The UNITY Programming Logic

Wolfgang Schreiner
Research Institute for Symbolic Computation
Johannes Kepler University, Linz, Austria

Wolfgang.Schreiner@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at/people/schreine

The UNITY Programming Logic

Logic for

- Specification,
- Design,
- Verification.

Assertions $\{p\}s\{q\}$

- Execution of statement s.
- If p holds before s,
- then q holds after s.

Axiomatic Logic

- We do not say what P *is*.
- We say what we can *expect* from P.

Assignment Statements

To prove

$$\{p\}x := E\{q\}$$

it suffices to show

$$p \Rightarrow q^x_E$$

(where q^x_E is q with E substituted for all occurences of x).

To prove

$$\{p\}x := e_0 \text{ if } b_0 \text{ ~ } \ldots \text{~ } e_n \text{ if } b_n \ \{q\}$$

it suffices to prove

$$\{p \wedge b_0\}x := e_0\{q\}$$
$$\ldots$$
$$\{p \wedge b_n\}x := e_n\{q\}$$
$$p \wedge (\neg b_0 \wedge \ldots \wedge \neg b_n) \Rightarrow \{q\}$$

Quantified Assertions

For program F, we write quantified assertions

- $\langle \forall\ s : s$ in $F :: \{p\}s\{q\}\rangle$
- $\langle \exists\ s : s$ in $F :: \{p\}s\{q\}\rangle$

to denote that $\{p\}s\{q\}$ holds for all statements (respectively for some statement) s in F.

- $\langle \forall\ s : s$ in $\langle [] \ i : b(i) :: t(i)\rangle :: \{p\}s\{q\}\rangle$
  - Show $\{p \wedge b(i)\}t(i)\{q\}$.
- $\langle \exists\ s : s$ in $\langle [] \ i : b(i) :: t(i)\rangle :: \{p\}s\{q\}\rangle$
  - Show $\{p \wedge b(k)\}t(k)\{q\}$
    (for some j with $b(j)$).

A Model of Program Execution

Formalize operational view of program execution.

- Set of (infinite) *execution sequences*.
  - Sequence R, i-th element R_i.
  - $R_i = (R_i.$*state*, $R_i.$*label*$)$.
  - $R_i.$*label* denotes statement selected for execution in i-th step.
  - $R_i.$*state* is program state (values of program variables) before i-th step.
- Sequence construction:
  - Initial state $R_0.$*state* (need not be the same for all sequences).
  - $R_i.$*state* and $R_i.$*label* determine $R_{i+1}.$*state*.
  - $R_0.$*label* and $R_j.$*state* (for all $j < i$) determine $R_i.$*state*.

Application of Execution Model

- Fair selection condition:
  - R_i.*label* $= s$
    (for an infinite number of i and any R and s).
- $p[R_i]$
  - p holds in R_i.*state*.
- $\{p\}s\{q\}$
  - $(p[R_i] \wedge R_i$.*label* $= s) \Rightarrow q[R_{i+1}]$
    (for all R and i).

Program execution model used to formalize operational understanding of program execution, not to reason about program properties!

Fundamental Concepts

Three fundamental logical relations

- Safety: p *unless* q.
  - Special case: **stable** p.
  - Special case: **invariant** p.
- Progress: p *ensures* q
- Progress: $p \mapsto q$ (p *leads-to* q)

Safety and progress properties.

Unless

p *unless* $q \equiv \langle \forall s : s$ in $F :: \{p \wedge \neg q\} s \{p \vee q\} \rangle$

- If p is true at some point in program execution, then in the next step
  - either p remains true,
  - or q becomes true.
- If p holds at any point during execution of F, then
  - either q never holds and p continues to hold forever,
  - or q holds eventually and p continues to hold at least until q holds.
- Program execution model:
  - $(p \wedge \neg q)[R_i] \Rightarrow (p \vee q)[R_{i+1}]$.
  - $p[R_i] \Rightarrow$
    $\langle \forall j : j \geq i :: (p \wedge \neg q)[R_j]) \rangle$
    $\vee\ [\langle \exists k : k \geq i :: q[R_k]$
    $\wedge\ \langle \forall j : i \leq j < k :: (p \wedge \neg q)[R_j]) \rangle]$

Special Cases of Unless

- p is *stable* $\equiv$ p *unless* false.
  - Stable predicates remain true once it becomes true (but it may never become true).
- q is *invariant* $\equiv$ (initial condition $\Rightarrow q$) $\wedge$ q is stable.
  - Invariant is always true from the beginning.
- **constant** p
  - p and $\neg p$ are stable.

Stable p is prserved by every statement s of the program.

Special Cases of Unless

- $$\frac{p \textit{ unless } q \quad \neg q \text{ is invariant}}{p \text{ is stable}}$$
  - $\neg q \equiv$ true ($\neg q$ is invariant).
  - p *unless* q (premise),
  - p *unless* false (substitution axiom on above two),
  - p is stable (definition).
- Always-Section
  - Defines an invariant.
  - Construction like that of initial condition by initially-section.

Ensures

p *ensures* $q \equiv (p$ *unless* q
$\wedge \langle \exists s : s \text{ in } F :: \{p \wedge \neg q\}\ s\ \{q\}\rangle)$

- If p is true at some point in computation, then
  - p remains true as long as q is false,
  - and eventually q becomes true.
- Program execution model:
  - $p[R_i] \Rightarrow$
    $\langle \exists j : j \geq i :: q[R_j]$
    $\wedge \langle \forall k : i \leq j < k :: p[R_k]\rangle\rangle$

Leads-to

$p \mapsto q$

- Derived by a finite number of applications of the following inference rules:
  - $\dfrac{p \textit{ ensures } q}{p \mapsto q}$
  - $\dfrac{p \mapsto q \quad q \mapsto r}{p \mapsto r}$
  - $\dfrac{\langle \forall m : m \in W :: p(m) \mapsto q \rangle}{\langle \exists m : m \in W :: p(m) \rangle \mapsto q}$
- Once that p becomes true, q is or will be true.
- Cannot assure that p will remain true as long as q is not.
- Program execution model:
  - $p[R_i] \Rightarrow \langle \exists j : j \geq i :: q[R_j] \rangle$

Fixed Point

- *Fixed Point* = program state
  - Execution of any statement in that state leaves state unchanged.
  - Once reached, program remains in this state.
- Equivalent to *termination* in sequential programming.
- Predicate *FP* for program *G*:
  - $\mathit{FP} \equiv \langle \forall s : s \text{ in } G \wedge s \text{ is } \mathsf{X} := E :: X = E \rangle$.
- Program execution model:
  - $\mathit{FP}[R_i] \equiv \langle \forall j : j \geq i :: R_i.\mathit{state} = R_j.\mathit{state} \rangle$

An Example

Program *division*
  declare x, y, z, k: integer
  initially $x, y, z, k = 0, M, N, 1$
  assign
    $z, k :=$
      $2 * z$, $2 * k$ if $y \geq 2 * z$ ~
      $N, 1$ if $y < 2 * z$
  [] $x, y := x + k, y - z$ if $y \geq z$
end $\{$*division*$\}$

- Specification
  - $x * N + y = M \wedge 0 \leq y < N$ holds in fixed point.
  - fixed point is reached eventually.

Proofs

- Invariant *I*
  - $y \geq 0 \wedge k \geq 1 \wedge z = N * k \wedge x * N + y = M.$
- Fixed point condition *FP*
  - $(y < 2 * z \vee (z = 2 * z \wedge k = 2 * k)) \wedge$
    $(y \geq 2 * z \vee (z = N \wedge k = 1)) \wedge$
    $(y < z \vee (x = x + k \wedge y = y - z))$
- *I* and *FP* imply
  - $x * N + y = M \wedge 0 \leq y < N.$
  - true $\mapsto$ *FP* (see later).

Program meets its specification!

Invariant Proof

- Show that initial condition implies *I*.
  - $M \geq 0 \wedge 1 \geq 1 \wedge N = N * 1 \wedge 0 * N + M = M$.
- Show that *I* is stable.
  - $\{I \wedge y \geq 2 * z\}\ z, k := 2 * z, 2 * k\ \{I\}$
  - $\{I \wedge y < 2 * z\}\ z, k := N, 1\ \{I\}$
  - $\{I \wedge y \geq z\}\ x, y := x + k, y - z\ \{I\}$

Application of basic rules.

Fixed Point Proof

- Show (true $\mapsto$ *FP*):
  - Know *I* $\wedge\ y < z \leq N \Rightarrow$ *FP*.
  - Suffices to show (true $\mapsto y < z \leq N$).
- Show (true $\mapsto y < z \leq N$).
  - Find well-founded ordering $\sqsubset$ with $(y,z)=(m,n) \wedge \neg(y < z \leq N) \mapsto (y,z)\sqsubset(m,n)$.
  - By induction, $\neg(y < z \leq N)$ cannot hold forever.

Find inductive term.

Fixed Point Proof

- Let $\sqsubset$ be the lexicographic ordering.
  - $(y,z) \sqsubset (m,n) := (y < m \vee (y = m \wedge z < n))$.
  - Show $(y,z)=(m,n) \wedge \neg(y < z \leq N) \mapsto (y < m \vee (y = m \wedge z < n))$.
  - General disjunction theorem for $\mapsto$.
- Show $(y,z)=(m,n) \wedge y \geq z \mapsto y < m$.
  - Suffices to show $y \geq z \wedge y = m$ *ensures* $y < m$.
- Show $(y,z)=(m,n) \wedge y < z \wedge z > N \mapsto (y = m \wedge z < n)$.
  - Suffices to show $(y,z)=(m,n) \wedge y < z \wedge z > N$ *ensures* $(y = m \wedge z < n)$.

Proof of Ensure Relations

- Show $y{\geq}z \wedge y{=}m$ ***ensures*** $y < m$.
  - Show $y{\geq}z \wedge y{=}m$ ***unless*** $y < m$ (later).
  - Show $\{y{\geq}z \wedge y{=}m\}$ $x{,}y := x{+}k{,}y{-}z$ if $y{\geq}z$ $\{y < m\}$ (easy).

Show that there exists one statement that establishes conclusion from state where premise but not conclusion holds.

Proof of Unless Relations

- Show $y \geq z \wedge y = m$ *unless* $y < m$.
  - Show $\{(y \geq z \wedge y = m) \wedge y \geq m\}$ s
    $\{(y \geq z \wedge y = m) \vee y, m\}$
    for every s.
- $\{(y \geq z \wedge y = m) \wedge y \geq m\}$
  $z, k := 2 * z, 2 * k$ if $y \geq 2 * z$ ~
  $\quad N, 1$ if $y < 2 * z$
  $\{(y \geq z \wedge y = m) \vee y, m\}$
  - $\{y \geq 2 * z \wedge (y \geq z \wedge y = m) \wedge y \geq m\}$
    $z, k := 2 * z, 2 * k$
    $\{(y \geq z \wedge y = m) \vee y, m\}$
  - $\{y < 2 * z \wedge (y \geq z \wedge y = m) \wedge y \geq m\}$
    $z, k := N, 1$
    $\{(y \geq z \wedge y = m) \vee y, m\}$

Show that in all statements premise or conclusion holds from state where premise but not conclusion holds.