Unifying Recursion in Partial, Total and General Correctness

Walter Guttmann

Department of Computer Science, University of Sheffield, UK
walter.guttmann@uni-ulm.de

Abstract. We give an algebraic semantics of non-deterministic, sequential programs which is valid for partial, total and general correctness. It covers full recursion based on a unified approximation order. We provide explicit solutions in terms of the refinement order. As an application, we systematically derive a semantics of while-programs common to the three correctness approaches.
UTP's designs and prescriptions represent programs as pairs of termination and state transition information in total and general correctness, respectively. We show that our unified semantics induces a pair-based representation which is common to the correctness approaches. Operations on the pairs, including finite and infinite iteration, can be derived systematically. We also provide the effect of full recursion on the unified, pair-based representation.

1 Introduction

In previous works [17, 18] we have identified common axioms which underly the approaches of partial, total and general correctness [24], and we have given a unified semantics of while-programs which is valid in all three correctness approaches. Results stated in terms of this semantics and proved by applying the common axioms hold in partial, total and general correctness. For example, this includes complex program transformations, such as those used to prove the normal form theorem for while-programs.

In this paper, we extend the unified semantics to cover full recursion. Fixpoints are taken with respect to a common approximation order, which is expressed in terms of the common refinement order and based on the Egli-Milner order typically used in general correctness. By adding axioms specific to partial, total and general correctness, respectively, we obtain the appropriate semantics of recursion in each particular approach. This covers in particular the sequential, non-deterministic fragment of UTP.

The unified recursion can be used to systematically derive a semantics of programming constructs which is common to all three correctness approaches. As an example, we calculate the unified semantics of while-programs. This improves the previous method of deriving or defining the semantics independently for partial, total and general correctness.

Theories of total and general correctness often represent a program as a pair whose components describe termination information and possible state transitions, respectively [5, 3, 30, 28, 19]. In UTP they emerge as designs for total correctness [22] and prescriptions for general correctness [14].

We show that our unified semantics induces such a pair-based representation, and systematically derive the operations and programming constructs on the pairs. Since it relies only on the common axioms, the same representation is valid for all correctness approaches. Informally speaking, this achieves a unification of UTP's designs and prescriptions.

We generalise previous investigations about the solution of recursions on the pair-based representations, which are specified by a pair of functions. This exemplifies that besides dealing with specific kinds of recursion, such as those necessary for while-programs, the unified recursion is also useful for general results.

Our approach is algebraic: laws of programs are taken as the axioms of algebraic structures with programs as elements. Thus our results hold in any model that satisfies the underlying axioms. In fact, this is essential for our unification as partial, total and general correctness have quite different models. Since the axioms are typically first-order conditional equations, another benefit is the support by automated theorem provers and counterexample generators.

Section 2 provides the axioms and some of their consequences. Most axioms are well investigated in the literature; the few of Section 2.3, including a new one, are specific to our aim of unification.

In Section 3 we contribute the unified approximation order, based on which we unify the semantics of recursion. Our main consequences are explicit representations of the unified fixpoint and explicit conditions for its existence. The application to while-loops is last.

The topic of Section 4 is to contribute a unified, pair-based representation of programs. Two more axioms are required and provided at the beginning. Based on them we map our algebraic structure to a structure of pairs. Operations on the pairs are derived via this isomorphism. We combine these results with the unified fixpoint representation of Section 3 to solve recursions on pairs.

To summarise, the contributions of this paper are to unify approximation, the semantics of recursion, representations of fixpoints, and pair-based representations of programs for partial, total and general correctness.

2 Axioms for Partial, Total and General Correctness

Partial-correctness approaches such as Hoare logic [21], weakest liberal preconditions [12] and Kleene algebra with tests [26] treat programs by ignoring their non-terminating executions. In total correctness, which includes weakest preconditions [12], UTP [22], demonic refinement algebra [31] and demonic algebra [6], terminating executions are ignored in the presence of any non-terminating ones starting from the same states. General correctness [2, 5, 24, 3, 30, 14, 28, 17] models terminating and non-terminating executions independently.

Although these approaches give programs different semantics, they support a number of common laws about programs. For example, these laws are sufficient to prove complex program transformations, which then hold in all three correctness approaches [18]. In this section we discuss some of the common axioms, which are subsequently used in this paper. We also mention laws characteristic for particular correctness approaches.

2.1 Basic Axioms

Throughout this paper we assume that programs are elements of an algebraic structure $(S, +, 0, \curlywedge, \top, \cdot, 1)$ such that $(S, +, 0, \curlywedge, \top)$ is a bounded distributive lattice and $(S, +, 0, \cdot, 1)$ is a semiring without the right zero law. We thus have the following axioms:

$$\begin{array}{rr} x + 0 = x & x \curlywedge \top = x \\ x + y = y + x & x \curlywedge y = y \curlywedge x \\ x + (y + z) = (x + y) + z & x \curlywedge (y \curlywedge z) = (x \curlywedge y) \curlywedge z \\ x \curlywedge (x + y) = x & x \curlywedge (y + z) = (x \curlywedge y) + (x \curlywedge z) \\ 1 \cdot x = x & x \cdot (y + z) = (x \cdot y) + (x \cdot z) \\ x \cdot 1 = x & (x + y) \cdot z = (x \cdot z) + (y \cdot z) \\ x \cdot (y \cdot z) = (x \cdot y) \cdot z & 0 \cdot x = 0 \end{array}$$

They characterise a lattice-ordered monoid [4] in which the lattice is bounded and distributive and the left zero law $0 \cdot x = 0$ holds. In contrast to our previous works [17, 18] we now include the $\curlywedge$ operation, which we use to represent fixpoints in Section 3.2 and programs as pairs in Section 4.2. The operation $\cdot$ has highest precedence; it is frequently omitted by writing xy instead of $x \cdot y$.

By $x \leq y \Leftrightarrow_{\text{def}} x + y = y \Leftrightarrow x \curlywedge y = x$ we obtain the partial order $\leq$ on S with join $+$, meet $\curlywedge$, least element 0 and greatest element $\top$. The operations $+$, $\curlywedge$ and $\cdot$ are $\leq$-isotone. Further consequences of the above axioms are

$$\begin{array}{ll} x + x = x & x + (x \curlywedge y) = x \\ x \curlywedge x = x & x + (y \curlywedge z) = (x + y) \curlywedge (x + z) \end{array}$$

In relational models such as UTP, the operation $+$ is interpreted as set union (non-deterministic choice), $\curlywedge$ as set intersection, and $\cdot$ as relational (sequential) composition. The relation $\leq$ is the subset or refinement order, where $x \leq y$ expresses that x refines y. In partial correctness, the constants 0, $\top$ and 1 are the empty, universal and identity relation, respectively. In terms of UTP designs, 0 is $(\mathit{true} \vdash \mathit{false})$, $\top$ is $(\mathit{false} \vdash \mathit{true})$, and 1 is $(\mathit{true} \vdash \vec{v}' = \vec{v})$; prescriptions have similar instances. Without further notice, we assume that designs and prescriptions satisfy the healthiness condition H3; these are called 'normal' by [14].

2.2 Domain

The domain $d(x)$ of the program x represents the initial states from which a transition under x is possible. Its complement, the anti-domain $a(x)$, is used in

a compact axiomatisation given in [11, 10], which we adopt:

$$\begin{aligned} d(x) &= a(a(x)) \\ a(x)x &= 0 \\ a(xd(y)) &= a(xy) \\ d(x) + a(x) &= 1 \end{aligned}$$

In our bounded setting the characterisation $d(x) \leq d(y) \Leftrightarrow x \leq d(y)\top$ given by [1] follows. The domain operation is idempotent and $\leq$-isotone, whence the domain elements $d(S)$ are the fixpoints of d. They form a Boolean algebra $(d(S), +, 0, \cdot, 1, a)$ [11, 10], in which the operations $\cdot$ and $\curlywedge$ coincide. Further consequences are

$$\begin{array}{lll} d(0) = 0 & d(xd(y)) = d(xy) & x \leq d(x)\top \\ d(\top) = 1 & d(d(x)y) = d(x)d(y) & x\top \leq d(x)\top \\ d(x)x = x & d(x+y) = d(x) + d(y) & xd(y)\top \leq d(xy)\top \end{array}$$

In UTP and other relational models of total correctness, the domain operation can be defined explicitly by $d(x) = x\top \curlywedge 1$, but this is not valid in general correctness. The domain of a design is reduced to that of its components by $d(P \vdash Q) = (\mathit{true} \vdash a(P) \vee d(Q))$, and similarly for prescriptions.

Domain elements can be used as tests to model conditions: for example, the sequential composition px of $p \in d(S)$ with the program $x \in S$ restricts the transitions of x to those starting in a state that satisfies p.

2.3 Loop

Notably missing from our axioms for S is the right zero law $x \cdot 0 = 0$, or equivalently, $\top \cdot 0 = 0$. This law is characteristic of partial correctness, but it does not hold in total and general correctness. In fact, $\top \cdot 0 = \top$ in total correctness, which in UTP is a consequence of the healthiness condition H1 for designs [22, Theorem 3.2.2]. In general correctness, the element $\top \cdot 0$ is neither 0 nor $\top$: in terms of prescriptions we obtain $(\mathit{false} \Vdash \mathit{true}) \cdot (\mathit{true} \Vdash \mathit{false}) = (\mathit{false} \Vdash \mathit{false})$.

The element $\top \cdot 0$ occurs in several contexts, such as infinite computations and temporal logic [13, 29, 27]. Its role in the present work is that in all three correctness approaches $\top \cdot 0$ represents the endless loop or never terminating program; we denote it by $\mathsf{L} =_{\mathrm{def}} \top \cdot 0$.

We assume the following, independent axioms about L:

$$\begin{array}{ll} (\mathsf{L1}) & d(x)\mathsf{L} = x\mathsf{L} \\ (\mathsf{L2}) & d(\mathsf{L})x \leq xd(\mathsf{L}) \end{array}$$

Axiom ($\mathsf{L1}$) and the more restrictive $d(\mathsf{L}) = 1$ are used in our algebraic treatment of general correctness [17, 18]; the current ($\mathsf{L2}$) is new.

Axiom ($\mathsf{L1}$) generalises a law typical for relational models of total correctness such as UTP, namely $d(x)\top = x\top$. The latter follows from the explicit $d(x) = x\top \curlywedge 1$ by $d(x)\top = (x\top \curlywedge 1)\top \leq x\top\top = x\top \leq d(x)\top$, but it is not valid in general correctness.

Axiom (L2) holds in partial correctness, where $\mathsf{L} = 0$ and hence $d(\mathsf{L}) = 0$; in total correctness, where $\mathsf{L} = \top$ and hence $d(\mathsf{L}) = 1$; and in general correctness where also $d(\mathsf{L}) = 1$. For prescriptions, the latter is obtained by

$$\begin{aligned} d(\mathsf{L}) &= d(\mathit{false} \Vdash \mathit{false}) = (\mathit{true} \Vdash a(\mathit{false}) \vee d(\mathit{false})) \\ &= (\mathit{true} \Vdash (\neg \mathit{false} \wedge \vec{v}' = \vec{v}) \vee \mathit{false}) = (\mathit{true} \Vdash \vec{v}' = \vec{v}) = 1 \,. \end{aligned}$$

The endless loop L is a particular case of an element being formed by sequential composition with 0. Intuitively, this operation cuts out all terminating executions, thus $x0$ retains only the non-terminating executions of x [27, 23]. In presence of (L2), the axiom (L1) can be replaced by its instance (L3) for such non-terminating elements:

$$\begin{array}{ll} (\mathsf{L3}) & d(x0)\mathsf{L} = x0 \\ (\mathsf{L3'}) & x \leq y \Leftrightarrow x \leq y + \mathsf{L} \wedge x \leq y + d(y0)\top \end{array}$$

Along with a few other facts, including the equivalence of (L3) and (L3'), this is shown in the following lemma.

Lemma 1.

1. $(\mathsf{L1}) \wedge (\mathsf{L2}) \Leftrightarrow (\mathsf{L3}) \wedge (\mathsf{L2})$.
2. $(\mathsf{L1}) \Rightarrow x\top \curlywedge \mathsf{L} = x\mathsf{L}$.
3. $(\mathsf{L1}) \Rightarrow (x0 \curlywedge y0)0 = x0 \curlywedge y0$.
4. $(\mathsf{L2}) \Rightarrow xa(\mathsf{L})\top \leq x0 + a(\mathsf{L})\top$.
5. $(\mathsf{L3}) \Leftrightarrow (\mathsf{L3'})$.

Proof.

1. The forward implication is clear since (L3) is an instance of (L1). The backward implication follows by
$$\begin{aligned} d(x)\mathsf{L} &= d(x)d(\mathsf{L})\mathsf{L} = d(\mathsf{L})d(x)\mathsf{L} = d(d(\mathsf{L})x)\mathsf{L} \leq d(xd(\mathsf{L}))\mathsf{L} = d(x\mathsf{L})\mathsf{L} \\ &= d(x\top 0)\mathsf{L} = x\top 0 = x\mathsf{L} \leq d(x)\top\mathsf{L} = d(x)\mathsf{L} \,. \end{aligned}$$
2. Since $\cdot$ is $\leq$-isotone we have $x\mathsf{L} \leq x\top$ and $x\mathsf{L} \leq \top\mathsf{L} = \mathsf{L}$. Hence $x\mathsf{L} \leq x\top \curlywedge \mathsf{L} = d(x\top \curlywedge \mathsf{L})(x\top \curlywedge \mathsf{L}) \leq d(x\top)\mathsf{L} = x\top\mathsf{L} = x\mathsf{L}$.
3. We use (L1) in
$$\begin{aligned} x0 \curlywedge y0 &= d(x0 \curlywedge y0)(x0 \curlywedge y0) \leq d(x0 \curlywedge y0)\mathsf{L} = (x0 \curlywedge y0)\mathsf{L} \\ &= (x0 \curlywedge y0)\top 0 \leq (x0\top \curlywedge y0\top)0 = (x0 \curlywedge y0)0 \leq x0 \curlywedge y0 \,. \end{aligned}$$
4. The claim is proved by separating the cases $d(\mathsf{L})$ and $a(\mathsf{L})$:
$$\begin{aligned} xa(\mathsf{L})\top &= 1xa(\mathsf{L})\top = (d(\mathsf{L}) + a(\mathsf{L}))xa(\mathsf{L})\top = d(\mathsf{L})xa(\mathsf{L})\top + a(\mathsf{L})xa(\mathsf{L})\top \\ &\leq xd(\mathsf{L})a(\mathsf{L})\top + a(\mathsf{L})\top = x0 + a(\mathsf{L})\top \,. \end{aligned}$$
5. Assume (L3). The forward implication of (L3') is clear since $+$ is the join operator. For the backward implication let $x \leq y + \mathsf{L}$ and $x \leq y + d(y0)\top$. We then obtain $x \leq y$ by separating the cases $d(y0)$ and $a(y0)$:
$$\begin{aligned} x &= d(y0)x + a(y0)x \leq d(y0)(y + \mathsf{L}) + a(y0)(y + d(y0)\top) \\ &= d(y0)y + d(y0)\mathsf{L} + a(y0)y + a(y0)d(y0)\top \leq y + y0 + 0 = y \,. \end{aligned}$$
Assume (L3'). We then obtain $d(x0)\mathsf{L} \leq x0$ since $d(x0)\mathsf{L} \leq \mathsf{L} \leq x0 + \mathsf{L}$ and $d(x0)\mathsf{L} \leq d(x0)\top \leq x0 + d(x0 \cdot 0)\top$, thus (L3) by $x0 = d(x0)x0 \leq d(x0)\mathsf{L}$. □

2.4 Specific Laws

The axioms discussed so far are common to partial, total and general correctness. Table 1 presents a sample of laws which hold in some correctness approaches only. These, and further ones, can be imposed as axioms or derived from other axioms when reasoning in a particular sub-theory. We do not consider them further as our goal is a theory unifying the three correctness approaches.

	partial	total	general
$\mathsf{L} = 1$	$-$	$-$	$-$
$\mathsf{L} = 0$	$+$	$-$	$-$
$\mathsf{L} = \top$	$-$	$+$	$-$
$d(\mathsf{L}) \leq \mathsf{L}$	$+$	$+$	$-$
$\mathsf{L} \curlywedge 1 = a(\mathsf{L})$	$-$	$-$	$+$
$\mathsf{L} \curlywedge 1 = 0$	$+$	$-$	$+$
$d(\mathsf{L}) = 1$	$-$	$+$	$+$
$d(\mathsf{L})\top \leq \top d(\mathsf{L})$	$+$	$+$	$+$

Table 1. Laws of particular correctness approaches

3 Unified Semantics

In this section, we describe the unified semantics of partial, total and general correctness. Sequential composition and non-deterministic choice are modelled by the operations $+$ and $\cdot$, respectively. Hence $\leq$ is the common refinement order. Also traditional is the conditional statement given by $\mathsf{if}\ p\ \mathsf{then}\ x\ \mathsf{else}\ y =_{\mathrm{def}} px + a(p)y$ for domain elements p. A unified semantics of finite and infinite iteration has been presented in [18]. It arises in Section 3.3 as a special case of the unified semantics of recursion, which we first discuss.

3.1 Approximation

In contrast to the refinement order, the approximation order is used to fix the meaning of recursively defined programs. Both orders coincide in the case of partial and total correctness, but not in general correctness, where approximation is given by the Egli-Milner order [30]. For general correctness, we have expressed the Egli-Milner order algebraically in terms of the refinement order [17, 18]. Based on that, we use the following generalisation to cover partial, total and general correctness:

$$x \sqsubseteq y \Leftrightarrow_{\mathrm{def}} x \leq y + \mathsf{L} \wedge d(\mathsf{L})y \leq x + d(x0)\top \ .$$

Let us instantiate the approximation relation $\sqsubseteq$ for each correctness approach to see why it is appropriate:

– In partial correctness we have $\mathsf{L} = 0$, hence $x \sqsubseteq y \Leftrightarrow x \leq y$. Refinement and approximation coincide.
– In total correctness we have $\mathsf{L} = \top$, hence $x \sqsubseteq y \Leftrightarrow y \leq x + d(x0)\top = x + x0 = x$ assuming the law $d(x)\top = x\top$ valid in UTP and other relational models. Thus approximation is the converse of refinement, which enables us to take the same (least) fixpoints as in the other approaches.
– In general correctness we exemplify $\sqsubseteq$ for prescriptions. A calculation similar to the one in [17] yields

$$(P_1 \Vdash Q_1) \sqsubseteq (P_2 \Vdash Q_2) \Leftrightarrow P_1 \leq P_2 \wedge Q_1 \leq Q_2 \wedge P_1 \mathbin{\downarrow} Q_2 \leq Q_1 \ .$$

This is the Egli-Milner order given by [15].

Although we know that in these three instances the relation $\sqsubseteq$ is a partial order, we still have to show this in general. This is done by the following theorem, which also shows that the basic operators are $\sqsubseteq$-isotone. Our previous result for the Egli-Milner order in [18] is similar, but due to subtle differences between the orders we provide a new proof.

Theorem 2. *The relation $\sqsubseteq$ is a preorder with least element L. It is a partial order if and only if* (L3) *holds. The operations $+$ and $\cdot z$ are $\sqsubseteq$-isotone. The operation $z\cdot$ is $\sqsubseteq$-isotone if* (L2) *holds.*

Proof. Reflexivity is clear since $x \leq x + \mathsf{L}$ and $d(\mathsf{L})x \leq x$. The least element is L since $\mathsf{L} \leq y + \mathsf{L}$ and $d(\mathsf{L})y \leq d(\mathsf{L})\top \leq \mathsf{L} + d(\mathsf{L}0)\top$. For transitivity let $x \sqsubseteq y$ and $y \sqsubseteq z$. Thus $x \leq y + \mathsf{L}$ and $y \leq z + \mathsf{L}$, whence $x \leq y + \mathsf{L} \leq z + \mathsf{L}$. Moreover $d(\mathsf{L})y \leq x + d(x0)\top$ and $d(\mathsf{L})z \leq y + d(y0)\top$, whence

$$\begin{aligned} d(\mathsf{L})z &= d(\mathsf{L})d(\mathsf{L})z \leq d(\mathsf{L})(y + d(y0)\top) = d(\mathsf{L})y + d(\mathsf{L})d(y0)\top \\ &= d(\mathsf{L})y + d(d(\mathsf{L})y0)\top \leq x + d(x0)\top + d((x + d(x0)\top)0)\top \\ &= x + d(x0)\top + d(x0 + d(x0)\mathsf{L})\top = x + d(x0)\top + d(x0)d(\mathsf{L})\top \\ &= x + d(x0)\top \ . \end{aligned}$$

Hence $x \sqsubseteq z$. By Lemma 1.5 we can use (L3') to show that $\sqsubseteq$ is antisymmetric. Let $x \sqsubseteq y$ and $y \sqsubseteq x$, then $x \leq y + \mathsf{L}$ and $d(\mathsf{L})x \leq y + d(y0)\top$. Thus

$$x = d(\mathsf{L})x + a(\mathsf{L})x \leq d(\mathsf{L})x + a(\mathsf{L})y + a(\mathsf{L})\mathsf{L} \leq y + d(y0)\top \ .$$

We therefore have $x \leq y$ by (L3'), and a symmetric argument shows $y \leq x$. On the other hand, we obtain (L3) by assuming that $\sqsubseteq$ is antisymmetric since $d(x0)\mathsf{L} \sqsubseteq x0$ and $x0 \sqsubseteq d(x0)\mathsf{L}$ follow from

$$\begin{aligned} d(x0)\mathsf{L} \leq \mathsf{L} = x0 + \mathsf{L} \ , \qquad & d(\mathsf{L})x0 \leq x0 \leq d(x0)\mathsf{L} \leq d(x0)\mathsf{L} + d(d(x0)\mathsf{L}0)\top \ , \\ x0 \leq \mathsf{L} = d(x0)\mathsf{L} + \mathsf{L} \ , \qquad & d(\mathsf{L})d(x0)\mathsf{L} \leq d(x0)\top \leq x0 + d(x0 \cdot 0)\top \ . \end{aligned}$$

For the isotony claims assume $x \sqsubseteq y$, hence $x \leq y + \mathsf{L}$ and $d(\mathsf{L})y \leq x + d(x0)\top$. Then $x + z \sqsubseteq y + z$ follows from $x + z \leq y + z + \mathsf{L}$ and $d(\mathsf{L})(y + z) \leq d(\mathsf{L})y + z \leq x + z + d(x0)\top \leq x + z + d((x + z)0)\top$. Moreover $xz \sqsubseteq yz$ follows from $xz \leq (y + \mathsf{L})z = yz + \mathsf{L}$ and $d(\mathsf{L})yz \leq (x + d(x0)\top)z = xz + d(x0)\top z \leq xz + d(xz0)\top$. Finally $zx \sqsubseteq zy$ follows from $zx \leq z(y + \mathsf{L}) \leq zy + \mathsf{L}$ and $d(\mathsf{L})zy \leq zd(\mathsf{L})y \leq z(x + d(x0)\top) = zx + zd(x0)\top \leq zx + d(zx0)\top$ using (L2). ◻

By developing the unified semantics algebraically in a first-order theory, we considerably profit from tools such as automated theorem provers and counterexample generators. For example, the converse of the last claim in Theorem 2 does not hold as witnessed by a 12-element counterexample generated by Mace4, even if (L1) is assumed additionally.

3.2 Recursion

With the approximation order in place, we can define the unified semantics of recursion. Let $f : S \to S$ be the characteristic function of the recursion. The unified semantics of the recursion is the $\sqsubseteq$-least fixpoint of f, denoted by κf provided it exists. We furthermore use the $\leq$-least prefixpoint and the $\leq$-greatest postfixpoint of f, which are denoted by μf and νf, respectively, provided they exist. To be precise, we require that these elements satisfy the following laws:

$$\begin{array}{ll} f(\kappa f) = \kappa f & f(x) = x \Rightarrow \kappa f \sqsubseteq x \\ f(\mu f) = \mu f & f(x) \leq x \Rightarrow \mu f \leq x \\ f(\nu f) = \nu f & x \leq f(x) \Rightarrow x \leq \nu f \end{array}$$

It follows that μf and νf are the $\leq$-least and $\leq$-greatest fixpoints of f, respectively. By the discussion about the unified approximation order in Section 3.1 we immediately obtain that κf is appropriate in all three correctness approaches. In particular, $\kappa f = \mu f$ in partial correctness, and $\kappa f = \nu f$ in total correctness.

The main result of this section gives explicit representations of κf and conditions for its existence. We denote by $x \sqcap y$ the greatest lower bound of x and y with respect to $\sqsubseteq$, provided it exists.

Theorem 3. *Let $f : S \to S$ be $\leq$- and $\sqsubseteq$-isotone, and assume that μf and νf exist. Then the following are equivalent:*

1. *κf exists.*
2. *$\kappa f = \mu f \sqcap \nu f$.*
3. *$\kappa f = (\nu f \curlywedge \mathsf{L}) + \mu f$.*
4. *$d(\mathsf{L})\nu f \leq (\nu f \curlywedge \mathsf{L}) + \mu f + d(\nu f 0)\top$.*
5. *$(\nu f \curlywedge \mathsf{L}) + \mu f \sqsubseteq \nu f$.*
6. *$\mu f \sqcap \nu f = (\nu f \curlywedge \mathsf{L}) + \mu f$.*
7. *$\mu f \sqcap \nu f \leq \nu f$.*

Proof. Abbreviate $\ell =_{\text{def}} (\nu f \curlywedge \mathsf{L}) + \mu f$ and $m =_{\text{def}} \mu f \sqcap \nu f$. Since $\nu f 0 \leq \nu f$ and $\nu f 0 \leq \mathsf{L}$ we have $\nu f 0 \leq \nu f \curlywedge \mathsf{L} \leq \ell \leq \nu f + \mu f = \nu f$, whence $\nu f 0 = \ell 0$. We first show that statements (4)–(7) are equivalent:

(4) $\Rightarrow$ (5): Because $\ell \leq \nu f \leq \nu f + \mathsf{L}$ and $d(\mathsf{L})\nu f \leq \ell + d(\nu f 0)\top = \ell + d(\ell 0)\top$ we obtain $\ell \sqsubseteq \nu f$.

(5) $\Rightarrow$ (6): We have $\ell \leq \mu f + \mathsf{L}$ and $d(\mathsf{L})\mu f \leq d(\mathsf{L})\nu f \leq \ell + d(\ell 0)\top$, thus $\ell \sqsubseteq \mu f$. Let $x \sqsubseteq \mu f$ and $x \sqsubseteq \nu f$, then $x \leq \mu f + \mathsf{L} \leq \ell + \mathsf{L}$ and $d(\mathsf{L})\ell \leq d(\mathsf{L})\nu f \leq x + d(x0)\top$, whence $x \sqsubseteq \ell$.

(6) $\Rightarrow$ (7): This is immediate since $\ell \leq \nu f$.

(7) $\Rightarrow$ (4): From $m \sqsubseteq \mu f$ we obtain $m \le \mu f + \mathsf{L}$, whence $m \le \nu f \curlywedge (\mu f + \mathsf{L}) = (\nu f \curlywedge \mu f) + (\nu f \curlywedge \mathsf{L}) = \ell$ by distributivity and the meet property of $\curlywedge$. Thus $m \sqsubseteq \nu f$ implies $d(\mathsf{L})\nu f \le m + d(m0)\top \le \ell + d(\ell 0)\top = \ell + d(\nu f 0)\top$.

We next add statements (1)–(3) to this cycle:

(1) $\Rightarrow$ (2): Clearly $\kappa f \sqsubseteq \mu f$ and $\kappa f \sqsubseteq \nu f$. Let $x \sqsubseteq \mu f$ and $x \sqsubseteq \nu f$, then $x \le \mu f + \mathsf{L} \le \kappa f + \mathsf{L}$ and $d(\mathsf{L})\kappa f \le d(\mathsf{L})\nu f \le x + d(x0)\top$, whence $x \sqsubseteq \kappa f$.
(2) $\Rightarrow$ (7): This is immediate since $\kappa f \le \nu f$.
(7) $\Rightarrow$ (3): This step uses isotony of f. From $m \sqsubseteq \mu f$ we get $f(m) \sqsubseteq f(\mu f) = \mu f$ and $m \le \mu f + \mathsf{L} = f(\mu f) + \mathsf{L} \le f(m) + \mathsf{L}$ since $\mu f \le m$ by (6). From $m \sqsubseteq \nu f$ we get $f(m) \sqsubseteq f(\nu f) = \nu f$ and $d(\mathsf{L})f(m) \le d(\mathsf{L})f(\nu f) = d(\mathsf{L})\nu f \le m + d(m0)\top$ by (7). Hence $m \sqsubseteq f(m) \sqsubseteq m$, thus $f(\ell) = \ell$ by (6) and Theorem 2.
Let $f(x) = x$, then $\mu f \le x \le \nu f$, whence $\ell \le \mu f + \mathsf{L} \le x + \mathsf{L}$ and $d(\mathsf{L})x \le d(\mathsf{L})\nu f \le \ell + d(\nu f 0)\top = \ell + d(\ell 0)\top$ by (4). Thus $\ell \sqsubseteq x$.
(3) $\Rightarrow$ (1): This is clear. □

Statements (3) and (4) of this theorem describe the $\sqsubseteq$-least fixpoint κf and its existence in terms of the refinement order $\le$. In all representations, the $\le$-least and $\le$-greatest fixpoints are separated; this is in contrast to the partitioned fixpoint of [7] which nests one fixpoint operator inside another.

In partial and general correctness the additional representation $\kappa f = \nu f 0 + \mu f$ holds [17, 18]. The counterexample $f(x) = x \curlywedge 1$ with $\mu f = 0$ and $\nu f = 1$ shows that in total correctness this representation cannot be added as an equivalent condition to Theorem 3. Yet we obtain the following, sufficient conditions that describe when $\kappa f = \nu f 0 + \mu f$ in partial, total and general correctness.

Corollary 4. *Let $f : S \to S$ be $\le$- and $\sqsubseteq$-isotone, and assume that μf and νf exist. Then the following are equivalent:*

1. $\kappa f = \nu f 0 + \mu f$.
2. $d(\mathsf{L})\nu f \le \mu f + d(\nu f 0)\top$.
3. $\nu f 0 + \mu f \sqsubseteq \nu f$.
4. $\mu f \sqcap \nu f = \nu f 0 + \mu f$.

They imply the statements of Theorem 3.

Proof. Abbreviate $\ell =_{\mathrm{def}} (\nu f \curlywedge \mathsf{L}) + \mu f$ and $n =_{\mathrm{def}} \nu f 0 + \mu f$. Since $\nu f 0 \le \nu f$ and $\nu f 0 \le \mathsf{L}$ we have $n \le \ell$. Assuming (2) we obtain $n = \ell$ since Lemma 1.2 gives

$$\begin{aligned} \nu f \curlywedge \mathsf{L} &= \nu f \curlywedge \mathsf{L} \curlywedge \mathsf{L} = d(\nu f \curlywedge \mathsf{L})(\nu f \curlywedge \mathsf{L}) \curlywedge \mathsf{L} \le d(\mathsf{L})\nu f \curlywedge \mathsf{L} \\ &\le (\mu f + d(\nu f 0)\top) \curlywedge \mathsf{L} \le \mu f + (d(\nu f 0)\top \curlywedge \mathsf{L}) = \mu f + d(\nu f 0)\mathsf{L} = n \,. \end{aligned}$$

But (2) also implies Theorem 3.4 and therefore (1), (3) and (4) by setting $\ell = n$ in statements (3), (5) and (6) of Theorem 3. Conversely, (3) $\Rightarrow$ (2) by

$$d(\mathsf{L})\nu f \le n + d(n0)\top = \nu f 0 + \mu f + d(\nu f 0)\top = \mu f + d(\nu f 0)\top \,,$$

and (1) $\Rightarrow$ (3) since $\kappa f \sqsubseteq \nu f$ and (4) $\Rightarrow$ (3) since $\mu f \sqcap \nu f \sqsubseteq \nu f$. □

3.3 Iteration

In partial, total and general correctness, the semantics of the loop $\mathsf{while}\ p\ \mathsf{do}\ y$ is obtained as the appropriate solution of the equation $x = pyx + a(p)$, using a domain element p. As discussed above it is given by the fixpoint operator κ. To represent the solution of this particular recursion we use the Kleene star and omega operations given by the following axioms [25, 8, 27]:

$$\begin{array}{ll} 1 + y^*y \leq y^* & z + xy \leq x \Rightarrow zy^* \leq x \\ 1 + yy^* \leq y^* & z + yx \leq x \Rightarrow y^*z \leq x \\ yy^\omega = y^\omega & x \leq yx + z \Rightarrow x \leq y^\omega + y^*z \end{array}$$

It follows that $y^*z = \mu(\lambda x.yx + z)$ and $y^\omega + y^*z = \nu(\lambda x.yx + z)$. We also have the decomposition laws $(x + y)^* = x^*(yx^*)^*$ and $(x + y)^\omega = (x^*y)^\omega + (x^*y)^*x^\omega$. Moreover $y^\omega = y^\omega\top$ and $y^*0 \leq y^\omega 0$. The operations * and $^\omega$ are $\leq$-isotone.

The unified semantics of the while-loop is obtained by the following result.

Corollary 5. *Let $y \in S$ and $p, q \in d(S)$ and $f(x) =_{\mathrm{def}} pyx + q$. Then $\kappa f = (py)^\omega 0 + (py)^*q$.*

Proof. We have $\mu f = (py)^*q$ and $\nu f = (py)^\omega + (py)^*q$. The function f is $\leq$-isotone and by Theorem 2 also $\sqsubseteq$-isotone. Thus κf exists since Corollary 4.2 holds by

$$\begin{array}{c} \nu f 0 = ((py)^\omega + (py)^*q)0 = (py)^\omega 0 + (py)^*0 = (py)^\omega 0 = (py)^\omega \mathsf{L}\ , \\ d(\mathsf{L})(py)^\omega \leq (py)^\omega d(\mathsf{L}) \leq d((py)^\omega d(\mathsf{L}))\top = d((py)^\omega \mathsf{L})\top = d(\nu f 0)\top\ , \\ d(\mathsf{L})\nu f = d(\mathsf{L})((py)^\omega + \mu f) \leq d(\mathsf{L})(py)^\omega + \mu f \leq d(\nu f 0)\top + \mu f\ , \end{array}$$

using (L2). Hence Corollary 4.1 gives $\kappa f = \nu f 0 + \mu f = (py)^\omega 0 + (py)^*q$. □

We have thus replaced the incidental observation of [18], that $(py)^\omega 0 + (py)^*a(p)$ is an adequate semantics of $\mathsf{while}\ p\ \mathsf{do}\ y$ for partial, total and general correctness, by a systematic derivation.

This section is concluded by showing that both finite and infinite iteration are $\sqsubseteq$-isotone, which generalises our previous result for finite iteration in general correctness [18]. Hence by Theorem 2 also $\mathsf{while}\ p\ \mathsf{do}\ y$ is $\sqsubseteq$-isotone in y.

Theorem 6. *Let $x, y \in S$ such that $x \sqsubseteq y$. Then $x^* \sqsubseteq y^*$ and $x^\omega \sqsubseteq y^\omega$.*

Proof. From $x \sqsubseteq y$ we obtain $x \leq y + \mathsf{L}$ and $d(\mathsf{L})y \leq x + d(x0)\top$. For $x^* \sqsubseteq y^*$ we have $x^* \leq (y + \mathsf{L})^* = y^*(\mathsf{L}y^*)^* = y^*\mathsf{L}^* = y^* + y^*\mathsf{L}\mathsf{L}^* = y^* + \mathsf{L}$, whence it suffices to show $d(\mathsf{L})y^* = d(\mathsf{L})(d(\mathsf{L})y)^* \leq z =_{\mathrm{def}} x^* + d(x^*0)\top$. The first step imports the test $d(\mathsf{L})$ into the iteration by [18, Lemma 9] using (L2). The second step follows by instantiating a star axiom with $d(\mathsf{L}) + zd(\mathsf{L})y \leq 1 + z(x + d(x0)\top) \leq z$ using $x^*d(x0)\top \leq d(x^*x0)\top \leq d(x^*0)\top$. For $x^\omega \sqsubseteq y^\omega$ we have

$$x^\omega \leq (y + \mathsf{L})^\omega = (y^*\mathsf{L})^\omega + (y^*\mathsf{L})^*y^\omega = y^*\mathsf{L}(y^*\mathsf{L})^\omega + y^\omega + y^*\mathsf{L}(y^*\mathsf{L})^*y^\omega = y^\omega + \mathsf{L}$$

since $y^*\mathsf{L}w = y^*\mathsf{L} = \mathsf{L}$ for any $w \in S$, and

$$\begin{array}{rl} d(\mathsf{L})y^\omega = (d(\mathsf{L})y)^\omega \leq (x + d(x0)\top)^\omega = (x^*d(x0)\top)^\omega + (x^*d(x0)\top)^*x^\omega & \\ = x^*d(x0)\top(x^*d(x0)\top)^\omega + x^\omega + x^*d(x0)\top(x^*d(x0)\top)^*x^\omega & \\ \leq x^\omega + x^*d(x0)\top \leq x^\omega + d(x^*0)\top \leq x^\omega + d(x^\omega 0)\top & \end{array}$$

again by importing $d(\mathsf{L})$, this time into the infinite iteration. □

4 Representation by Pairs

In this section, we give a unified representation of programs as pairs of termination and state transition information. Describing programs this way is standard in total correctness, exemplified by UTP's designs, and in general correctness, exemplified by the prescriptions of [14]. Hoare and He argue that the pair-based representation is helpful for practical application [22, page 81].

Algebraic accounts of designs and prescriptions are given in [28, 19, 17, 20]. In the following we describe a representation which is suitable for total and general correctness at the same time. Partial correctness is included as an extreme case, too.

4.1 Havoc

In Section 2.3 we have already discussed the operation $\cdot 0$ which cuts out the terminating executions of a program, giving rise to the loop element L. This operation can be used to obtain one component of the pair-based representation: that which describes the (non-)termination information. For the other component we need an operation that gives us the terminating executions, cutting out the non-terminating ones. We proceed in two steps.

First, we axiomatise the greatest element which contains only terminating executions. In general correctness, it corresponds to the command havoc of [30], whence we denote the element by H. In UTP's total-correctness approach, it corresponds to the design $(\mathit{true} \vdash \mathit{true})$. We use the following axioms (not to be confused with the healthiness conditions H1–H4 of UTP):

$$\begin{array}{ll} (\mathsf{H1}) & \mathsf{H}0 = 0 \\ (\mathsf{H2}) & x \leq x0 + \mathsf{H} \end{array}$$

Axiom (H2) and the more restrictive $x \leq y + \mathsf{L} \wedge x \leq y + \mathsf{H} \Rightarrow x \leq y$ are used in our algebraic treatment of general correctness [17]. The latter implies (H1), but is not suitable for total correctness.

A few facts about H are shown in the following lemma. By instantiating its first claim with $y = 0$, we obtain that H is indeed the greatest terminating element. As such, it is axiomatised in [31] in a total-correctness setting, but as a counterexample generated by Mace4 shows, those axioms do not imply (H2) which we need for our representation as pairs in Section 4.2.

Lemma 7. *Assume* (H1) *and* (H2).

1. $x0 \leq y \Leftrightarrow x \leq y + \mathsf{H}$.
2. $1 \leq \mathsf{H} = \mathsf{H}^2 = \mathsf{H}^*$.
3. $\mathsf{L} + \mathsf{H} = \top$.

Proof.

1. $(x + \mathsf{H})0 = x0 + \mathsf{H}0 = x0 + 0 \leq x$ by (H1). With (H2) and $\leq$-isotony of $\cdot$ and $+$ the claimed Galois connection follows by [9, Lemma 7.26]. Furthermore, the Galois connection conversely implies (H1) and (H2).

2. $1 \leq \mathsf{H}$ by part 1 since $1 \cdot 0 = 0$. Hence $\mathsf{H} = 1\mathsf{H} \leq \mathsf{H}^2$, while $\mathsf{H}^2 \leq \mathsf{H}$ by part 1 since $\mathsf{H}^2 0 = \mathsf{H}0 = 0$. Hence $1 + \mathsf{H}^2 \leq \mathsf{H}$, which implies $\mathsf{H}^* \leq \mathsf{H}$. Finally, $\mathsf{H} = 1\mathsf{H} \leq \mathsf{H}^*\mathsf{H} \leq \mathsf{H}^*$.
3. $\top \leq \mathsf{L} + \mathsf{H}$ by part 1 since $\top 0 = \mathsf{L}$. □

Second, we cut out the non-terminating executions by forming the meet with H. Intuitively, $x \curlywedge \mathsf{H}$ retains only the terminating executions of x since H contains all terminating executions and no non-terminating ones. It should be noted that although $\mathsf{L} + \mathsf{H} = \top$, we do not have $\mathsf{L} \curlywedge \mathsf{H} = 0$, so H is not a complement of L in S; hence they are not partitioning elements [7]. In particular, $\mathsf{L} \curlywedge \mathsf{H} = \mathsf{H} \neq 0$ in total correctness, where $\mathsf{L} = \top$. In partial correctness $\mathsf{H} = \top$ because $\mathsf{L} = 0$. There and in general correctness $\mathsf{L} \curlywedge \mathsf{H} = 0$ holds.

4.2 Pair Programming

We represent the program $x \in S$ by the pair $(x0, x \curlywedge \mathsf{H})$. The first component $x0$ describes those states from where non-terminating executions are possible. The second component $x \curlywedge \mathsf{H}$ describes the possible state transitions for terminating executions.

UTP's designs remind us that the representation as pairs is not necessarily unique. For example, the two designs $(\mathit{false} \vdash \mathit{false})$ and $(\mathit{false} \vdash \mathit{true})$ are equal. This is in contrast to prescriptions, which uniquely represent programs in general correctness. A unique representation for designs can be obtained by choosing a canonical representative, for example, by requiring $\neg P \Rightarrow Q$ for each design $(P \vdash Q)$. We follow this strategy to obtain a unique common representation.

Formally, we take representatives from the set $S' =_{\text{def}} \{(x0, x \curlywedge \mathsf{H}) \mid x \in S\}$. The representation and abstraction functions are given by

$$\begin{array}{ll} \rho : S \to S' & \pi : S' \to S \\ \rho(x) =_{\text{def}} (x0, x \curlywedge \mathsf{H}) & \pi((x, y)) =_{\text{def}} x + y \end{array}$$

We write $\pi(x, y)$ instead of $\pi((x, y))$ and similarly for other functions on pairs. The following lemma shows that ρ and π are in fact bijections.

Lemma 8.

1. $x = x0 + (x \curlywedge \mathsf{H})$, *hence* $\pi \circ \rho = \mathrm{id}_S$ *and* $\rho \circ \pi = \mathrm{id}_{S'}$.
2. $S' = \{(x0, y \curlywedge \mathsf{H}) \mid x, y \in S \wedge x0 \curlywedge \mathsf{H} \leq y\}$.

Proof.

1. $x = x \curlywedge (x0 + \mathsf{H}) = (x \curlywedge x0) + (x \curlywedge \mathsf{H}) = x0 + (x \curlywedge \mathsf{H})$ by (H2). This shows $\pi \circ \rho = \mathrm{id}_S$ or ρ is injective. But ρ is surjective by definition.
2. The inclusion ($\subseteq$) is immediate from $x0 \curlywedge \mathsf{H} \leq x0 \leq x$. For the inclusion ($\supseteq$) let $x, y \in S$ with $x0 \curlywedge \mathsf{H} \leq y$ and consider $z =_{\text{def}} x0 + (y \curlywedge \mathsf{H})$. Then

$$\begin{aligned} z0 &= (x0 + (y \curlywedge \mathsf{H}))0 = (x0)0 + (y \curlywedge \mathsf{H})0 = x0 + 0 = x0 \,, \\ z \curlywedge \mathsf{H} &= (x0 + (y \curlywedge \mathsf{H})) \curlywedge \mathsf{H} = (x0 \curlywedge \mathsf{H}) + (y \curlywedge \mathsf{H}) = y \curlywedge \mathsf{H} \,, \end{aligned}$$

   by using (H1) in the first calculation. Hence $(x0, y \curlywedge \mathsf{H}) = \rho(z) \in S'$. □

Part 2 of the preceding lemma shows the necessary restriction to obtain the dependence between the two components of a pair, and hence the unique representation. In partial and general correctness $x0 \leftthreetimes \mathsf{H} = 0$, whence the restriction is vacuous; it is only needed for total correctness.

Part 1 gives a decomposition of programs into their terminating and non-terminating executions. Similar decompositions of elements are admitted by separated IL-semirings [27] and quemirings [16]. We cannot use these structures since their axioms include $x0 \leftthreetimes \mathsf{H} = 0$, which is not valid in the unified setting.

In the extreme case of partial correctness, we have $\rho(x) = (x0, x \leftthreetimes \mathsf{H}) = (0, x)$, hence S' is just a copy of S with 0 attached in the first component of each pair. Nevertheless this correctly represents that there are no non-terminating executions in this approach.

4.3 Induced Operations on Pairs

Operations on the pairs S' can now be derived from operations on S by using the bijections ρ and π as an isomorphism. From the unary operation $f : S \to S$ we derive the operation $f' : S' \to S'$ defined by $f' =_{\text{def}} \rho \circ f \circ \pi$, and similarly for binary operations; for the constant $c \in S$ we obtain $c' =_{\text{def}} \rho(c) \in S'$. We denote an operation and its derived counterpart by the same symbol, relying on the context for disambiguation. The following theorem gives the derived operations.

Theorem 9. *Let $(t, x) \in S'$ and $(u, y) \in S'$. Then,*

$$
\begin{aligned}
(t,x) + (u,y) &= (t+u, x+y) & \qquad 0' &= (0,0)\\
(t,x) \cdot (u,y) &= (t + xu, (t + xu + xy) \leftthreetimes \mathsf{H}) & \qquad 1' &= (0,1)\\
(t,x) \leftthreetimes (u,y) &= (t \leftthreetimes u, x \leftthreetimes y) & \qquad \top' &= (\mathsf{L}, \mathsf{H})\\
d(t,x) &= (0, d(t+x)) & \qquad \mathsf{L}' &= (\mathsf{L}, \mathsf{L} \leftthreetimes \mathsf{H})\\
(t,x)^* &= (x^*t, (x^*t + x^*) \leftthreetimes \mathsf{H}) & \qquad \mathsf{H}' &= (0, \mathsf{H})\\
(t,x)^\omega &= (x^\omega 0 + x^*t, (x^\omega + x^*t) \leftthreetimes \mathsf{H})
\end{aligned}
$$

Proof. Observe that $t0 = t$ and $x \leftthreetimes \mathsf{H} = x$ and $t \leftthreetimes \mathsf{H} \leq x$ by Lemma 8.2 and $x0 = (x \leftthreetimes \mathsf{H})0 \leq \mathsf{H}0 = 0$ by (H1). Similar properties hold for u and y. Hence

$$
\begin{aligned}
(t,x) + (u,y) &= \rho(\pi(t,x) + \pi(u,y)) = \rho(t + x + u + y)\\
&= ((t + x + u + y)0, (t + x + u + y) \leftthreetimes \mathsf{H})\\
&= (t0 + x0 + u0 + y0, (t \leftthreetimes \mathsf{H}) + (x \leftthreetimes \mathsf{H}) + (u \leftthreetimes \mathsf{H}) + (y \leftthreetimes \mathsf{H}))\\
&= (t + u, x + y)\ .
\end{aligned}
$$

Moreover $xy0 = x0 = 0$ and $tz = t0z = t0 = t$ for any $z \in S$, whence

$$
\begin{aligned}
(t,x) \cdot (u,y) &= \rho(\pi(t,x) \cdot \pi(u,y)) = \rho((t+x)(u+y)) = \rho(t(u+y) + x(u+y))\\
&= \rho(t + xu + xy) = (t0 + xu0 + xy0, (t + xu + xy) \leftthreetimes \mathsf{H})\\
&= (t + xu, (t + xu + xy) \leftthreetimes \mathsf{H})\ .
\end{aligned}
$$

This equals $(t + xu, ((t + xu) \leftthreetimes \mathsf{H}) + xy)$ since $xy \leq \mathsf{HH} = \mathsf{H}$ by Lemma 7.2. A simplification to $(t + xu, (t \leftthreetimes \mathsf{H}) + xy)$ would require the additional axiom $(w \leftthreetimes \mathsf{H})z \leftthreetimes \mathsf{H} = (w \leftthreetimes \mathsf{H})(z \leftthreetimes \mathsf{H})$ that is valid in partial, total and general correctness.

Next, $t \curlywedge y = t \curlywedge y \curlywedge \mathsf{H} \leq x \curlywedge y$ and symmetrically $u \curlywedge x \leq x \curlywedge y$, whence

$$\begin{aligned}(t,x) \curlywedge (u,y) &= \rho(\pi(t,x) \curlywedge \pi(u,y)) = \rho((t+x) \curlywedge (u+y)) \\ &= \rho((t \curlywedge u) + (t \curlywedge y) + (x \curlywedge u) + (x \curlywedge y)) = \rho((t \curlywedge u) + (x \curlywedge y)) \\ &= ((t \curlywedge u)0 + (x \curlywedge y)0, (t \curlywedge u \curlywedge \mathsf{H}) + (x \curlywedge y \curlywedge \mathsf{H})) \\ &= ((t \curlywedge u)0, x \curlywedge y \curlywedge \mathsf{H}) = (t \curlywedge u, x \curlywedge y)\ ,\end{aligned}$$

using Lemma 1.3 in the last step. Since $d(z) \leq 1 \leq \mathsf{H}$ by Lemma 7.2, we obtain the domain operation as

$$d(t,x) = \rho(d(\pi(t,x))) = \rho(d(t+x)) = (d(t+x)0, d(t+x) \curlywedge \mathsf{H}) = (0, d(t+x))\ .$$

For the Kleene star we use $x^*0 \leq \mathsf{H}^*0 = \mathsf{H}0 = 0$ by Lemma 7.2 to calculate

$$\begin{aligned}(t,x)^* &= \rho((\pi(t,x))^*) = \rho((t+x)^*) = \rho(x^*(tx^*)^*) = \rho(x^*t^*) = \rho(x^* + x^*tt^*) \\ &= \rho(x^* + x^*t) = (x^*0 + x^*t0, (x^* + x^*t) \curlywedge \mathsf{H}) = (x^*t, (x^*t + x^*) \curlywedge \mathsf{H})\ .\end{aligned}$$

This equals $(x^*t, (x^*t \curlywedge \mathsf{H}) + x^*)$ since $x^* \leq \mathsf{H}$. With the additional axiom above this could be further simplified to (x^*t, x^*). For the omega operation we calculate

$$\begin{aligned}(t,x)^\omega &= \rho((\pi(t,x))^\omega) = \rho((t+x)^\omega) = \rho((x^*t)^\omega + (x^*t)^*x^\omega) \\ &= \rho(x^*t(x^*t)^\omega + x^\omega + x^*t(x^*t)^*x^\omega) = \rho(x^*t + x^\omega) \\ &= (x^\omega 0 + x^*t, (x^\omega + x^*t) \curlywedge \mathsf{H})\ .\end{aligned}$$

The derived constants are obtained by

$$\begin{aligned}0' &= \rho(0) = (0 \cdot 0, 0 \curlywedge \mathsf{H}) = (0,0) \\ 1' &= \rho(1) = (1 \cdot 0, 1 \curlywedge \mathsf{H}) = (0,1) \\ \top' &= \rho(\top) = (\top 0, \top \curlywedge \mathsf{H}) = (\mathsf{L}, \mathsf{H}) \\ \mathsf{L}' &= \rho(\mathsf{L}) = (\mathsf{L}0, \mathsf{L} \curlywedge \mathsf{H}) = (\mathsf{L}, \mathsf{L} \curlywedge \mathsf{H}) \\ \mathsf{H}' &= \rho(\mathsf{H}) = (\mathsf{H}0, \mathsf{H} \curlywedge \mathsf{H}) = (0, \mathsf{H})\end{aligned}$$

using Lemma 7.2 and (H1). □

Let us remark that the domain elements of S' are $d(S') = \{(0,p) \mid p \in d(S)\}$ with complements taken in the second component of each pair.

From the choice operation, we immediately obtain the refinement order on pairs $(t,x) \leq (u,y) \Leftrightarrow t \leq u \wedge x \leq y$, which is the same as the induced order $\pi(t,x) \leq \pi(u,y)$. Another calculation shows that the approximation order on pairs induced by $(t,x) \sqsubseteq (u,y) \Leftrightarrow_{\mathrm{def}} \pi(t,x) \sqsubseteq \pi(u,y)$ can be explicitly expressed, too, by $x \leq y + \mathsf{L} \wedge u \leq t \wedge d(\mathsf{L})y \leq x + d(t)\top$.

Note that both ρ and π preserve both $\leq$ and $\sqsubseteq$ by definition. Hence a function $f' : S' \to S'$ is $\leq$- or $\sqsubseteq$-isotone if and only if $f = \pi \circ f' \circ \rho$ is so. In particular, we obtain

$$\varphi f' = \varphi(\rho \circ \pi \circ f') = \rho(\varphi(\pi \circ f' \circ \rho)) = \rho(\varphi f)$$

for any fixpoint operator $\varphi \in \{\kappa, \mu, \nu\}$ by rolling [9, Rule 8.29]. For $\varphi = \kappa$ this requires that f' is $\sqsubseteq$-isotone, and otherwise that f' is $\leq$-isotone.

We thus obtain one method to calculate fixpoints of functions on S'. Another way, which works directly on the pairs, is discussed next.

4.4 Recursion

A function h on the pair-based representation may be specified by two functions f, g applied separately to the pairs as in

$$h(t,x) =_{\text{def}} (f(t,x), g(t,x)) \ .$$

For total correctness, this is investigated in UTP, where [22, Theorem 3.1.6] shows how to obtain the $\leq$-greatest fixpoint of h by a 'mutually recursive formula'. In an algebraic context, this is generalised to $\leq$-least fixpoints in [20] and to general correctness in [17].

In the following we show how to obtain the $\sqsubseteq$-least, $\leq$-least and $\leq$-greatest fixpoints of h for our unified representation, and hence for partial, total, and general correctness at the same time.

Consider the function $h : S' \to S'$ as defined above. Hence the types of f and g are $f : S' \to S_0$ and $g : S' \to S_{\mathsf{H}}$ where $S_0 =_{\text{def}} \{x0 \mid x \in S\}$ and $S_{\mathsf{H}} =_{\text{def}} \{x \curlywedge \mathsf{H} \mid x \in S\}$. As subsets of S, both S_0 and S_{H} are partially ordered by $\leq$. By Lemma 8.2 we also have $f(t,x) \curlywedge \mathsf{H} \leq g(t,x)$ for each $(t,x) \in S'$.

We assume that h is $\leq$-isotone, whence both f and g preserve $\leq$ in both arguments as shown by

$$\begin{aligned} t \leq u \wedge x \leq y &\Leftrightarrow (t,x) \leq (u,y) \\ &\Rightarrow (f(t,x), g(t,x)) = h(t,x) \leq h(u,y) = (f(u,y), g(u,y)) \\ &\Leftrightarrow f(t,x) \leq f(u,y) \wedge g(t,x) \leq g(u,y) \ . \end{aligned}$$

Motivated by [22] we define for $\varphi \in \{\mu, \nu\}$ the auxiliary functions

$$\begin{array}{ll} P_\varphi : S_{\mathsf{H}} \to S_0 & P_\varphi(x) =_{\text{def}} \varphi(\lambda t. f(t, (t \curlywedge \mathsf{H}) + x)) \\ R_\varphi : S_{\mathsf{H}} \to S_{\mathsf{H}} & R_\varphi(x) =_{\text{def}} (P_\varphi(x) \curlywedge \mathsf{H}) + g(P_\varphi(x), (P_\varphi(x) \curlywedge \mathsf{H}) + x) \\ Q_\varphi : S_{\mathsf{H}} & Q_\varphi =_{\text{def}} \varphi R_\varphi \end{array}$$

We assume that the fixpoints taken in P_φ and Q_φ exist. Being composed from $\leq$-preserving operations, including the fixpoint operators by [9, Rule 8.28], both P_φ and R_φ preserve $\leq$.

Theorem 10. *$\mu h = (P_\mu(Q_\mu), Q_\mu)$ and $\nu h = (P_\nu(Q_\nu), Q_\nu)$.*

Proof. Let $\varphi \in \{\mu, \nu\}$. We first show that $(P_\varphi(Q_\varphi), Q_\varphi)$ is a fixpoint of h:

$$\begin{aligned} P_\varphi(Q_\varphi) \curlywedge \mathsf{H} \leq R_\varphi(Q_\varphi) = Q_\varphi \ , & \\ P_\varphi(Q_\varphi) &= f(P_\varphi(Q_\varphi), (P_\varphi(Q_\varphi) \curlywedge \mathsf{H}) + Q_\varphi) = f(P_\varphi(Q_\varphi), Q_\varphi) \ , \\ R_\varphi(Q_\varphi) &= (P_\varphi(Q_\varphi) \curlywedge \mathsf{H}) + g(P_\varphi(Q_\varphi), (P_\varphi(Q_\varphi) \curlywedge \mathsf{H}) + Q_\varphi) \\ &= (P_\varphi(Q_\varphi) \curlywedge \mathsf{H}) + g(P_\varphi(Q_\varphi), Q_\varphi) \ , \\ h(P_\varphi(Q_\varphi), Q_\varphi) &= (f(P_\varphi(Q_\varphi), Q_\varphi), g(P_\varphi(Q_\varphi), Q_\varphi)) \\ &= (f(P_\varphi(Q_\varphi), Q_\varphi), (f(P_\varphi(Q_\varphi), Q_\varphi) \curlywedge \mathsf{H}) + g(P_\varphi(Q_\varphi), Q_\varphi)) \\ &= (P_\varphi(Q_\varphi), (P_\varphi(Q_\varphi) \curlywedge \mathsf{H}) + g(P_\varphi(Q_\varphi), Q_\varphi)) \\ &= (P_\varphi(Q_\varphi), R_\varphi(Q_\varphi)) = (P_\varphi(Q_\varphi), Q_\varphi) \ . \end{aligned}$$

Now let $(f(t,x), g(t,x)) = h(t,x) \leq (t,x)$, whence $t \curlywedge \mathsf{H} \leq x$ by Lemma 8.2 and $f(t, (t \curlywedge \mathsf{H}) + x) = f(t,x) \leq t$ and $g(t,x) \leq x$. Then

$$\begin{aligned} P_\mu(x) &\leq t \;, \\ P_\mu(x) \curlywedge \mathsf{H} &\leq t \curlywedge \mathsf{H} \leq x \;, \\ g(P_\mu(x), (P_\mu(x) \curlywedge \mathsf{H}) + x) = g(P_\mu(x), x) &\leq g(t,x) \leq x \;, \\ R_\mu(x) &\leq x \;, \\ Q_\mu &\leq x \;, \\ P_\mu(Q_\mu) \leq P_\mu(x) &\leq t \;. \end{aligned}$$

Hence $(P_\mu(Q_\mu), Q_\mu) \leq (t,x)$, and thus $\mu h = (P_\mu(Q_\mu), Q_\mu)$.

The proof for νh is a bit different. Let $(t,x) \leq h(t,x) = (f(t,x), g(t,x))$, whence $t \curlywedge \mathsf{H} \leq x$ by Lemma 8.2 and $t \leq f(t,x) = f(t, (t \curlywedge \mathsf{H}) + x)$ and $x \leq g(t,x)$. Then

$$\begin{aligned} & t \leq P_\nu(x) \;, \\ & x \leq g(t,x) \leq g(P_\nu(x), (P_\nu(x) \curlywedge \mathsf{H}) + x) \leq R_\nu(x) \;, \\ & x \leq Q_\nu \;, \\ & t \leq P_\nu(x) \leq P_\nu(Q_\nu) \;. \end{aligned}$$

Hence $(t,x) \leq (P_\nu(Q_\nu), Q_\nu)$, and thus $\nu h = (P_\nu(Q_\nu), Q_\nu)$. □

We thus obtain the $\leq$-least and $\leq$-greatest fixpoints on pairs. But now we can apply the unified fixpoint representation of Section 3.2 to obtain the $\sqsubseteq$-least fixpoint as well. To this end, assume additionally that h is $\sqsubseteq$-isotone, and observe that $P_\mu \leq P_\nu$ and $R_\mu \leq R_\nu$ and $Q_\mu \leq Q_\nu$.

Corollary 11. *The following are equivalent:*

1. *κh exists.*
2. *$\kappa h = (P_\nu(Q_\nu), (Q_\nu \curlywedge \mathsf{L}) + Q_\mu)$.*
3. *$d(\mathsf{L})Q_\nu \leq (Q_\nu \curlywedge \mathsf{L}) + Q_\mu + d(P_\nu(Q_\nu))\top$.*

Proof. Let $P =_{\text{def}} P_\nu(Q_\nu)$ and $Q =_{\text{def}} (Q_\nu \curlywedge \mathsf{L}) + Q_\mu$. By Theorems 9 and 10,

$$\begin{aligned} (\nu h \curlywedge \mathsf{L}') + \mu h &= ((P, Q_\nu) \curlywedge (\mathsf{L}, \mathsf{L} \curlywedge \mathsf{H})) + \mu h = (P \curlywedge \mathsf{L}, Q_\nu \curlywedge \mathsf{L} \curlywedge \mathsf{H}) + \mu h \\ &= (P, Q_\nu \curlywedge \mathsf{L}) + (P_\mu(Q_\mu), Q_\mu) = (P + P_\mu(Q_\mu), Q) = (P, Q) \end{aligned}$$

using $P_\mu(Q_\mu) \leq P \leq \mathsf{L}$ and $Q_\nu \leq \mathsf{H}$, and

$$d(\mathsf{L}')\nu h = d(\mathsf{L}, \mathsf{L} \curlywedge \mathsf{H})\nu h = (0, d(\mathsf{L}))(P, Q_\nu) = (d(\mathsf{L})P, d(\mathsf{L})(P + Q_\nu) \curlywedge \mathsf{H}) \;,$$

and

$$\begin{aligned} d(\nu h 0')\top' &= d((P, Q_\nu)(0,0))\top' = d(P + Q_\nu 0, (P + Q_\nu 0) \curlywedge \mathsf{H})\top' \\ &= d(P, P \curlywedge \mathsf{H})\top' = (0, d(P))(\mathsf{L}, \mathsf{H}) = (d(P)\mathsf{L}, d(P)(\mathsf{L} + \mathsf{H}) \curlywedge \mathsf{H}) \\ &= (P, d(P)\top \curlywedge \mathsf{H}) \end{aligned}$$

using (H1), (L3) and Lemma 7.3.

Hence we obtain

$$\begin{aligned}
& d(\mathsf{L}')\nu h \leq (\nu h \curlywedge \mathsf{L}') + \mu h + d(\nu h 0')\top' \\
\Leftrightarrow\ & (d(\mathsf{L})P, d(\mathsf{L})(P + Q_\nu) \curlywedge \mathsf{H}) \leq (P, Q + (d(P)\top \curlywedge \mathsf{H})) \\
\Leftrightarrow\ & d(\mathsf{L})(P + Q_\nu) \curlywedge \mathsf{H} \leq Q + (d(P)\top \curlywedge \mathsf{H}) \\
\Leftrightarrow\ & d(\mathsf{L})Q_\nu \curlywedge \mathsf{H} \leq Q + (d(P)\top \curlywedge \mathsf{H}) \\
\Leftrightarrow\ & d(\mathsf{L})Q_\nu \curlywedge \mathsf{H} \leq Q + d(P)\top \\
\Leftrightarrow\ & d(\mathsf{L})Q_\nu \leq Q + d(P)\top
\end{aligned}$$

using $d(\mathsf{L}) \leq 1$ and $P \leq d(P)\top$. The claim follows by the isomorphism of Section 4.3 and Theorem 3. □

5 Conclusion

This paper shows how to extend the unification of partial, total and general correctness all the way to full recursion. We obtain a common semantics of programs and common laws to reason about programs. It is possible to derive a unified semantics of recursively specified operations.

All of this applies as well to programs specified on pairs of termination information and state transitions. For this pair-based representation the lessons learned from UTP's designs were very helpful. We also observe that reasoning in an axiomatic style fits well into UTP.

Future work will unify pre-post specifications and refinement laws, based on their algebraic treatments in total and general correctness [31, 17, 15], and program transformations between different kinds of recursions, based on their development in total correctness [20].

Acknowledgement. I thank Georg Struth for helpful discussions and the anonymous referees for valuable remarks.

This work was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD).

References

1. C. J. Aarts. Galois connections presented calculationally. Master's thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1992.
2. 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.
3. R. Berghammer and H. Zierer. Relational algebraic semantics of deterministic and nondeterministic programs. *Theoretical Computer Science*, 43:123–147, 1986.
4. G. Birkhoff. *Lattice Theory*, volume XXV of *Colloquium Publications*. American Mathematical Society, third edition, 1967.

5. 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 *LNCS*, pages 553–592. Springer-Verlag, 1979.
6. J.-L. De Carufel and J. Desharnais. Demonic algebra with domain. In R. Schmidt, editor, *Relations and Kleene Algebra in Computer Science*, volume 4136 of *LNCS*, pages 120–134. Springer-Verlag, 2006.
7. Y. Chen. A fixpoint theory for non-monotonic parallelism. *Theoretical Computer Science*, 308(1–3):367–392, November 2003.
8. E. Cohen. Separation and reduction. In R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction*, volume 1837 of *LNCS*, pages 45–59. Springer-Verlag, 2000.
9. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, second edition, 2002.
10. J. Desharnais and G. Struth. Domain axioms for a family of near-semirings. In J. Meseguer and G. Roşu, editors, *Algebraic Methodology and Software Technology*, volume 5140 of *LNCS*, pages 330–345. Springer-Verlag, 2008.
11. J. Desharnais and G. Struth. Modal semirings revisited. In P. Audebaud and C. Paulin-Mohring, editors, *Mathematics of Program Construction*, volume 5133 of *LNCS*, pages 360–387. Springer-Verlag, 2008.
12. E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall, 1976.
13. R. M. Dijkstra. Computation calculus bridging a formalization gap. *Science of Computer Programming*, 37(1–3):3–36, May 2000.
14. 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.
15. S. E. Dunne, I. J. Hayes, and A. J. Galloway. Reasoning about loops in total and general correctness. In A. Butterfield, editor, *Unifying Theories of Programming, Second International Symposium, UTP 2008*, volume 5713 of *LNCS*, pages 62–81. Springer-Verlag, 2010.
16. C. C. Elgot. Matricial theories. *Journal of Algebra*, 42(2):391–421, October 1976.
17. W. Guttmann. General correctness algebra. In R. Berghammer, A. M. Jaoua, and B. Möller, editors, *Relations and Kleene Algebra in Computer Science*, volume 5827 of *LNCS*, pages 150–165. Springer-Verlag, 2009.
18. W. Guttmann. Partial, total and general correctness. In C. Bolduc, J. Desharnais, and B. Ktari, editors, *Mathematics of Program Construction*, volume 6120 of *LNCS*, pages 157–177. Springer-Verlag, 2010.
19. W. Guttmann and B. Möller. Modal design algebra. In S. Dunne and W. Stoddart, editors, *Unifying Theories of Programming*, volume 4010 of *LNCS*, pages 236–256. Springer-Verlag, 2006.
20. W. Guttmann and B. Möller. Normal design algebra. *Journal of Logic and Algebraic Programming*, 79(2):144–173, February 2010.
21. C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580/583, October 1969.
22. C. A. R. Hoare and J. He. *Unifying theories of programming*. Prentice Hall Europe, 1998.
23. P. Höfner and B. Möller. An algebra of hybrid systems. *Journal of Logic and Algebraic Programming*, 78(2):74–97, January 2009.
24. D. Jacobs and D. Gries. General correctness: A unification of partial and total correctness. *Acta Informatica*, 22(1):67–83, April 1985.

25. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, May 1994.
26. D. Kozen. On Hoare logic and Kleene algebra with tests. *ACM Transactions on Computational Logic*, 1(1):60–76, July 2000.
27. B. Möller. Kleene getting lazy. *Science of Computer Programming*, 65(2):195–214, March 2007.
28. 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 *LNCS*, pages 200–211. Springer-Verlag, 2006.
29. B. C. Moszkowski. A complete axiomatization of Interval Temporal Logic with infinite time. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*, pages 241–252. IEEE, 2000.
30. G. Nelson. A generalization of Dijkstra's calculus. *ACM Transactions on Programming Languages and Systems*, 11(4):517–561, October 1989.
31. J. von Wright. Towards a refinement algebra. *Science of Computer Programming*, 51(1–2):23–45, May 2004.