Substitution and Consistency Proof in Event-B

© Michael Butler

University of Southampton

May 22, 2010

Substitution

Substitute variable x with expression E in predicate Q:

$$[x := E]\ Q$$

In other words, replace all occurrences of variable x in predicate Q with expression E.

Example:

$$[n := 7]\ (\ 0 < n \ \wedge \ n \leq 10\ ) \quad = \quad 0 < 7 \ \wedge \ 7 \leq 10$$

Multiple Substitution

Can also have multiple substitution:

$$[\ x_1, x_2, \ldots, x_n\ :=\ E_1, E_2, \ldots, E_n\ ]\ Q$$

Example:

$$[\ l, m, n := 0, 10, 7\ ]\ (\ l < n\ \wedge\ n \leq m\ )\quad =\quad 0 < 7\ \wedge\ 7 \leq 10$$

$$[\ x, y := y, x\ ]\ (\ x \in S\ \wedge\ y \in T\ )\quad =\quad ?$$

$$[\ in,\ out\ :=\ in \setminus \{u\},\ out \cup \{u\}\ ]\ (\ in \cap out = \{\}\ )\quad =\quad ?$$

Preconditions and Postconditions

Precondition: holds before executing an assignment.

Postcondition: holds after executing an assignment.

Given an assignment A and a postcondition Q, can we determine the precondition under which A will establish Q?

Example:

Assignment: $x := x - 1$

Postcondition: $x \geq 0$

Under what *precondition* will $x := x - 1$ establish $x \geq 0$?

Computing a Precondition

Preconditions can be computed from postconditions by treating an assignment as a substitution.

Substitution $[A]\ Q$ is the precondition under which assignment A will establish postcondition Q.

Example:

$$\begin{aligned} [x := x - 1]\ (x \geq 0) \quad &= \quad x - 1 \geq 0 \\ &= \quad x \geq 1 \end{aligned}$$

$x := x - 1$ establishes postcondition $x \geq 0$ provided $x \geq 1$ holds beforehand.

Machine consistency (preserving invariants)

A machine is consistent if all its events preserve the invariants.

An assignment A preserves an invariant, if:

$$I \implies [A]\ I$$

I is both a precondition and a postcondition.

Executing A will establish I provided I is true beforehand.

Example:

$$x > 0 \implies [x := x + 1](x > 0)$$

Example

An assignment S *preserves an invariant*, if:

$$I \quad \Longrightarrow \quad [A]\ I$$

Invariant: $x + y = C$

Assignment: $x, y := x + 1, y - 1$

Prove that the assignment preserves the invariant.

Using Event Guards

The guards of an event can be used as assumptions when proving tht the action of an event preserves invariants.

$$Ev \quad = \quad \textbf{any } x \textbf{ where } \quad G \quad \textbf{then } \quad A \quad \textbf{end}$$

The event Ev will preserve invariant I if:

$$I \ \wedge \ G \qquad \Longrightarrow \qquad [A] \ I$$

Executing A will establish I provided both G and I are true beforehand.

Example:

Invariant: $ctr \leq max$

Event: **when** $ctr < max$ **then** $ctr := ctr + 1$ **end**

Example

$$Inv1 \;:\; in \cap out = \{\}$$

$$\begin{array}{lll} leave & \triangleq & \textbf{any } s \textbf{ where} \\ & & \quad s \in in \\ & & \textbf{then} \\ & & \quad in,\ out \;:=\; in \setminus \{s\},\ out \cup \{s\} \\ & & \textbf{end} \end{array}$$

Show that *leave* preserves *Inv*1.

Compositionality Principle

If

$$I1 \wedge I2 \quad \Longrightarrow \quad [A]\ I1$$

and

$$I1 \wedge I2 \quad \Longrightarrow \quad [A]\ I2$$

then

$$I1 \wedge I2 \quad \Longrightarrow \quad [A]\ (I1 \wedge I2)$$

The compositionality principle means that consistency can be proved separately for each invariant.

Example

$$\begin{array}{lcll} Inv1 & : & in \cap out & = & \{\} \\ Inv2 & : & in \cup out & = & register \end{array}$$

Can show that

leave preserves $Inv1$

leave preserves $Inv2$

hence by the Compositionality Principle:

leave preserves $Inv1 \wedge Inv2$.

Rodin tool checks consistency

The Rodin tool generates consistency proof obligations for each event and each invariant.

Given an event Ev,

$$Ev \quad = \quad \textbf{any } x \textbf{ where } \quad G \quad \textbf{then } \quad A \quad \textbf{end}$$

a consistency proof obligation for invariant I and event Ev is an obligation to prove the following theorem:

$$J \ \wedge \ I \ \wedge \ G \qquad \Longrightarrow \qquad [A] \ I$$

J represents the other invariants (besides I) of the machine being checked.

Consider the following B invariant and operation:

$$Inv1 \ : \quad in \cup out = register$$

$$\begin{array}{lll} leave & \widehat{=} & \textbf{any } s \textbf{ where} \\ & & \quad s \in in \\ & & \textbf{then} \\ & & \quad in,\ out \ := \ in \setminus \{s\},\ out \cup \{s\} \\ & & \textbf{end} \end{array}$$

Prove that *leave* preserves $Inv1$.
You can assume the following law

$$x \in A \quad \Longrightarrow \quad (A \setminus \{x\}) \ \cup \ (B \cup \{x\}) \ = \ A \cup B \qquad (1)$$