Adding Fair Choice to Dijkstra's Calculus

MANFRED BROY
Technische Universitaet Muenchen

GREG NELSON
Digital Equipment Corporation

The paper studies the incorporation of a fair nondeterministic choice operator into a generalization of Dijkstra's calculus of guarded commands. The generalization drops the law of the excluded miracle to allow commands that correspond to partial relations. Because of fairness, the new operator is not monotonic for the orderings that are generally used for proving the existence of least fixed points for recursive definitions. To prove the existence of fixed points it is necessary to consider several orderings at once, and to restrict the class of recursive definitions.

Categories and Subject Descriptors: D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*semantics*; D.3.3 [**Programming Languages**]: Language Constructs and Features—*concurrent programming structures*; F.1.2 [**Modes of Computation**]: Alternation and nondeterminism

General terms: Semantics, Fairness, Guarded commands

Additional Key Words and Phrases: dovetail, fairness, guarded commands, law of the excluded miracle, nondeterminism, partial commands, semantics

1. INTRODUCTION

The fixed-point method of denotational semantics for treating recursion is so successful that it is a surprise to find a programming language construct for which it does not seem to work. But this is the case for the dovetail operator, a simple construct that models the notion of the fair scheduling of two activities. The name "dovetail" refers to the fair interleaving of the parallel execution of the two arguments of the operation. The word is also used informally in recursion theory. The operational definition of dovetail (which we shall write $\triangledown$) is as follows:

$A \triangledown B$

$\equiv$ Execute the commands A and B fairly in parallel, on separate copies of the state, accepting as an outcome any proper (i.e., nonlooping) outcome of either A or B.

By "fairly" it is meant that neither computation is permanently neglected in favor of the other.

As a hint at the power of the dovetail operator, we show how it immediately leads to

Authors' addresses: M. Broy, Technische Universitaet Muenchen, Institut fuer Informatik, D 80290 Muenchen, Germany; G. Nelson, System Research Center, Digital Equipment Corporation, 130 Lytton Ave., Palo Alto, CA 94301

unbounded nondeterminism. Operationally, a recursive call can be treated by replacing the call with the right-hand side of the recursive definition whenever necessary. From this it follows that the recursion

$$X \equiv (n := 0 \bigtriangledown (X \ ; \ n := n + 1))$$

has the solution $X \equiv$ "set n to any natural number." This is in contrast to the recursion

$$Y \equiv (n := 0 \ [] \ (Y \ ; \ n := n + 1))$$

which has the solution $Y \equiv$ "set n to any natural number, or loop." (The semicolon operator represents sequential composition, and the operator [] represents nondeterministic choice. The recursion with [] can loop, since a recursive call is available at every choice. The recursion with $\bigtriangledown$ cannot loop, since at each level of recursion—in particular, at the outermost level—the $n := 0$ branch cannot be delayed indefinitely.)

Unbounded nondeterminism can be handled in Dijkstra's calculus—for example, see Boom [1982]. But the dovetail operator is more of a challenge.

The dovetail operator is the imperative counterpart of the ambiguity operator introduced by McCarthy [1963]: "We define a basic ambiguity operator amb(x, y) whose possible values are x or y when both are defined, otherwise, whichever is defined". The ambiguity operator is not monotonic in the orderings of either the Smyth or the Plotkin powerdomains. Therefore its fixed-point theory, presented in Broy [1986], is far from straightforward. The dovetail operation also is not monotonic, and to treat it by the fixed-point method requires some of Broy's techniques. But the presence of partial commands introduces additional difficulties. In fact, we shall find that not all recursions involving dovetail have solutions.

We believe it would be straightforward to give operational semantics for dovetail using, say, structured operational semantics in the Plotkin style, and to prove that the operational semantics are consistent with the axiomatic semantics given in this paper. However, we have not carried out this project.

When executing $A \bigtriangledown B$ it is obvious that A and B have to be evaluated in parallel, since the undecidability of the halting problem implies that it cannot be decided in advance if A or B diverges. Therefore choosing one of them to execute first might condemn the execution to divergence although the termination of the other is possible, or even guaranteed. This is why fair interleaved execution is the only way to deal operationally with dovetail. Furthermore, if the fair interleaving is done by a fixed scheduling strategy, then certain results consistent with the semantics of dovetail would be ruled out by the operational semantics. For example, if both A and B terminate but B takes many more steps than A, a fixed scheduling strategy could rule out the result computed by B. This is a well-known effect when dealing with fairness assumptions. Fairness leads to unbounded nondeterminism (see for instance Apt and Plotkin [1986]) and therefore leaves the classical framework of computability. Therefore one might say that fairness cannot be implemented and is of only theoretical interest. However, if one thinks about a fair program not as defining one algorithm, but as defining an algorithm for every scheduler that is correct with respect to the fairness assumption, then fairness becomes a realistic and practically meaningful concept. The same applies to the dovetail operator.

We conclude this introduction by sketching how dovetail can be used to model classical weak fairness.

Recall that Dijkstra's repetitive construct **do** $A \ [] \ B$ **od** is executed by repeatedly selecting and executing either A or B, subject only to the condition that the scheduled command's guard be true. Let us define **fairdo**(A, B) similarly, but with the additional scheduling

requirement of *weak fairness*. This means that the scheduler must not permanently neglect a command whose guard is permanently true.

To implement **fairdo** using dovetail, we first make three auxiliary definitions. Informally, we are aiming for the following definitions:

$\mathbf{Seq}(X) =$ execute X a finite number of times, without getting stuck in an infinite loop in X.

$X^+ =$ execute X a nonzero finite number of times.

$X^* =$ execute X a finite number of times.

Here are the formal definitions:

$$\mathbf{Seq}(X) = (X\,;\mathbf{Seq}(X)) \bigtriangledown \mathit{Skip}$$

$$X^+ = \mathbf{Seq}(X)\,;X$$

$$X^* = X^+ \,[]\, \mathit{Skip}\,.$$

Notice that $\mathbf{Seq}(X)$ cannot loop, even if X can, because the "$\bigtriangledown$ *Skip*" will guarantee termination. However, both X^+ and X^* can loop if X can.

Now we define

$$\mathbf{fairdo}(A,B) \equiv \mathbf{do}\ A^+\,;(B \,[\!]\, \mathit{Skip})\ []\ B^+\,;(A \,[\!]\, \mathit{Skip})\ \mathbf{od}\,.$$

(The command $A \,[\!]\, B$ is read "A else B"; it is like $A\ []\ B$ except that A is given precedence over B. That is, it means to execute A if A's guard is true, else to execute B. The formal definition is in the next section.)

It will be left to the reader to persuade himself of the appropriateness of the formula above for **fairdo**, and to generalize to $\mathbf{fairdo}(A,B,C)$.

Finally, to connect dovetail to an existing formalism for reasoning about fairness, we remark that the "leads-to" relation of the Unity system (Chandy and Misra [1988]) can now be defined: P leads to Q with respect to a Unity program whose body consists of the guarded commands A and B if

$$P \Rightarrow \mathrm{wp}.\mathbf{fairdo}(\neg Q \rightarrow A, \neg Q \rightarrow B).\mathrm{TRUE}\,.$$

So much for the motivation of dovetail. Our goal is to reason about it in the same compositional style as we reason about $[]$, ;, or $\rightarrow$.

2. PRELIMINARIES

Our framework is the generalization of Dijkstra's calculus (Dijkstra [1976]; Dijkstra and Scholten [1990]) described by Nelson [1989]. The definitions we need will be repeated in this section.

We use a left-associative infix dot to denote function application, together with Curry's convention for reducing n-ary functions to unary functions. That is, we write $f.x$ instead of $f(x)$, and $g.x.y$ instead of $g(x, y)$, and $g.x$ instead of $(\lambda y.\ g(x, y))$.

A command A is defined to be a pair of predicate transformers, written wp.A and wlp.A, satisfying the *pairing condition*, which is that for any predicate R,

$$\text{wp}.A.R \equiv \text{wp}.A.\text{TRUE} \wedge \text{wlp}.A.R\,,$$

and the *conjunctivity condition*, which is that wlp.A distributes over any conjunction. It follows that the predicate transformer wp.A distributes over any nonempty conjunction.

For any command A, we define two predicates, read *guard* of A and *halt* of A, as follows:

$$\text{grd}.A \equiv \neg\, \text{wp}.A.\text{FALSE}\,,$$
$$\text{hlt}.A \equiv \text{wp}.A.\text{TRUE}\,.$$

The predicate grd.A characterizes those states from which failure is impossible; the predicate hlt.A characterizes those states from which termination is guaranteed. If grd.$A \equiv$ TRUE, then A is a *total* command.

For commands A and B, we define

$$\begin{aligned} A \sqsubseteq_{\text{wp}} B \ :&\quad \text{wp}.A.R \Rightarrow \text{wp}.B.R \quad \text{for any } R \\ A \sqsubseteq_{\text{wlp}} B \ :&\quad \text{wlp}.B.R \Rightarrow \text{wlp}.A.R \quad \text{for any } R \\ A \sqsubseteq B \ :&\quad A \sqsubseteq_{\text{wp}} B \text{ and } A \sqsubseteq_{\text{wlp}} B\,. \end{aligned}$$

The relation $A \sqsubseteq B$ is read A *approximates* B; it is a complete partial order on the set of all commands. Informally, A approximates B if A can be obtained by substituting looping outcomes for some of B's outcomes. For example, executing a command for a limited amount of time produces an approximation to the command, provided that computations that exceed the time limit are classified as loops. Thus the always-diverging command *Loop* approximates every command, and a command with no looping outcomes approximates no command except itself.

We will use square brackets to denote the following drastic map on predicates: $[\text{TRUE}] = \text{TRUE}$, and $[P] = \text{FALSE}$ for all other P.

We will write

(operation dummies : range : term)

to denote the combination via the given operation of the values assumed by the given term as the dummies vary over the given range. The operation must be commutative, associative, and (if the range is empty) possess an identity. If the range is obvious from the context, it will be omitted. For example, the greatest lower bound of the set S of predicates is denoted by $(\wedge\ P : P \in S : P)$, or by $(\wedge\ P :: P)$ if S is obvious from the context.

In order to express formulas involving the two operators wp and wlp compactly, the *parenthesis convention* will be used: a formula containing parenthesized expressions represents two formulas, in one of which the parenthesized expressions are ignored, in the other of which each parenthesized expression is either inserted, or substituted for the item to its left, whichever is suggested by the context. For example, consider the following two formulas,

proved in Nelson's paper, in which A ranges over any $\sqsubseteq$-chain and $\sqcup$ denotes join (that is, least upper bound) with respect to $\sqsubseteq$:

$$\text{wlp}.(\sqcup\, A :: A).R \equiv (\wedge\, A :: \text{wlp}.A.R)$$
$$\text{wp}.(\sqcup\, A :: A).R \equiv (\vee\, A :: \text{wp}.A.R)\,.$$

Using the parenthesis convention, they are equivalent to the single formula

$$\text{w(l)p}.(\sqcup\, A :: A).R \equiv (\vee(\wedge)\, A :: \text{w(l)p}.A.R)\,.$$

Here are the definitions of the basic commands and operators:

$$\text{w(l)p}.\mathit{Fail}.R \equiv \text{TRUE}$$
$$\text{w(l)p}.\mathit{Skip}.R \equiv R$$
$$\text{w(l)p}.\mathit{Loop}.R \equiv \text{FALSE}\ (\text{TRUE})$$
$$\text{w(l)p}.\mathit{Havoc}.R \equiv [R]$$
$$\text{w(l)p}.(A \,[]\, B).R \equiv \text{w(l)p}.A.R \wedge \text{w(l)p}.B.R$$
$$\text{w(l)p}.(A\,;\,B).R \equiv \text{w(l)p}.A.(\text{w(l)p}.B.R)$$
$$\text{w(l)p}.(P \rightarrow A).R \equiv \neg\, P \vee \text{w(l)p}.A.R$$
$$\text{w(l)p}.[\![\, x \mid A \,]\!].R \equiv (\forall x :\ \text{w(l)p}.A.R)$$
$$\text{w(l)p}.(A \boxtimes B).R \equiv \text{w(l)p}.A.R \wedge (\text{grd}.A \vee \text{w(l)p}.B.R)\,.$$

In the equation for $\text{w(l)p}.[\![\, x \mid A \,]\!].R$ we assume that x is not free in R; otherwise x must be replaced by some fresh variable in R, then the precondition must be computed, and then in the result the fresh variable must be replaced by x.

All of these operations are monotonic with respect to the approximation order $\sqsubseteq$. Except for *Havoc* and $\boxtimes$, they are likely to be familiar. The command *Havoc* relates each initial state to every outcome except the looping outcome. The command $A \boxtimes B$ means "execute A unless it fails, in which case execute B." Its precondition equation can be derived from the formula

$$A \boxtimes B \ \equiv\ A \,[]\, (\neg\, \text{grd}.A \rightarrow B)\,.$$

3. DEFINITION AND ELEMENTARY PROPERTIES OF DOVETAIL

The precondition equations for $\bigtriangledown$ are somewhat subtle:

$$\text{wlp}.(A \bigtriangledown B).R \equiv \text{wlp}.A.R \wedge \text{wlp}.B.R$$

$$\begin{aligned}&\text{hlt}.(A \bigtriangledown B) \equiv\\ &\quad (\text{hlt}.A \vee \text{hlt}.B)\ \wedge\\ &\quad (\text{grd}.A \vee \text{hlt}.B)\ \wedge\\ &\quad (\text{grd}.B \vee \text{hlt}.A)\,.\end{aligned}$$

That is, as far as wlp is concerned, $\bigtriangledown$ is the same as $[]$. It differs by having a more liberal wp equation: to ensure that $A \bigtriangledown B$ halts, it suffices to forbid A and B from both looping and to forbid either from looping in a state where the other fails. The value of wp for postconditions other than TRUE is determined by the pairing condition. To verify that $A \bigtriangledown B$ is a command we must show that its wlp-transformer is conjunctive; but this is immediate.

The hlt equation for dovetail has an alternative form:

$$\begin{aligned}&\mathrm{hlt}.(A \bigtriangledown B) \equiv\ .\\ &\quad (\mathrm{hlt}.A \wedge \mathrm{hlt}.B) \vee\\ &\quad (\mathrm{grd}.A \wedge \mathrm{hlt}.A) \vee\\ &\quad (\mathrm{grd}.B \wedge \mathrm{hlt}.B)\,.\end{aligned}$$

The alternative form is sometimes useful, although we will not use it in this paper. It can be derived from the first form by distributing $\wedge$ over $\vee$ and simplifying.

LEMMA A. *For any A, B, we have* $\mathrm{grd}.(A \bigtriangledown B) \equiv \mathrm{grd}.A \vee \mathrm{grd}.B$.

PROOF. This is easy to see when A and B are viewed as relations, since a looping outcome of (say) A from some initial state can be excluded from $A \bigtriangledown B$ only if B has a proper outcome from that state. Thus, although $A \bigtriangledown B$ is smaller than the relational union, its domain is equal to the domain of the relational union.

The axiomatic proof begins with the observation that for any command A,

$$\mathrm{wlp}.A.\mathrm{FALSE} \Rightarrow (\mathrm{grd}A = \neg\,\mathrm{hlt}.A) \qquad (*)$$

whose proof is as follows: in any state where $\mathrm{wlp}.A.\mathrm{FALSE}$ holds,

$$\begin{aligned}&\quad \mathrm{grd}.A\\ &\equiv \neg\,\mathrm{wp}.A.\mathrm{FALSE}\\ &\equiv \neg\,(\mathrm{wp}.A.\mathrm{TRUE} \wedge \mathrm{wlp}.A.\mathrm{FALSE})\\ &\equiv \neg\,\mathrm{wp}.A.\mathrm{TRUE}\\ &\equiv \neg\,\mathrm{hlt}.A\end{aligned}$$

Armed with this observation, we prove Lemma A by deriving the complement of the right side from the complement of the left side:

$$\begin{aligned}&\quad \neg\,\mathrm{grd}.(A \bigtriangledown B)\\ &\equiv \mathrm{wp}.(A \bigtriangledown B).\mathrm{FALSE}\\ &\equiv \mathrm{hlt}.(A \bigtriangledown B) \wedge \mathrm{wlp}.(A \bigtriangledown B).\mathrm{FALSE}\\ &\equiv (\mathrm{hlt}.A \vee \mathrm{hlt}.B) \wedge (\mathrm{hlt}.A \vee \mathrm{grd}.B) \wedge (\mathrm{hlt}.B \vee \mathrm{grd}.A)\\ &\quad \wedge\ \mathrm{wlp}.A.\mathrm{FALSE} \wedge \mathrm{wlp}.B.\mathrm{FALSE}\\ &\equiv \{\ (*),\ \text{twice}\ \}\\ &\quad (\mathrm{hlt}.A \vee \mathrm{hlt}.B) \wedge (\mathrm{hlt}.A \vee \neg\,\mathrm{hlt}.B) \wedge (\mathrm{hlt}.B \vee \neg\,\mathrm{hlt}.A)\\ &\quad \wedge\ \mathrm{wlp}.A.\mathrm{FALSE} \wedge \mathrm{wlp}.B.\mathrm{FALSE}\\ &\equiv \{\ \text{Distributivity on conjuncts 1 and 2; 1 and 3}\ \}\\ &\quad \mathrm{hlt}.A \wedge \mathrm{hlt}.B \wedge \mathrm{wlp}.A.\mathrm{FALSE} \wedge \mathrm{wlp}.B.\mathrm{FALSE}\\ &\equiv \mathrm{wp}.A.\mathrm{FALSE} \wedge \mathrm{wp}.B.\mathrm{FALSE}\\ &\equiv \neg\,\mathrm{grd}.A \wedge \neg\,\mathrm{grd}.B\\ &\equiv \neg\,(\mathrm{grd}.A \vee \mathrm{grd}.B)\quad \square\end{aligned}$$

LEMMA B. *For any A, B, we have* $(A \,[]\, B) \sqsubseteq (A \bigtriangledown B)$.

PROOF. This is also easy to see when the commands are viewed as relations, since $A \,[]\, B$ can differ from $A \bigtriangledown B$ only by having extra looping outcomes from states where $A \bigtriangledown B$ has at least one nonlooping outcome, and this is precisely the difference that is allowed by the approximation relation.

The axiomatic proof is as follows. The $\sqsubseteq_{\text{wlp}}$ part of the proof is trivial, since $[]$ and $\triangledown$ have the same wlp-equation. Because of the pairing condition and the fact that the two sides are wlp-equivalent, the $\sqsubseteq_{\text{wp}}$ part of the proof can be completed by showing that $\text{hlt}.(A \,[]\, B) \Rightarrow \text{hlt}.(A \triangledown B)$. The proof of this is:

$$
\begin{aligned}
& \text{hlt}.(A \,[]\, B) \\
\equiv\ & \text{hlt}.A \wedge \text{hlt}.B \\
\Rightarrow\ & (\text{hlt}.A \vee \text{hlt}.B) \wedge (\text{grd}.A \vee \text{hlt}.B) \wedge (\text{grd}.B \vee \text{hlt}.A) \\
\equiv\ & \text{hlt}.(A \triangledown B) \quad \square
\end{aligned}
$$

4. NONMONOTONICITY OF DOVETAIL

The reason that the classical fixed-point method does not work for dovetail is that dovetail is not monotonic with respect to the approximation relation. For example,

$$\mathit{Loop} \sqsubseteq \mathit{Havoc}$$

but

$$\mathit{Loop} \triangledown \mathit{Skip} \not\sqsubseteq \mathit{Havoc} \triangledown \mathit{Skip}$$

since $\mathit{Loop} \triangledown \mathit{Skip} \equiv \mathit{Skip}$ and $\mathit{Havoc} \triangledown \mathit{Skip} \equiv \mathit{Havoc}$, but $\mathit{Skip} \not\sqsubseteq \mathit{Havoc}$.

5 TWO FIXED-POINT THEOREMS FOR DOVETAIL

The main results of this paper are two fixed-point theorems for dovetail. Because of dovetail's nonmonotonicity, neither seems to be provable by the simple fixed-point method. In this section we state the theorems and outline an alternative method of proof that works for both of them.

We write $A \equiv_{\text{wlp}} B$ to mean that $\text{wlp}.A = \text{wlp}.B$.

Here is the first result:

THEOREM 1. *Let f be a map from commands to commands defined by an expression of the form $f.X \equiv \mathcal{E}$, where $\mathcal{E}$ is an expression built from the five operations*

$$[] \quad \rightarrow \quad ; \quad [\,|\,] \quad \triangledown$$

as well as the command parameter X and any number of fixed commands and predicates. Then f has a least fixed point in the order $\leq$, defined by:

$$A \leq B \ \equiv\ (A \sqsubseteq_{\text{wlp}} B) \wedge ((A \equiv_{\text{wlp}} B) \Rightarrow (A \sqsubseteq B)) .$$

The approximation order $\sqsubseteq$ is the intersection of $\sqsubseteq_{\text{wlp}}$ with $\sqsubseteq_{\text{wp}}$; the new order $\leq$ is a sort of lexicographic combination of $\sqsubseteq_{\text{wlp}}$ with $\sqsubseteq_{\text{wp}}$.

Theorem 1 cannot be proved as a simple application of the Knaster-Tarski Theorem, since none of the operators are monotonic with respect to $\leq$. For example, consider sequential composition: we have

$$x := 1 \ \leq\ (x := 1 \,[]\, x := 2),$$

but with C given by $(x = 1 \rightarrow \mathit{Skip}) \,[]\, (x = 2 \rightarrow \mathit{Loop})$ we have

$$x := 1 \,;\, C \ \not\leq\ (x := 1 \,[]\, x := 2) \,;\, C$$

since $x := 1 \not\leq x := 1 \,[]\, \mathit{Loop}$.

A more complicated argument is required, which we now outline. First, we will change the recursion $f.X \equiv \mathcal{E}$ to the similar recursion $f^*.X \equiv \mathcal{E}^*$, where $\mathcal{E}^*$ is obtained from $\mathcal{E}$ by replacing all occurrences of $\triangledown$ by $[]$. Theorem 8 of Nelson [1989] shows that f^* has a fixed point, say X^*. Operational intuition suggests that the only difference between the two recursions is that $\triangledown$ will exclude some looping outcomes that are included by $[]$. Therefore we expect f to have a fixed point that differs from X^* only by having fewer looping outcomes. Let S be the set of commands that differ from X^* only by having fewer looping outcomes. It turns out that $\triangledown$ is monotonic with respect to approximation when it is restricted to S. (More generally, it is monotonic when restricted to any equivalence class of $\equiv_{\mathrm{wlp}}$.) Furthermore, S is closed with respect to joins. Thus the Knaster-Tarski theorem can be applied, showing that S contains a fixed point of f. This proof will be completed in Section 7.

For example, consider the recursion

$$f.X \equiv X \triangledown \mathit{Skip}. \tag{1}$$

The related recursion is

$$f^*.X \equiv X \,[]\, \mathit{Skip}.$$

The least fixed point of f^* is $\mathit{Loop} \,[]\, \mathit{Skip}$. The set S of commands that differ from $\mathit{Loop} \,[]\, \mathit{Skip}$ only by having (possibly) fewer looping outcomes is the set of commands of the form

$$(P \rightarrow \mathit{Loop}) \,[]\, \mathit{Skip}$$

for all predicates P. On this set f has an approximation-least fixed point, namely *Skip*. (In fact, *Skip* is the unique fixed point on this set.)

Notice that the fixed point is not approximation-least: for example, *Havoc* is a fixed point of (1), but *Skip* does not approximate *Havoc*. Indeed, the set of fixed points of (1) is the set of commands that, viewed as relations, contain *Skip* and are contained by *Havoc*. This set is completely flat with respect to the approximation relation.

Minimizing with respect to either $\leq$ or $\sqsubseteq$ has the effect of excluding proper outcomes and including looping outcomes; however, $\leq$ gives precedence to the former. This is consistent with the construction in the proof, which first uses a fixed-point construction to locate the $\equiv_{\mathrm{wlp}}$ equivalence class of the fixed point (thus determining its set of proper outcomes) and then uses a second fixed-point construction to maximize the number of looping outcomes within this equivalence class.

The second result is that if semicolon is restricted so that its second argument is total, then all recursions have solutions. To state this precisely, we introduce the operation ;; on commands defined by

$$A;;B \;\equiv\; A\,;(B \boxtimes \mathit{Loop}).$$

Operationally, $A;;B$ loops whenever $A\,;B$ would backtrack from B to A. If B is total, there is no difference between $A\,;B$ and $A;;B$.

If A and B are commands, we write $A \equiv_{\mathrm{grd}} B$ to mean $\mathrm{grd}.A \equiv \mathrm{grd}.B$, and we write $A \equiv_* B$ to mean $A \equiv_{\mathrm{wlp}} B$ and $A \equiv_{\mathrm{grd}} B$.

THEOREM 2. *Let f be a map from commands to commands defined by an expression of the form $f.X \equiv \mathcal{E}$, where $\mathcal{E}$ is an expression built from the six operations*

$$[] \quad \rightarrow \quad ;; \quad [|] \quad \triangledown \quad \boxtimes$$

as well as the command parameter X and any number of fixed commands and predicates. Then f has a least fixed point in the order $\leq$, defined by:

$$A \leq B \;\equiv\; (A \sqsubseteq_{\mathrm{wlp}} B) \wedge ((A \equiv_* B) \;\Rightarrow\; (A \sqsubseteq B))\,.$$

The proof of the second theorem is very similar to the proof of the first theorem. The substitution of f^* for f and the application of the Knaster-Tarski theorem within the set S are the same; the difference is that in the definition of the set S, $\equiv_*$ plays the role previously played by $\equiv_{\mathrm{wlp}}$. In order to avoid repeating the arguments that are common to both proofs, we will present them as a separate theorem that applies to any "acceptable" equivalence relation. Then Theorems 1 and 2 are proved by showing that $\equiv_{\mathrm{wlp}}$ and $\equiv_*$ are acceptable. This program is carried out in the next two sections.

6. A FIXED-POINT THEOREM FOR ACCEPTABLE RELATIONS

An operation f on commands *respects* an equivalence relation $\sim$ if for any commands A and B,

$$A \sim B \Rightarrow f.A \sim f.B\,.$$

An operation with more than one argument respects $\sim$ if it respects $\sim$ in each argument.

An equivalence relation $\sim$ on commands is *acceptable* if:

(A1) $[]$ respects $\sim$.

(A2) $A \bigtriangledown B \;\sim\; A \,[]\, B$ for all A, B.

(A3) $A \sim B$ implies $A \equiv_{\mathrm{wlp}} B$ for all A, B.

(A4) Join with respect to $\sqsubseteq$ preserves equivalence classes of $\sim$. That is, for any command B and nonempty family of commands A_i:

$$(\forall i :: A_i \sim B) \;\Rightarrow\; (\sqcup\, i :: A_i) \sim B\,.$$

LEMMA C. *If $\sim$ is acceptable, then $\bigtriangledown$ is $\sqsubseteq$-monotonic when the varying argument is restricted to any equivalence class of $\sim$. That is, for any commands A, B, and C:*

$$(A \sim B) \;\wedge\; (A \sqsubseteq B) \;\Rightarrow\; (A \bigtriangledown C) \sqsubseteq (B \bigtriangledown C)\,.$$

PROOF. Since $\sim$ is stronger than $\equiv_{\mathrm{wlp}}$, it suffices to show that dovetail is $\sqsubseteq$-monotonic when restricted to any equivalence class of $\equiv_{\mathrm{wlp}}$. The $\sqsubseteq_{\mathrm{wlp}}$ part of the proof is trivial; in fact $A \sim B$ implies that $A \bigtriangledown C$ and $B \bigtriangledown C$ are wlp-equivalent. Because of this fact and the pairing condition, the $\sqsubseteq_{\mathrm{wp}}$ part of the proof can be completed by showing that

$$(A \sqsubseteq B) \Rightarrow (\mathrm{hlt}.(A \bigtriangledown C) \Rightarrow \mathrm{hlt}.(B \bigtriangledown C))\,.$$

To prove this, assume $A \sqsubseteq B$, and compute

$\mathrm{hlt}.(A \bigtriangledown C)$

$\equiv$ {Definition of $\bigtriangledown$}

$(\mathrm{hlt}.A \vee \mathrm{hlt}.C) \wedge (\mathrm{grd}.A \vee \mathrm{hlt}.C) \wedge (\mathrm{hlt}.A \vee \mathrm{grd}.C)$

$\Rightarrow$ {$\mathrm{hlt}.A \wedge \mathrm{grd}.A \Rightarrow \mathrm{grd}.B$ was shown in Nelson [1989] to be a consequence of $A \sqsubseteq B$ (in the proof of continuity of $\boxtimes$)}

$(\mathrm{hlt}.A \vee \mathrm{hlt}.C) \wedge (\mathrm{grd}.B \vee \mathrm{hlt}.C) \wedge (\mathrm{hlt}.A \vee \mathrm{grd}.C)$

$\Rightarrow$ {$\mathrm{hlt}.A \Rightarrow \mathrm{hlt}.B$ is a consequence of $A \sqsubseteq B$}

$$
\begin{array}{ll}
& (\mathrm{hlt}.B \vee \mathrm{hlt}.C) \wedge (\mathrm{grd}.B \vee \mathrm{hlt}.C) \wedge (\mathrm{hlt}.B \vee \mathrm{grd}.C) \\
\equiv & \{\text{Definition of } \triangledown\} \\
& \mathrm{hlt}.(B \triangledown C) \quad \square
\end{array}
$$

If f is a function from commands to commands defined by an expression of the form $f.X = \mathcal{E}$, then by f^* we denote the function defined by $f^*.X = \mathcal{E}^*$, where $\mathcal{E}^*$ is $\mathcal{E}$ with all occurrences of $\triangledown$ replaced by $[]$.

LEMMA D. *If f is defined by an expression of the form $f.X = \mathcal{E}$, and if every operator in $\mathcal{E}$ respects the acceptable equivalence relation $\sim$, and if every operator occurring in $\mathcal{E}$ is $\sqsubseteq$-monotonic except for $\triangledown$, then for any commands A and B:*

(D1) $A \sim B \Rightarrow f^*.A \sim f.B$.

(D2) $A \sqsubseteq B \Rightarrow f^*.A \sqsubseteq f.B$.

(D3) $(A \sim B) \wedge (A \sqsubseteq B) \Rightarrow f.A \sqsubseteq f.B$.

PROOF. The three proofs are all straightforward inductions on the size of $\mathcal{E}$. In the base case, where f is the identity or a constant function, the three claims can be verified by inspection. Each of the three induction steps has two cases: the case where the outermost operator of $\mathcal{E}$ is $\triangledown$, in which $f.X \equiv g.X \triangledown h.X$, for two functions g and h defined by expressions smaller than $\mathcal{E}$; and the case where the outermost operator of $\mathcal{E}$ is not $\triangledown$, in which $f.X \equiv g.(h.X)$, where g is an operator other than $\triangledown$ and h is defined by an expression smaller than $\mathcal{E}$. Here are the proofs of the two cases for each of the three steps:

D1, $\triangledown$ case:

$$
\begin{array}{ll}
& f^*.A \sim f.B \\
\equiv & g^*.A \,[]\, h^*.A \sim g.B \triangledown h.B \\
\equiv & \{\sim \text{ is acceptable (A2) and transitive }\} \\
& g^*.A \,[]\, h^*.A \sim g.B \,[]\, h.B \\
\Leftarrow & \{\sim \text{ is acceptable (A1) }\} \\
& g^*.A \sim g.B \;\wedge\; h^*.A \sim h.B \\
\Leftarrow & \{\text{ induction }\} \\
& A \sim B
\end{array}
$$

D1, other case:

$$
\begin{array}{ll}
& f^*.A \sim f.B \\
\equiv & g.(h^*.A) \sim g.(h.B) \\
\Leftarrow & \{\ g \text{ respects } \sim \text{ by hypothesis }\} \\
& h^*.A \sim h.B \\
\Leftarrow & \{\text{ induction }\} \\
& A \sim B
\end{array}
$$

D2, $\triangledown$ case:

$$
\begin{array}{ll}
& f^*.A \sqsubseteq f.B \\
\equiv & g^*.A \,[]\, h^*.A \;\sqsubseteq\; g.B \triangledown h.B \\
\Leftarrow & \{\text{ Lemma B, transitivity of } \sqsubseteq\ \} \\
& g^*.A \,[]\, h^*.A \;\sqsubseteq\; g.B \,[]\, h.B \\
\Leftarrow & \{\ [] \text{ is } \sqsubseteq\text{-monotonic }\} \\
& g^*.A \sqsubseteq g.B \;\wedge\; h^*.A \sqsubseteq h.B
\end{array}
$$

$\Leftarrow$ { induction }
$A \sqsubseteq B$

D2, other case:

$f^*.A \sqsubseteq f.B$
$\equiv$ $g.(h^\gamma.A) \sqsubseteq g.(h.B)$
$\Leftarrow$ { g is $\sqsubseteq$-monotonic by hypothesis }
$h^*.A \sqsubseteq h.B$
$\Leftarrow$ { induction }
$A \sqsubseteq B$

D3, $\triangledown$ case:

$f.A \sqsubseteq f.B$
$\equiv$ $g.A \triangledown h.A \sqsubseteq g.B \triangledown h.B$
$\Leftarrow$ { $\sim$ is acceptable; Lemma C }
$g.A \sim g.B \wedge g.A \sqsubseteq g.B$
$\wedge$ $h.A \sim h.B \wedge h.A \sqsubseteq h.B$
$\Leftarrow$ { Every operator in $\mathcal{E}$ respects $\sim$, hence
by structural induction, h and g respect $\sim$ }
$g.A \sqsubseteq g.B \wedge h.A \sqsubseteq h.B \wedge A \sim B$
$\Leftarrow$ { induction }
$A \sqsubseteq B \wedge A \sim B$

D3, other case:

$f.A \sqsubseteq f.B$
$\equiv$ $g.(h.A) \sqsubseteq g.(h.B)$
$\Leftarrow$ { g is $\sqsubseteq$-monotonic by hypothesis }
$h.A \sqsubseteq h.B$
$\Leftarrow$ { induction }
$A \sqsubseteq B \wedge A \sim B$

This completes the proof of Lemma D. □

THEOREM 3. *Let f be a map from commands to commands defined by an expression of the form $f.X \equiv \mathcal{E}$. Let $\sim$ be an acceptable equivalence relation respected by each operation occurring in $\mathcal{E}$. Suppose that every operation occurring in $\mathcal{E}$ is $\sqsubseteq$-monotonic except for $\triangledown$. Then f has a least fixed point in the order $\leq$, defined by*

$$A \leq B \equiv (A \sqsubseteq_{\text{wlp}} B) \wedge ((A \sim B) \Rightarrow (A \sqsubseteq B)).$$

PROOF. The proof follows the outline sketched in the previous section. By Theorem 8 of Nelson [1989], f^* has a $\sqsubseteq$-least fixed point, say X^*. Let S be the set of all Y such that $X^* \sim Y$ and $X^* \sqsubseteq Y$. First, we show that f carries S into S:

$Y \in S$
$\equiv$ $(X^* \sim Y) \wedge (X^* \sqsubseteq Y)$
$\Rightarrow$ { Lemma D }
$(f^*.X^* \sim f.Y) \wedge (f^*.X^* \sqsubseteq f.Y)$
$\equiv$ { f^* fixes X^* }
$(X^* \sim f.Y) \wedge (X^* \sqsubseteq f.Y)$

$\equiv$ $f.Y \in S$

Second, we show that f has a $\sqsubseteq$-least fixed point on S, using the Knaster-Tarski Theorem. This theorem states that a function f has a $\sqsubseteq$-least fixed point if f is $\sqsubseteq$-monotonic on S and the $\sqsubseteq$-join of any $\sqsubseteq$-chain in S lies in S. By Lemma C, the restriction of f to S is $\sqsubseteq$-monotonic. By acceptability (A4), the join of any nonempty chain in S lies in S. By definition, S contains a $\sqsubseteq$-minimum element, namely X^*, and therefore the empty chain also has a join in S. Therefore the Knaster-Tarski Theorem applies, showing that f has a $\sqsubseteq$-least fixed point in S, which we will call X.

It remains to show that X is $\leq$-minimal among all fixed points of f. Let Y be a fixed point of f. To show $X \leq Y$, we must show that $X \sqsubseteq_{\text{wlp}} Y$ and that $X \sim Y$ implies $X \sqsubseteq Y$. As a stepping stone to these two goals, we first prove that $X^* \sqsubseteq Y$:

$X^* \sqsubseteq Y$
$\Leftarrow$ { By Knaster-Tarski, the least fixed point precedes every prepoint }
$f^*.Y \sqsubseteq Y$
$\equiv$ { Y is a fixed point of f }
$f^*.Y \sqsubseteq f.Y$
$\Leftarrow$ { Lemma D }
TRUE

Next we show that $X \sqsubseteq_{\text{wlp}} Y$:

$X \sqsubseteq_{\text{wlp}} Y$
$\Leftarrow$ { $X^* \sim X$, hence (A3) $X^* \equiv_{\text{wlp}} X$ }
$X^* \sqsubseteq_{\text{wlp}} Y$
$\Leftarrow$ $X^* \sqsubseteq Y$
$\equiv$ { Stepping stone }
TRUE

Finally we show that $X \sim Y$ implies $X \sqsubseteq Y$:

$X \sqsubseteq Y$
$\Leftarrow$ { X is the $\sqsubseteq$-least fixed point of f on S }
$Y \in S$
$\equiv$ $(X^* \sim Y) \wedge (X^* \sqsubseteq Y)$
$\equiv$ { Stepping stone }
$X^* \sim Y$
$\Leftarrow$ { $X \sim X^*$ }
$X \sim Y$

This completes the proof of Theorem 3. □

7. PROOFS OF THEOREMS 1 AND 2

In this section we deduce Theorems 1 and 2 from Theorem 3.

LEMMA E. *The equivalence relations* $\equiv_{\text{wlp}}$ *and* $\equiv_*$ *are acceptable.*

PROOF. We must verify conditions (A1)–(A4). Condition (A3) is immediate, since both $\equiv_{\text{wlp}}$ and $\equiv_*$ are as strong as $\equiv_{\text{wlp}}$. The other conditions will be verified for $\equiv_{\text{wlp}}$ and for $\equiv_{\text{grd}}$. This suffices to prove the lemma, since these conditions have the property that if they hold for two relations, then they also hold for the intersection of the two; and $\equiv_*$ is the intersection of $\equiv_{\text{wlp}}$ and $\equiv_{\text{grd}}$.

Condition (A1), that $[]$ respects $\equiv_{\mathrm{wlp}}$ and $\equiv_{\mathrm{grd}}$, follows from the wlp and guard equations for $[]$:

$$\mathrm{wlp}.(A \,[]\, B).R \equiv \mathrm{wlp}.A.R \wedge \mathrm{wlp}.B.R$$
$$\mathrm{grd}.(A \,[]\, B) \equiv \mathrm{grd}.A \vee \mathrm{grd}.B\,.$$

For example, the only occurrence of A on the right-hand side of the first equation is in $\mathrm{wlp}.A$; thus $\mathrm{wlp}.(A \,[]\, B).R$ depends on A only insofar as it depends on the $\equiv_{\mathrm{wlp}}$ equivalence class of A.

Condition (A2) is that $A \bigtriangledown B$ be equivalent to $A \,[]\, B$. For $\equiv_{\mathrm{wlp}}$, this follows because $[]$ and $\bigtriangledown$ have the same wlp-equation; for $\equiv_{\mathrm{grd}}$, this follows from Lemma A.

Condition (A4) is a consequence of the formula from Nelson [1989] for the precondition of the join of a chain that was presented in Section 2. Let A_i be a non-empty family of commands. If $A_i \equiv_{\mathrm{wlp}} B$ for all i, then

$$\begin{aligned}
&\mathrm{wlp}.(\sqcup\, i :: A_i).R \\
\equiv\ &(\wedge\, i :: \mathrm{wlp}.A_i.R) \\
\equiv\ &(\wedge\, i :: \mathrm{wlp}.B.R) \\
\equiv\ &\mathrm{wlp}.B.R\,.
\end{aligned}$$

If $A_i \equiv_{\mathrm{grd}} B$ for all i, then

$$\begin{aligned}
&\mathrm{grd}.(\sqcup\, i :: A_i) \\
\equiv\ &\neg\, \mathrm{wp}.(\sqcup\, i :: A_i).\textsc{false} \\
\equiv\ &\neg\, (\vee\, i :: \mathrm{wp}.A_i.\textsc{false}) \\
\equiv\ &\neg\, (\vee\, i :: \neg\, \mathrm{grd}.A_i) \\
\equiv\ &\neg\, (\vee\, i :: \neg\, \mathrm{grd}.B) \\
\equiv\ &\mathrm{grd}.B\,.
\end{aligned}$$

This completes the proof of Lemma E. □

PROOF OF THEOREM 1. Inspection of the wlp-equations for the five operators

$$[] \quad \rightarrow \quad ; \quad [\,|\,] \quad \bigtriangledown$$

shows that these operators respect $\equiv_{\mathrm{wlp}}$. Theorem 1 therefore follows from Theorem 3 and Lemma E. □

PROOF OF THEOREM 2. Simple calculations, which will be left to the reader, show that

$$\mathrm{wlp}.(A;;B).R \equiv \mathrm{wlp}.A.(\mathrm{wlp}.B.R)$$
$$\mathrm{grd}.(A;;B) \equiv \mathrm{grd}.A\,.$$

Thus ;; respects $\equiv_{\mathrm{wlp}}$ and $\equiv_{\mathrm{grd}}$, and therefore also respects $\equiv_*$. Inspection of the wlp and guard equations for the four operators

$$[] \quad \rightarrow \quad [\,|\,] \quad \bigtriangledown$$

shows that these operators respect $\equiv_{\mathrm{wlp}}$ and $\equiv_{\mathrm{grd}}$, and therefore also $\equiv_+$.

To prove that $\boxtimes$ respects $\equiv_*$, assume that $A \equiv_* A'$ and $B \equiv_* B'$, and compute:

$$\begin{aligned}
&\mathrm{wlp}.(A \boxtimes B).R \\
\equiv\ &\mathrm{wlp}.A.R \wedge (\mathrm{grd}.A \vee \mathrm{wlp}.B.R) \\
\equiv\ &\mathrm{wlp}.A'.R \wedge (\mathrm{grd}.A' \vee \mathrm{wlp}.B'.R) \\
\equiv\ &\mathrm{wlp}.(A' \boxtimes B').R
\end{aligned}$$

$$
\begin{aligned}
& \mathrm{grd}.(A \boxtimes B) \\
\equiv\ & \mathrm{grd}.A \vee \mathrm{grd}.B \\
\equiv\ & \mathrm{grd}.A' \vee \mathrm{grd}.B' \\
\equiv\ & \mathrm{grd}.(A' \boxtimes B') .
\end{aligned}
$$

Theorem 2 therefore follows from Theorem 3 and Lemma E. □

Notice that $\boxtimes$ does not respect $\equiv_{\mathrm{wlp}}$ in its first argument, and ; does not respect $\equiv_{\mathrm{grd}}$ in its first argument. Thus Theorem 3, which is our only tool for constructing fixed points involving $\triangledown$, cannot accommodate $\boxtimes$ and ; simultaneously. Thus any two of the three operators $\boxtimes$; $\triangledown$ can be handled together, but not all three.

In fact, using all three operators, there are recursions involving dovetail that have no solutions. Consider

$$f.X \ \equiv\ [\![b \mid ((X \boxtimes b := 0) \triangledown b := 1) \,;\, (b = 0 \rightarrow \mathit{Loop})]\!] .$$

If X is defined by this recursion, and recursion is implemented by the usual unfolding, then X will be equivalent operationally to *Loop*. The computation tree for X branches at $\triangledown$ at each level of recursion. Each $b := 1$ branch leads to a $b = 0$ guard, where it fails. The other branch leads to a recursive call. Thus the computation will search an infinite tree, failing to find any proper outcomes.

But *Loop* is not a fixed point of f. Since $\mathit{Loop} \boxtimes b := 0$ is equal to *Loop*, and $\mathit{Loop} \triangledown b := 1$ is equal to $b := 1$, direct computation yields $f.\mathit{Loop} = \mathit{Fail}$.

In fact, f has no fixed point. The commands that f operates on are commands on a zero-dimensional state space (that is, a point). There are only four such commands: *Loop*, *Fail*, *Skip*, and $\mathit{Skip} \,[]\, \mathit{Loop}$. (On a zero-dimensional state space *Skip* and *Havoc* coincide.) Computation yields:

$$
\begin{aligned}
& f.\mathit{Loop} = \mathit{Fail} \\
& f.\mathit{Fail} = \mathit{Loop} \\
& f.\mathit{Skip} = \mathit{Loop} \\
& f.(\mathit{Skip} \,[]\, \mathit{Loop}) = \mathit{Loop} .
\end{aligned}
$$

In fact it is not difficult to prove that f has no fixed point at all, even among commands with nontrivial state spaces.

We believe that if dovetail is given an operational semantics as sketched in the introduction, and if recursive calls are implemented in the standard way, then the resulting system will agree with the axiomatic semantics for all recursions for which fixed points exist. Where an axiomatic fixed point does not exist, the operational semantics will compute a fixed point that does not agree with the axiomatic semantics. For example, in a simple operational semantics that always unfolded any recursive call, the recursively defined X where $X = f.X$ would be equivalent to *Loop*. In a more sophisticated implementation that optimized the computation of $A \triangledown B$ by abandoning the computation of A whenever the computation was stuck in a detectable loop, X would be equivalent to *Fail*.

8. CONCLUSIONS

The formal treatment of dovetail is somewhat curious: a function f is proved to have a least fixed point with respect to an order $\leq$, although f is not monotonic with respect to $\leq$. The proof is based on an order $\sqsubseteq$, with respect to which the function does not have a least fixed point.

It is obvious that the verification of programs that include dovetail can be based on the theorems, provided that any recursions in the program obey the restrictions. In particular, Theorem 1 justifies the recursive definition of **Seq** used in **fairdo**, making it possible to compute the weakest preconditions of Unity programs.

Dovetail could be of practical importance in studying classes of implementations of loop-avoiding operators. For example, it could be used to model loop-avoiding communication when merging several communication lines.

REFERENCES

APT, K. R. AND PLOTKIN, G. D. 1986. Countable nondeterminism and random assignment *J. ACM, 33*, 4, 724–767.

BOOM, H. J. 1982. A weaker precondition for loops *Trans Program. Lang Syst. 4*, 4.

BROY, M. 1986. A theory for nondeterminism, parallelism, communication, and concurrency. *Theoret. Comput Sci. 45*, 1–61

CHANDY, K. M. AND MISRA, J 1988. *Parallel Program Design: A Foundation.* Addison–Wesley, Reading, Mass.

DIJKSTRA, E. W. 1976. *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, New Jersey.

DIJKSTRA, E. W. AND SCHOLTEN, C. S. 1990. *Predicate Calculus and Program Semantics.* Springer-Verlag, New York.

MCCARTHY, J. 1963. Towards a mathematical science of computation. In *Computer Programming and Formal Systems*, P. Braffort and D. Hirschberg, Eds., North-Holland, Amsterdam, 33–70.

NELSON, G. 1989. A generalization of Dijkstra's calculus. *Trans. Program Lang. Syst. 11*, 4, 517–61. Also available as Res. Rep. SRC-16, Digital Equipment Corp., Palo Alto, 1987