SEMANTIC MODELS FOR TOTAL CORRECTNESS AND FAIRNESS†

Michael G. Main and David L. Black

CU-CS-417-88 — March 1989

† This research has been supported in part by National Science Foundation grant CCR-8701946. David Black has been supported by Storage Technology Corporation, Louisville, CO 80028.

SEMANTIC MODELS FOR TOTAL CORRECTNESS AND FAIRNESS†

Michael G. Main and David L. Black
Department of Computer Science
University of Colorado
Boulder, CO 80309 USA
Email: main@ boulder.colorado.edu

ABSTRACT

Assertional s-rings are introduced to provide an algebraic setting in which the finite and infinite behavior of nondeterministic programs can be expressed and reasoned about. This includes expressing the fair infinite behavior of nondeterministic iterative programs, and reasoning about termination under various fairness assumptions. We also address the question of when the reasoning techniques are semantically complete.

† This research has been supported in part by National Science Foundation grant CCR-8701946. David Black has been supported by Storage Technology Corporation, Louisville, CO 80028.

1. Background and Motivation

The purpose of this paper is to provide an algebraic setting for reasoning about the control structures of iterative nondeterministic programs. The algebra supports reasoning about nontermination (*i.e., total* correctness) and about *fair* nondeterministic constructions. To quote Kuich and Salomaa: "The tools from linear algebra make the proofs computational in nature and, consequently, more satisfactory from the mathematical point of view than the customary proofs." [8]

This algebraic approach to the semantics of programs also underlies dynamic logic (*e.g.* [6,7]), and our particular approach owes much to the assertional categories of Manes [12,14,13] and the use of the Boolean algebra of guards within a zerosum-free semiring [15]. The major addition of our work to this earlier research is the algebraic treatment of nontermination and fairness.

Our starting point is a familiar idea: Nondeterministic programs denote elements in an algebraic structure, which is almost a semiring. Each of the usual syntactic constructions on programs (such as IF–THEN–ELSE or WHILE–DO) has a corresponding semantic operation that is defined in the algebra. Boolean expressions, used as pre-conditions and post-conditions about programs, are elements in a Boolean algebra of "guards", which exists as a subset of the algebra. Because of this, the usual program assertions (involving a pre-condition, a program, and a post-condition) can be proved algebraically, using only the laws of the algebra. Moreover, we show algebraically that the usual Hoare rules for proving total correctness assertions are valid — and if certain conditions (given in Section 6) are valid, then the entire Hoare calculus (including rules for termination of fair nondeterministic loops) is "semantically complete".

This use of the Boolean algebra of guards in a semiring was proposed by Manes and Benson [15] and had some of its motivation from [9,10]. It has been further developed by Manes and Arbib [12,13,14], who use partially additive semirings which are morphism sets in a certain kind of category, called an *assertional category*. They develop a calculus for proving partial correctness assertions about programs, and this calculus was recently shown by Bloom to be sound and complete in the setting of iterative algebraic theories [1]. The semirings used by Manes and Arbib possess a unary operation * which meets the axiom

$s^* = ss^* + 1$. The $*$ is used to provide semantic operations for iterative constructions by solving iteration equations – in the spirit of Elgot [3,4] and dynamic logic. In general, these semantic operations ignore infinite iterative behavior and are not appropriate for expressing total correctness assertions (*i.e.*, assertions where termination is guaranteed).†

With this in mind, we introduce another unary operation $^\infty$, where the axiom $s^\infty = ss^\infty$ is met. Intuitively, s^∞ is the result of executing s infinitely often (in much the same way that s^* is the result of executing s finitely often). The meaning of an iterative program is still a solution to the usual iteration equation, but the solution is constructed using both the $*$ and the $^\infty$ operations. Whereas the original approach (using only $*$) provided a calculus for partial correctness assertions, the new approach (with $*$ and $^\infty$) yields a calculus for total correctness assertions.

In addition, the two unary operations can be combined in various ways that express different kinds of fair behavior within a DO–OD loop (as presented by Francez [5]).

2. S-rings and a Concrete Example

2.1 S-rings

In the previous section, we said that nondeterministic programs will denote elements in a certain kind of algebra. This algebra is *almost* a semiring: The "almost" occurs because our primary example violates the semiring law $s0 = 0$. But apart from this violation, the structure is a semiring. We refer to this algebraic structure as an *s-ring* (think of a semiring with something missing). Formally, an s-ring is a set S with two binary operations ($+$ and $\cdot$) and two distinct constants (0 and 1) such that:

1. $(S, +, 0)$ is a commutative monoid.

2. $(S, \cdot, 1)$ is a monoid.

3. Multiplication ($\cdot$) distributes over addition ($+$) on both sides.

4. For all $s \in S$: $0s = 0$.

† Total correctness assertions have been studied by Manes and Arbib in one assertional category (the category of partial functions — suitable for deterministic programs), but it's unclear whether the technique extends to other assertional categories.

A *zerosum-free s-ring* also has the axiom:

5. For all $s, t \in S$: $s+t=0$ iff $s=0=t$.

We will use the typical semiring notation with s-rings, writing st instead of $s \cdot t$, and assuming that multiplication has precedence over addition in expressions like $s+tu$. If s is an element of an s-ring and i is a natural number, then s^i denotes the multiplication of i copies of s (with s^0 defined as 1).

2.2 The S-ring of Strict Relations

A simple example can clarify the representation of nondeterministic programs by elements of an s-ring. For this example, consider a setting where each program computes in some fixed "state space" D. Thus, each execution of a program starts in some state $d \in D$ and, if it terminates, will finish in some state $e \in D$. There may also be nonterminating executions, which start in some state, but fail to terminate.

A nondeterministic program denotes a binary relation on $D_\perp$, where $D_\perp$ is the set D plus a new element $\perp$ which represents the "result" of a program that is in an unending loop. Intuitively, the relation denoted by a program is the state-transition for the program: if s is a relation denoting a program, and $(d, e) \in s$, then the corresponding program is capable of mapping an initial state d to a final state e. If $e = \perp$ then the program has a nonterminating execution starting in state d. We also require each program's relation to map $\perp$ to $\perp$ and nowhere else ($(\perp, e) \in s$ iff $e = \perp$). This means that if the input to s came from a nonterminating program, then s cannot fix this. A relation with this behavior for $\perp$ is called *strict.*

Now we focus on the algebraic structure of the set of all strict binary relations on $D_\perp$. We call this set of relations A and note that it forms an s-ring. Addition in A is union of relations, so $(d, e) \in s+t$ iff $(d, e) \in s$ or $(d, e) \in t$. Multiplication is composition of relations, so $(d, e) \in st$ iff there exists some $c \in D_\perp$ with $(d, c) \in s$ and $(c, e) \in t$. The multiplicative identity (1) is the identity relation ($(d, d) | \ d \in D_\perp$), and the zero (0) is the smallest strict relation $\{(\perp, \perp)\}$. These s-ring operations correspond to operations on programs. If s and t represent programs, then st is the composite program ("first do s, then do t"). The union relation $s+t$ is a program which can behave like either s or t ("a nondeterministic choice between s and t").

Certain relations in A do not correspond to programs, but they have another important interpretation. These are the relations which are subrelations of 1. Such a relation, called a *guard,* has two choices for each $d \in D$: either d is related to d (and nothing else) or d is related to nothing. A predicate p on D can be interpreted as the guard $\{(d,d) |\ p(d) \text{ or } d = \bot\}$. Note that $(\bot,\bot)$ is included since we are dealing with strict relations. For a guard p and a state d, we say that d *satisfies* p provided that $(d,d) \in p$. Each guard p has a complement guard $\bar{p}$, such that $p + \bar{p} = 1$ and $p\bar{p} = 0 = \bar{p}p$. A state from D satisfies exactly one of p and $\bar{p}$.

2.3 Conditional Programs and Iteration

Let b be a guard and, let s and t be relations. The relation for a conditional program IF b THEN s ELSE t is expressed in the s-ring of relations as $bs + \bar{b}t$.

To express the relation for an iterative program WHILE b DO s, we use two unary operations on relations. For any $u \in A$, define u^* to be the transitive and reflexive closure of u. And define u^∞ to be $\{(d,\bot) |\ d$ lies on an infinite u-path$\}$. An *infinite u-path* is a countably infinite sequence $d_0, d_1, \cdots$ such that for all $i \geq 0$, $(d_i, d_{i+1}) \in u$. With these operations, the meaning of WHILE b DO s is $(bs)^\infty + (bs)^*\bar{b}$. The left term provides the nonterminating behavior of the loop, and the right term provides the terminating behavior.

The $*$ and $^\infty$ can also be used to express various kinds of *fair* iteration. For example, consider the program

$$\text{WHILE } b \text{ DO } (s_1 \,\Box\, s_2),$$

where $\Box$ indicates a nondeterministic choice. In the usual semantics, this WHILE-loop may have infinite computations which do not execute each of the choices (s_1 and s_2) infinitely often. If we let $t = bs_1 + bs_2$, then the usual meaning of the loop is $t^\infty + t^*\bar{b}$.

But this kind of program has also been studied using various *fairness assumptions* (see [5]). The simplest fairness assumption is to forbid infinite computations that don't choose both branches infinitely often. In our algebra, the meaning of the loop with this fairness assumption is:

$$(t^*bs_1t^*bs_2)^\infty + t^*\bar{b},$$

where t is defined as in the previous paragraph. Intuitively, the second term represents the finite behaviors and the first term represents fair infinite behaviors. Later we will show how to express other kinds of fairness.

In a moment, we will introduce *assertional s-rings*, which are s-rings with two additional operations $*$ and $^\infty$. But first we explain how assertions about programs are given in the s-ring of strict relations.

2.4 Assertion Semantics and $ps\overline{q}=0$

In assertion semantics, reasoning about programs occurs in terms of a pre-condition p and a post-condition q. These conditions are guards (*i.e.*, predicates on states), and in order for a program s to be totally correct with respect to the conditions, the following must hold: whenever the program s starts in a state which satisfies p, then it will end in a state which satisfies q. The notation $[p]\, s\, [q]$ is an assertion that s is totally correct for pre-condition p and post-condition q.

In the s-ring of strict relations, correctness of a program s with respect to pre-condition p and post-condition q is simply $ps\overline{q}=0$. The intuitive translation of this equality says that certain mappings are forbidden by s: Specifically, it is not possible to start in a state $d \in D$ such that $p(d)$ holds, $(d,e) \in s$, and $q(e)$ fails to hold. Also, it is not possible to start in a state $d \in D$ such that $p(d)$ holds and $(d,\bot) \in s$, since this would imply $(d,\bot) \in ps\overline{q} \neq 0$. Thus, $ps\overline{q}=0$ expresses *total* correctness: s cannot fail to terminate when it's started in a state that satisfies p.

Manes [12,14,13] used the equation $ps\overline{q}=0$ in the more general setting of *assertional categories*. However, in these papers, the equation expressed partial correctness of non-deterministic programs (termination was not guaranteed).

3. Program Denotations in Assertional S-rings

The previous section illustrated the idea that nondeterministic programs denote elements in an s-ring of relations, and Boolean expressions denote guards in this s-ring. We now generalize this idea as follows: nondeterministic programs denote elements in a certain kind of s-ring called an *assertional s-ring*, defined in this section. Boolean expressions denote a certain kind of element, which we will call a guard, and correctness assertions about programs are proved by showing algebraic identities of the form $ps\overline{q}=0$. We begin by defining the meaning of a guard in an arbitrary s-ring, and observing some of the properties of guards.

3.1. The Boolean Algebra of Guards

In the s-ring of relations, we defined special relations called guards, which correspond to "state predicates". A similar notion is available in any s-ring, as defined here:

Definition: Let S be an s-ring. A *guard* of S is an element p such that for some $\overline{p} \in S$: $p+\overline{p}=1$ and $p\overline{p}=0=\overline{p}p$. The element $\overline{p}$ is called the complement of p. The set of all guards of S is denoted $GUARD_S$.

Manes and Benson [15] showed that the set of guards in any zerosum-free semiring forms a Boolean algebra, with the partial order $s \leq t$ iff there exists u with $s+u=t$. This order, called the *sum-ordering* is not always a partial order on a semiring (anti-symmetry can fail). But for a zerosum-free semiring, it is a Boolean-algebra order on the guards and 0 is the minimum element in the semiring.

Manes and Benson's results also hold for zerosum-free s-rings: the guards form a Boolean algebra under the sum-order, and 0 is the minimal element in the s-ring. In the Boolean algebra of guards, the minimum guard is 0, the maximum guard is 1, the "meet" of p and q is pq and the "join" of p and q is $p+\overline{p}q$. If W is a set of guards and the join (or least upper-bound) of W exists, then we denote this by $\bigvee W$ or sometimes $\bigvee_{p \in W} p$.

3.2. Assertional S-rings

Now we can define assertional s-rings, which we will use as semantic models of nondeterministic programming languages.

Definition: An *assertional s-ring* is a zerosum-free s-ring S, with two unary operations ($*$ and $^\infty$) which meet these axioms: For all $s, t \in S$ and $W \subseteq GUARD_S$:

1. **Closure Axiom:** $s^* = ss^* + 1$.
2. **Iteration Axiom:** $(\forall\, i.\ rs^i t = 0)$ implies $rs^*t = 0$.
3. **Infinity Axiom:** $s^\infty = ss^\infty = s^\infty t$.
4. **Continuity Axiom:** When $\vee W$ exists then $(\forall\, p \in W.ps = 0)$ implies $(\vee W)s = 0$.

We write the unary operations using postfix notation, and these have higher precedence than the s-ring operations in expressions. For example ss^∞ is $s\,(s^\infty)$. The motivation for the two new operations and their axioms comes from the corresponding operations in the s-ring of strict relations over $D_\perp$ (see Section 2.3). The intuition behind the axiom $ss^\infty = s^\infty t$ is that s^∞ represents infinite behaviors, therefore anything which follows s^∞ will never be reached.

3.3. Program Constructs and Correctness Assertions

Assertional s-rings provide semantic models for nondeterministic programming languages. In general, nondeterministic programs denote elements in an assertional s-ring, and Boolean expressions denote guards in the same s-ring.

Within any assertional s-ring, the usual program constructs, such as IF–THEN–ELSE and WHILE–DO can be represented algebraically, as discussed in Section 2. Also, the identity $ps\bar{q} = 0$ is important when p and q are guards and s denotes a program, since this corresponds to the correctness assertion $[p]\,s\,[q]$. Because of this, we will use the $[p]\,s\,[q]$ notation to express the equality $ps\bar{q} = 0$. This can be either total or partial correctness, depending on the particular s-ring.

The remainder of the paper gives algebraic demonstrations of rules for showing correctness assertions for various different forms of the program s. When s has the form of one of the usual program constructions (such as $bs_1 + \bar{b}s_2$ for the program

IF b THEN s_1 ELSE s_2), then the demonstrated rules will be the usual Hoare rules for total correctness assertions. This includes rules for fair iterative constructions.

In effect: every assertional s-ring comes with the "semantic operations for programming" and the "Hoare calculus of programs" as standard equipment.

4. Some Basic Rules

This section provides basic rules for proving $[p]s\,[q]$, where s has one of the forms s_1s_2, s_1+s_2, $(s_1)^*$, or $(s_1)^\infty$. We also present rules that correspond to the Consequence Rule and the Disjunction Rule of the Hoare calculus. Some of the proofs are based on results in [14, Section 3.3] — although the proofs need changing to avoid using $s0=0$.

Throughout this section and the rest of the paper, we will assume a fixed assertional s-ring, using the letters s, t (sometimes with subscripts) for arbitrary elements. We'll use b, p, q, r (sometimes with subscripts) for arbitrary guards.

4.1. Composition

This section gives a rule for proving a correctness assertion of the form $[p]st\,[q]$.

Composition Rule: $[p]s\,[q]$ and $[q]t\,[r]$ imply $[p]st\,[r]$.

Proof: We are given $ps\bar{q}=0=qt\bar{r}$, and we must show $pst\bar{r}=0$. This is done here:

$$pst\bar{r}=ps(q+\bar{q})t\bar{r}=psqt\bar{r}+ps\bar{q}t\bar{r}=ps0+0t\bar{r}=ps0+0=ps0+ps\bar{q}=ps(0+\bar{q})=ps\bar{q}=0$$

Note the bit of extra work because we cannot immediately conclude that $ps0=0$. □

In general, this rule is not semantically complete – meaning that even when $[p]st\,[r]$ is true, there might not be any guard q such that $[p]s\,[q]$ and $[q]t\,[r]$. We'll address this more in Section 6.

4.2. Addition

This section gives a rule for proving a correctness assertion of the form $[p]\,s{+}t\,[q]$. This rule is semantically complete, meaning that it is sufficient for any assertion of this form.

Addition Rule: $[p]\ s+t\ [q]$ iff $[p]\,s\,[q]$ and $[p]\,t\,[q]$.

Proof:

$$[p]\ s+t\ [q]$$

$$\Leftrightarrow\quad ps\bar{q}+pt\bar{q}=0$$

$$\Leftrightarrow\quad ps\bar{q}=0=pt\bar{q}$$

$$\Leftrightarrow\quad [p]\,s\,[q]\ \text{ and }\ [p]\,t\,[q]$$

The second equivalence is valid from the zerosum-free law. □

4.3. The Consequence Rule and the Disjunction Rule

In the usual Hoare calculus, a valid program assertion remains valid when the pre-condition is weakened or the post-condition is strengthened. This section gives the corresponding rule for assertional s-rings. In this section, we use Boolean algebra terminology on guards, so for example "q_1 implies q_2" means $q_1q_2=q_1$ (which is also equivalent to the program assertion $[q_1]\,1\,[q_2]$).

Consequence Rule: Suppose p_2 implies p_1, and q_1 implies q_2, and $[p_1]\,s\,[q_1]$. Then $[p_2]\,s\,[q_2]$.

Proof: The two assumptions about implications are equivalent to the program assertions $[p_2]\,1\,[p_2]$ and $[q_1]\,1\,[q_2]$. From two applications of the composition rule, $[p_2]\,1\,[p_1]$ and $[p_1]\,s\,[q_1]$ and $[q_1]\,1\,[q_2]$ imply $[p_2]\,1s1\,[q_2]$. This is just the result we need, since $1s1=s$. □

Another rule in the Hoare calculus is the Disjunction Rule, which follows immediately from our Continuity Axiom:

Disjunction Rule: Let W be a set of guards such that $\vee W$ exists, and suppose that for all $p \in W$, $[p]\,s\,[q]$. Then $[\vee W]\,s\,[q]$. □

4.4. The * Operation

Here is the rule for $*$, which uses the Consequence Rule:

Iteration Rule: Suppose there exists a guard p such that q implies p, and $[p]\,s\,[p]$, and p implies r. Then $[q]\,s^*\,[r]$.

Proof: An induction on i shows that $[p]\,s^i\,[p]$ is valid for all i, and by the Iteration Axiom this implies $[p]\,s^*\,[p]$. Using the Consequence Rule (together with "q implies p" and "p implies r"), yields the needed result: $[q]\,s^*\,[r]$. □

4.5. The $^\infty$ Operation

The correctness rule for $[p]\,s^\infty[q]$ is notable because it is independent of the post-condition q. This matches our intuition that s^∞ consists of the behaviors resulting from executing s infinitely often – so that the post-condition is never reached! Thus, $[p]\,s^\infty[q]$ really means that the pre-condition p is sufficient to guarantee there will be no "infinite paths" in s^∞. Thus, instead of proving assertions $[p]\,s^\infty[q]$ with arbitrary post-conditions, we'll generally only prove them with the post-condition 1 (true). Here's the formal justification:

Theorem. $[p]\,s^\infty[q]$ iff $[p]\,s^\infty[1]$.

Proof: The equivalence follows from this derivation:

$$[p]\,s^\infty[q] \;\Leftrightarrow\; ps^\infty\overline{q}=0 \;\Leftrightarrow\; ps^\infty=0 \;\Leftrightarrow\; ps^\infty 0=0 \;\Leftrightarrow\; [p]\,s^\infty[1]$$

The second and third equivalences follow from $s^\infty t = s^\infty$, which is part of the Infinity Axiom. □

Notation. Because the post-condition doesn't matter, we will sometimes write TERMINATE(p, t) as notation for $[p]\, t^\infty\, [q]$. □

Now we'll give the general rule for proving an assertion TERMINATE(p, t):

Infinity Rule: Let $(I, <)$ be a well-founded set and for each $i \in I$ let p_i be a guard such that:

$$[p_i]\, t\, [\bigvee_{j<i} p_j].$$

Also suppose that p implies $\bigvee_{i \in I} p_i$. Then TERMINATE(p, t).

Proof: To begin, we show that for any $i \in I$, $[p_i]\, t^\infty [1]$. The proof is by well-founded induction on I. For this induction, let $i \in I$, and suppose (for the induction hypothesis) that whenever $j<i$ then $[p_j]\, t^\infty [1]$. From the Disjunction Rule, this implies $[\bigvee_{j<i} p_j]\, t^\infty [1]$. This is combined with the given assertion $[p_i]\, t\, [\bigvee_{j<i} p_j]$ to yield $[p_i]\, tt^\infty [1]$. But $tt^\infty = t^\infty$ (by the Infinity Axiom), so this last assertion is just $[p_i]\, t^\infty [1]$, and this completes the induction.

Finally, since we have shown $[p_i]\, t^\infty [1]$ for all $i \in I$, the Disjunction Rule implies $[\bigvee_{i \in I} p_i]\, t^\infty [1]$, or equivalently TERMINATE$(\bigvee_{i \in I} p_i,\ t)$. Since p implies $\bigvee_{i \in I} p_i$, the needed result then follows from the Consequence Rule. □

A combination of the Composition Rule and the ∞ Rule will be useful later on:

∞-Composition Rule: Suppose $t_1, t_2, \ldots, t_n$ are any elements in the s-ring. Let $(I, <)$ be a well-founded set and for each $i \in I$ let p_i be a guard such that:

$$\forall m\ (1 \le m \le n).\, [p_i]\, t_m\, [p_i].$$

$$\exists m\ (1 \le m \le n).\, [p_i]\, t_m\, [\bigvee_{j<i} p_j].$$

Also suppose that p implies $\bigvee_{i \in I} p_i$. Then TERMINATE$(p,\ t_1 t_2 \cdots t_n)$.

Proof: Let $t = t_1 t_2 \cdots t_n$. As in the previous proof, we can show (by well-founded induction on I) that for any $i \in I$, $[p_i]\, t^\infty [1]$. From this and the Disjunction Rule it follows that $[\bigvee_{i \in I} p_i]\, t^\infty [1]$. Since p implies $\bigvee_{i \in I} p_i$, the needed result then follows from the Consequence Rule. □

5. Hoare Calculus in Assertional S-rings

The previous section gave rules for proving correctness assertions about programs constructed with the multiplication, addition, * and $^\infty$ operations in any assertional s-ring. In this section, these four operations are used to define compound operations in an assertional s-ring, corresponding to typical operations on programs (such as IF–THEN–ELSE and WHILE–DO). The results of Section 4 are then used to prove that the usual rules of the Hoare calculus are valid in any s-ring. This includes rules about certain fair iteration constructions.

5.1. IF–THEN–ELSE

Suppose that b is a guard denoting a Boolean expression B, and s,t are elements denoting some syntactic programs S and T. Then the syntactic program IF B THEN S ELSE T is denoted by the element $bs+\bar{b}t$.

Notation. For any guard b and any elements s,t, we use the mnemonic notation IF b THEN s ELSE t for $bs+\bar{b}t$. □

The usual Hoare calculus rule for IF–THEN–ELSE provides a necessary and sufficient condition for proving assertions about these programs:

IF **Rule:** $[p]$ IF b THEN s ELSE t $[q]$ iff $[pb]\,s\,[q]$ and $[p\bar{b}]\,t\,[q]$.

Proof:

$[p]$ IF b THEN s ELSE t $[q]$ $\Leftrightarrow$ $[p]\;bs+\bar{b}t\;[q]$

$\Leftrightarrow$ $[p]\,bs\,[q]$ and $[p]\,\bar{b}t\,[q]$

$\Leftrightarrow$ $pbs\bar{q}=0=p\bar{b}t\bar{q}$

$\Leftrightarrow$ $[pb]\,s\,[q]$ and $[p\bar{b}]\,t\,[q]$

Note that the second equivalence follows from the Addition Rule. □

5.2. Iteration

Suppose that b is a guard denoting a Boolean expression B, and s is an element denoting some syntactic program S. Then the syntactic program WHILE B DO S is denoted by the element $(bs)^{\infty}+(bs)^{*}\bar{b}$. Intuitively, $(bs)^{\infty}$ gives the infinite behavior of the WHILE-loop, and $(bs)^{*}\bar{b}$ gives the finite behaviors.

Notation. For any guard b and any element s, we use the mnemonic notation WHILE b DO s for $(bs)^{\infty}+(bs)^{*}\bar{b}$. □

The usual Hoare calculus rule for WHILE–DO is valid for proving assertions about these programs:

WHILE **Rule:** Suppose the following two conditions hold (where "**or**" is join in the Boolean algebra of guards):

PARTIAL CORRECTNESS: There exists a guard p (the "loop invariant") such that q implies p, and $[pb]\,s\,[p]$, and p implies (b **or** r).

TERMINATION: There exists a well-founded set $(I,<)$ and a guard p_i for each $i \in I$ such that:

(i) For every $i \in I$: $[p_i b]\,s\,[\bigvee_{j<i} p_j]$, and

(ii) q implies $\bigvee_{i \in I} p_i$.

Then $[q]$ WHILE b DO s $[r]$.

Proof: Assume the two conditions hold. We must prove $[q]$ WHILE b DO s $[r]$, or equivalently both

(1) TERMINATE(q, bs), and

(2) $[q]\,(bs)^{*}\bar{b}\,[r]$.

Assertion (1) will follow from the TERMINATION condition, and assertion (2) from PARTIAL CORRECTNESS.

For (1), we can rewrite part (i) of TERMINATION as this:

$$\forall\, i \in I.\,[p_i]\,bs\,[\bigvee_{j<i} p_j]\,.$$

This rewriting makes TERMINATION equivalent to the hypothesis of the Infinity Rule (taking

bs as t in that rule). Therefore, the Infinity Rule implies TERMINATE($\bigvee_{i \in I} p_i$, bs). This and "q implies $\bigvee_{i \in I} p_i$", (together with the Consequence Rule) implies (1).

For (2), we can rewrite PARTIAL CORRECTNESS as:

$$q \text{ implies } p \text{, and } [p]\, bs\, [p] \text{, and } p \text{ implies } (b \textbf{ or } r).$$

This is now in the form to apply the Iteration Rule, which implies $[q]\,(bs)^*\,[b \textbf{ or } r]$, or equivalently $0 = q\,(bs)^*\,\overline{(b \textbf{ or } r)} = q\,(bs)^*\,\bar{b}\,\bar{r}$. Since this last expression is 0, we have $[q]\,(bs)^*\bar{b}\,[r]$, as required. □

5.3. Fair Iteration

In Section 2.3 we discussed the program WHILE b DO ($s_1 \Box s_2$). Using a simple fairness assumption which forbids an infinite computation from eventually ignoring one of the directions, this program has the algebraic meaning:

$$(t^*bs_1t^*bs_2)^\infty + t^*\bar{b}$$

where t is defined as $bs_1 + bs_2$.

In proving correctness assertions about such a program, the second term ($t^*\bar{b}$, which represents terminating behavior) can be handled using the PARTIAL CORRECTNESS condition of the WHILE-Rule. The first term – representing the fair infinite behavior – can be handled with the following rule, which is motivated by the "Unconditional Fair Termination" Rule of [5].

Fair-WHILE Termination Rule: Let $(I, <)$ be a well-founded set and for each $i \in I$ let p_i be a guard such that:

(i) For all k $(1 \le k \le 2)$: $[p_i b]\, s_k\, [p_i]$, and

(ii) For some k $(1 \le k \le 2)$: $[p_i b]\, s_k\, [\bigvee_{j<i} p_j]$.

Also suppose that p implies $\bigvee_{i \in I} p_i$.

Then TERMINATE(p, $t^*bs_1t^*bs_2$), where $t = (bs_1 + bs_2)$.

Proof: We will show that the hypothesis of the ∞-Composition Rule (Section 4.5) is met, by breaking $(t^* bs_1 t^* bs_2)$ into the four pieces $t_1 = t^*$, $t_2 = bs_1$, $t_3 = t^*$, and $t_4 = bs_2$.

For the first part of the ∞-Composition hypothesis, we must show that for all $i \in I$:

$$\forall m\ (1 \le m \le 4).\, [p_i]\, t_m\, [p_i].$$

For $m=2$ and $m=4$, this is just the statement (i). For $m=1$ and $m=3$, we have $t_m = t^*$, so we must show $[p_i]\, t^*\, [p_i]$ (for all $i \in I$). Toward this goal, let i be some element of I and note that by the Iteration Axiom it is sufficient to show that $[p_i]\, t^h\, [p_i]$ for all natural numbers h. We prove this by induction on h: For the base case ($[p_i]\, t^0\, [p_i]$):

$$p_i t^0 \overline{p}_i = p_i 1 \overline{p}_i = p_i \overline{p}_i = 0.$$

For the induction step, assume $[p_i]\, t^h\, [p_i]$ for some h. Also note that $[p_i]\, t\, [p_i]$ follows from the Addition Rule since:

$$t = bs_1 + bs_2 \text{ (by definition),}$$
$$[p_i]\, bs_1\, [p_i] \text{ (by (i)), and}$$
$$[p_i]\, bs_2\, [p_i] \text{ (by (i)).}$$

Combining $[p_i]\, t^h\, [p_i]$ and $[p_i]\, t\, [p_i]$ with the Composition Rule yields $[p_i]\, t^{h+1}\, [p_i]$, which completes the induction.

For the second part of the ∞-Composition hypothesis, we must show that for all $i \in I$:

$$\exists m\ (1 \le m \le 4).\, [p_i]\, t_m\, [\bigvee_{j<i} p_j].$$

But for any $i \in I$, this follows immediately from (ii) — in fact we know that it must be valid for either $m=2$ or $m=4$. □

5.4. Weak and Strong Fair Iteration

Guarded iteration, introduced by Dijksta [2], has the form:

```
DO
    $b_1 \rightarrow s_1$
    □
    $b_2 \rightarrow s_2$
OD
```

This is a loop where each iteration will execute one of the s_i where the corresponding guard

(b_i) is true. When both the guards become false, the loop terminates. In general, the number of directions could be more than two, but for clarity we'll only handle the two-direction case.

To define the meaning of the loop, it will be useful to use the abbreviations BODY for $(b_1 s_1 + b_2 s_2)$, and b for $b_1 + b_2$. Informally, BODY is one iteration of the body of the loop, and b is the condition for continuing the loop. With these definitions, the meaning of the guarded iteration will always have the following form:

$$(\cdots)^\infty + \text{BODY}^* \bar{b},$$

The $(\cdots)^\infty$ portion indicates the infinite computations, while $\text{BODY}^* \bar{b}$ indicates the terminating computations. The $(\cdots)$ will vary, depending on the particular fairness assumptions that we choose.

In general, proving a program assertion about a DO–OD program requires partial correctness $([p]\,\text{BODY}^* \bar{b}\,[q])$ and TERMINATE(p, $\cdots$). Partial correctness can be handled in the same way as Section 5.2, so we won't deal with that here. Thus, this section is concerned with proving termination assertions of the form TERMINATE(p, $\cdots$) for various forms of $(\cdots)$ which arise from different fairness assumptions about guarded iteration.

No Fairness Assumption: With no fairness assumption, the algebraic meaning of the infinite part of the DO–OD loop is just BODY^∞. Here's the rule for proving its termination:

DO–OD Termination Rule: Let $(I, <)$ be a well-founded set and for each $i \in I$ let p_i be a guard such that:

For all k $(1 \le k \le 2)$: $[p_i b_k]\, s_k\, [\bigvee_{j<i} p_j]$.

Also suppose that p implies $\bigvee_{i \in I} p_i$. Then TERMINATE(p, BODY).

Proof: Consider any $i \in I$. Since $[p_i b_k]\, s_k\, [\bigvee_{j<i} p_j]$ holds for both k=1 and k=2, we also have $[p_i]\, b_1 s_1 + b_2 s_2\, [\bigvee_{j<i} p_j]$, which is equivalent to:

$$[p_i]\,\text{BODY}\,[\bigvee_{j<i} p_j].$$

But this is just the hypothesis of the Infinity Rule of Section 4.5; therefore TERMINATE($\bigvee_{i \in I} p_i$, BODY). Since p implies $\bigvee_{i \in I} p_i$, the needed result then follows from the Consequence Rule. □

Weak Fairness Assumption: The assumption of *weak fairness* forbids an infinite path from ignoring a direction when the corresponding guard is always true at the choice point. More precisely: for each direction k, an infinite path must either take direction k infinitely often or have b_k false infinitely often when the choice occurs. Algebraically, the weakly-fair infinite paths are expressed as:

$$(\textsc{body}^* \ (b_1 s_1 + \bar{b}_1 b_2) \ \textsc{body}^* \ (b_2 s_2 + \bar{b}_2 b_1))^\infty .$$

Intuitively, a weakly-fair infinite path must execute direction 1 infinitely often, unless guard b_1 fails infinitely often – hence the subterm $(b_1 s_1 + \bar{b}_1 b_2)$. And similarly for direction 2.

Here's the rule for proving termination of a DO–OD loop with the weak fairness assumption. The rule is based on the Weakly-Fair Termination Rule of [5].

Weakly-Fair Termination Rule: Let $(I, <)$ be a well-founded set and for each $i \in I$ let p_i be a guard such that:

(i) For all k $(1 \le k \le 2)$: $[p_i b_k] \, s_k \, [p_i]$, and

(ii) For some k $(1 \le k \le 2)$: $[p_i b_k] \, s_k \, [\bigvee_{j<i} p_j]$, and $p_i \bar{b}_k$ implies $\bigvee_{j<i} p_j$.

Also suppose that p implies $\bigvee_{i \in I} p_i$.

Then $\textsc{terminate}(p, \ \textsc{body}^* \ (b_1 s_1 + \bar{b}_1 b_2) \ \textsc{body}^* \ (b_2 s_2 + \bar{b}_2 b_1))$.

Proof: We can show that the hypothesis of the $^\infty$-Composition Rule (Section 4.5) is met, by breaking $(\textsc{body}^* \ (b s_1 + \bar{b}_1 b_2) \ \textsc{body}^* \ (b s_2 + \bar{b}_2 b_1))$ into the four pieces

$$t_1 = \textsc{body}^*,$$
$$t_2 = b_1 s_1 + \bar{b}_1 b_2,$$
$$t_3 = \textsc{body}^*,$$
$$t_4 = b_2 s_2 + \bar{b}_2 b_1.$$

The remainder of the proof is similar to the previous Fair-WHILE Termination Rule. □

Strong Fairness Assumption: An alternate fairness assumption for a guarded iteration is called *strong fairness*. Under this assumption, an infinite path must execute each branch infinitely often, provided that the branch's guard is true infinitely often. In other words, an infinite path may eventually ignore a branch, provided that the branch's guard is eventually always false. Algebraically, the strongly-fair infinite paths can be expressed as:

$$(\text{BODY}^* \ (b_1 s_1 + (\bar{b}_1 b_2 s_2)^\infty) \ \text{BODY}^* \ (b_2 s_2 + (\bar{b}_2 b_1 s_1)^\infty))^\infty.$$

Intuitively, this expression says that a strongly-fair infinite path may execute a finite number of iterations (BODY^*), but eventually it must either execute direction 1 ($b_1 s_1$) or go into a loop where guard 1 remains false ($(\bar{b}_1 b_2 s_2)^\infty$). And similarly for direction 2.

Program assertions about strongly-fair guarded iteration can be proved with the following rule, based on the Strongly-Fair Termination Rule of [5].

Strongly-Fair Termination Rule: Let $(I, <)$ be a well-founded set and for each $i \in I$ let p_i be a guard such that:

(i) For all k $(1 \le k \le 2)$: $[p_i b_k]\, s_k\, [p_i]$.

(ii) Either $[p_i b_1]\, s_1\, [\bigvee_{j<i} p_j]$, and $\text{TERMINATE}(p_i,\ \bar{b}_1 b_2 s_2)$,

or $[p_i b_2]\, s_2\, [\bigvee_{j<i} p_j]$, and $\text{TERMINATE}(p_i,\ \bar{b}_2 b_1 s_1)$.

Also suppose that p implies $\bigvee_{i \in I} p_i$.

Then $\text{TERMINATE}(p,\ \text{BODY}^* \ (b_1 s_1 + (\bar{b}_1 b_2 s_2)^\infty) \ \text{BODY}^* \ (b s_2 + (\bar{b}_2 b_1 s_1)^\infty))$.

Proof: We can show that the hypothesis of the $^\infty$-Composition Rule (Section 4.5) is met, by breaking $\text{BODY}^* \ (b s_1 + (\bar{b}_1 b_2 s_2)^\infty) \ \text{BODY}^* \ (b s_2 + (\bar{b}_2 b_1 s_1)^\infty)$ into the four pieces

$$t_1 = \text{BODY}^*,$$
$$t_2 = b_1 s_1 + (\bar{b}_1 b_2 s_2)^\infty,$$
$$t_3 = \text{BODY}^*,$$
$$t_4 = b_2 s_2 + (\bar{b}_2 b_1 s_1)^\infty.$$

The remainder of the proof is similar to the previous Fair-WHILE Termination Rule. □

6. Semantic Completeness

We have given rules which are sufficient for proving program assertions for programs of the forms:

st	**WHILE** b **DO** s
$s+t$	**WHILE** b **DO** $s_1 \Box s_2$ (Fair)
s^*	**DO–OD** loop
s^∞	Weakly Fair **DO–OD** loop
IF b **THEN** s **ELSE** t	Strongly Fair **DO–OD** loop

Some of these rules are semantically complete. That is, it is always possible to prove program assertions of a given form by using the corresponding rule. For example, the Addition Rule states that $[p]\, s+t\, [q]$ if *and only if* $[p]\, s\, [q]$ and $[p]\, t\, [q]$. But in general, the rules are not semantically complete. For example, it is quite easy to define an assertional s-ring with an element s such that $[1]\, ss\, [1]$ is valid, but there is no guard q with both $[1]\, s\, [q]$ and $[q]\, s\, [1]$.

Ideally, we would like our rules to be semantically complete, since this guarantees that the rules are as strong as possible. This section gives a condition for all the rules in this paper to be semantically complete.

Completeness Theorem. Suppose that

(i) The Composition Rule is semantically complete, so that $[p]\, st\, [r]$ if *and only if* there exists some q with $[p]\, s\, [q]$ and $[q]\, t\, [r]$.

(ii) The Infinity Rule is semantically complete, so that TERMINATE(p, s) if *and only if* the hypothesis of the Infinity Rule is met.

(iii) The Boolean algebra of guards is a complete Boolean algebra, so that for any set W of guards, $\bigvee W$ exists.

Then the Iteration Rule and each of the rules of Sections 4 and 5 are also semantically complete.

Proof: The proof consists of a sequence of lemmas, showing that the three conditions imply that each of the indicated rules is semantically complete. These lemmas are given in the remainder of this section. □

It is not hard to show that the hypotheses of the Completeness Theorem hold for the the s-ring of strict relations; hence all the indicated rules are semantically complete when programs are represented by strict relations. In particular, the fairness rules are complete — and the demonstration of that completeness comes from completeness of simpler rules. This may be easier than the usual direct proof in [5].

For the remainder of this section, we will assume the hypotheses of the Completeness Theorem, and use this assumption to prove the "sequence of lemmas" mentioned in the Completeness Theorem's proof. Prior to this sequence of lemmas, we will prove a few results about useful guards called *kernels* and *domains*, which are guaranteed to exist by the hypotheses of the Completeness Theorem.

6.1 Kernels and Domains

The third hypothesis of the Completeness Theorem is that the Boolean algebra of guards is complete. Intuitively, this assures us that there are "enough" guards around. In particular, it allows us to define guards called kernels and domains, as follows.

Definition. Let s be an element in an s-ring S where the Boolean algebra of guards is complete. The *kernel* of t (denoted by $\mathrm{KER}(s)$) and the *domain* of s (denoted by $\mathrm{DOM}(s)$) are defined by:

$$\mathrm{KER}(s) = \bigvee\{p \in GUARDS_S \mid ps = 0\} \quad \text{and} \quad \mathrm{DOM}(s) = \bigwedge\{p \in GUARDS_S \mid ps = s\}$$

□

In the s-ring of strict relations, a state d satisfies $\mathrm{DOM}(s)$ provided that $(d,e) \in s$ for some e. For any s-ring, $\mathrm{KER}(s)$ is always the complement of $\mathrm{DOM}(s)$. This property and other properties of kernels are summarized below. The theorems make use of the first and third hypotheses of the Completeness Theorem.

Theorem 6.1.0 $\mathrm{KER}(s) = \overline{\mathrm{DOM}(s)}$.

Proof: By the infinite DeMorgan Law, $\overline{\mathrm{DOM}(s)} = \bigvee\{\bar{p} \in GUARDS_S \mid ps = s\}$. But $ps = s$ iff $\bar{p}s = 0$ in any s-ring. Therefore, $\overline{\mathrm{DOM}(s)} = \bigvee\{\bar{p} \in GUARDS_S \mid \bar{p}s = 0\}$, which is just $\mathrm{KER}(s)$. □

Theorem 6.1.1. $\text{KER}(s)s = 0$ and $\text{DOM}(s)s = s$.

Proof: The first equality follows immediately from the Continuity Axiom and the definition of kernels. The second equality follows from:

$$s = 1s = (\text{KER}(s)+\text{DOM}(s))s = \text{KER}(s)s + \text{DOM}(s)s = 0 + \text{DOM}(s)s = \text{DOM}(s)s.$$

□

Theorem 6.1.2. $st = 0$ if and only if $s\,\text{DOM}(t) = 0$.

Proof: Assume $s\,\text{DOM}(t) = 0$. Then by 6.1.1 we have $st = s\,\text{DOM}(t)t = 0t = 0$. On the other hand, assume $st = 0$. Therefore $[1]\,st\,[0]$, and by the first hypothesis of the Completeness Theorem, there exists some guard q such that $[1]\,s\,[q]$ and $[q]\,t\,[0]$. For this value of q the two assertions can be rewritten as:

$$(1)\ s\bar{q} = 0, \text{ and } (2)\ qt = 0.$$

From (2), we have that q is below $\text{KER}(t)$ in the Boolean algebra of guards, or equivalently $\bar{q}$ is above $\text{DOM}(t)$. But recall that the order on the guards is the summation order, so there exists some p such that $\bar{q} = p + \text{DOM}(t)$. Finally, since $s\bar{q} = 0$, and the s-ring is zerosum-free, this implies that $s\,\text{DOM}(t)$ is also 0. □

Theorem 6.1.3. $\text{KER}(st) = \text{KER}(s\,\text{DOM}(t))$.

Proof:

$$\text{KER}(st) = \bigvee\{p \mid pst = 0\} = \bigvee\{p \mid ps\,\text{DOM}(t) = 0\} = \text{KER}(s\,\text{DOM}(t)).$$

The second equality is from 6.1.2. □

Theorem 6.1.4. $\text{KER}(s+t) = \text{KER}(s)\text{KER}(t)$.

Proof: Let $A = \{p \mid ps = 0\}$ and $B = \{p \mid pt = 0\}$. Then

$$\text{KER}(s+t) = \bigvee\{p \mid p(s+t) = 0\} = \bigvee(A \cap B) = (\bigvee A)(\bigvee B) = \text{KER}(s)\text{KER}(t).$$

The third equality is not valid for arbitrary A and B, but in this case it is allowed since A and B are downward closed. □

Theorem 6.1.5. $[\text{KER}(s^*t)]\,s\,[\text{KER}(s^*t)]$.

Proof: From 6.1.1 we have $\text{KER}(s^*t)s^*t = 0$. Since $s^* = ss^* + 1$, this implies $\text{KER}(s^*t)ss^*t = 0$. From 6.1.2, this implies $\text{KER}(s^*t)s\,\text{DOM}(s^*t) = 0$. This last equality is just

$[\mathrm{KER}(s^*t)]\, s\, [\mathrm{KER}(s^*t)]$. □

6.2 Completeness of the Iteration Rule

We can now use kernels to demonstrate the completeness of the Iteration Rule:

Lemma 6.2. Assume the hypotheses of the Completeness Theorem, and that $[q]\, s^*\, [r]$. Then there exists a guard p such that q implies p, and $[p]\, s\, [p]$, and p implies r.

Proof: Define $p = \mathrm{KER}(s^*\bar{r})$. We will show that this choice of p meets the three requirements.

Step 1: Show that q implies p.

We must show that q is below p in the Boolean algebra order. From $[q]\, s^*\, [r]$ it follows that $qs^*\bar{r} = 0$, and therefore q is below $\bigvee \{q \mid qs^*\bar{r} = 0\} = \mathrm{KER}(s^*\bar{r}) = p$.

Step 2: Show that p implies r.

From 6.1.1 we have $ps^*\bar{r} = \mathrm{KER}(s^*\bar{r})s^*\bar{r} = 0$. Since $s^* = ss^* + 1$, this implies $p(ss^*+1)\bar{r} = 0$, and from the zerosum-free axiom we must have $p\bar{r} = 0$. This occurs only if p implies r in the Boolean algebra, which is the result we need.

Step 3: Show that $[p]\, s\, [p]$.

We must show $ps\bar{p} = 0$, which is done here:

$$
\begin{aligned}
ps\bar{p} &= \mathrm{KER}(s^*\bar{r})s\bar{p} && \text{(Definition of } p\text{)}\\
&= \mathrm{KER}(ss^*\bar{r} + \bar{r})\, s\bar{p} && (s^* = ss^* + 1)\\
&= \mathrm{KER}(\bar{r})\, \mathrm{KER}(ss^*\bar{r})\, s\bar{p} && (6.1.4)\\
&= \mathrm{KER}(\bar{r})\, \mathrm{KER}(s\, \mathrm{DOM}(s^*\bar{r}))\, s\bar{p} && (6.1.3)\\
&= \mathrm{KER}(\bar{r})\, \mathrm{KER}(s\bar{p})\, s\bar{p} && \text{(Definition of } p\text{)}\\
&= \mathrm{KER}(\bar{r})\, 0 && (6.1.1)\\
&= 0 && \text{(Guard times 0 is 0)}
\end{aligned}
$$

□

6.3 Completeness of the WHILE Rule

The semantic completeness of the **WHILE** Rule follows from the semantic completeness of the Iteration Rule and the Infinity Rule.

Lemma 6.3. Assume the hypotheses of the Completeness Theorem, and that $[q]$ **WHILE** b **DO** s $[r]$. Then the following two conditions hold (where "**or**" is join in the Boolean algebra of guards):

PARTIAL CORRECTNESS: There exists a guard p (the "loop invariant") such that q implies p, and $[pb]\,s\,[p]$, and p implies (b **or** r).

TERMINATION: There exists a well-founded set $(I,<)$ and a guard p_i for each $i \in I$ such that:

(i) For every $i \in I$: $[p_i b]\,s\,[\bigvee_{j<i} p_j]$, and

(ii) q implies $\bigvee_{i \in I} p_i$.

Proof:

$[q]$ **WHILE** b **DO** s $[r]$ => $[q]\,(bs)^\infty + (bs)^*\bar{b}\,[r]$ (Definition of **WHILE**)

=> $[q]\,(bs)^\infty\,[r]$ and $[q]\,(bs)^*\bar{b}\,[r]$ (Addition Rule)

=> $[q]\,(bs)^\infty\,[r]$ and $[q]\,(bs)^*\,[b$ **or** $r]$

By hypothesis (ii) of the Completeness Theorem, the first assertion in the last line implies TERMINATION. By Lemma 6.2, the second assertion in the last line implies PARTIAL CORRECTNESS. □

6.4 Completeness of the Fair-WHILE Termination Rule

Semantic completeness of the Fair-**WHILE** Termination Rule follows from the semantic completeness of the Infinity Rule. The proof is omitted, since it is just a simplification of the proof of Lemma 6.6.

Lemma 6.4. Assume the hypotheses of the Completeness Theorem, and that **TERMINATE**$(p,\, t^*bs_1t^*bs_2)$, where $t=(bs_1+bs_2)$. Then there exists a well-founded set $(I,<)$ and a guard p_i for each $i \in I$ such that p implies $\bigvee_{i \in I} p_i$ and for each $i \in I$:

(i) For all k $(1 \le k \le 2)$: $[p_i b]\, s_k\, [p_i]$, and

(ii) For some k $(1 \le k \le 2)$: $[p_i b]\, s_k\, [\bigvee_{j<i} p_j]$.

□

6.5 Completeness of the DO–OD Termination Rule

In this section, we show that the DO–OD Termination Rule is semantically complete for proving termination assertions of the form TERMINATE(p, BODY), where BODY is defined as $b_1 s_1 + b_2 s_2$. Throughout the rest of Section 6, we will always take BODY to be defined this way.

Lemma 6.5. Assume the hypotheses of the Completeness Theorem, and that TERMINATE(p, BODY). Then there exists a well-founded set $(I, <)$ and a guard p_i for each $i \in I$ such that p implies $\bigvee_{i \in I} p_i$ and for each $i \in I$:

For all k $(1 \le k \le 2)$: $[p_i b_k]\, s_k\, [\bigvee_{j<i} p_j]$.

Proof: By hypothesis (ii) of the Completeness Theorem, the assertion TERMINATE(p, BODY) implies the existence of a well-founded set $(I, <)$ and a guard p_i for each $i \in I$ such that p implies $\bigvee_{i \in I} p_i$ and for each $i \in I$:

$$[p_i]\,\text{BODY}\,[\bigvee_{j<i} p_j].$$

Since BODY$=b_1 s_1 + b_2 s_2$, the addition rule states that this is equivalent to the two assertions:

$$[p_i]\, b_1 s_1\, [\bigvee_{j<i} p_j] \quad \text{and} \quad [p_i]\, b_2 s_2\, [\bigvee_{j<i} p_j].$$

And this is equivalent to the two required assertions:

$$[p_i b_1]\, s_1\, [\bigvee_{j<i} p_j] \quad \text{and} \quad [p_i b_2]\, s_2\, [\bigvee_{j<i} p_j].$$

□

6.6 Completeness of the Weakly-Fair Termination Rule

This section shows the semantic completeness of the Weakly-Fair Termination Rule, for proving termination assertions of the form

$$\text{TERMINATE}(p,\ \text{BODY}^*\ (b_1s_1+\bar{b}_1b_2)\ \text{BODY}^*\ (b_2s_2+\bar{b}_2b_1)).$$

Recall that this is the form of termination assertion that is needed to prove termination of a DO–OD loop under the weak fairness assumption.

Lemma 6.6. Assume the hypotheses of the Completeness Theorem, and that $\text{TERMINATE}(p,\ \text{BODY}^*\ (b_1s_1+\bar{b}_1b_2)\ \text{BODY}^*\ (b_2s_2+\bar{b}_2b_1))$. Then there exists a well-founded set $(I, <)$ and a guard p_i for each $i \in I$ such that p implies $\bigvee_{i \in I} p_i$ and for each $i \in I$:

(i) For all k $(1 \le k \le 2)$: $[p_i b_k]\, s_k\, [p_i]$, and

(ii) For some k $(1 \le k \le 2)$: $[p_i b_k]\, s_k\, [\bigvee_{j<i} p_j]$, and $p_i \bar{b}_k$ implies $\bigvee_{j<i} p_j$.

Proof: By hypothesis (ii) of the Completeness Theorem, the assertion $\text{TERMINATE}(p,\ \text{BODY}^*\ (b_1s_1+\bar{b}_1b_2)\ \text{BODY}^*\ (b_2s_2+\bar{b}_2b_1))$ implies the existence of a well-founded set $(H, <)$ and a guard p_i for each $i \in H$ such that p implies $\bigvee_{i \in H} p_i$ and for each $i \in H$:

$$[p_i]\ \text{BODY}^*\ (b_1s_1+\bar{b}_1b_2)\ \text{BODY}^*\ (b_2s_2+\bar{b}_2b_1)\, [\bigvee_{j<i} p_j].$$

We will use this well-founded set H to construct another well-founded set I which meets the requirements of the lemma.

Let $I = H \times \{1,2\}$ be the well-founded set with an ordering defined by $(j,m) < (i,n)$ if and only if $j < i$ or $(j = i$ and $m=1$ and $n=2)$. For each $(i,n) \in I$, we define a guard p_{in} as follows:

$$p_{i1} = \text{KER}(\text{BODY}^* t_2\ \overline{\bigvee_{j<i} p_j})$$

$$p_{i2} = \text{KER}(\text{BODY}^* t_1\ \bar{p}_{i1}),$$

where $t_1 = b_1s_1+\bar{b}_1b_2$, and similarly for $t_2 = b_2s_2+\bar{b}_2b_1$. It remains to show that this definition of the set I and the guards p_{in} meets the requirements of the lemma. We do this in three steps, listed below.

Step 1: Show that p implies $\bigvee_{(i,n) \in I} p_{in}$.

We know that p implies $\bigvee_{i \in H} p_i$. Therefore, it is sufficient to show that for each

$i \in H$, p_i implies p_{i2}. Here is the proof, for an arbitrary $i \in H$:

$[p_i]\,\text{BODY}^*t_1\text{BODY}^*t_2\,[\bigvee_{j<i} p_j]$	Given about p_i
$\Rightarrow\; p_i\text{BODY}^*t_1\text{BODY}^*t_2\,\overline{\bigvee_{j<i} p_j} = 0$	Definition of Assertion
$\Rightarrow\; p_i\text{BODY}^*t_1\text{DOM}(\text{BODY}^*t_2\,\overline{\bigvee_{j<i} p_j}) = 0$	(6.1.2)
$\Rightarrow\; p_i\text{BODY}^*t_1\bar{p}_{i1} = 0$	Definition of p_{i1}
$\Rightarrow\; p_i\text{DOM}(\text{BODY}^*t_1\bar{p}_{i1}) = 0$	(6.1.2)
$\Rightarrow\; p_i\bar{p}_{i2} = 0$	Definition of p_{i2}
$\Rightarrow\; p_i$ implies p_{i2}	Meaning of "implies"

Step 2: Show that for all $(i,n) \in I$: there exists k such that $[p_{in}b_k]\,s_k\,[\bigvee_{(j,m)<(i,n)} p_{jm}]$.

We show this only for $n=2$, since the case of $n=1$ is similar. For $n=2$, we can always choose the value of k to be 1, since then:

$p_{i2}\text{BODY}^*t_1\bar{p}_{i1} = 0$	(6.1.1)
$\Rightarrow\; p_{i2}1t_1\bar{p}_{i1} = 0$	($\text{BODY}^* = \text{BODY}(\text{BODY}^*)+1$, and zerosum-free)
$\Rightarrow\; p_{i2}b_1s_1\bar{p}_{i1} = 0$	($t_1 = b_1s_1 + \bar{b}_1b_2$, and zerosum-free)
$\Rightarrow\; [p_{i2}b_1]\,s_1\,[p_{i1}]$	Definition of Assertion
$\Rightarrow\; [p_{i2}b_1]\,s_1\,[\bigvee_{(j,m)<(i,n)} p_{jm}]$	Consequence Rule

Step 3: Show that for all $(i,n) \in I$: and for $k=1,2$: $[p_{in}b_k]\,s_k\,[p_{in}]$.

We show this only for $n=2$, since the case of $n=1$ is similar. For both $k=1$ and $k=2$, we have:

$[\mathrm{KER}(\mathbf{BODY}^* t_1 \bar{p}_{i1})]\ \mathbf{BODY}\ [\mathrm{KER}(\mathbf{BODY}^* t_1 \bar{p}_{i1})]$	(6.1.5)
$\Rightarrow\ [p_{i2}]\ \mathbf{BODY}\ [p_{i2}]$	Definition of p_{i2}
$\Rightarrow\ [p_{i2}]\ b_k s_k\ [p_{i2}]$	($b_k s_k$ is one term of **BODY**)
$\Rightarrow\ [p_{i2} b_k]\ s_k\ [p_{i2}]$	Associativity

These three parts complete the proof. □

6.7 Completeness of the Strongly-Fair Termination Rule

This section discusses the semantic completeness of the Strongly-Fair Termination Rule, for proving termination assertions of the form

$$\mathbf{TERMINATE}(p,\ \mathbf{BODY}^*\ (b_1 s_1 + (\bar{b}_1 b_2 s_2)^\infty)\ \mathbf{BODY}^*\ (b_2 s_2 + (\bar{b}_2 b_1 s_1)^\infty)).$$

Recall that this is the form of termination assertion that is needed to prove termination of a DO–OD loop under the strong fairness assumption.

Lemma 6.7. Assume the hypotheses of the Completeness Theorem, and that $\mathbf{TERMINATE}(p,\ \mathbf{BODY}^*\ (b_1 s_1 + (\bar{b}_1 b_2 s_2)^\infty)\ \mathbf{BODY}^*\ (b_2 s_2 + (\bar{b}_2 b_1 s_1)^\infty))$. Then there exists a well-founded set $(I, <)$ and a guard p_i for each $i \in I$ such that p implies $\bigvee_{i \in I} p_i$ and for each $i \in I$:

(i) For all k $(1 \le k \le 2)$: $[p_i b_k]\ s_k\ [p_i]$.

(ii) Either $[p_i b_1]\ s_1\ [\bigvee_{j<i} p_j]$, and $\mathbf{TERMINATE}(p_i,\ \bar{b}_1 b_2 s_2)$,

or $[p_i b_2]\ s_2\ [\bigvee_{j<i} p_j]$, and $\mathbf{TERMINATE}(p_i,\ \bar{b}_2 b_1 s_1)$.

Proof: The proof is similar to that of Lemma 6.6, and will be included in a longer version of the paper. □

Acknowledgement

We would like to thank David Benson for comments on a preliminary version of this paper. Thanks also to Ernie Manes, whose comments in [11] prompted a simplification of the axioms of an assertional s-ring.

References

(1) S. Bloom and Z. Esik. Floyd-Hoare Logic in Iteration Theories, Technical Report #8801, Stevens Institute of Technology, Electrical Engineering and Computer Science Department (January 1988).

(2) E.W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs, *CACM* 18 (1975), 453-457.

(3) C.C. Elgot. Monadic computations and iterative algebraic theories, in: *Logic Colloquium 73* (H.E. Rose and J.C. Sheperdson, Eds.), (North-Holland Publishing Co., 1975), 175-230.

(4) C.C. Elgot. Matricial Theories, *J. Algebra* 42 (1976), 391-421.

(5) N. Francez. *Fairness*, Springer-Verlag, New York, 1986.

(6) D. Harel. First-order dynamic logic, LNCS 68 (Springer-Verlag, 1979).

(7) D. Harel and V.R. Pratt. Nondeterminism in Logic of Programs, in: *Proceedings of the 5th ACM Symposium on Principles of Programming Languages* (1978), 203-213.

(8) W. Kuich and A. Salomaa. *Semirings, Automata, Languages,* Springer-Verlag, 1985.

(9) M.G. Main and D.B. Benson. Functional behavior of nondeterministic programs, in: *Foundations of Computation Theory*, LNCS 158, (Springer-Verlag, 1983), 290-301.

(10) M.G. Main and D.B. Benson. Functional behavior of nondeterministic and concurrent programs, *Information and Control* 62 (1984), 144-189.

(11) E.G. Manes. Boolean Theories, preprint.

(12) E.G. Manes. Assertion semantics in a control category, *Theoretical Computer Science*, to appear.

(13) E.G. Manes. Assertional Categories, in: *Proceedings of the Third Workshop on Mathematical Semantics of Programming Languages*, Lecture Notes in Computer Science 298, Springer-Verlag (1988),85-120.

(14) E.G. Manes and M.A. Arbib. *Algebraic Approaches to Program Semantics*, Springer-Verlag, New York, 1986.

(15) E.G. Manes and D.B. Benson. The inverse semigroup of a sum-ordered semiring, *Semigroup Forum* 31 (1985), 129-152.