CS 4110

Programming Languages & Logics

Lecture 11
Weakest Preconditions

Review: Decorating Programs

$\{\textbf{true}\}$

$$
\begin{array}{l}
\mathsf{x} := \mathsf{m}; \\
\mathsf{y} := 0; \\
\textbf{while}\ (\mathsf{n} < \mathsf{x})\ \textbf{do}\ (\\
\quad \mathsf{x} := \mathsf{x} - \mathsf{n}; \\
\quad \mathsf{y} := \mathsf{y} + 1 \\
)
\end{array}
$$

$\{ \qquad\qquad\qquad\qquad \}$

Review: Decorating Programs

$\{\textbf{true}\}$

$$
\begin{array}{l}
x := m; \\
y := 0; \\
\textbf{while}\ (n < x)\ \textbf{do}\ (\\
\quad x := x - n; \\
\quad y := y + 1 \\
)
\end{array}
$$

$\{n \times y + x = m\}$

In other words, the program divides m by n, so y is the quotient and x is the remainder.

Generating Preconditions

To fill in a precondition:

$$\{\quad\}\ c\ \{Q\}$$

there are many possible preconditions—and some are more useful than others.

Weakest Preconditions

Intuition: The weakest liberal precondition for c and Q is the weakest assertion P such that $\{P\}\ c\ \{Q\}$ is valid.

Weakest Preconditions

Intuition: The weakest liberal precondition for c and Q is the weakest assertion P such that $\{P\}\ c\ \{Q\}$ is valid.

More formally...

Definition (Weakest Liberal Precondition)

P is a weakest liberal precondition of c and Q written $wlp(c, Q)$ if:

$$\forall \sigma, I.\ \sigma \vDash_I P \iff (\mathcal{C}[\![c]\!]\ \sigma) \text{ undefined } \vee (\mathcal{C}[\![c]\!]\sigma) \vDash_I Q$$

Weakest Preconditions

$$wlp(\mathbf{skip}, P) \quad = \quad P$$

Weakest Preconditions

$$
\begin{array}{rcl}
wlp(\textbf{skip}, P) & = & P \\
wlp(\mathsf{x} := a, P) & = & P[a/\mathsf{x}]
\end{array}
$$

Weakest Preconditions

$$
\begin{array}{rcl}
wlp(\textbf{skip}, P) & = & P \\
wlp(\mathsf{x} := a, P) & = & P[a/\mathsf{x}] \\
wlp((c_1; c_2), P) & = & wlp(c_1, wlp(c_2, P))
\end{array}
$$

Weakest Preconditions

$$
\begin{array}{rcl}
wlp(\mathbf{skip}, P) & = & P \\
wlp(\mathrm{x} := a, P) & = & P[a/\mathrm{x}] \\
wlp((c_1; c_2), P) & = & wlp(c_1, wlp(c_2, P)) \\
wlp(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, P) & = & (b \implies wlp(c_1, P)) \wedge \\
 & & (\neg b \implies wlp(c_2, P))
\end{array}
$$

Weakest Preconditions

$$
\begin{array}{rcl}
wlp(\textbf{skip}, P) & = & P \\
wlp(\mathrm{x} := a, P) & = & P[a/\mathrm{x}] \\
wlp((c_1; c_2), P) & = & wlp(c_1, wlp(c_2, P)) \\
wlp(\textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2, P) & = & (b \implies wlp(c_1, P)) \wedge \\
 & & (\neg b \implies wlp(c_2, P)) \\
wlp(\textbf{while } b \textbf{ do } c, P) & = & \bigwedge_i F_i(P)
\end{array}
$$

Weakest Preconditions

$$
\begin{array}{rcl}
wlp(\textbf{skip}, P) & = & P \\
wlp(\mathsf{x} := a, P) & = & P[a/\mathsf{x}] \\
wlp((c_1; c_2), P) & = & wlp(c_1, wlp(c_2, P)) \\
wlp(\textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2, P) & = & (b \implies wlp(c_1, P)) \wedge \\
 & & (\neg b \implies wlp(c_2, P)) \\
wlp(\textbf{while } b \textbf{ do } c, P) & = & \bigwedge_i F_i(P)
\end{array}
$$

where

$$
\begin{array}{rcl}
F_0(P) & = & \textbf{true} \\
F_{i+1}(P) & = & (\neg b \implies P) \wedge (b \implies wlp(c, F_i(P)))
\end{array}
$$

Applications of Weakest Preconditions

Failing fast: avoid wasting work on bad inputs.

p := getPacket();
processPacket(p);
assert P_{safe}

Applications of Weakest Preconditions

Failing fast: avoid wasting work on bad inputs.

$\text{p} := \text{getPacket}();$
$\text{processPacket}(\text{p});$
$\{P_{\text{safe}}\}$

Applications of Weakest Preconditions

Failing fast: avoid wasting work on bad inputs.

$$\begin{array}{l} \text{p} := \text{getPacket}(); \\ \{P_{\text{filter}}(p)\}; \\ \text{processPacket}(\text{p}); \\ \{P_{\text{safe}}\} \end{array}$$

Applications of Weakest Preconditions

Failing fast: avoid wasting work on bad inputs.

p $:=$ getPacket();
assert $P_{\text{filter}}(p)$;
processPacket(p);

Applications of Weakest Preconditions

Failing fast: avoid wasting work on bad inputs.

p := getPacket();
assert $P_{\text{filter}}(p)$;
processPacket(p);

P_{filter} should be the *weakest* precondition to avoid ruling out legitimate inputs.

David Brumley, Hao Wang, Somesh Jha, and Dawn Song. “Creating Vulnerability Signatures Using Weakest Preconditions.” In *Computer Security Foundations* (CSF), 2007.

Properties of Weakest Preconditions

Lemma (Correctness of Weakest Preconditions)

$$
\begin{aligned}
&\forall c \in \textbf{Com}, Q \in \textbf{Assn}.\\
&\qquad \vDash \{wlp(c, Q)\}\; c\; \{Q\}\;\; \textit{and}\\
&\qquad \forall R \in \textbf{Assn}.\;\; \vDash \{R\}\; c\; \{Q\}\; \textit{implies}\; (R \implies wlp(c, Q))
\end{aligned}
$$

Properties of Weakest Preconditions

Lemma (Correctness of Weakest Preconditions)

$$\begin{array}{l}\forall c \in \textbf{Com}, Q \in \textbf{Assn}. \\ \qquad \vDash \{wlp(c, Q)\}\ c\ \{Q\}\ \ \textit{and} \\ \qquad \forall R \in \textbf{Assn}.\ \vDash \{R\}\ c\ \{Q\}\ \textit{implies}\ (R \implies wlp(c, Q))\end{array}$$

Lemma (Provability of Weakest Preconditions)

$$\forall c \in \textbf{Com}, Q \in \textbf{Assn}.\ \vdash \{wlp(c, Q)\}\ c\ \{Q\}$$

Soundness and Completeness

Soundness: If we can prove it, then it’s actually true.

Completeness: If it’s true, then a proof exists.

Soundness and Completeness

Soundness: If we can prove it, then it's actually true.

Definition (Soundness)

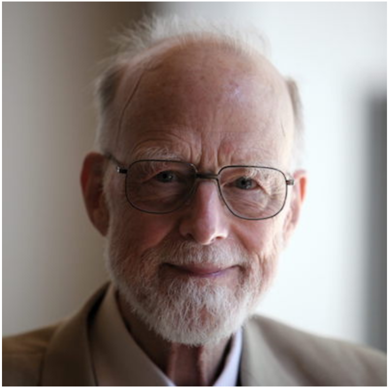
If $\vdash \{P\}\, c\, \{Q\}$ then $\models \{P\}\, c\, \{Q\}$.

Completeness: If it's true, then a proof exists.

Definition (Completeness)

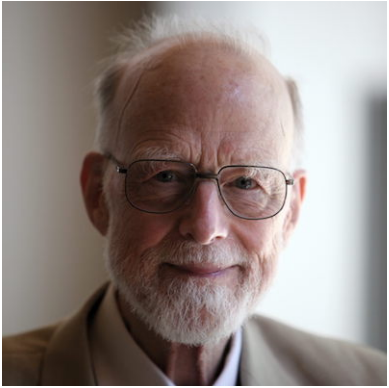
If $\models \{P\}\, c\, \{Q\}$ then $\vdash \{P\}\, c\, \{Q\}$.

vs.



Kurt Gödel

vs.



Sir Tony Hoare

Relative Completeness

Theorem (Cook (1974))

$\forall P, Q \in \mathbf{Assn}, c \in \mathbf{Com}.\ \vDash \{P\}\, c\, \{Q\}$ *implies* $\vdash \{P\}\, c\, \{Q\}$.

Relative Completeness

Theorem (Cook (1974))

$\forall P, Q \in \textbf{Assn}, c \in \textbf{Com}.\ \vDash \{P\}\, c\, \{Q\}$ *implies* $\vdash \{P\}\, c\, \{Q\}$.

Proof Sketch.

Let $\{P\}\, c\, \{Q\}$ be a valid partial correctness specification.

By the first Lemma we have $\vDash P \implies wlp(c, Q)$.

By the second Lemma we have $\vdash \{wlp(c, Q)\}\, c\, \{Q\}$.

We conclude $\vdash \{P\}\, c\, \{Q\}$ using the CONSEQUENCE rule. $\square$