G(u \vdash b) =_{\text{def}} \bar{t}\top + a\overline{u} + ab$. By Definition 23.1,

$$P(b) = \mu u.F(u \vdash b) = \mu u.\overline{\bar{t}\top + a\overline{u}} = \overline{\nu u.\bar{t}\top + au} = \overline{a^\omega + a^*\bar{t}\top}.$$

Since $P(b)$ is constant, let $P = P(b)$. By Definitions 23.2 and 23.3, as well as omega and star properties,

$$\begin{aligned} Q &= \nu b.\overline{P(b)} + G(P(b) \vdash b) = \nu b.\overline{P} + \bar{t}\top + a\overline{P} + ab = a^\omega + a^*(\overline{P} + \bar{t}\top + a\overline{P}) \\ &= a^\omega + a^*\overline{P} + a^*\bar{t}\top = a^\omega + a^*(a^\omega + a^*\bar{t}\top) + a^*\bar{t}\top = a^\omega + a^*\bar{t}\top = \overline{P}. \end{aligned}$$

By Theorem 25 and Corollary 8, $\nu H = P(Q) \vdash Q = P \vdash \overline{P} = P \vdash 0$. □

2.1.6.2 The least fixpoint

The least fixpoint of a function on designs can be calculated in a similar way. To this end, we swap μ and ν in the definitions of P and Q, and use $P(a) =_{\text{def}} \nu t.F(t \vdash a)$ and $Q =_{\text{def}} \mu R$ in the following. Lemma 24 and its proof remain unchanged except for swapping μ and ν. We only need to restate Theorem 25. Note that we cannot use duality as an argument since the underlying structure is an ideal semiring (S, T) but not a lattice. Its dual need not be an ideal semiring as witnessed, for example, by the restriction operation $\sqcap$ that is defined only on $T \times S$.

Theorem 27. $\mu H = P(Q) \vdash Q$.

Proof. The proof that $P(Q) \vdash Q$ is a fixpoint of H proceeds exactly as for Theorem 25. We now prove that $P(Q) \vdash Q$ is the least prefixpoint of H. To this end, assume $H(t \vdash a) = F(t \vdash a) \vdash G(t \vdash a) \leq t \vdash a$, which by Corollary 8 is equivalent to

$$t \leq F(t \vdash a) \text{ and } t \sqcap G(t \vdash a) \leq a.$$

Since $t \leq F(t \vdash a) = F(t \vdash \overline{t} + a)$, we have $t \leq P(\overline{t} + a)$ by definition of P as greatest fixpoint. Moreover,

$$\begin{aligned} t \sqcap G(P(\overline{t} + a) \vdash \overline{t} + a) &= t \sqcap F(t \vdash \overline{t} + a) \sqcap G(P(\overline{t} + a) \vdash \overline{t} + a) \leq t \sqcap G(t \vdash \overline{t} + a) \\ &= t \sqcap G(t \vdash a) \leq a \end{aligned}$$

by Lemma 24.1. By shunting, $R(\overline{t} + a) = \overline{P(\overline{t} + a)} + G(P(\overline{t} + a) \vdash \overline{t} + a) \leq \overline{t} + a$, hence $Q = \mu R \leq \overline{t} + a$. This implies $t \sqcap Q \leq a$, and $t \leq P(\overline{t} + a) \leq P(Q)$ by antitony of P shown in Lemma 24.2. Therefore $P(Q) \vdash Q \leq t \vdash a$ by Corollary 8. □

In Section 2.1.3 we have derived the star operation on normal designs. As an example of using Theorem 27, let us now characterise the Kleene star on designs over an ideal semiring (S, T) such that S is a weak Kleene algebra. Recall that a^* is the least fixpoint of the function $\lambda x.ax + 1$; this motivates the definition of H in the following result.

Corollary 28. *Let $t \vdash a$ be a design and define $H(u \vdash b) =_{\text{def}} (t \vdash a)(u \vdash b) + (\top \vdash 1)$. Then $\mu H = \overline{a^* \overline{t} \top} \vdash a^*$.*

Proof. Observe that H is isotone and, by Corollary 8,

$$H(u \vdash b) \;=\; (t \vdash a)(u \vdash b) + (\top \vdash 1) \;=\; \overline{\overline{t}\top + a\overline{u}} \vdash (ab + 1) \;=\; F(u \vdash b) \vdash G(u \vdash b),$$

where $F(u \vdash b) =_{\text{def}} \overline{\overline{t}\top + a\overline{u}}$ and $G(u \vdash b) =_{\text{def}} \overline{t}\top + a\overline{u} + ab + 1$. By the definition of P,

$$P(b) = \nu u.F(u \vdash b) = \nu u.\overline{\overline{t}\top + a\overline{u}} = \overline{\mu u.\overline{t}\top + au} = \overline{a^*\overline{t}\top}.$$

Since $P(b)$ is constant, let $P = P(b)$. By the definitions of R and Q, as well as star properties,

$$\begin{aligned} Q &= \mu b.\overline{P(b)} + G(P(b) \vdash b) = \mu b.\overline{P} + \overline{t}\top + a\overline{P} + ab + 1 = a^*(\overline{P} + \overline{t}\top + a\overline{P} + 1) \\ &= a^*\overline{P} + a^*\overline{t}\top + a^* = a^*a^*\overline{t}\top + a^*\overline{t}\top + a^* = a^*\overline{t}\top + a^* = \overline{P} + a^*. \end{aligned}$$

By Theorem 27 and Corollary 8, $\mu H \;=\; P(Q) \vdash Q \;=\; P \vdash \overline{P} + a^* \;=\; P \vdash a^*$. □

2.2 Relating recursive definitions

In this section we investigate the relation between different kinds of linear recursions. We first show how concrete recursions can be modelled in our general framework developed in Section 2.1. By instantiation we are then able to apply the results obtained there to derive

properties of the studied recursions. We also show how to replace the conditions of UTP by tests [Koz97] which enable a more convenient notation.

Besides the concrete investigation, a major goal is to establish the applicability of the general results of the previous section to UTP. Once this procedure is clear we can conduct further development at the abstract level without referring to concrete programs, and this is done in Section 2.3. In this sense, the present section may be seen as a preparation for the next, which also features extensive use of tests.

2.2.1 Tail-recursion and tests

As a motivating example we derive two variants of the computation of the factorial. Only one of the implementations is tail-recursive, which leads to considerable difficulties when trying to show their equivalence. Let us begin with the tail-recursive variant. We assume that the variables x and y have natural numbers as their values.

Example. We start with the specification $P_1 =_{\text{def}} x, y := 0, yx!$ and derive, using the notations of [HH98], namely $\_ \lhd \_ \rhd \_$ for the conditional and ; for sequential composition and $\mathbb{I}$ for skip,

$$\begin{array}{rcl} P_1 & = & x, y := 0, yx! \\ & = & x, y := 0, yx! \lhd x = 0 \rhd x, y := 0, yx! \\ & = & y := y \cdot 1 \lhd x = 0 \rhd x, y := 0, yx(x-1)! \\ & = & y := y \lhd x = 0 \rhd y := yx \,;\, x, y := 0, y(x-1)! \\ & = & \mathbb{I} \lhd x = 0 \rhd y := yx \,;\, x := x - 1 \,;\, x, y := 0, yx! \\ & = & \mathbb{I} \lhd x = 0 \rhd y := yx \,;\, x := x - 1 \,;\, P_1. \end{array}$$

The calculation of the factorial is thus realised by successively multiplying the numbers $x, x-1, \ldots, 1$ in decreasing order. In each recursive step, the variable x is decremented after the multiplication but before the recursive call. The recursion terminates when $x = 0$; the starting value of x is lost after this procedure.

In UTP, the solution to such a recursive equation is defined as a least fixpoint, so we obtain

$$\begin{array}{rcl} P_1 & = & \mu X \bullet \mathbb{I} \lhd x = 0 \rhd y := yx \,;\, x := x - 1 \,;\, X \\ & = & (x \neq 0) * (y := yx \,;\, x := x - 1). \end{array}$$

using the notation of [HH98]. However, in UTP the least fixpoint is taken with respect to the refinement order, which is the reverse of the implication order we are using in our model of UTP designs. So we shall have to investigate the greatest fixpoint with respect to the natural order.

The first goal is to describe this recursion in our framework of ideal semirings. It is clear that sequential composition is modelled by $\cdot$ and skip by 1. To represent the conditional algebraically, we use special semiring elements called *tests* [Koz97]. They are similar to conditions, but work symmetrically and hence can express pre- and post-restrictions in a uniform way. Rather than Kleene algebras with tests, we use the more liberal test semirings [Möl04]. In the case of relations, tests are subsets of the identity relation, occasionally called co-reflexives or partial identities.

Definition 29. A *test semiring* is an idempotent weak semiring $(S, +, 0, \cdot, 1)$ with a distinguished set of elements $\text{test}(S) \subseteq S$ called *tests* and a *negation* operation $\neg$ such that $(\text{test}(S), +, 0, \cdot, 1, \neg)$ is a Boolean algebra.

In particular, $p \leq 1$ for each test $p \in \text{test}(S)$. The following lemma shows how to turn an ideal condition semiring into a test semiring. Actually, conditions are isomorphic to tests: An isomorphism is given by $\lambda t.t \sqcap 1$ mapping conditions to tests and $\lambda p.p\top$ mapping tests to conditions.

Lemma 30. *In an ideal condition semiring* (S, T) *define* $\text{test}(S, T) =_{\text{def}} \{t \sqcap 1 \mid t \in T\} \subseteq S$ *and* $\neg p =_{\text{def}} \overline{p\top} \sqcap 1$. *Then the structures* $(T, +, 0, \sqcap, \top, \bar{})$ *and* $(\text{test}(S, T), +, 0, \cdot, 1, \neg)$ *are isomorphic Boolean algebras.*

Proof. Let $f : T \to \mathrm{test}(S,T)$ and $g : \mathrm{test}(S,T) \to T$ be given by $f(t) = t \sqcap 1$ and $g(p) = p\top$. We first show that f and g are inverse to each other, hence bijections. One direction is $g(f(t)) = (t \sqcap 1)\top = t \sqcap (1\top) = t \sqcap \top = t$ by Lemma 5.3. The other part is a consequence since, letting $p = t \sqcap 1$, we have $f(g(p)) = p\top \sqcap 1 = (t \sqcap 1)\top \sqcap 1 = t \sqcap 1 = p$.

We next show that f is a homomorphism. Meet, join, complement, bottom and top are preserved since

- $f(t \sqcap u) = (t \sqcap u) \sqcap 1 = t \sqcap (u \sqcap 1) = (t \sqcap 1)(u \sqcap 1) = f(t) \cdot f(u)$ by Lemmas 3.4 and 5.3,
- $f(t + u) = (t + u) \sqcap 1 = (t \sqcap 1) + (u \sqcap 1) = f(t) + f(u)$ by distributivity,
- $f(\bar{t}) = \bar{t} \sqcap 1 = \overline{(t \sqcap 1)\top} \sqcap 1 = \neg(t \sqcap 1) = \neg f(t)$ as above,
- $f(0) = 0 \sqcap 1 = 0$ and $f(\top) = \top \sqcap 1 = 1$.

The claim follows since the structure $(T, +, 0, \sqcap, \top, \bar{\ })$ is a Boolean algebra. □

We can apply this result to designs. Recall from Corollary 9 that the normal designs over (S,T) have $\{t \vdash 0 \mid t \in T\}$ as conditions. We therefore obtain as tests the normal designs

$$(t \vdash 0) \sqcap (\top \vdash 1) = \begin{pmatrix} \top & \top \\ \bar{t} & \bar{t} \end{pmatrix} \sqcap \begin{pmatrix} \top & \top \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \top & \top \\ 0 & \bar{t} \sqcap 1 \end{pmatrix} = \top \vdash (\bar{t} \sqcap 1).$$

Let t be a condition and $p = t \sqcap 1$ the corresponding test. We can then represent UTP's conditional as

$$(a \lhd t \rhd b) = t \sqcap a + \bar{t} \sqcap b = (t \sqcap 1)a + (\bar{t} \sqcap 1)b = pa + \neg(t \sqcap 1)b = pa + \neg pb$$

using Lemma 5.3. For a test p and an element a the term pa is called the *input restriction* of a by p. The *output restriction* is symmetrically obtained as ap. Thanks to Lemma 30 we can entirely replace the conditions of a semiring by tests in our general framework of Section 2.1, and we will do so in the remaining part of this chapter.

Using the semiring elements $m =_{\mathrm{def}} (y := yx)$ and $d =_{\mathrm{def}} (x := x-1)$ and $p =_{\mathrm{def}} (x \neq 0)$ to abstract from the concrete assignments and the condition, we thus obtain for our first example the fixpoint

$$P_1^\nu =_{\mathrm{def}} \nu x.pmdx + \neg p.$$

We furthermore investigate the least fixpoint $P_1^\mu =_{\mathrm{def}} \mu x.pmdx + \neg p$ that may be of interest in other theories. If we assume that the underlying semiring is a Kleene algebra or even an omega algebra, as are UTP designs, the fixpoints can be represented as $P_1^\mu = (pmd)^*\neg p$ and $P_1^\nu = (pmd)^\omega + (pmd)^*\neg p$.

2.2.2 Linear recursion

Let us introduce a second computation of the factorial. It is no longer tail-recursive but the recursion is still linear. We then relate it to the first example.

Example. We now start with the specification $P_2 =_{\mathrm{def}} y := yx!$ and derive

$$\begin{array}{lcl} P_2 & = & y := yx! \\ & = & y := yx! \lhd x = 0 \rhd y := yx! \\ & = & y := y \cdot 1 \lhd x = 0 \rhd y := yx(x-1)! \\ & = & y := y \lhd x = 0 \rhd y := yx \,;\, y := y(x-1)! \\ & = & \mathbb{I} \lhd x = 0 \rhd y := yx \,;\, x := x-1 \,;\, y := yx! \,;\, x := x+1 \\ & = & \mathbb{I} \lhd x = 0 \rhd y := yx \,;\, x := x-1 \,;\, P_2 \,;\, x := x+1. \end{array}$$

The calculation proceeds as in the first example, but the variable x is incremented after returning from the recursive call. Therefore the value of x after this procedure is the same as its starting value.

We reason similarly to the first example, and using $i =_{\text{def}} (x := x + 1)$ we thus obtain the fixpoints

$$P_2^\nu =_{\text{def}} \nu x.pmdxi + \neg p$$

and $P_2^\mu =_{\text{def}} \mu x.pmdxi + \neg p$. Since this implementation is not tail-recursive, there is no obvious representation of the fixpoints using the Kleene star or the omega operation.

Both specifications can be related as follows:

$$(P_2 \,;\, x := 0) \;=\; (y := yx! \,;\, x := 0) \;=\; (x, y := 0, yx!) \;=\; P_1.$$

To extend this relation to the implementations, both derivations above have to be performed independently. We could reduce the amount of work if we were able to transform one implementation into the other. Our objective in the following is, therefore, to algebraically relate both implementations. Using $z =_{\text{def}} (x := 0)$ we would like to obtain

$$P_2^\nu z \;=\; (\nu x.pmdxi + \neg p)z \;=\; (\nu x.pmdx + \neg p) \;=\; P_1^\nu$$

and similarly $P_2^\mu z = P_1^\mu$.

2.2.3 Relating tail-recursion and linear recursion

We have considered two versions of the computation of the factorial and expressed their connection using the semiring formalism. This relation can now be proved in a completely algebraic way. We start with the least fixpoints P_1^μ and P_2^μ.

Theorem 31. *Let a, b, c, d be elements of a weak Kleene algebra such that $bd = d$ and $cd = c$. Then $(\mu x.axb + c)d = a^*c$.*

Proof. Let $g(x) =_{\text{def}} axb + c$. If the Kleene algebra is complete and composition distributes over arbitrary sums, we can apply μ-fusion [ABB+95]. Let $f(x) =_{\text{def}} xd$ and $h(x) =_{\text{def}} ax+c$, then

$$f(g(x)) = (axb + c)d = axbd + cd = axd + c = h(f(x)),$$

from which the claim $f(\mu g) = \mu h$ follows.

Without these additional assumptions, we prove the claim as follows. For the part $(\geq)$ note first that $c = cd = (\mu x.c)d \leq (\mu x.axb + c)d = (\mu g)d$. Second, using the fixpoint law in the last step,

$$a(\mu g)d = a(\mu g)bd \leq a(\mu g)bd + cd = (a(\mu g)b + c)d = (\mu g)d.$$

Therefore $c + a(\mu g)d \leq (\mu g)d$, which implies $a^*c \leq (\mu g)d$ by star induction. To show the converse inequality, from $a(a^*cb^*)b+c \leq a^*cb^*$ we infer $\mu g \leq a^*cb^*$ by the fixpoint induction law. Therefore $(\mu g)d \leq a^*cb^*d = a^*cd = a^*c$, since $bd = d \Rightarrow b^*d = d$ by star axioms. □

Instantiating $d = 1$ in Theorem 31, and therefore also $b = 1$, we obtain the special case $(\mu x.ax + c) = a^*c$, the least fixpoint representation of a^*c. Another consequence is the equality of the two considered implementations of the factorial.

Corollary 32. $P_2^\mu z = P_1^\mu$.

Proof. Observe that $iz = z$ since incrementing x before setting it to 0 is superfluous. Furthermore $\neg pz = \neg p$ since setting x to 0 can be omitted if it already is 0. Therefore the assumptions of Theorem 31 are satisfied and we conclude

$$P_2^\mu z = (\mu x.pmdxi + \neg p)z = (pmd)^*\neg p = (\mu x.pmdx + \neg p) = P_1^\mu.$$ □

To relate the implementations P_1^ν and P_2^ν, we have to restrict ourselves to UTP algebras. Note that ν-fusion cannot be directly applied since composition does not distribute over meets.

Theorem 33. *Let a, b, c, d be elements of a UTP omega algebra.*

1. $\nu x.axb = a^\omega$.

2. *If $bd = d$ and $cd = c$, then $(\nu x.axb + c)d = a^\omega + a^*c$.*

Proof.

1. Let $e(x) =_{\text{def}} axb$. By Lemma 21.1 and omega unfold, $aa^\omega b = aa^\omega = a^\omega$, which shows that a^ω is a fixpoint of e and hence $a^\omega \leq \nu e$. For the converse inequality observe that for an arbitrary fixpoint e° of e we have $e^\circ\top = ae^\circ b\top \leq ae^\circ\top$, so that $e^\circ\top \leq a^\omega$ by omega co-induction. But $e^\circ \leq e^\circ\top$ and we are done.

2. Let $g(x) =_{\text{def}} axb + c$. Using the fixpoint law in the first step,

$$(\nu g)d = (a(\nu g)b + c)d = a(\nu g)bd + cd = a(\nu g)d + c.$$

   This implies the part $(\leq)$ by omega co-induction. By star induction, this also implies $a^*c \leq (\nu g)d$. Hence it remains to show $a^\omega \leq (\nu g)d$ for the part $(\geq)$. But this holds, since $a^\omega = a^\omega d = (\nu x.axb)d \leq (\nu g)d$ by Lemma 21.1, part 1 and isotony. □

Compared with the original aim to relate the two implementations of the factorial, the solution provided by Theorem 33.2 is considerably more general due to its algebraic nature. It abstracts from the concrete recursion to the recursion pattern $\nu x.axb+c$, from the concrete program statements to their properties $bd = d$ and $cd = c$, and from designs to any structure satisfying the axioms of a UTP omega algebra.

Corollary 34. *In a UTP algebra, $P_2^\nu z = P_1^\nu$.*

Proof. We proceed similarly to the proof of Corollary 32 and using Theorem 33.2 we obtain

$$P_2^\nu z = (\nu x.pmdxi + \neg p)z = (pmd)^\omega + (pmd)^*\neg p = (\nu x.pmdx + \neg p) = P_1^\nu. \qquad \square$$

This kind of reasoning is generalised in Section 2.3. Another version of the factorial computation is discussed in [GM06b]. It also has a linear recursion, but multiplies the numbers $1, 2, \ldots, x$ in increasing order. Therefore additional properties of the multiplication element m have to be introduced to deal with it.

2.3 Symmetric linear recursion

We further investigate fixpoints of the function $f(x) =_{\text{def}} axb + c$ using now modal Kleene algebras, that is, Kleene algebras with domain, diamond and box operators. The investigation proceeds by separately considering the finite and the infinite parts of the fixpoints in Sections 2.3.2 and 2.3.4, respectively. One goal is to implement the recursion described by f by two consecutive while-loops, as achieved in Section 2.3.5. However, the structures and techniques introduced in this section are also applicable in further contexts.

The elements a, b, c in the definition of the function f can be instantiated to normal designs due to the development of Section 2.1. The results in this section therefore also apply to UTP. This shows that one can reason about the programs of UTP and related theories in a purely algebraic manner.

Since certain results hold only if a is deterministic, we need to characterise determinacy algebraically. For this, we can again employ tests, together with the domain operation on which we can base the modal operators. We also use tests for the algebraic representation of (co)invariants that will simplify subsequent arguments. In Section 2.3.3 further properties are formulated algebraically, namely convergence and divergence.

2.3.1 Domain, modal operators, determinacy and invariants

The domain of a semiring element a characterises the starting states of a, that is, the states from which corresponding output states may be reached under a. We use the equational axiomatisation of [DMS06b].

Definition 35. Let S be a test semiring. The domain operation $\ulcorner : S \to \mathrm{test}(S)$ is characterised by the axioms

$$\begin{array}{rcll} a & \leq & \ulcorner a \cdot a & \text{(d1)} \\ \ulcorner(p \cdot a) & \leq & p & \text{(d2)} \\ \ulcorner(a \cdot \ulcorner b) & \leq & \ulcorner(a \cdot b) & \text{(d3)} \end{array}$$

for $a, b \in S$ and $p \in \mathrm{test}(S)$.

Observe that $\ulcorner aa \leq 1a = a$ by isotony. Therefore the axiom (d1) can be strengthened to $a = \ulcorner aa$ which states that restriction to the full domain has no effect. Axiom (d2) formalises that the domain of a restricted element indeed reflects the restriction by satisfying the restricting test. It can be shown that (d1) $\wedge$ (d2) is equivalent to the domain elimination law

$$\ulcorner a \leq p \quad \Leftrightarrow \quad a \leq p \cdot a \qquad \text{(dom)}$$

This implies that the domain operation is unique if it exists. Furthermore $\ulcorner$ is isotone, distributes over $+$, and satisfies $a \leq 0 \Leftrightarrow \ulcorner a \leq 0$ and $\ulcorner(pa) = p\ulcorner a$ and $\ulcorner p = p$ for each test p. Using (d1) and (d2), axiom (d3) can be strengthened to an equality. This intuitively means that in the interaction of a and b only their 'boundary' matters but not their inner structure, at least to obtain the domain of the composition. See [DMS06b] for further properties.

Remark. In a semiring having a greatest element $\top$ there is another equivalent characterisation $\ulcorner a \leq p \Leftrightarrow a \leq p\top$ in the form of a Galois connection [Aar92]. This certainly applies if we use the test semiring induced by Lemma 30 from an ideal condition semiring (S, T). But in the latter case, we can even do better and explicitly characterise the domain operation. Adapting a result of [GM06a] we can then show that $\ulcorner a = a\top \sqcap 1$ satisfies the axioms:

- (d1) follows since $\ulcorner aa = (a\top \sqcap 1)a = a\top \sqcap a \geq a$ by Lemmas 5.3 and 3.2.
- (d2) follows since $\ulcorner(pa) = pa\top \sqcap 1 \leq p\top \sqcap 1 = p$ by Lemmas 3.1 and 30.
- (d3) follows since $\ulcorner(a\ulcorner b) = a\ulcorner b\top \sqcap 1 = a(b\top \sqcap 1)\top \sqcap 1 \leq ab\top\top \sqcap 1 = ab\top \sqcap 1 = \ulcorner(ab)$ by Lemmas 3.1 and 3.2.

As a consequence, many operations we are about to introduce by definitions or axioms can be explicitly expressed in this case. Moreover we can calculate the domain of normal designs over strict ideal condition semirings due to Corollary 9:

$$\begin{aligned} \ulcorner\begin{pmatrix} \top & \top \\ t & a \end{pmatrix} &= \begin{pmatrix} \top & \top \\ t & a \end{pmatrix}\begin{pmatrix} \top & \top \\ \top & \top \end{pmatrix} \sqcap \begin{pmatrix} \top & \top \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \top & \top \\ (t+a)\top & (t+a)\top \end{pmatrix} \sqcap \begin{pmatrix} \top & \top \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \top & \top \\ 0 & a\top \sqcap 1 \end{pmatrix} = \begin{pmatrix} \top & \top \\ 0 & \ulcorner a \end{pmatrix}, \end{aligned}$$

since for designs we have $t \leq a$. The corresponding result using condition semirings appears in [Möl06]. (End of remark)

With the help of the domain operation we define the forward modal operators diamond and box (of semiring elements) as test transformers. In their terms we can also characterise determinacy.

Definition 36. Assume a test semiring S with domain. The operation *diamond of* a is given by $\langle a\rangle p =_{\mathrm{def}} \ulcorner(a \cdot p)$. Its dual operation *box of* a is $[a]p =_{\mathrm{def}} \neg\langle a\rangle\neg p$. For each $a \in S$ both operations $\langle a\rangle$ and $[a]$ map tests to tests. An element $a \in S$ is *modally deterministic* iff $\langle a\rangle \leq [a]$.

Thus $\langle a\rangle p$ characterises those states for which *some* a-successor state satisfies p, whereas $[a]p$ characterises those states for which *all* a-successor states satisfy p. The box operator is the abstract counterpart of the wlp-operator [Dij76], and diamond is an abstract pre-image operator. The operations $\langle \_ \rangle$, $\langle a\rangle$ and $[a]$ are isotone and enjoy many useful algebraic properties [DMS06b].

Intuitively, modal determinacy of a can be understood as follows: If, in a given state, there is a transition to some target state p, then all transitions lead to p; since this holds for every p, even the 'finest', there is at most one transition. This characterisation is well-known from modal logic [Pop94] and has been transferred to the semiring context in [DM01] that also investigates other notions of determinacy. In the remainder we omit the qualifier 'modally' and only speak of deterministic elements.

To reason about the interaction of an element and a test, we introduce the notion of a (co)invariant. To prepare it, we first note that the characterising property (dom) of domain entails the following characterisations of diamond and box as well as equivalent test propagation properties:

$$\langle a\rangle p \leq q \Leftrightarrow \lceil(ap) \leq q \Leftrightarrow ap \leq qap \Leftrightarrow ap \leq qa \qquad \text{(dia)}$$
$$p \leq [a]q \Leftrightarrow pa\neg q \leq 0 \Rightarrow pa \leq paq \Leftrightarrow pa \leq aq \qquad \text{(box)}$$

If 0 is a right annihilator, the implication in (box) becomes an equivalence. In the particular case where $q = p$, that test propagates backwards or forwards through a, respectively. Intuitively, $p \leq [a]p$ means that all a-transitions originating in states satisfying p lead to states that satisfy p. This motivates the following definition, calling p an invariant of a.

Definition 37. A test p is an *invariant* of a if $pa \leq ap$, and a *co-invariant* of a if $ap \leq pa$.

The following lemma shows a close relation between invariants and co-invariants. We say that two elements a and b *commute* iff $ab = ba$.

Lemma 38. *Consider a test semiring S and let $a, b \in S$ and $p \in \mathrm{test}(S)$.*

1. *If p is a co-invariant of a, then $\neg p$ is an invariant of a.*
2. *If 0 is a right annihilator, then p is a co-invariant of a iff $\neg p$ is an invariant of a.*
3. *If S has a domain operation and a and b commute, then $\neg\lceil b$ is an invariant of a.*

Proof.

1. Assuming $ap \leq pa$, we have $\neg pa = \neg pap + \neg pa\neg p \leq \neg ppa + \neg pa\neg p = \neg pa\neg p \leq a\neg p$.
2. Assuming $pa \leq ap$, we have $a\neg p = pa\neg p + \neg pa\neg p \leq ap\neg p + \neg pa\neg p = \neg pa\neg p \leq \neg pa$. The claim follows together with part 1.
3. $\langle a\rangle\lceil b = \lceil(a\lceil b) = \lceil(ab) = \lceil(ba) = \lceil(b\lceil a) \leq \lceil b$, hence $\lceil b$ is a co-invariant of a by (dia). The claim follows by part 1. □

We freely use the equivalent characterisations (dia) of co-invariants. Several sufficient criteria for co-invariance under a deterministic a are given by the following lemma.

Lemma 39. *Let a be deterministic.*

1. *If p is contracted by $[a]$, that is, $[a]p \leq p$, then p is a co-invariant of a.*
2. *If p is expanded by $\langle a\rangle$, that is, $p \leq \langle a\rangle p$, then $\neg p$ is a co-invariant of a.*

Proof.

1. $\langle a\rangle p \leq [a]p \leq p$.
2. $p \leq \langle a\rangle p \Leftrightarrow \neg\langle a\rangle p \leq \neg p \Leftrightarrow [a]\neg p \leq \neg p$, and apply part 1. □

Let us explain the intuition underlying part 2: p being expanded by $\langle a \rangle$ means pointwise that every point in p has an a-successor in p. Dually, $\neg p$ being a co-invariant of a means that all a-predecessors of points in $\neg p$ are in $\neg p$ again. Now assume that a is deterministic and some point x in $\neg p$ has an a-predecessor y in p. Then y has an a-successor in p. But by determinacy of a the only a-successor of y is x, which is in $\neg p$; we thus obtain a contradiction. Part 1 can be interpreted in a similar manner.

2.3.2 The finite part

Let us return to the discussion of our function $f(x) = axb + c$. We say that *the choice in f is deterministic* iff $\ulcorner a \urcorner c = 0$. Our main goal in the following is to derive an implementation of f by two consecutive while-loops. To prove the correctness of the implementation, we look at the finite part and at the infinite part of the recursion, in turn. The separate correctness results are then combined in Section 2.3.5.

We use elements t from the underlying semiring to describe the left context in which the computation modelled by a terminates after at most (or exactly) n recursive steps. In the special case of t being a test, it describes the states from which such a termination occurs.

Definition 40. Let $n \in \mathbb{N}$ and t be a semiring element. Then *t terminates a after at most n steps* if

$$ta^n = ta^n \neg \ulcorner a$$

and *t terminates a after exactly n steps* if, additionally,

$$\forall k < n : ta^k = ta^k \ulcorner a.$$

Intuitively, this means that after starting with t and repeating a, if possible, n times we are out of the domain of a, hence a cannot be repeated any more. As a consequence we obtain $\forall k > n : ta^k = ta^n aa^{k-n-1} = ta^n \neg \ulcorner a \ulcorner aaa^{k-n-1} = ta^n 0$. Exact termination additionally requires that indeed n repetitions are possible by staying in the domain of a until the n-th iteration. The following lemma simplifies the investigated recursion in such terminating contexts.

Lemma 41. *Let f° be a fixpoint of f.*

1. *If t terminates a after at most n steps, then $tf^\circ = \sum_{k=0}^{n} ta^k cb^k$.*
2. *If the choice in f is deterministic and t' terminates a after exactly n steps, then $t'f^\circ = t'a^n cb^n$.*

Proof.

1. We first show by induction that $\forall n \in \mathbb{N} : f^\circ = a^n f^\circ b^n + \sum_{k=0}^{n-1} a^k cb^k$. For $n = 0$ this is clear, and for $n \geq 0$ we obtain

$$\begin{aligned} &a^{n+1} f^\circ b^{n+1} + \textstyle\sum_{k=0}^{n} a^k cb^k = a^n(af^\circ b)b^n + a^n cb^n + \sum_{k=0}^{n-1} a^k cb^k \\ &= a^n(af^\circ b + c)b^n + \textstyle\sum_{k=0}^{n-1} a^k cb^k = a^n f^\circ b^n + \sum_{k=0}^{n-1} a^k cb^k = f^\circ \end{aligned}$$

   using the fixpoint property and the induction hypothesis in the last two steps. As shown above, $ta^{n+1} = ta^n 0$, and therefore

$$\begin{aligned} tf^\circ &= t(a^{n+1} f^\circ b^{n+1} + \textstyle\sum_{k=0}^{n} a^k cb^k) = ta^n 0 + ta^n cb^n + t\sum_{k=0}^{n-1} a^k cb^k \\ &= \textstyle\sum_{k=0}^{n} ta^k cb^k. \end{aligned}$$

2. Since the choice in f is deterministic, $\forall k < n : t'a^k c = t'a^k \ulcorner a \urcorner cc = t'a^k 0$. Therefore, continuing the proof of part 1,

$$t'f^\circ = \textstyle\sum_{k=0}^{n} t'a^k cb^k = t'a^n cb^n + \sum_{k=0}^{n-1} t'a^k cb^k = t'a^n cb^n + \sum_{k=0}^{n-1} t'a^k 0 = t'a^n cb^n,$$

   since $t'a^k 0 \leq t'a^n cb^n$ for $k < n$. □

Under additional assumptions, we can represent the finite part of a fixpoint of f by two consecutive while-loops. This may be compared to the construct 'dojustasoften' of [Bau76].

Remark. The essential reason why such a transformation is possible is that the recursion structure can be linearised, for example, using the property that sequential composition $\cdot$ is associative with 1 as right identity [Hen77]. The procedure can also be seen as an abstract variant of eliminating linear recursive calls in applicative programs, see the transformation rule 'Recursion simplification II' of [Par90, page 299] that replaces a linear recursion by two tail-recursive functions. (End of remark)

The assumptions concern four special elements o, i, d, z of the semiring which together implement an abstract counter. Intuitively,

- o initialises a new counter to 0,
- i increments that counter,
- d decrements the counter if its value is greater than 0 and
- z tests whether the counter is 0 and removes it.

These elements count the number of iterations, provided they do not interfere with the loop constituents which is ensured by commutativity conditions.

The idea now is to represent the recursion given by f as two consecutive while-loops. The first one performs the a operations and increases the counter as many times as recursive calls occur. After that it performs the termination action c. The second loop performs the b operations and decreases the counter till it becomes zero again. As usual, the while-loops are represented by fixpoints of tail-recursive functions.

Definition 42. Let $f(x) = axb + c$. Let o, i, d, z be such that they commute with a, b, c, and

$$oz = id = 1 \ \wedge \ od = iz = 0.$$

In this case we refer to o, i, d, z as *counter elements* and say that the *counter assumptions* are satisfied. We then define $g(x) =_{\text{def}} aix + c$ and $h(x) =_{\text{def}} dbx + z$.

The assumptions formalise that a new counter has 0 as its value and cannot be decremented, and that after incrementing a counter it is not 0 and decrementing restores its value. Observe that o has z and i has d as a right-inverse element. The least fixpoints of g and h are the loops $(ai)^*c$ and $(db)^*z$, respectively, just as described above. However, we reason about arbitrary fixpoints of these functions in the following lemma.

Lemma 43. *Under the counter assumptions, let f°, g° and h° be fixpoints of f, g and h, respectively. Let t terminate a after at most n steps. Then $tf^\circ = tog^\circ h^\circ$.*

Proof. The proof proceeds in three steps.

- We show that the first loop g correctly realises the iterations of a while accumulating a counting right context, that is, that $tog^\circ = \sum_{k=0}^{n} ta^k coi^k$. Observe that to terminates ai after at most n steps, since

  $$to(ai)^n = ta^n oi^n = ta^n \neg\ulcorner aoi^n \leq ta^n oi^n \neg\ulcorner a = to(ai)^n \neg\ulcorner a \leq to(ai)^n \neg\ulcorner(ai)$$

  by Lemma 38.3. Hence $tog^\circ = \sum_{k=0}^{n} to(ai)^k c = \sum_{k=0}^{n} ta^k coi^k$ by Lemma 41.1.

- We show that in the accumulated left context oi^k the second loop h correctly realises the iterations of b, that is, that $oi^k h^\circ = b^k$. Observe that oi^k terminates db after at most k steps, because $o\ulcorner d = 0$ and

  $$oi^k(db)^k = oi^k d^k b^k = o\neg\ulcorner db^k \leq ob^k \neg\ulcorner d = oi^k d^k b^k \neg\ulcorner d \leq oi^k (db)^k \neg\ulcorner(db)$$

by Lemma 38.3. Hence, again by Lemma 41.1,

$$\begin{aligned} oi^k h^\circ &= \textstyle\sum_{l=0}^{k} oi^k (db)^l z = oi^k d^k b^k z + \sum_{l=0}^{k-1} oi^{k-l-1} izb^l \\ &= oi^k d^k b^k z + \textstyle\sum_{l=0}^{k-1} oi^{k-l-1} 0 = oi^k d^k b^k z = ozb^k = b^k. \end{aligned}$$

$*$ We combine both facts obtained above to conclude

$$tog^\circ h^\circ = \textstyle\sum_{k=0}^{n} ta^k coi^k h^\circ = \sum_{k=0}^{n} ta^k cb^k = tf^\circ$$

by Lemma 41.1. □

Remark. Not all commutativity conditions concerning the counter elements are necessary for Lemma 43. The proof above actually only uses that a, c commute with o, i and that b commutes with d, z. (End of remark)

We finally show how terminating (test) elements can be chosen. Intuitively, the test q_n describes that a can be iterated n times but not $n+1$ times. It only terminates a after *at most* n steps because a may be non-deterministic and some transition paths could be shorter. That cannot happen if a is deterministic and then q'_n describes that a can be iterated exactly n times along the single transition path.

Lemma 44. *Let $n \in \mathbb{N}$ and let $q_n =_{\text{def}} {\ulcorner}(a^n)\neg{\ulcorner}(a^{n+1})$ and $q'_n =_{\text{def}} {\ulcorner}(a^n \neg {\ulcorner} a)$. Note that $q'_n = \langle a^n\rangle \neg {\ulcorner} a = \neg[a^n]{\ulcorner} a$.*

1. *$q_n \leq q'_n$, and if a is deterministic, then $q_n = q'_n$.*
2. *The test q_n terminates a after at most n steps.*
3. *If a is deterministic, the test q'_n terminates a after exactly n steps.*

Proof. Let us first state an observation that is used in parts 1 and 3 of the proof: If a is deterministic, then a^j is deterministic for every $j \in \mathbb{N}$. This follows by induction since determinacy is preserved under composition and the element 1 is deterministic [DM01].

1. The part $(\leq)$ follows from ${\ulcorner}(a^n) = {\ulcorner}(a^n {\ulcorner} a + a^n \neg {\ulcorner} a) = {\ulcorner}(a^{n+1}) + {\ulcorner}(a^n \neg {\ulcorner} a)$ by shunting. For $(\geq)$ observe ${\ulcorner}(a^n \neg {\ulcorner} a) \leq {\ulcorner}(a^n)$ and ${\ulcorner}(a^n \neg {\ulcorner} a) = \langle a^n\rangle \neg {\ulcorner} a \leq [a^n] \neg {\ulcorner} a = \neg\langle a^n\rangle {\ulcorner} a = \neg {\ulcorner}(a^{n+1})$.
2. Since ${\ulcorner}(q_n a^n {\ulcorner} a) = q_n {\ulcorner}(a^n {\ulcorner} a) = {\ulcorner}(a^n)\neg{\ulcorner}(a^{n+1}){\ulcorner}(a^{n+1}) = 0$ we have $q_n a^n {\ulcorner} a = 0$, hence $q_n a^n \leq q_n a^n \neg {\ulcorner} a$.
3. By parts 1 and 2, the test $q'_n = q_n$ terminates a after at most n steps. It remains to show that termination does not occur before n steps. Let $k < n$, then a^k is deterministic, and we have

$$q'_n = \langle a^n\rangle \neg {\ulcorner} a = \langle a^k\rangle\langle a\rangle\langle a^{n-k-1}\rangle \neg {\ulcorner} a \leq \langle a^k\rangle\langle a\rangle 1 = \langle a^k\rangle {\ulcorner} a \leq [a^k] {\ulcorner} a.$$

Therefore $q'_n a^k \leq q'_n a^k {\ulcorner} a$ by (box). The converse inequality is trivial. □

2.3.3 Convergence and divergence

Before we can treat the infinite part of the investigated recursion, we have to develop algebraic means to describe convergence. That is, we need to characterise the starting states of an element a from which no infinite transition paths emerge. This set is represented as the test Δa [MS06], which is axiomatised as follows.

Definition 45. Let S be a weak Kleene algebra with tests. The *convergence operation* $\Delta : S \to \text{test}(S)$ satisfies the unfold and induction laws

$$[a](\Delta a) = \Delta a \qquad\qquad p \cdot [a]q \leq q \Rightarrow \Delta a \cdot [a^*]p \leq q$$

for $a \in S$ and $p, q \in \text{test}(S)$.

Thus $\Delta a \cdot [a^*]p$ is the least fixpoint of $\lambda q.p \cdot [a]q$ and Δa is the least fixpoint of $[a]$. Moreover $\neg\lceil a \leq \Delta a \leq \neg\lceil(a^\omega)$ and hence $\Delta a \cdot a^\omega = 0$ provided the omega operation exists. Finally, Δ is antitone and $[a^*](\Delta a) = [a \cdot a^*](\Delta a) = [a](\Delta a) = \Delta a$. Except for $\Delta a = \mu[a]$ the proofs of these properties can be translated from [GM06a] to our present setting of weak semirings. We prove the missing claim without assuming that 0 is a right annihilator in Lemma 46.4.

In addition to the convergence of an element a we consider its *divergence* $\nabla a =_{\text{def}} \neg\Delta a$. It abstracts the set of points from which infinite a-transition paths start. By standard fixpoint theory and the de Morgan duality between box and diamond we have $\nabla a = \nu\langle a\rangle$. The following lemma uses (co)invariants to show the interaction between an element and its convergence and divergence. It also relates divergence to finite deterministic iteration.

Lemma 46.

1. *Δa is an invariant of a.*
2. *∇a is a co-invariant of a.*
3. *If $ab \leq ba$, then $a^*b \leq ba^*$ and $ab^* \leq b^*a$. In particular, a (co)invariant of a is also one of a^*.*
4. *Δa is the least fixpoint of $[a]$.*
5. *If a is deterministic, then Δa and ∇a commute with a.*
6. *If a is deterministic and $p \leq \nabla a$, then $pa^*\neg\lceil a = 0$.*

Proof.

1. The claim follows immediately from the definition of Δa and (box).
2. The claim follows immediately from the definition of ∇a and (dia) by duality of box and diamond.
3. Let $ab \leq ba$. Then $b + aba^* \leq b + baa^* = b(1 + aa^*) \leq ba^*$ by the star unfold law, hence the star induction law implies $a^*b \leq ba^*$. Symmetrically one can show $ab^* \leq b^*a$.
4. The least fixpoint of $[a]$ is $\Delta a \cdot [a^*]1$, but in absence of the right annihilation law we cannot show $[a^*]1 = 1$. However, we show $\Delta a \leq [a^*]1$ which is sufficient. By parts 2 and 3, ∇a is a co-invariant of a^*, hence $\langle a^*\rangle(\nabla a) \leq \nabla a$ by (dia). Using duality and isotony of $[a^*]$ we obtain $\Delta a \leq [a^*](\Delta a) \leq [a^*]1$.
5. The commutativity of Δa with a is immediate from Lemma 39.1 and part 1. The commutativity of ∇a with a now follows by Lemma 38.1 and part 2.
6. Observe that $\neg\lceil a \leq \neg\lceil(a\nabla a) = \neg\langle a\rangle(\nabla a) = [a](\Delta a) = \Delta a$. Hence $pa^*\neg\lceil a \leq \nabla a a^* \Delta a = \nabla a \Delta a a^* = 0$ by parts 5 and 3. □

For example, the intuition underlying part 2, that states $a\nabla a \leq \nabla a a$, is that a transition that leads to a divergent path is the beginning of a divergent path itself. Part 6 also has a nicely intuitive, pointwise interpretation: By starting in a divergent state and following the unique transition path we can never reach a dead end of a.

Remark. The assumption $p \leq \nabla a$ of Lemma 46.6, also used in Lemma 48 below, restricts p to divergent states. It is, in particular, satisfied by $p = \lceil(a^\omega)$. To see this, let us restate [DMS06a, Lemma 7.6] for weak omega algebras:

1. $\lceil(a^\omega) \leq \nabla a$, which follows from $\nabla a = \nu\langle a\rangle$ since $\langle a\rangle\lceil(a^\omega) = \lceil(a\lceil(a^\omega)) = \lceil(aa^\omega) = \lceil(a^\omega)$.
2. $(\forall a : a\top = \lceil a\top) \Rightarrow (\forall a : \nabla a \leq \lceil(a^\omega))$, which follows using $\nabla a = \lceil(\nabla a\top) \leq \lceil(a^\omega)$ since $\nabla a\top = (\langle a\rangle\nabla a)\top = \lceil(a\nabla a)\top = a\nabla a\top$ implies $\nabla a\top \leq a^\omega$ by omega co-induction.

The premise of the second claim is certainly satisfied if we use the test semiring induced by Lemma 30 from an ideal condition semiring (S, T). This follows using the explicit representation of domain stated in Section 2.3.1 because $\lceil a\top = (a\top \sqcap 1)\top = a\top$. For normal designs we thus obtain divergence explicitly as $\nabla a = \lceil(a^\omega)$. We can conversely express omega by divergence as $a^\omega = a^\omega\top = (a^\omega\top \sqcap 1)\top = \lceil(a^\omega)\top = \nabla a\top$.

If a is deterministic, the second claim may be simplified to $\lceil a \leq a\top \Rightarrow \nabla a \leq \lceil(a^\omega)$. To see this, note that ∇a commutes with a by Lemma 46.5. Hence $\nabla a\top = \nabla a\lceil a\top \leq \nabla a a\top = a\nabla a\top$, from which the proof is completed as above. (End of remark)

2.3.4 The infinite part

Let us again return to the discussion of our function $f(x) = axb + c$, now treating its infinite part. We first show the following lemma concerning invertible elements of weak omega algebras. The intended application is using the right-inverse counter elements z and d of o and i, respectively.

Lemma 47. *Let x and y be commuting elements of a weak omega algebra.*

1. $yx^\omega \leq x^\omega$.
2. $y(xy)^\omega \leq (xy)^\omega \leq x^\omega$.
3. *If y has a right-inverse r that commutes with x, then $yx^\omega = x^\omega$ and $(xy)^\omega = x^\omega$.*

Proof. Let $xy = yx$.

1. By omega unfold, isotony and commutativity, $yx^\omega \leq yxx^\omega = xyx^\omega$. The claim follows by omega co-induction.
2. Since y and xy commute by $y(xy) = (yx)y = (xy)y$ we have $y(xy)^\omega \leq (xy)^\omega$ by part 1. Therefore $(xy)^\omega \leq xy(xy)^\omega \leq x(xy)^\omega$ by omega unfold and isotony. Now the second inequality follows by omega co-induction.
3. Let $yr = 1$ and $xr = rx$. Then $x^\omega = yrx^\omega \leq yx^\omega \leq x^\omega$ by part 1. For the second claim, observe that $rx^\omega \leq x^\omega \leq xx^\omega = xyrx^\omega$ by part 1 and omega unfold. Hence $rx^\omega \leq (xy)^\omega$ by omega co-induction, which entails $x^\omega = yrx^\omega \leq y(xy)^\omega \leq (xy)^\omega \leq x^\omega$ by isotony and part 2. □

We can apply the algebraic characterisation of divergence and the results of Section 2.3.3 to both representations of our non-tail-recursion f, the fixpoint and the while-loops. The result parallels Lemmas 41 and 43.

Lemma 48. *Let a and the choice in f be deterministic and assume $p \leq \nabla a$.*

1. $pa^*c = 0 = p(\mu f)$.
2. *Under the counter assumptions, $po(\mu g) = 0 = po(\mu g)(\mu h)$.*
3. *In a UTP omega algebra, $p(\nu f) = pa^\omega$.*
4. *In a UTP omega algebra, under the counter assumptions, $po(\nu g)(\nu h) = p(\nu f)$.*

Proof.

1. $pa^*c = pa^*\lceil cc \leq pa^*\neg\lceil ac = 0$ by determinacy of the choice and Lemma 46.6. Since $f(a^*cb^*) = aa^*cb^*b + c \leq a^*cb^*$ we have $\mu f \leq a^*cb^*$, hence $p(\mu f) \leq pa^*cb^* = 0$.
2. Observe that $1 + aia^*i^* \leq 1 + aa^*ii^* \leq a^*i^*$ by Lemma 46.3 using the counter assumptions and star unfold. By star induction we obtain $(ai)^* \leq a^*i^*$, and therefore $po(\mu g) = po(ai)^*c \leq poa^*i^*c = pa^*coi^* = 0$ repeatedly using the counter assumptions, Lemma 46.3 and part 1. Hence $po(\mu g)(\mu h) = 0(\mu h) = 0$.

3. Let $e(x) = axb$. We show $p(\nu f) = p(\nu e)$ which implies the claim by Theorem 33.1. The part ($\geq$) is obvious by isotony, and for ($\leq$) note that by Lemma 46.5,

$$\nabla a(\nu f) = \nabla a(a(\nu f)b + c) = \nabla aa(\nu f)b + \nabla ac = a\nabla a(\nu f)b$$

since $\nabla a \leq \lceil a \leq \neg\lceil c$. By the greatest fixpoint property, $\nabla a(\nu f) \leq \nu e$. Therefore also $p(\nu f) = p\nabla a(\nu f) \leq p(\nu e)$.

4. $po(\nu g)(\nu h) = po(\mu g + (ai)^\omega)(\nu h) = po(\mu g)(\nu h) + po(ai)^\omega(\nu h) = po(ai)^\omega$ by part 2 and Lemma 21.1. By the counter assumptions, o has right-inverse z commuting with a, and i has right-inverse d commuting with a. We thus obtain $po(ai)^\omega = poa^\omega = pa^\omega = p(\nu f)$ by Lemma 47.3 and part 3. □

Remark. The assumptions of Lemma 48 may be relaxed by noting that the essential $pa^*c \leq 0$ is equivalent to $p \leq \neg\langle a^*\rangle\lceil c = [a^*]\neg\lceil c$, which also suffices to prove part 3. Intuitively, this characterises the states from which no a-transition paths into the domain of c exist. These are just the states where proper termination does not occur. (End of remark)

2.3.5 Putting the finite and infinite parts together

We can finally combine the results for the finite and infinite parts of the investigated recursion obtained in Sections 2.3.2 and 2.3.4. The next lemma shows how to split up the starting states into these two parts. In the following we assume that certain countable sums of tests exist. This is the case, for example, when the Boolean test algebra is complete. Distribution of arbitrary elements over these sums is also assumed.

Lemma 49.

1. *Let $q_n = \lceil(a^n)\neg\lceil(a^{n+1})$ as defined in Lemma 44 and assume that $r =_{\text{def}} \sum_{n\in\mathbb{N}} q_n$ exists. Then $a^\infty =_{\text{def}} \neg\sum_{n\in\mathbb{N}} \neg\lceil(a^n)$ exists and $a^\infty + r = 1$.*
2. *Let $q'_n = \langle a^n\rangle\neg\lceil a$ as defined in Lemma 44 and assume that $r' =_{\text{def}} \sum_{n\in\mathbb{N}} q'_n$ exists and a distributes over this sum. Then $r' = \langle a^*\rangle\neg\lceil a$ and $\nabla a + r' = 1$.*
3. *$\nabla a \leq a^\infty$, and if a is deterministic, then $\nabla a = a^\infty$.*

Proof.

1. We show that $\neg a^\infty = \sum_{n\in\mathbb{N}} \neg\lceil(a^n) = r$. Let x be an upper bound of the tests in the sum, that is, $\forall n \in \mathbb{N} : \neg\lceil(a^n) \leq x$. Since $q_n \leq \neg\lceil(a^{n+1})$, we have $\forall n \in \mathbb{N} : q_n \leq x$. Therefore x is also an upper bound of r.

   It remains to show that r is an upper bound of the tests in the sum, or $\neg\lceil(a^n) \leq r$ for all $n \in \mathbb{N}$. This follows from $\sum_{i=0}^{n-1} q_i = \sum_{i=0}^{n} \neg\lceil(a^i)$ which we prove by induction. For the induction base $n = 0$ this is clear since $0 = \neg 1 = \neg\lceil 1 = \neg\lceil(a^0)$. For the induction step $n \geq 0$ we use the Boolean law $\neg p + pq = \neg p + q$ to calculate

$$\begin{aligned}\textstyle\sum_{i=0}^{n} q_i &= \textstyle\sum_{i=0}^{n-1} q_i + q_n = \sum_{i=0}^{n} \neg\lceil(a^i) + q_n\\ &= \textstyle\sum_{i=0}^{n-1} \neg\lceil(a^i) + \neg\lceil(a^n) + \lceil(a^n)\neg\lceil(a^{n+1})\\ &= \textstyle\sum_{i=0}^{n-1} \neg\lceil(a^i) + \neg\lceil(a^n) + \neg\lceil(a^{n+1}) = \sum_{i=0}^{n+1} \neg\lceil(a^i).\end{aligned}$$

2. Since $q'_n \leq \langle a^*\rangle\neg\lceil a$ for each $n \in \mathbb{N}$ by isotony of diamond, we have $r' \leq \langle a^*\rangle\neg\lceil a$. The reverse inequality reduces by diamond star induction [DMS06b] to $\neg\lceil a + \langle a\rangle r' \leq r'$, which is equivalent to $\neg\lceil a \leq r'$ and $\langle a\rangle r' \leq r'$. The first property holds by definition of r' since $\neg\lceil a = q'_0$. The second property is equivalent to r' being a co-invariant of a. To show it, observe that $\langle a\rangle q'_n = \langle a\rangle\langle a^n\rangle\neg\lceil a = \langle a^{n+1}\rangle\neg\lceil a = q'_{n+1}$, hence $aq'_n \leq q'_{n+1}a$ by (dia). Therefore, and since $q'_0 a = \neg\lceil a\lceil aa = 0$,

$$ar' = \textstyle\sum_{n\in\mathbb{N}} aq'_n \leq \sum_{n\in\mathbb{N}} q'_{n+1}a = \sum_{n\in\mathbb{N}} q'_n a = r'a.$$

The existence of the intermediate sums is guaranteed by the existence of r' and the distributivity of a.

For the second claim, observe that $1 = \lceil a + \neg\lceil a = \langle a\rangle 1 + \neg\lceil a$ holds, which entails $1 \leq \nabla a + \langle a^*\rangle\neg\lceil a = \nabla a + r'$ by divergence co-induction.

3. The part $(\leq)$ follows since

$$\nabla a \leq \langle a^n\rangle\nabla a \leq \lceil(a^n) \Rightarrow \neg\lceil(a^n) \leq \neg\nabla a \Rightarrow \textstyle\sum_{n\in\mathbb{N}} \neg\lceil(a^n) \leq \neg\nabla a \Rightarrow \nabla a \leq a^\infty.$$

For the part $(\geq)$ observe that determinacy of a implies $q_n = q'_n$ by Lemma 44.1, and therefore $r = r'$. In part 1 we have shown $r = \neg a^\infty$, hence $\nabla a + \neg a^\infty = 1$ by part 2, from which $\nabla a \geq a^\infty$ follows by shunting. □

Note that part 1 gives a statement about q_n, which has been used in Section 2.3.2 to state the results about the finite part for non-deterministic a and choice in f. However, Lemma 48 does not carry on this generality to the infinite part. Part 3 shows that both ways of Lemma 49 to partition coincide if a is deterministic.

Remark. This result has the following interpretation in terms of [SS89]: $\neg\nabla a = \Delta a$ describes the *progressively finite* states, that is, the states from which no infinite a-transition paths emerge, also known as the *initial part* of a. The test $\neg a^\infty$ describes the *progressively bounded* states, that is, those having an upper bound on the lengths of the emerging a-transition paths. Every progressively bounded state is progressively finite, that is, $\neg a^\infty \leq \neg\nabla a$. With deterministic a the progressively bounded and the progressively finite states are the same. The result can be seen as a special case of König's Infinity Lemma. (End of remark)

We partition the states according to Lemma 49.2 into the finite and infinite parts. The finite parts are described by restricting the input states to q'_n for each $n \in \mathbb{N}$ and the infinite part is described by restriction to ∇a. Applying the results of Sections 2.3.2 and 2.3.4 we thus obtain an implementation of the recursion specified by f using two while-loops.

Theorem 50. *Let a and the choice in f be deterministic. Under the counter assumptions, $\mu f = o(\mu g)(\mu h)$ and $\nu f = o(\nu g)(\nu h)$.*

Proof. By Lemma 49.2, distributivity, Lemmas 48, 44.3 and 43, again distributivity, and again Lemma 49.2,

$$\begin{aligned}\mu f &= (\nabla a + \textstyle\sum_{n\in\mathbb{N}} q'_n)(\mu f) = \nabla a(\mu f) + \sum_{n\in\mathbb{N}} q'_n(\mu f)\\ &= \nabla a o(\mu g)(\mu h) + \textstyle\sum_{n\in\mathbb{N}} q'_n o(\mu g)(\mu h) = (\nabla a + \sum_{n\in\mathbb{N}} q'_n) o(\mu g)(\mu h) = o(\mu g)(\mu h).\end{aligned}$$

The calculation for ν proceeds similarly. □

If the underlying partial order is complete, we can generalise the μ-part of Theorem 50 to the non-deterministic case using μ-fusion.

Theorem 51. *Let S be a weak Kleene algebra in which arbitrary sums of elements exist and composition distributes over such sums. Under the counter assumptions, $\mu f = o(\mu g)(\mu h)$.*

Proof. For reasons that will become clear later, we restrict the domain of g. Let $C \subseteq S$ contain the elements that commute with i, that is, $C = \{x \in S \mid xi = ix\}$. Note that C is complete by the assumptions. Let $g' : C \to S$ be the restriction of g to C. By the commutativity assumptions,

$$g'(x)i = (aix + c)i = aixi + ci = iaix + ic = i(aix + c) = ig'(x),$$

hence the type of g' is even $g' : C \to C$. Closing our introductory remark, observe that $\mu g' = \mu g \in C$, since

$$(\mu g)i = (ai)^* ci = (ia)^* ic = i(ai)^* c = i(\mu g).$$

Let $e : C \to S$ be given by $e(x) = ox(\mu h)$. If we can prove $e \circ g' = f \circ e$, the claim follows by μ-fusion. Observe that

$$\begin{aligned} f(e(x)) &= ae(x)b + c = aox(\mu h)b + c, \\ e(g'(x)) &= og'(x)(\mu h) = o(aix + c)(\mu h) = oaix(\mu h) + oc(\mu h). \end{aligned}$$

But the latter equals $aoxi(\mu h)+co(\mu h)$ by the commutativity assumptions and the restriction to C. It therefore suffices to show $i(\mu h) = (\mu h)b$ and $o(\mu h) = 1$, which hold by

$$\begin{aligned} i(\mu h) &= i(db)^*z = i(1 + d(bd)^*b)z = iz + id(bd)^*bz = 0 + (db)^*zb = (\mu h)b, \\ o(\mu h) &= o(db)^*z = o(1 + d(bd)^*b)z = oz + od(bd)^*bz = 1 + 0 = 1, \end{aligned}$$

using star properties and the counter assumptions. □

Besides the implementation of f by loops, it would be useful to have a direct closed form representation of its extremal fixpoints. This is not possible in plain Kleene algebra since the recursion pattern of f is described by the non-regular, context-free language $\{a^n c b^n \mid n \in \mathbb{N}\}$. A direct axiomatisation of the least fixpoint of f is investigated in [GM06b]. An axiomatic treatment of least fixpoints of general context-free recursions can be found in [ÉL05]. The description of context-free languages by an iterated substitution operation is studied, for example, in [Red65].

2.4 Discussion

This chapter shows that almost all of the standard theory of normal designs carries over to our more general algebraic setting. It should be noted that the operations of complement and meet are not required for all semiring elements but only on the conditions. Additionally, we have pointed out normal designs as instances of various algebraic structures and thereby established connections to existing theories, enabling the application of well-known mathematical results. For example, the combination of the approach using ideal semirings with the matrix calculus of [Möl06] leads to considerably simpler reasoning, since results about the star and omega iterations of matrices can be reused.

We have shown that normal designs can be equipped with box and diamond operators. While the box on the underlying semiring is the abstract counterpart of the wlp-operator, the one on designs corresponds to wp. Hence the general soundness and completeness proof for the associated Hoare logic, originally developed for a partial correctness framework, can directly be applied to normal designs [MS06].

Our algebraic treatment can be carried over to the prescriptions of [Dun01] that are similar to designs but reflect a general correctness view. For example, the proof of [Möl06] showing that the normal prescriptions form a weak semiring can be adapted to our setting. The main drawback of using prescriptions is that their natural order cannot be used as the refinement order since they model erratic non-determinism. Indeed, neither $\mu x.x$ nor $\nu x.x$ is a suitable model of the infinitely looping program since $\mu x.x$ is neutral with respect to choice and $\nu x.x$ is not a left annihilator of the composition of prescriptions. To solve this problem one has to take the Egli-Milner order, see [MS06].

Connections to related algebraic approaches have been mentioned throughout this chapter. Additional relations to theories of total and general correctness, such as [Dij76, BGW79, BZ86, Nel89, Hes92, DMT06], are detailed in [GM06a].

Further applications of our generalised results could be handling trace semantics and other semantical models, thus dealing with healthiness conditions such as R1–R3 of UTP in a purely algebraic fashion. The presented method could also serve as a model for the extension by parameters that describe further observations as proposed in [HH98].

Chapter 3

Representing non-strictness

Our goal is to develop a relational model of non-strict computations in an imperative, non-deterministic context. As a simple motivation of the issues we are about to address, consider the statement $P =_{\text{def}} x_1, x_2{:=}\lightning_0, 2$ that simultaneously assigns an undefined value to x_1 and 2 to x_2. In a conventional language its execution aborts, but we want undefined expressions to remain harmless if their value is not needed. This is standard in functional programming languages with lazy evaluation like Haskell [PJ03], Clean [PE93] and Miranda [Tur85]. Yet also in an imperative language it can be reasonable to require $P \,;\, x_1{:=}x_2 = x_1, x_2{:=}2, 2$ since the value of x_1 after the execution of P is never used. To see this, consider the following Haskell program that implements $P \,;\, x_1{:=}x_2$ in monadic style:

```
import Data.IORef;
main = do r <- newIORef (div 1 0 , 2)
          modifyIORef r (\(x1,x2) -> (x2,x2))
          x <- readIORef r
          print x
```

It prints `(2,2)` terminating successfully, but would abort if `(x2,x2)` was changed to `(x1,x1)`.

The starting point of our investigation is the imperative, non-deterministic core of UTP. Considering again the statement P introduced above, we have to address that UTP models undefinedness as non-termination [HH98, page 78]. In particular $P = \mathbb{T}_{\text{D}}$, where $\mathbb{T}_{\text{D}}$ models the never terminating program. In consequence there is no distinction between undefinedness of individual variables; actually $P = x_1, x_2{:=}2, \lightning_0$ holds. Moreover computations are strict in the sense that $P \,;\, x_1{:=}x_2 = P$ is again non-termination.

In some contexts such a uniform treatment of undefinedness and non-termination is not appropriate. UTP's point of view is that of the specifier who does not care whether a program loops indefinitely or aborts due to an error, since in both cases it does not fulfil its objective. We can, however, argue for a differentiation between finite and infinite failure. From the users' point of view, errors can actually be observed about executions of programs whereas non-termination cannot. From the programmers' and language designers' point of view, errors might be recovered from, for example, by exception handling. From the theorists' point of view, some errors could be detectable in contrast to non-termination which is an undecidable property. We therefore strive for a theory that separates undefinedness and non-termination. It is then manifest to regard variables individually to obtain an even finer distinction.

This chapter presents a framework with these qualities. We first argue why UTP designs are not adequate for this purpose. After properly setting up the relational stage, we describe our new approach which is based on the following ideas. As usual we represent undefinedness of individual variables by adding a special value $\bot$ to their ranges. We add another special element ∞ to distinguish non-termination from undefinedness. The difficulty is to choose the relations and operations (that model computations) such that, on the one hand, they correctly handle these special values and, on the other hand, they are continuous. The latter

is required to iteratively approximate the solutions to recursive equations, which corresponds to the evaluation of recursion in practice. Furthermore, key constructs such as composition and choice should retain their familiar relational meaning to obtain nice algebraic properties. We solve this problem by introducing a partial order on the ranges of variables and states, and forming the closure of relations with respect to this order.

Section 3.2 presents a compendium of relations modelling the programming constructs known from UTP. The goal then is to identify several healthiness conditions they satisfy, starting with isotony and the left and right unit laws. In Section 3.3 we derive further properties, namely finite branching, continuity and totality. We thus obtain a theory similar to that of designs, but describing non-strict computations, able to yield defined results in spite of undefined inputs. Moreover it is sufficient to execute only those parts of a program necessary to calculate the final results, which can improve efficiency.

With lazy execution comes the need to consider dependences between individual computations. Such dependences also play a role in optimising program transformations like those performed in compilers. Their structure is investigated in Section 3.4 that gives a relational model. Starting from the observation that non-strict computations with defined results cannot depend on undefined inputs, we derive two additional healthiness conditions. Using another partial order we develop an equivalent, algebraically elegant form of these conditions. All our programming constructs satisfy them, but they can also be applied to relations modelling new constructs. As an example, we introduce the relational parallel composition to UTP to algebraically deal with local variables.

3.1 Prolegomena

We have seen the need to separate undefinedness from non-termination. Already modelling non-termination, UTP designs are obvious candidates for a modified treatment of undefinedness. We first show that although such an extension is possible, it leads to a fundamental problem. The conclusion is that UTP designs cannot adequately model non-strict computations. In Section 3.1.2 we introduce the relational foundations of an alternative model. It will be fully developed in the remainder of this chapter.

3.1.1 Designs

Recall that UTP designs are composed of two relations, modelling specifications in an assumption/commitment style. The precondition P of the design $P \vdash Q$ represents the terminating states, while Q represents the possible transitions starting in those states. Let us focus on the type of the relation Q between program states. The state of an imperative program is given by the values of its variables. Assume for the sake of exposition that the program has two variables x_1 and x_2 ranging over the natural numbers. A state then is an element of $\mathbb{N}^2 =_{\text{def}} \mathbb{N} \times \mathbb{N}$, and the transition relation Q is an element of $\mathbb{N}^2 \leftrightarrow \mathbb{N}^2$. No provisions are made to represent variables with undefined values. Indeed, there is no reason to, since undefinedness is modelled as non-termination in the component P of designs.

To separate undefinedness from non-termination we have to provide means to represent undefined values in the transition relation Q of designs. This is achieved by modifying the set of states in either of two ways. Both start by extending the range of each variable to $\mathbb{N} \cup \{\bot\}$, where the special element $\bot$ represents the undefined value.

- One can use the smash product $\mathbb{N}^2 \cup \{\bot\}$ as the set of states. A transition relation then is an element of $(\mathbb{N}^2 \cup \{\bot\}) \leftrightarrow (\mathbb{N}^2 \cup \{\bot\})$. For example,

$$\begin{array}{ll} & x_1{:=}1/0 \,;\, x_1, x_2{:=}2, 3 \\ = & \{(x, x') \mid x'{=}\bot\} \,;\, \{(x, x') \mid x'{=}(2, 3)\} \\ = & \{(x, x') \mid x'{=}(2, 3)\} \\ = & x_1, x_2{:=}2, 3. \end{array}$$

The special element $\bot$ represents the state after an assignment whose right hand side has an undefined value. Since that value is never used, the assignment has no effect, modelling the fact that evaluation of the right hand side is not needed. However, we still have

$$x_1{:=}\lightning_0 \;=\; \{(x, x') \mid x'{=}\bot\} \;=\; x_2{:=}\lightning_0.$$

This is because $\bot$ models undefinedness of the state as a whole but not of its constituents, the individual variables.

∗ One can use the Cartesian product $(\mathbb{N}\cup\{\bot\})^2$ as the set of states. A transition relation then is an element of $(\mathbb{N}\cup\{\bot\})^2 \leftrightarrow (\mathbb{N}\cup\{\bot\})^2$. Thus undefined and defined variables may coexist as exemplified by

$$\begin{array}{ll} & x_1, x_2{:=}\lightning_0, 2 \;;\; x_1{:=}x_2 \\ = & \{((x_1, x_2), (x_1', x_2')) \mid x_1'{=}\bot \wedge x_2'{=}2\} \;;\; \{((x_1, x_2), (x_1', x_2')) \mid x_1'{=}x_2 \wedge x_2'{=}x_2\} \\ = & \{((x_1, x_2), (x_1', x_2')) \mid x_1'{=}2 \wedge x_2'{=}2\} \\ = & x_1, x_2{:=}2, 2. \end{array}$$

This is because $\bot$ models undefinedness of an individual variable rather than the state as a whole.

We use the Cartesian product in the following, but the problem we are about to exhibit remains also with the smash product.

The transition relations, now elements of $(\mathbb{N}\cup\{\bot\})^2 \leftrightarrow (\mathbb{N}\cup\{\bot\})^2$, are built into designs to deal with non-termination. For the following argument, we redefine the assignment as the design

$$(x_1, x_2{:=}e_1, e_2) =_{\text{def}} (\top \vdash x_1'{=}e_1 \wedge x_2'{=}e_2),$$

reflecting the fact that an assignment always terminates as opposed to the original assignment of UTP, see Section 1.2. To complete the separation of undefinedness and non-termination, also conditional statements have to be redefined, since their conditions are expressions and can have undefined values, too. We omit the details, because they do not affect the following two facts. First,

$$\begin{array}{ll} & x_1{:=}\lightning_0 \;;\; x_1, x_2{:=}2, 3 \\ = & (\top \vdash x_1'{=}\bot \wedge x_2'{=}x_2) \;;\; (\top \vdash x_1'{=}2 \wedge x_2'{=}3) \\ = & (\top \vdash x_1'{=}2 \wedge x_2'{=}3) \\ = & x_1, x_2{:=}2, 3, \end{array}$$

using the composition formula of designs shown, for example, in Corollary 8. Second,

$$\top_{\mathrm{D}} \;;\; x_1, x_2{:=}2, 3 \;=\; (\bot\!\!\bot \vdash \bot\!\!\bot) \;;\; (\top \vdash x_1'{=}2 \wedge x_2'{=}3) \;=\; (\bot\!\!\bot \vdash \bot\!\!\bot) \;=\; \top_{\mathrm{D}},$$

recalling that the design $\top_{\mathrm{D}}$ represents non-termination. Consider the possible execution strategies for a program of the form $P \;;\; x_1, x_2{:=}2, 3$, assuming we do not know whether $P = x_1{:=}\lightning_0$ or $P = \top_{\mathrm{D}}$ since this is undecidable in general. Conventionally, one would first execute P and then $x_1, x_2{:=}2, 3$. This appropriately leads to non-termination if $P = \top_{\mathrm{D}}$, but undesirably aborts if $P = x_1{:=}\lightning_0$. To avoid this error, one could alternatively start with $x_1, x_2{:=}2, 3$, realising that the values of the variables prior to this assignment are not needed. The execution of P is thus omitted, which is not acceptable if $P = \top_{\mathrm{D}}$.

The conflict is summarised as follows: It is possible to recover from undefinedness, but it is not possible to recover from non-termination. To observe the latter, otherwise unnecessary calculations must be performed. They possibly abort due to undefined expressions, contradicting the former.

Since it is our goal to model non-strict computations, we are forced to give up an equation like $\top_{\mathrm{D}} \;;\; x_1, x_2{:=}2, 3 = \top_{\mathrm{D}}$. This is an instance of the healthiness condition $\top_{\mathrm{D}} \;;\; P = \top_{\mathrm{D}}$ that every design P satisfies, called H1b in Section 1.2. 'However, a lazy functional language does not satisfy this law.'[Hoa99a, page 24][Hoa99b, page 25] Although we are not specifically concerned with functional programming languages, we therefore cannot use UTP designs for our purpose.

3.1.2 Relations

In this section we set up the context of the investigation of non-strictness. We describe the relational model of imperative, non-deterministic programs in more detail. In doing so, we introduce terminology, notation and conventions used in this chapter.

3.1.2.1 Modelling computations

Characteristic features of imperative programming are variables, states and statements. We assume an infinite supply $x_1, x_2, \ldots$ of variables. Associated with each variable x_i is its type or range D_i, a set comprising all values the variable can take. Each D_i shall contain two special elements $\bot$ and ∞ with the following intuitive meaning: If the variable x_i has the value $\bot$ *and* this value is needed, the execution of the program aborts. If the variable x_i has the value ∞ *and* this value is needed, the execution of the program does not terminate.

A state is given by the values of a finite but unbounded number of variables $x_1, \ldots, x_m$. We abbreviate the sequence $(x_1, \ldots, x_m)$ as $\vec{x}$. Let $1..m$ denote the first m positive integers. The relative complement of a subset $I \subseteq 1..m$ is denoted by $\overline{I} =_{\text{def}} 1..m \setminus I$. We abbreviate $\overline{\{i\}}$ as $\overline{i}$. Let $D_I =_{\text{def}} \prod_{i \in I} D_i$ denote the Cartesian product of the ranges of the variables x_i with $i \in I$. A state is an element $\vec{x} \in D_{1..m}$.

An imperative program is composed of statements whose effect is to transform states into new states. We therefore distinguish the values of a variable x_i before and after the execution of a statement. The input value is denoted just as the variable by x_i and the output value is denoted by x_i'. In particular, both $x_i \in D_i$ and $x_i' \in D_i$. Statements may introduce new variables into the state and remove variables from the state. Using UTP terminology, the input alphabet is $\{x_1, \ldots, x_m\}$ and the output alphabet is $\{x_1', \ldots, x_n'\}$ with possibly different m and n. We abbreviate the sequence $(x_1', \ldots, x_n')$ as $\vec{x}'$.

A computation is modelled as a relation $R = R(\vec{x}, \vec{x}') \subseteq D_{1..m} \times D_{1..n}$. An element $(\vec{x}, \vec{x}') \in R$ intuitively means that the execution of R with input values $\vec{x}$ *may* yield the output values $\vec{x}'$. The image of a state $\vec{x}$ is given by $R(\vec{x}) =_{\text{def}} \{\vec{x}' \mid (\vec{x}, \vec{x}') \in R\}$. Non-determinism is modelled by having $|R(\vec{x})| > 1$. Compared to designs, the new models get by with just one relation instead of two, and this is compensated by the additional special elements $\bot$ and ∞.

3.1.2.2 Relation algebra

Another way to state the type of the relation is $R : D_{1..m} \leftrightarrow D_{1..n}$. The framework employed is that of heterogeneous relation algebra [SHW97]. It admits relations between different sets as required if $m \neq n$, in contrast to classical (homogeneous) relation algebra [Tar41, Mad96]. We omit any notational distinction of the types of relations and their operations and assume type-correctness in their use. Reasoning is algebraic whenever possible, but we revert to pointwise arguments if it is convenient.

We denote the zero, identity and universal relations by $\perp\!\!\!\perp$, $\mathbb{I}$ and $\top\!\!\!\top$, respectively. Lattice join, meet and order of relations are denoted by $\cup$, $\cap$ and $\subseteq$, respectively. The Boolean complement of R is $\overline{R}$, and the converse (transposition) of R is $R^\smile$. Relational (sequential) composition of P and Q is denoted by $P \,;\, Q$ and PQ. Converse has highest precedence, followed by sequential composition, followed by meet and join with lowest precedence.

A relation R is a vector iff $R\top\!\!\!\top = R$. The vectors form a Boolean subalgebra of the Boolean algebra of relations. Frequently used relational facts are

- the Dedekind law $PQ \cap R \subseteq (P \cap RQ^\smile)(P^\smile R \cap Q)$,
- the Schröder equivalences $PQ \subseteq R \Leftrightarrow P^\smile \overline{R} \subseteq \overline{Q} \Leftrightarrow \overline{R} Q^\smile \subseteq \overline{P}$,
- the Tarski rule $R \neq \perp\!\!\!\perp \Leftrightarrow \top\!\!\!\top R \top\!\!\!\top = \top\!\!\!\top$, and
- the vector property $(R\top\!\!\!\top \cap P)Q = R\top\!\!\!\top \cap PQ$, its converse $P(Q \cap \top\!\!\!\top R) = PQ \cap \top\!\!\!\top R$, and $P(R\top\!\!\!\top \cap Q) = (P \cap (R\top\!\!\!\top)^\smile)Q$.

Recall the functional properties of relations, given for example in [SS89]. A relation R is total iff $R\top = \top$ and univalent iff $R^\smile R \subseteq \mathbb{I}$. A relation is a mapping iff it is both total and univalent. A relation R is surjective iff $R^\smile$ is total and injective iff $R^\smile$ is univalent. Useful laws include

* $R(P \cap Q) = RP \cap RQ$ and $R\overline{P} = R\top \cap \overline{RP}$ if R is univalent, and
* $\overline{RP} = R\overline{P}$ and $PR \subseteq Q \Leftrightarrow P \subseteq QR^\smile$ if R is a mapping.

A relation R is reflexive iff $\mathbb{I} \subseteq R$, symmetric iff $R^\smile = R$ and transitive iff $RR \subseteq R$. A relation is an equivalence relation iff is reflexive, symmetric and transitive.

Relational constants representing computations may be specified by set comprehension as, for example, in $R = \{(\vec{x}, \vec{x}') \mid x'_1{=}x_2 \wedge x'_2{=}1\} = \{(\vec{x}, \vec{x}') \mid x'_1{=}x_2\} \cap \{(\vec{x}, \vec{x}') \mid x'_2{=}1\}$. We abbreviate such a comprehension by its constituent predicate, that is, $R = x'_1{=}x_2 \cap x'_2{=}1$. In doing so, we use the identifier x in a generic way, possibly decorated with an index, a prime or an arrow. It follows, for example, that $\vec{x}{=}\vec{c}$ is a vector for any constant $\vec{c}$.

To form heterogeneous relations and, more generally, to change their dimensions, we use the following operation that is related to the cylindrifications of [HMT71] and the projection of [Cod70]. Let I, J, K and L be index sets such that $I \cap K = \emptyset = J \cap L$. The dimensions of $R : D_{I\cup K} \leftrightarrow D_{J\cup L}$ are restricted by

$$(\exists \vec{x}_K, \vec{x}'_L : R) =_{\text{def}} \{(\vec{x}_I, \vec{x}'_J) \mid \exists \vec{x}_K, \vec{x}'_L : (\vec{x}_{I\cup K}, \vec{x}'_{J\cup L}) \in R\} : D_I \leftrightarrow D_J.$$

We abbreviate the case $L = \emptyset$ as $(\exists \vec{x}_K : R)$ and the case $K = \emptyset$ as $(\exists \vec{x}'_L : R)$. Observe that $(\exists \vec{x}_K : \mathbb{I})\ ;\ R = (\exists \vec{x}_K : R)$ and $R\ ;\ (\exists \vec{x}'_L : \mathbb{I}) = (\exists \vec{x}'_L : R)$ and $(\exists \vec{x}_K : \mathbb{I})\ ;\ R\ ;\ (\exists \vec{x}'_L : \mathbb{I}) = (\exists \vec{x}_K, \vec{x}'_L : R)$. Moreover $(\exists \vec{x}_K : \mathbb{I})$ is total and its converse $(\exists \vec{x}'_L : \mathbb{I})$ is a mapping. The dimension restriction $(\exists \vec{x}_K, \vec{x}'_L)$ is isotone.

Defined in terms of the projection, we furthermore use the following relational parallel composition operator, similar to that of [BK97]. Let again I, J, K and L be index sets such that $I \cap K = \emptyset = J \cap L$. The parallel composition of the relations $P : D_I \leftrightarrow D_J$ and $Q : D_K \leftrightarrow D_L$ is

$$P \,{}_I\|_K\, Q =_{\text{def}} (\exists \vec{x}'_K : \mathbb{I})\ ;\ P\ ;\ (\exists \vec{x}_L : \mathbb{I}) \cap (\exists \vec{x}'_I : \mathbb{I})\ ;\ Q\ ;\ (\exists \vec{x}_J : \mathbb{I}) : D_{I\cup K} \leftrightarrow D_{J\cup L}.$$

If the types of P and Q follow from the context or do not matter we omit the indices in $P \,{}_I\|_K\, Q$ and write $P \| Q$ instead. Observe that $\|$ is isotone, distributes over $\cup$ and has $\perp\!\!\!\perp$ as annihilator. Further discussion of parallel composition is provided in Section 3.6.3. It should be kept in mind that $\|$ models disjoint parallelism and not concurrency. The $\|$ operator has lower precedence than meet and join.

3.1.2.3 Further conventions

The function that maps x to $f(x)$ is denoted by $x \mapsto f(x)$, meaning the same as $\lambda x.f(x)$ does. The least fixpoint of a function f is denoted by μf, its greatest fixpoint by νf. The scope of $\exists$, $\forall$, $\bigcup$, $\bigcap$, $\mapsto$ and $\exists$ in a formula extends as far to the right as possible, that is, until the next unmatched closing bracket or the end of the formula.

Let $\vec{x}_I$ denote the subsequence of all x_i such that $i \in I$. Let $a{\in}\vec{x}_I$ denote $\exists i \in I : a{=}x_i$, and $a{\notin}\vec{x}_I$ its negation. Let $\vec{x}_I{=}a$ denote $\forall i \in I : x_i{=}a$, and $\vec{x}_I{\neq}a$ its negation. The set of subsets of a set S is $\mathscr{P}(S)$. We denote the non-negative integers or natural numbers by $\mathbb{N}$ and write $|S| \in \mathbb{N}$ to state that S is finite.

3.2 Compendium

We present a relational model of non-strict computations. Since we cannot use UTP designs, we have to reformulate the respective theory. In particular, we give new definitions for most programming constructs and identify several healthiness conditions they satisfy. The latter

starts with isotony and the unit laws in this section, followed by boundedness, continuity and totality in Section 3.3 and two dependence conditions in Section 3.4.

Basic statements comprise the assignment, skip, (un)declaration of variables and alphabet extension. Control flow is provided by the conditional, sequential and parallel composition. Relations may furthermore be composed by the non-deterministic choice. Its dual, conjunction, is technically useful for the treatment of recursion which is given by the greatest fixpoint. We moreover consider its dual, the least fixpoint. These constructs are introduced successively; a summary of their syntax and semantics will be given in Definition 60.

This selection of programming constructs subsumes the imperative, non-deterministic core of UTP [HH98, Chapters 2 and 3]. It has been chosen to validate the definitions we are about to introduce. We are not concerned with the computational power of our model, except for noting that under the moderate assumption that there are variables and expressions to do arithmetic, the programming constructs are Turing-complete. We do not present a special logic for our computations, but treat them as relations. Neither do we investigate implementation techniques, for example, closures and graph rewriting systems used in compilers and interpreters of functional programming languages [WM97b, PE93].

3.2.1 Relational operations

Recall that the range D_i of a variable contains the special elements $\bot$ and ∞ modelling undefinedness and non-termination, respectively. Let $\preccurlyeq : D_i \leftrightarrow D_i$ be the flat order on D_i with ∞ as its least element, that is, $x \preccurlyeq y \Leftrightarrow_{\text{def}} x{=}\infty \vee x{=}y$. It follows that $\preccurlyeq$ is a partial order and even a meet-semilattice. More generally, the infimum of a non-empty set S is given by

$$\inf S = \begin{cases} x, & \text{if } S = \{x\}, \text{ and} \\ \infty, & \text{if } |S| > 1. \end{cases}$$

Recall further that $D_I = \prod_{i \in I} D_i$. Let $\preccurlyeq : D_I \leftrightarrow D_I$ also denote the pointwise extension of that order, that is, $\vec{x}_I \preccurlyeq \vec{y}_I \Leftrightarrow_{\text{def}} \forall i \in I : x_i \preccurlyeq y_i$. It is again a partial order and its dual order is denoted by $\succcurlyeq =_{\text{def}} \preccurlyeq^{\smile}$. The meet operation is obtained by pointwise extension, too. As usual, a totally ordered subset of elements of a partially ordered set is called a *chain*. We exclusively work with finite I, indexing the variables of the current state. It is easily proved by induction on the size of the index set I, that $|C| \leq |I| + 1$ for any chain C in D_I ordered by $\preccurlyeq$. It follows that the corresponding strict order $\prec$ is regressively bounded and therefore also well-founded. These properties will be stated precisely and used in Section 3.3.1.

Most of the time we use the partial order $\preccurlyeq$ with the index set $I = 1..m$ of all variables, as in $\vec{x} \preccurlyeq \vec{x}'$. Indeed, we take this as the definition of the new relation modelling skip, denoted also by $\mathbb{1} =_{\text{def}} \preccurlyeq$. Some explanation for this apparently strange action is provided by the following connection to designs.

Remark. The intention underlying the definition of $\mathbb{1}$ is to enforce an upper closure of the image of each state with respect to $\preccurlyeq$. Traces of such a procedure can be found in the healthiness conditions of designs: 'The healthiness condition H2 states formally that the predicate R is upward closed in the variable ok': as ok' changes from false to true, R cannot change from true to false.'[HH98, page 83] Since H3 implies H2, every normal design is upper closed in this way. For normal designs, [Gut06] shows how to replace the auxiliary variables ok and ok' by a special element that corresponds to ∞ in our present discussion. In particular, [ibidem, Lemma 9.2] formulates the upper closure as $\overline{R}\top \cap R \subseteq \overline{V^{\smile}}$, where V corresponds to the vector $\vec{x}{=}\infty$. By the Schröder law,

$$\overline{R}\top \cap R \subseteq \overline{V^{\smile}} \Leftrightarrow \overline{R}\top \cap V^{\smile} \subseteq \overline{R} \Leftrightarrow \overline{R}V^{\smile} \subseteq \overline{R} \Leftrightarrow RV \subseteq R \Leftrightarrow RV \cup R = R \Leftrightarrow R(V \cup \mathbb{I}) = R.$$

If the state consists of only one variable, $V \cup \mathbb{I} = (x{=}\infty) \cup (x{=}x') = (x \preccurlyeq x')$, and we obtain the right unit law $R \mathbin{;} \preccurlyeq = R$. Our definition of $\mathbb{1}$ refines this by distinguishing individual variables. The refined right unit law corresponding to the healthiness condition H3 of designs is stated in Definition 52. (End of remark)

Since $\preccurlyeq$ is a partial order, we have $\mathbb{I} \subseteq \mathbb{1}$ and $\mathbb{1} \,;\, \mathbb{1} = \mathbb{1}$ and $\mathbb{1} \cap \mathbb{1}^\smile = \mathbb{I}$ stating reflexivity and transitivity and antisymmetry, respectively. These properties will be used without further reference in the following. As usual, skip should be a left and right unit of sequential composition. This is formally required by the healthiness conditions $\mathscr{H}_L$ and $\mathscr{H}_R$. We furthermore use their combination $\mathscr{H}_E$.

Definition 52.

$$\begin{array}{lll} \mathscr{H}_L(P) & \Leftrightarrow_{\text{def}} & \mathbb{1} \,;\, P \,=\, P, \\ \mathscr{H}_R(P) & \Leftrightarrow_{\text{def}} & P \,;\, \mathbb{1} \,=\, P, \\ \mathscr{H}_E(P) & \Leftrightarrow_{\text{def}} & \mathbb{1} \,;\, P \,;\, \mathbb{1} \,=\, P. \end{array}$$

By reflexivity of $\mathbb{1}$, it suffices to demand $\subseteq$ instead of equality. Observe furthermore that $\mathscr{H}_E(P) \Leftrightarrow \mathscr{H}_L(P) \wedge \mathscr{H}_R(P)$. The part $(\Leftarrow)$ follows by $\mathbb{1}P\mathbb{1} = P\mathbb{1} = P$ and the part $(\Rightarrow)$ follows by $\mathbb{1}P = \mathbb{1}\mathbb{1}P\mathbb{1} = \mathbb{1}P\mathbb{1} = P = \mathbb{1}P\mathbb{1} = \mathbb{1}P\mathbb{1}\mathbb{1} = P\mathbb{1}$. The rest of this section is devoted to giving definitions of programming constructs that satisfy or preserve these healthiness conditions.

Lemma 53. *Skip, sequential composition, non-deterministic choice and conjunction satisfy or preserve $\mathscr{H}_L$, $\mathscr{H}_R$ and $\mathscr{H}_E$ as follows.*

1. *$\mathscr{H}_E(\mathbb{1})$.*

2. *$\mathscr{H}_L(P) \Rightarrow \mathscr{H}_L(P \,;\, Q)$ and $\mathscr{H}_R(Q) \Rightarrow \mathscr{H}_R(P \,;\, Q)$. In particular, let $X \in \{E, L, R\}$, then $\mathscr{H}_X(P) \wedge \mathscr{H}_X(Q) \Rightarrow \mathscr{H}_X(P \,;\, Q)$.*

3. *Let $X \in \{E, L, R\}$, then $(\forall P \in S : \mathscr{H}_X(P)) \Rightarrow \mathscr{H}_X(\bigcup S)$. In particular, $\mathscr{H}_E(\perp\!\!\!\perp)$.*

4. *Let $X \in \{E, L, R\}$, then $(\forall P \in S : \mathscr{H}_X(P)) \Rightarrow \mathscr{H}_X(\bigcap S)$. In particular, $\mathscr{H}_E(\top)$.*

Proof.

1. $\mathbb{1}\mathbb{1}\mathbb{1} = \mathbb{1}\mathbb{1} = \mathbb{1}$.

2. $\mathbb{1}PQ = PQ$ and $PQ\mathbb{1} = PQ$ by the respective assumptions. The remaining claims are immediate consequences.

3. The case $X = L$ follows by $\mathbb{1}(\bigcup S) = (\bigcup_{P \in S} \mathbb{1}P) = (\bigcup_{P \in S} P) = \bigcup S$. The case $X = R$ follows by $(\bigcup S)\mathbb{1} = (\bigcup_{P \in S} P\mathbb{1}) = (\bigcup_{P \in S} P) = \bigcup S$. The case $X = E$ is immediate from these two. Using $S = \emptyset$ yields $\mathscr{H}_E(\perp\!\!\!\perp)$.

4. The case $X = L$ follows by $\mathbb{1}(\bigcap S) \subseteq (\bigcap_{P \in S} \mathbb{1}P) = (\bigcap_{P \in S} P) = (\bigcap S) \subseteq \mathbb{1}(\bigcap S)$. The case $X = R$ follows by $(\bigcap S)\mathbb{1} \subseteq (\bigcap_{P \in S} P\mathbb{1}) = (\bigcap_{P \in S} P) = (\bigcap S) \subseteq (\bigcap S)\mathbb{1}$. The case $X = E$ is immediate from these two. Using $S = \emptyset$ yields $\mathscr{H}_E(\top)$. □

Corollary 54. *Let $X \in \{E, L, R\}$, then the relations satisfying $\mathscr{H}_X$ form a complete lattice. Let f be isotone and $\mathscr{H}_X$-preserving, then $\mathscr{H}_X(\mu f)$ and $\mathscr{H}_X(\nu f)$.*

Proof. The function $P \mapsto \mathbb{1} \,;\, P$ is a closure operator (isotone, increasing and idempotent) by isotony of ; and reflexivity and transitivity of $\mathbb{1}$. Its image, comprising the relations that satisfy $\mathscr{H}_L$, thus forms a complete lattice [DP02, Proposition 7.2]. That lattice is closed under any $\mathscr{H}_L$-preserving function f and under suprema of chains, even arbitrary suprema by Lemma 53.3. It therefore contains μf by Lemma 88 of Section 3.6.1. The lattice is dually closed under infima by Lemma 53.4 and hence contains νf by Corollary 89. The same arguments apply to $P \mapsto P \,;\, \mathbb{1}$ and $\mathscr{H}_R$, as well as $P \mapsto \mathbb{1} \,;\, P \,;\, \mathbb{1}$ and $\mathscr{H}_E$. □

3.2.2 Expressions

Consider the assignment statement $x_i{:=}e$ that sets the new value of x_i to e and keeps the values of the other variables unchanged. UTP therefore defines $x_i{:=}e =_{\text{def}} x_i'{=}e \cap \vec{x}_{\bar{i}}'{=}\vec{x}_{\bar{i}}$. The expression e may depend on the old values $\vec{x}$ of the variables, as is elaborated in the following remark on the nature of expressions. More generally, the simultaneous assignment to several variables $\vec{x}_I$ is given by $\vec{x}_I{:=}\vec{e}_I =_{\text{def}} \vec{x}_I'{=}\vec{e}_I \cap \vec{x}_{\bar{I}}'{=}\vec{x}_{\bar{I}}$, where $\vec{e}_I$ is a sequence of expressions. Such an assignment can always be extended to a full assignment $\vec{x}{:=}\vec{e} = \vec{x}'{=}\vec{e}$ by choosing $e_i{=}x_i$ for any unaffected variable x_i.

Remark. Except for the discussion in Section 3.6.2 we are not concerned with the syntactic structure of expressions. Neither do we try to squeeze them into a particular relational form. An expression e can be regarded as an object that, given certain values $\vec{x}$ for the input variables, yields *exactly one* value $e(\vec{x})$ from the range D, the expression's type. Conditions are expressions of Boolean type, that is, where $D = \{\infty, \bot, \mathit{true}, \mathit{false}\}$. It is thus natural to interpret e as a function $e : D_{1..m} \to D$ which is of course also a relation $e : D_{1..m} \leftrightarrow D$. This view nicely extends to sequences of expressions $\vec{e} : D_{1..m} \to D_{1..n}$ and hence also $\vec{e} : D_{1..m} \leftrightarrow D_{1..n}$. Observe that the latter relation is precisely $\vec{x}'{=}\vec{e}$. If there are as many expressions as input variables ($m = n$) we obtain the full assignment $\vec{x}{:=}\vec{e}$, a homogeneous relation. This correspondence of expressions, functions, relations and assignments motivates Definition 56. (End of remark)

The new relation modelling the assignment is $\vec{x}{\leftarrow}\vec{e} =_{\text{def}} \mathbb{1} \,;\, \vec{x}{:=}\vec{e} \,;\, \mathbb{1}$. Forming the closure with respect to $\preccurlyeq$ guarantees that the healthiness conditions $\mathscr{H}_L$ and $\mathscr{H}_R$ are satisfied. This and other properties of the assignment statement are shown in the following lemma. Upper closure of the images perspicuously appears in part 3 which intuitively states that $\mathbb{T}$ models the never terminating program. Also skip can be expressed as an assignment by $\mathbb{1} = \vec{x}{\leftarrow}\vec{x}$. If the right hand side of a (simultaneous) assignment is a sequence of identical values, it is abbreviated by that value, as ∞ in parts 2 and 3.

Lemma 55.

1. $\mathscr{H}_E(\vec{x}{\leftarrow}\vec{e})$.
2. $\mathbb{1}^{\smile} = \bigcup_{I \subseteq 1..m} \vec{x}_I{:=}\infty$.
3. $\vec{x}{\leftarrow}\infty \;=\; \mathbb{T}$.
4. *Let* $\vec{c} \in D_{1..n}$ *such that* $\infty{\notin}\vec{c}$, *then* $\vec{x}{\leftarrow}\vec{c} \;=\; \vec{x}'{=}\vec{c} \;=\; \vec{x}{:=}\vec{c}$.
5. *Let* $\vec{c} \in D_{1..n}$ *such that* $\infty{\notin}\vec{c}$. *Let* $\vec{x}'{=}\vec{c} : D_{1..m} \leftrightarrow D_{1..n}$ *be a (heterogeneous) relation, then* $\mathscr{H}_E(\vec{x}'{=}\vec{c})$.

Proof.

1. The claim follows immediately by Lemmas 53.1 and 53.2.
2. To show the part ($\subseteq$), let $\vec{x}' \preccurlyeq \vec{x}$. Using $I =_{\text{def}} \{i \mid x_i'{=}\infty\}$, clearly $\vec{x}_I'{=}\infty \wedge \infty{\notin}\vec{x}_{\bar{I}}'$, thus $\vec{x}_{\bar{I}}'{=}\vec{x}_{\bar{I}}$. Therefore $(\vec{x}, \vec{x}') \in \vec{x}_I'{=}\infty \cap \vec{x}_{\bar{I}}'{=}\vec{x}_{\bar{I}} = \vec{x}_I{:=}\infty$. To show the part ($\supseteq$), let $(\vec{x}, \vec{x}') \in \bigcup_{I \subseteq 1..m} \vec{x}_I{:=}\infty$, hence $(\vec{x}, \vec{x}') \in \vec{x}_I{:=}\infty = \vec{x}_I'{=}\infty \cap \vec{x}_{\bar{I}}'{=}\vec{x}_{\bar{I}}$ for some $I \subseteq 1..m$. As a consequence $\vec{x}_I'{=}\infty{\preccurlyeq}\vec{x}_I$ and $\vec{x}_{\bar{I}}'{\preccurlyeq}\vec{x}_{\bar{I}}$, thus $\vec{x}' \preccurlyeq \vec{x}$.
3. The claim follows by $\vec{x}{\leftarrow}\infty = \mathbb{1} \,;\, \vec{x}{:=}\infty \,;\, \mathbb{1} \supseteq \vec{x}{:=}\infty \,;\, (\vec{x}{:=}\infty)^{\smile} = \vec{x}'{=}\infty \,;\, \vec{x}{=}\infty = \mathbb{T}$ using reflexivity of $\mathbb{1}$, part 2 and the definition of the assignment relation.
4. Using vector properties, the claim follows by

$$\begin{aligned}\vec{x}{\leftarrow}\vec{c} \;&=\; \mathbb{1} \,;\, \vec{x}{:=}\vec{c} \,;\, \mathbb{1} \;=\; \mathbb{1} \,;\, \vec{x}'{=}\vec{c} \,;\, \mathbb{1} \;=\; (\mathbb{1} \,;\, \mathbb{T} \cap \vec{x}'{=}\vec{c}) \,;\, \mathbb{1} \;=\; \vec{x}'{=}\vec{c} \,;\, \mathbb{1} \\ &=\; \mathbb{T} \,;\, (\vec{x}{=}\vec{c} \cap \vec{x}{\preccurlyeq}\vec{x}') \;=\; \mathbb{T} \,;\, (\vec{x}{=}\vec{c} \cap \vec{x}'{=}\vec{c}) \;=\; \mathbb{T} \,;\, \vec{x}{=}\vec{c} \cap \vec{x}'{=}\vec{c} \\ &=\; \vec{x}'{=}\vec{c} \;=\; \vec{x}{:=}\vec{c}.\end{aligned}$$

5. Using the homogeneous $\vec{x}'{=}\vec{c} : D_{1..n} \leftrightarrow D_{1..n}$ in the middle term and part 4 in the second step, we have $\vec{x}'{=}\vec{c} \;=\; \top\!\top \;;\; \vec{x}'{=}\vec{c} \;=\; \top\!\top \;;\; \vec{x}{\leftarrow}\vec{c}$. From $\mathscr{H}_E(\top\!\top)$ by Lemma 53.4 and $\mathscr{H}_E(\vec{x}{\leftarrow}\vec{c})$ by part 1 we thus have $\mathscr{H}_E(\vec{x}'{=}\vec{c})$ by Lemma 53.2. □

Resuming our introductory example we now have $x_1, x_2{\leftarrow}\bot, 2 \;;\; x_1{\leftarrow}x_2 = x_1, x_2{\leftarrow}2, 2$ and furthermore $\top\!\top \;;\; x_1, x_2{\leftarrow}2, 2 = x_1, x_2, \vec{x}_{3..n}{\leftarrow}2, 2, \infty$. This demonstrates that computations in our setting are indeed non-strict.

To deal with the conditional and later also with the assignment, we need to restrict the expressions that occur on the right hand side of assignments and as conditions. We assume that the expressions are isotone with respect to $\preccurlyeq$ as captured by the following condition.

Definition 56. Let E be a partial order. The sequence of expressions $\vec{e}$ is *isotone* with respect to E iff

$$\mathscr{T}_E(\vec{e}) \quad \Leftrightarrow_{\text{def}} \quad E \;;\; \vec{x}'{=}\vec{e} \;\subseteq\; \vec{x}'{=}\vec{e} \;;\; E.$$

At this stage we need $\mathscr{T}_{\preccurlyeq}(\vec{e})$ or $\preccurlyeq \;;\; \vec{x}'{=}\vec{e} \;\subseteq\; \vec{x}'{=}\vec{e} \;;\; \preccurlyeq$. If the expression e is viewed as a function, then $\mathscr{T}_{\preccurlyeq}(e)$ amounts to the usual isotony in partially ordered sets, namely $\forall \vec{x}, \vec{y} : \vec{x} \preccurlyeq \vec{y} \;\Rightarrow\; e(\vec{x}) \preccurlyeq e(\vec{y})$. Its relational formulation appears, for example, in [Sch06]. Another interpretation of $\mathscr{T}_{\preccurlyeq}$ is discussed in Section 3.6.2 where we also show that any expression composed of constants, variables and strict functions is isotone, thus the restriction is not too severe. Observe that $\vec{e}$ is isotone with respect to $\preccurlyeq$ iff it is with respect to $\succcurlyeq$ since $\vec{x}'{=}\vec{e}$ is a mapping and

$$EF \subseteq FE \Leftrightarrow E \subseteq FEF^\smile \Leftrightarrow E^\smile \subseteq (FEF^\smile)^\smile = FE^\smile F^\smile \Leftrightarrow E^\smile F \subseteq FE^\smile$$

for any partial order E and mapping F.

Let us elaborate the assignment $\vec{x}{\leftarrow}\vec{e}$ assuming $\mathscr{T}_{\preccurlyeq}(\vec{e})$. By Lemma 57.3 below, we obtain $\vec{x}{\leftarrow}\vec{e} \;=\; \mathbb{1} \;;\; \vec{x}{:=}\vec{e} \;;\; \mathbb{1} \;=\; \vec{x}'{=}\vec{e} \;;\; \mathbb{1} \;=\; \{(\vec{x}, \vec{x}') \mid \exists \vec{y} : \vec{y}{=}\vec{e}(\vec{x}) \wedge \vec{y} \preccurlyeq \vec{x}'\} \;=\; \{(\vec{x}, \vec{x}') \mid \vec{e}(\vec{x}) \preccurlyeq \vec{x}'\}$. Hence the successor states of $\vec{x}$ under this assignment comprise the usual successor $\vec{e}(\vec{x})$ and its upper closure with respect to $\preccurlyeq$.

Consider the conditional statement $(P \lhd b \rhd Q) = (b \cap P) \cup (\bar{b} \cap Q)$ of UTP, where the condition b is treated as a vector. In common terms this reads as 'if b then P else Q' but the definition does not take into account the possibility of b being undefined. Its extension to designs $(P \lhd b \rhd Q) = (\mathcal{D}b \Rightarrow (b \cap P) \cup (\bar{b} \cap Q))$ does, but yields non-termination whenever the condition b is undefined. We therefore have to adapt the definition. To this end, we no longer treat conditions as vectors but as expressions with values in $\{\infty, \bot, \mathit{true}, \mathit{false}\}$. Nevertheless, if b is a condition, the relation $b{=}c$ is a vector for any $c \in \{\infty, \bot, \mathit{true}, \mathit{false}\}$. Recalling how relational constants are specified, and using $\vec{x}_{1..m}$ as input variables, we obtain that $b{=}c = \{(\vec{x}, \vec{x}') \mid b(\vec{x}){=}c\} : D_{1..m} \leftrightarrow D_{1..n}$ for arbitrary $D_{1..n}$ depending on the context.

The new relation modelling the conditional is

$$(P \blacktriangleleft b \blacktriangleright Q) =_{\text{def}} b{=}\infty \;\cup\; (b{=}\bot \;\cap\; \vec{x}'{=}\bot) \;\cup\; (b{=}\mathit{true} \;\cap\; P) \;\cup\; (b{=}\mathit{false} \;\cap\; Q).$$

The effect of an undefined condition in a conditional statement is to set all variables of the current state undefined. Indeed, if P and Q are homogeneous we can write $(b{=}\bot \cap \vec{x}{\leftarrow}\bot)$ instead of $(b{=}\bot \cap \vec{x}'{=}\bot)$ by Lemma 55.4. Similarly, $b{=}\infty$ is a simplification of $(b{=}\infty \cap \vec{x}{\leftarrow}\infty)$ by Lemma 55.3. This models the fact that the evaluation of b is always necessary if the execution of the conditional is. Any non-termination or undefinedness is thus propagated. As a notational convention, we stipulate that the conditional $\_ \blacktriangleleft \_ \blacktriangleright \_$ has a lower precedence than meet and join.

Lemma 57. *Vectors and the conditional satisfy $\mathscr{H}_L$, $\mathscr{H}_R$ and $\mathscr{H}_E$ as follows.*

1. $V = V\top\!\top \Rightarrow \mathscr{H}_R(V)$.
2. $\mathscr{H}_R(P) \wedge \mathscr{H}_R(Q) \Rightarrow \mathscr{H}_R(P \blacktriangleleft b \blacktriangleright Q)$.
3. $\mathscr{T}_{\preccurlyeq}(\vec{e}) \Rightarrow \mathscr{H}_L(\vec{x}'{=}\vec{e} \;;\; \mathbb{1})$.
4. $\mathscr{T}_{\preccurlyeq}(b) \wedge \mathscr{H}_L(P) \wedge \mathscr{H}_L(Q) \Rightarrow \mathscr{H}_L(P \blacktriangleleft b \blacktriangleright Q)$.

Proof.

1. $V\mathbb{1} = V\,\mathbb{T}\,\mathbb{1} = V\,\mathbb{T} = V$.

2. By part 1, Lemma 55.5 and the assumptions,

$$\begin{array}{ll} & (P \lhd b \rhd Q)\,;\,\mathbb{1} \\ = & (b{=}\infty \cup (b{=}\bot \cap \vec{x}'{=}\bot) \cup (b{=}\mathit{true} \cap P) \cup (b{=}\mathit{false} \cap Q))\,;\,\mathbb{1} \\ = & b{=}\infty\,;\,\mathbb{1} \cup (b{=}\bot \cap \vec{x}'{=}\bot\,;\,\mathbb{1}) \cup (b{=}\mathit{true} \cap P\,;\,\mathbb{1}) \cup (b{=}\mathit{false} \cap Q\,;\,\mathbb{1}) \\ = & b{=}\infty \cup (b{=}\bot \cap \vec{x}'{=}\bot) \cup (b{=}\mathit{true} \cap P) \cup (b{=}\mathit{false} \cap Q) \\ = & (P \lhd b \rhd Q). \end{array}$$

3. $\mathbb{1}\,;\,\vec{x}'{=}\vec{e}\,;\,\mathbb{1} \subseteq \vec{x}'{=}\vec{e}\,;\,\mathbb{1}\,;\,\mathbb{1} = \vec{x}'{=}\vec{e}\,;\,\mathbb{1} \subseteq \mathbb{1}\,;\,\vec{x}'{=}\vec{e}\,;\,\mathbb{1}$.

4. It suffices to show $\mathbb{1}\,;\,(P \lhd b \rhd Q) \subseteq (P \lhd b \rhd Q)$. We distinguish four cases, depending on the value of b. First, $b{=}\infty \cap \mathbb{1}\,;\,(P \lhd b \rhd Q) \subseteq b{=}\infty$. For the remaining three cases, let $c \in \{\bot, \mathit{true}, \mathit{false}\}$. Then, using the single variable x,

$$\succcurlyeq\,;\,x{=}c = (x' \preccurlyeq x \cap x'{=}c)\,;\,\mathbb{T} \subseteq c{\preccurlyeq}x\,;\,\mathbb{T} = x{=}c\,;\,\mathbb{T} = x{=}c.$$

   By the Schröder law, $\mathbb{1}\,;\,x{\neq}c \subseteq x{\neq}c \subseteq \mathbb{1}\,;\,x{\neq}c$, hence

$$\begin{array}{rl} x'{=}b\,;\,\mathbb{1}\,;\,x{\neq}c = & x'{=}b\,;\,x{\neq}c = (x'{=}b \cap x'{\neq}c)\,;\,\mathbb{T} = (x'{=}b \cap b{\neq}c)\,;\,\mathbb{T} \\ = & b{\neq}c \cap x'{=}b\,;\,\mathbb{T} = b{\neq}c. \end{array}$$

   By part 3, we therefore have $\mathbb{1}\,;\,b{\neq}c = \mathbb{1}\,;\,x'{=}b\,;\,\mathbb{1}\,;\,x{\neq}c = x'{=}b\,;\,\mathbb{1}\,;\,x{\neq}c = b{\neq}c$. Again by the Schröder law, $\mathbb{1}^\smile\,;\,b{=}c \subseteq b{=}c$, hence by the Dedekind law

$$\begin{array}{rl} b{=}c \cap \mathbb{1}\,;\,(P \lhd b \rhd Q) \subseteq & b{=}c \cap \mathbb{1}\,;\,(\mathbb{1}^\smile\,;\,b{=}c \cap (P \lhd b \rhd Q)) \\ \subseteq & b{=}c \cap \mathbb{1}\,;\,(b{=}c \cap (P \lhd b \rhd Q)). \end{array}$$

   This states that the relation $b{=}c$ can be imported into the sequential composition. Therefore, finishing the remaining three cases,

$$\begin{array}{rl} b{=}\bot \cap \mathbb{1}\,;\,(P \lhd b \rhd Q) \subseteq & b{=}\bot \cap \mathbb{1}\,;\,(b{=}\bot \cap \vec{x}'{=}\bot) \subseteq b{=}\bot \cap \vec{x}'{=}\bot, \\ b{=}\mathit{true} \cap \mathbb{1}\,;\,(P \lhd b \rhd Q) \subseteq & b{=}\mathit{true} \cap \mathbb{1}\,;\,(b{=}\mathit{true} \cap P) \subseteq b{=}\mathit{true} \cap \mathbb{1}\,;\,P \\ = & b{=}\mathit{true} \cap P, \\ b{=}\mathit{false} \cap \mathbb{1}\,;\,(P \lhd b \rhd Q) \subseteq & b{=}\mathit{false} \cap \mathbb{1}\,;\,(b{=}\mathit{false} \cap Q) \subseteq b{=}\mathit{false} \cap \mathbb{1}\,;\,Q \\ = & b{=}\mathit{false} \cap Q. \end{array}$$

   The claim is obtained by joining both sides of the four inequalities. □

3.2.3 Local variables

Variables are added to and removed from the current state by the variable declaration and undeclaration statements. In UTP they are **var** $x_i = (\exists x_i : \mathbb{I})$ and **end** $x_i = (\exists x_i' : \mathbb{I})$, respectively. These relations are not homogeneous: The declaration includes x_i' in its range but not x_i in its domain. Conversely, the undeclaration includes x_i in its domain but not x_i' in its range. Therefore actually both **var** $x_i = (\vec{x}'_{\bar{i}}{=}\vec{x}_{\bar{i}})$ and **end** $x_i = (\vec{x}'_{\bar{i}}{=}\vec{x}_{\bar{i}})$, but the different dimensions have to be taken into account. Again we have to adapt the statements to respect the healthiness conditions $\mathscr{H}_L$ and $\mathscr{H}_R$. We moreover generalise them to simultaneously handle several variables. The new relations modelling variable (un)declaration are $\boldsymbol{var}\,\vec{x}_K =_{\text{def}} (\exists \vec{x}_K : \mathbb{1})$ and $\boldsymbol{end}\,\vec{x}_K =_{\text{def}} (\exists \vec{x}'_K : \mathbb{1})$.

Lemma 58. *Variable (un)declaration satisfies the following laws.*

1. $\boldsymbol{var}\,\vec{x}_K = A\,;\,(\exists \vec{x}_K : B)\,;\,C$ *for any* $\{\mathbb{1}\} \subseteq \{A, B, C\} \subseteq \{\mathbb{1}, \mathbb{I}\}$ *and* $\mathscr{H}_E(\boldsymbol{var}\,\vec{x}_K)$.

2. $\boldsymbol{end}\,\vec{x}_K = A\,;\,(\exists \vec{x}'_K : B)\,;\,C$ *for any* $\{\mathbb{1}\} \subseteq \{A, B, C\} \subseteq \{\mathbb{1}, \mathbb{I}\}$ *and* $\mathscr{H}_E(\boldsymbol{end}\,\vec{x}_K)$.

Proof.

1. We first show $(\exists \vec{x}_K : \mathbb{I})\ ;\ \mathbb{1} = \mathbb{1}\ ;\ (\exists \vec{x}_K : \mathbb{I})$. To this end, let $J = I \cup K$, then

$$\begin{array}{l}(\vec{x}_I, \vec{z}_J) \in (\exists \vec{x}_K : \mathbb{I})\ ;\ \mathbb{1} \Leftrightarrow (\exists \vec{y}_J : (\exists \vec{x}_K : \vec{x}_J{=}\vec{y}_J)\ \wedge\ \vec{y}_J{\preccurlyeq}\vec{z}_J)\\ \Leftrightarrow (\exists \vec{y}_J : \vec{x}_I{=}\vec{y}_I\ \wedge\ \vec{y}_J{\preccurlyeq}\vec{z}_J) \Leftrightarrow \vec{x}_I{\preccurlyeq}\vec{z}_I \Leftrightarrow (\exists \vec{y}_I : \vec{x}_I{\preccurlyeq}\vec{y}_I\ \wedge\ \vec{y}_I{=}\vec{z}_I)\\ \Leftrightarrow (\exists \vec{y}_I : \vec{x}_I{\preccurlyeq}\vec{y}_I\ \wedge\ \exists \vec{y}_K : \vec{y}_J{=}\vec{z}_J) \Leftrightarrow (\vec{x}_I, \vec{z}_J) \in \mathbb{1}\ ;\ (\exists \vec{x}_K : \mathbb{I}).\end{array}$$

   Hence $\boldsymbol{var}\ \vec{x}_K = (\exists \vec{x}_K : \mathbb{1}) = (\exists \vec{x}_K : \mathbb{I})\ ;\ \mathbb{1} = \mathbb{1}\ ;\ (\exists \vec{x}_K : \mathbb{I})$, which implies $\mathscr{H}_E(\boldsymbol{var}\ \vec{x}_K)$. This entails the remaining claims in turn.

2. We first show $(\exists \vec{x}'_K : \mathbb{I})\ ;\ \mathbb{1} = \mathbb{1}\ ;\ (\exists \vec{x}'_K : \mathbb{I})$. To this end, let $I = J \cup K$, then

$$\begin{array}{l}(\vec{x}_I, \vec{z}_J) \in (\exists \vec{x}'_K : \mathbb{I})\ ;\ \mathbb{1} \Leftrightarrow (\exists \vec{y}_J : (\exists \vec{y}_K : \vec{x}_I{=}\vec{y}_I)\ \wedge\ \vec{y}_J{\preccurlyeq}\vec{z}_J)\\ \Leftrightarrow (\exists \vec{y}_J : \vec{x}_J{=}\vec{y}_J\ \wedge\ \vec{y}_J{\preccurlyeq}\vec{z}_J) \Leftrightarrow \vec{x}_J{\preccurlyeq}\vec{z}_J \Leftrightarrow (\exists \vec{y}_I : \vec{x}_I{\preccurlyeq}\vec{y}_I\ \wedge\ \vec{y}_J{=}\vec{z}_J)\\ \Leftrightarrow (\exists \vec{y}_I : \vec{x}_I{\preccurlyeq}\vec{y}_I\ \wedge\ \exists \vec{z}_K : \vec{y}_I{=}\vec{z}_I) \Leftrightarrow (\vec{x}_I, \vec{z}_J) \in \mathbb{1}\ ;\ (\exists \vec{x}'_K : \mathbb{I}).\end{array}$$

   Hence $\boldsymbol{end}\ \vec{x}_K = (\exists \vec{x}'_K : \mathbb{1}) = \mathbb{1}\ ;\ (\exists \vec{x}'_K : \mathbb{I}) = (\exists \vec{x}'_K : \mathbb{I})\ ;\ \mathbb{1}$, which implies $\mathscr{H}_E(\boldsymbol{end}\ \vec{x}_K)$. This entails the remaining claims in turn. □

Since $\boldsymbol{var}\ \vec{x}_K = \mathbb{1}\ ;\ (\exists \vec{x}_K : \mathbb{I})$ by Lemma 58.1, the declaration itself does not impose any restriction on the new variables. This means that accessing a declared but uninitialised variable results in non-termination. A more appropriate statement that yields undefinedness instead can be obtained by using $\boldsymbol{var}\ \vec{x}_K\ ;\ \vec{x}_K{\leftarrow}\bot$. Alternatively, the language designer may opt to allow only initialised variable declarations $\boldsymbol{var}\ \vec{x}_K{\leftarrow}\vec{e}_K =_{\text{def}} \boldsymbol{var}\ \vec{x}_K\ ;\ \vec{x}_K{\leftarrow}\vec{e}_K$. In particular, the expressions $\vec{e}_K$ must not refer to the new variables $\vec{x}_K$ in this case.

The alphabet extension is UTP's mechanism to hide local variables from recursive calls. It is given by $P_{+x_i} = x'_i{=}x_i\ \cap\ \mathbf{end}\ x_i\ ;\ P\ ;\ \mathbf{var}\ x_i$ which improves on [HH98] by making explicit the change of P's type. The domain of P is extended by x_i and the range by x'_i, and both are equated.

Remark. The term $\mathbf{end}\ x_i\ ;\ P\ ;\ \mathbf{var}\ x_i$ may be viewed as a dimension extension, in contrast to $\exists$ that restricts the dimensions of relations. It follows from $\mathbf{var}\ x_i\ ;\ \mathbf{end}\ x_i = \mathbb{I}$ and $\mathbf{end}\ x_i\ ;\ \mathbf{var}\ x_i \supseteq \mathbb{I}$ that dimension restriction and extension are the lower and upper adjoints of a Galois connection. See [DP02] for a definition and properties of Galois connections which are also used by UTP to link theories. (End of remark)

We generalise the alphabet extension to several variables and heterogeneous relations. Let $P : D_I \leftrightarrow D_J$, then the new alphabet extension of P is $P^{+\vec{x}_K} : D_{I \cup K} \leftrightarrow D_{J \cup K}$ given by $P^{+\vec{x}_K} =_{\text{def}} \boldsymbol{end}\ \vec{x}_I\ ;\ \boldsymbol{var}\ \vec{x}_J\ \cap\ \boldsymbol{end}\ \vec{x}_K\ ;\ P\ ;\ \boldsymbol{var}\ \vec{x}_K$. Intuitively, the part $\boldsymbol{end}\ \vec{x}_I\ ;\ \boldsymbol{var}\ \vec{x}_J$ preserves the values of $\vec{x}_K$ and the part $\boldsymbol{end}\ \vec{x}_K\ ;\ P\ ;\ \boldsymbol{var}\ \vec{x}_K$ applies P to $\vec{x}_I$ to obtain $\vec{x}_J$. Just as the variable undeclaration may be seen as a projection, the alphabet extension is an instance of relational parallel composition as shown in the following lemma.

Lemma 59. *Let $P : D_I \leftrightarrow D_J$ and $Q : D_K \leftrightarrow D_L$ such that $I \cap K = \emptyset = J \cap L$.*

1. $\mathbb{1}P\mathbb{1} \| \mathbb{1}Q\mathbb{1} = \boldsymbol{end}\ \vec{x}_K\ ;\ P\ ;\ \boldsymbol{var}\ \vec{x}_L\ \cap\ \boldsymbol{end}\ \vec{x}_I\ ;\ Q\ ;\ \boldsymbol{var}\ \vec{x}_J$, thus $P^{+\vec{x}_K} = \mathbb{1}P\mathbb{1}_I\|_K\mathbb{1}$.

2. $\mathscr{H}_E(P) \wedge \mathscr{H}_E(Q) \Rightarrow \mathscr{H}_E(P \| Q)$, in particular $\mathscr{H}_E(P^{+\vec{x}_K})$.

Proof.

1. By Lemma 58 we obtain

$$\begin{array}{l}\quad \boldsymbol{end}\ \vec{x}_K\ ;\ P\ ;\ \boldsymbol{var}\ \vec{x}_L\ \cap\ \boldsymbol{end}\ \vec{x}_I\ ;\ Q\ ;\ \boldsymbol{var}\ \vec{x}_J\\ = (\exists \vec{x}'_K : \mathbb{1})\ ;\ P\ ;\ (\exists \vec{x}_L : \mathbb{1})\ \cap\ (\exists \vec{x}'_I : \mathbb{1})\ ;\ P\ ;\ (\exists \vec{x}_J : \mathbb{1})\\ = (\exists \vec{x}'_K : \mathbb{I})\ ;\ \mathbb{1}\ ;\ P\ ;\ \mathbb{1}\ ;\ (\exists \vec{x}_L : \mathbb{I})\ \cap\ (\exists \vec{x}'_I : \mathbb{I})\ ;\ \mathbb{1}\ ;\ Q\ ;\ \mathbb{1}\ ;\ (\exists \vec{x}_J : \mathbb{I})\\ = \mathbb{1}P\mathbb{1} \| \mathbb{1}Q\mathbb{1}.\end{array}$$

   It follows that $P^{+\vec{x}_K} = \mathbb{1}P\mathbb{1}_I\|_K\mathbb{1}\mathbb{I}\mathbb{1} = \mathbb{1}P\mathbb{1}_I\|_K\mathbb{1}$.

2. $\mathbb{1}\ ;\ (P\|Q)\ ;\ \mathbb{1} = (\mathbb{1}\|\mathbb{1})\ ;\ (P\|Q)\ ;\ (\mathbb{1}\|\mathbb{1}) = (\mathbb{1}P\mathbb{1}\|\mathbb{1}Q\mathbb{1}) = (P\|Q)$ using Lemmas 93.5 and 93.4 of Section 3.6.3. The particular claim follows by part 1 and Lemmas 53.1 and 53.2. □

We usually omit the indices as in $P^{+\vec{x}_K} = \mathbb{1}P\mathbb{1}\|\mathbb{1}$, tacitly assuming that the left operand applies to $\vec{x}_I$ and the right operand to $\vec{x}_K$. This simplifies to $P^{+\vec{x}_K} = P\|\mathbb{1}$ if $\mathscr{H}_E(P)$ holds. It is typically as complex to prove a result for the more general $P\|Q$ as it is for $P^{+\vec{x}_K}$. We therefore use the parallel composition in this chapter, keeping in mind that it includes the alphabet extension as a special case.

3.2.4 Isotony and neutrality

We have now introduced a selection of programming constructs as summarised in the following definition. This selection is inspired by UTP and rich enough to yield a basic programming and specification language. We are not concerned with further constructs that can be provided in terms of the available, or by new relations. Our intention is rather to show that we build a reasonable theory.

Definition 60. We use the following relations and operations:

skip	$\mathbb{1} =_{\text{def}} \preccurlyeq$
assignment	$\vec{x}{\leftarrow}\vec{e} =_{\text{def}} \mathbb{1}\ ;\ \vec{x}{:=}\vec{e}\ ;\ \mathbb{1}$
variable declaration	$\textbf{\textit{var}}\ \vec{x}_K =_{\text{def}} (\exists \vec{x}_K : \mathbb{1})$
variable undeclaration	$\textbf{\textit{end}}\ \vec{x}_K =_{\text{def}} (\exists \vec{x}'_K : \mathbb{1})$
parallel composition	$P\|Q$
sequential composition	$P\ ;\ Q$
conditional	$(P \lhd b \rhd Q) =_{\text{def}} b{=}\infty \cup (b{=}\bot \cap \vec{x}'{=}\bot) \cup (b{=}\mathit{true} \cap P) \cup (b{=}\mathit{false} \cap Q)$
non-deterministic choice	$\bigcup_{P\in S} P$
conjunction	$\bigcap_{P\in S} P$
greatest fixpoint	$\nu f = \bigcup\{P \mid f(P) = P\}$
least fixpoint	$\mu f = \bigcap\{P \mid f(P) = P\}$

No new definitions are given for the sequential composition, the non-deterministic choice and the fixpoint operators. They are just the familiar operations of relation algebra. This simplifies reasoning because it enables applying familiar laws, like distribution of ; over $\cup$, also to programs. We use the *greatest* fixpoint to define the semantics of specifications given by recursive equations, as does UTP (which uses the term 'weakest fixed point' and the notation μ, but with the reverse order). For example, the iteration $\textbf{\textit{while}}\ b\ \textbf{\textit{do}}\ P$ is just $\nu(X \mapsto P\ ;\ X \lhd b \rhd \mathbb{1})$, see also the examples in Section 2.2. The reason is that using the least fixpoint would model non-termination as $\mu(X \mapsto X) = \perp\!\!\!\perp$ which is a subset of, and hence implements, every specification. In terms of [SS92] the least fixpoint is appropriate for angelic non-determinism while the greatest fixpoint yields demonic non-determinism.

We conclude our compendium of programming constructs by two useful results. The first shows that any function composed of these operations is isotone, which is important for the existence of fixpoints needed to solve recursive equations. The second shows that any relation composed of these operations satisfies the healthiness conditions $\mathscr{H}_L$ and $\mathscr{H}_R$, the former with a small proviso concerning $\mathscr{T}_{\preccurlyeq}$. This corresponds to the left and right unit laws of UTP, called H1a and H3 in Section 1.2, and establishes $\mathbb{1}$ as a neutral element with respect to sequential composition, which is useful to terminate iterations and to obtain a one-sided conditional.

Further claims of this type will be stated and proved in the following sections. We always give the applicable constructs and restrictions; see the table in Section 3.5 for a summary.

Theorem 61. *Let $f(P)$ be a term composed of P, arbitrary constants, sequential and parallel composition, non-deterministic choice, conjunction, conditional, least and greatest fixpoint. Then $P \mapsto f(P)$ is isotone.*

Proof. A more general claim is the following: Let S be any subterm of $f(P)$ and Q any (relation) variable, then S is isotone in Q. The latter means that $Q \mapsto T$ is isotone where T is obtained by substituting in S any constant relations for the free variables of S except Q. Note that S is isotone in all variables that do not occur freely in S.

The proof is by induction on the term structure of $f(P)$, and distinguishes the following cases:

- If $S = P$, then P is the only free variable of S and $P \mapsto P$ is clearly isotone.
- If S is constant, it has no free variables and the claim holds trivially.
- If $S = S_1 \,;\, S_2$ or $S = S_1 \| S_2$, the claim follows by isotony of ; and $\|$ from the induction hypothesis.
- If $S = \bigcup_{i \in I} S_i$, the claim follows using $(\forall i \in I : A_i \subseteq B_i) \Rightarrow (\bigcup_{i \in I} A_i) \subseteq (\bigcup_{i \in I} B_i)$. A similar remark applies to $\bigcap$.
- If $S = (S_1 \blacktriangleleft b \blacktriangleright S_2)$, the claim follows since the conditional is composed of constants, $\cap$ and $\cup$, which are isotone.
- If $S = \xi(X \mapsto S_1)$ for $\xi \in \{\mu, \nu\}$, the claim follows since these fixpoint operators are isotone [DP02, 8.28 and duality]. □

Theorem 62. *Let $X \in \{E, L, R\}$. Let P be a term composed of constants satisfying $\mathscr{H}_X$ (in particular skip, (un)declaration and assignment), sequential and parallel composition, non-deterministic choice, conjunction, conditional, least and greatest fixpoint. If $X \in \{E, L\}$, let $\mathscr{T}_{\preccurlyeq}(b)$ for all conditions b that occur as $\_ \blacktriangleleft b \blacktriangleright \_$ in P. Then $\mathscr{H}_X(P)$.*

Proof. Let $X \in \{L, R\}$. A more general claim is the following: Let S be any subterm of P, then $\mathscr{H}_X(T)$ where T is obtained by substituting in S any constant relations satisfying $\mathscr{H}_X$ for the free variables of S. The proof is by induction on the term structure of P. The cases skip, (un)declaration, assignment, sequential and parallel composition, non-deterministic choice, conjunction and conditional are proved in Lemmas 53, 55, 57, 58 and 59.

We justify the remaining fixpoint cases with more detail. Let $S = \xi(Q \mapsto S_1)$ for $\xi \in \{\mu, \nu\}$ and let $T = \xi(Q \mapsto T_1)$ be obtained by substituting in S any constant relations satisfying $\mathscr{H}_X$ for the free variables of S. Observe that Q is free in T_1 iff it is free in S_1, and no other variables are free in T_1. First, $Q \mapsto T_1$ is $\mathscr{H}_X$-preserving since $\mathscr{H}_X(Q)$ implies $\mathscr{H}_X(T_1)$ by the induction hypothesis (the free variables of S_1 are either Q which satisfies $\mathscr{H}_X$ or have been substituted by constants satisfying $\mathscr{H}_X$). Second, $Q \mapsto T_1$ is isotone by Theorem 61. Corollary 54 thus implies $\mathscr{H}_X(\xi(Q \mapsto T_1))$, which is equivalent to $\mathscr{H}_X(T)$. □

3.3 Finite branching

From the computational perspective, it is necessary to regard the greatest fixpoint not as the supremum of all fixpoints but as the infimum of a certain chain. Not all properties, however, are preserved by infima of chains. It occasionally helps to restrict the attention to infima of chains of relations that model a finite degree of non-determinism. Such relations represent what are sometimes called *boundedly non-deterministic* programs, see [Dij76, AP86, Hes92] as well as the surveys [SS92, WM97a]. In graph theory, taking states as nodes and transitions as edges, one speaks of a finite outdegree. The pure condition of finite branching is not appropriate, as witnessed by the relation $\mathbb{T}$ that generally provides an infinite number of successor states. We therefore give a new, relaxed condition.

The stage is prepared by an elaborate order-theoretic discussion in Section 3.3.1. It is mostly of general nature, assuming only partially ordered sets, possibly satisfying further properties. The results, dealing with minima of partial orders, are applied in Section 3.3.2 to discuss finite branching. The latter is necessary to show the continuity of functions and the totality of relations in the following two sections.

3.3.1 Minimal elements

Let P be a set partially ordered by $\preceq$ (later on the instance will be $D_{1..n}$ ordered by $\preccurlyeq$). We denote the greatest lower bound or meet of $x, y \in P$ by $x \curlywedge y$ and their least upper bound or join by $x \curlyvee y$ whenever they exist. The minimal elements of a set $A \subseteq P$ are $\min A =_{\text{def}} \{x \mid x \in A \wedge \forall y : (y \in A \wedge y \preceq x) \Rightarrow y = x\}$. Our results mainly concern the minima of unions and intersections of sets. Additional assumptions about the partially ordered sets are introduced stepwise, as required.

Lemma 63. $\min(A \cup B) \subseteq \min A \cup \min B$.

Proof. Let $x \in \min(A \cup B)$, then $x \in A \cup B$ and $\forall y : (y \in A \cup B \wedge y \preceq x) \Rightarrow y = x$. If $x \in A$, we show $x \in \min A$. To this end, let $y \in A$ such that $y \preceq x$, then $y \in A \cup B$, hence $y = x$ since x is minimal in $A \cup B$. Symmetrically, $x \in B$ implies $x \in \min B$. □

The ordered set P is *well-founded* iff $\min A \neq \emptyset$ for all non-empty $A \subseteq P$. The *upper closure* of $A \subseteq P$ with respect to $\preceq$ is $\uparrow A =_{\text{def}} \{y \mid y \in P \wedge \exists x \in A : x \preceq y\}$. Observe that $A \subseteq \uparrow A$. The *lower closure* $\downarrow A$ is the upper closure with respect to $\preceq^{\smile}$. For individual elements, we abbreviate $\uparrow x =_{\text{def}} \uparrow\{x\}$ and $\downarrow x =_{\text{def}} \downarrow\{x\}$. The operations min and $\uparrow$ are related by the following ancillary lemma.

Lemma 64. *Let $A \subseteq P$, then $\min \uparrow A = \min A$. If P is well-founded, then $\uparrow \min A = \uparrow A$. In particular, the following claims hold.*

1. $\min \uparrow A \subseteq A$.
2. *If $A = \min A$, then $A = \min \uparrow A$.*
3. *If P is well-founded, then $A \subseteq \uparrow \min A$.*
4. *If P is well-founded and $A = \uparrow A$, then $A = \uparrow \min A$.*

Proof. We first show $\min \uparrow A = \min A$. For the part ($\subseteq$), let $x \in \min \uparrow A \subseteq \uparrow A$, hence there is some $z \in A \subseteq \uparrow A$ such that $z \preceq x$. Since x is minimal in $\uparrow A$, we have $z = x$, thus $x \in A$. Indeed, for any $y \in A \subseteq \uparrow A$ such that $y \preceq x$, we thus obtain $y = x$. Therefore $x \in \min A$. For the part ($\supseteq$), let $x \in \min A \subseteq A \subseteq \uparrow A$. Let $y \in \uparrow A$ such that $y \preceq x$, then there is some $z \in A$ satisfying $z \preceq y \preceq x$. Since x is minimal in A, we have $z = x = y$. Hence $x \in \min \uparrow A$.

We next show $\uparrow \min A = \uparrow A$ assuming that P is well-founded. The part ($\subseteq$) immediately follows by isotony of the operation $\uparrow$ from $\min A \subseteq A$. For the part ($\supseteq$), let $x \in \uparrow A$ and define $B =_{\text{def}} A \cap \downarrow x$. Since there is some $z \in A$ such that $z \preceq x$ we have $B \neq \emptyset$ and therefore $\min B \neq \emptyset$ by well-foundedness. Let $m \in \min B$, then $m \in B$, hence $m \in A$. To see that $m \in \min A$, let $y \in A$ such that $y \preceq m$, thus $y \preceq m \preceq x$, hence $y \in B$; since m is minimal in B, we have $y = m$. Together with $m \preceq x$ we obtain $x \in \uparrow \min A$.

The particular claims are immediate consequences of the main claims. □

Remark. The operations of taking the upper bounds of a set of elements and taking the lower bounds form (the order-reversing kind of) a Galois connection, see [SS89, Satz 3.3.3] for example. Parts 1 and 3 of the previous lemma show that a similar correspondence is available for taking the minima and the upper closure in well-founded orders. This is almost a Galois connection: Since the operation $\uparrow$ is isotone, the only property missing is the isotony of min which indeed does not hold. (End of remark)

The ordered set P is a *meet-semilattice* iff $x \curlywedge y$ exists for all $x, y \in P$. A set $A \subseteq P$ is an *upper set* iff $(x \in A \wedge x \preceq y) \Rightarrow y \in A$ for all $x, y \in P$ or, equivalently, $A = \uparrow A$. The introduced concepts are connected to computations by applying them to the image sets of a state with $\preccurlyeq$ as the partial order. We have already observed that $D_{1..n}$ is well-founded and a meet-semilattice, and Lemma 67 will establish these images as upper sets provided the computation satisfies $\mathscr{H}_R$.

The following lemma characterises the minimal elements of intersections. Our primary interest is the consequence that $\min(A \cap B)$ is finite provided $\min A$ and $\min B$ are finite.

Lemma 65. *Let P be a well-founded meet-semilattice. Let $A \subseteq P$ and $B \subseteq P$ be upper sets. Then* $\min(A \cap B) \subseteq \{x \curlyvee y \mid x \in \min A \wedge y \in \min B \wedge x \curlyvee y \text{ exists}\}$.

Proof. Let $x \in \min(A \cap B)$, then $x \in A \cap B$ and $\forall y : (y \in A \cap B \wedge y \preceq x) \Rightarrow y = x$. We show that $x = a \curlyvee b$ for some $a \in \min A$ and $b \in \min B$.

Since $x \in A \subseteq {\uparrow}\min A$ by Lemma 64.3, there is an $a \in \min A$ such that $a \preceq x$. We analogously obtain $b \in \min B$ such that $b \preceq x$, thus x is an upper bound of a and b.

Let y be an upper bound of a and b, that is, $a \preceq y$ and $b \preceq y$. By the meet property, $a \preceq x \curlywedge y$ and $b \preceq x \curlywedge y$. Thus $x \curlywedge y \in {\uparrow}\min A \cap {\uparrow}\min B = A \cap B$ by Lemma 64.4. Since x is minimal in $A \cap B$, we obtain $x \curlywedge y = x$, hence $x \preceq y$. It follows that x is the least upper bound of a and b. □

We extend this lemma to arbitrary intersections under the following assumption that is stronger than well-foundedness. The ordered set P has *finite height*, more precisely height h, iff $|C| = h + 1$ for some chain $C \subseteq P$ and $|C| \leq h + 1$ for every chain $C \subseteq P$. The height of $x \in P$ is the height of $\{y \mid y \in P \wedge y \preceq x\}$ provided the latter exists. It does if P has height h, and this induces the order-preserving *height function* $\hbar : P \to 0..h$ which maps every element to its height. Observe that the elements with the same height form an antichain of P, that is, a subset of pairwise incomparable elements. Again the applicability to computations is ensured since $D_{1..n}$ ordered by $\preccurlyeq$ has finite height n.

To see that finite height implies well-foundedness, let $\emptyset \neq A \subseteq P$ and we show $\min A \neq \emptyset$. Let $C \subseteq A$ be a maximal chain and $m \in C$ its least element which exists since C is finite but not empty. Let $x \in A$ such that $x \preceq m$, then $x = m$ by maximality of C, hence $m \in \min A$.

Lemma 66. *Let P be a meet-semilattice having finite height. Let S be a set of upper sets in P such that $|\min A| \in \mathbb{N}$ for each $A \in S$. Then $|\min \bigcap S| \in \mathbb{N}$.*

Proof. The following pseudo-algorithmic procedure constructs $\bigcap S$ in a finite number of steps. By convention, if $S = \emptyset$ we have $\bigcap S = P$.

```
1.  input P, S
2.  Q ← P
3.  R ← S \ {P}
4.  while R ≠ ∅ do
5.      let B ∈ R
6.      Q ← Q ∩ B
7.      R ← {A | A ∈ R ∧ Q ⊈ A}
8.  end
9.  output Q
```

We first establish by a sequence of five invariants that, if the procedure terminates, its output is $Q = \bigcap S$ and $|\min Q| \in \mathbb{N}$ which proves the lemma.

1. $R \subseteq S$ is clear after the initialisation in line 3 and maintained by the assignment in line 7.

2. $R = \{A \mid A \in S \wedge Q \not\subseteq A\}$ is clear after the initialisation in line 3. We show that it is preserved by the two assignments in lines 6 and 7. The part ($\subseteq$) is satisfied after line 7 by invariant 1. For the part ($\supseteq$), assuming $R \supseteq \{A \mid A \in S \wedge Q \not\subseteq A\}$ we have to show $\{A \mid A \in R \wedge Q \cap B \not\subseteq A\} \supseteq \{A \mid A \in S \wedge Q \cap B \not\subseteq A\}$. This follows since $Q \cap B \not\subseteq A \Rightarrow Q \not\subseteq A$ for any A.

3. $\bigcap S = Q \cap \bigcap R$ is clear after the initialisation in line 3. To conclude that it is preserved by the two assignments in lines 6 and 7, assuming $B \in R$ and $\bigcap S = Q \cap \bigcap R$ we have to show $\bigcap S = Q \cap B \cap \bigcap\{A \mid A \in R \wedge Q \cap B \not\subseteq A\}$. For the part ($\subseteq$), we have $\bigcap S = Q \cap \bigcap R = Q \cap B \cap \bigcap R \subseteq Q \cap B \cap \bigcap\{A \mid A \in R \wedge Q \cap B \not\subseteq A\}$ by the assumption and since $\bigcap$ is antitone. For the part ($\supseteq$), let $X \in S$, then either $Q \cap B \subseteq X$ or $Q \cap B \not\subseteq X$, in which case $Q \not\subseteq X$, hence $X \in R$ by invariant 2, thus $\bigcap\{A \mid A \in R \wedge Q \cap B \not\subseteq A\} \subseteq X$. In both cases, $Q \cap B \cap \bigcap\{A \mid A \in R \wedge Q \cap B \not\subseteq A\} \subseteq X$.

4. Q is an upper set, which is clear after the initialisation in line 2 and maintained by the assignment in line 6, since $B \in R \subseteq S$ by invariant 1, hence B is an upper set by the assumption, and the intersection of upper sets is an upper set.

5. $|\min Q| \in \mathbb{N}$ is satisfied after the initialisation in line 2 since P is a meet-semilattice having finite height. The special case $P = \emptyset$ is clear since $|\min P| = |\emptyset| = 0 \in \mathbb{N}$; in the other case we show that P has a least element. Let $C \subseteq P$ be a maximal chain and m its least element, which exists since C is non-empty and finite. Let $x \in P$, then $m \curlywedge x \preceq m$, hence $m \curlywedge x = m$ by maximality of C, and therefore $m \preceq x$. It follows that $|\min P| = |\{m\}| = 1 \in \mathbb{N}$. Invariant 5 is maintained by the assignment in line 6. To see this, observe that Q is an upper set by invariant 4 and B is an upper set by the assumption and invariant 1. Hence $|\min(Q \cap B)| \leq |\min Q| \cdot |\min B| \in \mathbb{N}$ by Lemma 65 and the assumption.

If the procedure terminates in line 9, we have $R = \emptyset$, hence $Q = \bigcap S$ by invariant 3 and therefore $|\min \bigcap S| = |\min Q| \in \mathbb{N}$ by invariant 5. It remains to show termination, which is clear if $P = \emptyset$. If $P \neq \emptyset$, let $\hbar : P \to 0..h$ be its height function. Let $L =_{\text{def}} 0..h \to \mathbb{N}$ be well-ordered lexicographically by $\leq$ and define

$$\begin{aligned} &v : \mathscr{P}(P) \to L \\ &v(A)(k) =_{\text{def}} |\{x \mid x \in \min A \wedge \hbar(x) = k\}| \end{aligned}$$

The finite sequence $v(A)$ describes the distribution of the minimal elements of A according to their height. We use v as the variant function on the variable Q to establish termination of the procedure. Observe that after the initialisation in line 2, we have

$$v(Q)(k) = v(P)(k) = \begin{cases} 1, \text{ if } k = 0 \\ 0, \text{ otherwise} \end{cases}$$

since $\min P = \{m\}$ as shown for invariant 5 and $\hbar(m) = 0$. We now show that v is decreased by the assignment in line 6, that is, $v(Q \cap B) < v(Q)$. Note that $v(Q \cap B)$ is a function because $|\min(Q \cap B)| \in \mathbb{N}$ by invariant 5. Since $B \in R$, we have $Q \nsubseteq B$ before the assignment by invariant 2. Defining $T =_{\text{def}} Q \setminus B$ we therefore have $T \neq \emptyset$. Let furthermore $l =_{\text{def}} \min\{\hbar(x) \mid x \in T\}$ where min refers to the natural number order. The decrease of v is established by showing $v(Q \cap B)(k) = v(Q)(k)$ for all $0 \leq k < l$ and $v(Q \cap B)(l) < v(Q)(l)$.

For $0 \leq k < l$ this claim is entailed by

$$\{x \mid x \in \min(Q \cap B) \wedge \hbar(x) = k\} = \{x \mid x \in \min Q \wedge \hbar(x) = k\}.$$

For the part ($\subseteq$), let $x \in \min(Q \cap B)$ such that $\hbar(x) = k$, hence $x \in Q \cap B \subseteq Q$. Let $y \in Q$ such that $y \preceq x$, then $\hbar(y) \leq \hbar(x) = k < l = \min\{\hbar(x) \mid x \in T\}$, hence $y \notin T$, thus $y \in B$ from which $y = x$ follows since x is minimal in $Q \cap B$. We therefore have $x \in \min Q$. For the part ($\supseteq$), let $x \in \min Q \subseteq Q$ such that $\hbar(x) = k < l = \min\{\hbar(x) \mid x \in T\}$, hence $x \notin T$, thus $x \in B$ and therefore $x \in Q \cap B$. Let $y \in Q \cap B \subseteq Q$ such that $y \preceq x$, then $y = x$ since x is minimal in Q. Therefore $x \in \min(Q \cap B)$.

The remaining claim is entailed by

$$\{x \mid x \in \min(Q \cap B) \wedge \hbar(x) = l\} \subset \{x \mid x \in \min Q \wedge \hbar(x) = l\}.$$

For the part ($\subseteq$), let $x \in \min(Q \cap B)$ such that $\hbar(x) = l$, hence $x \in Q \cap B \subseteq Q$. Let $y \in Q$ such that $y \preceq x$, then $\hbar(y) \leq \hbar(x) = l$. If $\hbar(y) < \hbar(x)$, the argument proceeds as before to show $y = x$, however if $\hbar(y) = \hbar(x)$, we immediately obtain $y = x$ since all elements with the same height form an antichain of P, but x and y are comparable. Therefore $x \in \min Q$. For the part ($\neq$), let $x \in T$ such that $\hbar(x) = l$, hence $x \in Q$ and $x \notin B$, thus $x \notin Q \cap B$ from which $x \notin \{x \mid x \in \min(Q \cap B) \wedge \hbar(x) = l\}$ follows. Let $y \in Q$ such that $y \preceq x$, then $y \notin B$ since B is an upper set, hence $y \in T$, thus $\hbar(y) \leq \hbar(x) = l \leq \hbar(y)$ from which $y = x$ follows again by the antichain property. Therefore $x \in \{x \mid x \in \min Q \wedge \hbar(x) = l\}$. □

Remark. Restricting to finite height is essential in the previous lemma, even if S is assumed to be a chain. This lemma is applied to $D_{1..n}$ ordered by $\preccurlyeq$ for proving Lemmas 69.3 and 71. Finite height is guaranteed since there are only a finite number of variables and the ranges D_i are flat orders. The latter suffices for data structures with strict constructors, but excludes infinite data structures which are modelled by non-flat orders. However, the problem is not caused by the infinite data structures themselves, but by having non-determinism at the same time. For example, let us prove the following deterministic version of Lemma 66 that does not require finite height:

> Let P be a well-founded meet-semilattice. Let S be a set of upper sets in P such that $|\min A| \leq 1$ for each $A \in S$. Then $|\min \bigcap S| \leq 1$.

If $|\min A| = 0$ for some $A \in S$, then $|\min \bigcap S| \leq |\bigcap S| \leq |A| = 0$ by well-foundedness. Otherwise assume $x, y \in \min \bigcap S$ and we show $x = y$. Defining $z =_{\text{def}} x \curlywedge y$, we first show $z \in A$ for each $A \in S$. By Lemma 64.3, $x \in \min \bigcap S \subseteq \bigcap S \subseteq A \subseteq \uparrow \min A$, hence there is an $a \in \min A$ such that $a \preceq x$. The same way we obtain an $m \in \min A$ such that $m \preceq y$. But $a = m$ since $|\min A| = 1$, hence $m \preceq z$, thus $z \in \uparrow \min A = A$ by Lemma 64.4. It follows that $z \in \bigcap S$, and therefore $x = z = y$ since x, y are minimal elements and $z \preceq x$ and $z \preceq y$.

A combined treatment of infinite data structures and non-determinism requiring general powerdomains [Sch86, Mai87] is not carried out in the present work. (End of remark)

We close by the announced lemma guaranteeing the applicability of this section's results to computations. Recall that $Q(\vec{x})$ denotes the image of the state $\vec{x}$ under the relation Q.

Lemma 67. *Let Q be a relation. Then $\mathscr{H}_R(Q)$ iff $Q(\vec{x})$ is an upper set for all $\vec{x}$.*

Proof. For the part ($\Rightarrow$), let $\vec{y} \in Q(\vec{x})$ and $\vec{z}$ such that $\vec{y} \preccurlyeq \vec{z}$, then $(\vec{x}, \vec{y}) \in Q$ and $(\vec{y}, \vec{z}) \in \mathbb{1}$, hence $(\vec{x}, \vec{z}) \in Q\mathbb{1} = Q$ since $\mathscr{H}_R(Q)$, thus $\vec{z} \in Q(\vec{x})$. For the part ($\Leftarrow$), it suffices to show $Q\mathbb{1} \subseteq Q$. To this end, let $(\vec{x}, \vec{z}) \in Q \mathbin{;} \mathbb{1}$, then $(\vec{x}, \vec{y}) \in Q$ and $(\vec{y}, \vec{z}) \in \mathbb{1}$ for some $\vec{y}$, hence $\vec{y} \in Q(\vec{x})$ and $\vec{y} \preccurlyeq \vec{z}$, thus $\vec{z} \in Q(\vec{x})$ since $Q(\vec{x})$ is an upper set, and therefore $(\vec{x}, \vec{z}) \in Q$. □

Remark. The previous lemma provides the connection between the relation-algebraic viewpoint of $\mathscr{H}_R$ and the pointwise upper sets. Indeed, one can represent and calculate with minima as relations. Similarly to a construction described in [SS89, page 43], we obtain from a relation Q its row-wise minima with respect to an irreflexive partial order C by $Q \cap \overline{QC}$. For example, Lemma 64.4 can then be stated as follows:

> Let E be a partial order and its irreflexive counterpart $C = E \cap \overline{\mathbb{I}}$ well-founded. Let $Q = QE$. Then $Q = (Q \cap \overline{QC})E$.

Its proof, however, remains essentially pointwise. The same can be said about a relational proof of the distribution Lemma 72. A taste of calculating with the relational form of minima is given in Section 3.6.4. (End of remark)

3.3.2 Boundedness

We are ready to state the condition for boundedly non-deterministic computations. Finite branching, as given by $\mathscr{H}_F$ in the following definition, cannot be used since we need $\mathbb{T}$ to represent non-termination. This is due to the demonic interpretation of non-deterministic choice. The condition that each state $\vec{x}$ has only a finite number of successor states can be relaxed by allowing additionally the case that every state in $D_{1..n}$ is a successor of $\vec{x}$ [Hes92]. This solves the problem with $\mathbb{T}$, that satisfies the relaxed condition, but is not fine enough for our purposes. We further need to distinguish the individual variables, which is done by the condition $\mathscr{H}_B$ using the pointwise minima with respect to $\preccurlyeq$.

Definition 68.

$$\begin{array}{lll} \mathscr{H}_B(P) & \Leftrightarrow_{\text{def}} & \forall \vec{x} : |\min P(\vec{x})| \in \mathbb{N}, \\ \mathscr{H}_F(P) & \Leftrightarrow_{\text{def}} & \forall \vec{x} : |P(\vec{x})| \in \mathbb{N}. \end{array}$$

Clearly $\mathscr{H}_F(P) \Rightarrow \mathscr{H}_B(P)$ since $\min A \subseteq A$. The intention of using min is the following: $\mathscr{H}_B$ will be applied to relations that satisfy $\mathscr{H}_R$. By Lemmas 67 and 64.4, the image sets of such relations are in a one-to-one correspondence with their minimal elements. Indeed, it is the minimal elements that actually represent the successor states, and their upper closure is formed to satisfy $\mathscr{H}_R$ and to avoid unboundedness. Thus $\mathscr{H}_B$ accounts for the proper successor states, excluding those that have been added for technical reasons. We now show that many relations from our compendium satisfy $\mathscr{H}_B$.

Lemma 69. *The constructs given in Definition 60 satisfy/preserve $\mathscr{H}_B$ as follows.*

1. $\mathscr{H}_B(P) \wedge \mathscr{H}_B(Q) \Rightarrow \mathscr{H}_B(P \cup Q)$.
2. $\mathscr{H}_R(P) \wedge \mathscr{H}_R(Q) \wedge \mathscr{H}_B(P) \wedge \mathscr{H}_B(Q) \Rightarrow \mathscr{H}_B(P \cap Q)$.
3. $(\forall P \in S : \mathscr{H}_R(P) \wedge \mathscr{H}_B(P)) \Rightarrow \mathscr{H}_B(\bigcap S)$.
4. $\mathscr{H}_F(P) \wedge \mathscr{H}_B(Q) \Rightarrow \mathscr{H}_B(P \,;\, Q)$.
5. $\mathscr{H}_R(P) \wedge \mathscr{H}_L(Q) \wedge \mathscr{H}_B(P) \wedge \mathscr{H}_B(Q) \Rightarrow \mathscr{H}_B(P \,;\, Q)$.
6. $\mathscr{H}_B(\mathbb{1})$.
7. $\mathscr{T}_{\preccurlyeq}(\vec{e}) \Rightarrow \mathscr{H}_B(\vec{x} \leftarrow \vec{e})$.
8. $\mathscr{H}_B(\textbf{\textit{end}}\ \vec{x}_K) \wedge \mathscr{H}_B(\textbf{\textit{var}}\ \vec{x}_K)$.
9. $\mathscr{H}_B(P) \wedge \mathscr{H}_B(Q) \Rightarrow \mathscr{H}_B(P \blacktriangleleft b \blacktriangleright Q)$.
10. $\mathscr{H}_E(P) \wedge \mathscr{H}_E(Q) \wedge \mathscr{H}_B(P) \wedge \mathscr{H}_B(Q) \Rightarrow \mathscr{H}_B(P \| Q)$.
11. *Let f be isotone and $(\mathscr{H}_E \wedge \mathscr{H}_B)$-preserving, then $\mathscr{H}_B(\nu f)$.*

Proof.

1. $|\min(P \cup Q)(\vec{x})| = |\min(P(\vec{x}) \cup Q(\vec{x}))| \leq |\min P(\vec{x}) \cup \min Q(\vec{x})| \leq |\min P(\vec{x})| + |\min Q(\vec{x})| \in \mathbb{N}$ by Lemma 63.
2. $|\min(P \cap Q)(\vec{x})| = |\min(P(\vec{x}) \cap Q(\vec{x}))| \leq |\min P(\vec{x})| \cdot |\min Q(\vec{x})| \in \mathbb{N}$ by Lemma 65 since $D_{1..n}$ ordered by $\preccurlyeq$ is a well-founded meet-semilattice, and $P(\vec{x})$ and $Q(\vec{x})$ are upper sets by Lemma 67.
3. $|\min(\bigcap S)(\vec{x})| = |\min \bigcap_{P \in S} P(\vec{x})| \in \mathbb{N}$ by Lemma 66 since $D_{1..n}$ is a meet-semilattice ordered by $\preccurlyeq$ with the height function $\hbar(\vec{x}) = |\{i \mid i \in 1..n \wedge x_i \neq \infty\}|$ and P is an upper set by Lemma 67 for each $P \in S$.
4. $|\min(P \,;\, Q)(\vec{x})| = |\min \bigcup_{\vec{y} \in P(\vec{x})} Q(\vec{y})| \leq |\bigcup_{\vec{y} \in P(\vec{x})} \min Q(\vec{y})| \leq \sum_{\vec{y} \in P(\vec{x})} |\min Q(\vec{y})| \in \mathbb{N}$ by Lemma 63 generalised to finite unions since $\mathscr{H}_F(P)$, and by $\mathscr{H}_B(Q)$ at last.
5. $P(\vec{x})$ is an upper set for each $\vec{x}$ by Lemma 67. Therefore $P(\vec{x}) = {\uparrow} \min P(\vec{x})$ by Lemma 64.4. Hence,

$$\begin{aligned}(P \,;\, Q)(\vec{x}) &= \textstyle\bigcup_{\vec{z} \in P(\vec{x})} Q(\vec{z}) = \bigcup_{\vec{z} \in {\uparrow} \min P(\vec{x})} Q(\vec{z}) = \bigcup_{\vec{y} \in \min P(\vec{x})} \bigcup_{\vec{z} \in {\uparrow}\vec{y}} Q(\vec{z}) \\ &= \textstyle\bigcup_{\vec{y} \in \min P(\vec{x})} (\preccurlyeq \,;\, Q)(\vec{y}) = \bigcup_{\vec{y} \in \min P(\vec{x})} Q(\vec{y})\end{aligned}$$

   since $\mathscr{H}_L(Q)$. Since $\min P(\vec{x})$ is finite, we conclude as in part 4.
6. $|\min \mathbb{1}(\vec{x})| = |\min {\preccurlyeq}(\vec{x})| = |\{\vec{x}\}| = 1 \in \mathbb{N}$ by Lemma 64.2 since $\min\{\vec{x}\} = \{\vec{x}\}$.
7. $\vec{x} {:=} \vec{e}$ is univalent, hence $\mathscr{H}_F(\vec{x} {:=} \vec{e})$, thus $\mathscr{H}_B(\vec{x} {:=} \vec{e} \,;\, \mathbb{1})$ by parts 6 and 4, and therefore $\mathscr{H}_B(\mathbb{1} \,;\, \vec{x} {:=} \vec{e} \,;\, \mathbb{1})$ by Lemmas 53.1 and 57.3 and parts 6 and 5.

8. We have $\mathscr{H}_F(\exists \vec{x}'_K : \mathbb{I})$ by univalence, hence $\mathscr{H}_B((\exists \vec{x}'_K : \mathbb{I}) ; \mathbb{1})$ by parts 6 and 4, thus $\mathscr{H}_B(\textbf{\textit{end}}\ \vec{x}_K)$ by Lemma 58.2. As concerns $\textbf{\textit{var}}\ \vec{x}_K$, observe that

$$\begin{aligned}(\textbf{\textit{var}}\ \vec{x}_K)(\vec{x}_I) &= \{\vec{x}'_{I\cup K} \mid (\vec{x}_I, \vec{x}'_{I\cup K}) \in \textbf{\textit{var}}\ \vec{x}_K\} = \{\vec{x}'_{I\cup K} \mid (\vec{x}_I, \vec{x}'_{I\cup K}) \in (\exists \vec{x}_K : \mathbb{1})\} \\ &= \{\vec{x}'_{I\cup K} \mid \exists \vec{x}_K : (\vec{x}_{I\cup K}, \vec{x}'_{I\cup K}) \in \mathbb{1}\} = \{\vec{x}'_{I\cup K} \mid \exists \vec{x}_K : \vec{x}_{I\cup K} \preccurlyeq \vec{x}'_{I\cup K}\} \\ &= \{\vec{x}'_{I\cup K} \mid \vec{x}_I \preccurlyeq \vec{x}'_I\}.\end{aligned}$$

Taking the minima pointwise, we obtain as the unique minimum the $\vec{y}_{I\cup K}$ with $\vec{y}_I{=}\vec{x}_I$ and $\vec{y}_K{=}\infty$. Hence $|\min(\textbf{\textit{var}}\ \vec{x}_K)(\vec{x}_I)| = |\{\vec{y}_{I\cup K} \mid \vec{y}_I{=}\vec{x}_I \wedge \vec{y}_K{=}\infty\}| = 1 \in \mathbb{N}$.

9. We first show $V = V\mathbb{T} \wedge \mathscr{H}_B(P) \Rightarrow \mathscr{H}_B(V \cap P)$, that is, that restriction by a vector preserves $\mathscr{H}_B$. To this end, let $\vec{x} \in D_{1..m}$, then either of the following two cases applies:

$$\begin{aligned}V(\vec{x}) = \emptyset &\Rightarrow |\min(V \cap P)(\vec{x})| = |\min(V(\vec{x}) \cap P(\vec{x}))| = |\min \emptyset| = 0 \in \mathbb{N}, \\ V(\vec{x}) = D_{1..n} &\Rightarrow |\min(V \cap P)(\vec{x})| = |\min(V(\vec{x}) \cap P(\vec{x}))| = |\min P(\vec{x})| \in \mathbb{N}.\end{aligned}$$

In particular it follows that every vector V satisfies $\mathscr{H}_B$ by choosing $P = \mathbb{T}$ and part 3. Therefore $\mathscr{H}_B(b{=}\infty)$, $\mathscr{H}_B(b{=}\bot \cap \vec{x}'{=}\bot)$ since $\mathscr{H}_B(\vec{x}'{=}\bot)$ by univalence, $\mathscr{H}_B(b{=}\mathit{true} \cap P)$ and $\mathscr{H}_B(b{=}\mathit{false} \cap Q)$ by the assumption, hence $\mathscr{H}_B(P \blacktriangleleft b \blacktriangleright Q)$ by part 1.

10. Observe that $P\|Q = \mathbb{1}P\mathbb{1}\|\mathbb{1}Q\mathbb{1} = \textbf{\textit{end}}\ \vec{x}_K\ ;\ P\ ;\ \textbf{\textit{var}}\ \vec{x}_L \cap \textbf{\textit{end}}\ \vec{x}_I\ ;\ Q\ ;\ \textbf{\textit{var}}\ \vec{x}_J$ by Lemma 59.1. The claim follows by Lemmas 58 and 53.2, and parts 8, 5 and 2.

11. Relations satisfying $\mathscr{H}_E$ and $\mathscr{H}_B$ are closed under infima by Lemma 53.4 and part 3. The claim therefore follows by Corollary 89. □

Theorem 70. *Let P be a term composed of constants satisfying $\mathscr{H}_E$ and $\mathscr{H}_B$ (in particular skip, (un)declaration and assignment), sequential and parallel composition, finite non-deterministic choice, conjunction, conditional and greatest fixpoint. Let $\mathscr{T}_{\preccurlyeq}(b)$ for all conditions b that occur as $\_ \blacktriangleleft b \blacktriangleright \_$ in P. Let $\mathscr{T}_{\preccurlyeq}(\vec{e})$ for all expressions $\vec{e}$ that occur as $\_{\leftarrow}\vec{e}$ in P. Then $\mathscr{H}_B(P)$.*

Proof. A more general claim is the following: Let S be any subterm of P, then $\mathscr{H}_E(T)$ and $\mathscr{H}_B(T)$ for any T obtained by substituting in S any constant relations satisfying $\mathscr{H}_E$ and $\mathscr{H}_B$ for the free variables of S. Any T so obtained satisfies $\mathscr{H}_E(T)$ by Theorem 62 which needs the assumption $\mathscr{T}_{\preccurlyeq}(b)$. The proof for $\mathscr{H}_B$ is also by induction and analogous to the proof of Theorem 62, this time using Lemma 69 for the individual cases, in particular Lemma 69.11 for the greatest fixpoint. The assumption $\mathscr{T}_{\preccurlyeq}(\vec{e})$ is necessary for Lemma 69.7. □

3.3.3 Continuity

We call a function f *continuous* iff it distributes over infima of non-empty chains of relations, formally $f(\bigcap C) = \bigcap_{P\in C} f(P)$ for each chain $C \neq \emptyset$. The importance of continuity comes from the permission to represent the greatest fixpoint νf by the constructive $\bigcap_{n\in\mathbb{N}} f^n(\mathbb{T})$. In its dual form Kleene uses this method to prove [Kle52, Theorem XXVI]; a modern account is given in [DP02, 8.15]. This enables the approximation of νf by repeatedly unfolding f which simulates recursive calls of the modelled computation.

That infinite branching or unbounded non-determinism breaks continuity is shown, for example, in [Dij76, Chapter 9] and [BGW79, Section 5.7]. We use the finite branching property $\mathscr{H}_B$ to establish the continuity of functions composed in our framework. Our development starts with two distribution lemmas.

Lemma 71. *Let C be a non-empty chain such that $\mathscr{H}_R(P)$ and $\mathscr{H}_B(P)$ for all $P \in C$. Then $(\bigcap_{P\in C} PQ) = (\bigcap C)Q$.*

Proof. The part ($\supseteq$) is clear. For the part ($\subseteq$), let $(\vec{x},\vec{x}') \in \bigcap_{P\in C} PQ$. Consider the chain $C' =_{\text{def}} \{P(\vec{x}) \mid P \in C\}$. Each $A \in C'$ is an upper set by Lemma 67, and $|\min A| \in \mathbb{N}$. By the proof of Lemma 66, $\bigcap C'$ can be represented as a finite, non-empty intersection of elements of C'. Since C' is a chain, this intersection coincides with one element $R(\vec{x}) = \bigcap C'$ for some $R \in C$. Observing that $(\vec{x},\vec{x}') \in (\bigcap_{P\in C} PQ) \subseteq RQ$, there is $\vec{y} \in R(\vec{x})$ such that $\vec{x}' \in Q(\vec{y})$. It follows that $\vec{y} \in (\bigcap C') = (\bigcap_{P\in C} P(\vec{x})) = (\bigcap_{P\in C} P)(\vec{x}) = (\bigcap C)(\vec{x})$, thus $(\vec{x},\vec{x}') \in (\bigcap C)Q$. □

Lemma 72. *Let C be a non-empty chain such that $\mathscr{H}_L(Q)$ for all $Q \in C$, and let P be such that $\mathscr{H}_R(P)$ and $\mathscr{H}_B(P)$. Then $(\bigcap_{Q\in C} PQ) = P(\bigcap C)$.*

Proof. The part ($\supseteq$) is clear. For the part ($\subseteq$), let $(\vec{x},\vec{x}') \in \bigcap_{Q\in C} PQ$. Observe that $M =_{\text{def}} \min P(\vec{x})$ is finite and $P(\vec{x}) = {\uparrow} M$ by the assumptions and Lemmas 67 and 64.4. Consider the chain $C' =_{\text{def}} \{M \cap Q^\smile(\vec{x}') \mid Q \in C\} \subseteq \mathscr{P}(M)$. Since C' is finite, it has a least element $M \cap R^\smile(\vec{x}')$ for some $R \in C$. Since $(\vec{x},\vec{x}') \in PR$, there is some $\vec{z} \in P(\vec{x}) = {\uparrow} M$ such that $(\vec{z},\vec{x}') \in R$, and therefore some $\vec{y} \in M$ such that $\vec{y} \preccurlyeq \vec{z}$. This implies $(\vec{y},\vec{x}') \in \preccurlyeq \,;\, R = R$ since $\mathscr{H}_L(R)$, thus $\vec{y} \in R^\smile(\vec{x}')$. For each $Q \in C$ it follows that

$$\vec{y} \in M \cap R^\smile(\vec{x}') \subseteq M \cap Q^\smile(\vec{x}') \subseteq P(\vec{x}) \cap Q^\smile(\vec{x}'),$$

that is, $(\vec{x},\vec{y}) \in P$ and $(\vec{y},\vec{x}') \in Q$. Hence $(\vec{y},\vec{x}') \in \bigcap C$ and we obtain $(\vec{x},\vec{x}') \in P(\bigcap C)$. □

Theorem 73. *Let $f(P)$ be a term composed of P, constants satisfying $\mathscr{H}_E$ and $\mathscr{H}_B$ (in particular skip, (un)declaration and assignment), sequential and parallel composition, finite non-deterministic choice, conjunction, conditional and greatest fixpoint. Let $\mathscr{T}_{\preccurlyeq}(b)$ for all conditions b that occur as $\_ \blacktriangleleft b \blacktriangleright \_$ in P. Let $\mathscr{T}_{\preccurlyeq}(\vec{e})$ for all expressions $\vec{e}$ that occur as $\_ \leftarrow \vec{e}$ in P. Then $P \mapsto f(P)$ is continuous, that is, distributes over infima of non-empty chains of relations satisfying $\mathscr{H}_E$ and $\mathscr{H}_B$.*

Proof. A more general claim is the following: Let S be any subterm of $f(P)$ and Q any (relation) variable, then S is continuous in Q. The latter means that $Q \mapsto T$ is continuous where T is obtained by substituting in S any constant relations satisfying $\mathscr{H}_E$ and $\mathscr{H}_B$ for the free variables of S except Q. Note that S is continuous in all variables that do not occur freely in S.

As for Theorem 61, the proof is by induction on the term structure of $f(P)$, and distinguishes the following cases:

- If $S = P$, then P is the only free variable of S and $P \mapsto P$ is clearly continuous.
- If S is constant, it has no free variables and the claim holds trivially.
- If $S = S_1 \,;\, S_2$, we have to show that $Q \mapsto T_1 \,;\, T_2$ is continuous for any $T_1 \,;\, T_2$ obtained by substitution as described above. In the remainder of the proof, let $C \neq \emptyset$ be a chain of elements satisfying $\mathscr{H}_E$ and $\mathscr{H}_B$, and let us write $T(A)$ for $(Q \mapsto T)(A)$. The claim follows by the induction hypothesis, if we can show the third step of

$$\begin{aligned}(T_1 \,;\, T_2)(\textstyle\bigcap C) &= T_1(\textstyle\bigcap C) \,;\, T_2(\textstyle\bigcap C) = (\textstyle\bigcap_{P\in C} T_1(P)) \,;\, (\textstyle\bigcap_{P\in C} T_2(P)) \\ &= (\textstyle\bigcap_{P\in C} T_1(P) \,;\, T_2(P)) = \textstyle\bigcap_{P\in C} (T_1 \,;\, T_2)(P).\end{aligned}$$

  Observe that $\{T_1(P) \mid P \in C\}$ and $\{T_2(P) \mid P \in C\}$ are chains by Theorem 61. Moreover $\mathscr{H}_E(T_1(P))$ and $\mathscr{H}_E(T_2(P))$ by Theorem 62 and $\mathscr{H}_B(T_1(P))$ by Theorem 70 for all $P \in C$. Therefore, by Lemmas 71 and 72,

$$\begin{aligned}&(\textstyle\bigcap_{P\in C} T_1(P)) \,;\, (\textstyle\bigcap_{R\in C} T_2(R)) = \textstyle\bigcap_{P\in C} (T_1(P) \,;\, \textstyle\bigcap_{R\in C} T_2(R)) \\ = {} &(\textstyle\bigcap_{P\in C} \textstyle\bigcap_{R\in C} T_1(P) \,;\, T_2(R)) = \textstyle\bigcap_{P\in C} T_1(P) \,;\, T_2(P).\end{aligned}$$

* If $S = S_1 \cup S_2$, we similarly obtain

$$\begin{aligned}(T_1 \cup T_2)(\bigcap C) &= T_1(\bigcap C) \cup T_2(\bigcap C) = (\bigcap_{P\in C} T_1(P)) \cup (\bigcap_{R\in C} T_2(R)) \\ &= (\bigcap_{P\in C} T_1(P) \cup \bigcap_{R\in C} T_2(R)) = \bigcap_{P\in C}\bigcap_{R\in C} T_1(P) \cup T_2(R) \\ &= (\bigcap_{P\in C} T_1(P) \cup T_2(P)) = \bigcap_{P\in C}(T_1 \cup T_2)(P).\end{aligned}$$

* If $S = \bigcap_{i\in I} S_i$, we similarly have

$$\begin{aligned}(\bigcap_{i\in I} T_i)(\bigcap C) &= (\bigcap_{i\in I} T_i(\bigcap C)) = \bigcap_{i\in I}\bigcap_{P\in C} T_i(P) \\ &= (\bigcap_{P\in C}\bigcap_{i\in I} T_i(P)) = \bigcap_{P\in C}(\bigcap_{i\in I} T_i)(P).\end{aligned}$$

* The claim for the conditional and the parallel composition follows since they are composed of constants, ;, $\cup$ and $\cap$.

* If $S = \nu(X \mapsto S_1)$, we have to show that $Q \mapsto \nu(X \mapsto T_1)$ is continuous. Let us write $T_1(A, B)$ for $((Q, X) \mapsto T_1)(A, B)$. For any $P \in C$, let $g_P(X) =_{\text{def}} T_1(P, X)$, and let $f(X) =_{\text{def}} T_1(\bigcap C, X)$. Then g_P and f are continuous by the induction hypothesis, since $\mathscr{H}_E(P)$, $\mathscr{H}_B(P)$, as well as $\mathscr{H}_E(\bigcap C)$ and $\mathscr{H}_B(\bigcap C)$ by Lemmas 53.4 and 69.3. If we can show $f^n(\top) = \bigcap_{P\in C} g_P^n(\top)$, the claim follows by using Kleene's fixpoint theorem in

$$\begin{aligned}&(Q \mapsto \nu(X \mapsto T_1))(\bigcap C) = \nu(X \mapsto T_1(\bigcap C, X)) = \nu f = \bigcap_{n\in\mathbb{N}} f^n(\top) \\ &= (\bigcap_{n\in\mathbb{N}}\bigcap_{P\in C} g_P^n(\top)) = (\bigcap_{P\in C}\bigcap_{n\in\mathbb{N}} g_P^n(\top)) = \bigcap_{P\in C} \nu(g_P) \\ &= (\bigcap_{P\in C} \nu(X \mapsto T_1(P, X))) = \bigcap_{P\in C}(Q \mapsto \nu(X \mapsto T_1))(P).\end{aligned}$$

  We prove the remaining $f^n(\top) = \bigcap_{P\in C} g_P^n(\top)$ by induction. The basis follows by $f^0(\top) = \top = \bigcap_{P\in C} \top = \bigcap_{P\in C} g_P^0(\top)$. The induction step follows by

$$\begin{aligned}f^{n+1}(\top) &= f(f^n(\top)) = f(\bigcap_{P\in C} g_P^n(\top)) = \bigcap_{P\in C} f(g_P^n(\top)) \\ &= (\bigcap_{P\in C} T_1(\bigcap C, g_P^n(\top))) = \bigcap_{P\in C}\bigcap_{Q\in C} T_1(Q, g_P^n(\top)) \\ &= (\bigcap_{P\in C} T_1(P, g_P^n(\top))) = (\bigcap_{P\in C} g_P(g_P^n(\top))) = \bigcap_{P\in C} g_P^{n+1}(\top),\end{aligned}$$

  noting that $\{g_P^n(\top) \mid P \in C\}$ is a chain since T_1 is isotone in P by Theorem 61, its elements satisfy $\mathscr{H}_E$ and $\mathscr{H}_B$ since $\top$ satisfies and g_P preserves these properties by Theorems 62 and 70, f is continuous and T_1 is continuous in Q. □

3.3.4 Totality

Besides finite branching, another reasonable condition for computation purposes is totality, or non-empty branching. Consider the usual interpretation of relations as programs and specifications. Then $\subseteq$ models refinement: $P \subseteq Q$ states that the program P implements the specification Q, because any observation of the execution of P is admitted by Q. But since the empty relation $\bot\!\!\!\bot$ is the least element with respect to $\subseteq$, it implements *any* specification. More generally, the refinement interpretation of P fails if some state has no successors under P. In practice every state has at least one successor, even if it only describes the occurrence of an error. This is formalised by the usual totality condition of relations.

Definition 74.

$$\mathscr{H}_T(P) \quad \Leftrightarrow_{\text{def}} \quad R\,;\top = \top.$$

Lemma 75. *The constructs given in Definition 60 satisfy/preserve $\mathscr{H}_T$ as follows.*

1. *The constants skip, variable (un)declaration and assignment are total.*
2. *The operations non-empty non-deterministic choice, conditional, sequential and parallel composition preserve totality.*

3. *Let C be a chain such that each $P \in C$ satisfies $\mathscr{H}_R$, $\mathscr{H}_B$ and $\mathscr{H}_T$, then $\mathscr{H}_T(\bigcap C)$.*

4. *Let f be isotone and $(\mathscr{H}_E \wedge \mathscr{H}_B \wedge \mathscr{H}_T)$-preserving, then $\mathscr{H}_T(\nu f)$.*

Proof.

1. $\mathbb{1}\top \supseteq \mathbb{I}\top = \top$. Moreover,

$$\begin{aligned} \textbf{\textit{var}}\ \vec{x}_K\ ;\ \top &= (\exists \vec{x}_K : \mathbb{1})\ ;\ \top \supseteq (\exists \vec{x}_K : \mathbb{I})\ ;\ \top = \top, \\ \textbf{\textit{end}}\ \vec{x}_K\ ;\ \top &= (\exists \vec{x}'_K : \mathbb{1})\ ;\ \top \supseteq (\exists \vec{x}'_K : \mathbb{I})\ ;\ \top = \top. \end{aligned}$$

   Finally, $\vec{x}{\leftarrow}\vec{e}\ ;\ \top = \mathbb{1}\ ;\ \vec{x}{:=}\vec{e}\ ;\ \mathbb{1}\ ;\ \top \supseteq \vec{x}{:=}\vec{e}\ ;\ \top = \top$.

2. Let $S \neq \emptyset$ be a set of total relations and $R \in S$, then $(\bigcup S)\ ;\ \top \supseteq R\ ;\ \top = \top$. Let P and Q be total, then $(P \| Q)\ ;\ \top = (P \| Q)\ ;\ (\top \| \top) = (P\top \| Q\top) = (\top \| \top) = \top$ by Lemmas 93.2 and 93.4. Moreover $PQ\top = P\top = \top$ and

$$\begin{aligned} & (P \blacktriangleleft b \blacktriangleright Q)\ ;\ \top \\ = \ & (b{=}\infty \cup (b{=}\bot \cap \vec{x}'{=}\bot) \cup (b{=}\mathit{true} \cap P) \cup (b{=}\mathit{false} \cap Q))\ ;\ \top \\ = \ & b{=}\infty\ ;\ \top \cup (b{=}\bot \cap \vec{x}'{=}\bot\ ;\ \top) \cup (b{=}\mathit{true} \cap P\ ;\ \top) \cup (b{=}\mathit{false} \cap Q\ ;\ \top) \\ = \ & b{=}\infty \cup b{=}\bot \cup b{=}\mathit{true} \cup b{=}\mathit{false} \\ = \ & \top. \end{aligned}$$

3. If $C = \emptyset$, the claim is trivial, else $(\bigcap C)\ ;\ \top = (\bigcap_{P \in C} P)\ ;\ \top = (\bigcap_{P \in C} P\ ;\ \top) = \top$ by Lemma 71.

4. Relations satisfying $\mathscr{H}_E$, $\mathscr{H}_B$ and $\mathscr{H}_T$ are closed under infima of chains by Lemmas 53.4 and 69.3 and part 3. The claim therefore follows by Corollary 89. □

Theorem 76. *Let P be a term composed of constants satisfying $\mathscr{H}_E$, $\mathscr{H}_B$ and $\mathscr{H}_T$ (in particular skip, (un)declaration and assignment), sequential and parallel composition, finite non-empty non-deterministic choice, conjunction of chains, conditional and greatest fixpoint. Let $\mathscr{T}_{\preccurlyeq}(b)$ for all conditions b that occur as $\_ \blacktriangleleft b \blacktriangleright \_$ in P. Let $\mathscr{T}_{\preccurlyeq}(\vec{e})$ for all expressions $\vec{e}$ that occur as $\_{\leftarrow}\vec{e}$ in P. Then $\mathscr{H}_T(P)$.*

Proof. A more general claim is the following: Let S be any subterm of P, then $\mathscr{H}_E(T)$, $\mathscr{H}_B(T)$ and $\mathscr{H}_T(T)$ for any T obtained by substituting in S any constant relations satisfying $\mathscr{H}_E$, $\mathscr{H}_B$ and $\mathscr{H}_T$ for the free variables of S. Any T so obtained satisfies $\mathscr{H}_E(T)$ by Theorem 62 and $\mathscr{H}_B(T)$ by Theorem 70. The proof for $\mathscr{H}_T$ is also by induction and analogous to the proofs of Theorems 62 and 70, this time using Lemma 75 for the individual cases. □

3.4 Dependence

We now have a relational theory of computations where undefined and defined variables coexist. In this section we discuss two aspects of non-strictness that can be described in terms of dependence of variables. The first gives conditions in case the computation has non-strict parts, and the second gives conditions if it has no strict parts. Let us illustrate the distinction in the case $m = 1$ and $n = 1$, that is, a single input and output variable.

The relation R has a non-strict part if there is some $x'_1 \neq \bot$ such that $(\bot, x'_1) \in R$. For this part, the value of x'_1 must not depend on the value of x_1. In other words, there must be a constant assignment to x'_1. We therefore obtain the condition $(x_1, x'_1) \in R$ for all x_1. This essentially reflects that one cannot test for undefinedness: If the value of a variable is undefined, such a test is necessarily undefined, too.

The relation R has no strict part if $(\bot, \bot) \notin R$. Then, the value of x'_1 must not depend on the value of x_1 for any part. Hence the above condition is not sufficient because we must assure that *only* constant assignments occur. This is achieved by requiring $(x_1, x'_1) \in R$ for all x_1, if $(x_1, x'_1) \in R$ for some x_1.

Each of the following two sections provides a detailed treatment of one of these aspects. Both sections employ the same schema: First, an inductive argument is applied to infer a healthiness condition for arbitrary values of m and n. Second, this healthiness condition is transformed and expressed in order-theoretic terms. Third, we prove it is satisfied by the programming constructs.

For a sequence $\vec{x}$ of length n let $\vec{x}_{i\mapsto a}$ denote $x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n$, that is, the replacement of x_i by a. If $I \subseteq 1..n$, let $\vec{x}_{I\mapsto a}$ denote the replacement of x_i by a in $\vec{x}$ for all $i \in I$. If $\vec{y}$ is another sequence of length n, let $\vec{x}_{I\mapsto \vec{y}}$ denote the replacement of x_i by y_i in $\vec{x}$ for all $i \in I$.

3.4.1 Non-strict parts

We first treat the non-strict parts of a relation. This starts with an inductive argument that leads to a healthiness condition, which is then expressed in relational terms and afterwards simplified. In Section 3.4.1.2 we restate this condition in terms of partial orders. We then show that the healthiness condition is satisfied by almost all of our programming constructs.

The purpose of the following two sections is to show how the healthiness condition has been derived. If the reader prefers to skip this motivation, we suggest to regard the definition of the order $\sqsubseteq$ at the beginning of Section 3.4.1.2 and to continue with Definition 80.

3.4.1.1 Inductive derivation

We will derive ever more complex predicates stating requirements for the non-strict parts of a computation. While stating (counter)examples we assume the same range $D = D_i$ for all $i \in 1..m$.

Case $m = 1$ and $n = 1$: Just as stated above, a non-strict part of the relation R is given by an outcome $x_1' \neq \bot$ for $x_1 = \bot$. Then x_1' must be an outcome for all x_1. We therefore have

$$\forall x_1' : (x_1' \neq \bot \ \wedge \ (\bot, x_1') \in R) \Rightarrow \forall x_1 : (x_1, x_1') \in R.$$

Case $m = 2$ and $n = 1$: If there is an outcome $x_1' \neq \bot$ for $x_1 = \bot$ and a given x_2, the argument applies as before, and x_1' must not depend on x_1. We therefore have

$$\forall x_2, x_1' : (x_1' \neq \bot \ \wedge \ (\bot, x_2, x_1') \in R) \Rightarrow \forall x_1 : (x_1, x_2, x_1') \in R,$$

using the more readable triple (x_1, x_2, x_1') instead of the formally correct pair $((x_1, x_2), x_1')$. Similarly, x_1' must not depend on x_2 if $(x_1, \bot, x_1') \in R$ for $x_1' \neq \bot$ and a given x_1. We therefore have

$$\forall x_1, x_1' : (x_1' \neq \bot \ \wedge \ (x_1, \bot, x_1') \in R) \Rightarrow \forall x_2 : (x_1, x_2, x_1') \in R.$$

One may ask if it is justified to consider dependence on x_1 and x_2 separately, or if their combination must be treated as well. This means, we would also like to have

$$\forall x_1' : (x_1' \neq \bot \ \wedge \ (\bot, \bot, x_1') \in R) \Rightarrow \forall x_1, x_2 : (x_1, x_2, x_1') \in R.$$

And indeed this follows from the first two predicates: Let $x_1' \neq \bot$ such that $(\bot, \bot, x_1') \in R$, then $\forall x_1 : (x_1, \bot, x_1') \in R$ by the first predicate. Therefore $\forall x_1, x_2 : (x_1, x_2, x_1') \in R$ by the second predicate.

Furthermore, by choosing the total $R = \{(x_1, x_2, \bot) \mid x_1, x_2 \in D\} \cup \{(\bot, a, a), (a, \bot, a)\}$ for some $a \neq \bot$, one can see that the combined predicate is strictly weaker than the conjunction of the first two. We therefore should take those two predicates.

Case $n = 1$: By generalising the previous arguments, we obtain a predicate for arbitrary m:

$$\forall i \in 1..m : \forall \vec{x} : \forall x'_1 : (x'_1 \neq \bot \ \wedge \ (\vec{x}_{i \mapsto \bot}, x'_1) \in R) \Rightarrow \forall \tilde{x}_i : (\vec{x}_{i \mapsto \tilde{x}_i}, x'_1) \in R.$$

Since x_i is not free in the antecedent, this may be written more compactly as

$$\forall i \in 1..m : \forall \vec{x} : \forall x'_1 : (x'_1 \neq \bot \ \wedge \ (\vec{x}_{i \mapsto \bot}, x'_1) \in R) \Rightarrow (\vec{x}, x'_1) \in R.$$

Case $m = 1$ and $n = 2$: We again use the more readable triple (x_1, x'_1, x'_2), now instead of the formally correct pair $(x_1, (x'_1, x'_2))$. If $(\bot, x'_1, x'_2) \in R$ for some $x'_1 \neq \bot$ and a given x'_2, we cannot assume that $(x_1, x'_1, x'_2) \in R$ for all x_1. This is because x'_2 may depend on x_1 in a non-constant manner. The assumption can be weakened, however, such that there is some $\tilde{x}'_2$ satisfying $(x_1, x'_1, \tilde{x}'_2) \in R$ for all x_1. We therefore have

$$\forall x'_1, x'_2 : (x'_1 \neq \bot \ \wedge \ (\bot, x'_1, x'_2) \in R) \Rightarrow \forall x_1 : \exists \tilde{x}'_2 : (x_1, x'_1, \tilde{x}'_2) \in R.$$

We similarly obtain

$$\forall x'_1, x'_2 : (x'_2 \neq \bot \ \wedge \ (\bot, x'_1, x'_2) \in R) \Rightarrow \forall x_1 : \exists \tilde{x}'_1 : (x_1, \tilde{x}'_1, x'_2) \in R.$$

It is nevertheless possible that both x'_1 and x'_2 do not depend on x_1. One may ask if it is also justified to consider dependence of x'_1 and x'_2 separately. This means, we would also like to have

$$\forall x'_1, x'_2 : (x'_1 \neq \bot \ \wedge \ x'_2 \neq \bot \ \wedge \ (\bot, x'_1, x'_2) \in R) \Rightarrow \forall x_1 : (x_1, x'_1, x'_2) \in R.$$

Unfortunately, this is not a consequence of the first two predicates. To see this, consider the total relation $R = \{(x_1, a, \bot) \mid x_1 \in D\} \cup \{(x_1, \bot, a) \mid x_1 \in D\} \cup \{(\bot, a, a)\}$ for some $a \neq \bot$. The first two predicates are satisfied, but the third is not since $(\bot, a, a) \in R$ and $(a, a, a) \notin R$. Therefore the combination of x'_1 and x'_2 must be treated as well.

Conversely, the third predicate does not imply any of the first two, as can be verified by choosing the total $R = \{(x_1, \bot, \bot) \mid x_1 \in D\} \cup \{(\bot, \bot, a), (\bot, a, \bot)\}$ for some $a \neq \bot$.

In the presence of the third predicate, the first may be simplified to

$$\forall x'_1 : (x'_1 \neq \bot \ \wedge \ (\bot, x'_1, \bot) \in R) \Rightarrow \forall x_1 : \exists \tilde{x}'_2 : (x_1, x'_1, \tilde{x}'_2) \in R,$$

because its instances for $x'_2 \neq \bot$ are already satisfied. The second predicate similarly simplifies to

$$\forall x'_2 : (x'_2 \neq \bot \ \wedge \ (\bot, \bot, x'_2) \in R) \Rightarrow \forall x_1 : \exists \tilde{x}'_1 : (x_1, \tilde{x}'_1, x'_2) \in R.$$

The picture may be completed by adding a fourth predicate

$$(\bot, \bot, \bot) \in R \Rightarrow \forall x_1 : \exists \tilde{x}'_1, \tilde{x}'_2 : (x_1, \tilde{x}'_1, \tilde{x}'_2) \in R.$$

This is clearly satisfied if R is total. We therefore should take all four predicates.

Case $m = 1$: By generalising the previous arguments, we obtain a predicate for arbitrary n:

$$\forall J \subseteq 1..n : \forall \vec{x}' : (\bot \notin \vec{x}'_{\overline{J}} \ \wedge \ (\bot, \vec{x}'_{J \mapsto \bot}) \in R) \Rightarrow \forall x_1 : \exists \vec{\tilde{x}}' : (x_1, \vec{\tilde{x}}'_{\overline{J} \mapsto \vec{x}'}) \in R.$$

Since x_1 and $\vec{x}'_J$ are not free in the antecedent, this may be written more compactly as

$$\forall J \subseteq 1..n : \forall x_1 : \forall \vec{x}'_{\overline{J}} : \exists \vec{x}'_J : (\bot \notin \vec{x}'_{\overline{J}} \ \wedge \ (\bot, \vec{x}'_{J \mapsto \bot}) \in R) \Rightarrow (x_1, \vec{x}') \in R.$$

The general case: Combining the previous arguments, we obtain a predicate for arbitrary m and n:

$$\forall i \in 1..m : \forall J \subseteq 1..n : \forall \vec{x}_{\overline{i}} : \forall \vec{x}'_{\overline{J}} : (\bot \notin \vec{x}'_{\overline{J}} \ \wedge \ (\vec{x}_{i \mapsto \bot}, \vec{x}'_{J \mapsto \bot}) \in R) \Rightarrow \forall x_i : \exists \vec{x}'_J : (\vec{x}, \vec{x}') \in R.$$

Intuitively, the antecedent states that for $x_i{=}\bot$ there is an outcome such that $x'_j{\neq}\bot$ if and only if $j \notin J$. Then all such x'_j must not depend on x_i. This means that there must be an outcome with these values of x'_j for all values of x_i.

Eliminating the implication, the general predicate is equivalent to

$$\forall i \in 1..m : \forall J \subseteq 1..n : \forall \vec{x} : \forall \vec{x}'_{\overline{J}} : \exists \vec{x}'_J : \bot \in \vec{x}'_{\overline{J}} \vee (\vec{x}_{i \mapsto \bot}, \vec{x}'_{J \mapsto \bot}) \notin R \vee (\vec{x}, \vec{x}') \in R.$$

Healthiness condition: We derive a relational healthiness condition for the general case. We start with

$$\forall i \in 1..m : \forall J \subseteq 1..n : \forall \vec{x} : \forall \vec{x}'_{\overline{J}} : (\bot \notin \vec{x}'_{\overline{J}} \ \wedge \ (\vec{x}_{i \mapsto \bot}, \vec{x}'_{J \mapsto \bot}) \in R) \ \Rightarrow \ \exists \vec{x}'_J : (\vec{x}, \vec{x}') \in R,$$

and proceed as follows:

$$\begin{array}{ll}
& (\bot \notin \vec{x}'_{\overline{J}} \ \wedge \ (\vec{x}_{i \mapsto \bot}, \vec{x}'_{J \mapsto \bot}) \in R) \ \Rightarrow \ \exists \vec{x}'_J : (\vec{x}, \vec{x}') \in R \\
\Leftrightarrow & \text{«introduction of intermediate variables and rearrangement»} \\
& (\exists \vec{y}, \vec{z} : \vec{x}_{i \mapsto \bot}{=}\vec{y} \ \wedge \ (\vec{y}, \vec{z}) \in R \ \wedge \ \vec{z}{=}\vec{x}'_{J \mapsto \bot} \ \wedge \ \bot \notin \vec{x}'_{\overline{J}}) \ \Rightarrow \\
& \quad \exists \vec{w} : (\vec{x}, \vec{w}) \in R \ \wedge \ \exists \vec{x}'_J : \vec{w}{=}\vec{x}' \\
\Leftrightarrow & \text{«explicit relations and sequence equality»} \\
& (\exists \vec{y}, \vec{z} : (\vec{x}, \vec{y}) \in \{(\vec{x}, \vec{x}') \mid \vec{x}_{\overline{i}}{=}\vec{x}'_{\overline{i}} \ \wedge \ \bot{=}x'_i\} \ \wedge \ (\vec{y}, \vec{z}) \in R \ \wedge \\
& \qquad (\vec{z}, \vec{x}'_{\overline{J}}) \in \{(\vec{x}, \vec{x}'_{\overline{J}}) \mid \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}} \ \wedge \ \vec{x}_J{=}\bot \ \wedge \ \bot \notin \vec{x}'_{\overline{J}}\}) \ \Rightarrow \\
& \quad \exists \vec{w} : (\vec{x}, \vec{w}) \in R \ \wedge \ (\vec{w}, \vec{x}'_{\overline{J}}) \in (\exists \vec{x}'_J : \mathbb{I}) \\
\Leftrightarrow & \text{«definition of assignment and sequential composition»} \\
& (\vec{x}, \vec{x}'_{\overline{J}}) \in x_i{:=}\bot \ ; R \ ; (\exists \vec{x}'_J : \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}} \ \cap \ \vec{x}_J{=}\bot \ \cap \ \bot \notin \vec{x}_{\overline{J}}) \ \Rightarrow \ (\vec{x}, \vec{x}'_{\overline{J}}) \in R \ ; (\exists \vec{x}'_J : \mathbb{I}).
\end{array}$$

Therefore, switching to the relational form,

$$\begin{array}{ll}
& \forall \vec{x} : \forall \vec{x}'_{\overline{J}} : (\bot \notin \vec{x}'_{\overline{J}} \ \wedge \ (\vec{x}_{i \mapsto \bot}, \vec{x}'_{J \mapsto \bot}) \in R) \ \Rightarrow \ \exists \vec{x}'_J : (\vec{x}, \vec{x}') \in R \\
\Leftrightarrow & \text{«definition of subset relation»} \\
& x_i{:=}\bot \ ; R \ ; (\exists \vec{x}'_J : \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}} \ \cap \ \vec{x}_J{=}\bot \ \cap \ \bot \notin \vec{x}_{\overline{J}}) \ \subseteq \ R \ ; (\exists \vec{x}'_J : \mathbb{I}) \\
\Leftrightarrow & \text{«dimension restriction»} \\
& x_i{:=}\bot \ ; R \ ; (\vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}} \ \cap \ \vec{x}_J{=}\bot \ \cap \ \bot \notin \vec{x}_{\overline{J}}) \ ; (\exists \vec{x}'_J : \mathbb{I}) \ \subseteq \ R \ ; (\exists \vec{x}'_J : \mathbb{I}) \\
\Leftrightarrow & \text{«vector and functional properties»} \\
& x_i{:=}\bot \ ; (R \ \cap \ \vec{x}'_J{=}\bot \ \cap \ \bot \notin \vec{x}'_{\overline{J}}) \ ; \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}} \ \subseteq \ R \ ; (\exists \vec{x}'_J : \mathbb{I}) \ ; (\exists \vec{x}_J : \mathbb{I}) \\
\Leftrightarrow & \text{«dimension restriction»} \\
& x_i{:=}\bot \ ; (R \ \cap \ \vec{x}'_J{=}\bot \ \cap \ \bot \notin \vec{x}'_{\overline{J}}) \ ; \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}} \ \subseteq \ R \ ; \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}} \\
\Leftrightarrow & \text{«}\vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}} \ ; \vec{x}_J{:=}\bot = \vec{x}_J{:=}\bot \text{ and } \vec{x}_J{:=}\bot \ ; \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}} = \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}}\text{»} \\
& x_i{:=}\bot \ ; (R \ \cap \ \vec{x}'_J{=}\bot \ \cap \ \bot \notin \vec{x}'_{\overline{J}}) \ ; \vec{x}_J{:=}\bot \ \subseteq \ R \ ; \vec{x}_J{:=}\bot \\
\Leftrightarrow & \text{«vectors and definition of assignment»} \\
& x_i{:=}\bot \ ; R \ ; (\vec{x}_J{=}\bot \ \cap \ \bot \notin \vec{x}_{\overline{J}} \ \cap \ \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \ \cap \ \vec{x}'_J{=}\bot) \ \subseteq \ R \ ; \vec{x}_J{:=}\bot \\
\Leftrightarrow & \text{«substitution of equals»} \\
& x_i{:=}\bot \ ; R \ ; (\vec{x}_J{=}\vec{x}'_J \ \cap \ \bot \notin \vec{x}'_{\overline{J}} \ \cap \ \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \ \cap \ \vec{x}'_J{=}\bot) \ \subseteq \ R \ ; \vec{x}_J{:=}\bot \\
\Leftrightarrow & \text{«definition of identity and vectors»} \\
& x_i{:=}\bot \ ; R \ \cap \ \vec{x}'_J{=}\bot \ \cap \ \bot \notin \vec{x}'_{\overline{J}} \ \subseteq \ R \ ; \vec{x}_J{:=}\bot \\
\Leftrightarrow & \text{«shunting»} \\
& x_i{:=}\bot \ ; R \ \cap \ \vec{x}'_J{=}\bot \ \subseteq \ R \ ; \vec{x}_J{:=}\bot \ \cup \ \bot \in \vec{x}'_{\overline{J}}.
\end{array}$$

Thus the healthiness condition for the non-strict parts of the relation R is

$$\forall i \in 1..m : \forall J \subseteq 1..n : x_i{:=}\bot \ ; R \ \cap \ \vec{x}'_J{=}\bot \ \subseteq \ R \ ; \vec{x}_J{:=}\bot \ \cup \ \bot \in \vec{x}'_{\overline{J}}.$$

3.4.1.2 Order formulation

We derive an order-theoretic representation of the healthiness condition for the non-strict parts. To this end, we introduce an order similar to $\preccurlyeq$, but now with respect to $\bot$. Let $\sqsubseteq : D_i \leftrightarrow D_i$ be the flat order on D_i with $\bot$ as its least element, that is, $x \sqsubseteq y \Leftrightarrow_{\text{def}} x{=}\bot \vee x{=}y$. Again, the order is extended pointwise to D_I by $\vec{x}_I \sqsubseteq \vec{y}_I \Leftrightarrow_{\text{def}} \forall i \in I : x_i \sqsubseteq y_i$, and its dual order is denoted by $\sqsupseteq =_{\text{def}} \sqsubseteq^{\smile}$. The properties of $\preccurlyeq$ can be transferred to $\sqsubseteq$.

Lemma 77.

1. $\sqsupseteq = \bigcup_{I \subseteq 1..m} \vec{x}_I{:=}\bot$.
2. $A \,;\, (\exists \vec{x}_K : B) \,;\, C$ *is the same for any* $\{\sqsubseteq\} \subseteq \{A, B, C\} \subseteq \{\sqsubseteq, \mathbb{I}\}$.
3. $A \,;\, (\exists \vec{x}'_K : B) \,;\, C$ *is the same for any* $\{\sqsubseteq\} \subseteq \{A, B, C\} \subseteq \{\sqsubseteq, \mathbb{I}\}$.

Proof. Analogous to the proofs of Lemmas 55.2 and 58. □

Using this order, we can algebraically define the healthiness condition for the non-strict parts, see Lemma 79.4. It is proved equivalent to the inductively derived result of Section 3.4.1.1. The intermediate steps generalise the index $i \in 1..m$ to a subset $I \subseteq 1..m$ and eliminate the term $\vec{x}'_J{=}\bot$ in favour of a disjunction on the right hand side. All of this is shown in Lemma 79, but before that we need the following ancillary result concerning a generalised distribution law.

Lemma 78.

1. *Let L be a bounded distributive lattice, I a finite set and $a_i, b_i \in L$ for all $i \in I$. Then* $\bigcap_{i\in I} a_i \cup b_i = \bigcup_{J\subseteq I}(\bigcap_{i\in J} a_i) \cap (\bigcap_{i\in I\setminus J} b_i)$.
2. *Let I be a finite set. For all $i \in I$, let A_i, B_i, C and D be relations such that $A_i C \subseteq D$ and $\bigcap_{i\in I} B_i = \perp\!\!\!\perp$, and A_i and B_i have the same type. Then* $(\bigcap_{i\in I} A_i \cup B_i)C \subseteq D$.
3. *Let I be a finite set, then* $\bigcap_{J\subseteq I} \vec{x}_J{\neq}\bot \;\cup\; \bot{\in}\vec{x}_{I\setminus J} = \perp\!\!\!\perp$.

Proof.

1. We use induction on the cardinality of I. If $I = \emptyset$, both sides of the equation are the greatest element of the lattice. Otherwise let $I = \{k\} \cup I'$ for some $k \notin I'$, then

$$\begin{array}{ll} & \bigcap_{i\in I} a_i \cup b_i \\ = & (a_k \cup b_k) \cap \bigcap_{i\in I'} a_i \cup b_i \\ = & (a_k \cup b_k) \cap \bigcup_{J\subseteq I'} (\bigcap_{i\in J} a_i) \cap (\bigcap_{i\in I'\setminus J} b_i) \\ = & (a_k \cap \bigcup_{J\subseteq I'} (\bigcap_{i\in J} a_i) \cap (\bigcap_{i\in I'\setminus J} b_i)) \cup (b_k \cap \bigcup_{J\subseteq I'} (\bigcap_{i\in J} a_i) \cap (\bigcap_{i\in I'\setminus J} b_i)) \\ = & (\bigcup_{J\subseteq I'} a_k \cap (\bigcap_{i\in J} a_i) \cap (\bigcap_{i\in I'\setminus J} b_i)) \cup (\bigcup_{J\subseteq I'} b_k \cap (\bigcap_{i\in J} a_i) \cap (\bigcap_{i\in I'\setminus J} b_i)) \\ = & (\bigcup_{k\in J\subseteq I} (\bigcap_{i\in J} a_i) \cap (\bigcap_{i\in I\setminus J} b_i)) \cup (\bigcup_{k\notin J\subseteq I} (\bigcap_{i\in J} a_i) \cap (\bigcap_{i\in I\setminus J} b_i)) \\ = & \bigcup_{J\subseteq I} (\bigcap_{i\in J} a_i) \cap (\bigcap_{i\in I\setminus J} b_i). \end{array}$$

2. By part 1,

$$\begin{array}{rcl} (\bigcap_{i\in I} A_i \cup B_i)C & = & (\bigcup_{J\subseteq I} (\bigcap_{i\in J} A_i) \cap (\bigcap_{i\in I\setminus J} B_i))C \\ & \subseteq & (\bigcup_{\emptyset\neq J\subseteq I} \bigcap_{i\in J} A_i)C \cup (\bigcap_{i\in I} B_i)C \\ & \subseteq & \bigcup_{\emptyset\neq J\subseteq I} \bigcap_{i\in J} A_i C \;\subseteq\; \bigcup_{\emptyset\neq J\subseteq I} \bigcap_{i\in J} D \;=\; \bigcup_{\emptyset\neq J\subseteq I} D \;\subseteq\; D. \end{array}$$

3. By the dual statement of part 1,

$$\begin{array}{rcl} \bigcap_{J\subseteq I} \vec{x}_J{\neq}\bot \;\cup\; \bot{\in}\vec{x}_{I\setminus J} & = & \bigcap_{J\subseteq I} (\bigcup_{i\in J} x_i{\neq}\bot) \;\cup\; (\bigcup_{i\in I\setminus J} x_i{=}\bot) \\ & = & \bigcup_{i\in I} x_i{\neq}\bot \;\cap\; x_i{=}\bot \;=\; \perp\!\!\!\perp. \end{array}$$

□

Lemma 79. *The following statements are equivalent:*

1. $\forall i \in 1..m : \forall J \subseteq 1..n : x_i{:=}\bot \,;\, P \cap \vec{x}'_J{=}\bot \subseteq P \,;\, \vec{x}_J{:=}\bot \cup \bot{\in}\vec{x}'_{\overline{J}}$.
2. $\forall I \subseteq 1..m : \forall J \subseteq 1..n : \vec{x}_I{:=}\bot \,;\, P \cap \vec{x}'_J{=}\bot \subseteq P \,;\, \vec{x}_J{:=}\bot \cup \bot{\in}\vec{x}'_{\overline{J}}$.
3. $\forall I \subseteq 1..m : \forall J \subseteq 1..n : \vec{x}_I{:=}\bot \,;\, P \cap \bot{\notin}\vec{x}'_{\overline{J}} \subseteq \bigcup_{K\subseteq J} P \,;\, \vec{x}_K{:=}\bot$.
4. $\sqsupseteq \,;\, P \subseteq P \,;\, \sqsupseteq$.

Proof. The part $(1 \Rightarrow 2)$ is proved by induction on the size of I. The basis $I = \emptyset$ follows by

$$P \cap \vec{x}'_J{=}\bot \;=\; P \,;\, (\mathbb{I} \cap \vec{x}'_J{=}\bot) \subseteq P \,;\, (\vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \cap \vec{x}'_J{=}\bot) \;=\; P \,;\, \vec{x}_J{:=}\bot$$

since $\vec{x}_\emptyset{:=}\bot \;=\; \mathbb{I}$. For the induction step assume $i \notin I$; then, using the shunted versions of the conditions,

$$\begin{array}{ll}
& \vec{x}_{I\cup\{i\}}{:=}\bot \,;\, P \\
= & \text{«property of assignment»} \\
& x_i{:=}\bot \,;\, \vec{x}_I{:=}\bot \,;\, P \\
\subseteq & \text{«induction hypothesis»} \\
& x_i{:=}\bot \,;\, (P \,;\, \vec{x}_J{:=}\bot \cup \vec{x}'_J{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{J}}) \\
\subseteq & \text{«relation algebra»} \\
& x_i{:=}\bot \,;\, P \,;\, (\vec{x}_J{:=}\bot \cap \bot{\notin}\vec{x}'_{\overline{J}}) \cup \top \,;\, \vec{x}'_J{\neq}\bot \cup \top \,;\, \bot{\in}\vec{x}'_{\overline{J}} \\
\subseteq & \text{«statement 1 for all } K \subseteq J\text{»} \\
& (\bigcap_{K\subseteq J} P \,;\, \vec{x}_K{:=}\bot \cup \vec{x}'_K{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{K}}) \,;\, (\bot{\notin}\vec{x}_{\overline{J}} \cap \vec{x}_J{:=}\bot) \cup \vec{x}'_J{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{J}} \\
= & \text{«relation algebra»} \\
& (\bigcap_{K\subseteq J} (P \,;\, \vec{x}_K{:=}\bot \cup \vec{x}'_K{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{K}}) \cap \bot{\notin}\vec{x}'_{\overline{J}}) \,;\, \vec{x}_J{:=}\bot \cup \vec{x}'_J{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{J}} \\
\subseteq & \text{«relation algebra»} \\
& (\bigcap_{K\subseteq J} P \,;\, \vec{x}_K{:=}\bot \cup \vec{x}'_K{\neq}\bot \cup \bot{\in}\vec{x}'_{J\cap\overline{K}}) \,;\, \vec{x}_J{:=}\bot \cup \vec{x}'_J{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{J}} \\
\subseteq & \text{«Lemma 78.2 using } P \,;\, \vec{x}_K{:=}\bot \,;\, \vec{x}_J{:=}\bot \;=\; P \,;\, \vec{x}_J{:=}\bot \text{ and Lemma 78.3»} \\
& P \,;\, \vec{x}_J{:=}\bot \cup \vec{x}'_J{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{J}}.
\end{array}$$

The part $(2 \Rightarrow 4)$ follows using Lemmas 77.1 and 78.3 in

$$\begin{array}{rl}
& \sqsupseteq \,;\, P \;=\; (\bigcup_{I\subseteq 1..m} \vec{x}_I{:=}\bot \,;\, P) \subseteq \bigcap_{J\subseteq 1..n} P \,;\, \vec{x}_J{:=}\bot \cup \vec{x}'_J{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{J}} \\
\subseteq & (\bigcap_{J\subseteq 1..n} P \,;\, \overline{\sqsupseteq} \cup \vec{x}'_J{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{J}}) \;=\; P \,;\, \sqsupseteq \cup (\bigcap_{J\subseteq 1..n} \vec{x}'_J{\neq}\bot \cup \bot{\in}\vec{x}'_{\overline{J}}) \;=\; P \,;\, \sqsupseteq.
\end{array}$$

For the part $(4 \Rightarrow 3)$ observe that by Lemma 77.1,

$$\begin{array}{ll}
& \vec{x}_I{:=}\bot \,;\, P \cap \bot{\notin}\vec{x}'_{\overline{J}} \subseteq \sqsupseteq \,;\, P \cap \bot{\notin}\vec{x}'_{\overline{J}} \subseteq P \,;\, \sqsupseteq \cap \bot{\notin}\vec{x}'_{\overline{J}} \\
= & P \,;\, (\bigcup_{K\subseteq 1..m} \vec{x}_K{:=}\bot) \cap \bot{\notin}\vec{x}'_{\overline{J}} = \bigcup_{K\subseteq 1..m} P \,;\, \vec{x}_K{:=}\bot \cap \vec{x}'_K{=}\bot \cap \bot{\notin}\vec{x}'_{\overline{J}} \\
= & (\bigcup_{K\subseteq J} P \,;\, \vec{x}_K{:=}\bot \cap \vec{x}'_K{=}\bot \cap \bot{\notin}\vec{x}'_{\overline{J}}) \subseteq \bigcup_{K\subseteq J} P \,;\, \vec{x}_K{:=}\bot,
\end{array}$$

using $K \cap \overline{J} \neq \emptyset \Rightarrow \vec{x}'_K{=}\bot \cap \bot{\notin}\vec{x}'_{\overline{J}} = \mathbb{\perp}$ in the fifth step. The part $(3 \Rightarrow 1)$ follows by

$$\begin{array}{ll}
& \vec{x}_i{:=}\bot \,;\, P \cap \vec{x}'_J{=}\bot \cap \bot{\notin}\vec{x}'_{\overline{J}} \subseteq (\bigcup_{K\subseteq J} P \,;\, \vec{x}_K{:=}\bot) \cap \vec{x}'_J{=}\bot \\
= & P \,;\, (\bigcup_{K\subseteq J} \vec{x}'_{\overline{K}}{=}\vec{x}_{\overline{K}} \cap \vec{x}'_K{=}\bot \cap \vec{x}'_J{=}\bot) \subseteq P \,;\, (\bigcup_{K\subseteq J} \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \cap \vec{x}'_J{=}\bot) \\
= & P \,;\, \vec{x}_J{:=}\bot.
\end{array}$$

□

We thus define the healthiness condition $\mathscr{H}_N$ for the non-strict parts of a relation P. If P is a mapping, the condition $\mathscr{H}_N(P)$ states that P is isotone with respect to $\sqsupseteq$ [Sch06]. Further remarks about the following, more general condition are given in Section 3.4.2.2 once the second healthiness condition is established.

Definition 80.

$$\mathscr{H}_N(P) \quad \Leftrightarrow_{\text{def}} \quad \sqsupseteq \,;\, P \subseteq P \,;\, \sqsupseteq.$$

3.4.1.3 Closure

We can now show that our programming constructs satisfy $\mathscr{H}_N$. To deal with the assignment and the conditional, we assume that the expressions are isotone with respect to $\sqsubseteq$. That is, we now need $\mathscr{T}_\sqsubseteq(\vec{e}) \Leftrightarrow \sqsubseteq\,;\,\vec{x}'{=}\vec{e} \subseteq \vec{x}'{=}\vec{e}\,;\,\sqsubseteq$. Since $\preccurlyeq$ and $\sqsubseteq$ are structurally similar, the properties of $\mathscr{T}_\preccurlyeq$ can be transferred to $\mathscr{T}_\sqsubseteq$. For example, $\vec{e}$ is isotone with respect to $\sqsubseteq$ iff it is with respect to $\sqsupseteq$. See furthermore the remarks after Definition 56 and in Section 3.6.2.

Lemma 81. *The constructs given in Definition 60 satisfy/preserve $\mathscr{H}_N$ as follows.*

1. $\mathscr{H}_N(\mathbb{1})$.
2. $(\forall P \in S : \mathscr{H}_N(P)) \Rightarrow \mathscr{H}_N(\bigcup S)$.
3. $\mathscr{H}_N(P) \wedge \mathscr{H}_N(Q) \Rightarrow \mathscr{H}_N(P\,;\,Q)$.
4. $\mathscr{T}_\sqsubseteq(\vec{e}) \Rightarrow \mathscr{H}_N(\vec{x}{\leftarrow}\vec{e})$.
5. $\mathscr{H}_N(\textbf{\textit{var}}\ \vec{x}_K) \wedge \mathscr{H}_N(\textbf{\textit{end}}\ \vec{x}_K)$.
6. $\mathscr{H}_T(P) \wedge \mathscr{H}_T(Q) \wedge \mathscr{H}_N(P) \wedge \mathscr{H}_N(Q) \wedge \mathscr{T}_\sqsubseteq(b) \Rightarrow \mathscr{H}_N(P \blacktriangleleft b \blacktriangleright Q)$.
7. $\mathscr{H}_N(P) \wedge \mathscr{H}_N(Q) \Rightarrow \mathscr{H}_N(P \| Q)$.
8. *Let C be a chain, then* $(\forall P \in C : \mathscr{H}_R(P) \wedge \mathscr{H}_B(P) \wedge \mathscr{H}_N(P)) \Rightarrow \mathscr{H}_N(\bigcap C)$. *In general,* $\mathscr{H}_E(P) \wedge \mathscr{H}_B(P) \wedge \mathscr{H}_N(P) \wedge \mathscr{H}_E(Q) \wedge \mathscr{H}_B(Q) \wedge \mathscr{H}_N(Q) \not\Rightarrow \mathscr{H}_N(P \cap Q)$.
9. *Let f be isotone and $\mathscr{H}_N$-preserving, then* $\mathscr{H}_N(\mu f)$.
10. *Let f be isotone and $(\mathscr{H}_E \wedge \mathscr{H}_B \wedge \mathscr{H}_T \wedge \mathscr{H}_N)$-preserving, then* $\mathscr{H}_N(\nu f)$.

Proof.

1. We show $\sqsupseteq\,;\preccurlyeq = \preccurlyeq\,;\sqsupseteq$, which follows using

$$\begin{aligned}(\vec{x},\vec{z}) \in \sqsupseteq\,;\preccurlyeq &\Leftrightarrow (\exists \vec{y} : \vec{y}{\sqsubseteq}\vec{x} \wedge \vec{y}{\preccurlyeq}\vec{z}) \Leftrightarrow (\forall i \in 1..n : x_i{=}\infty \vee z_i{=}\bot \vee x_i{=}z_i)\\ &\Leftrightarrow (\exists \vec{y} : \vec{x}{\preccurlyeq}\vec{y} \wedge \vec{z}{\sqsubseteq}\vec{y}) \Leftrightarrow (\vec{x},\vec{z}) \in \preccurlyeq\,;\sqsupseteq.\end{aligned}$$

   It remains to show both intermediate steps. First, let $\forall i \in 1..n : x_i{=}\infty \vee z_i{=}\bot \vee x_i{=}z_i$. Using $\vec{y}$ given by

$$y_i =_{\text{def}} \begin{cases} x_i, \text{ if } x_i{=}z_i \\ \infty, \text{ if } x_i{\neq}z_i \wedge x_i{=}\infty \\ \bot, \text{ if } x_i{\neq}z_i \wedge x_i{\neq}\infty \wedge z_i{=}\bot \end{cases}$$

   it is easy to verify $\vec{y}{\sqsubseteq}\vec{x} \wedge \vec{y}{\preccurlyeq}\vec{z}$ by case inspection. To obtain $\vec{x}{\preccurlyeq}\vec{y} \wedge \vec{z}{\sqsubseteq}\vec{y}$, use

$$y_i =_{\text{def}} \begin{cases} x_i, \text{ if } x_i{=}z_i \\ z_i, \text{ if } x_i{\neq}z_i \wedge x_i{=}\infty \\ x_i, \text{ if } x_i{\neq}z_i \wedge x_i{\neq}\infty \wedge z_i{=}\bot \end{cases}$$

   Second, let $\exists i \in 1..n : x_i{\neq}\infty \wedge z_i{\neq}\bot \wedge x_i{\neq}z_i$. Assuming $\exists \vec{y} : \vec{y}{\sqsubseteq}\vec{x} \wedge \vec{y}{\preccurlyeq}\vec{z}$, we obtain a contradiction as follows:

   - $y_i{=}\bot$ implies $z_i{=}\bot$ since $y_i{\preccurlyeq}z_i$.
   - $y_i{=}\infty$ implies $x_i{=}\infty$ since $y_i{\sqsubseteq}x_i$.
   - $y_i{\neq}\bot \wedge y_i{\neq}\infty$ implies $x_i{=}y_i{=}z_i$ since $y_i{\sqsubseteq}x_i \wedge y_i{\preccurlyeq}z_i$.

   Assuming $\exists \vec{y} : \vec{x}{\preccurlyeq}\vec{y} \wedge \vec{z}{\sqsubseteq}\vec{y}$, we have $z_i{=}y_i$ since $z_i{\neq}\bot \wedge z_i{\sqsubseteq}y_i$, and $x_i{=}y_i$ since $x_i{\neq}\infty \wedge x_i{\preccurlyeq}y_i$. Hence we obtain the contradiction $z_i{=}y_i{=}x_i$.

2. $\sqsupseteq\,;(\bigcup S) = (\bigcup_{P\in S} \sqsupseteq\,;P) \subseteq (\bigcup_{P\in S} P\,;\sqsupseteq) = (\bigcup S)\,;\sqsupseteq$.

3. $\sqsupseteq ; P ; Q \subseteq P ; \sqsupseteq ; Q \subseteq P ; Q ; \sqsupseteq$.

4. $\mathscr{T}_{\sqsubseteq}(\vec{e}) \Leftrightarrow (\sqsupseteq ; \vec{x}'{=}\vec{e} \subseteq \vec{x}'{=}\vec{e} ; \sqsupseteq) \Leftrightarrow (\sqsupseteq ; \vec{x}{:=}\vec{e} \subseteq \vec{x}{:=}\vec{e} ; \sqsupseteq) \Leftrightarrow \mathscr{H}_N(\vec{x}{:=}\vec{e})$. Therefore $\mathscr{H}_N(\mathbb{1} ; \vec{x}{:=}\vec{e} ; \mathbb{1})$ by parts 1 and 3.

5. We have $\mathscr{H}_N(\exists \vec{x}_K : \mathbb{I})$ and $\mathscr{H}_N(\exists \vec{x}'_K : \mathbb{I})$ by the converse of Lemmas 77.3 and 77.2, from which the claim follows again by parts 1 and 3.

6. We distinguish four cases, depending on the value of b. First,

$$b{=}\infty \cap \sqsupseteq ; (P \blacktriangleleft b \blacktriangleright Q) \subseteq b{=}\infty \subseteq b{=}\infty ; \sqsupseteq.$$

   For the remaining three cases, let $c \in \{\infty, \mathit{true}, \mathit{false}\}$, then

$$\sqsupseteq ; x{=}c = (x' \sqsubseteq x \cap x'{=}c) ; \top \subseteq c \sqsubseteq x ; \top = x{=}c ; \top = x{=}c.$$

   By the Schröder law and reflexivity, $\sqsubseteq ; x{\neq}c \subseteq x{\neq}c \subseteq \sqsubseteq ; x{\neq}c$, hence

$$\begin{aligned} x'{=}b ; \sqsubseteq ; x{\neq}c &= x'{=}b ; x{\neq}c = (x'{=}b \cap x'{\neq}c) ; \top = (x'{=}b \cap b{\neq}c) ; \top \\ &= b{\neq}c \cap x'{=}b ; \top = b{\neq}c. \end{aligned}$$

   Since $\mathscr{T}_{\sqsubseteq}(b)$ we obtain $\sqsubseteq ; b{\neq}c = \sqsubseteq ; x'{=}b ; \sqsubseteq ; x{\neq}c \subseteq x'{=}b ; \sqsubseteq ; x{\neq}c = b{\neq}c$. Again by the Schröder law, $\sqsupseteq ; b{=}c \subseteq b{=}c$. Moreover for any total relation R, in particular for $\vec{x}'{=}\bot$, P and Q, we have $\sqsupseteq ; \vec{x}'{=}\bot \subseteq R ; \top ; \vec{x}'{=}\bot = R ; \vec{x}{:=}\bot \subseteq R ; \sqsupseteq$ by Lemma 77.1. Therefore, finishing the remaining three cases,

$$\begin{aligned} b{=}\bot \cap \sqsupseteq ; (P \blacktriangleleft b \blacktriangleright Q) &\subseteq b{=}\bot \cap \sqsupseteq ; \vec{x}'{=}\bot \subseteq b{=}\bot \cap \vec{x}'{=}\bot ; \sqsupseteq, \\ b{=}\mathit{true} \cap \sqsupseteq ; (P \blacktriangleleft b \blacktriangleright Q) &\subseteq b{=}\mathit{true} \cap \sqsupseteq ; (\vec{x}'{=}\bot \cup P) \subseteq b{=}\mathit{true} \cap P ; \sqsupseteq, \\ b{=}\mathit{false} \cap \sqsupseteq ; (P \blacktriangleleft b \blacktriangleright Q) &\subseteq b{=}\mathit{false} \cap \sqsupseteq ; (\vec{x}'{=}\bot \cup Q) \subseteq b{=}\mathit{false} \cap Q ; \sqsupseteq. \end{aligned}$$

   The claim is obtained by joining both sides of the four inequalities.

7. By Lemmas 93.5 and 93.4 we obtain $\sqsupseteq ; (P \| Q) = (\sqsupseteq \| \sqsupseteq) ; (P \| Q) = (\sqsupseteq P \| \sqsupseteq Q) \subseteq (P \sqsupseteq \| Q \sqsupseteq) = (P \| Q) ; (\sqsupseteq \| \sqsupseteq) = (P \| Q) ; \sqsupseteq$.

8. If $C = \emptyset$, the claim follows since $\sqsupseteq ; \top \subseteq \top = \top ; \mathbb{I} \subseteq \top ; \sqsupseteq$. Otherwise, $\sqsupseteq ; (\bigcap C) \subseteq (\bigcap_{P \in C} \sqsupseteq ; P) \subseteq (\bigcap_{P \in C} P ; \sqsupseteq) = (\bigcap C) ; \sqsupseteq$ by Lemma 71.

   The counterexample is given by $P =_{\text{def}} \mathbb{1}$ and $Q =_{\text{def}} x{\leftarrow}\bot$ that satisfy $\mathscr{H}_E$, $\mathscr{H}_B$ and $\mathscr{H}_N$. By Lemma 55.4, we have $\mathbb{1} \cap x{\leftarrow}\bot = x{\preccurlyeq}x' \cap x'{=}\bot = x{\preccurlyeq}\bot \cap x'{=}\bot$. Then $(1, \bot) \in \sqsupseteq ; (x{\preccurlyeq}\bot \cap x'{=}\bot)$, but $(1, \bot) \notin (x{\preccurlyeq}\bot \cap x'{=}\bot) ; \sqsupseteq$, thus $\neg \mathscr{H}_N(P \cap Q)$.

9. Relations satisfying $\mathscr{H}_N$ are closed under suprema by part 2. The claim therefore follows by Lemma 88.

10. Relations satisfying $\mathscr{H}_E$, $\mathscr{H}_B$, $\mathscr{H}_T$ and $\mathscr{H}_N$ are closed under infima of chains by Lemmas 53.4, 69.3 and 75.3 and part 8. The claim therefore follows by Corollary 89. □

Theorem 82. *Let P be a term composed of constants satisfying $\mathscr{H}_E$, $\mathscr{H}_B$, $\mathscr{H}_T$ and $\mathscr{H}_N$ (in particular skip, (un)declaration and assignment), sequential and parallel composition, finite non-empty non-deterministic choice, conjunction of chains, conditional and greatest fixpoint. Let $\mathscr{T}_{\preccurlyeq}(b)$ and $\mathscr{T}_{\sqsubseteq}(b)$ for all conditions b that occur as $\_ \blacktriangleleft b \blacktriangleright \_$ in P. Let $\mathscr{T}_{\preccurlyeq}(\vec{e})$ and $\mathscr{T}_{\sqsubseteq}(\vec{e})$ for all expressions $\vec{e}$ that occur as $\_{\leftarrow}\vec{e}$ in P. Then $\mathscr{H}_N(P)$.*

Proof. A more general claim is the following: Let S be any subterm of P, then $\mathscr{H}_E(T)$, $\mathscr{H}_B(T)$, $\mathscr{H}_T(T)$ and $\mathscr{H}_N(T)$ for any T obtained by substituting in S any constant relations satisfying $\mathscr{H}_E$, $\mathscr{H}_B$, $\mathscr{H}_T$ and $\mathscr{H}_N$ for the free variables of S. Any T so obtained satisfies $\mathscr{H}_E(T)$ by Theorem 62, $\mathscr{H}_B(T)$ by Theorem 70 and $\mathscr{H}_T(T)$ by Theorem 76. The proof for $\mathscr{H}_N$ is also by induction and analogous to the proofs of Theorems 62, 70 and 76, this time using Lemma 81 for the individual cases. □

3.4.2 Absent strict parts

We now treat the case where relations have no strict parts. This again starts with an inductive argument that leads to a preliminary healthiness condition. It is expressed in relational terms and simplified, similarly to our procedure in Section 3.4.1.1. The situation concerning the new condition, however, deviates from the previous, and some strengthening is needed to be able to prove closure under sequential composition. In Section 3.4.2.2 we restate the condition in terms of partial orders again. We then show that the healthiness condition is satisfied by almost all of our programming constructs.

Again, the purpose of the following two sections is to show how the healthiness condition has been derived. If the reader prefers to skip this motivation, we suggest to continue with Definition 84.

3.4.2.1 Inductive derivation

We will derive ever more complex predicates stating requirements in case the computation has no strict parts. While stating (counter)examples we assume the same range $D = D_i$ for all $i \in 1..m$.

Case $m = 1$ and $n = 1$: Just as stated at the beginning of Section 3.4, the relation R has no strict part if $(\bot, \bot) \notin R$. We then must make sure that the value of x_1' does not depend on the value of x_1. In other words, any outcome x_1' must be an outcome for all x_1. We therefore have

$$(\bot, \bot) \notin R \Rightarrow \forall x_1, x_1' : (x_1, x_1') \in R \Rightarrow \forall \tilde{x}_1 : (\tilde{x}_1, x_1') \in R.$$

Case $m = 2$ and $n = 1$: The value of x_1' must, for a given value of x_2, not depend on the value of x_1 if $(\bot, x_2, \bot) \notin R$. We therefore have

$$\forall x_2 : (\bot, x_2, \bot) \notin R \Rightarrow \forall x_1, x_1' : (x_1, x_2, x_1') \in R \Rightarrow \forall \tilde{x}_1 : (\tilde{x}_1, x_2, x_1') \in R.$$

Similarly, the value of x_1' must, for a given value of x_1, not depend on the value of x_2 if $(x_1, \bot, \bot) \notin R$. We therefore have

$$\forall x_1 : (x_1, \bot, \bot) \notin R \Rightarrow \forall x_2, x_1' : (x_1, x_2, x_1') \in R \Rightarrow \forall \tilde{x}_2 : (x_1, \tilde{x}_2, x_1') \in R.$$

One may ask if it is justified to consider dependence on x_1 and x_2 separately, or if their combination must be treated as well. This means, we would also like to have

$$(\bot, \bot, \bot) \notin R \Rightarrow \forall x_1, x_2, x_1' : (x_1, x_2, x_1') \in R \Rightarrow \forall \tilde{x}_1, \tilde{x}_2 : (\tilde{x}_1, \tilde{x}_2, x_1') \in R.$$

And indeed this is a consequence of the first two predicates: Assume that $(\bot, \bot, \bot) \notin R$, then $\forall x_1, x_1' : (x_1, \bot, x_1') \in R \Rightarrow \forall \tilde{x}_1 : (\tilde{x}_1, \bot, x_1') \in R$ by the first predicate. The instance for $x_1'{=}\tilde{x}_1{=}\bot$ simplifies to $\forall x_1 : (x_1, \bot, \bot) \notin R$. Using the second predicate we therefore obtain $\forall x_1, x_2, x_1' : (x_1, x_2, x_1') \in R \Rightarrow \forall \tilde{x}_2 : (x_1, \tilde{x}_2, x_1') \in R$. By a symmetric argument, we have $\forall x_2 : (\bot, x_2, \bot) \notin R$ using the second predicate, hence we obtain by the first predicate $\forall x_1, x_2, x_1' : (x_1, x_2, x_1') \in R \Rightarrow \forall \tilde{x}_1 : (\tilde{x}_1, x_2, x_1') \in R$. Instantiating this by $x_2{=}\tilde{x}_2$ within the conclusion of the previous predicate yields the claimed consequence.

Furthermore, by choosing the total $R = \{(x_1, x_2, a) \mid x_1, x_2 \in D\} \cup \{(\bot, \bot, \bot), (a, a, \bot)\}$ for some $a{\neq}\bot$, one can see that the combined predicate is strictly weaker than the conjunction of the first two. We therefore should take those two predicates.

Case $n = 1$: By generalising the previous arguments, we obtain a predicate for arbitrary m:

$$\forall i \in 1..m : \forall \vec{x} : (\vec{x}_{i \mapsto \bot}, \bot) \notin R \Rightarrow \forall x_i, x_1' : (\vec{x}, x_1') \in R \Rightarrow \forall \tilde{x}_i : (\vec{x}_{i \mapsto \tilde{x}_i}, x_1') \in R.$$

Since x_i, $\tilde{x}_i$ and x_1' are not free in the antecedents, this may be written more compactly as

$$\forall i \in 1..m : \forall \vec{x} : \forall \tilde{x}_i : \forall x_1' : (\vec{x}, x_1') \in R \Rightarrow (\vec{x}_{i \mapsto \bot}, \bot) \in R \vee (\vec{x}_{i \mapsto \tilde{x}_i}, x_1') \in R.$$

Case $m = 1$ and $n = 2$: The value of x_1' must not depend on the value of x_1 if there is no x_2' such that $(\bot, \bot, x_2') \in R$. This means that if there is an outcome x_1' and x_2' for some input x_1, there must be an outcome with the same x_1' for any input. We therefore have

$$(\forall x_2' : (\bot, \bot, x_2') \notin R) \Rightarrow \forall x_1, x_1', x_2' : (x_1, x_1', x_2') \in R \Rightarrow \forall \tilde{x}_1 : \exists \tilde{x}_2' : (\tilde{x}_1, x_1', \tilde{x}_2') \in R.$$

Similarly, the value of x_2' must not depend on the value of x_1 if there is no x_1' such that $(\bot, x_1', \bot) \in R$. We therefore have

$$(\forall x_1' : (\bot, x_1', \bot) \notin R) \Rightarrow \forall x_1, x_1', x_2' : (x_1, x_1', x_2') \in R \Rightarrow \forall \tilde{x}_1 : \exists \tilde{x}_1' : (\tilde{x}_1, \tilde{x}_1', x_2') \in R.$$

Weakening the antecedent is possible, but will not be considered further.

Case $m = 1$: By generalising the previous arguments, we obtain a predicate for arbitrary n:

$$\forall j \in 1..n : (\forall \vec{x}_{\bar{j}}' : (\bot, \vec{x}'_{j \mapsto \bot}) \notin R) \Rightarrow \forall x_1 : \forall \vec{x}' : (x_1, \vec{x}') \in R \Rightarrow \forall \tilde{x}_1 : \exists \vec{\tilde{x}}_{\bar{j}}' : (\tilde{x}_1, \vec{\tilde{x}}'_{j \mapsto x_j'}) \in R.$$

Since x_1, $\vec{x}'$, $\tilde{x}_1$ and $\vec{\tilde{x}}_{\bar{j}}'$ are not free in the antecedents, this may be written more compactly as

$$\forall j \in 1..n : \forall x_1 : \forall \tilde{x}_1 : \forall \vec{x}' : \exists \vec{\tilde{x}}_{\bar{j}}' : \exists \vec{\hat{x}}_{\bar{j}}' : (x_1, \vec{x}') \in R \Rightarrow (\bot, \vec{\hat{x}}'_{j \mapsto \bot}) \in R \vee (\tilde{x}_1, \vec{\tilde{x}}'_{j \mapsto x_j'}) \in R.$$

The general case: Combining the previous arguments, we obtain a predicate for arbitrary m and n:

$$\begin{aligned} \forall i \in 1..m : \forall j \in 1..n : \forall \vec{x}_{\bar{i}} : (\forall \vec{x}_{\bar{j}}' : (\vec{x}_{i \mapsto \bot}, \vec{x}'_{j \mapsto \bot}) \notin R) \Rightarrow \\ \forall x_i : \forall \vec{x}' : (\vec{x}, \vec{x}') \in R \Rightarrow \forall \tilde{x}_i : \exists \vec{\tilde{x}}_{\bar{j}}' : (\vec{x}_{i \mapsto \tilde{x}_i}, \vec{\tilde{x}}'_{j \mapsto x_j'}) \in R. \end{aligned}$$

Intuitively, the first antecedent states that for $x_i{=}\bot$ there is no outcome such that $x_j'{=}\bot$. Then x_j' must not depend on x_i. This means that if there is an outcome $\vec{x}'$ for some value of x_i, there must be an outcome with the same value of x_j' for all values of x_i.

Eliminating the implications, the general predicate is equivalent to

$$\begin{aligned} \forall i \in 1..m : \forall j \in 1..n : \forall \vec{x} : \forall \vec{x}' : \forall \tilde{x}_i : \exists \vec{\tilde{x}}_{\bar{j}}' : \exists \vec{\hat{x}}_{\bar{j}}' : \\ (\vec{x}_{i \mapsto \bot}, \vec{\hat{x}}'_{j \mapsto \bot}) \in R \vee (\vec{x}, \vec{x}') \notin R \vee (\vec{x}_{i \mapsto \tilde{x}_i}, \vec{\tilde{x}}'_{j \mapsto x_j'}) \in R. \end{aligned}$$

Healthiness condition: We derive a relational healthiness condition for the general case. We start with

$$\begin{aligned} & \forall i \in 1..m : \forall j \in 1..n : \forall \vec{x} : \forall \vec{x}' : \forall \tilde{x}_i : \\ & \quad (\vec{x}, \vec{x}') \in R \Rightarrow (\exists \vec{\hat{x}}_{\bar{j}}' : (\vec{x}_{i \mapsto \bot}, \vec{\hat{x}}'_{j \mapsto \bot}) \in R) \vee (\exists \vec{\tilde{x}}_{\bar{j}}' : (\vec{x}_{i \mapsto \tilde{x}_i}, \vec{\tilde{x}}'_{j \mapsto x_j'}) \in R) \\ \Leftrightarrow\ & \text{«rearrangement of quantified variables»} \\ & \forall i \in 1..m : \forall j \in 1..n : \forall \vec{x}_{\bar{i}} : \forall \tilde{x}_i : \forall \vec{x}' : \\ & \quad (\exists x_i : (\vec{x}, \vec{x}') \in R) \Rightarrow (\exists \vec{\hat{x}}_{\bar{j}}' : (\vec{x}_{i \mapsto \bot}, \vec{\hat{x}}'_{j \mapsto \bot}) \in R) \vee (\exists \vec{\tilde{x}}_{\bar{j}}' : (\vec{x}_{i \mapsto \tilde{x}_i}, \vec{\tilde{x}}'_{j \mapsto x_j'}) \in R) \\ \Leftrightarrow\ & \text{«renaming } \tilde{x}_i \text{ to } x_i\text{»} \\ & \forall i \in 1..m : \forall j \in 1..n : \forall \vec{x} : \forall \vec{x}' : \\ & \quad (\exists x_i : (\vec{x}, \vec{x}') \in R) \Rightarrow (\exists \vec{\hat{x}}_{\bar{j}}' : (\vec{x}_{i \mapsto \bot}, \vec{\hat{x}}'_{j \mapsto \bot}) \in R) \vee (\exists \vec{\tilde{x}}_{\bar{j}}' : (\vec{x}, \vec{\tilde{x}}'_{j \mapsto x_j'}) \in R), \end{aligned}$$

from which we proceed as follows:

$$
\begin{array}{l}
\quad (\exists x_i : (\vec{x},\vec{x}') \in R) \Rightarrow (\exists \vec{\hat{x}}'_{\bar{j}} : (\vec{x}_{i\mapsto\bot}, \vec{\hat{x}}'_{j\mapsto\bot}) \in R) \vee (\exists \vec{\hat{x}}'_{\bar{j}} : (\vec{x}, \vec{\hat{x}}'_{j\mapsto x'_j}) \in R) \\
\Leftrightarrow \text{ «introduction of intermediate variables and rearrangement»} \\
\quad (\exists \vec{y} : (\exists x_i : \vec{x}{=}\vec{y}) \ \wedge \ (\vec{y},\vec{x}') \in R) \Rightarrow \\
\qquad (\exists \vec{z},\vec{w} : \vec{x}_{i\mapsto\bot}{=}\vec{z} \ \wedge \ (\vec{z},\vec{w}) \in R \ \wedge \ \exists \vec{\hat{x}}'_{\bar{j}} : \vec{w}{=}\vec{\hat{x}}'_{j\mapsto\bot}) \vee \\
\qquad (\exists \vec{v} : (\vec{x},\vec{v}) \in R \ \wedge \ \exists \vec{\hat{x}}'_{\bar{j}} : \vec{v}{=}\vec{\hat{x}}'_{j\mapsto x'_j}) \\
\Leftrightarrow \text{ «explicit relations and sequence equality»} \\
\quad (\exists \vec{y} : (\vec{x},\vec{y}) \in \{(\vec{x},\vec{x}') \mid \vec{x}_{\bar{i}}{=}\vec{x}'_{\bar{i}}\} \ \wedge \ (\vec{y},\vec{x}') \in R) \Rightarrow \\
\qquad (\exists \vec{z},\vec{w} : (\vec{x},\vec{z}) \in \{(\vec{x},\vec{x}') \mid \vec{x}_{\bar{i}}{=}\vec{x}'_{\bar{i}} \ \wedge \ \bot{=}x'_i\} \ \wedge \\
\qquad\qquad (\vec{z},\vec{w}) \in R \ \wedge \ (\vec{w},\vec{x}') \in \{(\vec{x},\vec{x}') \mid x_j{=}\bot\}) \vee \\
\qquad (\exists \vec{v} : (\vec{x},\vec{v}) \in R \ \wedge \ (\vec{v},\vec{x}') \in \{(\vec{x},\vec{x}') \mid x_j{=}x'_j\}) \\
\Leftrightarrow \text{ «assignment, disjunction and sequential composition»} \\
\quad (\vec{x},\vec{x}') \in \vec{x}_{\bar{i}}{=}\vec{x}'_{\bar{i}} \,;\, R \Rightarrow (\vec{x},\vec{x}') \in x_i{:=}\bot \,;\, R \,;\, x_j{=}\bot \ \cup \ R \,;\, x_j{=}x'_j.
\end{array}
$$

Therefore, switching to the relational form,

$$
\begin{array}{l}
\quad \forall \vec{x} : \forall \vec{x}' : (\exists x_i : (\vec{x},\vec{x}') \in R) \Rightarrow (\exists \vec{\hat{x}}'_{\bar{j}} : (\vec{x}_{i\mapsto\bot}, \vec{\hat{x}}'_{j\mapsto\bot}) \in R) \vee (\exists \vec{\hat{x}}'_{\bar{j}} : (\vec{x}, \vec{\hat{x}}'_{j\mapsto x'_j}) \in R) \\
\Leftrightarrow \text{ «definition of subset relation»} \\
\quad \vec{x}_{\bar{i}}{=}\vec{x}'_{\bar{i}} \,;\, R \ \subseteq \ x_i{:=}\bot \,;\, R \,;\, x_j{=}\bot \ \cup \ R \,;\, x_j{=}x'_j.
\end{array}
$$

Thus the preliminary healthiness condition about the absent strict parts of the relation R is

$$\forall i \in 1..m : \forall j \in 1..n : \vec{x}_{\bar{i}}{=}\vec{x}'_{\bar{i}} \,;\, R \ \subseteq \ x_i{:=}\bot \,;\, R \,;\, x_j{=}\bot \ \cup \ R \,;\, x_j{=}x'_j.$$

3.4.2.2 Order formulation

Using the order $\sqsubseteq$ introduced in Section 3.4.1.2, we can derive an algebraic representation also for the healthiness condition about the absent strict parts, see Lemma 83.6. It is proved to be stronger than the inductively derived result of Section 3.4.2.1 in the presence of $\mathscr{H}_N$. The intermediate steps generalise the index $i \in 1..m$ to a subset $I \subseteq 1..m$ and eliminate the quantified variable j in favour of relational operations. We are then able to apply the healthiness condition simultaneously for all j in a single step. The need for strengthening is discussed after the lemma.

Lemma 83. *The following statements satisfy* $(1 \Leftrightarrow 2 \Leftrightarrow 3 \Leftarrow 4 \Leftarrow 5 \Rightarrow 6)$*:*

1. $\forall i \in 1..m : \forall j \in 1..n : \vec{x}_{\bar{i}}{=}\vec{x}'_{\bar{i}} \,;\, P \ \subseteq \ x_i{:=}\bot \,;\, P \,;\, x_j{=}\bot \ \cup \ P \,;\, x_j{=}x'_j.$
2. $\forall I \subseteq 1..m : \forall j \in 1..n : \vec{x}_{\overline{I}}{=}\vec{x}'_{\overline{I}} \,;\, P \ \subseteq \ \vec{x}_I{:=}\bot \,;\, P \,;\, x_j{=}\bot \ \cup \ P \,;\, x_j{=}x'_j.$
3. $\forall I \subseteq 1..m : \vec{x}_{\overline{I}}{=}\vec{x}'_{\overline{I}} \,;\, P \ \subseteq \ \bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, (\bigcap_{j\in J} P \,;\, x_j{=}\bot) \ \cap \ (\bigcap_{j\in\overline{J}} P \,;\, x_j{=}x'_j).$
4. $\forall I \subseteq 1..m : \vec{x}_{\overline{I}}{=}\vec{x}'_{\overline{I}} \,;\, P \ \subseteq \ \bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, P \,;\, \vec{x}_J{=}\bot \ \cap \ P \,;\, \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}}.$
5. $\forall I \subseteq 1..m : \vec{x}_{\overline{I}}{=}\vec{x}'_{\overline{I}} \,;\, P \ \subseteq \ \bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, P \,;\, (\vec{x}_J{:=}\bot)^\smile \ \cap \ P \,;\, \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}}.$
6. $\sqsubseteq \,;\, P \ \subseteq \ P \,;\, \sqsubseteq.$

If $\mathscr{H}_N(P)$*, then* $(5 \Leftrightarrow 6)$*.*

Proof. The part $(1 \Leftarrow 2)$ is clear. The part $(1 \Rightarrow 2)$ is proved by induction on the size of I. The basis $I = \emptyset$ follows since $\vec{x}_{\overline{\emptyset}}{=}\vec{x}'_{\overline{\emptyset}} = \mathbb{I}$ and $P = P \,;\, \mathbb{I} \subseteq P \,;\, x_j{=}x'_j$. The basis $|I| = 1$ is

clear. For the induction step assume $i \notin I$ where $|I| \geq 1$, then

$$
\begin{array}{ll}
& \vec{x}_{\overline{I \cup \{i\}}} {=} \vec{x}'_{\overline{I \cup \{i\}}} \mathbin{;} P \\
= & \text{«sequential composition of equality»} \\
& \vec{x}_{\bar{i}} {=} \vec{x}'_{\bar{i}} \mathbin{;} \vec{x}_{\overline{I}} {=} \vec{x}'_{\overline{I}} \mathbin{;} P \\
\subseteq & \text{«induction hypothesis»} \\
& \vec{x}_{\bar{i}} {=} \vec{x}'_{\bar{i}} \mathbin{;} (\vec{x}_I {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup P \mathbin{;} x_j {=} x'_j) \\
= & \text{«}I \subseteq \bar{i} \text{ and relation algebra»} \\
& \vec{x}_I {:=} \bot \mathbin{;} \vec{x}_{\bar{i}} {=} \vec{x}'_{\bar{i}} \mathbin{;} P \mathbin{;} x_j {=} \bot \cup \vec{x}_{\bar{i}} {=} \vec{x}'_{\bar{i}} \mathbin{;} P \mathbin{;} x_j {=} x'_j \\
\subseteq & \text{«statement 1»} \\
& \vec{x}_I {:=} \bot \mathbin{;} (x_i {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup P \mathbin{;} x_j {=} x'_j) \mathbin{;} x_j {=} \bot \cup \\
& (x_i {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup P \mathbin{;} x_j {=} x'_j) \mathbin{;} x_j {=} x'_j \\
\subseteq & \text{«distribution and vector properties»} \\
& \vec{x}_I {:=} \bot \mathbin{;} x_i {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup \vec{x}_I {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} x'_j \mathbin{;} x_j {=} \bot \cup \\
& x_i {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup P \mathbin{;} x_j {=} x'_j \mathbin{;} x_j {=} x'_j \\
= & \text{«relation algebra»} \\
& \vec{x}_{I \cup \{i\}} {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup \vec{x}_I {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup x_i {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup P \mathbin{;} x_j {=} x'_j \\
= & \text{«see below»} \\
& \vec{x}_{I \cup \{i\}} {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup P \mathbin{;} x_j {=} x'_j.
\end{array}
$$

In the last step, we use $\vec{x}_A {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \subseteq \vec{x}_{A \cup B} {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot$ twice, once with $A = I$ and $B = \{i\}$ and once with $A = \{i\}$ and $B = I$. In both instances $|B| \leq |I|$, enabling the induction hypothesis for B. Hence this claim follows by

$$
\begin{array}{ll}
& \vec{x}_A {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \\
\subseteq & \vec{x}_A {:=} \bot \mathbin{;} \vec{x}_{\overline{B}} {=} \vec{x}'_{\overline{B}} \mathbin{;} P \mathbin{;} x_j {=} \bot \\
= & \vec{x}_A {:=} \bot \mathbin{;} \vec{x}_B {:=} \bot \mathbin{;} \vec{x}_{\overline{B}} {=} \vec{x}'_{\overline{B}} \mathbin{;} P \mathbin{;} x_j {=} \bot \\
\subseteq & \vec{x}_{A \cup B} {:=} \bot \mathbin{;} (\vec{x}_B {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup P \mathbin{;} x_j {=} x'_j) \mathbin{;} x_j {=} \bot \\
\subseteq & \vec{x}_{A \cup B} {:=} \bot \mathbin{;} \vec{x}_B {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup \vec{x}_{A \cup B} {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} x'_j \mathbin{;} x_j {=} \bot \\
= & \vec{x}_{A \cup B} {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot.
\end{array}
$$

The part $(2 \Leftrightarrow 3)$ follows by Lemma 78.1 and univalence of $\vec{x}_I {:=} \bot$ since

$$
\begin{array}{ll}
& \forall j \in 1..n : \vec{x}_{\overline{I}} {=} \vec{x}'_{\overline{I}} \mathbin{;} P \subseteq \vec{x}_I {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup P \mathbin{;} x_j {=} x'_j \\
\Leftrightarrow & \vec{x}_{\overline{I}} {=} \vec{x}'_{\overline{I}} \mathbin{;} P \subseteq \bigcap_{j \in 1..n} \vec{x}_I {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot \cup P \mathbin{;} x_j {=} x'_j \\
\Leftrightarrow & \vec{x}_{\overline{I}} {=} \vec{x}'_{\overline{I}} \mathbin{;} P \subseteq \bigcup_{J \subseteq 1..n} (\bigcap_{j \in J} \vec{x}_I {:=} \bot \mathbin{;} P \mathbin{;} x_j {=} \bot) \cap (\bigcap_{j \in \overline{J}} P \mathbin{;} x_j {=} x'_j) \\
\Leftrightarrow & \vec{x}_{\overline{I}} {=} \vec{x}'_{\overline{I}} \mathbin{;} P \subseteq \bigcup_{J \subseteq 1..n} \vec{x}_I {:=} \bot \mathbin{;} (\bigcap_{j \in J} P \mathbin{;} x_j {=} \bot) \cap (\bigcap_{j \in \overline{J}} P \mathbin{;} x_j {=} x'_j).
\end{array}
$$

The part $(3 \Leftarrow 4)$ is immediate from

$$
\begin{array}{l}
(\bigcap_{j \in J} P \mathbin{;} x_j {=} \bot) \supseteq P \mathbin{;} (\bigcap_{j \in J} x_j {=} \bot) = P \mathbin{;} \vec{x}_J {=} \bot, \\
(\bigcap_{j \in \overline{J}} P \mathbin{;} x_j {=} x'_j) \supseteq P \mathbin{;} (\bigcap_{j \in \overline{J}} x_j {=} x'_j) = P \mathbin{;} \vec{x}_{\overline{J}} {=} \vec{x}'_{\overline{J}}.
\end{array}
$$

The part $(4 \Leftarrow 5)$ follows since $(\vec{x}_J {:=} \bot)^\smile = (\vec{x}'_J {=} \bot \cap \vec{x}'_{\overline{J}} {=} \vec{x}_{\overline{J}})^\smile \subseteq (\vec{x}'_J {=} \bot)^\smile = \vec{x}_J {=} \bot$. The part $(5 \Rightarrow 6)$ follows using Lemma 77.1 in

$$
\begin{array}{ll}
& \sqsubseteq \mathbin{;} P = (\bigcup_{I \subseteq 1..m} \vec{x}_I {:=} \bot)^\smile \mathbin{;} P = (\bigcup_{I \subseteq 1..m} \vec{x}_I {=} \bot \cap \vec{x}'_{\overline{I}} {=} \vec{x}_{\overline{I}}) \mathbin{;} P \\
= & \bigcup_{I \subseteq 1..m} \vec{x}_I {=} \bot \cap \vec{x}'_{\overline{I}} {=} \vec{x}_{\overline{I}} \mathbin{;} P \\
\subseteq & \bigcup_{I \subseteq 1..m} \vec{x}_I {=} \bot \cap \bigcup_{J \subseteq 1..n} \vec{x}_I {:=} \bot \mathbin{;} P \mathbin{;} (\vec{x}_J {:=} \bot)^\smile \cap P \mathbin{;} \vec{x}'_{\overline{J}} {=} \vec{x}_{\overline{J}} \\
\subseteq & \bigcup_{I \subseteq 1..m} \bigcup_{J \subseteq 1..n} (\vec{x}_I {=} \bot \cap \vec{x}'_I {=} \bot \cap \vec{x}'_{\overline{I}} {=} \vec{x}_{\overline{I}}) \mathbin{;} P \mathbin{;} (\vec{x}_J {:=} \bot)^\smile \\
\subseteq & (\bigcup_{J \subseteq 1..n} \mathbb{I} \mathbin{;} P \mathbin{;} (\vec{x}_J {:=} \bot)^\smile) = P \mathbin{;} (\bigcup_{J \subseteq 1..n} \vec{x}_J {:=} \bot)^\smile = P \mathbin{;} \sqsubseteq.
\end{array}
$$

Finally, assume $\mathscr{H}_N(P)$ to show $(5 \Leftarrow 6)$. First, $\vec{x}_{\overline{I}} {=} \vec{x}'_{\overline{I}} \subseteq \vec{x}_I {:=} \bot \mathbin{;} \sqsubseteq$ since $\vec{x}_I {:=} \bot$ is a mapping and

$$
(\vec{x}_I {:=} \bot)^\smile \mathbin{;} \vec{x}_{\overline{I}} {=} \vec{x}'_{\overline{I}} = \vec{x}_I {=} \bot \cap \vec{x}_{\overline{I}} {=} \vec{x}'_{\overline{I}} \mathbin{;} \vec{x}_{\overline{I}} {=} \vec{x}'_{\overline{I}} \subseteq \vec{x}_I {\sqsubseteq} \vec{x}'_I \cap \vec{x}_{\overline{I}} {\sqsubseteq} \vec{x}'_{\overline{I}} = \sqsubseteq.
$$

Second, by the dual statement of Lemma 78.3,

$$\begin{array}{ll} & (\vec{x}_I{:=}\bot)^\smile \;=\; (\vec{x}'_I{=}\bot \cap \vec{x}'_{\overline{I}}{=}\vec{x}_{\overline{I}})^\smile \;=\; \vec{x}_I{=}\bot \cap \vec{x}'_{\overline{I}}{=}\vec{x}_{\overline{I}} \\ = & \vec{x}_I{=}\bot \cap \vec{x}'_{\overline{I}}{=}\vec{x}_{\overline{I}} \cap (\bigcup_{K\subseteq\overline{I}} \vec{x}_K{=}\bot \cap \bot{\notin}\vec{x}_{\overline{I}\setminus K}) \\ = & \bigcup_{K\subseteq\overline{I}} \vec{x}_I{=}\bot \cap \vec{x}_K{=}\bot \cap \vec{x}'_{\overline{I}}{=}\vec{x}_{\overline{I}} \cap \bot{\notin}\vec{x}_{\overline{I}\cap\overline{K}} \\ \subseteq & (\bigcup_{K\subseteq\overline{I}} \vec{x}_{I\cup K}{=}\bot \cap \vec{x}'_{\overline{I\cup K}}{=}\vec{x}_{\overline{I\cup K}} \cap \bot{\notin}\vec{x}_{\overline{I\cup K}}) \subseteq \bigcup_{J\subseteq 1..n} \vec{x}_J{=}\bot \cap \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \cap \bot{\notin}\vec{x}_{\overline{J}} \\ = & (\bigcup_{J\subseteq 1..n} (\vec{x}'_J{=}\bot \cap \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}})^\smile \cap \bot{\notin}\vec{x}_{\overline{J}}) = \bigcup_{J\subseteq 1..n} (\vec{x}_J{:=}\bot)^\smile \cap \bot{\notin}\vec{x}_{\overline{J}}, \end{array}$$

hence $\sqsubseteq \;=\; (\bigcup_{I\subseteq 1..m} (\vec{x}_I{:=}\bot)^\smile) \subseteq \bigcup_{J\subseteq 1..n} (\vec{x}_J{:=}\bot)^\smile \cap \bot{\notin}\vec{x}_{\overline{J}}$ by Lemma 77.1. Using both facts,

$$\begin{array}{ll} & \vec{x}'_{\overline{I}}{=}\vec{x}_{\overline{I}} \,;\, P \;\subseteq\; \vec{x}_I{:=}\bot \,;\, \sqsubseteq \,;\, P \;\subseteq\; \vec{x}_I{:=}\bot \,;\, P \,;\, \sqsubseteq \\ \subseteq & \vec{x}_I{:=}\bot \,;\, P \,;\, (\bigcup_{J\subseteq 1..n} (\vec{x}_J{:=}\bot)^\smile \cap \bot{\notin}\vec{x}_{\overline{J}}) \;=\; \bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, P \,;\, ((\vec{x}_J{:=}\bot)^\smile \cap \bot{\notin}\vec{x}_{\overline{J}}) \\ \subseteq & \bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, P \,;\, (\vec{x}_J{:=}\bot)^\smile \;\cap\; \sqsupseteq \,;\, P \,;\, (\vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \cap \bot{\notin}\vec{x}_{\overline{J}}) \\ \subseteq & \bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, P \,;\, (\vec{x}_J{:=}\bot)^\smile \;\cap\; P \,;\, \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}}, \end{array}$$

since the last step follows by

$$\begin{array}{ll} & \sqsupseteq \,;\, P \,;\, (\vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \cap \bot{\notin}\vec{x}_{\overline{J}}) \;\subseteq\; P \,;\, \sqsupseteq \,;\, (\vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \cap \bot{\notin}\vec{x}_{\overline{J}}) \\ \subseteq & P \,;\, (\vec{x}'_{\overline{J}}{\sqsubseteq}\vec{x}_{\overline{J}} \cap \bot{\notin}\vec{x}'_{\overline{J}}) \,;\, \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \;\subseteq\; P \,;\, \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \,;\, \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}} \;=\; P \,;\, \vec{x}'_{\overline{J}}{=}\vec{x}_{\overline{J}}. \end{array}$$ □

Remark. The preliminary healthiness condition in Lemma 83.3 does not yield the desired properties. This becomes apparent when we try to prove closure under sequential composition. An outline of a reasonable procedure is

$$\begin{array}{rl} \vec{x}_{\overline{I}}{=}\vec{x}'_{\overline{I}} \,;\, P \,;\, Q \;\subseteq & (\bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, (\bigcap_{j\in J} P \,;\, x_j{=}\bot) \cap (\bigcap_{j\in\overline{J}} P \,;\, x_j{=}x'_j)) \,;\, Q \\ \subseteq & \ldots \\ \subseteq & \bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, (\bigcap_{j\in J} P \,;\, Q \,;\, x_j{=}\bot) \cap (\bigcap_{j\in\overline{J}} P \,;\, Q \,;\, x_j{=}x'_j). \end{array}$$

The problem is that already the first step is an overestimate, so that any following inequalities generally do not hold. To see this, consider the instance $P =_{\mathrm{def}} x_1, x_2{\leftarrow}1, 2 \cup x_1, x_2{\leftarrow}3, \bot$ and $Q =_{\mathrm{def}} (x_1, x_2{\leftarrow}x_2, x_1 \blacktriangleleft x_1{=}1 \blacktriangleright \mathbb{1})$. Note that $P \;=\; x_1, x_2{:=}1, 2 \;\cup\; x_1, x_2{:=}3, \bot$ by Lemma 55.4 and $P \,;\, Q \;=\; x_1, x_2{:=}2, 1 \;\cup\; x_1, x_2{:=}3, \bot$. Choosing $J = \{2\}$, we have

$$\begin{array}{lll} ((5,5),(4,1)) & \in & (\bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, (\bigcap_{j\in J} P \,;\, x_j{=}\bot) \cap (\bigcap_{j\in\overline{J}} P \,;\, x_j{=}x'_j)) \,;\, Q, \\ ((5,5),(4,1)) & \notin & \bigcup_{J\subseteq 1..n} \vec{x}_I{:=}\bot \,;\, (\bigcap_{j\in J} P \,;\, Q \,;\, x_j{=}\bot) \cap (\bigcap_{j\in\overline{J}} P \,;\, Q \,;\, x_j{=}x'_j). \end{array}$$

This problem is caused because our example P consists of two parts that may be used independently in the two occurrences of P on the right hand side of the healthiness condition. The part $x_1, x_2{:=}3, \bot$ is used to satisfy $P \,;\, x_2{=}\bot$, indicating that only x'_1 is independent of $\vec{x}_I$. This way, the part $x_1, x_2{:=}1, 2$ is used in $P \,;\, x_j{=}x'_j$ to obtain $x_2{=}4$ before Q is post-composed. Obtaining $x_1{=}4$ is impossible after $P \,;\, Q$ if only the value of x_2 may be changed.

Such a problem did not arise with the healthiness condition $\mathscr{H}_N$ that contains only two occurrences of P, see Lemma 79.3. The current healthiness condition contains three and therefore is inherently more complex. To solve this problem, we more tightly couple both occurrences of P on the right hand side.

By making the right hand side of the healthiness condition smaller, we can hope to avoid the overestimate obtained above. It is manifest to try, and easy to obtain Lemma 83.4. For example, if P is univalent this step is an equivalent transformation. This is an improvement, but the above example still fails since no coupling has been established so far. The key now is to further strengthen $\vec{x}_I{:=}\bot \,;\, P \,;\, \vec{x}_J{=}\bot \cap P \,;\, \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}}$ to

$$\vec{x}_I{:=}\bot \,;\, P \,;\, (\vec{x}_J{=}\bot \cap \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}}) \cap P \,;\, \vec{x}_{\overline{J}}{=}\vec{x}'_{\overline{J}},$$

so that the values of $\vec{x}_{\overline{J}}$ must be the same in both parts of the intersection. Precisely this is done in Lemma 83.5. (End of remark)

We thus define the healthiness condition $\mathscr{H}_A$ concerning the absent strict parts of a relation P. Lemma 83 suggests to use the conjunction of $\mathscr{H}_A$ and $\mathscr{H}_N$ since it is equivalent to a stronger form of the inductively derived conditions. If P is a mapping, $\mathscr{H}_N(P) \Leftrightarrow \mathscr{H}_A(P)$ by the remark following Definition 56.

Definition 84.

$$\mathscr{H}_A(P) \quad \Leftrightarrow_{\text{def}} \quad \sqsubseteq ; P \subseteq P ; \sqsubseteq.$$

Remark. A general form of the conditions $\mathscr{H}_N$ and $\mathscr{H}_A$ appears in the literature, although in another context and not in relational form. Let $E : A \leftrightarrow A$ be a partial order and $R : A \leftrightarrow A$ a relation that satisfies $ER \subseteq RE$ and $E^\smile R \subseteq RE^\smile$. Then R is called an *isotone relation* [Wal84] and an *order preserving multifunction* [Smi71]. In both cases, the definition is given pointwise, requiring for all $(x_1, x_2) \in E$ that

- for each $y_1 \in R(x_1)$ there is a $y_2 \in R(x_2)$ such that $(y_1, y_2) \in E$, and
- for each $y_2 \in R(x_2)$ there is a $y_1 \in R(x_1)$ such that $(y_1, y_2) \in E$.

The investigation is concerned with the question whether A ordered by E satisfies the *relational fixed point property* [Sch03]. This is the case iff every total, isotone relation R has a fixed point $x \in A$ such that $x \in R(x)$. Such a study has the relations themselves, interpreted as orders, as its objects. This has to be contrasted with our effort, for example, to obtain fixpoints of isotone functions over relations.

Less coincidental is the observation that the two criteria stated above express precisely what constitutes the Egli-Milner order on powerdomains [Plo76, Sch86], more precisely those built from flat domains. One can interpret the conjunction of $\mathscr{H}_N$ and $\mathscr{H}_A$ as imposing the Egli-Milner order on the image sets of relations. This order is frequently used in semantics but in different ways and for a different purpose. For example, in [Bak76, BZ86] it orders relations, while in [HA76, BGW79, SS92] it orders domains of functional programming languages. All these sources use the Egli-Milner order to define the least fixpoint of functions. In our approach, however, fixpoints are ordered by the usual subset relation and the Egli-Milner order appears merely in the healthiness conditions $\mathscr{H}_N$ and $\mathscr{H}_A$ dealing with undefinedness. As a matter of fact the Egli-Milner order models erratic non-determinism or general correctness, while UTP's and our definitions model demonic non-determinism or total correctness. The difference is expounded in [Nel89, SS92] in more detail. A general correctness variant of UTP is explored in [Dun01].

The conditions $\mathscr{H}_N$ and $\mathscr{H}_A$ can also be seen as expressing an information preservation principle. In this interpretation $\sqsubseteq$ is the definedness information order and $\mathscr{H}_N$ and $\mathscr{H}_A$ convey definedness information. Corresponding healthiness conditions for the termination information order $\preccurlyeq$ are discussed in Section 3.6.4. This view fits well with the notion of *partiality* investigated in [Sch06]: 'A treatment of possibly partial availability of information may also be seen in descriptions of eager/data-driven evaluation as opposed to lazy/demand-driven evaluation.'[ibidem, page 213] (End of remark)

3.4.2.3 Closure

We finally show that our programming constructs satisfy $\mathscr{H}_A$ and therefore also the conjunction of $\mathscr{H}_N$ and $\mathscr{H}_A$.

Lemma 85. *The constructs given in Definition 60 satisfy/preserve $\mathscr{H}_A$ as follows.*

1. $\mathscr{H}_A(\mathbb{1})$.
2. $(\forall P \in S : \mathscr{H}_A(P)) \Rightarrow \mathscr{H}_A(\bigcup S)$.
3. $\mathscr{H}_A(P) \wedge \mathscr{H}_A(Q) \Rightarrow \mathscr{H}_A(P ; Q)$.

4. $\mathscr{T}_{\sqsubseteq}(\vec{e}) \Rightarrow \mathscr{H}_A(\vec{x}{\leftarrow}\vec{e})$.
5. $\mathscr{H}_A(\textbf{\textit{var}}\ \vec{x}_K) \wedge \mathscr{H}_A(\textbf{\textit{end}}\ \vec{x}_K)$.
6. $\mathscr{H}_A(P) \wedge \mathscr{H}_A(Q) \wedge \mathscr{T}_{\sqsubseteq}(b) \Rightarrow \mathscr{H}_A(P \blacktriangleleft b \blacktriangleright Q)$.
7. $\mathscr{H}_A(P) \wedge \mathscr{H}_A(Q) \Rightarrow \mathscr{H}_A(P \| Q)$.
8. *Let C be a chain, then* $(\forall P \in C : \mathscr{H}_R(P) \wedge \mathscr{H}_B(P) \wedge \mathscr{H}_A(P)) \Rightarrow \mathscr{H}_A(\bigcap C)$. *In general,* $\mathscr{H}_E(P) \wedge \mathscr{H}_B(P) \wedge \mathscr{H}_A(P) \wedge \mathscr{H}_E(Q) \wedge \mathscr{H}_B(Q) \wedge \mathscr{H}_A(Q) \not\Rightarrow \mathscr{H}_A(P \cap Q)$.
9. *Let f be isotone and $\mathscr{H}_A$-preserving, then* $\mathscr{H}_A(\mu f)$.
10. *Let f be isotone and $(\mathscr{H}_E \wedge \mathscr{H}_B \wedge \mathscr{H}_A)$-preserving, then* $\mathscr{H}_A(\nu f)$.

Proof.

1. The claim is equivalent to $\sqsubseteq\ ;\preccurlyeq\ \subseteq\ \preccurlyeq\ ;\sqsubseteq \Leftrightarrow \succcurlyeq\ ;\sqsupseteq\ \subseteq\ \sqsupseteq\ ;\succcurlyeq$, which follows using Lemmas 55.2 and 77.1 in
$$\begin{aligned}\succcurlyeq\ ;\sqsupseteq\ &=\ (\textstyle\bigcup_{I\subseteq 1..m} \vec{x}_I{:=}\infty)\ ;(\bigcup_{J\subseteq 1..m} \vec{x}_J{:=}\bot)\ =\ (\bigcup_{I,J\subseteq 1..m} \vec{x}_I{:=}\infty\ ;\vec{x}_J{:=}\bot)\\ &=\ (\textstyle\bigcup_{I,J\subseteq 1..m} \vec{x}_J{:=}\bot\ ;\vec{x}_{I\setminus J}{:=}\infty)\ =\ (\bigcup_{J\subseteq 1..m} \vec{x}_J{:=}\bot\ ;\bigcup_{I\subseteq 1..m} \vec{x}_{I\setminus J}{:=}\infty)\\ &\subseteq\ (\textstyle\bigcup_{J\subseteq 1..m} \vec{x}_J{:=}\bot\ ;\bigcup_{I\subseteq 1..m} \vec{x}_I{:=}\infty)\\ &=\ (\textstyle\bigcup_{J\subseteq 1..m} \vec{x}_J{:=}\bot)\ ;(\bigcup_{I\subseteq 1..m} \vec{x}_I{:=}\infty)\ =\ \sqsupseteq\ ;\succcurlyeq.\end{aligned}$$
2. $\sqsubseteq\ ;(\bigcup S) = (\bigcup_{P\in S} \sqsubseteq\ ;P) \subseteq (\bigcup_{P\in S} P\ ;\sqsubseteq) = (\bigcup S)\ ;\sqsubseteq$.
3. $\sqsubseteq\ ;P\ ;Q \subseteq P\ ;\sqsubseteq\ ;Q \subseteq P\ ;Q\ ;\sqsubseteq$.
4. In Lemma 81.4 we have proved $\mathscr{H}_N(\vec{x}{:=}\vec{e})$, which is equivalent to $\mathscr{H}_A(\vec{x}{:=}\vec{e})$ since $\vec{x}{:=}\vec{e}$ is a mapping. Therefore $\mathscr{H}_A(\mathbb{1}\ ;\vec{x}{:=}\vec{e}\ ;\mathbb{1})$ by parts 1 and 3.
5. We have $\mathscr{H}_A(\exists \vec{x}_K : \mathbb{I})$ and $\mathscr{H}_A(\exists \vec{x}'_K : \mathbb{I})$ by Lemmas 77.2 and 77.3, from which the claim follows again by parts 1 and 3.
6. We distinguish four cases, depending on the value of b. First,
$$b{=}\infty\ \cap\ \sqsubseteq\ ;(P \blacktriangleleft b \blacktriangleright Q)\ \subseteq\ b{=}\infty\ \subseteq\ b{=}\infty\ ;\sqsubseteq.$$
For the remaining three cases, let $c \in \{\infty, \mathit{true}, \mathit{false}\}$, then $\sqsubseteq\ ;b{\neq}c\ \subseteq\ b{\neq}c$ has been proved in Lemma 81.6. Moreover $\vec{x}'{=}\bot\ ;\sqsubseteq\ =\ \top\ ;(\vec{x}{=}\bot\ \cap\ \vec{x}{\sqsubseteq}\vec{x}')\ =\ \top\ ;\vec{x}{=}\bot\ =\ \top$. Therefore, finishing the remaining three cases,
$$\begin{aligned}b{=}\bot\ \cap\ \sqsubseteq\ ;(P \blacktriangleleft b \blacktriangleright Q)\ &\subseteq\ b{=}\bot\ \cap\ \vec{x}'{=}\bot\ ;\sqsubseteq,\\ b{=}\mathit{true}\ \cap\ \sqsubseteq\ ;(P \blacktriangleleft b \blacktriangleright Q)\ &\subseteq\ b{=}\mathit{true}\ \cap\ \sqsubseteq\ ;(b{\neq}\mathit{true}\ \cup\ P)\ \subseteq\ b{=}\mathit{true}\ \cap\ P\ ;\sqsubseteq,\\ b{=}\mathit{false}\ \cap\ \sqsubseteq\ ;(P \blacktriangleleft b \blacktriangleright Q)\ &\subseteq\ b{=}\mathit{false}\ \cap\ \sqsubseteq\ ;(b{\neq}\mathit{false}\ \cup\ Q)\ \subseteq\ b{=}\mathit{false}\ \cap\ Q\ ;\sqsubseteq.\end{aligned}$$
The claim is obtained by joining both sides of the four inequalities.
7. By Lemmas 93.5 and 93.4 we obtain $\sqsubseteq\ ;(P\|Q) = (\sqsubseteq\|\sqsubseteq)\ ;(P\|Q) = (\sqsubseteq P\|\sqsubseteq Q) \subseteq (P\sqsubseteq\|Q\sqsubseteq) = (P\|Q)\ ;(\sqsubseteq\|\sqsubseteq) = (P\|Q)\ ;\sqsubseteq$.
8. If $C = \emptyset$, the claim follows since $\sqsubseteq\ ;\top \subseteq \top = \top\ ;\mathbb{I} \subseteq \top\ ;\sqsubseteq$. Otherwise, $\sqsubseteq\ ;(\bigcap C) \subseteq (\bigcap_{P\in C} \sqsubseteq\ ;P) \subseteq (\bigcap_{P\in C} P\ ;\sqsubseteq) = (\bigcap C)\ ;\sqsubseteq$ by Lemma 71.
The counterexample is given by $P =_{\text{def}} \mathbb{1}$ and $Q =_{\text{def}} x{\leftarrow}1$ that satisfy $\mathscr{H}_E$, $\mathscr{H}_B$ and $\mathscr{H}_A$. By Lemma 55.4, we have $\mathbb{1} \cap x{\leftarrow}1\ =\ x{\preccurlyeq}x' \cap x'{=}1\ =\ x{\preccurlyeq}1 \cap x'{=}1$. Then $(\bot, 1) \in \sqsubseteq\ ;(x{\preccurlyeq}1 \cap x'{=}1)$, but $(\bot, 1) \notin (x{\preccurlyeq}1 \cap x'{=}1)\ ;\sqsubseteq$, thus $\neg\mathscr{H}_A(P \cap Q)$.
9. Relations satisfying $\mathscr{H}_A$ are closed under suprema by part 2. The claim therefore follows by Lemma 88.
10. Relations satisfying $\mathscr{H}_E$, $\mathscr{H}_B$ and $\mathscr{H}_A$ are closed under infima of chains by Lemmas 53.4 and 69.3 and part 8. The claim therefore follows by Corollary 89. □

Theorem 86. *Let P be a term composed of constants satisfying $\mathscr{H}_E$, $\mathscr{H}_B$ and $\mathscr{H}_A$ (in particular skip, (un)declaration and assignment), sequential and parallel composition, finite non-deterministic choice, conjunction of chains, conditional and greatest fixpoint. Let $\mathscr{T}_{\preccurlyeq}(b)$ and $\mathscr{T}_{\sqsubseteq}(b)$ for all conditions b that occur as $\_ \blacktriangleleft b \blacktriangleright \_$ in P. Let $\mathscr{T}_{\preccurlyeq}(\vec{e})$ and $\mathscr{T}_{\sqsubseteq}(\vec{e})$ for all expressions $\vec{e}$ that occur as $\_ \leftarrow \vec{e}$ in P. Then $\mathscr{H}_A(P)$.*

Proof. A more general claim is the following: Let S be any subterm of P, then $\mathscr{H}_E(T)$, $\mathscr{H}_B(T)$ and $\mathscr{H}_A(T)$ for any T obtained by substituting in S any constant relations satisfying $\mathscr{H}_E$, $\mathscr{H}_B$ and $\mathscr{H}_A$ for the free variables of S. Any T so obtained satisfies $\mathscr{H}_E(T)$ by Theorem 62 and $\mathscr{H}_B(T)$ by Theorem 70. The proof for $\mathscr{H}_A$ is also by induction and analogous to the proofs of Theorems 62 and 70, this time using Lemma 85 for the individual cases. □

3.5 Discussion

Let us recapitulate the results of this chapter and discuss the adequacy of the developed theory. To this end we argue that the programs of our framework compute the intended results, and do so in a non-strict way. We finally point out related work not mentioned in the previous sections.

The following table summarises the closure properties of the conditions investigated in this chapter. It lists for each condition $\mathscr{H}$ those constructs that are allowed in the construction of a relation or function R such that $\mathscr{H}(R)$ can be shown. The column $\exists$ refers to skip and (un)declaration, and the following columns refer to assignment, arbitrary constant relations, parallel composition, sequential composition, conditional, non-deterministic choice, conjunction, greatest and least fixpoints, in this sequence.

Theorem	$\exists$	$\leftarrow\vec{e}$	constant	$\parallel$	;	$\blacktriangleleft b \blacktriangleright$	$\bigcup$	$\bigcap$	ν	μ
61 : isotony	_	_	_	_	_	_	_	_	_	_
62 : $\mathscr{H}_R$	_	_	$\mathscr{H}_R$	_	_	_	_	_	_	_
62 : $\mathscr{H}_L$	_	_	$\mathscr{H}_L$	_	_	$\mathscr{T}_{\preccurlyeq}$	_	_	_	_
62 : $\mathscr{H}_E$	_	_	$\mathscr{H}_E$	_	_	$\mathscr{T}_{\preccurlyeq}$	_	_	_	_
70 : $\mathscr{H}_B$	_	$\mathscr{T}_{\preccurlyeq}$	$\mathscr{H}_{EB}$	_	_	$\mathscr{T}_{\preccurlyeq}$	$\cup$	_	_	$\times$
73 : continuity	_	$\mathscr{T}_{\preccurlyeq}$	$\mathscr{H}_{EB}$	_	_	$\mathscr{T}_{\preccurlyeq}$	$\cup$	_	_	$\times$
76 : $\mathscr{H}_T$	_	$\mathscr{T}_{\preccurlyeq}$	$\mathscr{H}_{EBT}$	_	_	$\mathscr{T}_{\preccurlyeq}$	$\uplus$	$\wr$	_	$\times$
82 : $\mathscr{H}_N$	_	$\mathscr{T}_{\preccurlyeq\sqsubseteq}$	$\mathscr{H}_{EBTN}$	_	_	$\mathscr{T}_{\preccurlyeq\sqsubseteq}$	$\uplus$	$\wr$	_	$\times$
86 : $\mathscr{H}_A$	_	$\mathscr{T}_{\preccurlyeq\sqsubseteq}$	$\mathscr{H}_{EBA}$	_	_	$\mathscr{T}_{\preccurlyeq\sqsubseteq}$	$\cup$	$\wr$	_	$\times$
96 : $\mathscr{H}_M$	_	$\mathscr{T}_{\preccurlyeq}$	$\mathscr{H}_{EBTM}$	_	_	$\mathscr{T}_{\preccurlyeq}$	$\uplus$	$\wr$	_	$\times$
97 : $\mathscr{H}_W$	_	_	$\mathscr{H}_L$	_	_	$\mathscr{T}_{\preccurlyeq}$	_	_	_	_

An entry _ means that construct is permitted unconditionally. An entry $\mathscr{T}_S$ means $\mathscr{T}_X(\vec{e})$ or $\mathscr{T}_X(b)$ must hold for all $X \in S$. An entry $\mathscr{H}_S$ means the constant must satisfy $\mathscr{H}_X$ for all $X \in S$. An entry $\cup$ means only finite choice is allowed, and $\uplus$ requires finite non-empty choice. An entry $\wr$ means only chains are allowed. Finally, an entry $\times$ means that construct is not permitted.

By Lemma 59.1 the alphabet extension is composed of skip, sequential and parallel composition and therefore is unconditionally allowed, too.

3.5.1 Adequacy

We thus obtain a theory similar to that of designs but modelling non-strict computations. In particular, the left and right unit laws $\mathscr{H}_L$ and $\mathscr{H}_R$ and the right zero law $\mathscr{H}_T$ correspond to the healthiness conditions H1–H4 of designs without the left zero law $\top ; R = \top$. As is the case with designs, all functions composed of programming constructs are continuous and all relations composed of programming constructs are boundedly non-deterministic. Additionally, they satisfy the healthiness conditions $\mathscr{H}_N$ and $\mathscr{H}_A$ modelling the dependence of variables.

To conclude that our programs compute what they are supposed to, let us describe how our theory resembles the theory of designs. To this end, we set up a connection between the constructs introduced in Definition 60 and the respective designs of UTP. Excluding the special values ∞ and $\bot$, let the ranges $D'_i =_{\text{def}} D_i \setminus \{\infty, \bot\}$ and $D'_I =_{\text{def}} \prod_{i \in I} D'_i$ model the *defined* input values. Define the relation $\sim$ between normal designs with components of type $D'_I \leftrightarrow D'_J$ and relations of type $D_I \leftrightarrow D_J$ by

$$(P \vdash Q) \sim R \quad \Leftrightarrow_{\text{def}} \quad \forall \vec{x} \in D'_I : P(\vec{x}) = \emptyset \vee Q(\vec{x}) = R(\vec{x}).$$

Intuitively, this states that for each *defined* input $\vec{x}$, the computation modelled by $P \vdash Q$ does not terminate or yields the same results as the computation modelled by R. It is possible to show that $\sim$ holds between the corresponding constants and is preserved by the corresponding operations. In particular, let $\mathcal{D}\vec{e}$ be the states where $\vec{e}$ is defined, then $(\mathcal{D}\vec{e} \vdash \vec{x}{:=}\vec{e}) \sim \vec{x}{\leftarrow}\vec{e}$. Furthermore $(P_1 \vdash Q_1) \sim R_1$ and $(P_2 \vdash Q_2) \sim R_2$ imply

$$\begin{gathered}(P_1 \vdash Q_1 \lhd b \rhd P_2 \vdash Q_2) \sim (R_1 \blacktriangleleft b \blacktriangleright R_2),\\ (P_1 \vdash Q_1)\,;\,(P_2 \vdash Q_2) \sim R_1\,;\,R_2,\\ (P_1 \vdash Q_1) \cup (P_2 \vdash Q_2) \sim R_1 \cup R_2.\end{gathered}$$

For continuous f and g such that $(P \vdash Q) \sim R \Rightarrow f(P \vdash Q) \sim g(R)$ we finally have $\nu f \sim \nu g$. It follows that our relations are at least as appropriate as the corresponding designs. Investigating the correspondence for *undefined* values is left as future work.

To conclude that we obtain non-strict computations, note that an undefined value has no effect if it is not used since, for example, $x_1{\leftarrow}\bot\,;\,x_1{\leftarrow}2 = x_1{\leftarrow}2$. A similar remark holds for non-terminating parts of programs since, for example, $\Top\,;\,\vec{x}{\leftarrow}\vec{c} = \vec{x}{\leftarrow}\vec{c}$ for any constant $\vec{c}$. These informal remarks can be made precise by setting up an operational semantics along the following lines. Intuitively, the execution proceeds backwards, evaluating only those parts actually needed to obtain the result of the computation. We outline the procedure for deterministic programs composed of assignments, conditionals, sequential compositions and greatest fixpoints. Let $R : D_I \leftrightarrow D_J$ be such a program, $\vec{s} \in D_I$ the initial values of the variables and $L \subseteq J$ the set of variables whose final values $\vec{x}'_L$ we are interested in.

If $L = \emptyset$, we are done. Otherwise, if R is a sequential composition, consider its last statement Q, that is, let $R = P\,;\,Q$ such that Q is not a sequential composition.

- If $Q = \vec{x}{\leftarrow}\vec{e}$, we first determine which variables K are needed to calculate $\vec{e}_L$. We then (recursively) calculate the final values $\vec{x}'_K$ of these variables for the preceding part P. If this succeeds, that is, it terminates and does not abort, we calculate $\vec{e}_L$ using these values.

- If $Q = (S \blacktriangleleft b \blacktriangleright T)$, we similarly proceed to obtain b, again calculating just the necessary values. If this succeeds, then either $b = \mathit{true}$ or $b = \mathit{false}$, and we continue recursively with $P\,;\,S$ or $P\,;\,T$, respectively.

- If $Q = \nu f$, we unfold Q to $f(\nu f)$ and continue recursively.

If R is not a sequential composition, the procedure is the same except that the initial values $\vec{s}$ are used instead of recursive calculations for the preceding part P. As an optimisation, repeated calculation of the value of a variable can be avoided by memoisation.

This brief account can be extended to the other programming constructs. Local variable (un)declaration and parallel composition are straight-forward to include. Non-determinism can be treated in either of two ways. The first is to allow non-deterministic transitions in the operational semantics, as in [NN92]. Then the fact that non-deterministic choice is demonic must be considered outside of the transition system. Alternatively, one can describe calculations of sets of values. A more detailed investigation of the operational semantics is again left as future work.

3.5.2 Related work

Connections to related work have been pointed out throughout this chapter. The following description of further approaches is primarily focused on the similarities and differences to the present work.

Undefinedness and non-termination are addressed by [WD96] using the Z notation. The former is represented by a distinguished element $\perp$ that is propagated through sequential composition and thus models strict computations. Termination is treated by pairs of predicates describing pre- and postconditions, similar to designs. A combination of both aspects is not examined. The Z notation itself does neither deal with undefinedness nor with termination issues [ISO02].

Instead of modelling non-strict computations in an imperative programming language, one can proceed the other way around and introduce state into a lazy functional programming language. A restricted form of state are variables which can be assigned only once as, for example, in [Jos86]. Mutable state is provided by the Haskell I/O monad used in the example at the beginning of this chapter. It has the property that all actions are forced, regardless of their contribution to the final result [Lau93, PJ03]. This is avoided using the more general *state transformers* [LPJ95], combining lazy evaluation with stateful computation. Since the base language is functional, the semantics is given in the λ-calculus passing around environments and states. For our imperative context this is less adequate as using relations. Non-determinism is not treated and there is no distinction between undefinedness and non-termination.

A multi-paradigm language that supports lazy functions, exception handling, mutable state and non-deterministic choice points is Oz [RH04]. That book gives a formal operational semantics of the kernel language which, however, does not cover non-determinism. According to the reduction rule for sequential composition, the execution of statements is forced similar to the Haskell I/O monad. Besides the mentioned concepts, Oz supports object-oriented, concurrent, distributed, constraint and logic programming. It could therefore serve well as an operational basis to complement and explore the approach of UTP.

The correspondence between relational semantics and weakest (liberal) preconditions is elaborated in [Nel89, Hes92, Mad96].

3.6 Appendix

In this section we discuss several issues that support the previous investigation but might have disturbed the stream of arguments had they been inserted before.

3.6.1 Fixpoint theorems

We show a generalisation of Tarski's fixpoint theorem frequently used in this chapter. Our proof is based on the following ancillary result. Call a partially ordered set P *chain-complete* iff every chain has a supremum in P.

Proposition 87. *Every isotone function on a chain-complete partially ordered set has a fixpoint.*

Proof. The statement is proved in [AB61, Theorem 2]; a more elegant proof appears in [Mar76, Theorem 9(i)]. The latter makes essential use of the following theorem of Nicolas Bourbaki:

> Let P be a non-empty, partially ordered set closed under suprema of non-empty chains, and $f : P \to P$ a function such that $x \leq f(x)$ for all $x \in P$. Then f has a fixpoint.

It is proved in [Bou49, Corollaire 1]; a more elegant proof, requiring closure under suprema of all chains, appears in [Wit51]. □

Lemma 88. *Let L be a complete lattice, $f : L \to L$ isotone, $P \subseteq L$ closed under f and suprema of chains. Then $\mu f \in P$.*

Proof. The least fixpoint μf exists by Tarski's fixpoint theorem [Tar55]. We first show that $A =_{\text{def}} \{x \mid x \in P \wedge x \leq \mu f\}$ is chain-complete. Let C be a chain in A, then $\sup C \in P$ by closure of P under suprema of chains and $\sup C \leq \mu f$ by the join property. It is essential that the previous statement includes the empty chain. We next show that f is a function on A. Let $x \in A$, then $x \in P$ and $x \leq \mu f$, hence $f(x) \in P$ since P is closed under f and $f(x) \leq f(\mu f) = \mu f$ by isotony and the fixpoint property, thus $f(x) \in A$. We finally conclude by Proposition 87 that f has a fixpoint $x^\circ \in A$, hence $x^\circ = \mu f$, and therefore $\mu f \in P$. □

Corollary 89. *Let L be a complete lattice, $f : L \to L$ isotone, $P \subseteq L$ closed under f and infima of chains. Then $\nu f \in P$.*

Proof. Apply Lemma 88 to the dual lattice of L. □

3.6.2 Isotony of expressions

In this section we provide an alternative interpretation of the conditions $\mathscr{T}_{\preccurlyeq}$ and $\mathscr{T}_{\sqsubseteq}$ that formalise isotony of expressions with respect to $\preccurlyeq$ and $\sqsubseteq$, respectively. This is particularly useful if the expressions are constructed according to a given syntax. For definiteness and whenever necessary, we use a simple language of terms built from constants, variables and functions. The new isotony conditions $\mathscr{T}_\infty$ and $\mathscr{T}_\perp$ are stated in the following definition.

Definition 90. Let $k \in \{\infty, \perp\}$, then

$$\begin{array}{lll} \mathscr{T}_k^I(e) & \Leftrightarrow_{\text{def}} & \forall \vec{x}_{\overline{I}} : e[k/\vec{x}_I] = k \ \vee \ \forall \vec{y}_I, \vec{z}_I : e[\vec{y}_I/\vec{x}_I] = e[\vec{z}_I/\vec{x}_I], \\ \mathscr{T}_k(e) & \Leftrightarrow_{\text{def}} & \forall I \subseteq 1..m : \mathscr{T}_k^I(e), \\ \mathscr{T}_k(\vec{e}) & \Leftrightarrow_{\text{def}} & \forall e \in \vec{e} : \mathscr{T}_k(e). \end{array}$$

The notation $e[\vec{y}_I/\vec{x}_I]$ designates the value of the expression e when the input variables $\vec{x}_I$ have the values $\vec{y}_I$, given certain values for the input variables $\vec{x}_{\overline{I}}$. That value can be calculated, for example, by substitution of $\vec{y}_I$ for the free occurrences of $\vec{x}_I$ in e provided the syntactic structure of e is available, as in Lemma 92. First of all we show that the new and the original isotony conditions coincide.

Lemma 91. *$\mathscr{T}_{\preccurlyeq}(\vec{e}) \Leftrightarrow \mathscr{T}_\infty(\vec{e})$ and $\mathscr{T}_{\sqsubseteq}(\vec{e}) \Leftrightarrow \mathscr{T}_\perp(\vec{e})$.*

Proof. We show the first claim; the second follows analogously by replacing $\preccurlyeq$ with $\sqsubseteq$ and ∞ with $\perp$. For the part ($\Leftarrow$), let $(\vec{w}, \vec{y}) \in \preccurlyeq \,;\, \vec{x}'{=}\vec{e}$, hence there is $\vec{x}$ such that $\vec{w} \preccurlyeq \vec{x}$ and $\vec{y}{=}\vec{e}$. Let $I \subseteq 1..m$ such that $\vec{w}_I{=}\infty \wedge \infty \notin \vec{w}_{\overline{I}}$, thus $\vec{w}_{\overline{I}}{=}\vec{x}_{\overline{I}}$. For any $e_i \in \vec{e}$, we have $\mathscr{T}_\infty(e_i)$, and therefore

- either $e_i[\infty/\vec{x}_I] = \infty$, in which case $e_i[\vec{w}/\vec{x}] = e_i[\vec{w}_I, \vec{w}_{\overline{I}}/\vec{x}_I, \vec{x}_{\overline{I}}] = e_i[\infty/\vec{x}_I] = \infty \preccurlyeq y_i$,
- or $\forall \vec{y}_I, \vec{z}_I : e_i[\vec{y}_I/\vec{x}_I] = e_i[\vec{z}_I/\vec{x}_I]$, thus $e_i[\vec{w}/\vec{x}] = e_i[\vec{w}_I/\vec{x}_I] = e_i[\vec{x}_I/\vec{x}_I] = e_i = y_i$.

In both cases $e_i[\vec{w}/\vec{x}] \preccurlyeq y_i$, hence $\vec{e}[\vec{w}/\vec{x}] \preccurlyeq \vec{y}$, which is equivalent to

$$(\exists \vec{v} : \vec{v}{=}\vec{e}[\vec{w}/\vec{x}] \ \wedge \ \vec{v} \preccurlyeq \vec{y}) \Leftrightarrow (\exists \vec{v} : (\vec{w}, \vec{v}) \in \vec{x}'{=}\vec{e} \ \wedge \ \vec{v} \preccurlyeq \vec{y}) \Leftrightarrow (\vec{w}, \vec{y}) \in \vec{x}'{=}\vec{e} \,;\, \preccurlyeq.$$

For the part ($\Rightarrow$), let $e_i \in \vec{e}$ and $I \subseteq 1..m$. To show $\mathscr{T}_\infty^I(e_i)$, let furthermore $\vec{x}_{\overline{I}}$ be given such that $e_i[\infty/\vec{x}_I] \neq \infty$. Let $\vec{y}_I$ be given; we show $e_i[\vec{y}_I/\vec{x}_I] = e_i[\infty/\vec{x}_I]$ from which the claim follows. To this end, define $\vec{w}$ and $\vec{y}_{\overline{I}}$ such that $\vec{w}_I{=}\infty \preccurlyeq \vec{y}_I$ and $\vec{w}_{\overline{I}}{=}\vec{x}_{\overline{I}}{=}\vec{y}_{\overline{I}}$, hence $\vec{w} \preccurlyeq \vec{y}$. Observe that $(\vec{y}, \vec{e}[\vec{y}/\vec{x}]) \in \vec{x}'{=}\vec{e}$, hence $(\vec{w}, \vec{e}[\vec{y}/\vec{x}]) \in \preccurlyeq \,;\, \vec{x}'{=}\vec{e} \subseteq \vec{x}'{=}\vec{e} \,;\, \preccurlyeq \,=\, \vec{e} \preccurlyeq \vec{x}'$. It follows that $\vec{e}[\vec{w}/\vec{x}] \preccurlyeq \vec{e}[\vec{y}/\vec{x}]$, and therefore $e_i[\infty/\vec{x}_I] = e_i[\vec{w}/\vec{x}] \preccurlyeq e_i[\vec{y}/\vec{x}] = e_i[\vec{y}_I/\vec{x}_I]$. Since $e_i[\infty/\vec{x}_I] \neq \infty$, we thus have $e_i[\infty/\vec{x}_I] = e_i[\vec{y}_I/\vec{x}_I]$. □

The next lemma shows that the isotony condition is satisfied by strict expressions. Recall that a function f is strict iff $\bot{\in}\vec{x} \Rightarrow f(\vec{x}) = \bot$, where $\bot$ denotes the undefined value. This suffices to talk about $\mathscr{T}_\bot$, but for the condition $\mathscr{T}_\infty$ strictness must be extended to the other special element ∞. We cannot analogously demand $\infty{\in}\vec{x} \Rightarrow f(\vec{x}) = \infty$, since this leads to a conflict for terms such as $f(\bot, \infty)$. We fix the interaction between $\bot$ and ∞ by adding the requirement $(\bot{\notin}\vec{x} \ \wedge \ \infty{\in}\vec{x}) \Rightarrow f(\vec{x}) = \infty$. This models a simultaneous evaluation of arguments, such that undefinedness takes precedence over non-termination. Note that this decision is not material: Other strategies, for example, evaluating the arguments sequentially from the left to the right, can be modelled as well and yield the same result.

Lemma 92. *Let e be an expression composed of constants, variables and strict functions. Then $\mathscr{T}_{\preccurlyeq}(e)$ and $\mathscr{T}_{\sqsubseteq}(e)$.*

Proof. The claim follows by Lemma 91 if we can show $\mathscr{T}_\bot(e)$ and $\mathscr{T}_\infty(e)$. Let $I \subseteq 1..m$. We prove $\mathscr{T}^I_\bot(e)$ and $\mathscr{T}^I_\infty(e)$ by induction on the structure of e.

- Let $e = c$ for some constant c, then $e[\vec{y}_I/\vec{x}_I] = c[\vec{y}_I/\vec{x}_I] = c = c[\vec{z}_I/\vec{x}_I] = e[\vec{z}_I/\vec{x}_I]$.
- Let $e = x_i$ for some variable x_i. If $i \in I$, then $e[\bot/\vec{x}_I] = x_i[\bot/\vec{x}_I] = \bot$ and $e[\infty/\vec{x}_I] = x_i[\infty/\vec{x}_I] = \infty$. If $i \notin I$, then $e[\vec{y}_I/\vec{x}_I] = x_i[\vec{y}_I/\vec{x}_I] = x_i = x_i[\vec{z}_I/\vec{x}_I] = e[\vec{z}_I/\vec{x}_I]$.
- Let $e = f(e_1, \ldots, e_n)$ for some strict function f, and assume $\mathscr{T}_\bot(e_i)$ and $\mathscr{T}_\infty(e_i)$ for all $i \in 1..n$ by the induction hypothesis. To show $\mathscr{T}^I_\bot(e)$, we distinguish two cases. If $e_i[\bot/\vec{x}_I] = \bot$ for some $i \in 1..n$, then

  $$e[\bot/\vec{x}_I] = f(e_1, \ldots, e_n)[\bot/\vec{x}_I] = f(e_1[\bot/\vec{x}_I], \ldots, e_n[\bot/\vec{x}_I]) = f(\ldots, \bot, \ldots) = \bot$$

  by strictness of f. If $e_i[\bot/\vec{x}_I] \neq \bot$ for all $i \in 1..n$, then $e_i[\vec{y}_I/\vec{x}_I] = e_i[\vec{z}_I/\vec{x}_I]$ for all $i \in 1..n$ by the induction hypothesis, and therefore

  $$\begin{aligned} e[\vec{y}_I/\vec{x}_I] &= f(e_1, \ldots, e_n)[\vec{y}_I/\vec{x}_I] = f(e_1[\vec{y}_I/\vec{x}_I], \ldots, e_n[\vec{y}_I/\vec{x}_I]) \\ &= f(e_1[\vec{z}_I/\vec{x}_I], \ldots, e_n[\vec{z}_I/\vec{x}_I]) = f(e_1, \ldots, e_n)[\vec{z}_I/\vec{x}_I] = e[\vec{z}_I/\vec{x}_I]. \end{aligned}$$

  To show $\mathscr{T}^I_\infty(e)$, we distinguish three cases. If $e_i[\infty/\vec{x}_I] = \bot$ for some $i \in 1..n$, then $e_i[\vec{y}_I/\vec{x}_I] = e_i[\infty/\vec{x}_I] = \bot$ for all $\vec{y}_I$ by the induction hypothesis, and therefore

  $$e[\vec{y}_I/\vec{x}_I] = f(\ldots, e_i, \ldots)[\vec{y}_I/\vec{x}_I] = f(\ldots, e_i[\vec{y}_I/\vec{x}_I], \ldots) = f(\ldots, \bot, \ldots) = \bot$$

  for all $\vec{y}_I$ by strictness of f. If $e_j[\infty/\vec{x}_I] \neq \bot$ for all $j \in 1..n$, and $e_i[\infty/\vec{x}_I] = \infty$ for some $i \in 1..n$, then

  $$e[\infty/\vec{x}_I] = f(e_1, \ldots, e_n)[\infty/\vec{x}_I] = f(e_1[\infty/\vec{x}_I], \ldots, e_n[\infty/\vec{x}_I]) = f(\ldots, \infty, \ldots) = \infty$$

  by strictness of f. If $e_i[\infty/\vec{x}_I] \notin \{\bot, \infty\}$ for all $i \in 1..n$, then $e_i[\vec{y}_I/\vec{x}_I] = e_i[\vec{z}_I/\vec{x}_I]$ for all $i \in 1..n$ by the induction hypothesis, and therefore, as above,

  $$\begin{aligned} e[\vec{y}_I/\vec{x}_I] &= f(e_1, \ldots, e_n)[\vec{y}_I/\vec{x}_I] = f(e_1[\vec{y}_I/\vec{x}_I], \ldots, e_n[\vec{y}_I/\vec{x}_I]) \\ &= f(e_1[\vec{z}_I/\vec{x}_I], \ldots, e_n[\vec{z}_I/\vec{x}_I]) = f(e_1, \ldots, e_n)[\vec{z}_I/\vec{x}_I] = e[\vec{z}_I/\vec{x}_I]. \end{aligned}$$

  □

Observe that isotony is more general than strictness in our context. For example, the non-strict conditional expression $e_1 \blacktriangleleft b \blacktriangleright e_2$ can be accommodated as well. Its value is usually obtained by

$$(e_1 \blacktriangleleft b \blacktriangleright e_2) =_{\text{def}} \begin{cases} \bot, \text{ if } b = \bot \\ \infty, \text{ if } b = \infty \\ e_1, \text{ if } b = \mathit{true} \\ e_2, \text{ if } b = \mathit{false} \end{cases}$$

It follows that the conditional $e_1 \blacktriangleleft b \blacktriangleright e_2$ is isotone provided b, e_1 and e_2 are isotone. We conclude that isotony of expressions is a condition not too severe.

3.6.3 Parallel composition

In this section we discuss a version of the parallel composition operator [BK97] suited to represent the alphabet extension. Parallel composition of relations 'models simultaneous evaluation without interaction'[ibidem, page 124] and in this spirit also appears as the basic operator of concurrency in [HH98], but the application to UTP's alphabet extension is new. Additional sources defining the operator are [Roe76, BBH+92]. It is called *cross* in fork algebra [FVB04], however we will not use that axiomatisation since it is for homogeneous relation algebras and for some results we need to argue pointwise anyway.

Recall that the parallel composition of the relations $P : D_I \leftrightarrow D_J$ and $Q : D_K \leftrightarrow D_L$ such that $I \cap K = \emptyset = J \cap L$ is

$$P \| Q = (\exists \vec{x}'_K : \mathbb{I}) \,;\, P \,;\, (\exists \vec{x}_L : \mathbb{I}) \cap (\exists \vec{x}'_I : \mathbb{I}) \,;\, Q \,;\, (\exists \vec{x}_J : \mathbb{I}) : D_{I \cup K} \leftrightarrow D_{J \cup L}.$$

Immediate consequences are isotony, distribution over $\cup$ and annihilation by $\bot\!\!\!\bot$. Further properties of $\|$ used in this chapter are stated in the following lemma.

Lemma 93.

1. $(P\|Q) \cap (R\|S) = P \cap R \| Q \cap S$.
2. $\overline{P\|\top} = \overline{P}\|\top$ *and* $\overline{\top\|Q} = \top\|\overline{Q}$ *and* $\overline{P\|Q} = (\overline{P}\|\top) \cup (\top\|\overline{Q})$ *and* $\top\|\top = \top$.
3. $(P\|Q) \cup (R\|S) = (P \cup R \| Q \cup S) \cap \overline{\overline{P}\|\overline{S}} \cap \overline{\overline{R}\|\overline{Q}}$.
4. $(P\|Q) \,;\, (R\|S) = PR\|QS$.
5. $\preccurlyeq = \preccurlyeq\|\preccurlyeq$ *and* $\prec = (\prec\|\preccurlyeq) \cup (\preccurlyeq\|\prec)$ *and analogously for other pointwise orders.*

Proof.

1. Since $\exists \vec{x}' : \mathbb{I}$ is univalent and $\exists \vec{x} : \mathbb{I}$ is injective we obtain

$$\begin{aligned} P \cap R \| Q \cap S &= (\exists \vec{x}'_K : \mathbb{I})(P \cap R)(\exists \vec{x}_L : \mathbb{I}) \cap (\exists \vec{x}'_I : \mathbb{I})(Q \cap S)(\exists \vec{x}_J : \mathbb{I}) \\ &= (\exists \vec{x}'_K : \mathbb{I})P(\exists \vec{x}_L : \mathbb{I}) \cap (\exists \vec{x}'_K : \mathbb{I})R(\exists \vec{x}_L : \mathbb{I}) \cap \\ &\quad (\exists \vec{x}'_I : \mathbb{I})Q(\exists \vec{x}_J : \mathbb{I}) \cap (\exists \vec{x}'_I : \mathbb{I})S(\exists \vec{x}_J : \mathbb{I}) \\ &= (P\|Q) \cap (R\|S). \end{aligned}$$

2. Since $\exists \vec{x}' : \mathbb{I}$ is a mapping and $\exists \vec{x} : \mathbb{I}$ is injective and surjective we obtain

$$\begin{aligned} \overline{P\|\top} &= \overline{(\exists \vec{x}'_K : \mathbb{I})P(\exists \vec{x}_L : \mathbb{I}) \cap (\exists \vec{x}'_I : \mathbb{I})\top(\exists \vec{x}_J : \mathbb{I})} = \overline{(\exists \vec{x}'_K : \mathbb{I})P(\exists \vec{x}_L : \mathbb{I})} \\ &= (\exists \vec{x}'_K : \mathbb{I})\overline{P}(\exists \vec{x}_L : \mathbb{I}) \cap (\exists \vec{x}'_I : \mathbb{I})\top(\exists \vec{x}_J : \mathbb{I}) = \overline{P}\|\top. \end{aligned}$$

   The proof of $\overline{\top\|Q} = \top\|\overline{Q}$ is symmetrical. By these two facts and part 1,

$$\overline{P\|Q} = \overline{(P\|\top) \cap (\top\|Q)} = \overline{P\|\top} \cup \overline{\top\|Q} = (\overline{P}\|\top) \cup (\top\|\overline{Q}).$$

   Finally, $\top\|\top = \overline{\overline{\top\|\top}} = \overline{\overline{\top}\|\overline{\top}} = \overline{\bot\|\top} = \overline{\bot} = \top$.

3. By parts 1 and 2,

$$\begin{aligned} (P\|Q) \cup (R\|S) &= ((P\|\top) \cap (\top\|Q)) \cup ((R\|\top) \cap (\top\|S)) \\ &= ((P\|\top) \cup (R\|\top)) \cap ((P\|\top) \cup (\top\|S)) \cap \\ &\quad ((\top\|Q) \cup (R\|\top)) \cap ((\top\|Q) \cup (\top\|S)) \\ &= (P \cup R\|\top) \cap \overline{P}\|\overline{S} \cap \overline{R}\|\overline{Q} \cap (\top\|Q \cup S) \\ &= (P \cup R\|Q \cup S) \cap \overline{\overline{P}\|\overline{S}} \cap \overline{\overline{R}\|\overline{Q}}. \end{aligned}$$

4. A statement of [SS89, page 174] implies that the part ($\supseteq$) cannot be derived by relation-algebraic means; we therefore proceed pointwise. Let $P : D_I \leftrightarrow D_J$ and $Q : D_K \leftrightarrow D_L$ and $R : D_J \leftrightarrow D_M$ and $S : D_L \leftrightarrow D_N$, then

$$\begin{array}{ll} & (\vec{x}_{I\cup K}, \vec{z}_{M\cup N}) \in PR\|QS \\ \Leftrightarrow & (\vec{x}_I, \vec{z}_M) \in PR \wedge (\vec{x}_K, \vec{z}_N) \in QS \\ \Leftrightarrow & (\exists \vec{y}_J : (\vec{x}_I, \vec{y}_J) \in P \wedge (\vec{y}_J, \vec{z}_M) \in R) \wedge (\exists \vec{y}_L : (\vec{x}_K, \vec{y}_L) \in Q \wedge (\vec{y}_L, \vec{z}_N) \in S) \\ \Leftrightarrow & \exists \vec{y}_{J\cup L} : (\vec{x}_I, \vec{y}_J) \in P \wedge (\vec{x}_K, \vec{y}_L) \in Q \wedge (\vec{y}_J, \vec{z}_M) \in R \wedge (\vec{y}_L, \vec{z}_N) \in S \\ \Leftrightarrow & \exists \vec{y}_{J\cup L} : (\vec{x}_{I\cup K}, \vec{y}_{J\cup L}) \in (P\|Q) \wedge (\vec{y}_{J\cup L}, \vec{z}_{M\cup N}) \in (R\|S) \\ \Leftrightarrow & (\vec{x}_{I\cup K}, \vec{z}_{M\cup N}) \in (P\|Q)(R\|S). \end{array}$$

5. First, we have $(\vec{x}_{I\cup K}, \vec{x}'_{I\cup K}) \in \preccurlyeq\|\preccurlyeq \Leftrightarrow \vec{x}_I \preccurlyeq \vec{x}'_I \wedge \vec{x}_K \preccurlyeq \vec{x}'_K \Leftrightarrow \vec{x}_{I\cup K} \preccurlyeq \vec{x}'_{I\cup K}$. We can analogously derive $\mathbb{I} = \mathbb{I}\|\mathbb{I}$. Together with parts 2 and 1 we obtain

$$\begin{array}{ll} \prec = \preccurlyeq \cap \bar{\mathbb{I}} = (\preccurlyeq\|\preccurlyeq) \cap \overline{\mathbb{I}\|\mathbb{I}} = (\preccurlyeq\|\preccurlyeq) \cap ((\bar{\mathbb{I}}\|\top) \cup (\top\|\bar{\mathbb{I}})) \\ \quad = ((\preccurlyeq\|\preccurlyeq) \cap (\bar{\mathbb{I}}\|\top)) \cup ((\preccurlyeq\|\preccurlyeq) \cap (\top\|\bar{\mathbb{I}})) = (\preccurlyeq \cap \bar{\mathbb{I}}\|\preccurlyeq) \cup (\preccurlyeq\|\preccurlyeq \cap \bar{\mathbb{I}}) \\ \quad = (\prec\|\preccurlyeq) \cup (\preccurlyeq\|\prec). \end{array}$$ □

3.6.4 Undefinedness and non-termination

In Section 3.4 we have derived the healthiness conditions $\mathscr{H}_N$ and $\mathscr{H}_A$ modelling dependence of undefinedness. This manifests itself in the use of $\sqsubseteq$ with its least element $\bot$ in the definitions of these healthiness conditions. It is legitimate to ask whether analogous conditions using $\preccurlyeq$ with its least element ∞ also hold. More generally, we should elaborate on the relationship between $\sqsubseteq$ and $\preccurlyeq$. Although these orders are structurally very similar and thus share several properties, there is an essential difference in their use. It is expressed by the healthiness conditions $\mathscr{H}_L$ and $\mathscr{H}_R$ enforcing closure with respect to $\preccurlyeq$. The reason why $\preccurlyeq$ is used for closure is the chosen model of the never terminating program: This relation should be both $\vec{x}{\leftarrow}\infty$ and the solution of the recursive equation $X = X$, that is, $\nu(X \mapsto X) = \top$. We thus obtain the requirement $\vec{x}{\leftarrow}\infty = \top$ which is satisfied by upper closure as Lemma 55.3 shows. A similar upper closure with respect to $\sqsubseteq$ is neither necessary nor advisable for this would again identify non-termination with undefinedness.

This explains why we use conditions of type $\mathscr{H}_L$ and $\mathscr{H}_R$ with respect to $\preccurlyeq$ but not $\sqsubseteq$. Let us return to the question of using $\mathscr{H}_N$- and $\mathscr{H}_A$-type conditions also with respect to $\preccurlyeq$. Such conditions can indeed be stated but we must take into account that relations are $\mathscr{H}_L$- and $\mathscr{H}_R$-closed. Otherwise, simply requiring $\succcurlyeq ; R \subseteq R ; \succcurlyeq$ does not work since already for $R = \mathbb{1} = \preccurlyeq$ we would obtain $\top = \succcurlyeq ; \preccurlyeq \subseteq \preccurlyeq ; \succcurlyeq$ which does not hold in general. Instead, we have to undo the effects of the upper closure and state conditions analogous to $\mathscr{H}_N$ and $\mathscr{H}_A$ using the minimal elements of the images as in Section 3.3.1. As announced at the end of that section, we use the relational formulation of min.

Definition 94. Let $\min P =_{\text{def}} P \cap \overline{P{\prec}}$, then

$$\begin{array}{lll} \mathscr{H}_W(P) & \Leftrightarrow_{\text{def}} & \preccurlyeq ; \min P \subseteq P ; \preccurlyeq, \\ \mathscr{H}_M(P) & \Leftrightarrow_{\text{def}} & \succcurlyeq ; \min P \subseteq P ; \succcurlyeq. \end{array}$$

It turns out that $\mathscr{H}_W$ is easily established, see Theorem 97. But first, we give the closure properties of $\mathscr{H}_M$ in the next lemma. Observe that $\mathscr{H}_R(P) \Rightarrow P = \min P ; \preccurlyeq$ as already remarked after Lemma 67.

Lemma 95. *The constructs given in Definition 60 satisfy/preserve $\mathscr{H}_M$ as follows.*

1. $\mathscr{H}_M(\mathbb{1}) \wedge \mathscr{H}_M(\textbf{\textit{var}}\ \vec{x}_K) \wedge \mathscr{H}_M(\textbf{\textit{end}}\ \vec{x}_K) \wedge (\mathscr{T}_{\preccurlyeq}(\vec{e}) \Rightarrow \mathscr{H}_M(\vec{x}{\leftarrow}\vec{e}))$.
2. $\mathscr{H}_R(P) \wedge \mathscr{H}_L(Q) \wedge \mathscr{H}_M(P) \wedge \mathscr{H}_M(Q) \Rightarrow \mathscr{H}_M(P ; Q)$.
3. $(\forall P \in S : \mathscr{H}_M(P)) \Rightarrow \mathscr{H}_M(\bigcup S)$.

4. *Let C be a chain, then* $(\forall P \in C : \mathscr{H}_R(P) \wedge \mathscr{H}_B(P) \wedge \mathscr{H}_M(P)) \Rightarrow \mathscr{H}_M(\bigcap C)$.
5. $\mathscr{H}_T(P) \wedge \mathscr{H}_T(Q) \wedge \mathscr{H}_M(P) \wedge \mathscr{H}_M(Q) \wedge \mathscr{T}_{\preccurlyeq}(b) \Rightarrow \mathscr{H}_M(P \blacktriangleleft b \blacktriangleright Q)$.
6. $\mathscr{H}_R(P) \wedge \mathscr{H}_R(Q) \wedge \mathscr{H}_M(P) \wedge \mathscr{H}_M(Q) \Rightarrow \mathscr{H}_M(P \| Q)$.
7. *Let f be isotone and $\mathscr{H}_M$-preserving, then* $\mathscr{H}_M(\mu f)$.
8. *Let f be isotone and $(\mathscr{H}_E \wedge \mathscr{H}_B \wedge \mathscr{H}_T \wedge \mathscr{H}_M)$-preserving, then* $\mathscr{H}_M(\nu f)$.

Proof.

1. We first show that $\min(\preccurlyeq P \preccurlyeq) = P$ provided P is univalent and $\preccurlyeq P \subseteq P \preccurlyeq$. The part $(\subseteq)$ follows from $\preccurlyeq P \preccurlyeq \subseteq P \preccurlyeq \preccurlyeq = P \preccurlyeq = P(\mathbb{I} \cup \prec) = P\mathbb{I} \cup P \prec = P \cup P \prec \subseteq P \cup \preccurlyeq P \prec$ by shunting, since $\min(\preccurlyeq P \preccurlyeq) = \preccurlyeq P \preccurlyeq \cap \overline{\preccurlyeq P \preccurlyeq \prec} = \preccurlyeq P \preccurlyeq \cap \overline{\preccurlyeq P \prec}$. The part $(\supseteq)$ follows from $P \cap \preccurlyeq P \prec \subseteq P \cap P \preccurlyeq \prec = P\mathbb{I} \cap P \prec = P(\mathbb{I} \cap \prec) = P(\mathbb{I} \cap \preccurlyeq \cap \overline{\mathbb{I}}) = P\bot\!\!\!\bot = \bot\!\!\!\bot$ again by shunting, since $P \subseteq \overline{\preccurlyeq P \prec}$ and $P \subseteq \preccurlyeq P \preccurlyeq$.

   If P is even a mapping, this implies $\mathscr{H}_M(\preccurlyeq P \preccurlyeq)$ as follows. Then $\preccurlyeq P \subseteq P \preccurlyeq$ is equivalent to $\succcurlyeq P \subseteq P \succcurlyeq$ and we obtain $\succcurlyeq ; \min(\preccurlyeq P \preccurlyeq) = \succcurlyeq P \subseteq P \succcurlyeq$.

   To instantiate these results, decompose $\mathbb{1} = \preccurlyeq \mathbb{I} \preccurlyeq$ and $\vec{x} \leftarrow \vec{e} = \preccurlyeq ; \vec{x} := \vec{e} ; \preccurlyeq$, as well as

$$\preccurlyeq ; (\exists \vec{x}_K : \vec{x}_K := \infty) ; \preccurlyeq \; = \; \preccurlyeq ; (\exists \vec{x}_K : \vec{x}_K := \infty ; \preccurlyeq) \; = \; \preccurlyeq ; (\exists \vec{x}_K : \preccurlyeq) \; = \; \boldsymbol{var}\, \vec{x}_K$$

   and $\boldsymbol{end}\, \vec{x}_K = \preccurlyeq ; (\exists \vec{x}'_K : \mathbb{I}) ; \preccurlyeq$ by Lemma 58. The claim now follows by observing that $\mathbb{I}$, $\vec{x} := \vec{e}$, $(\exists \vec{x}_K : \vec{x}_K := \infty)$ and $(\exists \vec{x}'_K : \mathbb{I})$ are mappings and satisfy $\preccurlyeq P \subseteq P \preccurlyeq$ by $\mathscr{T}_{\preccurlyeq}(\vec{e})$ and Lemma 58.

2. Using $\mathscr{H}_R(P)$, $\mathscr{H}_L(Q)$ and the Dedekind and Schröder laws, the claim follows by

$$\begin{array}{ll} & \succcurlyeq ; \min(PQ) \; = \; \succcurlyeq ; (PQ \cap \overline{PQ \prec}) \; = \; \succcurlyeq ; (\min P \; ; \preccurlyeq ; Q \cap \overline{PQ \prec}) \\ = & \succcurlyeq ; (\min P \; ; Q \cap \overline{PQ \prec}) \; \subseteq \; \succcurlyeq ; \min P \; ; (Q \cap (\min P)^{\smile} \; ; \overline{PQ \prec}) \\ \subseteq & \succcurlyeq ; \min P \; ; (Q \cap P^{\smile} \overline{PQ \prec}) \; \subseteq \; \succcurlyeq ; \min P \; ; (Q \cap \overline{Q \prec}) \\ = & \succcurlyeq ; \min P \; ; \min Q \; \subseteq \; P \; ; \succcurlyeq ; \min Q \; \subseteq \; PQ \succcurlyeq . \end{array}$$

3. The claim follows by

$$\begin{array}{ll} & \succcurlyeq ; \min(\bigcup S) \; = \; \succcurlyeq((\bigcup S) \cap \overline{(\bigcup S) \prec}) \; = \; \succcurlyeq(\bigcup_{P \in S} P \cap \overline{(\bigcup S) \prec}) \\ \subseteq & \succcurlyeq(\bigcup_{P \in S} P \cap \overline{P \prec}) \; = \; (\bigcup_{P \in S} \succcurlyeq ; \min P) \; \subseteq \; (\bigcup_{P \in S} P \succcurlyeq) \; = \; (\bigcup S) \succcurlyeq . \end{array}$$

4. For empty C, the claim holds since $\succcurlyeq ; \min \top \subseteq \top = \top\mathbb{I} \subseteq \top \succcurlyeq$. In the following, let $C \neq \emptyset$. We first show that $\succcurlyeq((\bigcap C) \cap \overline{P \prec}) \subseteq Q \succcurlyeq$ for all $P, Q \in C$:

   * either $P \subseteq Q$, then $\succcurlyeq((\bigcap C) \cap \overline{P \prec}) \subseteq \succcurlyeq(P \cap \overline{P \prec}) = \succcurlyeq ; \min P \subseteq P \succcurlyeq \subseteq Q \succcurlyeq$,
   * or $Q \subseteq P$, then $\succcurlyeq((\bigcap C) \cap \overline{P \prec}) \subseteq \succcurlyeq(Q \cap \overline{P \prec}) \subseteq \succcurlyeq(Q \cap \overline{Q \prec}) = \succcurlyeq ; \min Q \subseteq Q \succcurlyeq$.

   Using this in the sixth step and Lemma 71 in the second and last steps, we obtain

$$\begin{array}{ll} & \succcurlyeq ; \min(\bigcap C) \; = \; \succcurlyeq((\bigcap C) \cap \overline{(\bigcap C) \prec}) \; = \; \succcurlyeq((\bigcap C) \cap \overline{\bigcap_{P \in C} P \prec}) \\ = & \succcurlyeq((\bigcap C) \cap \bigcup_{P \in C} \overline{P \prec}) \; = \; \succcurlyeq(\bigcup_{P \in C} (\bigcap C) \cap \overline{P \prec}) \; = \; \bigcup_{P \in C} \succcurlyeq((\bigcap C) \cap \overline{P \prec}) \\ \subseteq & (\bigcap_{Q \in C} Q \succcurlyeq) \; = \; (\bigcap C) \succcurlyeq . \end{array}$$

5. We first show that $\min((V \cap P) \cup (W \cap Q)) = (V \cap \min P) \cup (W \cap \min Q)$ for vectors V and W such that $V \cap W = \bot\!\!\!\bot$. Using $V \subseteq \overline{W}$ and $W \subseteq \overline{V}$, this claim follows by

$$\begin{array}{ll} & \min((V \cap P) \cup (W \cap Q)) \; = \; ((V \cap P) \cup (W \cap Q)) \cap \overline{((V \cap P) \cup (W \cap Q)) \prec} \\ = & ((V \cap P) \cup (W \cap Q)) \cap \overline{(V \cap P) \prec} \cap \overline{(W \cap Q) \prec} \\ = & ((V \cap P) \cup (W \cap Q)) \cap (\overline{V} \cup \overline{P \prec}) \cap (\overline{W} \cup \overline{Q \prec}) \\ = & (V \cap P \cap \overline{P \prec} \cap (\overline{W} \cup \overline{Q \prec})) \cup (W \cap Q \cap (\overline{V} \cup \overline{P \prec}) \cap \overline{Q \prec}) \\ = & (V \cap P \cap \overline{P \prec}) \cup (W \cap Q \cap \overline{Q \prec}) \; = \; (V \cap \min P) \cup (W \cap \min Q) . \end{array}$$

Therefore,

$$\begin{aligned}
&\min(P \blacktriangleleft b \blacktriangleright Q)\\
=\ &\min(b{=}\infty \cup (b{=}\bot \cap \vec{x}'{=}\bot) \cup (b{=}\mathit{true} \cap P) \cup (b{=}\mathit{false} \cap Q))\\
=\ &(b{=}\infty \cap \min \top) \cup (b{=}\bot \cap \min \vec{x}'{=}\bot) \cup (b{=}\mathit{true} \cap \min P) \cup (b{=}\mathit{false} \cap \min Q)\\
=\ &(b{=}\infty \cap \vec{x}'{=}\infty) \cup (b{=}\bot \cap \vec{x}'{=}\bot) \cup (b{=}\mathit{true} \cap \min P) \cup (b{=}\mathit{false} \cap \min Q).
\end{aligned}$$

Moreover $\succcurlyeq\ ;\ b{=}c \subseteq b{=}c$ for any $c \in \{\bot, \mathit{true}, \mathit{false}\}$ has been proved in Lemma 57.4. For any total R we obtain $\succcurlyeq\ ;\ \vec{x}'{=}\infty \subseteq R\ ;\ \top\ ;\ \vec{x}'{=}\infty = R\ ;\ \vec{x}{:=}\infty \subseteq R\ ;\ \succcurlyeq$ by Lemma 55.2. Letting $R_\bot = \vec{x}'{=}\bot$ and $R_{\mathit{true}} = P$ and $R_{\mathit{false}} = Q$, we thus have

$$\begin{aligned}
&b{=}c \cap \succcurlyeq\ ;\ \min(P \blacktriangleleft b \blacktriangleright Q) = b{=}c \cap \succcurlyeq\ ;\ ((b{=}\infty \cap \vec{x}'{=}\infty) \cup (b{=}c \cap \min R_c))\\
\subseteq\ &b{=}c \cap (\succcurlyeq\ ;\ \vec{x}'{=}\infty \cup \succcurlyeq\ ;\ \min R_c) \subseteq b{=}c \cap R_c\ ;\ \succcurlyeq = (b{=}c \cap R_c)\ ;\ \succcurlyeq\\
\subseteq\ &(P \blacktriangleleft b \blacktriangleright Q)\ ;\ \succcurlyeq
\end{aligned}$$

since R_c is total and $\succcurlyeq\ ;\ \vec{x}'{=}\bot \subseteq \vec{x}'{=}\bot\ ;\ \succcurlyeq$ holds by $\succcurlyeq \subseteq \vec{x}'{=}\bot\ ;\ \succcurlyeq\ ;\ \vec{x}{=}\bot = \top$. Finally, $b{=}\infty \cap \succcurlyeq\ ;\ \min(P \blacktriangleleft b \blacktriangleright Q) \subseteq b{=}\infty \subseteq b{=}\infty\ ;\ \succcurlyeq \subseteq (P \blacktriangleleft b \blacktriangleright Q)\ ;\ \succcurlyeq$. The claim is obtained by joining the left sides of the four inequalities.

6. The following calculations use several properties of $\|$ stated in Lemma 93. First,

$$\begin{aligned}
\overline{(P\|Q)\prec} &= \overline{(P\|Q)((\prec\|\preccurlyeq) \cup (\preccurlyeq\|\prec))} = \overline{(P\|Q)(\prec\|\preccurlyeq) \cup (P\|Q)(\preccurlyeq\|\prec)}\\
&= \overline{(P\prec\|Q\preccurlyeq) \cup (P\preccurlyeq\|Q\prec)} = \overline{(P\prec \cup P\preccurlyeq\|Q\preccurlyeq \cup Q\prec) \cap \overline{\overline{P\prec}\|\overline{Q\prec}} \cap \overline{\overline{P\preccurlyeq}\|\overline{Q\preccurlyeq}}}\\
&= \overline{P\preccurlyeq\|Q\preccurlyeq} \cup (\overline{P\prec}\|\overline{Q\prec}) \cup (\overline{P\preccurlyeq}\|\overline{Q\preccurlyeq}) = \overline{P\|Q} \cup (\overline{P\prec}\|\overline{Q\prec}),
\end{aligned}$$

using $\mathscr{H}_R(P)$, $\mathscr{H}_R(Q)$ and $\prec \subseteq \preccurlyeq$. Therefore,

$$\begin{aligned}
\succcurlyeq\ ;\ \min(P\|Q) &= \succcurlyeq((P\|Q) \cap \overline{(P\|Q)\prec}) = \succcurlyeq((P\|Q) \cap (\overline{P\|Q} \cup (\overline{P\prec}\|\overline{Q\prec})))\\
&= \succcurlyeq((P\|Q) \cap (\overline{P\prec}\|\overline{Q\prec})) = \succcurlyeq(P \cap \overline{P\prec}\|Q \cap \overline{Q\prec})\\
&= (\succcurlyeq\|\succcurlyeq)(\min P\|\min Q) = (\succcurlyeq\ ;\ \min P\|\succcurlyeq\ ;\ \min Q)\\
&\subseteq (P\succcurlyeq\|Q\succcurlyeq) = (P\|Q)(\succcurlyeq\|\succcurlyeq) = (P\|Q)\succcurlyeq.
\end{aligned}$$

7. Relations satisfying $\mathscr{H}_M$ are closed under suprema by part 3. The claim therefore follows by Lemma 88.

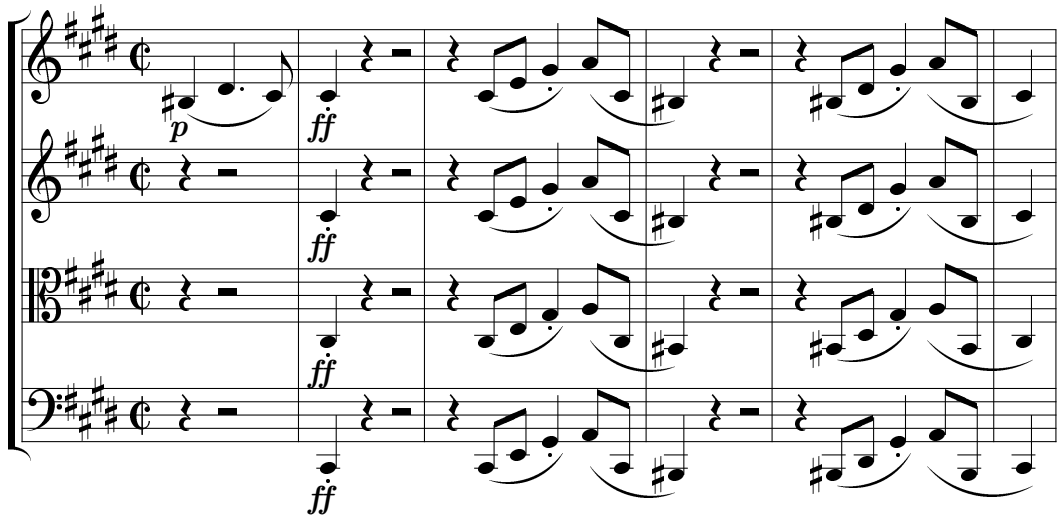
8. Relations satisfying $\mathscr{H}_E$, $\mathscr{H}_B$, $\mathscr{H}_T$ and $\mathscr{H}_M$ are closed under infima of chains by Lemmas 53.4, 69.3 and 75.3 and part 4. The claim therefore follows by Corollary 89. □

Theorem 96. *Let P be a term composed of constants satisfying $\mathscr{H}_E$, $\mathscr{H}_B$, $\mathscr{H}_T$ and $\mathscr{H}_M$ (in particular skip, (un)declaration and assignment), sequential and parallel composition, finite non-empty non-deterministic choice, conjunction of chains, conditional and greatest fixpoint. Let $\mathscr{T}_\preccurlyeq(b)$ for all conditions b that occur as $\_ \blacktriangleleft b \blacktriangleright \_$ in P. Let $\mathscr{T}_\preccurlyeq(\vec{e})$ for all expressions $\vec{e}$ that occur as $\_ \leftarrow \vec{e}$ in P. Then $\mathscr{H}_M(P)$.*

Proof. The proof is analogous to the proof of Theorem 82, this time using Lemma 95 for the individual cases. □

Theorem 97. *Let P be a term composed of constants satisfying $\mathscr{H}_L$ (in particular skip, (un)declaration and assignment), sequential and parallel composition, non-deterministic choice, conjunction, conditional, least and greatest fixpoint. Let $\mathscr{T}_\preccurlyeq(b)$ for all conditions b that occur as $\_ \blacktriangleleft b \blacktriangleright \_$ in P. Then $\mathscr{H}_W(P)$.*

Proof. Observe that $\mathscr{H}_L(P) \Rightarrow \mathscr{H}_W(P)$, since $\preccurlyeq\ ;\ \min P \subseteq \preccurlyeq\ ;\ P = P \subseteq P\ ;\ \preccurlyeq$. The claim therefore follows by Theorem 62. □


p
ff
ff
ff
ff

Chapter 4

Conclusion

Let us finally return to the two questions raised at the end of the introduction. Does the theory of designs need relations? The answer is 'no' as we have shown in Chapter 2. A semiring structure enriched by an ideal of Boolean conditions is sufficient to reconstruct the standard theory of UTP designs. Compared to UTP, this new approach has several benefits.

* Both the internal and external structure of designs is now completely transparent. Algebraic properties at both levels can thus be easily recognised and exploited. New operators providing further structure can simply be added.

* Designs are now consistently built on matrices and semirings rather than predicates. Existing results about these established and well-investigated mathematical structures can thus be directly reused to derive properties of designs. Links to further mathematical theories can be tied to export results from the context of UTP.

* The framework now concentrates on the essential features of designs. Calculations with designs therefore become more safe, simple and compact. Reasoning is entirely algebraic and thus yields clearly expressed results.

* The theory is now considerably more general by imposing fewer axioms. Unnecessary operations such as transposition and complement of general elements are removed. Connections to other theories of general and total correctness are established.

The disadvantage of having fewer axioms is of course that less specific properties can be shown. Concrete results we have obtained, both about the theory and in applying it, are summarised at the beginning of Chapter 2.

We now turn our attention to the second question. Does UTP need a left-absorbing loop? The answer is 'yes' if we stick with traditional designs and 'no' if we are willing to replace them by a new model of computations. As we have shown in Chapter 3 this question is closely related to undefined values and non-strictness. In particular we propose a new relational approach to define the semantics of imperative, non-deterministic programs. Compared to UTP, it has several benefits.

* Undefinedness and non-termination are now treated independently of each other. Finite and infinite failure can thus be distinguished which is closer to practice and allows one to model recovery from errors. A fine distinction is offered by dealing with undefinedness separately for individual variables.

* The theory now provides a relational model of dependence in computations. Additional healthiness conditions are stated in an algebraic form and can therefore be applied to new programs given as relations. Their compact formulation using orders is validated by an inductive derivation and verified by a deductive proof of closure under existing programming constructs.

* The framework is now extended by an operator for the parallel composition of relations. It is used to treat local variable declarations and alphabet extension adequately also in the context of non-termination. Relation algebra is used whenever possible for clear and concise arguments.

* The relations now model non-strict computations in an imperative context. Efficiency can thus be improved by executing only those parts of programs necessary to obtain the final results. The theory can serve as a basis to link to the semantics of functional programming languages.

The disadvantages of a possibly lazy evaluation are of course a potential overhead and reduced predictability of execution time, space and order. The findings of our investigation are summarised in Section 3.5.

Possible extensions and further research are mentioned in Sections 2.4 and 3.5.1. Let us point out a few more topics that deserve to be investigated. One of them concerns the implementation of the theory presented in Chapter 3. This involves a deeper study of the operational semantics and its connection to the relational model. It could be continued by an extension to a comprehensive programming and specification language, investigating the mutual effects of new constructs. Another thread is to explore the relational model as an intermediate for the translation of functional programming languages. The latter should be accompanied by comparing the semantics of lazy evaluation in both frameworks. A different domain is touched by applying the presented model of dependence in computations to develop optimising transformations used, for example, in compilers. Connections are anticipated to abstract interpretation and data flow analysis, where the partial availability of information also plays a role.

References

[Aar92] C. J. Aarts. Galois connections presented calculationally. Master's thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1992.

[AB61] S. Abian and A. B. Brown. A theorem on partially ordered sets, with applications to fixed point theorems. *Canadian Journal of Mathematics*, 13:78–82, 1961.

[ABB+95] C. J. Aarts, R. C. Backhouse, E. A. Boiten, H. Doornbos, N. van Gasteren, R. van Geldrop, P. F. Hoogendijk, E. Voermans, and J. van der Woude. Fixed-point calculus. *Information Processing Letters*, 53(3):131–136, 10 February 1995.

[AP86] K. R. Apt and G. D. Plotkin. Countable nondeterminism and random assignment. *Journal of the ACM*, 33(4):724–767, October 1986.

[Bak76] J. W. de Bakker. Semantics and termination of nondeterministic recursive programs. In S. Michaelson and R. Milner, editors, *Automata, Languages and Programming: Third International Colloquium*, pages 435–477. Edinburgh University Press, 1976.

[Bau76] F. L. Bauer. Programming as an evolutionary process. In *Proceedings of the 2nd International Conference on Software Engineering*, pages 223–234. IEEE Computer Society Press, 1976.

[BBH+92] R. C. Backhouse, P. J. de Bruin, P. Hoogendijk, G. Malcolm, E. Voermans, and J. van der Woude. Polynomial relators (extended abstract). In M. Nivat, C. Rattray, T. Rus, and G. Scollo, editors, *Algebraic Methodology and Software Technology*, pages 303–326. Springer-Verlag, 1992.

[BGW79] M. Broy, R. Gnatz, and M. Wirsing. Semantics of nondeterministic and noncontinuous constructs. In F. L. Bauer and M. Broy, editors, *Program Construction*, volume 69 of *Lecture Notes in Computer Science*, pages 553–592. Springer-Verlag, 1979.

[BK97] R. Berghammer and B. von Karger. Relational semantics of functional programs. In C. Brink, W. Kahl, and G. Schmidt, editors, *Relational Methods in Computer Science*, chapter 8, pages 115–130. Springer-Verlag, Wien, 1997.

[Bou49] N. Bourbaki. Sur le théorème de Zorn. *Archiv der Mathematik*, 2:434–437, 1949/1950.

[BZ86] R. Berghammer and H. Zierer. Relational algebraic semantics of deterministic and nondeterministic programs. *Theoretical Computer Science*, 43:123–147, 1986.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[Coh00] E. Cohen. Separation and reduction. In R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction*, volume 1837 of *Lecture Notes in Computer Science*, pages 45–59. Springer-Verlag, 2000.

[Con71] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[Dij76] E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall, 1976.

[DM01] J. Desharnais and B. Möller. Characterizing determinacy in Kleene algebras. *Information Sciences*, 139(3):253–273, December 2001.

[DMS06a] J. Desharnais, B. Möller, and G. Struth. Algebraic notions of termination. Report 2006-23, Institut für Informatik, Universität Augsburg, October 2006.

[DMS06b] J. Desharnais, B. Möller, and G. Struth. Kleene algebra with domain. *ACM Transactions on Computational Logic*, 7(4):798–833, October 2006.

[DMT06] J. Desharnais, B. Möller, and F. Tchier. Kleene under a modal demonic star. *Journal of Logic and Algebraic Programming*, 66(2):127–160, February–March 2006.

[DP02] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, second edition, 2002.

[Dun01] S. Dunne. Recasting Hoare and He's Unifying Theory of Programs in the context of general correctness. In A. Butterfield, G. Strong, and C. Pahl, editors, *5th Irish Workshop on Formal Methods*, Electronic Workshops in Computing. The British Computer Society, July 2001.

[ÉL05] Z. Ésik and H. Leiß. Algebraically complete semirings and Greibach normal form. *Annals of Pure and Applied Logic*, 3(1–3):173–203, May 2005.

[FVB04] M. F. Frias, P. A. S. Veloso, and G. A. Baum. Fork algebras: Past, present and future. *Journal on Relational Methods in Computer Science*, 1:181–216, 2004.

[GM06a] W. Guttmann and B. Möller. Modal design algebra. In S. Dunne and W. Stoddart, editors, *Unifying Theories of Programming*, volume 4010 of *Lecture Notes in Computer Science*, pages 236–256. Springer-Verlag, 2006.

[GM06b] W. Guttmann and B. Möller. Normal design algebra. Report 2006-28, Institut für Informatik, Universität Augsburg, December 2006.

[Gut06] W. Guttmann. Non-termination in Unifying Theories of Programming. In W. MacCaull, M. Winter, and I. Düntsch, editors, *Relational Methods in Computer Science 2005*, volume 3929 of *Lecture Notes in Computer Science*, pages 108–120. Springer-Verlag, 2006.

[HA76] M. Hennessy and E. A. Ashcroft. The semantics of nondeterminism. In S. Michaelson and R. Milner, editors, *Automata, Languages and Programming: Third International Colloquium*, pages 478–493. Edinburgh University Press, 1976.

[Hen77] F. W. von Henke. Formal transformations and the development of programs. In J. Gruska, editor, *Mathematical Foundations of Computer Science 1977*, volume 53 of *Lecture Notes in Computer Science*, pages 288–296. Springer-Verlag, 1977.

[Hes92] W. H. Hesselink. *Programs, Recursion and Unbounded Choice*. Cambridge University Press, 1992.

[HH98] C. A. R. Hoare and J. He. *Unifying theories of programming.* Prentice Hall Europe, 1998.

[HH01] C. A. R. Hoare and J. He. Unifying theories for logic programming. In C. A. R. Hoare, M. Broy, and R. Steinbrüggen, editors, *Engineering Theories of Software Construction*, pages 21–45. IOS Press, 2001.

[HLL06] J. He, X. Li, and Z. Liu. rCOS: A refinement calculus of object systems. *Theoretical Computer Science*, 365(1–2):109–142, 10 November 2006.

[HMS06] P. Höfner, B. Möller, and K. Solin. Omega algebra, demonic refinement algebra and commands. In R. Schmidt, editor, *Relations and Kleene Algebra in Computer Science*, volume 4136 of *Lecture Notes in Computer Science*, pages 222–234. Springer-Verlag, 2006.

[HMT71] L. Henkin, J. D. Monk, and A. Tarski. *Cylindric Algebras, Part I.* North-Holland Publishing Company, 1971.

[Hoa99a] C. A. R. Hoare. Theories of programming: Top-down and bottom-up and meeting in the middle. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM'99: Formal Methods*, volume 1708 of *Lecture Notes in Computer Science*, pages 1–27. Springer-Verlag, 1999.

[Hoa99b] C. A. R. Hoare. Theories of programming: Top-down and bottom-up and meeting in the middle. In E.-R. Olderog and B. Steffen, editors, *Correct System Design*, volume 1710 of *Lecture Notes in Computer Science*, pages 3–28. Springer-Verlag, 1999.

[HW93] U. Hebisch and H. J. Weinert. *Halbringe.* Teubner, 1993.

[ISO02] ISO/IEC. Information technology: Z formal specification notation: Syntax, type system and semantics. ISO/IEC 13568:2002(E), July 2002.

[Jos86] M. B. Joseph. Functional programming with side-effects. *Science of Computer Programming*, 7:279–296, 1986.

[Kle52] S. C. Kleene. *Introduction to Metamathematics.* North-Holland Publishing Company, 1952.

[Koz90] D. Kozen. On Kleene algebras and closed semirings. In B. Rovan, editor, *Mathematical Foundations of Computer Science 1990*, volume 452 of *Lecture Notes in Computer Science*, pages 26–47. Springer-Verlag, 1990.

[Koz94] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, May 1994.

[Koz97] D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, May 1997.

[Lau93] J. Launchbury. Lazy imperative programming. In P. Hudak, editor, *Proceedings of the ACM SIGPLAN Workshop on State in Programming Languages*, Yale University Research Report YALEU/DCS/RR-968, pages 46–56, June 1993.

[LHLL04] Z. Liu, J. He, J. Liu, and X. Li. Unifying views of UML. In F. de Boer and M. Bonsangue, editors, *Proceedings of the Workshop on the Compositional Verification of UML Models*, volume 101 of *Electronic Notes in Theoretical Computer Science*, pages 95–127. Elsevier B.V., 1 November 2004.

[LPJ95] J. Launchbury and S. Peyton Jones. State in Haskell. *Lisp and Symbolic Computation*, 8(4):293–341, December 1995.

[Mad96] R. D. Maddux. Relation-algebraic semantics. *Theoretical Computer Science*, 160(1–2):1–85, June 1996.

[Mai87] M. G. Main. A powerdomain primer. *Bulletin of the European Association for Theoretical Computer Science*, 33:115–147, October 1987. Also available as Technical Report CU-CS-375-87, University of Colorado at Boulder, September 1987.

[Mar76] G. Markowsky. Chain-complete posets and directed sets with applications. *Algebra Universalis*, 6(1):53–68, 1976.

[MD06] V. Mathieu and J. Desharnais. Verification of pushdown systems using omega algebra with domain. In W. MacCaull, M. Winter, and I. Düntsch, editors, *Relational Methods in Computer Science 2005*, volume 3929 of *Lecture Notes in Computer Science*, pages 188–199. Springer-Verlag, 2006.

[Möl93] B. Möller. Derivation of graph and pointer algorithms. In B. Möller, H. A. Partsch, and S. A. Schuman, editors, *Formal Program Development*, volume 755 of *Lecture Notes in Computer Science*, pages 123–160. Springer-Verlag, 1993.

[Möl04] B. Möller. Lazy Kleene algebra. In D. Kozen, editor, *Mathematics of Program Construction*, volume 3125 of *Lecture Notes in Computer Science*, pages 252–273. Springer-Verlag, 2004.

[Möl06] B. Möller. The linear algebra of UTP. In T. Uustalu, editor, *Mathematics of Program Construction*, volume 4014 of *Lecture Notes in Computer Science*, pages 338–358. Springer-Verlag, 2006.

[MS06] B. Möller and G. Struth. WP is WLP. In W. MacCaull, M. Winter, and I. Düntsch, editors, *Relational Methods in Computer Science 2005*, volume 3929 of *Lecture Notes in Computer Science*, pages 200–211. Springer-Verlag, 2006.

[Nel89] G. Nelson. A generalization of Dijkstra's calculus. *ACM Transactions on Programming Languages and Systems*, 11(4):517–561, October 1989.

[NN92] H. R. Nielson and F. Nielson. *Semantics with Applications: A Formal Introduction*. John Wiley & Sons, 1992.

[OCW07] M. Oliveira, A. Cavalcanti, and J. Woodcock. A denotational semantics for Circus. In B. Aichernig, E. Boiten, J. Derrick, and L. Groves, editors, *Proceedings of the 11th Refinement Workshop*, volume 187 of *Electronic Notes in Theoretical Computer Science*, pages 107–123. Elsevier B.V., 15 July 2007.

[Par90] H. A. Partsch. *Specification and Transformation of Programs: A Formal Approach to Software Development*. Springer-Verlag, 1990.

[PE93] R. Plasmeijer and M. van Eekelen. *Functional Programming and Parallel Graph Rewriting*. Addison-Wesley, 1993.

[PJ03] S. Peyton Jones, editor. *Haskell 98 Language and Libraries: The Revised Report*. Cambridge University Press, 2003.

[Plo76] G. D. Plotkin. A powerdomain construction. *SIAM Journal on Computing*, 5(3):452–487, September 1976.

[Pop94] S. Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.

[Red65] V. N. Red'ko. Some aspects of the theory of languages. *Cybernetics and Systems Analysis*, 1(4):15–26, July 1965. Translation of Редько В. Н., Некоторые вопросы теории языков, Кибернетика, С. 12–21, № 4, 1965.

[RH04] P. Van Roy and S. Haridi. *Concepts, Techniques, and Models of Computer Programming*. MIT Press, 2004.

[Roe76] W.-P. de Roever. *Recursive program schemes: semantics and proof theory*. Number 70 in Mathematical Centre Tracts. Mathematisch Centrum, Amsterdam, 1976.

[Sch86] D. A. Schmidt. *Denotational Semantics: A Methodology for Language Development*. William C. Brown Publishers, 1986.

[Sch03] B. S. W. Schröder. *Ordered Sets: An Introduction*. Birkhäuser, 2003.

[Sch06] G. Schmidt. Partiality I: Embedding relation algebras. *Journal of Logic and Algebraic Programming*, 66(2):212–238, February–March 2006.

[SHW97] G. Schmidt, C. Hattensperger, and M. Winter. Heterogeneous relation algebra. In C. Brink, W. Kahl, and G. Schmidt, editors, *Relational Methods in Computer Science*, chapter 3, pages 39–53. Springer-Verlag, Wien, 1997.

[Smi71] R. E. Smithson. Fixed points of order preserving multifunctions. *Proceedings of the American Mathematical Society*, 28(1):304–310, April 1971.

[SS89] G. Schmidt and T. Ströhlein. *Relationen und Graphen*. Springer-Verlag, 1989.

[SS92] H. Søndergaard and P. Sestoft. Non-determinism in functional languages. *The Computer Journal*, 35(5):514–523, October 1992.

[Tar41] A. Tarski. On the calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89, September 1941.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[Tur85] D. A. Turner. Miranda: A non-strict functional language with polymorphic types. In J.-P. Jouannaud, editor, *Functional Programming Languages and Computer Architecture*, volume 201 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1985.

[Wal84] J. W. Walker. Isotone relations and the fixed point property for posets. *Discrete Mathematics*, 48(2–3):275–288, February 1984.

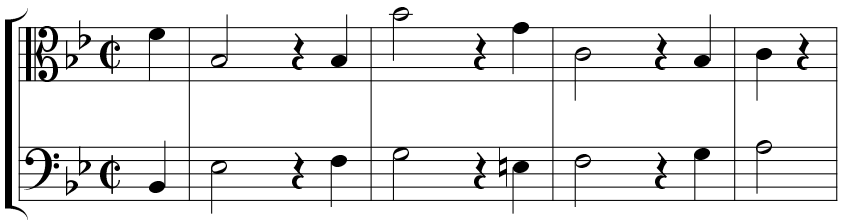
[WD96] J. Woodcock and J. Davies. *Using Z*. Prentice Hall, 1996.

[Wit51] E. Witt. Beweisstudien zum Satz von M. Zorn. *Mathematische Nachrichten*, 4:434–438, 1951.

[WM97a] M. Walicki and S. Meldal. Algebraic approaches to nondeterminism: An overview. *ACM Computing Surveys*, 29(1):30–81, March 1997.

[WM97b] R. Wilhelm and D. Maurer. *Übersetzerbau*. Springer-Verlag, second edition, 1997.

[Wri04] J. von Wright. Towards a refinement algebra. *Science of Computer Programming*, 51(1–2):23–45, May 2004.



Zusammenfassung

Die *Unifying Theories of Programming* von Hoare und He beschreiben die Semantik von Spezifikationen und Programmen unterschiedlicher Sprachparadigmen in einem gemeinsamen Formalismus. Im Mittelpunkt steht die Modellierung nichtdeterministischer, imperativer Programme durch spezielle Relationen, sogenannte *designs*, die Zustandsübergänge und Terminierungseigenschaften kombinieren. Sie bilden eine solide Grundlage für den Aufbau weiterer Theorien, schränken aber den Anwendungsbereich unnötig ein. Es werden nämlich einige Bedingungen, die sogenannten *healthiness conditions* auferlegt, die sämtliche Programme erfüllen müssen. Diese Arbeit beschäftigt sich mit der Frage was geschieht, wenn derartige Einschränkungen fallengelassen werden. Davon erhofft man sich eine weitergehende Anwendbarkeit der Theorie in vergleichbaren oder auf neuen Gebieten.

Die Ergebnisse dieser Arbeit sind die Bestimmung der Struktur, die den *designs* zugrundeliegt, und eine Erweiterung der Theorie um die Behandlung nichtstrikter Berechnungen. Um diese Ziele zu erreichen werden jene *healthiness conditions*, die unnötige Einschränkungen verursachen, aufgespürt und weggelassen. Insgesamt erhalten wir eine klare Darlegung der algebraischen Grundlagen der *Unifying Theories of Programming*.

Wir untersuchen zwei Arten von Einschränkungen, die beide die innere und äußere Struktur der *designs* betreffen. Unter der inneren Struktur verstehen wir die Bestandteile von *designs*, nämlich zwei Relationen, die gewisse Zustände und Zustandsübergänge beschreiben. Wie bei imperativen Programmen üblich ist dabei ein Zustand durch die Werte der vorkommenden Variablen festgelegt. Eine der Relationen beschreibt die Startzustände aus denen die Ausführung des Programms garantiert terminiert, während die andere die dort zugelassenen Zustandsübergänge wiedergibt. Die äußere Struktur bezieht sich auf die Regeln des Zusammenwirkens der *designs*, die im wesentlichen durch die oben erwähnten *healthiness conditions* vorgegeben sind.

Eine kurze Einführung in die zum Verständnis der Arbeit wichtigen Teile der *Unifying Theories of Programming* wird in Kapitel 1 gegeben. Unsere Untersuchung spielt sich in den Kapiteln 2 und 3 ab, auf die wir im folgenden näher eingehen. Kapitel 4 schließt die Arbeit indem es die wesentlichen Ergebnisse zusammenfaßt.

Kapitel 2 geht von der inneren Struktur der *designs* aus, die wie eben geschildert aus zwei Relationen besteht. Wir verallgemeinern diese Relationen indem wir sie durch Elemente von Halbringen und Halbringidealen ersetzen. Halbringe besitzen nur einen Teil der Struktur der Relationen, das heißt weniger Axiome. Insbesondere fehlt die Transpositionsoperation und die Komplementbildung auf beliebigen Halbringelementen, beides Operationen deren Anwendung auf Programme nicht möglich ist. Das Komplement bleibt hingegen auf jenen Teil der Elemente anwendbar, die aus dem Ideal stammen und Bedingungen repräsentieren.

Die Verallgemeinerung der inneren Struktur überträgt sich auf die äußere Struktur, das heißt die *designs* selbst sind nicht mehr Relationen sondern Elemente allgemeinerer Halbringe, siehe Theorem 7 und Folgeresultate. Die verringerten Eigenschaften reichen aber immer noch aus, um die Theorie der *designs* zu rekonstruieren, etwa die *healthiness conditions*. Ein Vorteil der Verallgemeinerung ist, daß man Verbindungen zu verwandten Ansätzen zur Semantikbeschreibung herstellen kann, etwa um diese miteinander zu vergleichen.

Durch die Offenlegung der tatsächlich benötigten Axiome wird die innere und äußere Struktur der *designs* klar dargestellt. Außerdem erkennt man algebraische Eigenschaften

besser und kann sie einfacher verwerten. Bei Bedarf kann man zusätzliche Struktur auferlegen indem man neue Operationen mit entsprechenden Axiomen einführt. Im weiteren Verlauf des Kapitels machen wir ausgiebig Gebrauch davon und beschreiben auf diese algebraische Art endliche und unendliche Iterationen, Definitionsbereich, Urbild, Determiniertheit, Invarianten und Terminierung. Ein konkretes Ergebnis ist, daß die *designs* eine Kleene-Algebra und eine Omega-Algebra bilden, siehe Abschnitte 2.1.3 und 2.1.4. Bei dem entsprechenden Beweis können wir dank der Darstellung von *designs* als Matrizen auf bekannte Aussagen aus der Mathematik zurückgreifen. Obwohl diese Wiederverwendung schon vorher möglich gewesen wäre, wurde sie erst durch die Klärung der Struktur erkennbar.

Die Kleene- und Omega-Algebren erlauben Aussagen etwa über die Äquivalenz verschieden strukturierter Rekursionen. Auf dieses Problem gehen wir detailliert in den Abschnitten 2.2 und 2.3 ein, wo wir algebraisch zeigen wie man eine symmetrische lineare Rekursion durch zwei aufeinanderfolgende Schleifen ersetzen kann. Dies geschieht durch getrennte Betrachtung der terminierenden und nichtterminierenden Zustände unter Verwendung der oben erwähnten Beschreibungsmittel. Derartige Beweise und Rechnungen werden durch die neue, abstrahierende Darstellung vereinfacht. Sie sind auch kompakter darstellbar und weniger fehleranfällig, da algebraisch, und sie haben klar formulierbare Ergebnisse.

Kapitel 3 geht von der Beobachtung aus, daß die *Unifying Theories of Programming* nicht vorsehen, Variablen und Ausdrücke mit undefiniertem Wert unabhängig von Nichtterminierung darzustellen. Eine derartige, feinere Unterscheidung ist zum Beispiel für das kontrollierte Auffangen von Berechnungen mit undefiniertem Ergebnis notwendig. Sie kann herbeigeführt werden indem man die innere Struktur von *designs* anpaßt, siehe Abschnitt 3.1.1. Möchte man nichtstrikte Berechnungen zulassen, also solche, die mit undefinierten Werten klarkommen, so führt eine solche Anpassung zu einem Konflikt mit der äußeren Struktur. Eine der *healthiness conditions* schreibt nämlich vor, daß die Relation, die eine nichtterminierende Berechnung beschreibt, linksabsorbierend bezüglich der Komposition ist. Der Hintergrund ist die Beobachtung, daß das Ausführen eines Programms „nach“ einer Endlosschleife keine Auswirkungen hat. Wir können daher keine *designs* mehr verwenden.

Stattdessen definieren wir in Abschnitt 3.2 neue Relationen, die nichtstrikte Berechnungen modellieren. Um darstellen zu können, daß einzelne Variablen undefiniert sind, werden ihre Wertebereiche um ein spezielles Element erweitert. Ein weiteres spezielles Element ermöglicht die Unterscheidung zwischen Undefiniertheit und Nichtterminierung, die auch beim Ablauf von Programmen beobachtbar ist. Die Schwierigkeit besteht darin, die Modelle so zu wählen, daß einerseits diese speziellen Werte bei ihrer Verwendung propagiert werden und andererseits die Programmkonstrukte stetig sind. Letzteres ist erforderlich, um die Lösungen von Rekursionsgleichungen iterativ zu berechnen, was dem tatsächlichen Ablauf rekursiver Programme entspricht. Wir lösen dieses Problem durch die Einführung einer partiellen Ordnung auf den Wertebereichen und dem beidseitigen Abschluß der Relationen bezüglich dieser Ordnung.

In Abschnitt 3.3 wird hergeleitet, daß die modellierten Berechnungen die gewünschten Stetigkeits- und Totalitätseigenschaften besitzen. Bei den *designs* ist Totalität durch eine der *healthiness conditions* gefordert; weitere zwei werden in Abschnitt 3.2.4 gezeigt. Die verbleibende *healthiness condition* kann, wie oben dargelegt, in unserem Modell nicht gelten. Insgesamt erhalten wir eine Theorie, die nichtstrikte Berechnungen in einem imperativen, nichtdeterministischen Kontext beschreibt. Das ermöglicht die Übertragung der aus funktionalen Programmiersprachen bekannten verzögerten Auswertung. Es ist nun ausreichend, die zur Berechnung der Endergebnisse notwendigen Programmteile auszuführen, was zu einer Verbesserung der Laufzeit führen kann. Eine entsprechende operationale Semantik wird in Abschnitt 3.5.1 kurz skizziert; eine Weiterführung der Forschung in diese Richtung ist sinnvoll.

Bei einer verzögerten Ausführung ist es notwendig, die Abhängigkeiten zwischen den einzelnen Berechnungen zu berücksichtigen. Solche Abhängigkeiten spielen unter anderem auch in optimierenden Programmtransformationen bei Übersetzern eine Rolle. Ihre Struktur wird in Abschnitt 3.4 untersucht. Ausgehend von der Beobachtung, daß nichtstrikte Berech-

nungen mit definierten Ergebnissen nicht von undefinierten Eingaben abhängen, können wir zwei zusätzliche *healthiness conditions* herleiten. Diese können auch auf Relationen, die neue Programmkonstrukte modellieren, angewendet werden. Wir entwickeln außerdem eine äquivalente, algebraisch elegante Form dieser Bedingungen unter Verwendung einer weiteren partiellen Ordnung. Sämtliche behandelten Programmkonstrukte erfüllen diese zusätzlichen *healthiness conditions*. Dies beinhaltet die parallele Komposition von Relationen, die wir in die *Unifying Theories of Programming* einführen, um eine algebraische Behandlung von lokalen Variablen zu ermöglichen.