A Simple Language

Syntax

Sequential Constructs

Given a statement P, a labeled statement P^L is defined as follows:

- If P is a simple statement (x: = e, skip, wait, lock, unlock etc.), P^L = P;
- If P = P1;P2, then P^L = $P1^L$; l'': $P2^L$;
- If P = **if** b **then** P1 **else** P2 **end if**, then P^L = **if** b **then** l1**:** P1 **else** l2: P2 **end if**,
- If P = **while** b **do** P1 **end while**, then P^L = **while** b **do** l1: P1 **end while**.

Concurrent Construct

- If P = **cobegin** P1 || P2 || … || Pn **coend**, then
  P^L = **cobegin** l1: $P1^L$ l1'|| l2: $P2^L$ l2'|| … || ln: Pn^L ln' **coend**,

Semantics Defined as State Transitions

Let pc be a special variable (program counter) that ranges over the set of program labels and a special value ⊥ (indicating the program is not active). Let V denote the set of program variables, and V' be its primed version. m and m' denote the entry and exit points; pre(V) denote the initial values of V.

The Initial state:

$S0(V, pc) \equiv pre(V) \wedge pc = m$

The translation is defined as a set of rules, one for each statement type in terms of a tuple C(l, P, l') as follows:

- Assignment: C(l, v ← e, l') ≡ pc = l ∧ pc' = l' ∧ v' = e ∧ same(V\{v})
- Skip: C(l, skip, l') ≡ pc = l ∧ pc' = l' ∧ same(V)
- Sequential Composition: C(l, P1; l'':P2, l') ≡ C(l, P1; l'') ∨ C(l'', P2, l')
- Conditional: C(l, **if** b **then** l1: P1 **else** l2: P2 **end if**, l') ≡ is the disjunction of the following

  (pc = l ∧ pc' = l1 ∧ b ∧ same (V)) ∨ (pc = l ∧ pc' = l2 ∧ ¬b ∧ same (V)) ∨ C(l1, P1, l') ∨ C(l2, P2, l')
- While: C(l, **while** b **do** l1: P1 **end while**, l') ≡ is the disjunction of the following

  (pc = l ∧ pc' = l1 ∧ b ∧ same (V)) ∨ (pc = l ∧ pc' = l' ∧ ¬b ∧ same (V)) ∨ C(l1, P1, l)

Concurrent Programs (Interleaving Execution)

The Initial State

S0(V, PC) ≡ pre(V) ∧ pc = m $\bigwedge_{i=1}^{n}(pc_i = \perp)$

- C(l, **cobegin** l1: P1 ^L^ l1'|| l2: P2 ^L^ l2'|| … || ln: Pn ^L^ ln'**coend**, l') is the disjunction of:

$(pc = l \wedge pc_1' = l_1 \wedge \cdots \wedge pc_n' = l_n \wedge pc' = \perp) \vee$ //entry point of the statement

$(pc = \perp \wedge pc_1 = l_1' \wedge \cdots \wedge pc_n = l_n' \wedge pc' = l' \bigwedge_{i=1}^{n} pc_i' = \perp) \vee$ //termination state

$\bigvee_{i=1}^{n}(C(l_i, P_i, l_i') \wedge same(V \setminus V_i) \wedge same(PC \setminus \{pc_i\}))$ //interleaved execution

Shared Variables

- Wait: C(l, wait(b), l') is the disjunction of //wait (v=0)

  $(pc_i = l \wedge pc_i' = l \wedge \neg b \wedge same(V_i)) \vee (pc_i = l \wedge pc_i' = l' \wedge b \wedge same(V_i))$

- Lock: C(l, lock(v), l') is the disjunction of

  $(pc_i = l \wedge pc_i' = l \wedge v = 1 \wedge same(V_i)) \vee (pc_i = l \wedge pc_i' = l' \wedge v = 0 \wedge v' = 1 \wedge same(V_i \setminus \{v\}))$

- Unlck: C(l, unlock(v), l') $\equiv (pc_i = l \wedge pc_i' = l' \wedge v' = 0 \wedge same(V_i \setminus \{v\}))$

  // locked (v=1)