Kim Solin | Joakim von Wright

Refinement Algebra Extended with Operators for Enabledness and Termination

Turku Centre *for* Computer Science

TUCS Technical Report
No 658, January 2005

Refinement Algebra Extended with Operators for Enabledness and Termination

Kim Solin
University of Turku, Department of Mathematics
FIN-20014 Turku, Finland
kim.solin@utu.fi
Joakim von Wright
Åbo Akademi University, Department of Computer Science
Lemminkäisenkatu 14 A
FIN-20520 Turku, Finland
jockum.wright@abo.fi

TUCS Technical Report
No 658, January 2005

Abstract

Refinement algebras are axiomatisations intended for reasoning about programs in a total correctness framework. We reduce von Wright's demonic refinement algebra to only allow strong iteration and introduce two operators for modelling enabledness and termination of programs, respectively. We show how the operators can be used for expressing properties between programs and apply the algebra to reasoning about action systems.

Keywords: refinement algebra, total correctness, action systems, predicate transformers

TUCS Laboratory
Discrete Mathematics for Information Technology
Learning and Reasoning

and

CREST – the Centre for Reliable Software Technology

1 Introduction

Refinement algebras are axiomatisations for reasoning about program refinement [1, 15, 4], total-correctness preserving program transformations. The intended models are different classes of predicate transformers over a fixed state space. Applications include reasoning about distributed systems, data refinement, and program inversion [17, 18].

In this report, which is an elaboration of our earlier work [16], we introduce a reduct of von Wright's refinement algebra [17]. It differs from previous approaches to abstract algebraic total-correctness reasoning [17, 18, 16, 8, 9] in that it only has one iteration operator, the strong iteration operator. In a program intuition, a strong iteration of a statement *either* terminates or goes on infinitely. Conjunctive predicate transformers form a motivating model for the axiomatisation.

Along the lines of von Wright in [17, 18], we extend the algebra with guards and assertions. Guards should be thought of as programs that check if some predicate holds, skips if that is the case, and otherwise a miracle occurs. Assertions are similar, but instead of performing a miracle when the predicate does not hold, they abort. That is to say, an assertion that is executed in a state where the predicate does not hold establishes no postcondition. We prove a result concerning the characterisation of assertions: defining assertions explicitly as a function of guards is equivalent to defining assertions implicitly by means of Galois connections.

We also extend the refinement algebra with two operators. The first one maps elements in the carrier set to the subset of guards. The intuition is that the operator applied to a program returns a guard that skips in those states in which the program is enabled. This operator is axiomatised in the same way as the domain operator in [10]. The second operator, which is a "dual" operator to the first one, maps elements in the carrier set to the subset of assertions. This operator determines if a program will terminate or not.

Different properties between programs, such as exclusion and a program inversion condition, can be expressed using the new operators. Moreover, we encode actions systems [3, 2] into refinement algebra and use it for proving refinement properties of them.

Five papers stand out in the lineage of this report. Kozen's axiomatisation of Kleene algebra and his introduction of tests into the algebra has been a very significant inspiration for this work [13, 14]. Von Wright's non-conservative extension of Kleene algebra with tests was the first axiomatisation that was genuinely an algebra for total correctness [17]. It rests upon previous work on algebraic program reasoning by Back and von Wright [5]. Finally, Desharnais, Möller, and Struth's Kleene algebra with domain extended Kleene algebra with a domain operator [10]. This algebra has successfully been applied to reasoning about different structures [6]. The domain operator appears again in this report and is given a program-theoretic interpretation.

The report is organised as follows. We begin by presenting a refinement algebra and extend it with guards and assertions in Section 2. In the following section, we introduce the new operators and investigate their basic algebraic properties. Section 4 contains applications, upon which the succeeding section gives an account of a predicate-transformer model for the algebra. We end with some concluding

remarks and an outlook on future work.

We strive to use a calculational proof format.

2 Refinement Algebra

In this section we present a refinement algebra and extend it with guards and assertions.

2.1 A refinement algebra

With an $\mathcal{R}$ *structure* we shall understand a structure over the signature

$$(\sqcap, ; , ^\omega , \top, 1)$$

that satisfies the identities (; left implicit)

$$\begin{aligned}
x \sqcap (y \sqcap z) &= (x \sqcap y) \sqcap z && (1)\\
x \sqcap y &= y \sqcap x && (2)\\
x \sqcap \top &= x && (3)\\
x \sqcap x &= x && (4)\\
x(yz) &= (xy)z && (5)\\
1x &= x && (6)\\
x1 &= x && (7)\\
\top x &= \top && (8)\\
x(y \sqcap z) &= xy \sqcap xz && (9)\\
(x \sqcap y)z &= xz \sqcap yz && (10)\\
x^\omega &= xx^\omega \sqcap 1 && (11)
\end{aligned}$$

and the equational implication

$$xz \sqcap y \sqcap z = xz \sqcap y \;\Rightarrow\; x^\omega y \sqcap z = x^\omega y$$

Moreover, define $\sqsubseteq$ as

$$x \sqsubseteq y \overset{\text{def}}{\Leftrightarrow} x \sqcap y = x \qquad (12)$$

Then the equational implication can be written as

$$xz \sqcap y \sqsubseteq z \;\Rightarrow\; x^\omega y \sqsubseteq z \qquad (13)$$

and (3) as

$$x \sqsubseteq \top$$

If $x \sqsubseteq y$, we can also write $y \sqsupseteq x$.

Note that $\sqsubseteq$ is a partial order and, by (3), $\top$ is its top element, and also that x^ω is a least fixpoint with respect to $\sqsubseteq$. All operators are isotone in all their arguments with respect to $\sqsubseteq$.

The axiomatisation is similar to Kozen's Kleene algebra [13]. The difference is that there is no right annihilation axiom, so $x\top = \top$ does not hold in general, and that * is replaced by $^\omega$.

The operator $^\omega$ bears similarities to the iteration operator in Cohen's conservative extension of Kleene algebra, ω-algebra [8, 9], but is different. Cohen's $^\omega$ can be given the classic interpretation of infinite strings over an alphabet or be interpreted as an infinite repetition of a program statement. Our $^\omega$ should be seen as a repetition of a program statement that *either* terminates *or* goes on infinitely. The Kleene-algebra * can clearly not be defined in terms of the other operators of an $\mathcal{R}$ structure.

The reason for not having a right annihilation axiom is that we want to reason about non-termination, we want a total correctness framework. Right annihilation would prevent that (this is elaborated further below, and a semantical clarification is given in Section 5). In ω-algebra right annihilation holds, so it is really an algebra for partial correctness.

The intention to reason about total correctness also motivates the restriction of the framework to one iteration operator. The refinement algebra by von Wright in [17, 18] has two related iteration operators, one equal to our $^\omega$ and the other equal to * in Kleene algebra. The * can be seen as a terminating repetition of a program statement. However, since total correctness is what we are interested in, we exclude * in our axiomatisation to get a more comprehensible framework.

In a program intuition the elements of the carrier set should be seen as program statements. The operator ; is sequential composition and $\sqcap$ is demonic choice. When $x \sqcap y$ is executed, a choice over which we have no influence is made between x and y. The iteration operator $^\omega$ is, as mentioned earlier, thought of as a terminating *or* infinitely repeating execution of a program statement. The order $\sqsubseteq$ is refinement, $x \sqsubseteq y$ means that y establishes anything that x does and possibly more. Finally, $\top$ is interpreted as magic, a non-implementable program statement that establishes *any* postcondition, and 1 is skip. A semantical justification for this intuition is given in terms of predicate transformers in Section 5.

We also introduce a syntactic constant $\bot$ with the intuition that it stands for an always infinitely executing program statement, an abort statement [17]:

$$\bot \stackrel{\text{def}}{=} 1^\omega$$

The statement 1^ω can be thought of as a loop with the loop condition true. It is easily seen that $\bot$ is a bottom element

$$\bot \sqsubseteq x \tag{14}$$

and that it is left annihilating

$$\bot x = \bot \tag{15}$$

The algebraic reason for excluding $x\top = x$ is now apparent. Since $\bot$ is a bottom element and since

$$\bot = \bot\top = \top$$

holds if $x\top = \top$, we would only get a one-point model.

Many properties of Kleene-algebra * have analogs for $^\omega$. For example,

$$x(yx)^\omega = (xy)^\omega x \tag{16}$$

$$(x \sqcap y)^\omega = x^\omega (yx^\omega)^\omega \tag{17}$$

and

$$yx \sqsubseteq xz \Rightarrow y^\omega x \sqsubseteq xz^\omega \tag{18}$$

hold. However, there are differences as the fact that

$$xz \sqsubseteq yx \Rightarrow xz^\omega \sqsubseteq y^\omega x \tag{19}$$

does not hold in general (take $y = 1$) reveals [17].

2.2 Guards and assertions

An element p of the carrier set that has a complement $\bar{p}$ satisfying

$$p\bar{p} = \top \qquad \text{and} \qquad p \sqcap \bar{p} = 1 \tag{20}$$

is called a *guard*. It is easily established that the guards form a Boolean algebra over $(\sqcap, ;, \bar{}, 1, \top)$, where ; is meet, $\sqcap$ is join, $\bar{}$ is complement, 1 is the zero, and $\top$ is the unit.

Every guard is defined to have a corresponding *assertion*

$$p^\circ = \bar{p}\bot \sqcap 1 \tag{21}$$

Thus $^\circ$ is a mapping from guards to a subset of the carrier set.

Intuitively, guards are statements that check if a predicate holds and, if so, skip, otherwise abort. Assertions are similar, but do magic if the predicate does not hold. The assertions have the properties

$$(p_1 p_2)^\circ = p_1^\circ p_2^\circ = p_1^\circ \sqcap p_2^\circ, \quad p^\circ \bar{p}^\circ = \bot, \quad \text{and} \quad p^\circ p^\circ = p \tag{22}$$

These are easily verified; similarly that we have

$$p^\circ \sqsubseteq 1 \sqsubseteq p \tag{23}$$

for any assertion and any guard [17].

The assertions could implicitly have been defined by Galois connections as the following proposition shows.

Proposition 1. *Let x and y be any elements in the carrier set of an $\mathcal{R}$ structure and let p be any guard in the same set. Then the equality*

$$p^\circ = \bar{p}\bot \sqcap 1$$

holds if and only if the Galois connections

$$p^\circ x \sqsubseteq y \;\Leftrightarrow\; x \sqsubseteq py \qquad \textit{and} \qquad xp \sqsubseteq y \;\Leftrightarrow\; x \sqsubseteq yp^\circ \tag{24}$$

do.

Proof. The calculations make use of the monotonicity of the operators and the easily proved facts that

$$\bot \sqsubseteq 1 \Rightarrow \bar{p}\bot \sqcap p \sqsubseteq \bar{p} \sqcap p \tag{25}$$

and that

$$p(\bar{p}\bot \sqcap 1) = \top \sqcap p \tag{26}$$

For one direction we follow [17]. Assume then that (21) holds. If $p^\circ x \sqsubseteq y$, we have

$$\begin{array}{ll} & py \\ \sqsupseteq & \{\text{assumption}\} \\ & p(\bar{p}\bot \sqcap 1)x \\ = & \{(8, 26)\} \\ & \top \sqcap px \\ \sqsupseteq & \{(23, 3)\} \\ & x \end{array}$$

If $x \sqsubseteq py$, then

$$\begin{array}{ll} & (\bar{p}\bot \sqcap 1)x \\ \sqsubseteq & \{\text{assumption, (10)}\} \\ & \bar{p}\bot \sqcap py \\ \sqsubseteq & \{(8, 10)\} \\ & (\bar{p}\bot \sqcap p)y \\ \sqsubseteq & \{(25, 20)\} \\ & y \end{array}$$

If $xp \sqsubseteq y$, then

$$\begin{array}{ll} & y(\bar{p}\bot \sqcap 1) \\ \sqsupseteq & \{\text{assumption, (9)}\} \\ & xp\bar{p}\bot \sqcap xp \\ \sqsupseteq & \{(10, 26)\} \\ & x(\top \sqcap p) \\ \sqsupseteq & \{(23, 3)\} \\ & x \end{array}$$

Finally, if $x \sqsubseteq yp^\circ$, then

$$\begin{array}{ll} & xp \\ \sqsubseteq & \{\text{assumption, (21)}\} \\ & y\bar{p}\bot \sqcap yp \\ \sqsubseteq & \{(9, 25)\} \\ & y(\bar{p} \sqcap p) \\ = & \{(20)\} \\ & y \end{array}$$

For the other direction it suffices to observe that one part of a Galois connection is *uniquely* defined by the other and that (21) satisfies (24). □

The following propositions summarise some important properties of guards and assertions that will be used later on.

Proposition 2 ([17]). *Let x be an element in the carrier set of an $\mathcal{R}$ structure and let p_1 and p_2 be any guards in the same set. Then*

$$\top \sqsubseteq p_1 x \bar{p_2} \Rightarrow p_1 x \sqsupseteq p_1 x p_2 \Leftrightarrow x p_2 \sqsubseteq p_1 x \tag{27}$$

Proposition 3. *Let x be an element in the carrier set of an $\mathcal{R}$ structure and let p be any guard in the same set. Then*

$$\top \sqsubseteq \bar{p} x \quad \Leftrightarrow \quad p x \sqsubseteq x \tag{28}$$
$$\bar{p}^{\circ} x \sqsubseteq \bot \quad \Leftrightarrow \quad x \sqsubseteq p^{\circ} x \tag{29}$$

Proof. Firstly,

$$\begin{array}{ll} & gx \sqsubseteq x \\ \Rightarrow & \{\text{monotonicity}\} \\ & \bar{g}gx \sqsubseteq \bar{g}x \\ \Leftrightarrow & \{\text{guard property}\} \\ & \top x \sqsubseteq \bar{g}x \\ \Leftrightarrow & \{(8)\} \\ & \top \sqsubseteq \bar{g}x \end{array}$$

and secondly

$$\begin{array}{ll} & \top \sqsubseteq \bar{g}x \\ \Rightarrow & \{\text{monotonicity}\} \\ & \top \sqcap gx \sqsubseteq \bar{g}x \sqcap gx \\ \Leftrightarrow & \{(3,\ 2)\} \\ & gx \sqsubseteq (\bar{g} \sqcap g)x \\ \Leftrightarrow & \{\text{guard property}\} \\ & gx \sqsubseteq 1x \\ \Leftrightarrow & \{(6)\} \\ & gx \sqsubseteq x \end{array}$$

This establishes (28).

Then, for one direction of (29) calculate

$$\begin{array}{ll} & x \sqsubseteq p^{\circ} x \\ \Rightarrow & \{\text{monotonicity}\} \\ & \bar{p}^{\circ} x \sqsubseteq \bar{p}^{\circ} p^{\circ} x \\ \Leftrightarrow & \{\text{assertion property}\} \\ & \bar{p}^{\circ} x \sqsubseteq \bot x \\ \Leftrightarrow & \{\bot \text{ property}\} \\ & \bar{p}^{\circ} x \sqsubseteq \bot \end{array}$$

For the other direction, first note that the left hand side is equivalent to

$$p\bot \sqcap x \sqsubseteq \bot$$

by (21), (10), and (15). Now assume that this holds. Then

$$\begin{array}{ll} & x \\ \sqsubseteq & \{\text{guard property}\} \\ & \bar{p}x \\ = & \{(3)\} \\ & (\bar{p} \sqcap \top)x \\ = & \{(26)\} \\ & \bar{p}(p\bot \sqcap 1)x \\ \sqsubseteq & \{\text{assumption}\} \\ & \bar{p}\bot x \\ = & \{(15)\} \\ & \bar{p}\bot \end{array}$$

Since

$$x \sqsubseteq p^\circ x \Leftrightarrow x \sqsubseteq \bar{p}\bot \sqcap x \Leftrightarrow x \sqsubseteq \bar{p}\bot$$

this proves the claim. □

Proposition 4. *Let x be an element in the carrier set of an $\mathcal{R}$-structure and let p_1 and p_2 be any guards in the same set. Then*

$$xp_1 \sqsupseteq p_2xp_1 \Leftrightarrow p_2x \sqsubseteq xp_1 \tag{30}$$

$$xp_1^\circ \sqsubseteq p_2^\circ xp_1^\circ \Leftrightarrow p_2x \sqsupseteq xp_1 \tag{31}$$

Proof. Assume $xp_1 \sqsupseteq p_2xp_1$. Since $1 \sqsubseteq p$ for any guard p, this means that

$$xp_1 = p_2xp_1$$

Then

$$xp_1 = p_2xp_1 \sqsupseteq p_2x$$

Conversely, assume that $p_2x \sqsubseteq xp_1$. Then

$$p_2xp_1 \sqsubseteq xp_1p_1 = xp_1$$

The case for assertions is proved in a similar fashion. □

Proposition 5. *Let p be a guard in the carrier set of any $\mathcal{R}$ structure. Then*

$$pp^\circ = p \tag{32}$$

$$p^\circ p = p^\circ \tag{33}$$

Proof. The first statement is proved directly by

$$\begin{array}{ll} & pp^\circ \\ = & \{(21,\,9)\} \\ & p\bar{p}\bot \sqcap p \\ = & \{\text{guard property}\} \\ & \top\bot \sqcap p \\ = & \{(8,\,3)\} \\ & p \end{array}$$

For the second statement, first note that it is clear by (23) that $p^\circ \sqsubseteq p^\circ p$. For the other direction calculate

$$\begin{array}{ll} & p^\circ p \sqsubseteq p^\circ \\ \Leftrightarrow & \{(24)\} \\ & p \sqsubseteq pp^\circ \\ \Leftrightarrow & \{\text{first direction}\} \\ & p \sqsubseteq p \\ \Leftrightarrow & \{\sqsubseteq \text{ partial order}\} \\ & \mathsf{True} \end{array}$$

□

3 Enabledness and Termination

In this section we introduce two new operators, the enabledness operator and the termination operator, and investigate their basic properties.

3.1 Enabledness

Let ϵ be a unary operator on an $\mathcal{R}$ structure, such that it maps an element of the carrier set to a guard and satisfies the following axioms

$$\epsilon x x = x \tag{34}$$

$$p \sqsubseteq \epsilon(px) \tag{35}$$

$$\epsilon(xy) = \epsilon(x\epsilon y) \tag{36}$$

By convention, ϵ binds stronger than the other operators, so e.g. $\epsilon x x$ is $(\epsilon x)x$.

The intuition behind ϵ is that it maps any program to a guard that skips in those states in which the program is enabled, that is, in those states from which the program will not terminate miracuosly. Axiom (34), for example, says that a program x equals a program that first checks if x is enabled and then executes x.

ϵ is axiomatised as the domain operator of KAD (Kleene algebra with domain) [10]. To see this, note that our $\sqsubseteq$ corresponds to $\geq$ in Kleene algebra. This means that many properties of the domain operator in [10] will also hold for ϵ in

our framework, but we need to remember the lack of right annihilation and that $^\omega$ is different from *.

As in KAD, the first two axioms of ϵ can be replaced by the equivalence

$$px \sqsubseteq x \Leftrightarrow p \sqsubseteq \epsilon x \tag{37}$$

This can proved by reusing and slightly modifying proofs from [10], as can the properties

$$\begin{aligned} \epsilon(x \sqcap y) &= \epsilon x \sqcap \epsilon y & (38)\\ x \sqsubseteq y &\Rightarrow \epsilon x \sqsubseteq \epsilon y & (39) \end{aligned}$$

3.2 Termination

Let τ be a unary operator on an $\mathcal{R}$ structure such that it maps an element in the carrier set to an assertion, and satisfies the following axioms

$$\begin{aligned} x &= \tau x x & (40)\\ \tau(p^\circ x) &\sqsubseteq p^\circ & (41)\\ \tau(x\tau y) &= \tau(xy) & (42) \end{aligned}$$

By convention, τ has the same precedence as ϵ.

The operator τ applied to a program denotes those states from which the program is guaranteed to terminate, that is, states from which it will not abort. Axiom (41) says that a program that checks if the program $p^\circ x$ will terminate can be replaced by the assertion p°. That this holds, is due to the fact that the program $p^\circ x$'s termination is determined by the assertion. The same line of thinking, gives that

$$\tau p^\circ = p^\circ$$

That the first two τ-axioms have a characterization analogous to (37) and is additive and isotone can be shown.

Proposition 6. *Let x be an element in the carrier set of an $\mathcal{R}$ structure and let p be any guard in the same set. Then*

$$x \sqsubseteq p^\circ x \Leftrightarrow \tau x \sqsubseteq p^\circ \tag{43}$$

Proof. Right to left is equivalent to the first axiom:

$$x \sqsubseteq \tau x x \;\Leftrightarrow\; (x \sqsubseteq p^\circ x \Leftarrow \tau x \sqsubseteq p^\circ)$$

$(\Leftarrow)$:

$$\begin{aligned} & x \sqsubseteq p^\circ x \Leftarrow \tau x \sqsubseteq p^\circ\\ \Rightarrow\; & \{\text{let } p^\circ := \tau x\}\\ & x \sqsubseteq \tau x x \Leftarrow \tau x \sqsubseteq \tau x\\ \Leftrightarrow\; & \{\sqsubseteq \text{ reflexive}\}\\ & x \sqsubseteq \tau x x \Leftarrow \mathsf{True}\\ \Leftrightarrow\; & \{\text{logic}\}\\ & x \sqsubseteq \tau x x \end{aligned}$$

$(\Rightarrow)$:

$$\begin{array}{ll} & x \sqsubseteq \tau x x \wedge \tau x \sqsubseteq p^\circ \\ \Rightarrow & \{\text{monotonicity of ;, } \sqsubseteq \text{ transitivity}\} \\ & x \sqsubseteq p^\circ x \end{array}$$

Left to right is equivalent to the second axiom:

$$\tau(p^\circ x) \sqsubseteq p^\circ \Leftrightarrow (x \sqsubseteq p^\circ x \Rightarrow \tau x \sqsubseteq p^\circ)$$

$(\Leftarrow)$:

$$\begin{array}{ll} & x \sqsubseteq p^\circ x \Rightarrow \tau x \sqsubseteq p^\circ \\ \Rightarrow & \{\text{let } x := p^\circ x\} \\ & p^\circ x \sqsubseteq p^\circ p^\circ x \Rightarrow \tau(p^\circ x) \sqsubseteq p^\circ \\ \Leftrightarrow & \{(22), \sqsubseteq \text{ reflexive }\} \\ & \tau(p^\circ x) \sqsubseteq p^\circ \end{array}$$

$(\Rightarrow)$:

$$\begin{array}{ll} & x \sqsubseteq p^\circ x \wedge \tau(p^\circ x) \sqsubseteq p^\circ \\ \Rightarrow & \{(23)\} \\ & x = p^\circ x \wedge \tau(p^\circ x) \sqsubseteq p^\circ \\ \Rightarrow & \{\sqsubseteq \text{ transitive}\} \\ & \tau x \sqsubseteq p^\circ \end{array}$$

□

We can also define τ explicitly by

$$\tau x = x\top \sqcap 1 \tag{44}$$

It is quite elementary to show that the right hand side of (44) satisfies the axioms. To see that τ is unique, suppose that another function f satisfies the axioms. Then by (43)

$$\tau x \sqsubseteq p \Leftrightarrow fx \sqsubseteq p$$

which by the principle of indirect equality means that

$$\tau x = fx$$

Proposition 7. *Let x and y be any elements in the carrier set of an $\mathcal{R}$ structure and let p be any guard in the same set. Then*

$$\tau x \sqsubseteq p^\circ \quad \Leftrightarrow \quad \bar{p}^\circ x \sqsubseteq \bot \tag{45}$$

$$\tau(x \sqcap y) \quad = \quad \tau x \sqcap \tau y \tag{46}$$

$$x \sqsubseteq y \quad \Rightarrow \quad \tau x \sqsubseteq \tau y \tag{47}$$

Proof. Propositions 6 and 3 establish (45). The calculation

$$\begin{array}{ll} & \tau(x \sqcap y) \\ = & \{(44)\} \\ & (x \sqcap y)\top \sqcap 1 \\ = & \{(10)\} \\ & x\top \sqcap y\top \sqcap 1 \\ = & \{(2,\ 4)\} \\ & x\top \sqcap 1 \sqcap y\top \sqcap 1 \\ = & \{(44)\} \\ & \tau x \sqcap \tau y \end{array}$$

settles (46). Last, too see that τ is monotone first assume that $x \sqsubseteq y$. By (12) this means that $x = x \sqcap y$. Using (46) and the assumption we can then see that

$$\tau x \sqcap \tau y = \tau(x \sqcap y) = \tau x$$

This says exactly that $\tau x \sqsubseteq \tau y$. □

3.3 Some basic properties

In this section we investigate some of the basic properties of ϵ, τ and $^\omega$. The investigation reveals that all propositions that can be shown in KAD regarding δ and * [10], for example the induction rule, do not necessarily hold for ϵ and $^\omega$ in an $\mathcal{R}$ structure.

Proposition 8. *Let* 1, $\top$ *and* $\bot$ *be the constants in an* $\mathcal{R}$ *structure. Then*

$$\epsilon 1 = 1 \tag{48}$$
$$\tau 1 = 1 \tag{49}$$
$$\epsilon \top = \top \tag{50}$$
$$\tau \bot = \bot \tag{51}$$

Proof. The first two statements follow from the fact that $11 = 1$ and axioms (34) and (40), respectively. The third part follows from $\top$ being a top element and axiom (35), whereas the fourth part follows from $\bot$ being a bottom element and (41). □

Proposition 9. *Let* x *and* y *be any elements in an* $\mathcal{R}$ *structure. Then*

$$\epsilon(\tau x) = 1 = \tau(\epsilon x) \tag{52}$$
$$\epsilon x \tau x = \tau x \epsilon x \tag{53}$$
$$\epsilon(x \tau y) = \epsilon x \tag{54}$$
$$\tau(x \epsilon y) = \tau x \tag{55}$$

Proof. The first part. $1 \sqsubseteq \epsilon(\tau x)$ follows from (22), whereas

$$\epsilon(\tau x) \sqsubseteq \epsilon(1) \sqsubseteq \epsilon(11) = 1$$

follows from (22), (34) and properties of 1. $\tau(\epsilon x) \sqsubseteq 1$ follows from (22), whereas

$$1 = \tau 11 = \tau 1 \sqsubseteq \tau(\epsilon x)$$

follows from (22), (40) and properties of 1.

The second part can be done as a direct calculation:

$$\begin{array}{ll} & \epsilon x \tau x \\ = & \{(34, 9, 44)\} \\ & x\top \sqcap \epsilon x \\ = & \{(8)\} \\ & x\top \epsilon x \sqcap \epsilon x \\ = & \{(10)\} \\ & (x\top \sqcap 1)\epsilon x \\ = & \{(44)\} \\ & \tau x \epsilon x \end{array}$$

The third part follows from

$$\begin{array}{ll} & \epsilon(x\tau y) \\ = & \{(36)\} \\ & \epsilon(x\epsilon(\tau y) \\ = & \{(23)\} \\ & \epsilon(x1) \\ = & \{(7)\} \\ & \epsilon x \end{array}$$

The fourth part is shown similarly as the third. □

Proposition 10. *Let x be an element in an $\mathcal{R}$ structure. Then*

$$(\epsilon x)^\omega \sqsubseteq 1 \tag{56}$$
$$\epsilon(x^\omega) = 1 \tag{57}$$
$$(\tau x)^\omega = \bot \tag{58}$$
$$\tau(x^\omega) \sqsubseteq 1 \tag{59}$$

Proof. The first part holds since $(\epsilon x)^\omega = \epsilon x(\epsilon x)^\omega \sqsubseteq 1$. For a counter example to see that the converse does not hold, take $x = 1$. The second part holds since $\epsilon(x^\omega) \sqsubseteq \epsilon 1 = 1$ by (11) and (34), and the converse follows from (23). For the third part, note that one way follows from $\bot$ being a bottom element. The other way follows from (23) by $(\tau x)^\omega \sqsubseteq 1^\omega = \bot$. The last part follows from (23). To see that the converse does not hold, take $x = 1$. □

Proposition 11. *Let x and y be elements in an $\mathcal{R}$ structure. Then*

$$\epsilon(x^\omega y) = \epsilon(x\epsilon(x^\omega y)) \sqcap \epsilon(y) \tag{60}$$
$$\tau(x^\omega y) = \tau(x\tau(x^\omega y)) \sqcap \tau(y) \tag{61}$$

Proof. The calculation

$$\begin{array}{ll} & \epsilon(x^\omega y) \\ = & \{(11)\} \\ & \epsilon((xx^\omega \sqcap 1)y) \\ = & \{(10)\} \\ & \epsilon((xx^\omega y \sqcap y) \\ = & \{(46)\} \\ & \epsilon(xx^\omega y) \sqcap \epsilon(y) \\ = & \{(36)\} \\ & \epsilon(x\epsilon(x^\omega y)) \sqcap \epsilon(y) \end{array}$$

establishes the first part. The second part is proved in a similar fashion. □

Proposition 12. *Let x be any element and p be any guard in an $\mathcal{R}$ structure. Then the implication*

$$p \sqsubseteq \epsilon(xp) \Rightarrow p \sqsubseteq \epsilon(x^\omega p)$$

does not *hold in general. That is, there is an instantiation of x and p, such that*

$$p \sqsubseteq \epsilon(xp)$$

holds, but

$$p \sqsubseteq \epsilon(x^\omega p)$$

does not.

Proof. Take $x = 1$. Then the antecedent becomes $p \sqsubseteq \epsilon p = p$, which clearly holds for any p. The consequent becomes $p \sqsubseteq \epsilon(1^\omega p) = \epsilon(\bot p) = \epsilon\bot = \epsilon(\tau\bot) = 1$, which clearly does not hold generally for any p. □

The reason that this does not hold stems from the fact that we cannot prove (19)

$$xz \sqsubseteq yx \Rightarrow xz^\omega \sqsubseteq y^\omega x$$

in an $\mathcal{R}$ structure. This is easily seen when trying to prove the rule along the lines of [10]:

$$\begin{array}{ll} & p \sqsubseteq \epsilon(xp) \\ \Leftrightarrow & \{(37)\} \\ & pxp = xp \\ \Leftrightarrow & \{(30)\} \\ & px \sqsubseteq xp \\ \not\Leftarrow & \{(19)\} \\ & px^\omega \sqsubseteq x^\omega p \\ \Leftrightarrow & \{(30)\} \\ & px^\omega p = x^\omega p \\ \Leftrightarrow & \{(37)\} \\ & p \sqsubseteq \epsilon(x^\omega p) \end{array}$$

But as can be seen from this, we do nevertheless have the following result.

Proposition 13. *Let x be any element and p, p_1, and p_2 be any guards in an $\mathcal{R}$ structure. Then*

$$px^\omega \sqsubseteq x^\omega p \quad \Leftrightarrow \quad p \sqsubseteq \epsilon(x^\omega p) \tag{62}$$

$$p_1 x^\omega \sqsubseteq x^\omega p_2 \quad \Leftrightarrow \quad p_1 \sqsubseteq \epsilon(x^\omega p_2) \tag{63}$$

□

On the other hand, we have an induction rule for τ.

Proposition 14. *Let x be any element and p be any guard in an $\mathcal{R}$ structure. Then*

$$\tau(xp^\circ) \sqsubseteq p^\circ \quad \Rightarrow \quad \tau(x^\omega p^\circ) \sqsubseteq p^\circ \tag{64}$$

Proof. The derivation

$$\begin{array}{ll} & \tau(xp^\circ) \sqsubseteq p^\circ \\ \Leftrightarrow & \{(6)\} \\ & xp^\circ \sqsubseteq p^\circ x p^\circ \\ \Leftrightarrow & \{(31)\} \\ & xp^\circ \sqsubseteq p^\circ x \\ \Rightarrow & \{(18)\} \\ & x^\omega p^\circ \sqsubseteq p^\circ x^\omega \\ \Leftrightarrow & \{(31, 6)\} \\ & \tau(x^\omega p^\circ) \sqsubseteq p^\circ \end{array}$$

proves the claim. □

4 The Algebra in Action

Here we apply the algebra for expressing different properties between programs. We also demonstrate the algebra's applicability by using it for proving some properties of action systems.

4.1 Expressing properties between programs

The enabledness and the termination operator can be used to express properties between programs. We list here some examples.

First note that $\overline{\epsilon x}$ is a guard that skips in those states where x is *disabled*. Analogously, $\overline{\tau x}$ denotes an assertion that skips in the states from which x will abort.

Excludes. A program x *excludes* a program y if whenever x is enabled y is not. This can be formalised by saying that x is equal to first executing a guard that checks that y is disabled and then executing x:

$$x = \overline{\epsilon y} x$$

Enables. A program x *enables* y if y is enabled after having executed x:

$$x = x \epsilon y$$

Disables. Similarly as above x *disables* y if

$$x = x\overline{\epsilon y}$$

Using the algebra, we can prove that exclusion is commutative, x excludes y if and only if y excludes x:

$$\begin{array}{ll} & x = \overline{\epsilon y} \\ \Leftrightarrow & \{(37)\} \\ & \overline{\epsilon y} \sqsubseteq \epsilon x \\ \Leftrightarrow & \{\text{guards Boolean algebra}\} \\ & \epsilon y \sqsubseteq \overline{\epsilon x} \\ \Leftrightarrow & \{(37)\} \\ & y = \overline{\epsilon x} y \end{array}$$

We can also express that termination of x requires termination or enabledness of y, respectively:

$$x = \tau y x$$
$$x = \epsilon y x$$

If x requires that y both terminates and is enabled, then by (53) the order of the requirements is all the same:

$$x = \epsilon y \tau y x = \tau y \epsilon y x$$

Program inversion. A program x' inverts a program x when execution of the sequence xx' under the same precondition results in the final state being the same as the initial state [12, 7].

If we assume that the precondition is included as part of the program to be inverted, that is

$$x = p^\circ y$$

for some program y and some assertion p° that specifies the precondition, then this means that

$$\tau x = \tau(p^\circ y) \sqsubseteq p^\circ$$

Program inversion can then be defined as

$$x' \text{ inverts } x \Leftrightarrow \tau x \sqsubseteq xx'$$

Intuitively, this says that the assertion that skips in those states from which x terminates and aborts in all other states, can be replaced by the program xx': if x terminates and x' inverts x then xx' skips, otherwise xx' aborts.

In [17] von Wright uses the explicit definition of τ to derive several rules for program inversion.

4.2 Action systems

Action systems comprise a framework for reasoning about parallel programs [3, 2]. The intuition is that an action system is an iteration of n demonic choices $x_1 \sqcap \cdots \sqcap x_n$, that terminates when none of the actions $x_1, \ldots, x_n$ is longer enabled. It is denoted

$$\mathsf{do}\, x_1 [\!] \ldots [\!] x_n\, \mathsf{od}$$

Using our extended refinement algebra, we can encode an action system as

$$(x_1 \sqcap \ldots \sqcap x_n)^\omega \overline{\epsilon(x_1)} \ldots \overline{\epsilon(x_n)}$$

When the operators are interpreted as in Section 5, this definition gives rise the classical predicate-transformer definition of an action system [5].

We begin by showing that action systems have a leapfrog property.

$$x; \mathsf{do}\; y; x\; \mathsf{od} \sqsubseteq \mathsf{do}\; x; y\; \mathsf{od}; x$$

We will prove this property in the algebra and at the same time expose a methodology for performing derivations.

Action-system leapfrog takes the form

$$x(yx)^\omega \overline{\epsilon(yx)} \sqsubseteq (xy)^\omega \overline{\epsilon(xy)} x \tag{65}$$

in our algebra. We can now embark on proving (65) with basic rules, collecting assumptions as needed. The assumptions are then, in turn, proved.

$$\begin{array}{ll} & x(yx)^\omega \overline{\epsilon(yx)} \\ = & \{(16)\} \\ & (xy)^\omega x \overline{\epsilon(yx)} \\ \sqsubseteq & \{\textbf{collect:}\ \text{if}\ x\overline{\epsilon(yx)} \sqsubseteq \overline{\epsilon(xy)}x\} \\ & (xy)^\omega \overline{\epsilon(xy)} x \end{array}$$

The assumption collected in the second step is then shown to hold by the following derivation.

$$\begin{array}{ll} & x\overline{\epsilon(yx)} \sqsubseteq \overline{\epsilon(xy)}x \\ \Leftarrow & \{(27)\} \\ & \top \sqsubseteq \overline{\epsilon(xy)}x\epsilon(yx) \\ \Leftrightarrow & \{(37,\ 3)\} \\ & \epsilon(xy) \sqsubseteq \epsilon(x\epsilon(yx)) \\ \Leftrightarrow & \{(36)\} \\ & \epsilon(xy) \sqsubseteq \epsilon(xyx) \\ \Leftrightarrow & \{(36)\} \\ & \epsilon(xy) \sqsubseteq \epsilon(xy\epsilon x)) \\ \Leftarrow & \{(23)\ \} \\ & \mathsf{True} \end{array}$$

The same result has been shown in the predicate transformer model [5], but our proof is much cleaner and simpler.

An action system of the form

$$\mathsf{do}\, x_1 \,[\!]\, x_2 \,\mathsf{od}$$

can be refined by

$$\mathsf{do}\, x_2 \,\mathsf{od};\ \mathsf{do}\, x_1;\ \mathsf{do}\, x_2 \,\mathsf{od}\,\mathsf{od}$$

provided that x_1 excludes x_2.

Action-system decomposition is encoded as

$$(x \sqcap y)^\omega \overline{\epsilon x}\ \overline{\epsilon y} \sqsubseteq y^\omega \overline{\epsilon y}(xy^\omega \overline{\epsilon y})^\omega \overline{\epsilon(xy^\omega \overline{\epsilon y})} \tag{66}$$

and the assumption as

$$x = \overline{\epsilon y} x$$

We can then prove (66).

$$\begin{array}{ll} & (x \sqcap y)^\omega \overline{\epsilon x}\ \overline{\epsilon y} \\ = & \{(17)\} \\ & y^\omega (xy^\omega)^\omega \overline{\epsilon x}\ \overline{\epsilon y} \\ = & \{\text{assumption}\} \\ & y^\omega (\overline{\epsilon y} x y^\omega)^\omega \overline{\epsilon x}\ \overline{\epsilon y} \\ = & \{\text{guards Boolean algebra}\} \\ & y^\omega (\overline{\epsilon y} x y^\omega)^\omega \overline{\epsilon y}\ \overline{\epsilon x} \\ = & \{(16)\} \\ & y^\omega \overline{\epsilon y}(x y^\omega \overline{\epsilon y})^\omega\ \overline{\epsilon x} \\ \sqsubseteq & \{\textbf{collect:}\ \text{if}\ \epsilon x \sqsubseteq \epsilon(xy^\omega \overline{\epsilon y})\} \\ & y^\omega \overline{\epsilon y}(x y^\omega \overline{\epsilon y})^\omega\ \overline{\epsilon(xy^\omega \overline{\epsilon y})} \end{array}$$

The collected assumption follows from the fact that

$$\epsilon x = \epsilon(x1) \sqsubseteq \epsilon(x \epsilon y) = \epsilon(xy)$$

for any x and y.

In [17] a similar result has been shown for loops with explicit guards. Here, the enabledness operator ϵ allows action systems with implicit guards.

With predicate-transformer semantics, the opposite direction has also been shown to hold [5]. We have not been able to prove this direction in the algebra. However, the derivation goes in the "right" direction, since the the initial program is refined to a program where the order of execution is clearly expressed.

5 Predicate Transformers as a Model

We show how conjunctive predicate transformers form a model for the algebra described in the previous sections.

5.1 Predicate transformers

A predicate transformer [11] is a function

$$S : \wp(\Sigma) \to \wp(\Sigma)$$

where Σ is any set. Let $p, q \in \wp(\Sigma)$. If a predicate transformer S satisfies

$$p \subseteq q \Rightarrow S.p \subseteq S.q$$

it is *monotone* and if it satisfies

$$S.(p \cap q) = S.p \cap S.q \tag{67}$$

it is *conjunctive*. A predicate transformer S is *universally conjunctive* if S is conjunctive and $S.\Sigma = \Sigma$.

Clearly, any universally conjunctive predicate transformer is conjunctive and it is easily shown that any conjunctive predicate transformer is monotone.

5.2 Interpreting programs

Programs can be modelled by predicate transformers according to a weakest precondition semantics [11]: $S.q$ denotes those sets of states from which the execution of S is bound to terminate in q.

Universally conjunctive predicate transformers cannot model non-termination. Too see this, suppose that S is an always non-terminating program, that is,

$$(\forall q \in \wp(\Sigma) \bullet S.q = \emptyset) \tag{68}$$

Now, if S is universally conjunctive, then

$$S.\Sigma = \Sigma$$

so clearly (68) does not hold.

5.3 Refinement algebra

There are three distinguished predicate transformers

$$\mathsf{abort} = (\lambda q \bullet \emptyset) \tag{69}$$
$$\mathsf{magic} = (\lambda q \bullet \Sigma) \tag{70}$$
$$\mathsf{skip} = (\lambda q \bullet q) \tag{71}$$

and a predicate transformer S_1 *is refined by* S_2, written $S_1 \sqsubseteq S_2$, if

$$(\forall q \in \wp(\Sigma) \bullet S_1.q \subseteq S_2.q)$$

This report deals with three operations on predicate transformers [4, 15] defined by

$$(S;T).q = S.T.q \tag{72}$$
$$(S \sqcap T).q = S.q \cap T.q \tag{73}$$
$$S^\omega = \mu.(\lambda X \bullet S; X \sqcap \mathsf{skip}) \tag{74}$$

where μ denotes the least fixpoint with respect to $\sqsubseteq$.

Let Ctran_Σ be the set of conjunctive predicate transformers over a set Σ. Then it is quite easily verified that

$$(\mathsf{Ctran}_\Sigma, \sqcap, ;, ^\omega, \mathsf{magic}, \mathsf{skip})$$

is an algebra of the type $\mathcal{R}$ described in Section 2. It is also clear that abort models $\bot$, since it can be shown that $\mathsf{skip}^\omega = \mathsf{abort}$ [4].

We can now give semantical justification for not having a right annihilation axiom

$$x\top = \top$$

If we would have right annihilation, then for any predicate transformer S and any $q \in \wp(\Sigma)$

$$S.\Sigma = S.(\mathsf{magic}.q) = \mathsf{magic}.q = \Sigma$$

so our predicate transformer model would be universally conjunctive. As noted above, universally conjunctive predicate transformers cannot model non-termination, that is, they do not facilitate total-correctness reasoning.

5.4 Guards and assertions

Consider the function $[\cdot] : \wp(\Sigma) \to (\wp(\Sigma) \to \wp(\Sigma))$ such that when $p, q \in \wp(\Sigma)$

$$[p].q = \neg p \cup q$$

For every element $p \in \wp(\Sigma)$ there is thus a predicate transformer

$$S_p : \wp(\Sigma) \to \wp(\Sigma),\ \ S_p : q \mapsto \neg p \cup q$$

These predicate transformers are called *guards*. There is an analog that is an *assertion* and it is defined by

$$\{p\}.q = p \cap q$$

Negation $\neg$ is defined on guards and assertions by

$$\neg[p] = [\neg p] \ \text{ and } \ \neg\{p\} = \{\neg p\}$$

where $\neg p$ is set complement.

Let Grd_Σ be the set of all guards over a a set Σ. Then $\mathsf{Grd}_\Sigma \subseteq \mathsf{Ctran}_\Sigma$, since if $[p]$ is any guard, it holds that

$$[p].(q_1 \cap q_2) = \neg p \cup (q_1 \cap q_2) = (\neg p \cup q_1) \cap (\neg p \cup q_2) = [p].q_1 \cap [p].q_2$$

for any $q_1, q_2 \in \wp(\Sigma)$.

It is also easily established that

$$(\mathsf{Grd}_\Sigma, ;, \sqcap, \neg, \mathsf{magic}, \mathsf{skip})$$

is a Boolean algebra, where ; is meet, $\sqcap$ is join, and $\neg$ is complement. For example, if $g \in \mathsf{Grd}_\Sigma$, then $g \sqcap \neg g = \mathsf{skip}$ as the following shows: Let $[p]$ be any guard and $q \in \wp(\Sigma)$. Then

$$([p] \sqcap \neg[p]).q = ([p] \sqcap [\neg p]).q = (\neg p \cup q) \cap (\neg(\neg p) \cup q) = q = \mathsf{skip}.q.$$

The rest of the axioms for Boolean algebra are verified similarly.

The guards and assertions in the predicate-transformer sense thus correspond to the guards and assertions of the algebraic structure in Section 2.2.

5.5 Enabledness and Termination

In the predicate-transformer model, we interpret

ϵS as $[\neg S.\emptyset]$

and

τS as $\{S.\Sigma\}$

where S is a conjunctive predicate transformer. It can easily be established that this interpretation is sound for the axioms of ϵ, (34 – 36) and of τ, (40 – 42). For example, the axioms (34) and (41) are verified by

$$\begin{array}{ll} & [\neg S.\emptyset]; S \sqsubseteq S \\ \Leftrightarrow & \{\text{definitions}\} \\ & (\forall q \in \wp(\Sigma) \bullet [\neg S.\emptyset].(S.q) \subseteq S.q) \\ \Leftrightarrow & \{\text{definitions}\} \\ & (\forall q \in \wp(\Sigma) \bullet S.\emptyset \cup S.q \subseteq S.q) \\ \Leftrightarrow & \{\text{monotonicity of } S\} \\ & (\forall q \in \wp(\Sigma) \bullet \mathsf{True}) \\ \Leftrightarrow & \{\text{logic}\} \\ & \mathsf{True} \end{array}$$

and

$$\begin{array}{ll} & (\forall p \in \wp(\Sigma) \bullet \{(\{p\}; S).\Sigma\} \sqsubseteq \{p\}) \\ \Leftrightarrow & \{\text{definitions}\} \\ & (\forall p, q \in \wp(\Sigma) \bullet \{\{p\}.(S.\Sigma)\}.q \subseteq \{p\}.q) \\ \Leftrightarrow & \{\text{definitions}\} \\ & (\forall p, q \in \wp(\Sigma) \bullet p \cap S.\Sigma \cap q \subseteq p \cap q) \\ \Leftrightarrow & \{\text{set theory}\} \\ & (\forall p, q \in \wp(\Sigma) \bullet \mathsf{True}) \\ \Leftrightarrow & \{\text{logic}\} \\ & \mathsf{True} \end{array}$$

respectively. The other axioms can be verified similarly.

6 Concluding Remarks

We have introduced a refinement algebra restricted to strong iteration and extended it with the enabledness operator and the termination operator. We have shown that the axiomatisation is sound with respect to a predicate-transformer model and applied the algebra to reasoning about action systems.

The reduced refinement algebra and its extensions deserve further investigation. Since total correctness is what we are interested in, the restriction of the framework to merely the strong iteration operator is motivated. However, some propositions concerning $^\omega$ that were proved in related algebras, e.g.

$$xy \sqsubseteq yx \Rightarrow (x \sqcap y)^\omega = x^\omega y^\omega$$

rely on axioms for the Kleene * operator in their proofs. To what extent these types of propositions can be proved in the reduced algebra, which lacks the * operator, is under investigation.

Applying the new operators, ϵ and τ, to larger problems is yet to be done. Two important metaresults, completeness and decidability, are also pending questions that we hope to return to elsewhere.

Acknowledgements. Thanks are due to Orieta Celiku and Viorel Preoteasa for elucidating discussions, and to Herman Norrgrann and anonymous referees for comments on an earlier draft of this report.

References

[1] R.J. Back. *Correctness Preserving Program Refinements: Proof Theory and Applications*, volume 131 of *Mathematical Centre Tracts*. Mathematical Centre, Amsterdam, 1980.

[2] R.J. Back. Refining atomicity in parallel algorithms. In *PARLE Conference on Parallel Architectures and Languages Europe*, Eindhoven, the Netherlands, June 1989. Springer-Verlag, 1989.

[3] R.J. Back and K. Sere. Stepwise refinement of action systems. *Structured Programming*, 12:17–30, 1991.

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

[5] R.J. Back and J. von Wright. Reasoning algebraically about loops. *Acta Informatica*, 36:295–334, 1999.

[6] R. Berghammer, B. Möller, and G. Struth, editors. *Relational and Kleene-Algebraic methods in Computer Science*, LNCS 3051. Springer, 2004.

[7] W. Chen and J.T. Udding. Program inversion: more than fun! Report CS 8903, Dept. of Mathematics and Computer Science, University of Groningen, 1988.

[8] E. Cohen. Hypotheses in Kleene Algebra. Unpublished manuscript, Telcordia, 1994.

[9] E. Cohen. Separation and reduction. In *Mathematics of Program Construction*, volume 1837 of *Lecture Notes in Computer Science*, Portugal, July 2000. Springer-Verlag.

[10] J. Desharnais, B. Möller and G. Struth. Kleene algebra with domain. Technical Report 2003-7, Universität Augsburg, Institut für Informatik, June 2003.

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

[12] D. Gries. *The Science of Programming*. Springer-Verlag, New York, 1981.

[13] D. Kozen A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Inf. Comput.* 110(2): 366-390,1994.

[14] D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, 1999.

[15] C.C. Morgan. *Programming from Specifications* (2nd edition). Prentice-Hall, 1994.

[16] K. Solin and J. von Wright. Demonic refinement algebra with domain. In K. Sere and M. Waldén, editors, Proceedings of the 15th Nordic Workshop on Programming Theory - NWPT'03, page 43, Åbo Akademi, Reports on Computer Science and Mathematics, Ser. B, No. 34, October 2003. (Extended Abstract.)

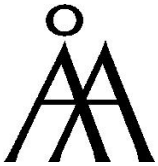
[17] J. von Wright. From Kleene algebra to refinement algebra. In *Mathematics of Program Construction*, volume 2386 of *Lecture Notes in Computer Science*, Germany, July 2002. Springer-Verlag.

[18] J. von Wright. Towards a refinement algebra. *Science of Computer Programming*, 51:23–45, 2003.

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics



Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 952-12-1493-7
ISSN 1239-1891