Properties of Concurrent Programs

K. Mani Chandy
California Institute of Technology 256-80,
Pasadena, California 91125
mani@vlsi.caltech.edu *

June 15, 1993

Abstract

A program property is a predicate on programs. In this paper we explore program properties of the form $U \rightharpoonup V$ where U and V are either predicates on states of a program or program properties, and $\rightharpoonup$ satisfies three rules that are also used in reasoning about sequential programs and safety properties of parallel programs. We show how such properties can be used to reason about concurrent programs.

1 Introduction

Let U and V be predicates on a space $\mathcal{S}$. In the definition of the progress property $\mathcal{S}$ is a space of program states, and for the definition of the compositional property, $\mathcal{S}$ is a space of programs. We define an operator $\rightharpoonup$ on predicates on $\mathcal{S}$, where $U \rightharpoonup V$ is a predicate on programs. We explore operators that satisfy the following three rules:

For any predicate V on $\mathcal{S}$ and any bag X of predicates on $\mathcal{S}$ (where X can be empty or infinite):

$$[(\forall U : U \in X : U \rightharpoonup V) \quad \equiv \quad (\exists U : U \in X : U) \rightharpoonup V] \tag{1}$$

For any any predicate U on $\mathcal{S}$ and any bag X of predicates on $\mathcal{S}$, where X can be empty and we postpone consideration of whether X can be infinite:

$$[(\forall V : V \in X : U \rightharpoonup V) \quad \equiv \quad U \rightharpoonup (\forall V : V \in X : V)] \tag{2}$$

For any predicates U, V and W on $\mathcal{S}$:

$$[(U \rightharpoonup V) \wedge (V \rightharpoonup W) \quad \Rightarrow \quad (U \rightharpoonup W)] \tag{3}$$

We suggest two such program properties: one to reason about progress properties and the other to reason about parallel composition of programs.

*Supported in Part by AFOSR 91-0070.

Motivation Our goal is to explore the possibility of using a common set of rules for reasoning about sequential programs, and safety, progress and composition in concurrent programs. A program property for reasoning about safety, suggested by Misra [5], satisfies the three rules. The Hoare triple [3] satisfies the first two rules with U and V as the pre and post conditions, respectively.

Observations From (1), setting X to the empty set:

$$[false \rightharpoonup V] \tag{4}$$

From (2), setting X to the empty set:

$$[U \rightharpoonup true] \tag{5}$$

From (2) and (5), for any predicates U, U' and V on $\mathcal{S}$:

$$[U \Leftarrow U'] \;\Rightarrow\; [(U \rightharpoonup V) \;\Rightarrow\; (U' \rightharpoonup V)] \tag{6}$$

From (1) and (4), for any predicates U, V and V' on $\mathcal{S}$.

$$[V \Rightarrow V'] \;\Rightarrow\; [(U \rightharpoonup V) \;\Rightarrow\; (U \rightharpoonup V')] \tag{7}$$

Restriction on Program Properties We define functions h and g from predicates on $\mathcal{S}$ to predicates on some space $\mathcal{C}.P$, where for the time being we do not specify $\mathcal{C}.P$, though later we shall define it as the computations of P when dealing with progress properties, the states of P when dealing with safety properties, and the set of programs that has P as a component when dealing with compositional properties.

We restrict attention to program properties defined as follows:

$$(U \rightharpoonup V).P \;\equiv\; [h.U \Rightarrow g.V] \tag{8}$$

and where (i) h is universally disjunctive, (ii) g is finitely conjunctive and we postpone consideration of whether g is conjunctive over infinite sets, and (iii) g is stronger than h, (9) - (11), respectively:

For any bag X of predicates on $\mathcal{S}$:

$$[(\exists U : U \in X : h.U) \equiv h.(\exists U : U \in X : U)] \tag{9}$$

For any bag X of predicates on $\mathcal{S}$, where X can be empty, and we postpone consideration of whether X can be infinite:

$$[(\forall V : V \in X : g.V) \equiv g.(\forall V : V \in X : V)] \tag{10}$$

$$[g.V \Rightarrow h.V] \tag{11}$$

We show that (1) - (3) follow from (9) - (11), respectively. Transitivity (3) of $\rightharpoonup$ follows from (11) and transitivity of $\Rightarrow$. Next, we show that (1) follows from (9). Likewise, we show that (2) follows from (10), and set X can be infinite in (2) if it can be infinite in (10).

Theorem 1

$$[(\forall U : U \in X : U \rightarrow V) \equiv (\exists U : U \in X : U) \rightharpoonup V]$$

Proof:

$(\forall U : U \in X : U \rightarrow V).P$

$\equiv$ { definition of $\rightharpoonup$ }

$(\forall U : U \in X : [h.U \Rightarrow g.V])$

$\equiv$ { interchange quantification }

$[(\forall U : U \in X : h.U \Rightarrow g.V)]$

$\equiv$ { predicate calculus }

$[(\exists U : U \in X : h.U) \Rightarrow g.V]$

$\equiv$ { disjunctivity of h (9) }

$[h.(\exists U : U \in X : U) \Rightarrow g.V]$

$\equiv$ { definition of $\rightharpoonup$}

$((\exists U : U \in X : U) \rightharpoonup V).P$

Theorem 2

$$[(\forall V : V \in X : U \rightharpoonup V) \equiv U \rightharpoonup (\forall V : V \in X : V)]$$

Proof:

$(\forall V : V \in X : U \rightarrow V).P$

$\equiv$ { definition of $\rightharpoonup$ }

$(\forall V : V \in X : [h.U \Rightarrow g.V])$

$\equiv$ { interchange quantification }

$[(\forall V : V \in X : h.U \Rightarrow g.V)]$

$\equiv$ { predicate calculus }

$[h.U \Rightarrow (\forall V : V \in X : g.V)]$

$\equiv$ { conjunctivity of g (10)}

$[h.U \Rightarrow g.(\forall V : V \in X : V)]$

$\equiv$ { definition of $\rightharpoonup$}

$(U \rightharpoonup (\forall V : V \in X : V)).P$

Next we define programs and computations, based on [1] and later discuss $\rightharpoonup$, h and g that satisfy (8) - (11).

2 Programs

A *program* is a pair:

1. a finite nonempty set of typed variables, and
2. a finite, nonempty set of statements.

The state of a program is given by the values of its variables. Execution of any statement in any state terminates, and therefore $wp.s$ is universally-conjunctive.

A *computation* C of a program is a state S_0 and an infinite sequence of pairs $(c_i, S_i), i > 0$, where c_i is a statement of the program, and S_i is a state of the program for all i, and

1. execution of c_i in state S_{i-1} takes the program to state S_i, for $i > 0$, and
2. each statement of the program appears infinitely often in the computation.

The *parallel composition* of programs F and G is denoted by $F \| G$, and is defined if and only if the variable declarations in F and G are compatible. The set of variables of $F \| G$ is the union of the sets of variables of F and G. Likewise, the set of statements of $F \| G$ is the union of the sets of statements of F and G. From the definition, $\|$ is associative, commutative and idempotent.

3 Progress Properties

We introduce a program property, $U \hookrightarrow V$, read as "U leads to always V," defined as follows:

$$(U \hookrightarrow V).P \equiv (\forall C : C \text{ is a computation of } P : h.U.C \Rightarrow g.V.C)$$

where h and g are defined as follows. For any computation C where C is a state S_0 and sequence (c_i, S_i), $i > 0$, where S_i is a state and c_i is a statement, for all $i > 0$:

$$g.V.C \equiv (\exists i :: (\forall j : i < j : V.S_j))$$

$$h.U.C \equiv (\exists i :: U.S_i)$$

Theorem 3 With $\rightharpoonup$ defined as $\hookrightarrow$, conditions (1) - (3) are satisfied where set X is finite in (2).
Proof: Let $\mathcal{C}.P$ be the set of computations of P. For any bag X of predicates, from the interchange-quantification rule [2],

$$(\exists i :: (\exists U : U \in X : U.S_i)) \equiv (\exists U : U \in X : (\exists i :: U.S_i))$$

and hence, h is universally disjunctive (9).

If V holds at all points of a suffix of a sequence, and V' holds at all points of a suffix of the same sequence, then $V \wedge V'$ holds at all points of the smaller of the suffixes; therefore, g is conjunctive over finite nonempty sets. Also, $g.true$ holds for all computations C and hence g is conjunctive over the empty set. Therefore g is finitely conjunctive (10). Formula (11) follows directly from the definitions of g and h. Hence (9) - (11) hold. Therefore (1) - (3) hold.

Operational Meaning If $U \hookrightarrow V$ is a property of a program P then for all computations of P if U holds at any point in the computation then there is a point in the computation after which V continues to hold for ever.

4 Safety Properties

This section is brief review of material from UNITY [1] and [5].

Define a program property $U \ co \ V$ where U and V are predicates on states, as:

$$(U \ co \ V).P \ \equiv \ [U \Rightarrow V \wedge (\forall s : s \text{ is a statement of } P : wp.s.V)]$$

Theorem 4 With $\rightharpoonup$ defined as co, conditions (1) - (3) are satisfied where set X can be infinite in (2).
Proof: Define h to be the identity function, and g as:

$$g.V \equiv V \wedge (\forall s : s \text{ is a statement of } P : wp.s.V)$$

and $\mathcal{C}.P$ to be the set of states of P. Since h is the identity function, it is universally disjunctive. Since $wp.s$ is universally conjunctive for all statements s of the program, g is universally conjunctive. From the definitions of g and h it follows that g is stronger than h. Therefore, (9) - (11) are satisfied. Therefore (1) - (3) hold.

Operational Meaning If U *co* V is a property of a program P then in all computations of P, if U holds at any point in the computation then V holds at the next point of the computation and U is stronger than V.

5 Parallel Composition

This section is motivated primarily by the hypothesis-conclusion rules in UNITY [1] and also by Jones' work on Rely-Guarantee [4].

We define an operator $\succ$ (read "guarantees") on properties, where $U \succ V$ is the property defined as

$$(U \succ V).P \equiv (\forall \text{ programs } Q : U.(Q\|P) : V.(Q\|P)) \tag{12}$$

Theorem 5 With $\rightharpoonup$ defined as $\succ$, formulae (1) - (3) hold, where set X can be infinite in (2).

Proof: Formulae (9) - (11) hold with $\mathcal{C}.P$ defined to be the set of all programs $Q\|P$, all Q, and where h and g are identity functions.

Theorem 6

$$(U \succ V).Q \equiv (\forall Q' :: (U \succ V).Q\|Q') \tag{13}$$

Proof:

$(U \succ V).Q$

$\equiv$ {definition of $\succ$ }

$(\forall P : U.Q\|P : V.Q\|P)$

$\equiv$ $\{P := Q'\|P'$ and since $Q'\|P' = P$ with $Q' = Q,\ P' = P\}$

$(\forall Q', P' : U.Q\|Q'\|P' : V.Q\|Q'\|P')$

$\equiv$ { predicate calculus }

$(\forall Q' :: (\forall P' :: U.Q\|Q'\|P' : V.Q\|Q'\|P'))$

$\equiv$ {definition of $\succ$ }

$(\forall Q' :: (U \succ V).Q\|Q')$

Corollary For a set of programs $P_0 \ldots P_n$:

$$(\forall i :: (U_i \succ V_i).P_i) \Rightarrow (\forall i :: (U_i \succ V_i).(P_0\| \ldots \|P_n))$$

Proof: Follows from the last theorem, and the associativity and commutativity of parallel composition.

Theorem 7

$$(\forall i : 0 \leq i \leq n : (U_i \ \succ \ V_i).P_i) \Rightarrow ((\forall i :: U_i) \ \succ \ (\forall i :: V_i)).(P_0 \| \ldots \| P_n)$$

Proof:

$(\forall i :: (U_i \ \succ \ V_i).P_i)$

$\Rightarrow$ { the corollary }

$(\forall i :: (U_i \ \succ \ V_i).(P_0 \| \ldots \| P_n))$

$\Rightarrow$ { antimonotonicity with respect to first argument }

$(\forall i :: ((\forall i :: U_i) \ \succ \ V_i).(P_0 \| \ldots \| P_n))$

$\equiv$ { conjunctivity with respect to second argument }

$((\forall i :: U_i) \ \succ \ (\forall i :: V_i)).(P_0 \| \ldots \| P_n)$

6 Proofs of Compositions of Concurrent Programs

6.1 Properties of Compositions of Concurrent Programs

A property b is defined to be an *all-component* property if and only if for any set of programs $P_0 \ldots P_n$ such that $P_0 \| \ldots \| P_n$ is defined:

$$(\forall i :: b.P_i) \equiv b.(P_0 \| \ldots \| P_n)$$

A property b is defined to be an *any-component* property if and only if:

$$b.P \equiv (\forall Q : P \| Q \text{ is defined } : b.(P \| Q))$$

$U \succ V$ is an any-component property, and $U \ co \ V$ is an all-component property. The conjunction of all-component properties is an all-component property and the conjunction of any-component properties is an any-component property. Let U, V and W be program properties, and let U be an any-component property. From the definition of $\succ$:

$$[U \Rightarrow (V \Rightarrow W)] \quad \Rightarrow \quad [U \Rightarrow (V \succ W)] \tag{14}$$

Theorem 7 suggests one way of approaching parallel program design. We can prove that the parallel composition of programs $P_0 \ldots P_n$ has property V if we can find properties V_i and all-component properties U_i such that:

$$(\forall i :: U_i) = U \quad \wedge \quad (\forall i :: V_i) = V$$

and where U and $U_i \succ V_i$ are properties of P_i, all i.

6.2 Examples of Properties

Define program properties stable.V [1] and transient.V, where V is a predicate on states, as follows:

$$[\text{stable}.V \ \equiv \ V \ co \ V]$$

$$\text{transient}.V.P \ \equiv \ (\exists s : s \text{ statement of } P : [V \Rightarrow wp.s.\neg V])$$

From the definitions, stable.V is an all-component property and transient.V is an any-component property. If stable.V is a property of a program P then in all computations of P, if V holds at any point in a computation then V continues to hold for ever thereafter in that computation. If transient.V is a property of a program P then in all computations of P: $\neg V$ holds infinitely often.

From arguments in [1]:

$$\text{transient}.(U \wedge \neg V) \wedge (U \wedge \neg V \ co \ U \vee V) \wedge \text{stable}.V \ \Rightarrow \ (U \hookrightarrow V) \qquad (15)$$

Setting U to V in the above, and since

$$[\text{transient}.\mathit{false}]$$

and, for all V

$$[\mathit{false} \ co \ V]$$

we get

$$[\text{stable}.V \ \Rightarrow \ (V \hookrightarrow V)] \qquad (16)$$

Since transient.V is an any-component property, from (14) and (15) and predicate calculus:

$$[\text{transient}.(U \wedge \neg V) \quad \Rightarrow \quad (((U \wedge \neg V \ co \ U \vee V) \wedge \text{stable}.V) \ \succ \ (U \hookrightarrow V))]$$

7 Example

7.1 Example: Specification

The problem is a more complex version of the *earliest meeting time* example in [1]. A parallel composition of a finite nonempty set P of professors, $P = \{P_i | 0 \leq i \leq n\}$, and a secretary Sec, computes the earliest time at which all professors can meet. Time ranges over the natural numbers. Associated with P_i is a monotone nondecreasing function f_i from integers to integers, where

$$f_i(t) \geq t$$

and for any given time t, professor P_i cannot meet in the interval $[t, f_i(t))$ and can meet at $f_i(t)$. Define $g(t)$ as

$$g(t) = \max_i f_i(t)$$

Since $f_i(t) \geq t$, all i:

$$t = g(t) \;\equiv\; (\forall i :: t = f_i(t))$$

Therefore, the earliest meeting time, is:

$$e = (\min t : t = g(t) : t)$$

The specification states that e exists.

Since g is monotone nondecreasing and $g(e) = e$:

$$t < e \;\Rightarrow\; g(t) = e \;\;\vee\;\; (e > g(t) \geq t + 1)$$

Let $g^{(m)}(t)$ be m successive applications of g to t, where $g^{(0)}$ is the identity function. From the previous formula, for all $m > 0$:

$$(\forall t : t \leq e : g^{(m)}(t) = e \;\;\vee\;\; (e > g^{(m)}(t) \geq t + m))$$

Therefore:

$$(\forall t : t \leq e : g^{(e-t)}(t) = e)$$

Therefore:

$$g^{(e)}(0) = e \tag{17}$$

For convenience, we define a program property $asc.z$ (for "ascending") as follows:

$$[asc.z \;=\; (\forall k :: \text{stable.}(z \geq k))]$$

and for brevity we extend the definitions to lists of integer variables as follows

$$[asc.(y_0, \ldots, y_n) = (\forall i : 0 \leq i \leq n : asc.y_i)]$$

In this section i is implicitly quantified over $0..n$, and k and m_i are quantified over all natural numbers.

Properties of Professors Variables of P_i are x and y_i, where x is not modified by P_i and y_i is not decreased by P_i. We are given the following properties for P_i:

$$(\forall m :: \text{stable.}(x = m))$$

$$asc.y_i$$

$$((\forall k :: (x \leq k) \wedge (y_i \leq f_i(k))) \quad co \quad (y_i \leq f_i(k)))$$

$$asc(x, y_i) \quad \succ \quad (\forall k :: (x \geq k) \hookrightarrow (y_i \geq f_i(k)))$$

Properties of the Secretary Variables of *Sec* are x and y_i, where *Sec* does not modify y_i, all i, and does not decrease x. We are given the following properties for *Sec*:

$$(\forall i :: \text{stable.}(y_i = m_i))$$

$$asc(x)$$

$$(\forall k :: (\forall i :: y_i \leq k) \wedge (x \leq k) \;\; co \;\; (x \leq k))$$

$$asc(x, y_0, \ldots, y_n) \;\succ\; (\forall k :: (\forall i :: y_i \geq k) \hookrightarrow (x \geq k))$$

Specification Define program *dept* as:

$$dept = Sec \| P_0 \| \ldots \| P_n$$

Our proof obligation is to show that *dept* has the following property:

$$(0 \leq x \leq e) \wedge (\forall i :: 0 \leq y_i \leq e) \;\hookrightarrow\; (x = e) \wedge (\forall i :: y_i = e) \qquad (18)$$

7.2 Proof that the Program satisfies its Specifications

Lemma

$(\forall i, k :: (x \geq k \;\hookrightarrow\; y_i \geq f_i(k)) \wedge (y_i \geq k \hookrightarrow x \geq k)) \;\Rightarrow$
$\quad x \geq 0 \hookrightarrow (x \geq e) \wedge (\forall i :: y_i \geq e)$

Proof:

$(\forall i, k :: (x \geq k \;\hookrightarrow\; y_i \geq f_i(k)) \wedge (y_i \geq k \hookrightarrow x \geq k))$

$\Rightarrow$ { transitivity of $\hookrightarrow$, and $k := f_i(k)$ in the second conjunct }

$(\forall i, k :: x \geq k \;\hookrightarrow\; x \geq f_i(k))$

$\equiv$ { conjunctivity with respect to second operand }

$(\forall k :: x \geq k \;\hookrightarrow\; (\forall i :: x \geq f_i(k)))$

$\equiv$ { predicate calculus }

$(\forall k :: x \geq k \;\hookrightarrow\; x \geq \max_i f_i(k))$

$\equiv$ { definition of g }

$(\forall k :: x \geq k \;\hookrightarrow\; x \geq g(k))$

$\Rightarrow$ { transitivity of $\hookrightarrow$ and induction on j}

$(\forall k, j : \text{finite } j : x \geq k \;\hookrightarrow\; x \geq g^{(j)}(k))$

$\Rightarrow$ { $k := 0$ and $j := e$, and (17) }

$x \geq 0 \;\hookrightarrow\; x \geq e$

$\Rightarrow$ { transitivity with $x \geq e \;\hookrightarrow\; y_i \geq f_i(e) = e$ }

$(\forall i :: x \geq 0 \;\hookrightarrow\; y_i \geq e)$

$\Rightarrow$ { conjunction of the last two formulae }

$x \geq 0 \;\hookrightarrow\; (x \geq e) \wedge (\forall i :: y_i \geq e)$

From Theorem 7, and properties of P_i all i, and *Sec*, the following property holds for *dept*:

$$asc(x, y_0, \ldots, y_n) \;\succ\; (\forall i, k :: (x \geq k \;\hookrightarrow\; y_i \geq f_i(k)) \wedge (y_i \geq k \hookrightarrow x \geq k))$$

From the last Lemma, and monotonicity of $\succ$ with respect to its second argument, we have the following property for *dept*:

$$asc(x, y_0, \ldots, y_n) \;\succ\; (x \geq 0 \;\hookrightarrow\; (x \geq e) \wedge (\forall i :: y_i \geq e)) \tag{19}$$

We prove that *dept* has the properties:

$$asc(x, y_0, \ldots, y_n) \tag{20}$$

$$\text{stable.}((x \leq e) \wedge (\forall i :: y_i \leq e)) \tag{21}$$

by proving that each of its components has the properties (since these properties are all-components properties). From (16) and (21), *dept* has the following property:

$$((x \leq e) \wedge (\forall i :: y_i \leq e)) \hookrightarrow ((x \leq e) \wedge (\forall i :: y_i \leq e))$$

From (19) and (20), *dept* has the property:

$$x \geq 0 \;\hookrightarrow\; (x \geq e) \wedge (\forall i :: y_i \geq e))$$

The desired property (18) follows from the conjunction of the last two formulae and weakening the left side.

8 Acknowledgment

The paper was completely rewritten based on constructive criticism from J. Misra, E.W. Dijkstra and B. Sanders. Thanks to suggestions from Rustan Leino and many at Caltech. Finally, many thanks to Cliff Jones and David Gries for careful reading and key insights.

References

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

[2] Dijkstra, E.W., and C.S. Scholten, *Predicate Calculus and Program Semantics*, Springer-Verlag, New York, 1990.

[3] Hoare, C. A. R., An Axiomatic Basis for Computer Programming, *CACM*, 12(1969) 576-580.

[4] Jones, C.B., Specification and Design of (Parallel) Programs. *Proc.IFIP'83*, North-Holland,321-332, 1983

[5] Misra, J., Safety Properties, Computer Sciences Dept., Univ. of Texas, Austin, TX78712, July 24, 1992.