A COMPARISON OF *LED-FROM* AND *LEADS-TO*

Edgar Knapp

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-88-35 October 1988

Abstract

Progress properties of parallel programs are often expressed using the operator *leads-to*, a binary relation over predicates on the program state. Informally, p *leads-to* q means that from a state satisfying p a state satisfying q is reached eventually. We investigate the notion of the weakest predicate that *leads-to* q, for some given q. We formalize this notion by defining a predicate transformer called *led-from* and show that *led-from* maps a predicate q to the weakest predicate that *leads-to* q. We also demonstrate that *led-from* and *leads-to* are equivalent in expressive power by showing that each can be defined in terms of the other. The advantages of basing concurrent program semantics on the predicate transformer *led-from* rather than the relation *leads-to* are similar to those of basing sequential program semantics on the predicate transformer **wp** rather than on the relation of Hoare-Triples $\{p\}S\{q\}$. In particular, questions about junctivity properties can be raised and answered. Among other things we show that *led-from* is monotonic and idempotent, but not **or**-continuous and neither finitely disjunctive nor finitely conjunctive.

Contents

Introduction

When specifying concurrent systems it is necessary to formulate requirements that "something good is going to happen eventually", where eventually means "in finite time" or "in a finite number of steps". These kinds of requirements are traditionally called *progress* conditions. Examples of such requirements are: a requested resource is granted eventually; a message sent along a channel is received eventually; deadlock is detected eventually.

In the theory of UNITY [Chandy and Misra 1988], which attempts a unified approach to the specification and verification of concurrent programs, properties like the ones above are expressed using the binary operator *leads–to* ($\mapsto$), a relation over predicates on program states. This operator was first introduced in [Lamport 1977]. Informally, $p \mapsto q$ means that from a state satisfying predicate p a state satisfying predicate q is reached eventually. For example, the property $true \mapsto p$ specifies that a state satisfying p occurs infinitely often.

In [Chandy and Misra 1988] $\mapsto$ is defined as the strongest relation that satisfies a set of axioms. These axioms formalize the requirements that the $\mapsto$ relation is a superset of some fixed relation called **ensures**, and that it is transitively and disjunctively closed. The **ensures** relation itself depends only on the program text and also captures fairness. Our first result formally verifies that the definition of $\mapsto$ given in [Chandy and Misra 1988] is sound by showing that this strongest relation exists and is unique.

The main contribution of this paper is to investigate an alternative way of defining $\mapsto$ using a predicate transformer (i.e. a function from predicates to predicates) which we call *led-from*. This predicate transformer, too, can be defined as an extreme solution of a set of equations. The intuitive meaning of *led-from* is that it maps a predicate p to the *weakest predicate that leads-to* p. We show that this alternative definition of $\mapsto$ is equivalent to the definition that appears in [Chandy and Misra 1988].

The advantage of using *led-from* as the basis for the definition of $\mapsto$ is that *led-from* is a *function* on predicates. Therefore, questions about monotonicity, continuity, etc. can be easily posed for *led-from*. To draw an analogy with sequential program verification, the advantages of *led-from* over $\mapsto$ are similar to the advantages of basing program semantics on the predicate transformer **wp** rather than on the relation of Hoare-Triples $\{p\}\ S\ \{q\}$. It turns out that *led-from* lacks many of the nice properties of **wp**. In particular, we show that for UNITY logic, *led-from* is not **or**-continuous, and neither finitely conjunctive nor finitely disjunctive.

Our paper is organized as follows. Section 1 introduces our notation and gives formal definitions of $\mapsto$ and *led-from* as extreme solutions of certain sets of equations. It is proved that these solutions uniquely define both operators. Section 2 is devoted to the proof that all information about $\mapsto$ is captured within *led-from*. More precisely, we show that each concept can be defined in terms of the other. In Section 3, we study interesting properties of *led-from*. We derive negative answers to a number of questions about junctivity properties. The paper concludes with a discussion of our results.

1 Definition of *leads-to* and *led-from*

First we present the notation necessary to express our concepts. We then introduce the notion of an extreme solution to a set of equations in order to define the operators *leads-to* and *led-from*.

1.1 Notation

We will use the following notational conventions: the expresssion

$$\langle \mathbf{Q}\, x : r.x : t.x \rangle,$$

where $\mathbf{Q} \in \{\forall, \exists\}$ denotes quantification over all $t.x$ for which x satisfies $r.x$. We call x the **dummy**, $r.x$ the **range**, and $t.x$ the **term** of the quantification. We adopt the convention that all formulae are universally quantified over all free variables occuring in them (these are variables that are neither dummies nor program variables).

We will write f, g, h to denote predicate transformers, p, q, r to stand for predicates on program states, and R, S to denote relations on predicates.

Universal quantification over all program variables is denoted by surrounding a predicate by square brackets ([], read: everywhere). This unary operator has all the properties of universal quantification over a non-empty range. For a detailed discussion of this notation the reader is referred to [Dijkstra 1985].

The other operators we use are summarized below, ordered by increasing binding powers.

$$\equiv, \not\equiv$$
$$\Rightarrow, \Leftarrow$$
$$\textbf{ensures}, \mapsto$$
$$\wedge, \vee$$
$$\neg$$
$$=, \neq, \leq, <$$
$$+, -$$
$$\text{"."} \text{ (function application)}$$

All boolean and arithmetic operators have their usual meanings. We define $\Leftarrow$ (read: *follows-from*) by $[p \Leftarrow q \;\equiv\; q \Rightarrow p]$. For relations R, S on predicates we say that R is *stronger* than S (in formulae $R \Rightarrow S$) if and only if $\langle\forall p, q :: p\ R\ q \;\Rightarrow\; p\ S\ q\rangle$. For predicate transformers f, g we say that f is *stronger* than g (in formulae $f \Rightarrow g$) if and only if $\langle\forall p :: [f.p \Rightarrow g.p]\rangle$. Note that for predicates, relations, and predicate transformers $\Rightarrow$ is a partial order. The relations **ensures** and $\mapsto$ will be defined later.

1.2 Extreme Solutions of a Set of Equations

Most operators we introduce are defined as extreme solutions of sets of equations. We write $x : E$ to make explicit that E is a set of equations in the unknown x. Given a partial order $\Rightarrow$ on the solutions of E, we say that y is the *strongest solution* of E if and only if

(0) y solves E, and

(1) $\langle\forall z : z \text{ solves } E : y \Rightarrow z\rangle$.

The weakest solution of E is defined analogously.

1.3 Definition of *leads-to*

Let **ensures** be a given binary relation over predicates on program states; **ensures** describes the basic progress properties of a program. We assume that **ensures** fulfils the following requirements:

(D0) $\quad p \textbf{ ensures } p$

(D1) $\quad p \textbf{ ensures } \mathit{false} \;\Rightarrow\; [\neg p]$

(D2) $\quad (p \textbf{ ensures } r) \wedge [r \Rightarrow q] \;\Rightarrow\; p \textbf{ ensures } q$

The first requirement states that **ensures** is a reflexive relation. The second requirement excludes absurd progress properties. D2 states that the right-hand side of **ensures** may be weakened. Note that by combining D0 and D2 we get

Lemma 0 $[p \Rightarrow q] \;\Rightarrow\; p \;\mathbf{ensures}\; q$

Remark: The relation **ensures** was first defined in UNITY logic[Chandy and Misra 1988]. It is possible, however, to define this relation for other programming formalisms besides UNITY. Note that in such a case, **ensures** should also capture assumptions about fairness. (End of Remark)

Consider now the set A of equations A0–A2 in the unknown relation $\rhd$:

(A0) $$p \;\mathbf{ensures}\; q \;\Rightarrow\; p \rhd q$$
(A1) $$(p \rhd q) \wedge (q \rhd r) \;\Rightarrow\; p \rhd r$$
(A2) $$\langle \forall p : p \in W : p \rhd q \rangle \;\Rightarrow\; \langle \exists p : p \in W : p \rangle \rhd q$$

A0 states that the **ensures** relation is a subset of the relation $\rhd$. Transitivity of $\rhd$ is expressed by A1. A2 means that $\rhd$ is disjunctively closed over W, where W is any set of predicates.

With "$\Rightarrow$" as a partial order on relations we now prove the following

Lemma 1 *There is a unique strongest solution of* $\rhd : A$.

Proof (of Lemma 1, due to Dijkstra): We show that the conjunction of all solutions of A is a solution of A. Obviously then, this is the strongest solution of A, since it implies all solutions.

Now define a relation R as the conjunction of all solutions:

$$p \; R \; q \;\;\equiv\;\; \langle \forall S : S \text{ solves } A : p \; S \; q \rangle$$

Our proof obligation is to show that R solves A0, A1, and A2.
Ad A0:

$$
\begin{array}{ll}
 & p \; R \; q \\
= & \{\text{definition } R\} \\
 & \langle \forall S : S \text{ solves } A : p \; S \; q \rangle \\
\Leftarrow & \{\text{each } S \text{ solves A0}\} \\
 & \langle \forall S : S \text{ solves } A : p \;\mathbf{ensures}\; q \rangle \\
\Leftarrow & \{\text{term does not depend on dummy}\} \\
 & p \;\mathbf{ensures}\; q
\end{array}
$$

Ad A1:

$$
\begin{array}{ll}
 & (p\ R\ q) \land (q\ R\ r) \\
= & \{\text{definition } R\} \\
 & \langle \forall S : S \text{ solves } A : p\ S\ q\rangle \land \langle \forall S : S \text{ solves } A : q\ S\ r\rangle \\
= & \{\text{predicate calculus}\} \\
 & \langle \forall S : S \text{ solves } A : (p\ S\ q) \land (q\ S\ r)\rangle \\
\Rightarrow & \{\text{each } S \text{ solves A1}\} \\
 & \langle \forall S : S \text{ solves } A : p\ S\ r\rangle \\
= & \{\text{definition } R\} \\
 & p\ R\ r
\end{array}
$$

Ad A2 (we omit the range $p \in W$):

$$
\begin{array}{ll}
 & \langle \forall p :: p\ R\ q\rangle \\
= & \{\text{definition } R\} \\
 & \langle \forall p :: \langle \forall S : S \text{ solves } A : p\ S\ q\rangle\rangle \\
= & \{\text{interchange of quantifications}\} \\
 & \langle \forall S : S \text{ solves } A : \langle \forall p :: p\ S\ q\rangle\rangle \\
\Rightarrow & \{\text{each } S \text{ solves A2}\} \\
 & \langle \forall S : S \text{ solves } A : \langle \exists p :: p\rangle\ S\ q\rangle \\
= & \{\text{definition } R\} \\
 & \langle \exists p :: p\rangle\ R\ q
\end{array}
$$

(End of Proof)

We use Lemma 1 as the basis for the following definition[0]:

Definition 0 *The unique strongest solution of A is called* leads-to ($\mapsto$).

Using Lemma 0 and A0 we now observe:

Lemma 2 $\quad [p \Rightarrow q] \;\Rightarrow\; p \mapsto q$

As a Corollary of Lemma 2 we get

Corollary 0 $\quad p \mapsto p$

Remark: In [Chandy and Misra 1988] it was shown that if equation A1 is replaced by the weaker A1′ defined as

(A1′) $$(p \textbf{ ensures } q) \land (q \mapsto r) \;\Rightarrow\; p \mapsto r$$

then the two sets of equations A0, A1, A2 and A0, A1′, A2 are equivalent. In our proofs we will use whichever formulation is more convenient. (End of Remark)

[0]Except for Lemma 1 this is essentially the same definition as in [Chandy and Misra 1988]

1.4 Definition of *led-from*

We said earlier that the predicate transformer *led-from* formalizes our notion of the weakest predicate that *leads-to* q. In the following we define *led-from* formally, again by considering the strongest solution of the following set B of equations B0–B2 in the unknown predicate transformer f:

$$
\begin{array}{ll}
\text{(B0)} & p\ \mathbf{ensures}\ q \ \Rightarrow\ [p \Rightarrow f.q] \\
\text{(B1)} & [p \Rightarrow q] \ \Rightarrow\ [f.p \Rightarrow f.q] \\
\text{(B2)} & [f.(f.p) \Rightarrow f.p]
\end{array}
$$

B0 states that any predicate that **ensures** q is stronger than $f.q$. Monotonicity of f is expressed in B1. Equation B2 represents one half of the idempotence of f. Observe that from the reflexivity of **ensures** we can infer the other half:

$$
\begin{array}{ll}
 & p\ \mathbf{ensures}\ p \\
\Rightarrow & \{\text{B0}\} \\
 & [p \Rightarrow f.p] \\
\Rightarrow & \{\text{B1}\} \\
 & [f.p \Rightarrow f.(f.p)]
\end{array}
$$

Lemma 3 *There is a unique strongest solution of f: B.*

Proof (of Lemma 3): We again show that the conjunction of all solutions of B solves B. In the following we abbreviate

$$[g.p \ \equiv\ \langle \forall h : h \text{ solves } B : h.p \rangle]$$

Ad B0:

$$
\begin{array}{ll}
 & [p \Rightarrow g.q] \\
= & \{\text{definition } g\} \\
 & [p \Rightarrow \langle \forall h : h \text{ solves } B : h.q \rangle] \\
= & \{\text{predicate calculus}\} \\
 & [\langle \forall h : h \text{ solves } B : p \Rightarrow h.q \rangle] \\
= & \{\text{interchange of quantifications}\} \\
 & \langle \forall h : h \text{ solves } B : [p \Rightarrow h.q] \rangle \\
\Leftarrow & \{\text{each } h \text{ solves B0}\} \\
 & \langle \forall h : h \text{ solves } B : p\ \mathbf{ensures}\ q \rangle \\
\Leftarrow & \{\text{term does not depend on dummy}\} \\
 & p\ \mathbf{ensures}\ q
\end{array}
$$

Ad B1:

$$
\begin{array}{cl}
 & [g.p \Rightarrow g.q] \\
= & \{\text{definition } g\} \\
 & [g.p \Rightarrow \langle \forall h : h \text{ solves } B : h.q \rangle] \\
= & \{\text{predicate calculus}\} \\
 & [\langle \forall h : h \text{ solves } B : g.p \Rightarrow h.q \rangle] \\
= & \{\text{interchange of quantifications}\} \\
 & \langle \forall h : h \text{ solves } B : [g.p \Rightarrow h.q] \rangle \\
\Leftarrow & \{[g.p \Rightarrow h.p] \text{ and transitivity of } \Rightarrow\} \\
 & \langle \forall h : h \text{ solves } B : [h.p \Rightarrow h.q] \rangle \\
\Leftarrow & \{\text{each } h \text{ solves B1}\} \\
 & \langle \forall h : h \text{ solves } B : [p \Rightarrow q] \rangle \\
\Leftarrow & \{\text{term does not depend on dummy}\} \\
 & [p \Rightarrow q]
\end{array}
$$

Ad B2:

$$
\begin{array}{cl}
 & g.p \\
= & \{\text{definition } g\} \\
 & \langle \forall h : h \text{ solves } B : h.p \rangle \\
\Leftarrow & \{\text{each } h \text{ solves B2}\} \\
 & \langle \forall h : h \text{ solves } B : h.(h.p) \rangle \\
\Leftarrow & \{[g.p \Rightarrow h.p] \text{ and B1}\} \\
 & \langle \forall h : h \text{ solves } B : h.(g.p) \rangle \\
= & \{\text{definition } g\} \\
 & g.(g.p)
\end{array}
$$

(End of Proof)

Definition 1 *The unique strongest solution of B is called* led-from *(or* led *for short).*

The following Corollary is immediate:

Corollary 1 *led is monotonic and idempotent, and* $[p \Rightarrow led.p]$.

2 The relationship between *leads-to* and *led-from*

We said earlier that $led.p$ formalizes the notion of the weakest predicate that *leads-to* p. We will now make this relationship precise in the following

Theorem 0 (0) $p \mapsto q \;\equiv\; [p \Rightarrow led.q]$
(1) *For any q, $led.q$ is the weakest solution of p: $p \mapsto q$.*

Proof (of Theorem 0): Abbreviate $C \equiv p{:}\, p \mapsto q$. Our proof proceeds in five steps. We demonstrate that

(a) For any q, C has a unique weakest solution $f.q$,

(b) $led \Rightarrow f$,

(c) $p \mapsto q \;\Rightarrow\; [p \Rightarrow led.q]$,

(d) $[p \Rightarrow led.q] \;\Rightarrow\; p \mapsto q$,

(e) $f \Rightarrow led$.

Ad (a): Observe that $[p \equiv false]$ is a solution of C. We show that the disjunction of all solutions of C solves C. Hence, this is the weakest solution, since it follows from any solution.

Define $[f.q \equiv \langle \exists p : p \mapsto q : p\rangle]$. Our proof obligation is to show that for any q, $f.q$ solves C.

$$
\begin{array}{ll}
 & f.q \mapsto q \\
= & \{\text{definition } f\} \\
 & \langle \exists p : p \mapsto q : p\rangle \mapsto q \\
\Leftarrow & \{\text{A2}\} \\
 & \langle \forall p : p \mapsto q : p \mapsto q\rangle \\
= & \{\text{predicate calculus}\} \\
 & true
\end{array}
$$

We summarize the following two facts about $f.q$: it solves C, and any solution of C is stronger than $f.q$. Formally:

(F0) $$f.q \mapsto q$$
(F1) $$p \mapsto q \;\Rightarrow\; [p \Rightarrow f.q]$$

Ad (b): We show that f solves B, the defining equations for *led*. Then, since *led* is the strongest solution of B, the result follows.

Ad B0:

$$
\begin{array}{ll}
 & p \textbf{ ensures } q \\
\Rightarrow & \{\text{A0}\} \\
 & p \mapsto q \\
\Rightarrow & \{\text{F1}\} \\
 & [p \Rightarrow f.q]
\end{array}
$$

Ad B1:

$$
\begin{array}{cl}
 & [p \Rightarrow q] \\
\Rightarrow & \{\text{Lemma 2}\} \\
 & p \mapsto q \\
= & \{\text{F0}\} \\
 & (f.p \mapsto p) \wedge (p \mapsto q) \\
\Rightarrow & \{\text{A1}\} \\
 & f.p \mapsto q \\
\Rightarrow & \{\text{F1}\} \\
 & [f.p \Rightarrow f.q]
\end{array}
$$

Ad B2:

$$
\begin{array}{cl}
 & [f.(f.p) \Rightarrow f.p] \\
\Leftarrow & \{\text{F1}\} \\
 & f.(f.p) \mapsto p \\
\Leftarrow & \{\text{A1}\} \\
 & (f.(f.p) \mapsto f.p) \wedge (f.p \mapsto p) \\
= & \{\text{F0 twice, once with } p := f.p\} \\
 & \mathit{true}
\end{array}
$$

Ad (c): We show that the relation R defined as $p\ R\ q \quad \equiv \quad [p \Rightarrow \mathit{led}.q]$ solves A. Then, since $\mapsto$ is the strongest solution of A, the result follows.

Ad A0:

$$
\begin{array}{cl}
 & p\ R\ q \\
= & \{\text{definition } R\} \\
 & [p \Rightarrow \mathit{led}.q] \\
\Leftarrow & \{\text{B0}\} \\
 & p\ \textbf{ensures}\ q
\end{array}
$$

Ad A1:

$$
\begin{array}{cl}
 & (p\ R\ q) \wedge (q\ R\ r) \\
= & \{\text{definition } R\} \\
 & [p \Rightarrow \mathit{led}.q] \wedge [q \Rightarrow \mathit{led}.r] \\
\Rightarrow & \{\text{B1}\} \\
 & [p \Rightarrow \mathit{led}.q] \wedge [\mathit{led}.q \Rightarrow \mathit{led}.(\mathit{led}.r)] \\
\Rightarrow & \{\text{transitivity of } \Rightarrow\} \\
 & [p \Rightarrow \mathit{led}.(\mathit{led}.r)] \\
= & \{\text{Corollary 1}\} \\
 & [p \Rightarrow \mathit{led}.r] \\
= & \{\text{definition } R\} \\
 & p\ R\ r
\end{array}
$$

Ad A2 (the range $p \in W$ is omitted):

$$
\begin{array}{ll}
 & \langle \forall p :: p \; R \; q \rangle \\
= & \{\text{definition } R\} \\
 & \langle \forall p :: [p \Rightarrow led.q] \rangle \\
= & \{\text{interchange of quantifications}\} \\
 & [\langle \forall p :: p \Rightarrow led.q \rangle] \\
= & \{\text{predicate calculus}\} \\
 & [\langle \exists p :: p \rangle \Rightarrow led.q] \\
= & \{\text{definition } R\} \\
 & \langle \exists p :: p \rangle \; R \; q
\end{array}
$$

Ad (d): We observe for any p, q:

$$
\begin{array}{ll}
 & [p \Rightarrow led.q] \\
\Rightarrow & \{\text{part (b) and transitivity of } \Rightarrow\} \\
 & [p \Rightarrow f.q] \\
\Rightarrow & \{\text{Lemma 2}\} \\
 & p \mapsto f.q \\
= & \{\text{F0}\} \\
 & (p \mapsto f.q) \wedge (f.q \mapsto q) \\
\Rightarrow & \{\text{A1}\} \\
 & p \mapsto q
\end{array}
$$

Ad (e): We observe for any p:

$$
\begin{array}{ll}
 & [f.p \Rightarrow led.p] \\
= & \{\text{parts (c) and (d)}\} \\
 & f.p \mapsto p \\
= & \{\text{F0}\} \\
 & true
\end{array}
$$

(End of Proof)

3 Theorems about *led-from*

We will now derive a few more properties of *led* and show that in the UNITY logic, *led* does not enjoy a number of junctivity properties such as or-continuity and finite conjunctivity and disjunctivity.

3.1 More Properties of *led-from*

As a direct consequence of Theorem 0 we get as our first result the following Lemma, which allows us to compute *led* for a special case.

Lemma 4 $\quad true \mapsto p \quad \equiv \quad [led.p \equiv true]$

Since by Lemma 2 $true \mapsto true$ we get from the above

Corollary 2 $\quad [led.true \equiv true]$.

Lemma 5 $\quad [p \Rightarrow led.q] \quad \equiv \quad [led.p \Rightarrow led.q]$

Proof (of Lemma 5):

$$
\begin{array}{lll}
\text{"}\Rightarrow\text{"} & & [p \Rightarrow led.q] \\
 & \Rightarrow & \{\text{B1}\} \\
 & & [led.p \Rightarrow led.(led.q)] \\
 & = & \{\text{Corollary 1}\} \\
 & & [led.p \Rightarrow led.q]
\end{array}
$$

$$
\begin{array}{lll}
\text{"}\Leftarrow\text{"} & & [p \Rightarrow led.q] \\
 & \Leftarrow & \{\text{by Corollary 1, } [p \Rightarrow led.p]\} \\
 & & [led.p \Rightarrow led.q]
\end{array}
$$

(End of Proof)

Next we show that if all **ensures** properties with respect to some predicate q are of a certain trivial form, then $[q \equiv led.q]$. More precisely we have

Theorem 1 $\quad \langle \forall p :: p \ \mathbf{ensures}\ q \ \Rightarrow \ [p \Rightarrow q] \rangle \ \Rightarrow \ [q \equiv led.q]$

Proof (of Theorem 1): We show that q is the weakest solution of the equation $p{:}\, p \mapsto q$. Then by Theorem 0 part (1), $[q \equiv led.q]$.

So our proof obligation is twofold:

(a) q solves $p{:}\, p \mapsto q$, i.e. $q \mapsto q$

(b) q is weakest, i.e. $\langle \forall p :: p \mapsto q \ \Rightarrow \ [p \Rightarrow q] \rangle$

We note that (a) follows trivially from Lemma 2. We will prove (b) by showing that, for any p, R defined as

$$p \, R \, q \quad \equiv \quad [p \Rightarrow q]$$

satisfies equations A0,A1$'$, and A2. Then we observe for any p

$$
\begin{array}{ll}
 & p \mapsto q \\
\Rightarrow & \{\mapsto \text{ is the strongest solution of A0, A1}', \text{ and A2}\} \\
 & p \; R \; q \\
= & \{\text{definition } R\} \\
 & [p \Rightarrow q]
\end{array}
$$

Ad A0:

$$
\begin{array}{ll}
 & [p \Rightarrow q] \\
\Leftarrow & \{\text{antecedent of Theorem}\} \\
 & p \textbf{ ensures } q
\end{array}
$$

Ad A1′:

$$
\begin{array}{ll}
 & (p \textbf{ ensures } r) \wedge [r \Rightarrow q] \\
\Rightarrow & \{\text{D2}\} \\
 & p \textbf{ ensures } q \\
\Rightarrow & \{\text{antecedent of Theorem}\} \\
 & [p \Rightarrow q]
\end{array}
$$

Ad A2 (the range $p \in W$ is omitted):

$$
\begin{array}{ll}
 & \langle \forall p :: [p \Rightarrow q] \rangle \\
= & \{\text{interchange of quantifications}\} \\
 & [\langle \forall p :: p \Rightarrow q \rangle] \\
= & \{\text{predicate calculus}\} \\
 & [\langle \exists p :: p \rangle \Rightarrow q]
\end{array}
$$

(End of Proof)

From D1 we now have the following Corollary of Theorem 1:

Corollary 3 $[\mathit{led}.\mathit{false} \equiv \mathit{false}]$

3.2 Some Results about *led-from* in UNITY

We show that in UNITY *led* is not **or**-continuous, and neither finitely disjunctive nor finitely conjunctive. In the following we first define **ensures** for UNITY[Chandy and Misra 1988], together with the notions of **or**-continuity and finite conjunctivity and disjunctivity. We then give UNITY programs that show that for UNITY *led* has none of these properties.

3.2.1 A Few More Definitions

We give a very brief and incomplete description of UNITY programs and the UNITY logic. For our purposes, a UNITY program consists of three parts: a collection of variable declarations, a set of initial conditions, and a finite set of multiple assignment statements. We call these parts **declare**, **initially**, and **assign**, respectively. We define the predicate transformer **wp** for each multiple assignment in **assign** in the usual way. From an operational point of view, the execution of a UNITY program starts from any state satisfying the initial conditions and proceeds by repeatedly picking any statement in **assign** and executing it. The only fairness constraint we impose is that each statement in **assign** is picked infinitely often.

This operational interpretation is the motivation for the definition of **ensures** that is given below. However, neither our definitions nor our proofs will mention program executions or fairness.

Definition 2

$$\begin{aligned} p \ \mathbf{ensures}\ q \quad \equiv \quad & \langle \forall s : s \in \mathbf{assign} : [p \wedge \neg q \ \Rightarrow\ \mathbf{wp}.s.(p \vee q)] \rangle \ \wedge \\ & \langle \exists s : s \in \mathbf{assign} : [p \wedge \neg q \ \Rightarrow\ \mathbf{wp}.s.q] \rangle \end{aligned}$$

It is straightforward to verify that this definition of **ensures** satifies requirements D0, D1, and D2 of Section 1.3.

For a multiple assignment statement $x := f.x$, where x is a list of program variables and $f.x$ is a list of expressions matching x in number and type, we define **wp** in the following standard way:

Definition 3 $[\mathbf{wp}.\text{“}x := f.x\text{”}.p \ \equiv\ (f.x =: x).p]$, *where* $(f.x =: x).p$ *denotes* p *with every occurrence of* x *replaced by* $f.x$.

Next we define **or**-continuity, finite conjunctivity and finite disjunctivity:

Definition 4 *Let* f *be a predicate transformer and* $Y(i{\geq}0)$ *a weakening sequence of predicates. We say that* f *is* **or**-continuous *if and only if*

$$[f.\langle \exists i : i{\geq}0 : Y.i \rangle \ \equiv\ \langle \exists i : i{\geq}0 : f.(Y.i) \rangle]$$

Definition 5 *A predicate transformer* f *is finitely conjunctive if and only if*

$$\langle \forall p, q :: [f.(p \wedge q) \ \equiv\ f.p \wedge f.q] \rangle$$

It is finitely disjunctive if and only if

$$\langle \forall p, q :: [f.(p \vee q) \ \equiv\ f.p \vee f.q] \rangle$$

3.2.2 Junctivity Results for *led-from*

We now show that with these definitions, *led* is not **or**-continuous.

Theorem 2 *led is not* **or**-*continuous.*

Proof (of Theorem 2): Consider the following program, which can be thought of as adding up a series of random numbers:

$$\begin{array}{ll} \textbf{declare} & x, n : \textbf{integer} \\ \textbf{initially} & \mathit{true} \\ \textbf{assign} & n := n+1 \ [\!] \ x, n := x+n, 0 \\ \textbf{end} & \end{array}$$

We will show the following:

$$\text{(F2)} \qquad \langle \forall i : i \geq 0 : \mathit{led}.(0 \leq x \land x < i) \quad \equiv \quad 0 \leq x \land x < i \rangle$$

Notice that $0 \leq x \ \land \ x < i \, (i \geq 0)$ is a weakening sequence of predicates. Furthermore observe that:

$$\begin{array}{cl} & \langle \exists i : i \geq 0 : 0 \leq x \land x < i \rangle \\ = & \{\text{predicate calculus}\} \\ & 0 \leq x \land \langle \exists i : i \geq 0 : x < i \rangle \\ = & \{\text{arithmetic}\} \\ & 0 \leq x \end{array}$$

Calling this result $(*)$, and assuming that F2 has been proved, we can now calculate

$$\begin{array}{cl} & \langle \exists i : i \geq 0 : \mathit{led}.(0 \leq x \land x < i) \rangle \\ = & \{\text{F2}\} \\ & \langle \exists i : i \geq 0 : 0 \leq x \land x < i \rangle \\ = & \{(*)\} \\ & 0 \leq x \end{array}$$

On the other hand we have (the proof is straightforward, but requires some more UNITY machinery and is therefore omitted) $\mathit{true} \mapsto 0 \leq x$. Hence by Lemma 4 we obtain $(**)$ $[\mathit{led}.(0 \leq x) \equiv \mathit{true}]$, and we observe

$$\begin{array}{cl} & \mathit{led}.\langle \exists i : i \geq 0 : 0 \leq x \land x < i \rangle \\ = & \{(*)\} \\ & \mathit{led}.(0 \leq x) \\ = & \{(**)\} \\ & \mathit{true} \end{array}$$

and we see that *led* is not or-continuous. So we are left with demonstrating F2.

To this end abbreviate $0 \leq x \wedge x < i$ by q. By Theorem 1 it suffices to show that

$$\langle \forall i : i \geq 0 : \langle \forall p :: p \ \mathbf{ensures} \ q \ \ \Rightarrow \ \ [p \Rightarrow q] \rangle \rangle$$

Massaging p **ensures** q we get for any i and p, using the universal part of the definition of **ensures**:

$$\begin{array}{ll} & p \wedge \neg q \\ \Rightarrow & \{\text{interchange of quantifications}\} \\ & \mathbf{wp}\,.\text{“}n =: n+1\text{”}.(p \vee q) \wedge \mathbf{wp}\,.\text{“}x, n := x+n, 0\text{”}.(p \vee q) \\ \Rightarrow & \{\text{predicate calculus}\} \\ & \mathbf{wp}\,.\text{“}n =: n+1\text{”}.(p \vee q) \\ = & \{\text{definition } \mathbf{wp}\text{; } n \text{ does not occur in } q\} \\ & (n+1 =: n).p \ \vee \ q \end{array}$$

Rewriting this slightly yields

$$[p \ \ \Rightarrow \ \ (n+1 =: n).p \ \vee \ q]$$

For the existential part of **ensures** we get:

$$\begin{array}{ll} & [p \wedge \neg q \ \ \Rightarrow \ \ \mathbf{wp}\,.\text{“}n =: n+1\text{”}.q] \vee [p \wedge \neg q \ \ \Rightarrow \ \ \mathbf{wp}\,.\text{“}x, n := x+n, 0\text{”}.q] \\ = & \{\text{definition } \mathbf{wp}\text{; } n \text{ does not occur in } q\} \\ & [p \wedge \neg q \ \ \Rightarrow \ \ q] \vee [p \wedge \neg q \ \ \Rightarrow \ \ (x+n =: x).q] \\ = & \{\text{predicate calculus}\} \\ & [p \ \ \Rightarrow \ \ q] \vee [p \ \ \Rightarrow \ \ q \vee (x+n =: x).q] \\ \Rightarrow & \{\text{predicate calculus}\} \\ & [p \ \ \Rightarrow \ \ q \vee (x+n =: x).q] \end{array}$$

Combining both we arrive at

$$[p \ \ \Rightarrow \ \ q \vee ((n+1 =: n).p \wedge (x+n =: x).q)]$$

Now we generalize this result by a straightforward induction (omitted) to:

$$\langle \forall j : j \geq 0 : [p \ \ \Rightarrow \ \ q \vee ((n+j+1 =: n).p \wedge (x+n+j =: x).q)] \rangle$$

Next we observe, using the above result, that

$$\begin{array}{ll} & \langle \forall j : j \geq 0 : [p \ \ \Rightarrow \ \ q \vee ((n+j+1 =: n).p \wedge (x+n+j =: x).q)] \rangle \\ \Rightarrow & \{\text{predicate calculus}\} \\ & \langle \forall j : j \geq 0 : [p \ \ \Rightarrow \ \ q \vee (x+n+j =: x).q] \rangle \\ \Rightarrow & \{\text{predicate calculus}\} \\ & [p \ \ \Rightarrow \ \ q \vee \langle \forall j : j \geq 0 : (x+n+j =: x).q \rangle] \end{array}$$

We now concentrate on the second disjunct:

$$
\begin{array}{ll}
 & \langle \forall j : j \geq 0 : (x+n+j =: x).q \rangle \\
= & \{\text{definition } q\} \\
 & \langle \forall j : j \geq 0 : 0 \leq x+n+j \ \wedge \ x+n+j < i \rangle \\
= & \{j \text{ does not have an upper limit}\} \\
 & \mathit{false}
\end{array}
$$

from which $[p \Rightarrow q]$ follows. (End of Proof)

Theorem 3 *led is not finitely conjunctive.*

Proof (of Theorem 3): Consider the program

$$
\begin{array}{ll}
\textbf{declare} & x : \textbf{integer} \\
\textbf{initially} & \mathit{true} \\
\textbf{assign} & x := 0 \ [] \ x := 1 \\
\textbf{end} &
\end{array}
$$

Since (without proof) $\mathit{true} \mapsto x{=}0$ and $\mathit{true} \mapsto x{=}1$ are properties of this program we know $[\mathit{led}.(x{=}0) \equiv \mathit{true}]$ and $[\mathit{led}.(x{=}1) \equiv \mathit{true}]$. Hence we have $[\mathit{led}.(x{=}0) \wedge \mathit{led}.(x{=}1) \equiv \mathit{true}]$. On the other hand, however, we observe

$$
\begin{array}{ll}
 & \mathit{led}.(x{=}0 \wedge x{=}1) \\
= & \{\text{arithmetic}\} \\
 & \mathit{led}.\mathit{false} \\
= & \{\text{Corollary 3}\} \\
 & \mathit{false}
\end{array}
$$

(End of Proof)

Theorem 4 *led is not finitely disjunctive.*

Proof (of Theorem 4): Consider the program

$$
\begin{array}{ll}
\textbf{declare} & x : \textbf{integer} \\
\textbf{initially} & \mathit{true} \\
\textbf{assign} & x := x+1 \ [] \ x := x+2 \\
\textbf{end} &
\end{array}
$$

By a technique similar to the one employed in the proof of Theorem 2 we can show

$$
\begin{array}{lcl}
[\,\mathit{led}.(x{=}0) & \equiv & x{=}0] \\
[\,\mathit{led}.(x{=}1) & \equiv & x{=}1]
\end{array}
$$

Also since $x = -1 \mapsto x = 0 \vee x = 1$ we know that

$$[\,led.(x=0 \vee x=1) \;\Leftarrow\; x = -1]$$

As a consequence, *led* is not finitely disjunctive. (End of Proof)

4 Discussion

We used the technique of defining an operator as an extreme solution of a set of equations to formalize progress properties of parallel programs. We have shown that it is possible to formalize the notion of the weakest predicate that *leads-to* q, for any q, by defining a predicate transformer *led-from*. We demonstrated that *led-from* and *leads-to* are equivalent in expressive power. We investigated the properties of *led-from* in the context of UNITY and discovered that it was idempotent and monotonic, but neither **or**-continuous nor finitely junctive.

Future work has to concentrate on finding a fixpoint characterization of *led-from* that allows $led.q$ to be computed from a given program. Such a definition should drastically shorten the proofs of Theorems 2, 3, and 4. The usefulness of *led-from* will also depend on whether it is possible to prove a number of Meta-Theorems about *leads-to* (e.g. PSP-Theorem, Completion Theorem, cf. [Chandy and Misra 1988]), without resorting to induction on the structure of a proof of *leads-to*.

Acknowledgements

We are indebted to Dr. Jayadev Misra for posing the problem and contributing valuable ideas. We would also like to thank the members of the Austin Tuesday Afternoon Club for their comments and suggestions, in particluar Edsger W. Dijkstra, who helped clean up the proof of Theorem 1. This work was supported in part by ONR contract N00014-87-K-0510.

References

[Chandy and Misra 1988] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison Wesley, 1988.

[Dijkstra 1985] E. W. Dijkstra. *On Structures.* EWD 928, University of Texas at Austin, Nov. 1985.

[Lamport 1977] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, 3(2):125–143, March 1977.