Involutions on Relational Program Calculi

I. M. Rewitzky[1] and J. W. Sanders[2]

Abstract

The standard Galois connection between the relational and predicate-transformer models of sequential programming (defined in terms of weakest precondition) confers a certain similarity between them. This paper investigates the extent to which the important involution on transformers (which, for instance, interchanges demonic and angelic nondeterminism, and reduces the two kinds of simulation in the relational model to one kind in the transformer model) carries over to relations. It is shown that no exact analogue exists; that the two complement-based involutions are too weak to be of much use; but that the translation to relations of transformer involution under the Galois connection is *just* strong enough to support Boolean-algebra-style reasoning, a claim that is substantiated by proving properties of deterministic computations. Throughout, the setting is that of the guarded-command language augmented by the usual specification commands; and where possible algebraic reasoning is used in place of the more conventional semantic reasoning.

1 Introduction

We adopt the familiar view that a semantic model for programming, and for the development of programs from specifications through designs to code, consists of a partially-ordered space. The elements of the space are designs expressed in code—'programs'—and designs (including specifications)

---

[1]Department of Mathematical Sciences, University of Stellenbosch, Stellenbosch, South Africa. Email: `rewitzky@sun.ac.za`

[2]International Institute for Software Technology, United Nations University, Macao. Email: `jeff@iist.unu.edu`

expressed using more general 'commands'. The partial order is that of ('more-deterministic-than') refinement.

The two main semantic models for sequential programming, the relational model (see, for example, Hoare *et al.*'s [10]) and the predicate-transformer model (Dijkstra's [5]), are congruent on programs—which we take to consist essentially of Dijkstra's guarded-command language (*loc. cit.*). The congruence is established by the Galois connection consisting of the weakest-precondition function from relations to predicate transformers and its adjoint, relational projection, in the other direction.

But the two models diverge with the incorporation of commands more general than programs, like partially-enabled computations (guarded commands), unbounded demonic nondeterminism and angelic nondeterminism (the work of Back [1], Morgan [14], Nelson [16] and Morris [15]). For example they handle angelic nondeterminism quite differently, and the transformer model is endowed with an involution that accounts for its quantitatively-better structure. Indeed transformer involution interchanges demonic and angelic nondeterminism (see Back and von Wright's [2]), interchanges precondition and guard (the same authors' [3]), reduces the two simulations required for completeness of data refinement in the relational model to a single complete rule in the transformer model (see Gardiner and Morgan's [8] or de Roever and Engelhardt's text [7]), and facilitates familiar Boolean-algebra style of reasoning (Back and von Wright's [3]).

So on one hand the binary-relation and predicate-transformer models share substantial similarities and on the other they exhibit important differences. In this paper we investigate the extent to which the relational model retains vestiges of transformer involution, and how useful that is.

We begin with a simple result: relations possess no equivalent of transformer involution. That means, of course, that any attempt to define an involution by structural induction on relations must fail. But it leaves open the possibility of 'weak' involutions: functions on the relational model that satisfy merely some of the properties enjoyed by an involution.

The first two putative weak involutions we consider are based on set complement in the relational model. Unfortunately both turn out to be too weak because they identify too many computations. So we consider the translation of transformer involution to relations using the Galois connection, and call the result Galois star. It is better behaved than the previous candidates, but an experiment is required to determine whether or not its properties are strong enough to support the kind of reasoning that trans-

former involution permits on transformers.

For that experiment we choose a topic that has emerged as a 'benchmark' [5, 6, 13] for such kinds of reasoning (at least when restricted to the guarded-command language proper): the consideration of determinism. Extending that concept to partially-enabled computations, we distinguish deterministic, predeterministic and postdeterministic computations. A deterministic computation terminates in each initial state with one (statedependent) value; a predeterministic computation at each initial state either fails to terminate or terminates with one value; and a postdeterministic computation at each initial state either fails to be enabled or terminates with one value. (Note that some authors use 'deterministic' for our 'predeterministic'.) Then a standard 'test' of a formalism for reasoning about computations is the ease with which it is able to reason about preservation of determinism: if computations P and Q are predeterministic then so too is their sequential composition $P \mathbin{\mathring{,}} Q$ and their conditional $\mathbf{if}\ a \to P\ [\,]\ b \to Q\ \mathbf{fi}$ with disjoint guards a and b; whilst the binary conditional $P \mathbin{\lhd} b \mathbin{\rhd} Q$ preserves all three kinds of determinism. Reasoning in the transformer model was originally due to Dijkstra [5], then to Dijkstra and Scholten [6], and in the relational model with Tarski's axioms to Maddux [13]. For comparison we also give a proof in the program calculus itself—without any particular semantic model. Such a step we regard as hugely preferable. Indeed one of the techniques promoted in this paper is algebraic reasoning about concepts normally handled semantically.

The paper proper begins, in Section 3, with a summary of programs (the guarded-command language) and their more general commands. It summarises, in Section 4, the relational and transformer models and the Galois connection between them. It then proves, in Section 5, the absence of an involution on the relational model, considers the two complement-based candidates in Sections 6 and 7 before settling, in Section 8, on Galois star which is applied in reasoning about forms of determinism in Section 9. But first, notation must be established.

2   Notation

Pertaining to logic, we write: $\mathbin{\widehat{=}}$ for *equals by definition*; $a : A$ to mean a is of type A; pred.X for the type of predicates on X, where predicates are Boolean-valued functions and substitution is functional application (possible because we consider predicates over only a single state space of Cartesian-

product type); formulae like $\forall\, a : A\, \cdot\, p$ in which the dot simply acts as a syntactic separator; $p \lhd b \rhd q$ for the *binary conditional*, 'p if b else q'; $\leq$ for *implication* on predicates; and the infix relation it engenders we write instead $\Rightarrow$.

Pertaining to binary relations: $A{\leftrightarrow}B$ denotes the type of *binary relations* from A to B and $A{\rightarrow}B$ the type of (total) *functions* from A to B; $id[A]$ denotes the *identity function* on A; *functional application* is written '.' (as in $f.x$) and associates to the left; *composition of functions* is written $\circ$ (as in $f \circ g$); binary relations are written in *infix* (thus r relates a and b is written $a\, r\, b$); (forward) *relational composition*, as well as sequential composition in the programming language, is written $\,{}_9^o\,$ (so that if binary relations f and g are actually functions then their forward relational composition $f\, {}_9^o\, g$ equals their functional composition $g \circ f$); and *relational image* at a point is written $r.(\!| a |\!)$ (and equals $\{y \mid \exists\, x : a\, \cdot\, x\, r\, y\}$).

¿From partial orders we need the following concepts. If $(X, \leq)$ and $(Y, \leq)$ are partial orders then a pair of functions $f : X \rightarrow Y$ and $g : Y \rightarrow X$ forms a *Galois connection* [17] means that they satisfy the equivalence

$$f.x \leq y \;\;\equiv\;\; x \leq g.y\,.$$

A function $f : X \rightarrow Y$ between partial orders is said to be *universally [positively] disjunctive* iff

$$f.\vee\mathcal{X} \;=\; \vee\{f.x \mid x \in \mathcal{X}\}$$

for all [all nonempty] subsets $\mathcal{X}$ of X (and analogously for conjunctivity). Whilst it is very simple, the theory of Galois connections has substantial applications in program semantics [11] and software engineering [4]. Because it is one of those subjects that has been developed largely by folklore, we include in Section 4.3 a summary relevant to our needs. For more general book treatments we refer to the standards [9, 12].

The semantic denotation of a command P is written $[\![P]\!]$. Two semantic models are considered (binary relations and predicate transformers). When confusion could arise by our use of the same notation for two distinct semantics, we clarify which is meant.

More specific notation (like the healthy closure of a binary relation) is introduced as it is needed.

3 Command Calculus

This section summarises the language this paper uses for describing 'programs' (or 'code') and their generalisations 'commands' (or 'specification computations'), and the laws they satisfy. Concepts like 'deterministic' and 'terminating' that are normally applied just to programs (and defined semantically) are here extended to commands and defined algebraically. That enables us to reason algebraically about those concepts.

3.1 Programs

We denote by X the global state space of the programs under consideration; it is the Cartesian product of the types of the various program variables. The state of a program is thus denoted by a vector $x \colon X$.

The syntax of our version [10] of Dijkstra's guarded-command language is summarised in Figure 1. Computation **skip** terminates without changing state and computation **abort** corresponds to divergence. In assignment, the state $x \colon X$ is updated to take the value of the well-typed (well defined: terminating and single-valued) expression e. Sequential composition is standard. Binary conditional is written $P \triangleleft b \triangleright Q$ to express P if b else Q, where b is a (totally defined) predicate on state.

Demonic nondeterminism arises from abstraction of blocks defined at a lower level of abstraction together with the requirement for local reasoning; it corresponds to a choice between its arguments whose resolution lies beyond the current level of abstraction. Recursion is modelled as least prefixed point. For simplicity we do not here include local block or procedure invocation. Refinement corresponds to removal of demonic nondeterminism, so that $P \sqsubseteq Q \;\; \widehat{=} \;\; P \sqcap Q = P$.

3.2 Commands

A computation that is a program, or code, is regarded as *executable*. The commands which extend programs are summarised in Figure 2. Extending the operators (and hence also the ordering) in Figure 1 to commands, the result is a partially-ordered space of computations that we denote $\mathcal{L}(X)$.

Arbitrary demonic nondeterminism is infimum in the refinement ordering and so extends the binary version expressed in programs. Empty demonic nondeterminism is thus the greatest element of $\mathcal{L}(X)$: $\sqcap\{\,\} = \mathbf{magic}$; it refines every command and represents a computation which is never en-

skip	no op
abort	divergence
$x := e$	assignment
$P \,\mathbin{\substack{\circ \\ \circ}}\, Q$	sequential composition
$P \lhd b \rhd Q$	binary conditional
$P \sqcap Q$	demonic nondeterminism
μF	recursion
$P \sqsubseteq Q$	refinement

Figure 1: Syntax for programs and the refinement pre-order.

$\sqcap \mathcal{P}$	arbitrary demonic nondeterminism
choose	arbitrary assignment
magic	unenabled computation
$\{p\}$	assertion
$\langle p \rangle$	coercion
$\sqcup \mathcal{P}$	arbitrary angelic nondeterminism

Figure 2: Syntax for commands (although an assertion is code).

abled. Since arbitrary infima exist, so too do arbitrary suprema. Arbitrary angelic nondeterminism is supremum in the refinement ordering; empty angelic nondeterminism is thus the least element: $\sqcup\{\,\} = \textbf{abort}$.

For a predicate b on state space X, the computation *assert* b skips if b holds but otherwise fails to terminate:

$$\{b\} \quad \widehat{=} \quad \textbf{skip} \lhd b.x \rhd \textbf{abort}\,; \tag{1}$$

thus it is actually code. Computation *coerce* b skips if b holds but otherwise is not enabled:

$$\langle b \rangle \quad \widehat{=} \quad \textbf{skip} \lhd b.x \rhd \textbf{magic}\,. \tag{2}$$

The computation **choose** terminates in an arbitrary state:

$$\textbf{choose} \quad \widehat{=} \quad \sqcap\{x := y \mid y \in X\}\,. \tag{3}$$

When the state space is infinite that is not code. The computation **neq** terminates in a final state different from its initial state:

$$\textbf{neq} \quad \widehat{=} \quad \sqcap\{x := y \mid y \neq x_0\} \tag{4}$$

134

where x_0 denotes the initial state and the predicate $y \neq x_0$ is a finite conjunct if the state space is finite.

The important computational concepts of termination, enabledness and determinism are expressed algebraically as follows. Suppose that P is a command and $x_0 : X$ is a state. Then P *aborts at* x_0 iff the computation might not (equivalently 'will not' in the standard (Hoare/Dijkstra) model we follow here) terminate there

$$\{x = x_0\} \mathbin{\stackrel{\circ}{,}} P \;\; = \;\; \mathbf{abort} \,.$$

Command P is *enabled at* x_0 iff it may (equivalently 'does') begin there

$$\langle x = x_0 \rangle \mathbin{\stackrel{\circ}{,}} P \;\; \neq \;\; \mathbf{magic}$$

which is equivalent (in view of Law (17) to follow) to

$$\langle x = x_0 \rangle \mathbin{\stackrel{\circ}{,}} P \mathbin{\stackrel{\circ}{,}} \mathbf{abort} \;\; = \;\; \mathbf{abort} \,.$$

P *terminates at* x_0 means that it is enabled but does not abort there.

Computation P is *deterministic* at x_0 means that P is enabled there and terminates in only a single final state. To define that term: command P is *co-atomic* at x_0 iff at x_0, P does not equal $\mathbf{magic}$ and no commands lie strictly between P and $\mathbf{magic}$

$$\langle x = x_0 \rangle \mathbin{\stackrel{\circ}{,}} P \;\; \neq \;\; \mathbf{magic} \tag{5}$$
$$\forall\, R : \mathcal{L}(X) \,\cdot\, \langle x = x_0 \rangle \mathbin{\stackrel{\circ}{,}} P \sqsubseteq R \;\; \Rightarrow \;\; R = \mathbf{magic} \,. \tag{6}$$

Then P is *deterministic* at x_0 iff

$$\forall\, x_0 : X \,\cdot\, \langle x = x_0 \rangle \mathbin{\stackrel{\circ}{,}} P \;\;\; \text{is co-atomic} \,.$$

Command P is *postdeterministic* at x_0 iff either it is not enabled there or it terminates in only a single value:

$$\forall\, x_0 : X \,\cdot\, \langle x = x_0 \rangle \mathbin{\stackrel{\circ}{,}} P \neq \mathbf{magic} \;\; \Rightarrow \;\; \langle x = x_0 \rangle \mathbin{\stackrel{\circ}{,}} P \;\; \text{is co-atomic} \,.$$

Finally P is *predeterministic* at x_0 iff it is enabled there and either does not terminate or is deterministic:

$$\forall\, x_0 : X \,\cdot\, \langle x = x_0 \rangle \mathbin{\stackrel{\circ}{,}} P \neq \mathbf{magic}$$
$$\wedge$$
$$\{x = x_0\} \mathbin{\stackrel{\circ}{,}} P \neq \mathbf{abort} \;\; \Rightarrow \;\; \langle x = x_0 \rangle \mathbin{\stackrel{\circ}{,}} P \;\; \text{is co-atomic} \,.$$

A command is terminating [always-enabled, deterministic, predeterministic, postdeterministic] means that it is terminating [enabled, deterministic, predeterministic, postdeterministic] at each initial state x_0. In particular a command is deterministic iff it is predeterministic and postdeterministic. Code is always enabled; and the celebrated loop rule ensures termination of code in the form of an iteration. But **magic**, for example, is not enabled (hence) not terminating, but is postdeterministic. It is convenient to keep in mind that in the transformer model (at least), enabledness plays for commands a rôle dual to that which termination plays for code. A consequence of our interest in relational involutions is the extent to which that remains true for the relational model.

3.3 Calculus

The language $\mathcal{L}(X)$ has the algebraic structure summarised in Figure 3 (which does not claim to list all laws).

Commands **abort** and **magic** are not zeroes on both sides for sequential composition, for that would imply their degeneration (to the same command). Nor does sequential composition distribute demonic nondeterminism and angelic nondeterminism on both sides. Nonetheless under demonic nondeterminism and sequential composition, $\mathcal{L}(X)$ forms what might be called a *pre-quantal* (by comparison with the definitions in Rosenthal's text [18] on quantales): the $\sqcap$ of arbitrary subsets exists and sequential composition is associative with an identity, **skip**; also sequential composition distributes arbitrary demonic nondeterminism in its left-hand argument, and distributes *nonempty* (the reason for the 'pre') demonic nondeterminism in its right-hand argument.

In spite of the failure of sequential composition to distribute angelic nondeterminism, $\mathcal{L}(X)$ forms a complete lattice in which each command is the angelic choice of the compact commands[3] it refines. But equality holds in refinements (14) and (16) if each command being distributed is predeterministic.

The operators of sequential composition and binary conditional are monotone in each argument.

---

[3]A command is *compact* if it aborts outside a finite set (on part of which it may, of course, be unenabled; by comparison, a compact program aborts outside a finite set but is everywhere enabled).

$$(\mathcal{L}(X), \sqcap, \fatsemi, \mathbf{skip}) \text{ forms a pre-quantal} \tag{7}$$

$$(\sqcap \mathcal{P}) \fatsemi R \;\; = \;\; \sqcap\{P \fatsemi R \mid P \in \mathcal{P}\} \tag{8}$$

$$P \fatsemi (\sqcap \mathcal{Q}) \;\; = \;\; \sqcap\{P \fatsemi Q \mid Q \in \mathcal{Q}\}, \quad \text{if } \mathcal{Q} \text{ nonempty} \tag{9}$$

$$(\mathcal{L}(X), \sqsubseteq) \text{ forms a complete lattice and a domain} \tag{10}$$

$$\text{with maximum } \mathbf{magic} \text{ and minimum } \mathbf{abort} \tag{11}$$

$$\mathbf{magic} \fatsemi R \;\; = \;\; \mathbf{magic} \tag{12}$$

$$P \fatsemi \mathbf{magic} \;\; = \;\; \mathbf{magic}, \quad \text{if } P \text{ always terminates} \tag{13}$$

$$(\sqcup \mathcal{P}) \fatsemi Q \;\; \sqsupseteq \;\; \sqcup\{P \fatsemi Q \mid P \in \mathcal{Q}\}, \; = \; \text{if all } P \text{ are predet.} \tag{14}$$

$$\mathbf{abort} \fatsemi Q \;\; = \;\; \mathbf{abort} \tag{15}$$

$$P \fatsemi (\sqcup \mathcal{Q}) \;\; \sqsupseteq \;\; \sqcup\{P \fatsemi Q \mid Q \in \mathcal{Q}\}, \; = \; \text{if all } Q \text{ are predet.} \tag{16}$$

$$P \fatsemi \mathbf{abort} \;\; = \;\; \mathbf{abort}, \quad \text{if } P \text{ always enabled} \tag{17}$$

$$(P \lhd b \rhd Q) \sqcap R \;\; = \;\; (P \sqcap R) \lhd b \rhd (Q \sqcap R) \tag{18}$$

$$(P \lhd b \rhd Q) \sqcup R \;\; = \;\; (P \sqcup R) \lhd b \rhd (Q \sqcup R) \tag{19}$$

$$(P \lhd b \rhd Q) \fatsemi R \;\; = \;\; (P \fatsemi R) \lhd b \rhd (Q \fatsemi R) \tag{20}$$

$$P \lhd b \rhd Q \;\; = \;\; \{b\} \fatsemi P \;\sqcup\; \{\neg b\} \fatsemi Q \tag{21}$$

$$= \;\; \langle b \rangle \fatsemi P \;\sqcap\; \langle \neg b \rangle \fatsemi Q \tag{22}$$

$$(x := e) \fatsemi (x := f) \;\; = \;\; x := f \circ e \tag{23}$$

Figure 3: Laws for the language $\mathcal{L}$.

4 Program Semantics

This section summarises the relational and predicate-transformer semantics of $\mathcal{L}(X)$ and the (Galois) connection between them. In each case the semantics yields a pre-quantal-right module.

4.1 Relational Semantics

In this section we give the relational semantics of the language $\mathcal{L}(X)$, first for code. Each command is represented as a relation from initial states to the final states attainable from each initial state.

4.1.1 Relations

The state space augmented with the improper state $\bot$ (representing nontermination) is denoted $X_\bot \;\widehat{=}\; X \cup \{\bot\}$. As usual we treat $X_\bot$ as the flat domain X augmented with least element $\bot$. The improper state $\bot$ is not part of the programming notation and it is not a value which can be assigned to the global variable. It is simply a semantic artifact, enabling nontermination to be distinguished from arbitrary termination.

We write relations in infix, and use the convention that, for a relation e with domain X and range $X_\bot$, $e_\bot$ denotes the relation on $X_\bot$

$$e_\bot \;\;\widehat{=}\;\; e \,\cup\, (\{\bot\} {\times} X_\bot) \;.$$

The semantic space for the relational semantics is the subspace of relations on $X_\bot$ that are strict and whose relational images are upclosed

$$\mathcal{R}(X) \;\;\widehat{=}\;\; \{d : X_\bot {\leftrightarrow} X_\bot \mid (\bot\, d\, \bot) \;\;\wedge\;\; (x\, d\, \bot \Rightarrow d.(\!\lvert x \rvert\!) = X_\bot)\}$$

with the inclusion ordering $\supseteq$ for 'more-deterministic-than' refinement (since multiple-valuedness of a relation captures demonic nondeterminism of the command it represents).

It is readily confirmed that $(\mathcal{R}(X), \supseteq)$ is a domain and a complete lattice with least element $X_\bot {\times} X_\bot$, greatest element $\{\,\}_\bot$ and compact elements the relations which are cofinite subsets of $X_\bot {\times} X_\bot$; moreover it is a Boolean algebra under the usual set-theoretic complement.

The *healthy closure* of any relation r on $X_\bot$ is given by the value at r of a function $h : (X_\bot {\leftrightarrow} X_\bot) {\rightarrow} \mathcal{R}(X)$, characterised by:

$$x\,(h.r)\,y \;\;=\;\; (x = \bot) \,\vee\, (x\, r\, \bot) \,\vee\, (x\, r\, y)\,.$$

$$\begin{aligned}
[\![\mathbf{skip}]\!] &\ \hat{=}\ id[X]_\bot \\
[\![\mathbf{abort}]\!] &\ \hat{=}\ X_\bot \times X_\bot \\
[\![\chi := e]\!] &\ \hat{=}\ (\{(x, e.x) \mid x \in X\ \wedge\ e.x \text{ terminates}\})_\bot \\
[\![P \mathbin{\text{\textsemicolon}} Q]\!] &\ \hat{=}\ [\![P]\!] \mathbin{\text{\textsemicolon}} [Q] \\
[\![P \lhd b \rhd Q]\!] &\ \hat{=}\ \{(x, y) \mid x\, [\![P]\!]\, y \lhd b.x \rhd x\, [\![Q]\!]\, y\} \\
[\![P \sqcap Q]\!] &\ \hat{=}\ [\![P]\!] \cup [\![Q]\!] \\
[\![\mu F]\!] &\ \hat{=}\ \cup\{d \mid F.d \supseteq d\}, \quad F \text{ monotone on code}
\end{aligned}$$

Figure 4: Relational semantics of code.

Furthermore r is called *healthy* if $h.r = r$. Evidently h is well defined, increasing ($r \subseteq h.r$), monotone with respect to inclusion on both sides ($r \subseteq s \implies h.r \subseteq h.s$) and $h.r$ is the smallest healthy relation containing r.

4.1.2 Semantics

The relational semantics ascribes to each command P a relation $[\![P]\!] : \mathcal{R}(X)$.

The semantics of code is given in Figure 4. Denotations of code satisfy this healthiness condition: for each $x : X$ the relational image $d.(\!|\, x\, |\!)$ is nonempty and either finite or all of $X_\bot$. The subspace of such healthy relations forms a domain with least element $X_\bot \times X_\bot$, maximal elements the (total) functions and compact elements the members for which only finitely many elements of X are *not* related to every element of $X_\bot$. In the definition of recursion, the function F is defined on that healthy subspace of $\mathcal{R}(X)$ and the relation d ranges over it.

The semantics of commands is given in Figure 5. They satisfy as healthiness condition just the defining condition of $\mathcal{R}(X)$. Evidently the space $\mathcal{R}(X)$ contains the space of denotations of code. However the injection fails to form the embedding in a Galois connection. Otherwise its companion projection would map the maximum $\{\,\}_\bot : \mathcal{R}(X)$ to a maximum denotation of code; but no such denotation exists. So the two spaces have an uneasy relationship, compared with the corresponding domains in the predicate-transformer semantics.

4.2 Transformer Semantics

A command may be viewed as a predicate transformer in two, adjoint, ways. We follow (Dijkstra's) tradition and consider it as a function from postcon-

$$\begin{aligned}
\llbracket \sqcap \mathcal{S} \rrbracket &\;\;\widehat{=}\;\; \cup \{\llbracket S \rrbracket \mid S \in \mathcal{S}\} \\
\llbracket \mathbf{magic} \rrbracket &\;\;\widehat{=}\;\; \{\;\}_{\perp} \\
\llbracket \{b\} \rrbracket &\;\;\widehat{=}\;\; \{(x,y) : X \times X_{\perp} \mid b.x \Rightarrow x = y\}_{\perp} \\
\llbracket \langle b \rangle \rrbracket &\;\;\widehat{=}\;\; \{(x,x) : X \times X \mid b.x\}_{\perp} \\
\llbracket \sqcup \mathcal{S} \rrbracket &\;\;\widehat{=}\;\; \cap \{\llbracket S \rrbracket \mid S \in \mathcal{S}\}
\end{aligned}$$

Figure 5: Relational semantics for commands.

ditions to preconditions: postcondition q is mapped to the precondition true at just those states from which the command is certain to terminate in a state satisfying p: the *weakest precondition of the command evaluated at q.* (The possibility of demonic nondeterminism is responsible for the word 'certain'.)

4.2.1 Transformers

Each such function is monotonic under the usual ordering on predicates: a weaker postcondition engenders a weaker weakest precondition. We write $(\mathrm{pred}.X, \leq)$ for the space of all predicates (Boolean-valued functions) on X under the implication (pointwise) partial ordering. The space for the transformer semantics is then the space of all functions on that space that are monotone

$$\mathcal{T}(X) \;\;\widehat{=}\;\; \{t : \mathrm{pred}.X \to \mathrm{pred}.X \mid \forall\, q, r : \mathrm{pred}.X \;\cdot\; q \leq r \Rightarrow t.q \leq t.r\},$$

ordered by the pointwise lifting of the order $\leq$ on predicates

$$t \leq u \;\;\widehat{=}\;\; \forall\, q : \mathrm{pred}.X \;\cdot\; t.q \leq u.q\,.$$

Thus t is refined by u iff for each postcondition q, the weakest precondition of t at q is at least as strong as the weakest precondition of u at q; whenever t achieves q so too does u.

Because the order on $\mathcal{T}(X)$ is the lifting of implication, least upper bounds and greatest lower bounds of arbitrary sets exist pointwise. So $(\mathcal{T}(X), \leq)$ is readily seen to be a pre-quantal, a complete lattice and a domain.

$$
\begin{aligned}
\llbracket \mathbf{skip} \rrbracket &\ \widehat{=}\ id[\mathrm{pred}.X] \\
\llbracket \mathbf{abort} \rrbracket &\ \widehat{=}\ false \\
\llbracket x := e \rrbracket .q.x &\ \widehat{=}\ q.(e.x) \quad (\ =\ q[e/x]) \\
\llbracket P \fatsemi Q \rrbracket &\ \widehat{=}\ \llbracket P \rrbracket \circ \llbracket Q \rrbracket \\
\llbracket P \triangleleft b \triangleright Q \rrbracket &\ \widehat{=}\ \llbracket P \rrbracket \triangleleft b \triangleright \llbracket Q \rrbracket \\
\llbracket P \sqcap Q \rrbracket &\ \widehat{=}\ \llbracket P \rrbracket \wedge \llbracket Q \rrbracket \\
\llbracket \mu F \rrbracket &\ \widehat{=}\ \vee \{t : \mathcal{T}(X) \mid F.t \geq t\}, \quad F \text{ monotone on } \mathcal{T}(X)
\end{aligned}
$$

Figure 6: Transformer semantics of code.

$$
\begin{aligned}
\llbracket \sqcap \mathcal{S} \rrbracket &\ \widehat{=}\ \wedge \{\llbracket S \rrbracket \mid S \in \mathcal{S}\} \\
\llbracket \mathbf{magic} \rrbracket &\ \widehat{=}\ true \\
\llbracket \{b\} \rrbracket .q &\ \widehat{=}\ b \wedge q \\
\llbracket \langle b \rangle \rrbracket .q &\ \widehat{=}\ b \Rightarrow q \\
\llbracket \sqcup \mathcal{S} \rrbracket &\ \widehat{=}\ \vee \{\llbracket S \rrbracket \mid S \in \mathcal{S}\}
\end{aligned}
$$

Figure 7: Transformer semantics for commands.

4.2.2 Semantics

The transformer semantics ascribes to each command P a predicate-transformer $\llbracket P \rrbracket$ in $\mathcal{T}(X)$.

The semantics of code is given in Figure 6. Denotations of code satisfy Dijkstra's healthiness conditions: the transformer is positively conjunctive and $\leq$-continuous.

The semantics of commands is given in Figure 7. The healthiness condition is simply monotonicity, the defining property of $\mathcal{T}(S)$.

4.2.3 Involution

For transformer $t : \mathcal{T}(X)$, its *involute* $t^* : \mathcal{T}(X)$ is defined:

$$
t^*.q \quad \widehat{=} \quad \neg t.\neg q\,.
$$

Involution plays an important rôle in transformer semantics: it is useful for calculation because it obeys de Morgan's laws, interchanging demonic and angelic nondeterminism; it provides a duality (as a result) between termination and enabledness; it converts one of the two simulation conditions

necessary in the relational model for data refinement to the other, thus ensuring that one simulation condition is alone sufficient for data refinement in the transformer semantics. Its properties are:

Theorem 1 *Involution is well defined on* $(\mathcal{T}(X), \leq)$ *and satisfies*

$$
\begin{array}{rcll}
t \; monotone & \equiv & t^* \; monotone & (24) \\
t^{**} & = & t & (25) \\
false^* = true & and & true^* = false & (26) \\
(t \mathbin{\,;\,} u)^* & = & t^* \mathbin{\,;\,} u^* & (27) \\
(id[\mathcal{T}(X)])^* & = & id[\mathcal{T}(X)] & (28) \\
t^* \sqsubseteq u & \equiv & t \sqsupseteq u^* & (29) \\
(t \sqcap u)^* = t^* \sqcup u^* & and & (t \sqcup u)^* = t^* \sqcap u^* & (30) \\
(\neg t)^* & = & \neg(t^*) & (31) \\
t \; [universally] \; conjunctive & \equiv & t^* \; [universally] \; disjunctive. & (32)
\end{array}
$$

4.3 Galois Connection

The function $wp : \mathcal{R}(X) \to \mathcal{T}(X)$ is defined, for a relational computation $r : \mathcal{R}(X)$, postcondition $q : \mathrm{pred}.X$ and state $x : X$:

$$
wp.r.q.x \quad \widehat{=} \quad \forall \, y : X_\perp \cdot x \, r \, y \Rightarrow (y \neq \perp \wedge q.y) \, .
$$

That says, as it ought, that $wp.r.q$ holds at just those states from which termination is ensured, in a state satisfying q. The consequent can be simplified: since the domain of q is X, q is not defined at $\perp$; so the first conjunct can, in the presence of the type statement $y : X_\perp$ and the understanding that $\neg(q. \perp)$, be omitted:

$$
wp.r.q.x \quad = \quad \forall \, y : X_\perp \cdot x \, r \, y \Rightarrow q.y \, .
$$

Verification that wp is well defined (that $wp.r$ is monotone) is immediate.

It is also routine to show that wp is universally $(\cup, \geq)$-junctive, *i.e.* from $(\mathcal{R}(X), \subseteq)$ to $(\mathcal{T}(X), \geq)$. Thus wp has an adjoint, called the *relational projection, rp*, that can be defined as follows. For $t : \mathcal{T}(X)$, $rp.t$ is the relation on $X_\perp$ defined to be strict and to satisfy, for $x : X$ and $y : X_\perp$,

$$
x \, (rp.t) \, y \quad \widehat{=} \quad \forall \, q : \mathrm{pred}.X \cdot t.q.x \Rightarrow q.y \, ,
$$

again with the convention $\neg(q. \perp)$. In particular $rp.false = X_\perp \times X_\perp$.

Adjunction means that

$$t \leq wp.r \;\equiv\; r \subseteq rp.t \tag{33}$$

so that the functions wp and rp form a Galois connection between the relational and transformer spaces with their orders reversed: from the space $(\mathcal{R}(X), \subseteq)$ to the space $(\mathcal{T}(X), \geq)$.

Standard theory [17] shows that the Galois connection preserves much of the structure on the two semantics models, except for angelic nondeterminism. Gathering the (elementary) properties we need, in spite of some being consequences of others:

Theorem 2 *The Galois connection satisfies*

$$
\begin{align}
r \subseteq s \;&\Rightarrow\; wp.r \geq wp.s \tag{34} \\
t \geq u \;&\Rightarrow\; rp.t \subseteq wp.u \tag{35} \\
rp \circ wp \;&=\; id[\mathcal{R}(X)] \tag{36} \\
id[\mathcal{T}(X)] \;&\leq\; wp \circ rp \tag{37} \\
wp.(r \,\fatsemi\, s) \;&=\; (wp.r) \circ (wp.s) \tag{38} \\
wp.(id[X]_\perp) \;&=\; id[\mathcal{T}(X)] \tag{39} \\
rp.(t \circ u) \;&=\; (rp.t) \,\fatsemi\, (rp.u) \tag{40} \\
rp.id[\mathcal{T}(X)] \;&=\; id[X]_\perp \tag{41} \\
\forall\, U \subseteq \mathcal{T}(X) \cdot rp.\vee U \;&=\; \cap\, rp.(\!|\, U\,|\!) \tag{42} \\
\forall\, U \subseteq \mathcal{T}(X) \cdot rp.\wedge U \;&=\; \cup\, rp.(\!|\, U\,|\!) \tag{43} \\
rp.true \;&=\; \{\,\}_\perp \tag{44} \\
rp.false \;&=\; X_\perp \times X_\perp \tag{45} \\
\forall\, S \subseteq \mathcal{R}(X) \cdot wp.(\cup S) \;&=\; \wedge\, wp.(\!|\, S\,|\!) \tag{46} \\
wp.(X_\perp \times X_\perp) \;&=\; false \tag{47} \\
wp.\{\,\}_\perp \;&=\; true \tag{48} \\
\forall\, S \subseteq \mathcal{R}(X) \cdot wp.(\cap S) \;&\geq\; \vee\, wp.(\!|\, S\,|\!) \tag{49}
\end{align}
$$

The fact that inclusion (49) may be strict indicates why the embedding wp cannot be used to lift angelic nondeterminism from relations to transformers. Otherwise, the transformer semantics (Figures 4 and 5) is obtained from the relational semantics (Figures 6 and 7) under wp.

5   No Relational Involution

In this section we establish that there is no function on $\mathcal{R}(X)$ satisfying the minimum requirements of an involution—as exhibited by * on transformers. Henceforth, semantic brackets refer solely to the relational semantics.

Theorem 3 *There is no function * on $\mathcal{R}(X)$ that is involutive, obeys either of the De Morgan laws and distributes sequential composition, i.e. that satisfies*

(a) $\forall\, r : \mathcal{R}(X) \cdot r^{**} = r$

(b) *either* $\forall\, r, s : \mathcal{R}(X) \cdot (r \sqcup s)^* = r^* \sqcap s^*$

    or $\forall\, r, s : \mathcal{R}(X) \cdot (r \sqcap s)^* = r^* \sqcup s^*$

(c) $\forall\, r, s : \mathcal{R}(X) \cdot (r \,\mathring{\,}_9\, s)^* = r^* \,\mathring{\,}_9\, s^*$.

Proof: We argue by contradiction, establishing an untenable identity. First we observe that Assumptions (a) and (b) are sufficient to establish the equivalence of the two De Morgan laws in Assumption (b):

(d) $(r \sqcap s)^* = r^* \sqcup s^*$ $\equiv$ $(r \sqcup s)^* = r^* \sqcap s^*$.

The proof is trivial: if the first De Morgan law holds then

$(r \sqcap s)^*$

$=$ Assumption (a)

$(r^{**} \sqcap s^{**})^*$

$=$ Assumption (b)

$(r^* \sqcup s^*)^{**}$

$=$ Assumption (a)

$r^* \sqcup s^*$,

so that the second De Morgan law holds, and the result follows by symmetry.
    Now for relations

$$r \,\mathring{\,}_9\, (s \sqcap t) \;\; = \;\; (r \,\mathring{\,}_9\, s) \sqcap (r \,\mathring{\,}_9\, t) . \tag{50}$$

But in the light of the calculation above, that is inconsistent with the potentially strict Law (16), because from (50) we can infer, for any $r, s, t : \mathcal{R}(X)$,

$$r \mathbin{\fatsemi} (s \sqcup t) \;\;=\;\; (r \mathbin{\fatsemi} s) \sqcup (r \mathbin{\fatsemi} t). \tag{51}$$

Indeed

$r \mathbin{\fatsemi} (s \sqcup t)$

$=$ Assumption (a)

$(r \mathbin{\fatsemi} (s \sqcup t))^{**}$

$=$ Assumption (c)

$(r^* \mathbin{\fatsemi} (s \sqcup t)^*)^*$

$=$ Assumption (b)

$(r^* \mathbin{\fatsemi} (s^* \sqcap t^*))^*$

$=$ Law (50)

$((r^* \mathbin{\fatsemi} s^*) \sqcap (r^* \mathbin{\fatsemi} t^*))^*$

$=$ Assumption (c)

$((r \mathbin{\fatsemi} s)^* \sqcap (r \mathbin{\fatsemi} t)^*)^*$

$=$ Assumption (b)

$(r \mathbin{\fatsemi} s)^{**} \sqcup (r \mathbin{\fatsemi} t)^{**}$

$=$ Assumption (a)

$(r \mathbin{\fatsemi} s) \sqcup (r \mathbin{\fatsemi} t)$.

It remains to demonstrate that Claim (51) fails, for which a simple example suffices. Considering $X \triangleq \{0, 1\}$ with

$$
\begin{aligned}
r &\triangleq \{(0,0), (0,1)\}_\perp \\
s &\triangleq \{(0,0)\}_\perp \\
t &\triangleq \{(1,0)\}_\perp,
\end{aligned}
$$

we find

$$r \mathbin{\fatsemi} (s \sqcup t) \;\;=\;\; \{\} \;\;\subset\;\; \{(0,0)\} \;\;=\;\; (r \mathbin{\fatsemi} s) \sqcup (r \mathbin{\fatsemi} t).$$

$\square$

As an application of our theme—the investigation of the extent to which reasoning that is normally done semantically can be done algebraically—the following corollary lifts that counterexample, and hence the theorem, to $\mathcal{R}(X)$.

Corollary 1 *Theorem 3 holds in any $\mathcal{R}(X)$ where X has at least two elements.*

Proof: In the previous proof, only the counterexample used semantic reasoning and hence needs to reworked in $\mathcal{R}(X)$. Without loss we assume that X has exactly two elements (otherwise the following construction can be embedded within X).

Consider doubleton state space $X \mathrel{\widehat=} \{x_0, x_1\}$ and computations

$$
\begin{aligned}
R &\mathrel{\widehat=} \langle x = x_0 \rangle \mathbin{\mathring{,}} (x := x_0 \sqcap x := x_1) \\
S &\mathrel{\widehat=} \langle x = x_0 \rangle \\
T &\mathrel{\widehat=} \langle x \neq x_0 \rangle \mathbin{\mathring{,}} x := x_0 .
\end{aligned}
$$

We claim that:

$$
\begin{aligned}
S \sqcup T &= \mathbf{magic} & (52) \\
(R \mathbin{\mathring{,}} S) \sqcup (R \mathbin{\mathring{,}} T) &= S & (53) \\
R \mathbin{\mathring{,}} (S \sqcup T) &= \mathbf{magic}. & (54)
\end{aligned}
$$

For Claim (52), we reason

$S \ \sqcup \ T$

$=$ definition

$\langle x = x_0 \rangle \ \sqcup \ \langle x \neq x_0 \rangle \mathbin{\mathring{,}} x := x_0$

$=$ definition of coercion

$\mathbf{skip} \mathbin{\lhd} x = x_0 \mathbin{\rhd} \mathbf{magic}$
$\sqcup$
$(\mathbf{skip} \mathbin{\lhd} x \neq x_0 \mathbin{\rhd} \mathbf{magic}) \mathbin{\mathring{,}} x := x_0$

$=$ Law (20)

$\mathbf{skip} \mathbin{\lhd} x = x_0 \mathbin{\rhd} \mathbf{magic}$
$\sqcup$
$(\mathbf{skip} \mathbin{\mathring{,}} x := x_0) \mathbin{\lhd} x \neq x_0 \mathbin{\rhd} (\mathbf{magic} \mathbin{\mathring{,}} x := x_0)$

146

$=$ Laws (7), (12), calculus

skip $\lhd\ x = x_0\ \rhd$ **magic**
$\sqcup$
magic $\lhd\ x = x_0\ \rhd\ x := x_0$

$=$ Law (19), calculus

$(\textbf{skip} \sqcup \textbf{magic}) \lhd\ x = x_0\ \rhd\ (\textbf{magic} \sqcup x := x_0)$

$=$ Law (11)

magic $\lhd\ x = x_0\ \rhd$ **magic**

$=$ calculus

magic .

For Claim (53) we show $(R \mathbin{\fatsemi} S) = (R \mathbin{\fatsemi} T) = S$ and, since each is similar, we prove just one equality:

$R \mathbin{\fatsemi} S$

$=$ definitions

$\langle\!\langle x = x_0 \rangle\!\rangle \mathbin{\fatsemi} (x := x_0 \sqcap x := x_1) \mathbin{\fatsemi} \langle\!\langle x = x_0 \rangle\!\rangle$

$=$ definition of coercion

$(\textbf{skip} \lhd\ x = x_0\ \rhd\ \textbf{magic}) \mathbin{\fatsemi} (x := x_0 \sqcap x := x_1) \mathbin{\fatsemi}$
$(\textbf{skip} \lhd\ x = x_0\ \rhd\ \textbf{magic})$

$=$ Laws (20), (7), (9), (12)

$((x := x_0 \sqcap x := x_1) \lhd\ x = x_0\ \rhd\ \textbf{magic}) \mathbin{\fatsemi}$
$(\textbf{skip} \lhd\ x = x_0\ \rhd\ \textbf{magic})$

$=$ Laws (20), (12) again

$(x := x_0 \sqcap x := x_1) \mathbin{\fatsemi} (\textbf{skip} \lhd\ x = x_0\ \rhd\ \textbf{magic})$
$\lhd\ \ x = x_0\ \ \rhd$
magic

$=$ Law (8)

$(\ \ (x := x_0 \mathbin{\fatsemi} (\textbf{skip} \lhd\ x = x_0\ \rhd\ \textbf{magic}))$
   $\sqcap$
   $(x := x_1 \mathbin{\fatsemi} (\textbf{skip} \lhd\ x = x_0\ \rhd\ \textbf{magic}))\ \ )$
$\lhd\ \ x = x_0\ \ \rhd$
magic

$$= \qquad \text{Law (23)}$$

$$(\mathbf{skip} \lhd \textit{true} \rhd \mathbf{magic}) \sqcap (\mathbf{skip} \lhd \textit{false} \rhd \mathbf{magic})$$
$$\lhd \;\; x = x_0 \;\; \rhd$$
$$\mathbf{magic}$$

$$= \qquad \text{calculus}$$

$$\mathbf{skip} \sqcap \mathbf{magic}$$
$$\lhd \;\; x = x_0 \;\; \rhd$$
$$\mathbf{magic}$$

$$= \qquad \text{Law (11)}$$

$$\mathbf{skip} \lhd x = x_0 \rhd \mathbf{magic}$$

$$= \qquad \text{definition of coercion}$$

$$\langle x = x_0 \rangle$$

$$= \qquad \text{definition}$$

$$S \,.$$

For Claim (54), we start by observing that if $y : X$ then the assignment $x := y$ is total and hence by Law (13)

$$x := y \mathbin{\substack{\circ \\ 9}} \mathbf{magic} \;=\; \mathbf{magic} \,. \tag{55}$$

Thus

$$(x := x_0 \sqcap x := x_1) \mathbin{\substack{\circ \\ 9}} \mathbf{magic}$$

$$= \qquad \text{Law (8)}$$

$$(x := x_0 \mathbin{\substack{\circ \\ 9}} \mathbf{magic}) \;\sqcap\; (x := x_1 \mathbin{\substack{\circ \\ 9}} \mathbf{magic})$$

$$= \qquad \text{Claim (55) and } \sqcap \text{ idempotent}$$

$$\mathbf{magic} \,.$$

Now we observe

$$R \mathbin{\substack{\circ \\ 9}} (S \sqcup T)$$

$$= \qquad \text{definition and Claim (52)}$$

$$\langle x = x_0 \rangle \mathbin{\substack{\circ \\ 9}} (x := x_0 \sqcap x := x_1) \mathbin{\substack{\circ \\ 9}} \mathbf{magic}$$

$$= \qquad \text{just established}$$

$\langle x = x_0 \rangle \mathbin{\raise0.3ex\hbox{$\scriptstyle\circ$}\mkern-0.5mu\raise-0.3ex\hbox{$\scriptstyle\circ$}}$ **magic**

$=$ definition of coercion

$(\textbf{skip} \lhd x = x_0 \rhd \textbf{magic}) \mathbin{\raise0.3ex\hbox{$\scriptstyle\circ$}\mkern-0.5mu\raise-0.3ex\hbox{$\scriptstyle\circ$}}$ **magic**

$=$ Laws (20), (12), (13)

$\textbf{magic} \lhd x = x_0 \rhd \textbf{magic}$

$=$ calculus

magic .

Having established the three claims we infer the strict inclusion sought by comparing Claims (53) and (54). $\qquad\qquad\qquad\qquad\qquad\qquad\square$

The result of this section may be re-phrased: any attempt to define an involution on $\mathcal{R}(X)$ by structural induction over commands leads to inconsistency unless at least one of the properties (a) to (c) is violated. In the following three sections we consider weaker alternatives to an involution by weakening our requirements and being guided by semantic considerations.

6   Complement

To begin our quest for 'weak involutions' on $\mathcal{R}(X)$, the most obvious *a priori* choice seems to lie with some form of set-theoretic complement on the grounds that it satisfies the De Morgan laws, interchanging unions (*i.e.* demonic nondeterminism) and intersections (*i.e.* angelic nondeterminism).

On inspection we are confronted with two alternatives. Before making the result healthy we could complement with respect to all states (*i.e.* including $\bot$) or with respect to just proper states (*i.e.* excluding $\bot$). The former operation we call 'complement' and investigate in this section; the latter we call 'proper complement' and investigate in the next. We shall see that both are severely restricted compared with involution on transformers.

The *complement* of a relation $r : X_\bot \leftrightarrow X_\bot$ is defined to be the relation $\overline{r} : X_\bot \leftrightarrow X_\bot$ that is the complement of r in $X_\bot \times X_\bot$, made healthy:

$$\overline{r} \quad \widehat{=} \quad h.((X_\bot \times X_\bot) \setminus r) .$$

Simple examples show that both $\bot$-closure and upclosure are necessary in order for complement to be well-defined on $\mathcal{R}(X)$, the object of our interest. An equivalent definition is $\overline{r} = h.((X \times X_\bot) \setminus r)$.

Complement shares only a few of the properties of involution on $\mathcal{T}(X)$ because it is so severe. Indeed translated to computational terms, the following result shows that if, from an initial state, a command is either not enabled or terminating then its complement diverges there; whilst if the command diverges then its complement is not enabled. So from any initial state the complement is either not enabled or divergent.

Theorem 4 *The complement of a relation is healthy: for any relation r on $X_\perp$, $\overline{r} : \mathcal{R}(X)$. In particular, complement is well defined on $\mathcal{R}(X)$. Furthermore, for any $r : \mathcal{R}(X)$ and any $x : X$,*

$$\overline{r}.(\!| x |\!) \;\; = \;\; (\{\} \lhd x \, r \perp \rhd X_\perp) \, .$$

Proof: By definition $\overline{r}$ is both strict and upclosed, and hence in $\mathcal{R}(X)$.

We observe that if r is healthy and $x \, r \perp$ then $r.(\!| x |\!) = X_\perp$, hence $\overline{r}.(\!| x |\!) = \{\}$ and so the first part of the dichotomy holds. For the second part, if $\neg(x \, r \perp)$ then $\perp \notin r.(\!| x |\!)$ and so $\perp \in \overline{r}.(\!| x |\!)$ which by healthiness means $\overline{r}.(\!| x |\!) = X_\perp$. $\qquad\square$

By having at most two outcomes from each initial state, complement identifies quite different computations. As a result it satisfies its laws really by default:

Corollary 2 *1. For any $r : \mathcal{R}(X)$, $\overline{\overline{r}} \subseteq r$ and for any $x : X_\perp$*

$$\overline{\overline{r}}.(\!| x |\!) \;\; = \;\; r.(\!| x |\!) \quad \textit{iff} \quad r.(\!| x |\!) = \{\} \; \textit{or} \; X_\perp \, .$$

2. For any $r, s : \mathcal{R}(X)$,

$$r \subseteq s \;\; \textit{implies} \;\; \overline{r} \supseteq \overline{s} \qquad \textit{and} \qquad \overline{r \,\fatsemi\, s} \subseteq \overline{r} \,\fatsemi\, \overline{s} \, .$$

Also $\overline{id[X]_\perp} \;=\; X_\perp \times X_\perp$.

3. For any subset E of $\mathcal{R}(X)$, $\overline{\cup E} = \cap \overline{E}$ and $\overline{\cap E} = \cup \overline{E}$. In particular

$$\overline{\{\}_\perp} \;\; = \;\; X_\perp \times X_\perp \;\; = \;\; \overline{(X \times X)_\perp} \quad \textit{and} \quad \overline{X_\perp \times X_\perp} \;\; = \;\; \{\}_\perp \, .$$

4. For any predicate b on X and any $x : X$,

$$[\![\overline{\{b\}}]\!].(\!| x |\!) \;\; = \;\; X_\perp \lhd b.x \rhd \{\} \qquad \textit{and} \qquad [\![\overline{\langle b \rangle}]\!].(\!| x |\!) \;\; = \;\; X_\perp \, .$$

For any relations r and s on $X_\perp$,

$$\overline{r \lhd b \rhd s} \;\; \supseteq \;\; \overline{r} \lhd b \rhd \overline{s} \, .$$

Furthermore each containment (in 1, 2 and 4) may be strict.

Proof:

1. We reason

$$\overline{\overline{r}}.(\!|\,x\,|\!)$$

$=$ previous theorem

$$\{\,\} \lhd x\overline{r} \perp \rhd X_\perp$$

$=$ $\overline{r}$ healthy

$$\{\,\} \lhd \overline{r}.(\!|\,x\,|\!) = X_\perp \rhd X_\perp$$

$=$ r healthy

$$\{\,\} \lhd \neg(x\,r\,\perp) \rhd X_\perp$$

$=$ calculus

$$X_\perp \lhd x\,r\,\perp \rhd \{\,\}$$

from which the claims follow.

2. Co-monotonicity follows because complement is obtained by composing the co-monotone function 'set complement' with the monotone function 'healthy closure'.

For sequential composition we reason

$$\overline{r \,\fatsemi\, s}.(\!|\,x\,|\!)$$

$=$ previous theorem

$$\{\,\} \lhd x\,(r \,\fatsemi\, s)\,\perp \rhd X_\perp$$

$=$ definition of composition

$$\{\,\} \lhd (x\,r\perp \ \vee \ (\neg x\,r\perp \wedge (\exists\,y : r.(\!|\,x\,|\!) \ \cdot \ y\,s\perp))) \rhd X_\perp$$

$=$ calculus

$$\{\,\} \lhd (x\,r\perp \ \vee \ \exists\,y : r.(\!|\,x\,|\!) \ \cdot \ y\,s\perp) \rhd X_\perp$$

$\Rightarrow$ definition of complement

$$\{\,\} \lhd (\overline{r}(\!|\,x\,|\!) = \{\,\} \ \vee \ \exists\,y : r.(\!|\,x\,|\!) \ \cdot \ \overline{s}(y) = \{\,\}) \rhd X_\perp$$

$\Rightarrow$ set theory

$$\{\,\} \lhd (\overline{r}(\!|\,x\,|\!) = \{\,\} \ \vee \ \exists\,y : X_\perp \ \cdot \ \overline{s}(y) = \{\,\}) \rhd X_\perp$$

$$= \qquad\qquad\qquad\qquad \text{definitions of composition and complement}$$

$$\overline{r} \mathbin{\overset{\circ}{\,}_9} \overline{s}.(\!|\, x \,|\!)$$

In general the reverse inclusion does not hold, as shown by considering the relational semantics $r = [\![\mathbf{skip}]\!]$ and $s = [\![\mathbf{abort}]\!]$:

$$\overline{r \mathbin{\overset{\circ}{\,}_9} s} = \overline{[\![\mathbf{skip}]\!] \mathbin{\overset{\circ}{\,}_9} [\![\mathbf{abort}]\!]} = \overline{[\![\mathbf{abort}]\!]} = [\![\mathbf{magic}]\!]$$

whilst

$$\overline{r} \mathbin{\overset{\circ}{\,}_9} \overline{s} = \overline{[\![\mathbf{skip}]\!]} \mathbin{\overset{\circ}{\,}_9} \overline{[\![\mathbf{abort}]\!]} = [\![\mathbf{abort}]\!] \mathbin{\overset{\circ}{\,}_9} [\![\mathbf{magic}]\!] = [\![\mathbf{abort}]\!] \,,$$

which establishes $\overline{r \mathbin{\overset{\circ}{\,}_9} s} \subset \overline{r} \mathbin{\overset{\circ}{\,}_9} \overline{s}$.

The calculation for the identity is routine.

3. De Morgan's laws, and hence the extreme cases, follow similarly.

4. The first two claims result from routine calculation. Firstly

$$\overline{[\![\{b\}]\!]}.(\!|\, x \,|\!)$$
$$= \qquad\qquad\qquad\qquad \text{previous theorem}$$
$$\{\,\} \vartriangleleft x\,[\![\{b\}]\!] \perp \vartriangleright X_\perp$$
$$= \qquad\qquad\qquad\qquad x\,[\![\{b\}]\!] \perp \;=\; \neg b.x$$
$$X_\perp \vartriangleleft b.x \vartriangleright \{\,\}\,,$$

and secondly

$$\overline{[\![\langle b \rangle]\!]}.(\!|\, x \,|\!)$$
$$= \qquad\qquad\qquad\qquad \text{previous theorem}$$
$$\{\,\} \vartriangleleft x\,[\![\langle b \rangle]\!] \perp \vartriangleright X_\perp$$
$$= \qquad\qquad\qquad\qquad \neg x\,[\![\langle b \rangle]\!] \perp \;=\; \textit{false}$$
$$X_\perp\,.$$

For the third claim, for any $x : X_\perp$,

$(\overline{r} \lhd b \rhd \overline{s}).(\!| x |\!)$

$=$ definition of binary conditional

$\overline{r}.(\!| x |\!) \lhd b.x \rhd \overline{s}.(\!| x |\!)$

$=$ definition of complement

$(\{\,\} \lhd x\, r\bot \rhd X_\bot) \lhd b.x \rhd (\{\,\} \lhd x\, s\bot \rhd X_\bot)$

$\Rrightarrow$ calculus

$\{\,\} \lhd (b.x \wedge x\, r\bot) \vee (\neg b.x \wedge x\, s\bot) \rhd X_\bot$

$=$ definition of binary conditional

$\{\,\} \lhd x\, (r \lhd b \rhd x)\bot \rhd X_\bot$

$=$ definition of complement

$\overline{(r \lhd b \rhd s)}.(\!| x |\!)\,.$

To show the inclusion may be strict take, in the relational semantics, $r = [\![\mathbf{skip}]\!]$ and $s = [\![\mathbf{abort}]\!]$ so that the conditional is an assertion. So using the first claim,

$$\overline{r} \lhd b \rhd \overline{s} \;=\; \{\,\}_\bot \qquad \text{whilst} \qquad \overline{r \lhd b \rhd s} \;=\; X_\bot \lhd b.x \rhd \{\,\}\,,$$

hence

$$\overline{r} \lhd b \rhd \overline{s} \;\subset\; \overline{r \lhd b \rhd s}\,.$$

$\square$

Thus complement happens to be sub-involutive on $\mathcal{R}(X)$, co-monotone, sub-distributes sequential composition, interchanges angelic and demonic nondeterminism, (hence) interchanging **abort** and **magic**, takes **skip** to **magic**, takes an assertion to a binary conditional between **abort** and **magic**, and takes a coercion to **abort**. However from the theorem we see that it identifies too many computations to be useful.

7   Proper Complement

The *proper complement* of a relation $r : X_\bot \leftrightarrow X_\bot$ is defined to be the relation $\underline{r} : X_\bot \leftrightarrow X_\bot$ that is the complement of r in $X \times X$, made healthy:

$$\underline{r} \;\; \widehat{=} \;\; h.((X \times X) \setminus r)\,.$$

In fact only $\perp$-closure is necessary there in order for proper complement to be well-defined on $\mathcal{R}(X)$, because proper complement loses information about divergence and so upclosure is redundant.

We find, as a result, that proper complement identifies far fewer computations than does complement: the only behaviours that are identified are divergence and arbitrary terminating behaviour from some state (both, we shall see, being mapped to not enabled behaviour from that state).

Theorem 5 *For any relation r on $X_\perp$,*

$$\underline{r}.(\!|\,x\,|\!) \quad = \quad (X_\perp \lhd x = \perp \rhd X \setminus r.(\!|\,x\,|\!))\,.$$

In particular, proper complement is well defined on $\mathcal{R}(X)$.

Proof: The first identity follows by definition and well-definedness follows from it immediately. $\qquad\qquad\Box$

Proper complement is sub-involutive, $\underline{\underline{r}} \subset r$, is contained in complement, is co-monotone, satisfies De Morgan's laws over nonempty sets, and takes not enabled behaviour to arbitrary termination.

Corollary 3 *1. For any relation r on $X_\perp$ and any $x \colon X_\perp$,*

$$\underline{\underline{r}}.(\!|\,x\,|\!) \quad = \quad (X_\perp \lhd x = \perp \rhd X \cap r.(\!|\,x\,|\!))\,.$$

   Hence proper complement is sub-involutive: $\underline{\underline{r}} \subseteq r$.

2. Proper complement is contained in complement: for any relation r on $X_\perp$, $\underline{r} \subseteq \overline{r}$.

3. For any relations r and s on $X_\perp$

$$r \subseteq s \quad implies \quad \underline{r} \supseteq \underline{s}\,.$$

   But in general $\underline{r \,\fatsemi\, s} \neq \underline{r} \,\fatsemi\, \underline{s}$, and in particular, $\underline{id[X]_\perp} \neq \{\,\}_\perp$.

4. For any set E of relations on $X_\perp$, $\underline{\cap E} = \cup \underline{E}$, and if E is nonempty then $\underline{\cup E} = \cap \underline{E}$. In particular,

$$\underline{\{\,\}_\perp} = (X \times X)_\perp \quad and \quad \underline{X_\perp \times X_\perp} = \{\,\}_\perp = \underline{(X \times X)_\perp}\,.$$

5. *For any predicate b on X and any $x\!:\!X$, in the relational semantics,*

$$\llbracket \underline{\{b\}} \rrbracket.(\!| x |\!) \;\; = \;\; \llbracket \mathbf{neq} \lhd b \rhd \mathbf{magic} \rrbracket.(\!| x |\!)$$

$$\llbracket \underline{\langle b \rangle} \rrbracket.(\!| x |\!) \;\; = \;\; \llbracket \mathbf{neq} \lhd b \rhd \mathbf{choose} \rrbracket.(\!| x |\!) \,.$$

and for any relations r and s on $X_\perp$,

$$\underline{r \lhd b \rhd s} \;\; = \;\; \underline{r} \lhd b \rhd \underline{s} \,.$$

Furthermore each containment (in 1, 2 and 3) may be strict.

Proof:

1. We reason

$$\underline{\underline{r}}.(\!| x |\!)$$
$$=$$ previous theorem
$$X_\perp \lhd x =\perp \rhd (X \setminus \underline{r}.(\!| x |\!))$$
$$=$$ previous theorem and calculus
$$X_\perp \lhd x =\perp \rhd (X \setminus (X \setminus r.(\!| x |\!)))$$
$$=$$ set theory
$$X_\perp \lhd x =\perp \rhd X \cap r.(\!| x |\!) \,.$$

Therefore

$$\underline{\underline{r}} \;\; = \;\; (r \cap (X \times X))_\perp \;\; \subseteq \;\; r \,.$$

The claimed sub-involutivity follows. Routine verification shows that the relational semantics of the commands **choose** and **abort** have the same proper complement, so strict inclusion may hold.

2. We reason from elementary set theory:

$$\mathit{true}$$
$$\Rightarrow$$
$$X \times X \;\; \subseteq \;\; X_\perp \times X_\perp$$
$$\Rightarrow$$ calculus

155

$$(X{\times}X) \setminus r \;\subseteq\; (X_\perp{\times}X_\perp) \setminus r$$

$\Rightarrow$ \hfill h monotone

$$h.((X{\times}X) \setminus r) \;\subseteq\; h.((X_\perp{\times}X_\perp) \setminus r)$$

$\equiv$ \hfill definitions

$$\underline{r} \;\subseteq\; \overline{r}\,.$$

Strictness follows by taking, for example, $r \;\widehat{=}\; X{\times}X$, so that the left-hand side is $\{\perp\}{\times}X_\perp$ and the right-hand side is $X_\perp{\times}X_\perp$.

3. Antimonotonicity follows as in the previous corollary.

However proper complement fails to distribute sequential composition,

$$\underline{r \mathbin{\raisebox{0.2ex}{\tiny$\circ$}\raisebox{-0.2ex}{\tiny$\circ$}} s} \;\neq\; \underline{r} \mathbin{\raisebox{0.2ex}{\tiny$\circ$}\raisebox{-0.2ex}{\tiny$\circ$}} \underline{s}\,, \tag{56}$$

even weakly (*i.e.* in one direction). A simple example demonstrating that in general no inclusion holds between those two sides is obtained by considering state space $X \;\widehat{=}\; \{0,1\}$ and commands **skip** and **neq** (recall that the latter chooses a final state different from the initial state; we are concerned only with state space having more than one element). Their relational semantics are, respectively,

$$r \;=\; [\![\mathbf{skip}]\!] \;=\; \{(0,0),(1,1)\}_\perp$$
$$s \;=\; [\![\mathbf{neq}]\!] \;=\; \{(0,1),(1,0)\}_\perp$$

which are proper complements. We infer that

$$\underline{r \mathbin{\raisebox{0.2ex}{\tiny$\circ$}\raisebox{-0.2ex}{\tiny$\circ$}} s} = \underline{s} = r \qquad \text{and} \qquad \underline{r} \mathbin{\raisebox{0.2ex}{\tiny$\circ$}\raisebox{-0.2ex}{\tiny$\circ$}} \underline{s} = s \mathbin{\raisebox{0.2ex}{\tiny$\circ$}\raisebox{-0.2ex}{\tiny$\circ$}} r = s\,,$$

so the two sides of (56) are nonempty and disjoint on proper states.

It also follows that

$$\underline{id[X]_\perp} \;=\; [\![\mathbf{neq}]\!] \;\neq\; id[X]_\perp\,.$$

4. For a nonempty set E of relations we reason

$$\underline{\cup E}$$

$=$ \hfill definition of proper complement

156

$h.((X {\times} X) \setminus \cup E)$

$=$ calculus and R nonempty

$h.(\cap\{(X {\times} X) \setminus r \mid r \in E\})$

$=$ h distributes intersections

$\cap\{h.((X {\times} X) \setminus r) \mid r \in E\}$

$=$ definition of proper complement

$\cap\{\underline{r} \mid r \in E\}$

$=$ notation

$\cap\underline{E}$.

Taking $E \mathrel{\widehat{=}} \{\,\}$ we find

$\underline{\cup\{\,\}}$

$=$ definition of proper complement

$h.((X \times X) \setminus \cup\{\,\})$

$=$ calculus

$h.(X \times X)$

$=$ definition of h

$(X \times X)_{\perp}$

$\supset$ calculus

$X_{\perp} \times X_{\perp}$

$=$ calculus

$\cap\{\underline{r} \mid r \in \{\,\}\}$

$=$ definition

$\cap\underline{\{\,\}}$.

The dual $\cap\underline{E} = \cup\underline{E}$ follows similarly but there is no need for the restriction to nonempty sets since, if $R \mathrel{\widehat{=}} \{\,\}$,

$\underline{\cap\{\,\}}$

$=$ definition of proper complement

157

$h.((X \times X) \setminus \cap\{\,\})$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀calculus

$h.((X \times X) \setminus (X_\perp \times X_\perp))$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀calculus

$h.(\{\,\})$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀definition of h

$\{\,\}_\perp$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀calculus

$\cup\{\underline{r} \mid r \in \{\,\}\}$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀definition

$\cup\underline{\{\,\}}\,.$

5. We reason

$\underline{[\![\{b\}]\!]}.(\!|\,x\,|\!)$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀characterisation of proper complement

$X_\perp \lhd\ x = \perp\ \rhd (X \setminus [\![\{b\}]\!].(\!|\,x\,|\!))$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀definition of assertion

$X_\perp \lhd\ x = \perp\ \rhd (X \setminus (\{x\} \lhd\ b.x\ \rhd X_\perp))$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀calculus

$X_\perp \lhd\ x = \perp\ \rhd ((X \setminus \{x\}) \lhd\ b.x\ \rhd \{\,\})$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀definitions of relational semantics and (4)

$[\![\mathbf{neq} \lhd\ b\ \rhd \mathbf{magic}]\!].(\!|\,x\,|\!)\,,$

and similarly for coercion.

For the last part

$(\underline{r} \lhd\ b\ \rhd \underline{s}).(\!|\,x\,|\!)$

$=$ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀definition of binary conditional

$\underline{r}.(\!|\,x\,|\!) \lhd\ b.x\ \rhd \underline{s}.(\!|\,x\,|\!)$

158

$$= \qquad\qquad \text{definition of proper complement}$$

$$(X_\perp \;\lhd\; x = \perp \;\rhd\; (X \setminus r.(\!|\,x\,|\!)))$$
$$\lhd \;\; b.x \;\; \rhd$$
$$X_\perp \;\lhd\; x = \perp \;\rhd\; (X \setminus s.(\!|\,x\,|\!))))$$

$$= \qquad\qquad\qquad\qquad \text{calculus}$$

$$X_\perp \;\lhd\; x = \perp \;\rhd\; ((X \setminus r.(\!|\,x\,|\!)) \lhd \;\; b.x \;\; \rhd\; (X \setminus s.(\!|\,x\,|\!)))$$

$$= \qquad\qquad \text{definition of binary conditional}$$

$$X_\perp \;\lhd\; x = \perp \;\rhd\; X \setminus (r.(\!|\,x\,|\!) \lhd \;\; b.x \;\; \rhd\; s.(\!|\,x\,|\!)))$$

$$= \qquad\qquad \text{definition of proper complement}$$

$$\underline{(r \;\lhd\; b \;\rhd\; s)}.(\!|\,x\,|\!) \,.$$

$$\square$$

8 Galois Star

The previous two complement-based attempts at defining an involution on $\mathcal{R}(X)$ satisfied too few laws to be of use, because they failed to preserve important computational distinctions.

An alternative weak involution may be defined by translating to relations the involution on transformers, using the Galois connection of section 4. Accordingly we define the *Galois star* of a relation $r : X_\perp \leftrightarrow X_\perp$ to be the relation $r^\dagger : X_\perp \leftrightarrow X_\perp$,

$$r^\dagger \;\;\hat{=}\;\; rp.((wp.r)^*) \,.$$

Theorem 6 *Galois star is well defined on $\mathcal{R}(X)$: indeed for any relation r on $X_\perp$, $r^\dagger \in \mathcal{R}(X)$. Furthermore for any proper state y,*

$$x \, r^\dagger \, y \;\;\equiv\;\; x = \perp \vee r.(\!|\,x\,|\!) \subseteq \{y\} \,.$$

Proof: Galois star is the composition of three functions, the last of whose range lies in $\mathcal{R}(X)$, and so the first claim follows.

For the second claim we argue routinely that, if $x, y : X_\perp$, then

$$x \, r^\dagger \, y$$

$$\equiv \qquad\qquad\qquad\qquad \text{definition of Galois star}$$

$x\,(rp.(wp.r)^*)\,y$

$\equiv$
definition of rp

$x = \perp \;\lor\; \forall\,q\!:\!\mathrm{pred}\,.X \;\cdot\; (wp.r)^*.q.x \Rightarrow q.y$

$\equiv$
definition of wp and calculus

$x = \perp \;\lor\; \forall\,q\!:\!\mathrm{pred}\,.X \;\cdot\; (\forall\,w\!:\!X \;\cdot\; \neg q.w \lor \neg\,x\,r\,w) \lor q.y$

$\equiv$
calculus

$x = \perp \;\lor\; \forall\,w\!:\!X \;\cdot\; x\,r\,w \Rightarrow (\forall\,q\!:\!\mathrm{pred}\,.X \;\cdot\; q.w \Rightarrow q.y)$

$\equiv$
calculus

$x = \perp \;\lor\; \forall\,w\!:\!X \;\cdot\; x\,r\,w \Rightarrow (w = y)$

$\equiv$
calculus

$x = \perp \;\lor\; r.(\!|\,x\,|\!) \subseteq \{y\}\,.$

$\square$

As a result, $r^\dagger.(\!|\,x\,|\!)$ is severely constrained: it can be empty, a singleton or all of $X_\perp$. In particular we are already able to see the extent to which Galois star † retains the switching of demonic and angelic nondeterminism, exhibited by the transformer involution *. Indeed a command P exhibits demonic nondeterminism or diverges at (proper state) x iff, in the relational semantics, there is no y for which $[\![P]\!].(\!|\,x\,|\!) \subseteq \{y\}$; which holds iff $x \notin dom.[\![P]\!]^\dagger$; which holds iff $[\![P]\!]^\dagger$ is not enabled at x. More precisely, we have:

Corollary 4 *1. For any relation r on $X_\perp$,* $\quad r^{\dagger\dagger} \supseteq r\,.$

2. For any relations r and s on $X_\perp$,

$$r \subseteq s \quad iff \quad r^\dagger \supseteq s^\dagger$$
$$(r \,\mathring{,}\, s)^\dagger \;=\; r^\dagger \,\mathring{,}\, s^\dagger$$
$$id[X]_\perp{}^\dagger = id[X]_\perp\,.$$

3. For any subset E of relations on $X_\perp$,

$$(\cup E)^\dagger = \cap E^\dagger \quad and \quad (\cap E)^\dagger \supseteq \cup E^\dagger\,.$$

In particular,

$$\{\,\}_\perp{}^\dagger = X_\perp \times X_\perp \quad and \quad (X_\perp \times X_\perp)^\dagger = \{\,\}_\perp = ((X \times X)_\perp)^\dagger\,.$$

4. *For any predicate b on X,*

$$[\![\{b\}]\!]^{\dagger} = [\![\langle b \rangle]\!] \quad and \quad [\![\langle b \rangle]\!]^{\dagger} = [\![\{b\}]\!],$$

and for any relations r and s on $X_{\perp}$,

$$(r \lhd b \rhd s)^{\dagger} \;\; = \;\; r^{\dagger} \lhd b \rhd s^{\dagger}.$$

Furthermore the containments in Parts 1 and 3 may be strict.

Proof:

1. Iterating the characterisation of the theorem we find

$$x\, r^{\dagger\dagger}\, y \;\;\; \equiv \;\;\; x = \perp \vee \;\forall\, z : X_{\perp} \;\cdot\; r.(\!|\, x\, |\!) \subseteq \{z\} \Rightarrow (z = y)\,.$$

Evidently if $x\, r\, y$ then that condition holds, and so $r^{\dagger\dagger} \supseteq r$ as required.

Routine calculation shows that Galois star is not injective and so not involutive; a particular strict containment is:

$$(X \times X)_{\perp}{}^{\dagger\dagger} \;\; = \;\; (X_{\perp} \times X_{\perp}) \;\supset\; (X \times X)_{\perp}\,.$$

2. For co-monotonicity we reason

$$r \subseteq s$$
$$\Rightarrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Law (34)}$$
$$wp.r \geq wp.s$$
$$\Rightarrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Law (29)}$$
$$(wp.r)^{*} \leq (wp.s)^{*}$$
$$\Rightarrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Law (35)}$$
$$rp.((wp.r)^{*}) \supseteq rp.((wp.s)^{*})$$
$$\equiv \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{definition of Galois star}$$
$$r^{\dagger} \supseteq s^{\dagger}\,.$$

For sequential composition we reason using simple properties

161

$(r \mathbin{\mathring{\,}_9} s)^{\dagger}$

$=$ definition of Galois star

$rp.((wp.(r \mathbin{\mathring{\,}_9} s))^*)$

$=$ Law (38)

$rp.((wp.r \circ wp.s)^*)$

$=$ Law (27)

$rp.((wp.r)^* \circ (wp.s)^*)$

$=$ Law (40)

$rp.((wp.r)^*) \mathbin{\mathring{\,}_9} rp.((wp.s)^*)$

$=$ definition of Galois star

$r^{\dagger} \mathbin{\mathring{\,}_9} s^{\dagger} \,.$

Let us continue to write $id[X]_{\perp}$ for the healthy identity relation on $X_{\perp}$ but also $id[\mathcal{T}]$ for the identity predicate transformer. Then for the last subclaim we reason

$id[X]_{\perp}{}^{\dagger}$

$=$ definition of Galois star

$rp.((wp.id[X]_{\perp})^*)$

$=$ Law (39)

$rp.(id[\mathcal{T}]^*)$

$=$ Law (28)

$rp.(id[\mathcal{T}])$

$=$ Law (41)

$id[X]_{\perp} \,.$

3. For any subset E of relations on $X_{\perp}$ we reason for the first identity as follows.

$(\cup E)^{\dagger}$

$=$ definition of Galois star

$rp.((wp. \cup E)^*)$

$=$ .... Law (46)

$rp.((\wedge\{wp.r \mid r \in E\})^*)$

$=$ .... Law (30)

$rp. \vee \{(wp.r)^* \mid r \in E\}$

$=$ .... Law (42)

$\cap\{rp.((wp.r)^*) \mid r \in E\}$

$=$ .... definition of Galois star

$\cap\{r^\dagger \mid r \in E\}$

$=$ .... definition

$\cap E^\dagger .$

For the refinement we reason

$\cup E^\dagger$

$=$ .... definition of Galois star

$\cup\{rp.((wp.r)^*) \mid r \in E\}$

$=$ .... Law (43)

$rp. \wedge \{(wp.r)^* \mid r \in E\}$

$=$ .... Law (30)

$rp.((\vee\{wp.r \mid r \in E\})^*)$

$\subseteq$ .... Laws (35), (49)

$rp.((wp. \cap \{r \mid r \in E\})^*)$

$=$ .... definition

$rp.((wp. \cap E)^*)$

$=$ .... definition of Galois star

$(\cap E)^\dagger .$

A simple example showing that the De Morgan containment may be strict, is obtained by taking state space $X \stackrel{\wedge}{=} \{0, 1\}$ and the relational

163

semantics of assignments

$$r = [\![x := 0]\!] = \{(x,0) \mid x \in X\}_\perp$$
$$s = [\![x := 1]\!] = \{(x,1) \mid x \in X\}_\perp ,$$

so that $r \cap s = \{\}_\perp$, $r^\dagger = r$ and $s^\dagger = s$. Thus

$$(r \cap s)^\dagger = X_\perp \times X_\perp \supset r \cup s = r^\dagger \cup s^\dagger .$$

The first extreme case is the vacuous case of the first De Morgan identity. Since the second does not follow from the De Morgan containment just proved, we calculate (the remaining equality being similar),

$(X_\perp \times X_\perp)^\dagger$

$=$ definition of Galois star

$rp.(wp.(X_\perp \times X_\perp)^*)$

$=$ Law (47)

$rp.(false^*)$

$=$ Law (26)

$rp.true$

$=$ Law (44)

$\{\}_\perp .$

4. The arguments for assertions and coercions are similar so we present just one. Choosing to reason from first principles in the relational semantics we have, for any proper states x, y,

$x \, [\![\{b\}]\!]^\dagger \, y$

$\equiv$ definition of Galois star

$x \, rp.((wp.[\![\{b\}]\!])^*) \, y$

$\equiv$ definitions of rp, wp, assertion, involution and calculus

$\forall q : \mathrm{pred}.X \cdot ((b.x \Rightarrow q.x) \wedge (q \neq false)) \Rightarrow q.y$

$\equiv$ calculus

$b.x \wedge (x = y)$

164

$$\equiv \qquad\qquad\qquad\qquad \text{relational semantics, Figure 5}$$

$$x \, [\![\langle b\rangle]\!] \, y \,.$$

For binary conditional we reason from the two algebraic representations:

$$(r \,\vartriangleleft\, b \,\vartriangleright\, s)^\dagger$$
$$= \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Law (22)}$$
$$([\![\langle b\rangle]\!] \mathbin{\overset{\circ}{\circ}} r \,\cup\, [\![\langle\neg b\rangle]\!] \mathbin{\overset{\circ}{\circ}} s)^\dagger$$
$$= \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Part 3}$$
$$([\![\langle b\rangle]\!] \mathbin{\overset{\circ}{\circ}} r)^\dagger \,\cap\, ([\![\langle\neg b\rangle]\!] \mathbin{\overset{\circ}{\circ}} s)^\dagger$$
$$= \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Part 2}$$
$$([\![\langle b\rangle]\!]^\dagger \mathbin{\overset{\circ}{\circ}} r^\dagger) \,\cap\, ([\![\langle\neg b\rangle]\!]^\dagger \mathbin{\overset{\circ}{\circ}} s^\dagger)$$
$$= \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{previous sub-part of 4}$$
$$[\![\{b\}]\!] \mathbin{\overset{\circ}{\circ}} r^\dagger \,\cap\, [\![\{\neg b\}]\!] \mathbin{\overset{\circ}{\circ}} s^\dagger$$
$$= \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Law (21)}$$
$$r^\dagger \,\vartriangleleft\, b \,\vartriangleright\, s^\dagger \,.$$

$$\square$$

Thus Galois star is sup-involutive and co-monotone, obeys just one of the De Morgan laws between angelic and demonic nondeterminism, and half the other. Vitally it distributes sequential composition and binary conditional, preserves **skip**, interchanges assertions and coercions, and in particular the trivial cases of *both* De Morgan laws hold: it interchanges **abort** and **magic**.

9   Applications

Dijkstra and Scholten [6] prove, from their axioms for predicate calculus, that in the transformer semantics the property of being predeterministic is preserved by (general) conditional with pairwise disjoint guards and by iteration. Maddux [13] infers the same results in the transformer semantics (and the analogous result for sequential composition) from the relational

semantics and Tarski's axiomatisation of the calculus of relations due to Boole, De Morgan, Peirce and Schröder.

Here we extend the treatment to include commands as well as code. Concentrating on just the conditional, we provide proofs in the relational semantics that are moderately compelling because they resemble Boolean-algebra proofs by exploiting weak inversion in the shape of Galois star. And we compare those proofs with straight algebraic proofs that require results about co-atomicity under refinement, reflecting our earlier algebraic formalisation of notions related to determinism in Section 3.2.

9.1   Algebraic Approach

The result for binary conditional, proved algebraically, is as follows.

Theorem 7 *If b is a predicate on state space and commands P and Q are deterministic [predeterministic, postdeterministic] then so too is the binary conditional $P \lhd b \rhd Q$.*

Proof: We expand a coerced binary conditional

$$\langle x = x_0 \rangle \mathbin{\S} (P \lhd b \rhd Q)$$

$$= \qquad\qquad\qquad\qquad\qquad \text{definition of coercion and Laws (20), (7)}$$

$$(P \lhd b \rhd Q) \lhd x = x_0 \rhd \mathbf{magic}$$

$$= \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{calculus}$$

$$(P \lhd b.x_0 \rhd Q) \lhd x = x_0 \rhd \mathbf{magic}.$$

From this the claims follow. For example if P and Q are enabled at x_0 then so too is the coerced binary conditional since either $b.x_0$ or not; and if the coerced binary conditional is strictly refined by R then at x_0 either P or Q is strictly refined by R and so R is **magic**.   □

However that result does not extend to (general) conditionals, of which it suffices here to consider the following special case. Recall that for predicates a and b on state space and commands P and Q, the conditional **if** $a \to P \;[]\; b \to Q$ **fi** aborts unless either a or b holds; if just a holds it behaves like P; if just b holds it behaves like Q; and if both hold it behaves (demonic) nondeterministically like either P or Q. Conditional can be defined, without having to define guarded commands, by extending Law (21) (rather than its successor):

$$\mathbf{if}\ a \to P \;[]\; b \to Q\ \mathbf{fi}\ \ \widehat{=}\ \ (\{a\} \mathbin{\S} P) \sqcup (\{b\} \mathbin{\S} Q). \tag{57}$$

We say that a and b are *disjoint* iff their conjunction is false. Then if a and b are not disjoint and P and Q are deterministic [postdeterministic], the conditional is not deterministic [postdeterministic] at some states. Nonetheless the previous theorem generalises from binary conditionals to conditionals in the following result, whose proof extends that of the previous theorem.

Theorem 8 *If a and b are disjoint predicates on state space and P and Q are predeterministic commands then so too is the conditional command* **if** $a \to P \,[]\, b \to Q$ **fi** .

Proof: Now the coerced conditional is expanded one step further since a and b need not be complementary:

$$(\{a\} \,\mathring{,}\, P \,\sqcup\, \{b\} \,\mathring{,}\, Q) \,\triangleleft\, x = x_0 \,\triangleright\, \mathbf{magic}$$

$$= \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{as before}$$

$$((P \,\triangleleft\, a \,\triangleright\, \mathbf{abort}) \,\sqcup\, (Q \,\triangleleft\, b \,\triangleright\, \mathbf{abort})) \,\triangleleft\, x = x_0 \,\triangleright\, \mathbf{magic}$$

$$= \qquad\qquad\qquad\qquad\qquad\qquad\text{calculus, using disjointness}$$

$$(P \,\triangleleft\, a.x_0 \,\triangleright\, (Q \,\triangleleft\, b.x_0 \,\triangleright\, \mathbf{abort})) \,\triangleleft\, x = x_0 \,\triangleright\, \mathbf{magic}$$

and the desired results follow from that. For example if P and Q are enabled at x_0 then so too is the coerced conditional, by case analysis; and if the coerced conditional is strictly refined by R then either P or Q is strictly refined by R at x_0; and so the conditional is co-atomic.   □

9.2   Relational Approach

In this section we use Galois star to provide proofs in the relational semantics that share with Boolean algebra the use of (weak) inversion. The observation that enables us to do so is Law (62):

Theorem 9 *Let P be a command. In the relational semantics, at an arbitrary state x,*

$$P \text{ is postdeterministic} \quad\equiv\quad [\![P]\!]^\dagger.(\!|\,x\,|\!) \supseteq [\![P]\!].(\!|\,x\,|\!) \tag{58}$$

$$P \text{ is enabled} \quad\equiv\quad [\![P]\!]^\dagger.(\!|\,x\,|\!) \subseteq [\![P]\!].(\!|\,x\,|\!) \tag{59}$$

$$P \text{ is deterministic} \quad\equiv\quad [\![P]\!]^\dagger.(\!|\,x\,|\!) = [\![P]\!].(\!|\,x\,|\!) \tag{60}$$

$$P \text{ is post or pre deterministic} \quad\equiv\quad [\![P]\!]^{\dagger\dagger}.(\!|\,x\,|\!) \subseteq [\![P]\!].(\!|\,x\,|\!) \tag{61}$$

$$P \text{ is predeterministic} \quad\equiv\quad ([\![P]\!]^{\dagger\dagger} \cup [\![P]\!]^\dagger).(\!|\,x\,|\!) \subseteq [\![P]\!].(\!|\,x\,|\!) \tag{62}$$

167

Proof: Write $r = [\![P]\!]$ in the relational semantics.

For Equivalence (58), we have

$$r^{\dagger}.(\!| \, x \, |\!) \supseteq r.(\!| \, x \, |\!)$$

$\equiv$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ definition of relational image

$$\forall \, y \colon X_{\perp} \cdot \ x \, r \, y \ \Rightarrow \ x \, r^{\dagger} \, y$$

$\equiv$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\dagger$ theorem

$$\forall \, y \colon X_{\perp} \cdot \ x \, r \, y \ \Rightarrow \ r.(\!| \, x \, |\!) \subseteq \{y\}$$

$\equiv$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ calculus

$$r.(\!| \, x \, |\!) = \{\,\} \ \lor \ \# r.(\!| \, x \, |\!) = 1$$

$\equiv$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ definitions

P not enabled or deterministic at x.

For Equivalence (59),

$$r^{\dagger}.(\!| \, x \, |\!) \subseteq r.(\!| \, x \, |\!)$$

$\equiv$ $\qquad\qquad\qquad\qquad\qquad$ definition of relational image and $\dagger$ theorem

$$\forall \, y \colon X_{\perp} \cdot \ r.(\!| \, x \, |\!) \subseteq \{y\} \ \Rightarrow \ x \, r \, y$$

$\equiv$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ calculus

$$\exists \, y \colon X_{\perp} \cdot \ x \, r \, y$$

$\equiv$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ definition

P enabled at x.

Equivalence (60) is an immediate consequence of the first two.

For (61), the usual calculations show implication from left to right. Assuming now the inclusion on the right, recall that $r^{\dagger\dagger}.(\!| \, x \, |\!)$ can only be empty, singleton or all of $X_{\perp}$. The first case yields nothing; the second case yields $r.(\!| \, x \, |\!)$ a singleton (because if it were larger then $r^{\dagger}.(\!| \, x \, |\!)$ would be empty and so $r^{\dagger\dagger}.(\!| \, x \, |\!)$ would be all of $X_{\perp}$, a contradiction); and the third case yields $r^{\dagger\dagger}.(\!| \, x \, |\!) = X_{\perp} = r.(\!| \, x \, |\!)$. In the first case P is not enabled at x; in the second case P is deterministic at x; and in the third case P aborts at x. Thus P is postdeterministic or predeterministic at x.

Finally (62) follows from (59) and (61), with the observation that P is predeterministic iff it is enabled and either postdeterministic or predeterministic. $\square$

The last theorem of the previous section can now be proved using weak inversion in the guise of (62). For convenience we again write $r = [\![P]\!]$ and $s = [\![Q]\!]$ in the relational semantics and abbreviate the conditional as **iffi**.

Proof: As before, we start by using disjointness to infer

$$[\![\mathbf{iffi}]\!].(\!|\,x\,|\!) \;=\; r.(\!|\,x\,|\!) \;\lhd\; a.x \;\rhd\; (s.(\!|\,x\,|\!) \;\lhd\; b.x \;\rhd\; [\![\mathbf{abort}]\!].(\!|\,x\,|\!))\,.$$

Thus, reasoning to establish (62),

$[\![\mathbf{iffi}]\!]^{\dagger\dagger}.(\!|\,x\,|\!) \cup [\![\mathbf{iffi}]\!]^{\dagger}.(\!|\,x\,|\!)$

$=$ $\hspace{6cm}$ previous inference

$(r.(\!|\,x\,|\!) \;\lhd\; a.x \;\rhd\; (s.(\!|\,x\,|\!) \;\lhd\; b.x \;\rhd\; [\![\mathbf{abort}]\!].(\!|\,x\,|\!)))^{\dagger\dagger}$
$\cup$
$(r.(\!|\,x\,|\!) \;\lhd\; a.x \;\rhd\; (s.(\!|\,x\,|\!) \;\lhd\; b.x \;\rhd\; [\![\mathbf{abort}]\!].(\!|\,x\,|\!)))^{\dagger}$

$=$ $\hspace{4cm}$ Parts 4, 3 of † corollary and calculus

$(r^{\dagger\dagger}.(\!|\,x\,|\!) \cup r^{\dagger}.(\!|\,x\,|\!)$
$\lhd\; a.x \;\rhd$
$(s^{\dagger\dagger}.(\!|\,x\,|\!) \cup s^{\dagger}.(\!|\,x\,|\!)) \lhd\; b.x \;\rhd\; [\![\mathbf{abort}]\!].(\!|\,x\,|\!))$

$\subseteq$ $\hspace{6cm}$ assumption on r and s

$r.(\!|\,x\,|\!) \;\lhd\; a.x \;\rhd\; (s.(\!|\,x\,|\!) \;\lhd\; b.x \;\rhd\; [\![\mathbf{abort}]\!].(\!|\,x\,|\!))$

$=$ $\hspace{7cm}$ notation

$[\![\mathbf{iffi}]\!].(\!|\,x\,|\!)\,,$

as required. $\hspace{12cm}$ □

The theorem of the previous section, for binary conditional, can be established by similar techniques (as can the appropriate determinism of other combinators).

10 Conclusion

This paper has promoted the use of algebra—of combinators in the program and command calculi—to express properties, and reason about them, in situations where semantic reasoning is more usual (*e.g.* termination). That has enabled us to extend certain definitions from programs to commands, and reason in a uniform way about both (*e.g.* in Section 9). The approach is continued in [19].

But our explicit target has been the extent to which the structurally-important and well-behaved involution on predicate transformers carries over to binary relations. After showing that the relational structure is not consistent with an involution, two 'weak' involutions are considered but discarded as being too weak in the sense that they fail to distinguish too many computations. A third involution—still weak but better behaved than the others—is defined by translating the transformer involution using the weakest-precondition Galois connection between relations and transformers. That weak involution, Galois star, is shown to be just strong enough to facilitate algebraic reasoning in a simple benchmark situation: preservation of forms of determinism (but generalised from programs to commands). The conclusion is that Galois star does afford the kind of reasoning with which we are familiar from Boolean algebra, even though it is inevitably doomed to be weaker.

Acknowledgements

References

[1] R.-J. R. Back. Combining angels, demons and miracles in program specifications. *Theoretical Computer Science*, **100**(2):365–383, 1992.

[2] R.-J. R. Back and J. von Wright. Duality in specification languages: a lattice-theoretical approach. *Acta Informatica*, **27**(7):583–625, 1990.

[3] R.-J. R. Back and J. von Wright. *Refinement Calculus: a Systematic Introduction*. Springer Verlag, 1998.

[4] P. Cousot and R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages*, 238–252, January 1977.

[5] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall International, 1976.

[6] E. W. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics*. Springer Verlag, 1990.

[7] W.-P. de Roever and K. Engelhardt, *Data Refinement: Model-Oriented Proof Methods and their Comparison.* Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1998.

[8] P. H. B. Gardiner and Carroll Morgan. A single complete rule for data refinement. *Formal Aspects of Computing,* **5**(4):367–382, 1993.

[9] G. Gierz, K. H. Hofman, K. Keimel, J. D. Lawson, M. Mislove and D. S. Scott, *A Compendium of Continuous Lattices.* Springer Verlag, 1980.

[10] C. A. R. Hoare *et al.* The laws of programming. *Communications of the ACM,* **30**(8):672–686, 1987.

[11] C. A. R. Hoare and J. He, *Unifying Theories of Programming.* Prentice-Hall series in Computer Science, 1998.

[12] J. Lambek and P. J. Scott, *Introduction to Higher-Order Categorical Logic.* Cambridge University Press, 1986.

[13] R. D. Maddux. A working relational model: The derivation of the Dijkstra-Scholten predicate transformer semantics from Tarski's axioms for the Peirce-Schröder calculus of relations. *South African Computer Journal,* **9**:92–130, 1993.

[14] C. Morgan. *Programming from Specifications,* second edition. Prentice-Hall International, 1994.

[15] J. M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Science of Computer Programming,* **9**(3):287–306, 1987.

[16] G. Nelson. A generalisation of Dijkstra's calculus. *ACM Transactions on Programming Language and Systems,* **11**(4):517–561, 1989.

[17] O. Ore, Galois connexions. *Transactions of the American Mathematical Society,* **55**:494–513, 1944.

[18] K. I. Rosenthal. *Quantales and their Applications.* Pitman Research Notes in Mathematics, **234**, Longman, 1990.

[19] J. W. Sanders. Computations as fibre bundles. In *Proceedings of Relational Methods in Computer Science 2006,* edited by R. A. Schmidt, *LNCS* **4136**:30–62, Springer-Verlag, 2006.