38

Can fair choice be added to Dijkstra's calculus?

by Manfred Broy and Greg Nelson

February 16, 1989

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

Systems Research Center

DEC's business and technology objectives require a strong research program. The Systems Research Center (SRC) and three other research laboratories are committed to filling that need.

SRC began recruiting its first research scientists in 1984 — their charter, to advance the state of knowledge in all aspects of computer systems research. Our current work includes exploring high-performance personal computing, distributed computing, programming environments, system modelling techniques, specification technology, and tightly-coupled multiprocessors.

Our approach to both hardware and software research is to create and use real systems so that we can investigate their properties fully. Complex systems cannot be evaluated solely in the abstract. Based on this belief, our strategy is to demonstrate the technical and practical feasibility of our ideas by building prototypes and using them as daily tools. The experience we gain is useful in the short term in enabling us to refine our designs, and invaluable in the long term in helping us to advance the state of knowledge about those systems. Most of the major advances in information systems have come through this strategy, including time-sharing, the ArpaNet, and distributed personal computing.

SRC also performs work of a more mathematical flavor which complements our systems research. Some of this work is in established fields of theoretical computer science, such as the analysis of algorithms, computational geometry, and logics of programming. The rest of this work explores new ground motivated by problems that arise in our systems research.

DEC has a strong commitment to communicating the results and experience gained through pursuing these activities. The Company values the improved understanding that comes with exposing and testing our ideas within the research community. SRC will therefore report results in conferences, in professional journals, and in our research report series. We will seek users for our prototype systems among those with whom we have common research interests, and we will encourage collaboration with university researchers.

Robert W. Taylor, Director

Can fair choice be added to Dijkstra's calculus?

Manfred Broy and Greg Nelson

February 16, 1989

Manfred Broy is a faculty member at the University of Passau, Residenzplatz 8, 8390 Passau, Federal Republic of Germany.

This paper will also appear as a research report to be published by the Fakultät für Mathematik und Informatik, MIP 8902, Universität Passau.

Authors' abstract

The paper studies the incorporation of a fair nondeterministic choice operator into a generalization of Dijkstra's calculus of guarded commands. The new operator is not monotonic for the orderings that are generally used for proving the existence of least fixpoints for recursive definitions. To prove the existence of a fixpoint it is necessary to consider several orderings at once, and to restrict the class of recursive definitions.

Manfred Broy and Greg Nelson

Perspective

Over a decade ago, Dijkstra developed a weakest-precondition calculus for reasoning about programs written in a simple language of guarded commands. The basis of the calculus is the idea that the meaning of a statement S is a predicate transformer $wp(S, .)$, where $wp(S, P)$ asserts what must be true initially if an execution of S has to terminate with P true.

Dijkstra's language was distinguished from conventional toy languages by the fundamental position it accorded to nondeterministic operations. It may also have been the first language proposed since BASIC that did not include recursive procedures, which were omitted because the weakest-precondition calculus could not handle recursive programs.

In SRC Research Report 16, Greg Nelson extended Dijkstra's calculus to general recursive programs. In the spirit of denotational semantics, the meaning of a recursive program is defined by a least fixpoint. A key element of his approach was extending the language to allow partial commands—ones that could fail and cause backtracking. This allowed him to give meanings to individual components of a statement, and to derive the meaning of a complete statement from the meanings of its components. For example, the [] of Dijkstra's language becomes an operator for composing partial commands, where $A \,[]\, B$ means execute either A or B.

The present report generalizes the prior results to a further extension of Dijkstra's language, obtained by adding the "dovetail" operator $\bigtriangledown$. Intuitively, $A \bigtriangledown B$ means execute both A and B, independently, and take the result produced by either of them that terminates. Thus, the results that can be produced by executing $A \,[]\, B$ and $A \bigtriangledown B$ are the same. However, $A \,[]\, B$ may fail to terminate—not producing any result—if either A or B fails to terminate, while $A \bigtriangledown B$ must terminate if either A or B does.

The dovetail operator introduces fairness into the language. The command $A \bigtriangledown B$ can be implemented by running A and B in parallel, with a fair scheduler, and taking the result of whichever finishes first. Fairness traditionally means trouble. The dovetail operator, together with recursion, provides unbounded nondeterminism—for example, the ability to write a terminating, nondeterministic statement that can set x to any integer. Dijkstra's original calculus is unsound for any language with unbounded nondeterminism. Fairness and unbounded nondeterminism also wreak havoc with denotational methods, since they lead to discontinuity.

This report uses least fixpoints to define the meaning of commands in the extended language, which includes recursion and the dovetail operator. The price of handling fairness is the possibility that a recursive equation purporting to define a command does not have a solution. It appears that computer scientists, like mathematicians, must face the existence of equations that have no solutions.

Leslie Lamport

Contents

1. Introduction

The fixpoint method of denotational semantics is so successful that it is a surprise to find a programming language construct for which it doesn't seem to work. But this is the case for the dovetail operator, which, although it is not popular among programming language designers, is of theoretical, and maybe also practical, importance. The operational definition of dovetail (which we shall write $\triangledown$) is as follows:

$A \triangledown B$

$\equiv$ Execute the commands A and B in parallel, on separate copies of the state, interleaving the two computations non-deterministically but fairly, accepting as an outcome any proper (i.e., non-looping) outcome of either A or B.

By "fairly" it is meant that neither computation is starved: if the computation is infinite, then each of the A and B parts are either infinite, or else run to completion without producing an outcome, as can happen in the case of partial commands. Partial commands are those that do not satisfy the Law of the Excluded Miracle: viewed as relations, they are partial; viewed operationally, they may "fail"—that is, backtrack.

As a hint at the power of the dovetail operator, we show how it immediately leads to unbounded nondeterminism. Operationally, a recursive call can be treated by replacing the call with the righthand side of the recursive definition whenever necessary. This makes it obvious that the recursion

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

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 $\triangledown$ 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's paper [0]. But the dovetail operator is more of a challenge.

The dovetail operator is the imperative counterpart of the ambiguity operator introduced by McCarthy in 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" [2]. The ambiguity operator is not monotonic in the orderings of either the Smyth or the Plotkin powerdomains. Therefore its fixpoint theory, presented by Broy in 1986, is far from straightforward [1]. The dovetail operation also is not monotonic, and to treat it by the fixpoint method requires some of Broy's techniques. But the presence of partial commands introduces additional difficulties. In fact, in the axiomatic definitions that we propose, not all recursions involving dovetail have solutions.

2. Preliminaries

Our framework is the generalization of Dijkstra's calculus described by Nelson in SRC-16 [3], which will be briefly described 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 non-empty 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

$$A \sqsubseteq_{\text{wp}} B\ :\quad \text{wp}.A.R \Rightarrow \text{wp}.B.R \quad \text{for any } R$$
$$A \sqsubseteq_{\text{wlp}} B\ :\quad \text{wlp}.B.R \Rightarrow \text{wlp}.A.R \quad \text{for any } R$$
$$A \sqsubseteq B\ :\quad A \sqsubseteq_{\text{wp}} B \text{ and } A \sqsubseteq_{\text{wlp}} B$$

The relation $A \sqsubseteq B$ is read *A approximates B*; it is a complete partial order on the set of all commands. Operationally, 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 *Loop* approximates every command, and a command with no looping outcomes approximates no command except itself.

Notice that we have inverted the definition of $\sqsubseteq_{\text{wlp}}$ given in SRC-16; this change makes formulas more readable.

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 SRC-16, 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:

$$\text{w(l)p}.\mathit{Fail}.R \equiv \text{TRUE}$$
$$\text{w(l)p}.\mathit{Skip}.R \equiv R$$

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

Here we write $[R]$ for $R \equiv \text{TRUE}$.

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

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

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:

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

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

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 $\text{grd}.(A \bigtriangledown B) \equiv \text{grd}.A \vee \text{grd}.B$.

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,

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

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

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

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

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

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 non-looping 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 $\bigtriangledown$ 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 $\text{hlt}.(A \,[]\, B) \Rightarrow \text{hlt}.(A \bigtriangledown B)$. The proof of this is:

$$
\begin{aligned}
& \text{hlt}.(A \,[]\, B) \\
\equiv\ & \text{hlt}.A \wedge \text{hlt}.B \\
\Rightarrow\ & (\text{hlt}.A \vee \text{hlt}.B) \wedge (\text{grd}.A \vee \text{hlt}.B) \wedge (\text{grd}.B \vee \text{hlt}.A) \\
\equiv\ & \text{hlt}.(A \bigtriangledown B) \qquad \blacksquare
\end{aligned}
$$

4. Nonmonotonicity of dovetail

The reason that the classical fixpoint method doesn't work for dovetail is that dovetail is not monotonic with respect to the approximation relation. For example,

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

but

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

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

In fact, under our definitions, there are recursions involving dovetail that have no solutions. Consider

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

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 $\bigtriangledown$ 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 fixpoint of f. Since $\mathit{Loop} \,[]\, b := 0$ is equal to *Loop*, and $\mathit{Loop} \bigtriangledown b := 1$ is equal to $b := 1$, direct computation yields $f.\mathit{Loop} = \mathit{Fail}$.

In fact, f has no fixpoint. 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 *Skip* $\Box$ *Loop*. (On a zero-dimensional state space, *Skip* and *Havoc* coincide.) Computation yields:

$f.Loop = Fail$
$f.Fail = Loop$
$f.Skip = Loop$
$f.(Skip \Box Loop) = Loop$

5. Two fixpoint theorems for dovetail

If operational semantics is the touchstone by which other semantics are judged, then the example above is a serious blow to our semantic definitions, one that strongly suggests changing the axiomatic semantics to agree with some operational semantics. But there is another approach, which we will explore in the remainder of this paper: we take the axiomatic semantics as the touchstone by which other semantics are judged. The axiomatic semantics are in effect a specification language; perhaps too rich to be handled in its entirety by an operational implementation, but well defined nonetheless.

This is as much a question of technique as of philosophy. If the nineteenth century mathematicians who made sense of infinite series had known denotational semantics, they might have added $\perp$ to the real numbers R (as well as one new open set $R \cup \{\perp\}$ to R's topology) after which they would have been able to prove that every series has a unique "least convergence point" in the inverse Scott order determined by the topology. For example, the least convergence point of $1 + 1 - 1 + 1 \ldots$ is $\perp$; the least convergence point of $1/2 + 1/4 + 1/8 \ldots$ is 1. But instead, these mathematicians worked with the unextended real numbers, in which not all series converge. The choice is not too serious, since with either technique, the important thing is to find tests that guarantee various kinds of convergence.

Our approach will be to find restricted classes of recursions that are guaranteed to have solutions in our calculus, analogous to restricted classes of infinite series that are guaranteed to converge. In SRC-16 it was proved by the simple fixpoint method that if ∇ is excluded, then all recursions have solutions. In this paper, we prove two results that include dovetail: (1) if $\boxtimes$ is excluded, then all recursions have solutions, and (2) if semicolon is restricted so that its second argument is always total, then all recursions have solutions.

Neither of these results seem to be provable by the simple fixpoint method. This section outlines an alternative method of proof that works for both results.

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

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 fixpoint in the order $\leq$, defined by:

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

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

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 $[]$. Thereom 8 of SRC-16 shows that f^* has a fixpoint, 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 fixpoint 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 fixpoint 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 fixpoint 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 fixpoint, namely *Skip*. (In fact, *Skip* is the unique fixpoint on this set.)

Notice that the fixpoint is not approximation-minimal: for example, *Havoc* is a fixpoint of (1), but *Skip* does not approximate *Havoc*. Indeed, the set of fixpoints 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 $\leq$ and $\sqsubseteq$ 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 fixpoint construction to locate the $\equiv_{\mathrm{wlp}}$ equivalence class of the fixpoint (thus determining its set of proper outcomes) and then uses a second fixpoint 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

$$A;;B \;\equiv\; A\,;(B \,[\!]\, \mathit{Loop})\,.$$

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

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

as well as the command parameter X and any number of fixed commands and predicates. Then f has a least fixpoint 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 fixpoint 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 non-empty 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

$$
\begin{array}{ll}
 & \mathrm{hlt}.(A \bigtriangledown C) \\
\equiv & \{\text{Definition of } \bigtriangledown\} \\
 & (\mathrm{hlt}.A \vee \mathrm{hlt}.C) \wedge (\mathrm{grd}.A \vee \mathrm{hlt}.C) \wedge (\mathrm{hlt}.A \vee \mathrm{grd}.C) \\
\Rightarrow & \{\mathrm{hlt}.A \wedge \mathrm{grd}.A \Rightarrow \mathrm{grd}.B \text{ was shown in SRC-16 to be a consequence of} \\
 & A \sqsubseteq B \text{ (in the proof of continuity of } \boxtimes)\} \\
 & (\mathrm{hlt}.A \vee \mathrm{hlt}.C) \wedge (\mathrm{grd}.B \vee \mathrm{hlt}.C) \wedge (\mathrm{hlt}.A \vee \mathrm{grd}.C) \\
\Rightarrow & \{\mathrm{hlt}.A \Rightarrow \mathrm{hlt}.B \text{ is a consequence of } A \sqsubseteq B\} \\
 & (\mathrm{hlt}.B \vee \mathrm{hlt}.C) \wedge (\mathrm{grd}.B \vee \mathrm{hlt}.C) \wedge (\mathrm{hlt}.B \vee \mathrm{grd}.C) \\
\equiv & \{\text{Definition of } \bigtriangledown\} \\
 & \mathrm{hlt}.(B \bigtriangledown C) \quad \blacksquare
\end{array}
$$

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 $\bigtriangledown$ 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 occuring in $\mathcal{E}$ is $\sqsubseteq$-monotonic except for $\bigtriangledown$, 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 $\bigtriangledown$, in which $f.X \equiv g.X \bigtriangledown h.X$, 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 $\bigtriangledown$, in which $f.X \equiv g.(h.X)$, where g is an operator other than $\bigtriangledown$ 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, $\bigtriangledown$ case:

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

$g^*.A \,[]\, h^*.A \sim g.B \,[]\, h.B$
$\Leftarrow$ { $\sim$ is acceptable (A1) }
$g^*.A \sim g.B \;\wedge\; h^*.A \sim h.B$
$\Leftarrow$ { induction }
$A \sim B$

D1, other case:

$f^*.A \sim f.B$
$\equiv$ $g.(h^*.A) \sim g.(h.B)$
$\Leftarrow$ { g respects $\sim$ by hypothesis }
$h^*.A \sim h.B$
$\Leftarrow$ { induction }
$A \sim B$

D2, ∇ case:

$f^*.A \sqsubseteq f.B$
$\equiv$ $g^*.A \,[]\, h^*.A \;\sqsubseteq\; g.B \,\nabla\, h.B$
$\Leftarrow$ { Lemma B, transitivity of $\sqsubseteq$ }
$g^*.A \,[]\, h^*.A \;\sqsubseteq\; g.B \,[]\, h.B$
$\Leftarrow$ { $[]$ is $\sqsubseteq$-monotonic }
$g^*.A \sqsubseteq g.B \;\wedge\; h^*.A \sqsubseteq h.B$
$\Leftarrow$ { induction }
$A \sim B$

D2, 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 }
$A \sqsubseteq B$

D3, ∇ case:

$f.A \sqsubseteq f.B$
$\equiv$ $g.A \,\nabla\, h.A \;\sqsubseteq\; g.B \,\nabla\, h.B$
$\Leftarrow$ { $\sim$ 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 $\sim$ }

$$
\begin{array}{ll}
 & g.A \sqsubseteq g.B \ \wedge \ h.A \sqsubseteq h.B \ \wedge \ A \sim B \\
\Leftarrow & \{ \text{ induction } \} \\
 & A \sqsubseteq B \ \wedge \ A \sim B
\end{array}
$$

D3, other case

$$
\begin{array}{ll}
 & f.A \sqsubseteq f.B \\
\equiv & g.(h.A) \sqsubseteq g.(h.B) \\
\Leftarrow & \{ \ g \text{ is } \sqsubseteq\text{-monotonic by hypothesis } \} \\
 & h.A \sqsubseteq h.B \\
\Leftarrow & \{ \text{ induction } \} \\
 & A \sqsubseteq B \ \wedge \ A \sim B
\end{array}
$$

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 ∇. Then f has a least fixpoint in the order $\leq$, defined by

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

Proof. The proof follows the outline sketched in the previous section. By Theorem 8 of SRC-16, f^* has a $\sqsubseteq$-least fixpoint, 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:

$$
\begin{array}{ll}
 & Y \in S \\
\equiv & (X^* \sim Y) \wedge (X^* \sqsubseteq Y) \\
\Rightarrow & \{ \text{ Lemma D } \} \\
 & (f^*.X^* \sim f.Y) \wedge (f^*.X^* \sqsubseteq f.Y) \\
\equiv & \{ \ f^* \text{ fixes } X^* \ \} \\
 & (X^* \sim f.Y) \wedge (X^* \sqsubseteq f.Y) \\
\equiv & f.Y \in S
\end{array}
$$

Second, we show that f has a $\sqsubseteq$-least fixpoint on S, using the Knaster-Tarski Theorem. This theorem requires that f be $\sqsubseteq$-monotonic on S and that the $\sqsubseteq$-join of any $\sqsubseteq$-chain in S lie in S. By Lemma C, the restriction of f to S is $\sqsubseteq$-monotonic. By acceptability (A4), the join of any non-empty 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 fixpoint in S, which we will call X.

It remains to show that X is $\leq$-minimal among all fixpoints of f. Let Y be a fixpoint 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 fixpoint precedes every prepoint }
$f^*.Y \sqsubseteq Y$
$\equiv$ { Y is a fixpoint 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 fixpoint 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 $\equiv_{\text{wlp}}$ and $\equiv_{\text{grd}}$, follows from the wlp and guard equations for $[]$:

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

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

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

Condition (A4) is a consequence of the formula from SRC-16 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 $A_i \equiv_{\text{wlp}} B$ for all i, then

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

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

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

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 $\equiv_{\text{wlp}}$. 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

$$\begin{aligned}&\text{wlp}.(A;;B).R \equiv \text{wlp}.A.(\text{wlp}.B.R)\\&\text{grd}.(A;;B) \equiv \text{grd}.A\end{aligned}$$

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

$$[] \quad \rightarrow \quad [\![\,|\,|\,]\!] \quad \nabla$$

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

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

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

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

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

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

8. Conclusions

The formal treatment of dovetail is somewhat curious: a function f is proved to have a least fixpoint 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 fixpoint.

It is obvious that proof techniques can be based on the given construction.

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 communcation lines.

References

[0] H. J. Boom. A weaker precondition for loops. *TOPLAS* 4, October 1982.

[1] M. Broy. A theory for nondeterminism, parallelism, communication, and concurrency. *TCS* 45, 1986, pp. 1–61.

[2] J. McCarthy. Towards a mathematical science of computation. In P. Braffort, D. Hirschberg (eds.): *Computer Programming and Formal Systems*, Amsterdam, North-Holland 1963, pp. 33–70.

[3] Greg Nelson. A generalization of Dijkstra's calculus. Research report SRC-16, Digital Equipment Corp., Palo Alto, 1987. Submitted to *TOPLAS*.

SRC Reports

"A Kernel Language for Modules and Abstract Data Types."
R. Burstall and B. Lampson.
Research Report 1, September 1, 1984.

"Optimal Point Location in a Monotone Subdivision."
Herbert Edelsbrunner, Leo J. Guibas, and Jorge Stolfi.
Research Report 2, October 25, 1984.

"On Extending Modula-2 for Building Large, Integrated Systems."
Paul Rovner, Roy Levin, John Wick.
Research Report 3, January 11, 1985.

"Eliminating go to's while Preserving Program Structure."
Lyle Ramshaw.
Research Report 4, July 15, 1985.

"Larch in Five Easy Pieces."
J. V. Guttag, J. J. Horning, and J. M. Wing.
Research Report 5, July 24, 1985.

"A Caching File System for a Programmer's Workstation."
Michael D. Schroeder, David K. Gifford, and Roger M. Needham.
Research Report 6, October 19, 1985.

"A Fast Mutual Exclusion Algorithm."
Leslie Lamport.
Research Report 7, November 14, 1985.

"On Interprocess Communication."
Leslie Lamport.
Research Report 8, December 25, 1985.

"Topologically Sweeping an Arrangement."
Herbert Edelsbrunner and Leonidas J. Guibas.
Research Report 9, April 1, 1986.

"A Polymorphic λ-calculus with Type:Type."
Luca Cardelli.
Research Report 10, May 1, 1986.

"Control Predicates Are Better Than Dummy Variables For Reasoning About Program Control."
Leslie Lamport.
Research Report 11, May 5, 1986.

"Fractional Cascading."
Bernard Chazelle and Leonidas J. Guibas.
Research Report 12, June 23, 1986.

"Retiming Synchronous Circuitry."
Charles E. Leiserson and James B. Saxe.
Research Report 13, August 20, 1986.

"An $O(n^2)$ Shortest Path Algorithm for a Non-Rotating Convex Body."
John Hershberger and Leonidas J. Guibas.
Research Report 14, November 27, 1986.

"A Simple Approach to Specifying Concurrent Systems."
Leslie Lamport.
Research Report 15, December 25, 1986. Revised January 26, 1988

"A Generalization of Dijkstra's Calculus."
Greg Nelson.
Research Report 16, April 2, 1987.

"*win* and *sin*: Predicate Transformers for Concurrency."
Leslie Lamport.
Research Report 17, May 1, 1987.

"Synchronising Time Servers."
Leslie Lamport.
Research Report 18, June 1, 1987.

"Blossoming: A Connect-the-Dots Approach to Splines."
Lyle Ramshaw.
Research Report 19, June 21, 1987.

"Synchronization Primitives for a Multiprocessor: A Formal Specification."
A. D. Birrell, J. V. Guttag, J. J. Horning, R. Levin.
Research Report 20, August 20, 1987.

"Evolving the UNIX System Interface to Support Multithreaded Programs."
Paul R. McJones and Garret F. Swart.
Research Report 21, September 28, 1987.

"Building User Interfaces by Direct Manipulation."
Luca Cardelli.
Research Report 22, October 2, 1987.

"Firefly: A Multiprocessor Workstation."
C. P. Thacker, L. C. Stewart, and E. H. Satterthwaite, Jr.
Research Report 23, December 30, 1987.

"A Simple and Efficient Implementation for Small Databases."
Andrew D. Birrell, Michael B. Jones, and Edward P. Wobber.
Research Report 24, January 30, 1988.

"Real-time Concurrent Collection on Stock Multiprocessors."
John R. Ellis, Kai Li, and Andrew W. Appel.
Research Report 25, February 14, 1988.

"Parallel Compilation on a Tightly Coupled Multiprocessor."
Mark Thierry Vandevoorde.
Research Report 26, March 1, 1988.

"Concurrent Reading and Writing of Clocks."
Leslie Lamport.
Research Report 27, April 1, 1988.

"A Theorem on Atomicity in Distributed Algorithms."
Leslie Lamport.
Research Report 28, May 1, 1988.

"The Existence of Refinement Mappings."
Martín Abadi and Leslie Lamport.
Research Report 29, August 14, 1988.

"The Power of Temporal Proofs."
Martín Abadi.
Research Report 30, August 15, 1988.

"Modula-3 Report."
Luca Cardelli, James Donahue, Lucille Glassman, Mick Jordan, Bill Kalsow, Greg Nelson.
Research Report 31, August 25, 1988.

"Bounds on the Cover Time."
Andrei Broder and Anna Karlin.
Research Report 32, October 15, 1988.

"A Two-view Document Editor with User-definable Document Structure."
Kenneth Brooks.
Research Report 33, November 1, 1988.

"Blossoms are Polar Forms."
Lyle Ramshaw.
Research Report 34, January 2, 1989.

"An Introduction to Programming with Threads."
Andrew Birrell.
Research Report 35, January 6, 1989.

"Primitives for Computational Geometry."
Jorge Stolfi.
Research Report 36, January 27, 1989.

"Ruler, Compass, and Computer: The Design and Analysis of Geometric Algorithms."
Leonidas J. Guibas and Jorge Stolfi.
Research Report 37, February 14, 1989.

SRC Report 30
by Manfred Broy and Greg Nelson

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301