The Importance of Ensuring
Notes on UNITY: 11-90

Jayadev Misra*
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712
(512) 471-9547
misra@cs.utexas.edu

1/11/90

The definition of *ensures* in a UNITY program is tied to the set of statements in the program. This seems to be too restrictive because "semantically equivalent" programs that consist of different statements may not have the same *ensures* properties: In this context, two programs are semantically equivalent if all possible sequence of states, arising in fair executions, are identical for both programs (formal definitions are given below). Thus, semantically equivalent programs have the same *unless* and *leads-to* properties. We show that this notion of semantic equivalence is too coarse. In particular, we show two semantically equivalent programs that exhibit different behavior when composed with a third program; *ensures* properties of the two programs, however, are distinct. We claim that *ensures* provides a finer distinction that is essential if we have to deduce progress properties of a composite program from the properties of its components.

Consider a program F, whose state transitions are shown pictorially in Figure 1. The program has three possible states—0,1,2—and two transitions (i.e., statements)—α, β. Any transition can be applied in any state and it results in a unique next state; a transition that leaves the state unchanged is not shown in Figure 1. The initial state is 0.

A *fair execution sequence* is an infinite sequence of transitions in which each transition—α and β for F—appears infinitely often. A *fair history*, corresponding to an initial state and a fair execution sequence, is an infinite sequence of states in which the first state is

*This work was partially supported by ONR Contract 26-0679-4200 and by Texas Advanced Research Program grant 003658–065.

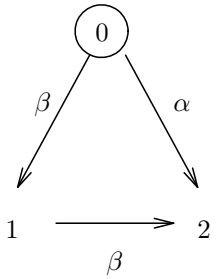

Figure 1: State transitions in Program F

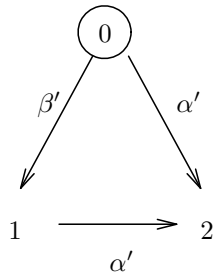

Figure 2: Program G

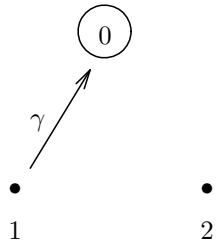

Figure 3: Program H

the given initial state and each subsequent state is obtained by applying the next transition from the given fair execution sequence to the current state.

Observation: Every fair history of program F is of the form $O\ 1^*\ 2^w$, and vice versa (where 1^* is a finite sequence of 1s and 2^w is the infinite sequence of 2s).

Consider program G given in Figure 2. It has the same set of states and the same initial state as F. It has two transitions, α' and β', that *nearly* correspond to α, β, respectively: In every state except state 1, (α, α') and (β, β') result in identical next states.
Observation: Every fair history of program G is of the form $0\ 1^*\ 2^w$, and vice versa.

Thus, programs F and G cannot be distinguished by their fair histories. In particular, they have the same set of *unless* and *leads-to* properties. Now consider program H given in Figure 3; it has one transition—γ—and it has the same states: 0,1,2.

Consider compositions of programs F, H and G, H. The composition of F, H—written as $F \parallel H$ in UNITY—consists of the three transitions α, β, γ; similarly, for $G \parallel H$.

Observation: Fair histories of $F \parallel H$ and $G \parallel H$ are different.

Proof: We show that $(0\ 1\ 1)^w$ is a fair history of $F \parallel H$, but it is not a fair history of $G \parallel H$. To see that $(0\ 1\ 1)^w$ is a fair history of $F \parallel H$ consider its fair execution sequence $(\beta\ \alpha\ \gamma)^w$. To see that $(0\ 1\ 1)^w$ is not a fair history of $G \parallel H$: Every fair execution sequence in $G \parallel H$ contains α'—by definition—and application of α' in state 0 or state 1 results in state 2; hence, every fair history of $G \parallel H$ contains a "2". □

From this proof it is clear that state 2 is eventually reached in every fair execution in $G \parallel H$, though this claim is not true for $F \parallel H$. Equating programs by their fair histories is too coarse for such deductions. Programs F, G are distinguished by *ensures*:

(State 0 $\vee$ State 1) *ensures* State 2 in G

A similar propety does not hold in F. Also we have

(State 0 $\vee$ State 1) stable in H.

Hence, we can conclude, applying the union theorem, that

(State 0 $\vee$ State 1) *ensures* State 2 in $G \parallel H$

and hence,

(State 0 $\vee$ State 1) $\mapsto$ State 2 in $G \parallel H$.

No such property holds for $F \parallel H$.