Non-Termination in Unifying Theories of Programming

Walter Guttmann

Abteilung Programmiermethodik und Compilerbau
Universität Ulm, 89069 Ulm, Germany
walter.guttmann@uni-ulm.de

Abstract. Within the *Unifying Theories of Programming* framework, program initiation and termination has been modelled by introducing a pair of variables in order to satisfy the required algebraic properties. We replace these variables with the improper value $\perp$ that is frequently used to denote undefinedness. Both approaches are proved isomorphic using the relation calculus, and the existing operations and laws are carried over. We split the isomorphism by interposing "intuitive" relations.

1 Introduction

The *Unifying Theories of Programming* framework [1], hereafter abbreviated as UTP, takes a relational view on semantics: The meaning of a non-deterministic, imperative program is described by a predicate relating the initial and final values of its observable variables. This intuitive view, however, turns out to be too simplistic. Certain laws observed in practice for programs are not satisfied by arbitrary predicates, e.g., the zero laws of sequential composition:

$$true; P \;=\; true \;=\; P; true,$$

where *true* is the meaning of the totally unpredictable program. To rectify the situation one might, e.g., redefine the sequential composition operator, but we appreciate a solution that retains its meaning as relational composition. The class of predicates employed to describe programs is instead restricted by so-called healthiness conditions.

The definition of the proper subclass depends on the introduction of a free variable – actually a pair ok and ok' denoting the initial and final value – to model that a program has been started and terminated. Such information about the definedness of a program is often denoted by the distinguished, improper value $\perp$ that, depending on the context, represents an undefined variable, expression, or state of execution.

In Sect. 3 we show how these two views coincide. To this end, we replace ok and ok' with $\perp$ to deal with non-termination in UTP and prove that the new and original approaches are isomorphic. This alleviates the objection to introduce the improper value put forward by [2]. The formal development is carried out in the relation calculus [3].

We continue the investigation in Sect. 4 by proving that both views are in one-to-one correspondence with the intuitive relational reading of predicates mentioned above. The latter class is installed as a seamless intermediate, thus splitting the isomorphism.

First of all, we describe the principles of [1] we will need in the rest of the paper. We also state several structural properties.

2 Basics of Unifying Theories of Programming

In practice one can observe about the execution of a program the initial and final values of its variables. UTP therefore models a program as a predicate P whose free variables come from a set αP, its alphabet, that is partitioned according to

$$\alpha P \;=\; in\alpha P \cup out\alpha P$$

into a set of undashed variables $in\alpha P$ standing for initial values and a set of dashed variables $out\alpha P$ standing for final values. We will focus on the case $out\alpha P = (in\alpha P)'$, the extension to the inhomogeneous case being uncomplicated. Unless stated otherwise we will use $v = in\alpha P$ and $v' = out\alpha P$ treating both v and v' as a single variable rather than a set of variables. Thus, v takes as its value a tuple of values, one for each element of $in\alpha P$, and similarly does v'.

Every predicate P may be identified with the set $\{(v, v') : P(v, v')\}$ of all pairs of observations that satisfy it [1]. We will adopt this relational attitude at the end of this section.

2.1 Combining Predicates

The *non-deterministic choice* between predicates P and Q is just

$$P \vee Q,$$

so we do not have to introduce a new notation.

The *conditional* uses a predicate without dashed variables b, to choose between predicates P and Q as per

$$P \lhd b \rhd Q \;=_{\mathrm{def}}\; (b \wedge P) \vee (\neg b \wedge Q).$$

The *sequential composition* of predicates P and Q with a common alphabet is defined by

$$P; Q \;=_{\mathrm{def}}\; \exists v_0 : P[v_0/v'] \wedge Q[v_0/v],$$

where $P[v_0/v']$ denotes the substitution of v_0 for v' in P. Attention has to be paid when a substitution is applied to $P; Q$ since v' in P and v in Q are replaced by v_0 which is bound by the existential quantifier. For example, if e does not depend on v and v_0, $(P; Q)[e/v'] = P; (Q[e/v'])$.

2.2 Designs

Given predicates P_1 and P_2 that do not contain the auxiliary variables ok and ok' a *design* is defined as

$$P_1 \vdash P_2 \;=_{\text{def}}\; ok \wedge P_1 \implies ok' \wedge P_2.$$

Its informal reading is that "if the program starts in a state satisfying P_1, it will terminate, and on termination P_2 will be true".

The *no-operation* program is the design

$$\mathbb{I}_D \;=_{\text{def}}\; true \vdash v = v',$$

where v and v' vary depending on the context.

The *assignment* of the value of an expression e to a variable x is the design

$$x := e \;=_{\text{def}}\; true \vdash x' = e \wedge w' = w$$

where w denotes all variables except x.

2.3 Healthiness Conditions

The restriction to designs does not yield all required algebraic properties. These are enforced by the four *healthiness conditions* imposed on a predicate P:

H1. $P = (ok \implies P)$.
H2. $\big[P[false/ok'] \implies P[true/ok']\big]$.
H3. $P = P; \mathbb{I}_D$.
H4. $P; true = true$.

The outer brackets in H2 denote the universal closure. A characterisation of H1 is given by [1, Theorem 3.2.2] stating

$$P \text{ is H1} \iff (true; P = true) \wedge (\mathbb{I}_D; P = P),$$

only H2 lacks a convincing algebraic formulation, as already remarked by [2]. We solve this problem by restricting our notion of design.

Proposition 1. *Every predicate that is H3 is also H2.*

Proof. Let P be a predicate that is H3 and has v and ok as its variables, then $P[false/ok'] = (P; \mathbb{I}_D)[false/ok'] = P; (\mathbb{I}_D[false/ok']) \implies P; (\mathbb{I}_D[true/ok']) = (P; \mathbb{I}_D)[true/ok'] = P[true/ok']$ since implication is monotonic and $\mathbb{I}_D$ is H2.

Designs are characterised by [1, Theorem 3.2.3] as just those predicates that are H1 and H2, thus by Proposition 1 a predicate that is H1 and H3 is a design, too. Following the terminology of [2] we call such a predicate a *normal* design. Finally, a predicate that is H4 is called *feasible*. The relations of predicates satisfying the various laws are displayed in Fig. 1.

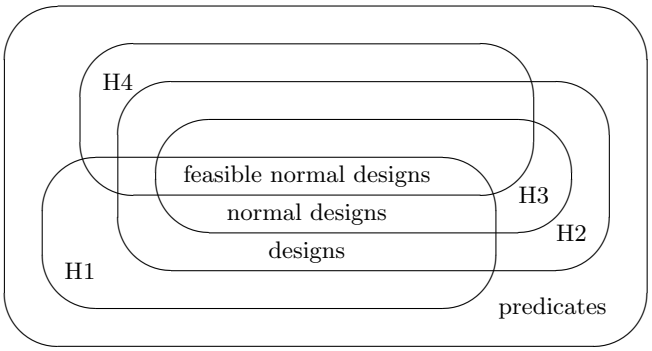

Fig. 1. Predicates satisfying different healthiness conditions

2.4 The Complete Lattice of Predicates

Using conjunction and disjunction as the lattice operators, predicates form a complete lattice. The predicates satisfying any combination of H1, H2, and H3 (but not H4) form a complete sub-lattice [4]. Rather unfortunately, the underlying order employed by UTP is the *reverse* implication ordering where $P \sqsubseteq Q$ if and only if $[Q \Longrightarrow P]$. Thus, the weakest fixed point μF of a monotonic function F from predicates to predicates exists and it is used to define *recursion*:

$$\mu F \ =_{\text{def}} \ \bigvee \{X : [X \Longrightarrow F(X)]\} .$$

With recursion, all constructs are in place to define the programming language of UTP. Following [1, Table 5.0.1], a program may be constructed from the constants *true* and assignment, the combinators introduced in Sect. 2.1, and recursion. From [1, Chapter 5.4] it follows that F is continuous if it is composed from these program constructs.

The program constructs preserve H1–H4 with one exception: In general, μF need not be feasible even if F is continuous and preserves all healthiness conditions, including feasibility. A counterexample is given by $\mu F_0 = \neg ok$ for

$$F_0 \ =_{\text{def}} \ P_1 \vdash P_2 \ \mapsto \ \mathit{true} \vdash x + x' > \min\{x + x' : P_1 \Longrightarrow P_2\},$$

with the natural numbers as the domain of x and x'. Nevertheless, if F is composed from program constructs, μF is feasible.

2.5 Predicates and Relations

Up to now we have used the language of predicates since this is the formalism employed by [1]. We will retain these terms whenever we discuss the connection to UTP but will switch to relational terms [3] for the formal development.

To establish a common notation we will denote, for relations R and S, union as $R \vee S$, intersection as $R \wedge S$, inclusion as $R \leq S$, composition as $R; S$, transposition as R^{T}, and complement as $\overline{R}$, and the constants empty relation as $\perp\!\!\!\perp$,

identity relation as $\mathbb{I}$, and universal relation as $\mathbb{T}$. Unary operators have highest precedence, followed by composition, union and intersection, conditional, the formation of designs, finally equality and inclusion with lowest precedence.

A relation R is a *vector* if $R = R; \mathbb{T}$. The design $P_1 \vdash P_2$ is a normal design if and only if P_1 is a vector [1, Theorem 3.2.4].

3 Representing Non-Termination with Improper Values

Given a predicate P, for any choice of values for v either there exists a choice for v' that satisfies P or not. We may say accordingly that, for the given assignment to v, P is defined or not. Recall that in UTP v' actually represents a set of variables, yet there is no notion of separate definedness for each variable. One cannot state that, for the given assignment to v, certain variables represented by v' are defined whereas others are not. When we introduce the improper value $\bot$ we therefore do not extend the range of every variable by distinct instances of $\bot$ but add just one value, respectively, to the range of v and v'. This corresponds to the construction of the *smash-product* of semantic domains.

3.1 Bottom Predicates

In the following we need to distinguish two kinds of predicates.

1. Presented in Sects. 2.2 and 2.3, (feasible) normal designs contain the auxiliary variables ok and ok'. We assume that $\bot$ denotes a value that is not in the range of the variables of designs.
2. We call a predicate that does not contain ok and ok' and has the improper value $\bot$ in the range of its variables a *$\bot$-predicate*. Two special $\bot$-predicates are the vector $\mathbb{V} =_{\text{def}} (v = \bot)$ and $\mathbb{V}^{\mathsf{T}} = (v' = \bot)$.

We will use the same symbols to denote the operations on $\bot$-predicates and designs, relying on the context for disambiguation.

A design $P_1 \vdash P_2$ is composed of predicates P_1 and P_2 that neither contain ok and ok', nor $\bot$. Let us call predicates of this type *basic relations*. They play a formal role in the following development, but will receive additional interpretation in Sect. 4 as "intuitive" relations. To facilitate the construction of $\bot$-predicates we introduce two operations that convert to and from basic relations.

Definition 2. *The operations $\cdot^+$ from basic relations to $\bot$-predicates and $\cdot^-$ from $\bot$-predicates to basic relations are lattice homomorphisms and satisfy the axioms*

$$(P;Q)^+ = P^+;Q^+ \qquad P^{+-} = P$$
$$\overline{P}^- = \overline{P^-} \qquad P^{-+} = P \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^{\mathsf{T}}}$$

When a relation is viewed as a set of pairs, the intention for $\cdot^+$ is the identity, and $\cdot^-$ removes all pairs having $\bot$ as one component. The following development nevertheless uses just the stated axioms and the homomorphism qualities, namely monotonicity and distributivity over union and intersection. Lemma 3 provides derived properties of the operations for subsequent use.

Lemma 3. *1. The lower adjoint $\cdot^+$ and the upper adjoint $\cdot^-$ form a Galois connection, $\bot\!\!\bot^+ = \bot\!\!\bot$, $\bot\!\!\bot^- = \bot\!\!\bot$, $\top^- = \top$, $\top^+ = \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}$, $\overline{\mathbb{V}}^- = \top$, $\mathbb{I}^- = \mathbb{I}$, and $\mathbb{I}^+ = \mathbb{I} \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}$.*
2. $\overline{P^+} = P^+ \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}$ and $P^+ \wedge Q^{-+} = P^+ \wedge Q$.
3. $\overline{P^+}; \top = \top$ and $\overline{P^+; \top} \geq \mathbb{V}$.
4. $(P^+; Q)^- = P; Q^-$ and $(P^+; \top)^- = P; \top$.
5. $P^+ \wedge (Q; R^-)^+ = P^+ \wedge Q^+; R$ and $P^+ \wedge \overline{Q; R^-}^+ = P^+ \wedge \overline{Q^+; R}$.
6. If B is a vector, $P^+ \wedge B^+ = P^+ \wedge B^+; \top$ and $P^+ \wedge \overline{B}^+ = P^+ \wedge \overline{B^+; \top}$.
7. If B is a vector, $(P \lhd B \rhd Q)^+ = P^+ \lhd B^+; \top \rhd Q^+$.
8. If P is a vector, $\overline{P^{-+}; \top} = \overline{P} \vee \mathbb{V}$.
9. $(P; \top)^+; \top = P^+; \top$.

Proof.

1. $P^+ \leq Q \Longrightarrow P = P^{+-} \leq Q^- \Longrightarrow P^+ \leq Q^{-+} = Q \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}} \leq Q$.
   $\bot\!\!\bot \leq \bot\!\!\bot^- \Longrightarrow \bot\!\!\bot^+ \leq \bot\!\!\bot \Longrightarrow \bot\!\!\bot^+ = \bot\!\!\bot$, hence $\bot\!\!\bot^- = \bot\!\!\bot^{+-} = \bot\!\!\bot$.
   $\top^+ \leq \top \Longrightarrow \top \leq \top^- \Longrightarrow \top = \top^-$, hence $\top^+ = \top^{-+} = \top \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}} = \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}$, hence $\top = \top^{+-} = (\overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}})^- \leq \overline{\mathbb{V}}^-$.
   $\mathbb{I}^- = (\mathbb{I}^-; \mathbb{I})^{+-} = (\mathbb{I}^{-+}; \mathbb{I}^+)^- = ((\mathbb{I} \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}); \mathbb{I}^+)^- = (\overline{\mathbb{V}} \wedge \mathbb{I}; (\overline{\mathbb{V}} \wedge \mathbb{I}^+))^- = (\overline{\mathbb{V}} \wedge \mathbb{I}^+)^- = \overline{\mathbb{V}}^- \wedge \mathbb{I}^{+-} = \mathbb{I}$, hence $\mathbb{I}^+ = \mathbb{I}^{-+} = \mathbb{I} \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}$.
2. $P^+ = P^{+-+} = P^+ \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}$, hence $P^+ \wedge Q^{-+} = P^+ \wedge Q \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}} = P^+ \wedge Q$.
3. By (2), $\overline{P^+}; \top = \overline{P^+ \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}}; \top \geq \overline{\overline{\mathbb{V}^\mathsf{T}}}; \top = \mathbb{V}^\mathsf{T}; \top = \top$ and $\overline{P^+; \top} = \overline{(P^+ \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}); \top} \geq \overline{\overline{\mathbb{V}}; \top} = \overline{\overline{\mathbb{V}}} = \mathbb{V}$.
4. $(P^+; Q)^- = (P^+; Q)^{-+-} = (P^+; Q \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}})^- = ((\overline{\mathbb{V}} \wedge P^+); (Q \wedge \overline{\mathbb{V}^\mathsf{T}}))^- = ((P^+ \wedge \overline{\mathbb{V}^\mathsf{T}}); (Q \wedge \overline{\mathbb{V}^\mathsf{T}}))^- = (P^+; (Q \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}))^- = (P^+; Q^{-+})^- = P; Q^-$ by (2), hence $(P^+; \top)^- = P; \top^- = P; \top$ by (1).
5. By (4) and (2), $P^+ \wedge (Q; R^-)^+ = P^+ \wedge (Q^+; R)^{-+} = P^+ \wedge Q^+; R$ and $P^+ \wedge \overline{Q; R^-}^+ = P^+ \wedge \overline{(Q^+; R)^-}^+ = P^+ \wedge \overline{Q^+; R}^{-+} = P^+ \wedge \overline{Q^+; R}$.
6. By (1) and (5), $P^+ \wedge B^+ = P^+ \wedge (B; \top)^+ = P^+ \wedge (B; \top^-)^+ = P^+ \wedge B^+; \top$ and $P^+ \wedge \overline{B}^+ = P^+ \wedge \overline{B; \top}^+ = P^+ \wedge \overline{B; \top^-}^+ = P^+ \wedge \overline{B^+; \top}$.
7. $(P \lhd B \rhd Q)^+ = (B^+ \wedge P^+) \vee (\overline{B}^+ \wedge Q^+) = (B^+; \top \wedge P^+) \vee (\overline{B^+; \top} \wedge Q^+) = P^+ \lhd B^+; \top \rhd Q^+$ by (6).
8. $\overline{P^{-+}; \top} = \overline{(P \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}); \top} = \overline{P \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}; \top} = \overline{P} \vee \mathbb{V} \vee \overline{\overline{\mathbb{V}^\mathsf{T}}; \top} = \overline{P} \vee \mathbb{V}$.
9. $P^+; \top \leq (P; \top)^+; \top = P^+; \top^+; \top \leq P^+; \top$.

3.2 From Auxiliary Variables to Improper Values

The transition from normal designs to $\bot$-predicates is accomplished by the function $\mathcal{I}$, whose effect on the constructs of UTP introduced in Sect. 2 is given by Lemma 5.

Definition 4. *The mapping $\mathcal{I}$ from normal designs to $\bot$-predicates is*

$$\mathcal{I} =_{\text{def}} P_1 \vdash P_2 \mapsto \overline{{P_1}^+; \top} \vee {P_2}^+.$$

The no-operation program is transformed to $\mathbb{I}_\bot =_{\text{def}} \mathcal{I}(\mathbb{I}_D)$.

Lemma 5. *For a vector B and normal designs P and Q,*

1. $\mathcal{I}(P \vee Q) = \mathcal{I}(P) \vee \mathcal{I}(Q)$,
2. $\mathcal{I}(P \lhd B \rhd Q) = \mathcal{I}(P) \lhd B^+; \mathbb{T} \rhd \mathcal{I}(Q)$,
3. $\mathcal{I}(P; Q) = \mathcal{I}(P); \mathcal{I}(Q)$,
4. $\mathbb{I}_\perp = \mathbb{I} \vee \mathbb{V}$, *and*
5. $\mathcal{I}(\mathbb{T}) = \mathbb{T}$.

Proof. We will use [1, Theorem 3.1.4] that describes non-deterministic choice, conditional, and sequential composition of two designs as a design. Let $P_1 \vdash P_2$ and $Q_1 \vdash Q_2$ be normal designs.

1. By Lemma 3(6), $\mathcal{I}((P_1 \vdash P_2) \vee (Q_1 \vdash Q_2)) = \mathcal{I}(P_1 \wedge P_2 \vdash Q_1 \vee Q_2) = \overline{(P_1{}^+ \wedge Q_1{}^+); \mathbb{T}} \vee (P_2 \vee Q_2)^+ = \overline{(P_1{}^+ \wedge Q_1{}^+; \mathbb{T}); \mathbb{T}} \vee P_2{}^+ \vee Q_2{}^+ = \overline{P_1{}^+; \mathbb{T} \wedge Q_1{}^+; \mathbb{T}} \vee P_2{}^+ \vee Q_2{}^+ = \overline{P_1{}^+; \mathbb{T}} \vee P_2{}^+ \vee \overline{Q_1{}^+; \mathbb{T}} \vee Q_2{}^+ = \mathcal{I}(P_1 \vdash P_2) \vee \mathcal{I}(Q_1 \vdash Q_2)$.
2. By Lemma 3(7), $\mathcal{I}((P_1 \vdash P_2) \lhd B \rhd (Q_1 \vdash Q_2)) = \mathcal{I}(P_1 \lhd B \rhd Q_1 \vdash P_2 \lhd B \rhd Q_2) = \overline{(P_1 \lhd B \rhd Q_1)^+; \mathbb{T}} \vee (P_2 \lhd B \rhd Q_2)^+ = \overline{(P_1{}^+ \lhd B^+; \mathbb{T} \rhd Q_1{}^+); \mathbb{T}} \vee (P_2{}^+ \lhd B^+; \mathbb{T} \rhd Q_2{}^+) = \overline{(P_1{}^+; \mathbb{T} \lhd B^+; \mathbb{T} \rhd Q_1{}^+; \mathbb{T})} \vee (P_2{}^+ \lhd B^+; \mathbb{T} \rhd Q_2{}^+) = \overline{P_1{}^+; \mathbb{T}} \vee P_2{}^+ \lhd B^+; \mathbb{T} \rhd \overline{Q_1{}^+; \mathbb{T}} \vee Q_2{}^+ = \mathcal{I}(P_1 \vdash P_2) \lhd B^+; \mathbb{T} \rhd \mathcal{I}(Q_1 \vdash Q_2)$.
3. $\overline{P_1{}^+; \mathbb{T}} = \overline{P_1{}^+; \mathbb{T}; \mathbb{T}; \mathbb{V}} \leq \overline{P_1{}^+; \mathbb{T}; Q_1{}^+; \mathbb{T}} \leq \overline{P_1{}^+; \mathbb{T}; (Q_1{}^+; \mathbb{T} \vee Q_2{}^+)} \leq \overline{P_1{}^+; \mathbb{T}; \mathbb{T}} = \overline{P_1{}^+; \mathbb{T}}$ by Lemma 3(3).
   $P_1{}^+ \wedge \overline{P_2{}^+; \overline{Q_1{}^+; \mathbb{T}}} = P_1{}^+ \wedge \overline{P_2; \overline{Q_1{}^+; \mathbb{T}}^{-+}} = P_1{}^+ \wedge \overline{P_2; \overline{Q_1; \mathbb{T}}^+} = P_1{}^+ \wedge \overline{P_2; \overline{Q_1}^+}$ by Lemma 3(5&4).
   Therefore, $\mathcal{I}(P_1 \vdash P_2); \mathcal{I}(Q_1 \vdash Q_2) = (\overline{P_1{}^+; \mathbb{T}} \vee P_2{}^+); (\overline{Q_1{}^+; \mathbb{T}} \vee Q_2{}^+) = \overline{P_1{}^+; \mathbb{T}}; (\overline{Q_1{}^+; \mathbb{T}} \vee Q_2{}^+) \vee P_2{}^+; (\overline{Q_1{}^+; \mathbb{T}} \vee Q_2{}^+) = \overline{P_1{}^+; \mathbb{T}} \vee P_2{}^+; \overline{Q_1{}^+; \mathbb{T}} \vee P_2{}^+; Q_2{}^+ = P_1{}^+; \mathbb{T} \wedge \overline{P_2{}^+; \overline{Q_1{}^+; \mathbb{T}}} \vee (P_2; Q_2)^+ = (P_1{}^+ \wedge \overline{P_2{}^+; \overline{Q_1{}^+; \mathbb{T}}}); \mathbb{T} \vee (P_2; Q_2)^+ = (P_1{}^+ \wedge \overline{P_2; \overline{Q_1}^+}); \mathbb{T} \vee (P_2; Q_2)^+ = \mathcal{I}(P_1 \wedge \overline{P_2; \overline{Q_1}} \vdash P_2; Q_2) = \mathcal{I}((P_1 \vdash P_2); (Q_1 \vdash Q_2))$.
4. By Lemma 3(1&8), $\mathbb{I}_\perp = \mathcal{I}(\mathbb{I}_D) = \mathcal{I}(\mathbb{T} \vdash \mathbb{I}) = \overline{\mathbb{T}^+; \mathbb{T}} \vee \mathbb{I}^+ = \overline{\mathbb{T}^{-+}; \mathbb{T}} \vee \mathbb{I}^+ = \overline{\mathbb{T}} \vee \mathbb{V} \vee (\mathbb{I} \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}) = \mathbb{V} \vee (\mathbb{I} \wedge \overline{\mathbb{V}^\mathsf{T}}) = \mathbb{V} \vee (\mathbb{I} \wedge \overline{\mathbb{V}^\mathsf{T}})^\mathsf{T} = \mathbb{V} \vee (\mathbb{I} \wedge \overline{\mathbb{V}}) = \mathbb{V} \vee \mathbb{I}$.
5. By Lemma 3(1), $\mathcal{I}(\mathbb{T}) = \mathcal{I}(\perp\!\!\!\perp \vdash \perp\!\!\!\perp) = \overline{\perp\!\!\!\perp^+; \mathbb{T}} \vee \perp\!\!\!\perp^+ = \overline{\perp\!\!\!\perp; \mathbb{T}} \vee \perp\!\!\!\perp = \overline{\perp\!\!\!\perp} = \mathbb{T}$.

Corollary 6. *$\mathcal{I}$ is monotonic.*

Proof. $\mathcal{I}$ is a join homomorphism by Lemma 5(1), therefore an order homomorphism [4].

3.3 Healthiness Conditions

We introduce healthiness conditions similar to those presented in Sect. 2.3 for $\perp$-predicates. The predicates in the image of $\mathcal{I}$ satisfy these laws. Lemma 9 states consequences of various combinations of the healthiness conditions.

Definition 7. *A $\bot$-predicate P is a normal $\bot$-predicate if it satisfies the left zero law $\top; P = \top$, and the unit laws $\mathbb{I}_\bot; P = P = P; \mathbb{I}_\bot$. P is called feasible if it satisfies the right zero law $P; \top = \top$.*

Lemma 8. *$\mathcal{I}(P)$ is a (feasible) normal $\bot$-predicate for every (feasible) normal design P.*

Proof. $\mathbb{I}_\bot; \mathcal{I}(P) = \mathcal{I}(\mathbb{I}_D); \mathcal{I}(P) = \mathcal{I}(\mathbb{I}_D; P) = \mathcal{I}(P) = \mathcal{I}(P; \mathbb{I}_D) = \mathcal{I}(P); \mathcal{I}(\mathbb{I}_D) = \mathcal{I}(P); \mathbb{I}_\bot$ by Lemma 5(3). $\top; \mathcal{I}(P) = \mathcal{I}(\top); \mathcal{I}(P) = \mathcal{I}(\top; P) = \mathcal{I}(\top) = \top = \mathcal{I}(\top) = \mathcal{I}(P; \top) = \mathcal{I}(P); \mathcal{I}(\top) = \mathcal{I}(P); \top$ by Lemma 5(5&3).

Lemma 9. *Let P be a $\bot$-predicate.*

1. *If P satisfies the left zero and left unit laws, $\mathbb{V} \le P$.*
2. *If P satisfies the right unit law, $\overline{P}; \top \wedge P \le \overline{\mathbb{V}^\mathsf{T}}$.*
3. *If P is a vector or P satisfies the right unit law, $P^-; \top = (P; \top)^-$.*
4. *If P satisfies the right unit and right zero laws, $P^-; \top = \top$.*

Proof. We will use Lemma 5(4) in the first three parts of the proof.

1. $\mathbb{V} = \mathbb{V} \wedge \top = \mathbb{V} \wedge \top; P = \mathbb{V}; P \le (\mathbb{I} \vee \mathbb{V}); P = \mathbb{I}_\bot; P = P$.
2. $\overline{P}; \top \wedge P \le (\overline{P} \wedge P; \top^\mathsf{T}); (\top \wedge \overline{P^\mathsf{T}}; P) \le \top; \overline{P^\mathsf{T}}; P \le \top; \overline{\mathbb{I}_\bot{}^\mathsf{T}} \le \top; \overline{\mathbb{V}^\mathsf{T}} = \overline{\mathbb{V}^\mathsf{T}}$ by the Dedekind and Schröder rules [3].
3. If P is a vector, $P; \mathbb{I}_\bot = P; (\mathbb{I} \vee \mathbb{V}) = P; \mathbb{I} \vee P; \top; \mathbb{V} = P \vee P; \top = P$, thus P satisfies the right unit law. $\mathbb{I}_\bot; \overline{\mathbb{V}} = (\mathbb{I} \vee \mathbb{V}); \overline{\mathbb{V}} = \mathbb{I}; \overline{\mathbb{V}} \vee \mathbb{V}; \overline{\mathbb{V}} = \overline{\mathbb{V}}; \overline{\mathbb{V}} \vee \mathbb{V}; \overline{\mathbb{V}} = \top; \overline{\mathbb{V}} = \top$, hence, $(P; \top)^- = (P; \mathbb{I}_\bot; \overline{\mathbb{V}})^- = (P; \overline{\mathbb{V}})^- \wedge \overline{\mathbb{V}}^- = ((P \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\mathsf{T}}); \top)^- = (P^{-+}; \top)^- = P^-; \top$ by Lemma 3(1&4).
4. By (3) and Lemma 3(1), $P^-; \top = (P; \top)^- = \top^- = \top$.

3.4 From Improper Values to Auxiliary Variables

We now move forward to reverse the mapping to $\bot$-predicates by the function $\mathcal{H}$. The predicates in the image of $\mathcal{H}$ satisfy the healthiness conditions of Sect. 2.3.

Definition 10. *The mapping $\mathcal{H}$ from normal $\bot$-predicates to designs is*

$$\mathcal{H} \;=_{\text{def}}\; P \;\mapsto\; (\overline{P}; \top)^- \vdash P^-.$$

Lemma 11. *$\mathcal{H}$ is monotonic.*

Proof. $P \le Q$ implies both $(\overline{Q}; \top)^- \le (\overline{P}; \top)^-$ and $(\overline{Q}; \top)^- \wedge P^- \le P^- \le Q^-$. By [1, Theorem 3.1.2], $(\overline{P}; \top)^- \vdash P^- \le (\overline{Q}; \top)^- \vdash Q^-$, thus $\mathcal{H}(P) \le \mathcal{H}(Q)$.

Lemma 12. *$\mathcal{H}(P)$ is a (feasible) normal design for every (feasible) normal $\bot$-predicate P.*

Proof. By Lemma 9(3), $(\overline{P}; \top)^-; \top = (\overline{P}; \top; \top)^- = (\overline{P}; \top)^-$, thus the design $\mathcal{H}(P)$ is a normal design by [1, Theorem 3.2.4]. If P is a feasible normal $\bot$-predicate, $\mathcal{H}(P); \top = ((\overline{P}; \top)^- \vdash P^-); (\bot \vdash \bot) = (\overline{P}; \top)^- \wedge \overline{P^-}; \overline{\top} \vdash P^-; \bot = (\overline{P}; \top)^- \wedge \overline{\top} \vdash \bot = \bot \vdash \bot = \top$ by [1, Theorem 3.1.4] and Lemma 9(4), hence $\mathcal{H}(P)$ is a feasible normal design.

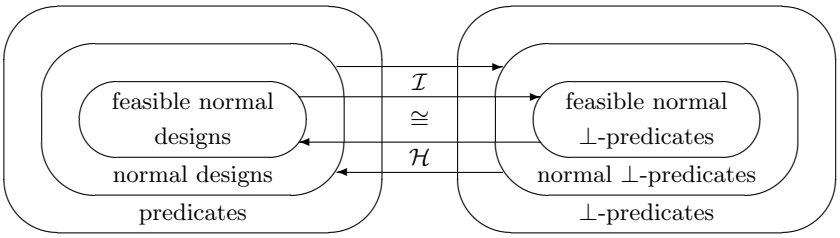

Fig. 2. Isomorphism mapping between (feasible) normal designs and (feasible) normal $\bot$-predicates

3.5 Isomorphism

The main result of this section shows that $\mathcal{H}$ exactly undoes the effect of $\mathcal{I}$. Therefore, sharpening Lemma 8, the image of $\mathcal{I}$ for (feasible) normal designs consists precisely of the (feasible) normal $\bot$-predicates. Analogously, sharpening Lemma 12, the image of $\mathcal{H}$ for (feasible) normal $\bot$-predicates consists precisely of the (feasible) normal designs. The situation is displayed in Fig. 2.

Theorem 13. *$\mathcal{I}$ and $\mathcal{H}$ are inverse to each other.*

Proof. By Lemmas 8 and 12 the compositions are well-defined.
Let $P_1 \vdash P_2$ be a normal design, then $\overline{\mathcal{I}(P_1 \vdash P_2)}; \top = \overline{\overline{{P_1}^+; \top} \vee {P_2}^+}; \top = ({P_1}^+; \top \wedge \overline{{P_2}^+}); \top = {P_1}^+; \top \wedge \overline{{P_2}^+}; \top = {P_1}^+; \top$ by Lemma 3(3).
Therefore, by Lemma 3(4), $\mathcal{H}(\mathcal{I}(P_1 \vdash P_2)) = (\overline{\mathcal{I}(P_1 \vdash P_2)}; \top)^- \vdash \mathcal{I}(P_1 \vdash P_2)^- = ({P_1}^+; \top)^- \vdash \overline{{P_1}^+; \top}^- \vee {P_2}^{+-} = P_1; \top \vdash \overline{P_1; \top} \vee P_2 = P_1 \vdash \overline{P_1} \vee P_2 = P_1 \vdash P_2$.
Conversely, let P be a normal $\bot$-predicate, then, by Lemmas 3(8) and 9(2&1), $\mathcal{I}(\mathcal{H}(P)) = \mathcal{I}((\overline{P}; \top)^- \vdash P^-) = \overline{(\overline{P}; \top)^{-+}; \top \vee P^{-+}} = \overline{\overline{P}; \top} \vee \mathbb{V} \vee (P \wedge \overline{\mathbb{V}} \wedge \overline{\mathbb{V}^\top}) = \overline{\overline{P}; \top \vee \mathbb{V} \vee (P \wedge \overline{\mathbb{V}^\top})} = \overline{\overline{P}; \top \vee \mathbb{V} \vee P} = \overline{\overline{P}; \top \wedge \overline{P}} = P$.

Corollary 14. *The complete lattice of normal designs is isomorphic to the complete lattice of normal $\bot$-predicates. Non-deterministic choice, conditional, sequential composition, and recursion are simulated as*

$$\begin{aligned} \mathcal{I}(\mathcal{H}(P) \vee \mathcal{H}(Q)) &= P \vee Q, \\ \mathcal{I}(\mathcal{H}(P) \lhd B \rhd \mathcal{H}(Q)) &= P \lhd B^+; \top \rhd Q, \\ \mathcal{I}(\mathcal{H}(P); \mathcal{H}(Q)) &= P; Q, \\ \mathcal{I}(\mu F) &= \mu F_\bot, \end{aligned}$$

where $F_\bot =_{\text{def}} \mathcal{I} \circ F \circ \mathcal{H}$.

Proof. Together with Corollary 6 and Lemma 11 follows the isomorphism [4]. The simulation of non-deterministic choice, conditional, and sequential composition is a consequence along with the distributivity of $\mathcal{I}$ granted by Lemma 5.

Being an isomorphism of complete lattices, $\mathcal{I}$ even distributes over arbitrary disjunction. For the simulation of recursion, we therefore have

$$\begin{aligned}\mathcal{I}(\mu F) &= \mathcal{I}(\bigvee \{X : [X \leq F(X)]\}) = \bigvee \{\mathcal{I}(X) : [X \leq F(X)]\} \\ &= \bigvee \{\mathcal{I}(X) : [\mathcal{H}(\mathcal{I}(X)) \leq F(\mathcal{H}(\mathcal{I}(X)))]\} \\ &= \bigvee \{Y : [\mathcal{H}(Y) \leq F(\mathcal{H}(Y))]\} = \bigvee \{Y : [Y \leq \mathcal{I}(F(\mathcal{H}(Y)))]\} \\ &= \bigvee \{Y : [Y \leq F_\perp(Y)]\} = \mu F_\perp.\end{aligned}$$

by the isomorphism, the surjectivity of $\mathcal{I}$, Corollary 6 and Lemma 11.

4 Representing Non-Termination Intuitively

To every predicate P modelling a program there is a corresponding relation R consisting of just those tuples (v, v') with $P(v, v')$. The intuitive reading of R is that $(v, v') \in R$ if and only if it is possible to observe the final values v' for the variables of the program, when they have been initialised with v. For a given initial assignment v, non-determinism is thus modelled by having more than one v' with $(v, v') \in R$, and non-termination by having no such v'.

In this section we show that these "intuitive" relations are in one-to-one correspondence with feasible normal designs and feasible normal $\perp$-predicates. To this end, we will split the isomorphisms $\mathcal{I}$ and $\mathcal{H}$ in two. Although intuitive relations form a complete lattice and the resulting functions are bijective, they do not preserve the lattice structure of normal designs, nor the structure of feasible normal designs.

In the following, an *intuitive relation* is a predicate that does not contain ok and ok' and does not have $\perp$ in the range of its variables. This distinguishes intuitive relations from both designs and $\perp$-predicates.

4.1 Eliminating Auxiliary Variables

The transition from feasible normal designs to intuitive relations and back is accomplished by the functions $\mathcal{I}_d$ and $\mathcal{H}_d$. Although they are inverse to each other, they do not preserve the structure since they are not even monotonic.

Definition 15. *The mapping $\mathcal{I}_d$ from feasible normal designs to intuitive relations is*

$$\mathcal{I}_d \;=_{\text{def}}\; P_1 \vdash P_2 \;\mapsto\; P_1 \wedge P_2.$$

The mapping $\mathcal{H}_d$ from intuitive relations to feasible normal designs is

$$\mathcal{H}_d \;=_{\text{def}}\; P \;\mapsto\; P; \top \vdash P.$$

The normal design $\mathcal{H}_d(P)$ is feasible since $\mathcal{H}_d(P); \top = (P; \top \vdash P); (\perp\!\!\!\perp \vdash \perp\!\!\!\perp) = P; \top \wedge \overline{P; \top} \vdash P; \perp\!\!\!\perp = \perp\!\!\!\perp \vdash \perp\!\!\!\perp = \top$ by [1, Theorem 3.1.4].

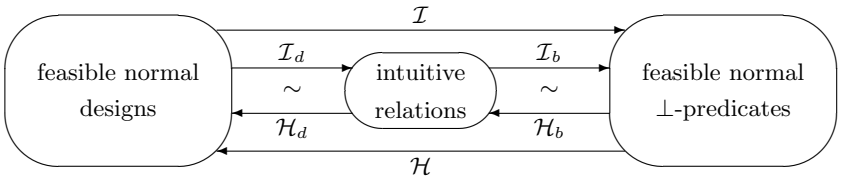

Fig. 3. Intuitive relations as an intermediate

Lemma 16. *$\mathcal{I}_d$ and $\mathcal{H}_d$ are inverse to each other.*

Proof. If P is intuitive relation, $\mathcal{I}_d(\mathcal{H}_d(P)) = \mathcal{I}_d(P; \mathbb{T} \vdash P) = P; \mathbb{T} \wedge P = P$. Let $P_1 \vdash P_2$ be a feasible normal design, then $\mathbb{1} \vdash \mathbb{1} = \mathbb{T} = (P_1 \vdash P_2); \mathbb{T} = (P_1 \vdash P_2); (\mathbb{1} \vdash \mathbb{1}) = P_1 \wedge \overline{P_2; \mathbb{T}} \vdash P_2; \mathbb{1}$ by [1, Theorem 3.1.4]. This implies $P_1 \wedge \overline{P_2; \mathbb{T}} \leq \mathbb{1}$, therefore $\mathcal{H}_d(\mathcal{I}_d(P_1 \vdash P_2)) = (P_1 \wedge P_2); \mathbb{T} \vdash P_1 \wedge P_2 = P_1 \wedge P_2; \mathbb{T} \vdash P_1 \wedge P_2 = P_1 \wedge P_2 \vdash P_1 \wedge P_2 = P_1 \vdash P_2$, both by [1, Theorem 3.1.2].

4.2 Eliminating Improper Values

To complete the picture, the transition from intuitive relations to feasible normal $\perp$-predicates and back is accomplished by the functions $\mathcal{I}_b$ and $\mathcal{H}_b$. Although they are inverse to each other, they too do not preserve the structure since they are not even monotonic. Their definition combines the bijections of Sects. 3 and 4.1. The structure-preserving mappings $\mathcal{I}$ and $\mathcal{H}$ are thus split in two, with intuitive relations as an intermediate lattice whose structure is different from that of normal designs and normal $\perp$-predicates. The result is displayed in Fig. 3.

Definition 17. *The mapping $\mathcal{I}_b$ from intuitive relations to $\perp$-predicates is given by $\mathcal{I}_b =_{\text{def}} \mathcal{I} \circ \mathcal{H}_d$. It maps to feasible normal $\perp$-predicates by Lemma 8, since $\mathcal{H}_d$ maps to feasible normal designs.*
The mapping $\mathcal{H}_b$ from feasible normal $\perp$-predicates to intuitive relations is given by $\mathcal{H}_b =_{\text{def}} \mathcal{I}_d \circ \mathcal{H}$. The composition is well-defined by Lemma 12.

Lemma 18. *$\mathcal{I}_b(P) = \overline{P^+; \mathbb{T}} \vee P^+$ and $\mathcal{H}_b(P) = (\overline{P}; \mathbb{T})^- \wedge P^-$.*

Proof. $\mathcal{I}_b(P) = \mathcal{I}(\mathcal{H}_d(P)) = \mathcal{I}(P; \mathbb{T} \vdash P) = \overline{(P; \mathbb{T})^+; \mathbb{T}} \vee P^+ = \overline{P^+; \mathbb{T}} \vee P^+$ by Lemma 3(9), and $\mathcal{H}_b(P) = \mathcal{I}_d(\mathcal{H}(P)) = \mathcal{I}_d((\overline{P}; \mathbb{T})^- \vdash P^-) = (\overline{P}; \mathbb{T})^- \wedge P^-$.

Theorem 19. *$\mathcal{I}_b \circ \mathcal{I}_d = \mathcal{I}$, $\mathcal{H}_d \circ \mathcal{H}_b = \mathcal{H}$, $\mathcal{I}_b$ and $\mathcal{H}_b$ are inverse to each other.*

Proof. By Lemma 16, $\mathcal{I}_b \circ \mathcal{I}_d = \mathcal{I} \circ \mathcal{H}_d \circ \mathcal{I}_d = \mathcal{I}$, and $\mathcal{H}_d \circ \mathcal{H}_b = \mathcal{H}_d \circ \mathcal{I}_d \circ \mathcal{H} = \mathcal{H}$. By Theorem 13 and Lemma 16, $\mathcal{H}_b \circ \mathcal{I}_b = \mathcal{I}_d \circ \mathcal{H} \circ \mathcal{I} \circ \mathcal{H}_d = \mathcal{I}_d \circ \mathcal{H}_d = 1$ and $\mathcal{I}_b \circ \mathcal{H}_b = \mathcal{I} \circ \mathcal{H}_d \circ \mathcal{I}_d \circ \mathcal{H} = \mathcal{I} \circ \mathcal{H} = 1$.

5 Conclusion

Let us summarise the three contributions of our paper. In Sect. 3 we have defined an alternative basis to model non-termination in UTP, and proved both views isomorphic. This provides further confidence into UTP as an adequate tool for modelling program semantics. In Sect. 4 we have set up a one-to-one correspondence to the intuitive relational reading of predicates. A minor contribution, Sect. 2 is a concise presentation of the building blocks of UTP and its structural properties.

Our results are in line with [5] who discuss the need for a stability convention, and argue that "it is really of no interest which convention we choose". The two examples of conventions presented in [5] correspond to our normal designs and normal $\perp$-predicates, and so do the "partial relations" of [5] to the intuitive reading, but their connections are not investigated further there.

In terms of the classification of [6], $\perp$-predicates and intuitive relations correspond to the general and partial semantic models, respectively. Normal $\perp$-predicates and feasible normal $\perp$-predicates, however, differ from the general and total semantic models by relating a state to all proper outcomes whenever it is related to the looping outcome.

Relations augmented by $\perp$ subjected to the Egli-Milner ordering are used to define the semantics of an imperative language in [7]. The same ordering on relations extended with a "definedness predicate" is used for an applicative language in [8], then addressed by relation algebraic means in [9]. Such relations are shown isomorphic to state transformers by [10] who also discuss the Smyth ordering. For further pointers in connection with UTP we refer to the bibliography of [1].

The technique used in Sect. 3 can be applied analogously to a modified notion of designs, called *prescriptions*, defined in [2] to deal with general correctness, as opposed to the goal of total correctness pursued by UTP. The results from Sect. 4, however, do not carry over, for prescriptions provide a strictly finer description of program behaviour than designs.

Demonic operators are defined by [11] for modelling total correctness in modal semirings. Using abstract versions of the mappings defined in Sect. 4.1 they can be derived from the corresponding operators on designs. The development is given in [12] where the mappings are also used to calculate the semantics of the demonic while loop.

Further work is concerned with the links of UTP to functional programming, where the introduction of $\perp$ is conventional.

References

1. Hoare, C.A.R., He, J.: Unifying theories of programming. Prentice Hall Europe (1998)
2. Dunne, S.: Recasting Hoare and He's unifying theory of programs in the context of general correctness. In Butterfield, A., Strong, G., Pahl, C., eds.: 5th Irish Workshop on Formal Methods. EWiC, The British Computer Society (2001)
3. Schmidt, G., Ströhlein, T.: Relationen und Graphen. Springer-Verlag (1989)

4. Szász, G.: Introduction to Lattice Theory. 3rd edn. Academic Press (1963)
5. Hehner, E.C.R., Malton, A.J.: Termination conventions and comparative semantics. Acta Informatica **25**(1) (1988) 1–14
6. Nelson, G.: A generalization of Dijkstra's calculus. ACM Transactions on Programming Languages and Systems **11**(4) (1989) 517–561
7. de Bakker, J.W.: Semantics and termination of nondeterministic recursive programs. In Michaelson, S., Milner, R., eds.: (Third International Colloquium on) Automata, Languages and Programming, Edinburgh University Press (1976) 435–477
8. Broy, M., Gnatz, R., Wirsing, M.: Semantics of nondeterministic and noncontinuous constructs. In Bauer, F., Broy, M., eds.: Program Construction. Number 69 in Lecture Notes in Computer Science, Springer-Verlag (1979) 553–592
9. Berghammer, R., Zierer, H.: Relational algebraic semantics of deterministic and nondeterministic programs. Theoretical Computer Science **43**(2-3) (1986) 123–147
10. Apt, K.R., Plotkin, G.D.: Countable nondeterminism and random assignment. Journal of the ACM **33**(4) (1986) 724–767
11. Desharnais, J., Möller, B., Tchier, F.: Kleene under a modal demonic star. Journal of Logic and Algebraic Programming, special issue on Relation Algebra and Kleene Algebra (in press 2005)
12. Guttmann, W., Möller, B.: Modal design algebra. First International Symposium on Unifying Theories of Programming (to appear 2006)