Weakest Preconditions

COMPSCI 631

University of Massachusetts Amherst

October 24, 2017

Recap: The WHILE Language (Syntax)

Arithmetic Expressions

$$
\begin{array}{lcl}
aexp & ::= & n \\
& | & x \\
& | & a_1 + a_2 \\
& | & a_1 - a_2 \\
& | & a_1 * a_2
\end{array}
$$

Boolean Expressions

$$
\begin{array}{lcl}
bexp & ::= & \textbf{true} \\
& | & b1 \ \&\& b2 \\
& | & a_1 > a_2 \\
& & \ldots
\end{array}
$$

Commands

$$
\begin{array}{lcl}
cmd & ::= & \textbf{skip} \\
& | & \textbf{abort} \\
& | & x := a \\
& | & c_1 ;\ c_2 \\
& | & \textbf{if} \ \ (b) \ \textbf{then} \ c_1 \textbf{else} \ \ c_2 \\
& | & \textbf{while} \ \ (b) \ \ c
\end{array}
$$

Recap: The WHILE Language (Axiomatic Semantics)

$$\textsc{Skip}\ \{P\}\mathbf{skip}\{P\} \qquad \textsc{Abort}\ \{P\}\mathbf{abort}\{\mathbf{false}\}$$

$$\textsc{Assign}\ \{P[x/a]\}x := a\{P\}$$

$$\textsc{Seq}\ \frac{\{P\}c_1\{Q\} \qquad \{Q\}c_2\{R\}}{\{P\}c_1;\ c_2\{R\}}$$

$$\textsc{If}\ \frac{\{P \wedge b\}c_1\{Q\} \qquad \{P \wedge \neg b\}c_2\{Q\}}{\{P\}\mathbf{if}\ \ (b)\ \mathbf{then}\ c_1\mathbf{else}\ \ c_2\{Q\}}$$

$$\textsc{Loop}\ \frac{\{P \wedge b\}c\{P\}}{\{P\}\mathbf{while}\ \ (b)\ \ c\{P \wedge \neg b\}}$$

$$\textsc{Consequence}\ \frac{P' \Rightarrow P \qquad \{P\}c\{Q\} \qquad Q \Rightarrow Q'}{\{P'\}c\{Q'\}}$$

Weakest Preconditions

Definition

Given command c and postcondition Q, P is the *weakest precondition* for c and Q if:

1. $\{P\}c\{Q\}$ and
2. $\forall P'$, if $\{P'\}c\{Q\}$ then $P' \Rightarrow P$.

Weakest Preconditions

Definition

Given command c and postcondition Q, P is the *weakest precondition* for c and Q if:

1. $\{P\}c\{Q\}$ and
2. $\forall P'$, if $\{P'\}c\{Q\}$ then $P' \Rightarrow P$.

Suppose we want to prove that $\{P'\}c\{Q\}$ If we know the weakest precondition P, then by the rule of consequence, we can prove:

$$\frac{P' \Rightarrow P \qquad \{P\}c\{Q\}}{\{P'\}c\{Q\}}$$

Here is our plan:

1. We will define a function that calculates the weakest precondition: $wp(c, Q) = P$.
2. Thus, we only need to prove that $P' \Rightarrow P$.

Weakest Preconditions

Definition

Given command c and postcondition Q, P is the *weakest precondition* for c and Q if:

1. $\{P\}c\{Q\}$ and
2. $\forall P'$, if $\{P'\}c\{Q\}$ then $P' \Rightarrow P$.

Suppose we want to prove that $\{P'\}c\{Q\}$ If we know the weakest precondition P, then by the rule of consequence, we can prove:

$$\frac{P' \Rightarrow P \qquad \{P\}c\{Q\}}{\{P'\}c\{Q\}}$$

Here is our plan:

1. We will define a function that calculates the weakest precondition: $wp(c, Q) = P$.
2. Thus, we only need to prove that $P' \Rightarrow P$.

Catch: Preconditions (and postconditions) are evaluated with respect to a particular store, e.g., $\sigma \models P$ means that P is true given the initial store σ and $\sigma' \models Q$ means that Q is true given the final store σ'. We need to prove that $\forall \sigma. \sigma \models P \Rightarrow P'$.

Calculating Weakest Preconditions

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

Calculating Weakest Preconditions

$$\begin{array}{rcl} wp(\mathbf{skip}, Q) & = & Q \\ wp(x := a, Q) & = & Q[x/a] \\ wp(c_1 ; c_2, Q) & = & wp(c_1, wp(c_2, Q)) \end{array}$$

Calculating Weakest Preconditions

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

Calculate wp(x := x0, y := y0, t := x, x := y, y := t, $y = x_0 \wedge x = y_0$)

Calculating Weakest Preconditions

$$
\begin{array}{rcl}
wp(\textbf{skip}, Q) & = & Q \\
wp(x := a, Q) & = & Q[x/a] \\
wp(c_1; c_2, Q) & = & wp(c_1, wp(c_2, Q)) \\
wp(\textbf{if } (b) \textbf{ then } c_1 \textbf{ else } c_2, Q) & = & b \Rightarrow wp(c_1, Q) \wedge \neg b \Rightarrow wp(c_2, Q)
\end{array}
$$

Calculating Weakest Preconditions

$$
\begin{array}{rcl}
wp(\textbf{skip}, Q) & = & Q \\
wp(x := a, Q) & = & Q[x/a] \\
wp(c_1; c_2, Q) & = & wp(c_1, wp(c_2, Q)) \\
wp(\textbf{if } (b) \textbf{ then } c_1 \textbf{ else } c_2, Q) & = & b \Rightarrow wp(c_1, Q) \land \neg b \Rightarrow wp(c_2, Q)
\end{array}
$$

Calculate $\mathrm{wp}(\textbf{if } \mathrm{x} > 0 \textbf{ then } \mathrm{r} := \mathrm{x} \textbf{ else } \mathrm{r} := -\mathrm{x}, r = | x |)$

Calculating Weakest Preconditions

$$
\begin{array}{rcl}
wp(\textbf{skip}, Q) & = & Q \\
wp(x := a, Q) & = & Q[x/a] \\
wp(c_1; c_2, Q) & = & wp(c_1, wp(c_2, Q)) \\
wp(\textbf{if } (b) \textbf{ then } c_1 \textbf{ else } c_2, Q) & = & b \Rightarrow wp(c_1, Q) \wedge \neg b \Rightarrow wp(c_2, Q) \\
 & & \\
wp(\textbf{while } b \textbf{ invariant } \mathrm{I} \text{ do } c, Q) & = & I \textit{ notice that loop is annotated with } I
\end{array}
$$

I is a precondition

Calculating Weakest Preconditions

$$
\begin{aligned}
wp(\textbf{skip}, Q) &= Q \\
wp(x := a, Q) &= Q[x/a] \\
wp(c_1; c_2, Q) &= wp(c_1, wp(c_2, Q)) \\
wp(\textbf{if } (b) \textbf{ then } c_1 \textbf{ else } c_2, Q) &= b \Rightarrow wp(c_1, Q) \wedge \neg b \Rightarrow wp(c_2, Q)
\end{aligned}
$$

$$wp(\textbf{while } b \textbf{ invariant } \mathrm{I} \text{ do } c, Q) = I \textit{ notice that loop is annotated with } I$$

I is a precondition

$$wp(\textbf{while } b \textbf{ invariant } \mathrm{I} \text{ do } c, Q) = I \wedge (\forall x \cdots . \neg b \wedge I \Rightarrow Q)$$

... and I is a postcondition

Calculating Weakest Preconditions

$$
\begin{array}{rcl}
wp(\textbf{skip}, Q) & = & Q \\
wp(x := a, Q) & = & Q[x/a] \\
wp(c_1; c_2, Q) & = & wp(c_1, wp(c_2, Q)) \\
wp(\textbf{if}\ (b)\ \textbf{then}\ c_1\ \textbf{else}\ c_2, Q) & = & b \Rightarrow wp(c_1, Q) \wedge \neg b \Rightarrow wp(c_2, Q)
\end{array}
$$

$$wp(\textbf{while}\ b\ \textbf{invariant}\ \mathrm{I}\ \text{do}\ c, Q) \quad = \quad I \textit{ notice that loop is annotated with } I$$

I is a precondition

$$wp(\textbf{while}\ b\ \textbf{invariant}\ \mathrm{I}\ \text{do}\ c, Q) \quad = \quad I \wedge (\forall x \cdots .\neg b \wedge I \Rightarrow Q)$$

... and I is a postcondition

$$wp(\textbf{while}\ b\ \textbf{invariant}\ \mathrm{I}\ \text{do}\ c, Q) \quad = \quad I \wedge (\forall x \cdots .\neg b \wedge I \Rightarrow Q) \wedge (\forall x \cdots .b \wedge I \Rightarrow wp(c, I))$$

... and I holds before and after c

Calculate wp(n := n0; r := 0; **while** (n > 0) **invariant** I do (r := r + m; n := n − 1), $r = m \cdot n_0$)

Calculating Weakest Preconditions

$$
\begin{array}{rcl}
wp(\textbf{skip}, Q) & = & Q \\
wp(x{:=}\ a, Q) & = & Q[x/a] \\
wp(c_1; c_2, Q) & = & wp(c_1, wp(c_2, Q)) \\
wp(\textbf{if}\ \ (b)\ \textbf{then}\ c_1 \textbf{else}\ \ c_2, Q) & = & b \Rightarrow wp(c_1, Q) \wedge \neg b \Rightarrow wp(c_2, Q) \\
wp(\textbf{while}\ \ b\ \textbf{invariant}\ \ \mathrm{I}\ \ \mathrm{do}\ c, Q) & = & I \wedge (\forall x \cdots . \neg b \wedge I \Rightarrow Q) \wedge (\forall x \cdots . b \wedge I \Rightarrow wp(c, I))
\end{array}
$$

Calculate wp(n := n0; r := 0; **while** (n > 0) **invariant** I do (r := r + m; n := n − 1),$r = m \cdot n_0$)
What is the loop invariant I?

Calculating Weakest Preconditions

$$\begin{array}{rcl} wp(\textbf{skip}, Q) & = & Q \\ wp(x := a, Q) & = & Q[x/a] \\ wp(c_1; c_2, Q) & = & wp(c_1, wp(c_2, Q)) \\ wp(\textbf{if } (b) \textbf{ then } c_1 \textbf{ else } c_2, Q) & = & b \Rightarrow wp(c_1, Q) \wedge \neg b \Rightarrow wp(c_2, Q) \\ wp(\textbf{while } b \textbf{ invariant } I \textbf{ do } c, Q) & = & I \wedge (\forall x \cdots . \neg b \wedge I \Rightarrow Q) \wedge (\forall x \cdots . b \wedge I \Rightarrow wp(c, I)) \end{array}$$

Calculate wp(n := n0; r := 0; **while** (n > 0) **invariant** I do (r := r + m; n := n − 1), $r = m \cdot n_0$)
What is the loop invariant I?

Let I be $m \cdot n_0 = r + n \cdot m$:

1. Before the loop, $r = 0$ and $n_0 = n$.
2. If $m \cdot n_0 = r + n \cdot m$ before an iteration, then the loop body first updates r := r + m and then n := n − 1, thus we have $(r + m) + (n - 1) \cdot m$. **Notice that the invariant is broken between the two updates within the loop.**
3. After the loop, $r = m \cdot n_0$ and $n = 0$.

Verification using Weakest Preconditions

Is $\{y > 15\}$x := y + 10$\{x > 20\}$ verifiable?

1. Calculate wp(x := y + 10, x > 20)$) = y > 10$
2. Verify that $y > 15 \Rightarrow y > 10$

Satisfiability vs. Validity

The formula $\phi(x)$ is *valid* if it is true for all values of x. We write $\forall x.\phi(x)$ to mean "Is $\phi(x)$ valid?".
The formula $\phi(x)$ is *satisfiable* if it is true for some value of x. We write $\exists x.\phi(x)$ to mean "Is $\phi(x)$ satisfiable?"

Satisfiability vs. Validity

The formula $\phi(x)$ is *valid* if it is true for all values of x. We write $\forall x.\phi(x)$ to mean "Is $\phi(x)$ valid?".
The formula $\phi(x)$ is *satisfiable* if it is true for some value of x. We write $\exists x.\phi(x)$ to mean "Is $\phi(x)$ satisfiable?"
Are the following formulas valid (or not)? Are they satisfiable (or not)?

1. $x > 0 \Rightarrow x + y > 0$

Satisfiability vs. Validity

The formula $\phi(x)$ is *valid* if it is true for all values of x. We write $\forall x.\phi(x)$ to mean "Is $\phi(x)$ valid?".
The formula $\phi(x)$ is *satisfiable* if it is true for some value of x. We write $\exists x.\phi(x)$ to mean "Is $\phi(x)$ satisfiable?"
Are the following formulas valid (or not)? Are they satisfiable (or not)?

1. $x > 0 \Rightarrow x + y > 0$
2. $x > 0 \wedge y > 0 \Rightarrow x + y > 0$

Satisfiability vs. Validity

The formula $\phi(x)$ is *valid* if it is true for all values of x. We write $\forall x.\phi(x)$ to mean "Is $\phi(x)$ valid?".
The formula $\phi(x)$ is *satisfiable* if it is true for some value of x. We write $\exists x.\phi(x)$ to mean "Is $\phi(x)$ satisfiable?"
Are the following formulas valid (or not)? Are they satisfiable (or not)?

1. $x > 0 \Rightarrow x + y > 0$
2. $x > 0 \wedge y > 0 \Rightarrow x + y > 0$
3. $x < 0 \Rightarrow x \cdot x < 0$

Satisfiability vs. Validity

The formula $\phi(x)$ is *valid* if it is true for all values of x. We write $\forall x.\phi(x)$ to mean "Is $\phi(x)$ valid?".
The formula $\phi(x)$ is *satisfiable* if it is true for some value of x. We write $\exists x.\phi(x)$ to mean "Is $\phi(x)$ satisfiable?"
Are the following formulas valid (or not)? Are they satisfiable (or not)?

1. $x > 0 \Rightarrow x + y > 0$
2. $x > 0 \wedge y > 0 \Rightarrow x + y > 0$
3. $x < 0 \Rightarrow x \cdot x < 0$

When we say "verify that $y > 15 \Rightarrow y > 10$" we mean "is $y > 15 \Rightarrow y > 10$ valid?".

Satisfiability vs. Validity

The formula $\phi(x)$ is *valid* if it is true for all values of x. We write $\forall x.\phi(x)$ to mean "Is $\phi(x)$ valid?".
The formula $\phi(x)$ is *satisfiable* if it is true for some value of x. We write $\exists x.\phi(x)$ to mean "Is $\phi(x)$ satisfiable?"
Are the following formulas valid (or not)? Are they satisfiable (or not)?

1. $x > 0 \Rightarrow x + y > 0$
2. $x > 0 \wedge y > 0 \Rightarrow x + y > 0$
3. $x < 0 \Rightarrow x \cdot x < 0$

When we say "verify that $y > 15 \Rightarrow y > 10$" we mean "is $y > 15 \Rightarrow y > 10$ valid?".

Observation: $\forall x.\phi(x)$ holds if and only if $\exists x.\neg\phi(x)$ does not hold.
Alternatively, if $\exists x.\neg\phi(x)$ holds, then the value of x that makes $\neg\phi(x)$ true is a *counterexample* that contradicts a claim that $\forall x.\phi(x)$ holds.