Lecturecise 6
More on Postconditions and Preconditions. Loops and Recursion

Viktor Kuncak

Review of Key Definitions

Hoare triple:

$$\{P\}\ r\ \{Q\} \iff \forall s, s' \in S. \big((s \in P \land (s, s') \in r) \to s' \in Q\big)$$

$\{P\}$ does not denote a singleton set containing P but is just a notation for an "assertion" around a command. Likewise for $\{Q\}$.

Strongest postcondition:

$$sp(P, r) = \{s' \mid \exists s.\, s \in P \land (s, s') \in r\}$$

Weakest precondition:

$$wp(r, Q) = \{s \mid \forall s'.(s, s') \in r \to s' \in Q\}$$

More Laws on Preconditions and Postconditions

Disjunctivity of sp

$$sp(P_1 \cup P_2, r) = sp(P_1, r) \cup sp(P_2, r)$$

$$sp(P, r_1 \cup r_2) = sp(P, r_1) \cup sp(P, r_2)$$

Conjunctivity of wp

$$wp(r, Q_1 \cap Q_2) = wp(r, Q_1) \cap wp(r, Q_2)$$

$$wp(r_1 \cup r_2, Q) = wp(r_1, Q) \cap wp(r_2, Q)$$

Pointwise wp

$$wp(r, Q) = \{s \mid s \in S \wedge sp(\{s\}, r) \subseteq Q\}$$

Pointwise sp

$$sp(P, r) = \bigcup_{s \in P} sp(\{s\}, r)$$

Hoare Logic for Loop-free Code

Expanding Paths

The condition

$$\{P\}\ \Big(\bigcup_{i \in J} r_i\Big)\ \{Q\}$$

is equivalent to

$$\forall i. i \in J \rightarrow \{P\} r_i \{Q\}$$

Proof: By definition, or use that the first condition is equivalent to $sp(P, \bigcup_{i \in J} r_i) \subseteq Q$ and $\{P\} r_i \{Q\}$ to $sp(P, r_i) \subseteq Q$

Transitivity

If $\{P\} s_1 \{Q\}$ and $\{Q\} s_2 \{R\}$ then also $\{P\} s_1 \circ s_2 \{R\}$.
We write this as the following inference rule:

$$\frac{\{P\} s_1 \{Q\}, \quad \{Q\} s_2 \{R\}}{\{P\} s_1 \circ s_2 \{R\}}$$

Exercise

We call a relation $r \subseteq S \times S$ functional if $\forall x, y, z \in S.(x, y) \in r \wedge (x, z) \in r \rightarrow y = z$. For each of the following statements either give a counterexample or prove it. In the following, $Q \subseteq S$.

(i) for any r, $wp(r, S \setminus Q) = S \setminus wp(r, Q)$

(ii) if r is functional, $wp(r, S \setminus Q) = S \setminus wp(r, Q)$

(iii) for any r, $wp(r, Q) = sp(Q, r^{-1})$

(iv) if r is functional, $wp(r, Q) = sp(Q, r^{-1})$

(v) for any r, $wp(r, Q_1 \cup Q_2) = wp(r, Q_1) \cup wp(r, Q_2)$

(vi) if r is functional, $wp(r, Q_1 \cup Q_2) = wp(r, Q_1) \cup wp(r, Q_2)$

(vii) for any r, $wp(r_1 \cup r_2, Q) = wp(r_1, Q) \cup wp(r_2, Q)$

(viii) Alice has a conjecture: For all sets S and relations $r \subseteq S \times S$ it holds:

$$\Big(S \neq \emptyset \wedge dom(r) = S \wedge \triangle_S \cap r = \emptyset\Big) \rightarrow \Big(r \circ r \cap ((S \times S) \setminus r) \neq \emptyset\Big)$$

where $\Delta_S = \{(x, x) \mid x \in S\}$, $dom(r) = \{x \mid \exists y.(x, y) \in r\}$. She tried many sets and relations and did not find any counterexample. Is her conjecture true? If so, prove it; if false, provide a counterexample for which S is as small as possible.

Formulas for Strongest Postconditions

Forward Verification Condition Generation

Computing Formulas for Strongest Postcondition

Let $\bar{x}, \bar{x}'$ range over the sets of states S
We gave definition of strongest postcondition (sp) on sets and relations $P_1 \subseteq S$ and $r \subseteq S \times S$:

$$sp(P_1, r) = \{\bar{x}' \mid \exists \bar{x}.\ \bar{x} \in P_1 \land (\bar{x}, \bar{x}') \in r\}$$

We now consider how to compute with *representations* of those sets and relations in terms of formulas. Let

- $P_1 = \{\bar{x} \mid P\}$ for some formula P with $FV(P)$ among $\bar{x}$
- $r = \rho(c) = \{(\bar{x}, \bar{x}') \mid F\}$ for some formula F with $FV(F)$ among $\bar{x}$, $\bar{x}'$

We can then conclude $sp(P_1, r) = \{\bar{x}' \mid \exists \bar{x}.\ P \land F\}$
Denote a formula equivalent to $(\exists \bar{x}.\ P \land F)[\bar{x}' := \bar{x}]$ by $\mathbf{sp_F(P, c)}$

- we renamed variables so that the result is in terms of $\bar{x}$, not $\bar{x}'$
- multiple syntactic choices for $sp(P_1, r)$; all logically equivalent

Strongest Postcondition Formula

If P is a formula on state and c a command, we define $sp_F(P, c)$ as the formula version of the strongest postcondition operator. $sp_F(P, c)$ is therefore the formula Q that describes the set of states that can result from executing c in a state satisfying P. Thus, we have that

$$sp_F(P, c) = Q$$

implies

$$sp(\{\bar{x}|P\}, \rho(c)) = \{\bar{x}|Q\}$$

We will denote the set of states satisfying a predicate by underscore s, i.e. for a predicate P, let $\tilde{P}$ be the set of states that satisfies it:

$$\tilde{P} = \{\bar{x}|P\}$$

Forward VCG: Using Strongest Postcondition

Remember: $\{\tilde{P}\}\ \rho(c)\ \{\tilde{Q}\}$ is equivalent to

$$sp(\tilde{P}, \rho(c)) \subseteq \tilde{Q}$$

A syntactic form of Hoare triple is $\{P\}c\{Q\}$

That syntactic form is therefore equivalent to proving

$$\forall \bar{x}.\ (sp_F(P, c) \rightarrow Q)$$

We can use the sp_F operator to compute verification conditions

Forward VCG: Using Strongest Postcondition

Remember: $\{\tilde{P}\}\ \rho(c)\ \{\tilde{Q}\}$ is equivalent to

$$sp(\tilde{P}, \rho(c)) \subseteq \tilde{Q}$$

A syntactic form of Hoare triple is $\{P\}c\{Q\}$

That syntactic form is therefore equivalent to proving

$$\forall \bar{x}.\ (sp_F(P, c) \rightarrow Q)$$

We can use the sp_F operator to compute verification conditions

We give rules to compute $sp_F(P, c)$ for our commands such that

$$(sp_F(P, c) = Q)\ \text{ implies }\ (sp(\tilde{P}, \rho(c)) = \tilde{Q})$$

Finding Formula for sp_F

Given the goal of the formula

$$(sp_F(P, c) = Q) \text{ implies } (sp(\tilde{P}, \rho(c)) = \tilde{Q})$$

All Q with $FV(Q) \subseteq \bar{x}$ satisfying $sp(\tilde{P}, \rho(c)) = \tilde{Q}$ are equivalent to formula

$$(\exists \bar{x}.\ P \wedge F)[\bar{x}' := \bar{x}] \qquad (*)$$

where $\rho(c) = \{(\bar{x}, \bar{x}') \mid F\}$

- we are looking for some syntactic simplification of $(*)$

Assume Statement

Consider

- a precondition P, with $FV(P)$ among $\bar{x}$ and
- a property E, also with $FV(E)$ among $\bar{x}$

Note that $\rho(assume(E)) = \Delta_{\tilde{E}}$. Therefore

$$\begin{aligned}
& sp(\tilde{P}, \rho(assume(E))) \\
& = sp(\tilde{P}, \Delta_{\tilde{E}}) \\
& = \{\bar{x}' \mid \exists \bar{x} \in \tilde{P}.\ (\bar{x}, \bar{x}') \in \Delta_{\tilde{E}}\} \\
& = \{\bar{x}' \mid \exists \bar{x} \in \tilde{P}.\ (\bar{x} = \bar{x}' \wedge \bar{x} \in \tilde{E})\} \\
& = \{\bar{x}' \mid \bar{x}' \in \tilde{P} \wedge \bar{x}' \in \tilde{E}\} = \{\bar{x} \mid \bar{x} \in \tilde{P} \wedge \bar{x} \in \tilde{E}\} \\
& = \{\bar{x} \mid P \wedge E\}
\end{aligned}$$

So, we define:

$$sp_F(P, \text{assume(E)}) = P \wedge E$$

Strongest Postcondition of Havoc

Formula for havoc. Let $\bar{x} = x_1, \ldots, x_i, \ldots, x_n$

$$R(havoc(x_i)) = \bigwedge_{v \neq x} v = v' \qquad\qquad = \; F$$

General formula for postcondition is:

$$(\exists \bar{x}.\ P \land F)[\bar{x}' := \bar{x}] \qquad (*)$$

It becomes here

$$(\exists \bar{x}.\ P \land \bigwedge_{j \neq i} x_j = x_j')[\bar{x}' := \bar{x}]$$

Equalities over all variables except x_i are eliminated, so we obtain

$$(\exists x_i . P)[\bar{x}' := \bar{x}]$$

No primed variables left, renaming does nothing. Result: $\mathbf{(\exists x_i . P)}$.

Strongest Postcondition of Havoc

To avoid many nested quantifiers and name clashes, we rename first:

$$sp_F(P, \text{havoc}(x)) = \exists x_0 . P[x := x_0] \;\; \text{which is same as } \exists x . P$$

Strongest Postcondition of Havoc

To avoid many nested quantifiers and name clashes, we rename first:

$$sp_F(P, \text{havoc}(x)) = \exists x_0 . P[x := x_0] \;\; \text{which is same as } \exists x . P$$

Exercise:
Precondition: $\{x \geq 2 \land y \leq 5 \land x \leq y\}$.
Code: havoc(x)

Strongest Postcondition of Havoc

To avoid many nested quantifiers and name clashes, we rename first:

$$sp_F(P, \text{havoc}(x)) = \exists x_0 . P[x := x_0] \;\; \text{which is same as } \exists x . P$$

Exercise:
Precondition: $\{x \geq 2 \wedge y \leq 5 \wedge x \leq y\}$.
Code: havoc(x)

$$\exists x_0 .\ x_0 \geq 2 \wedge y \leq 5 \wedge x_0 \leq y$$

i.e.

$$\exists x_0 .\ 2 \leq x_0 \leq y \wedge y \leq 5$$

i.e.

$$2 \leq y \wedge y \leq 5$$

Note: If we simply removed conjuncts containing x, we would get just $y \leq 5$.

Rules for Computing Strongest Postcondition

Assignment Statement

Define:

$$sp_F(P, x = e) = \exists x_0.(P[x := x_0] \wedge x = e[x := x_0])$$

Indeed:

$$\begin{aligned}
&sp(\tilde{P}, \rho(x = e)) \\
&= \{\bar{x}' \mid \exists \bar{x}.\ (\bar{x} \in \tilde{P} \wedge (\bar{x}, \bar{x}') \in \rho(x = e))\} \\
&= \{\bar{x}' \mid \exists \bar{x}.\ (\bar{x} \in \tilde{P} \wedge \bar{x}' = \bar{x}[x := e(\bar{x})])\}
\end{aligned}$$

Exercise

Precondition: $\{x \geq 5 \wedge y \geq 3\}$.
Code: x = x + y + 10

$$sp(x \geq 5 \wedge y \geq 3, x = x + y + 10) =$$

Exercise

Precondition: $\{x \geq 5 \wedge y \geq 3\}$.
Code: x = x + y + 10

$$sp(x \geq 5 \wedge y \geq 3, x = x + y + 10) =$$

$$\exists x_0.\ x_0 \geq 5 \wedge y \geq 3 \ \wedge \ x = x_0 + y + 10$$

$$\leftrightarrow \ \ y \geq 3 \wedge x \geq y + 15$$

Rules for Computing Strongest Postcondition

Sequential Composition

For relations we proved

$$sp(\tilde{P}, r_1 \circ r_2) = sp(sp(\tilde{P}, r_1), r_2)$$

Therefore, define

$$sp_F(P, c_1; c_2) = sp_F(sp_F(P, c_1), c_2)$$

Nondeterministic Choice (Branches)

We had $sp(\tilde{P}, r_1 \cup r_2) = sp(\tilde{P}, r_1) \cup sp(\tilde{P}, r_2)$. Therefore define:

$$sp_F(P, c_1 \,[]\, c_2) = sp_F(P, c_1) \vee sp_F(P, c_2)$$

Correctness

We can show by easy induction on c_1 that for all P:

$$sp(\tilde{P}, \rho(c_1)) = \{\bar{x} \mid sp_F(P, c_1)\}$$

Size of Generated Formulas

The size of the formula can be exponential because each time we have a nondeterministic choice, we double formula size:

$$
\begin{aligned}
& sp_F(P, (c_1 \,\square\, c_2); (c_3 \,\square\, c_4)) = \\
& sp_F(sp_F(P, c_1 \,\square\, c_2), c_3 \,\square\, c_4) = \\
& sp_F(sp_F(P, c_1) \vee sp_F(P, c_2), c_3 \,\square\, c_4) = \\
& sp_F(sp_F(P, c_1) \vee sp_F(P, c_2), c_3) \vee sp_F(sp_F(P, c_1) \vee sp_F(P, c_2), c_4)
\end{aligned}
$$

Another Useful Characterization of sp

For any relation $\sigma \subseteq S \times S$ we define its range by

$$ran(\sigma) = \{s' \mid \exists s \in S.(s, s') \in \sigma\}$$

Lemma: suppose that

- $A \subseteq S$ and $r \subseteq S \times S$
- $\Delta = \{(s, s) \mid s \in S\}$

Then

$$sp(A, r) = ran(\Delta_A \circ r)$$

Proof of the previous fact

$$
\begin{aligned}
ran(\Delta_A \circ r) &= ran(\{(x,z) \mid \exists y.\ (x,y) \in \Delta_A \wedge (y,z) \in r\}) \\
&= ran(\{(x,z) \mid \exists y.\ x = y \wedge x \in A \wedge (y,z) \in r\}) \\
&= ran(\{(x,z) \mid x \in A \wedge (x,z) \in r\}) \\
&= \{z \mid \exists x.\ x \in A \wedge (x,z) \in r\} \\
&= sp(A, r)
\end{aligned}
$$

Reducing sp to Relation Composition

The following identity holds for relations:

$$sp(\tilde{P}, r) = ran(\Delta_P \circ r)$$

Based on this, we can compute $sp(\tilde{P}, \rho(c_1))$ in two steps:

- compute formula $R(assume(P); c_1)$
- existentially quantify over initial (non-primed) variables

Indeed, if F_1 is a formula denoting relation r_1, that is,

$$r_1 = \{(\bar{x}, \bar{x}\,') \mid F_1(\bar{x}, \bar{x}\,')\}$$

then $\exists \bar{x}. F_1(\bar{x}, \bar{x}\,')$ is formula denoting the range of r_1:

$$ran(r_1) = \{\bar{x}' \mid \exists \bar{x}. F_1(\bar{x}, \bar{x}\,')\}$$

Moreover, the resulting approach does not have exponentially large formulas.

Computing Weakest Precondition Formulas

Rules for Computing Weakest Preconditions

We derive the rules below from the definition of weakest precondition on sets and relations

$$wp(r, \tilde{Q}) = \{s \mid \forall s'.\ (s, s') \in r \rightarrow s' \in \tilde{Q}\}$$

Let now $r = \rho(c) = \{(\bar{x}, \bar{x}') \mid F\}$ and $\tilde{Q} = \{\bar{x} \mid Q\}$. Then

$$wp(r, \tilde{Q}) = \{\bar{x} \mid \forall \bar{x}'.(F \rightarrow Q[\bar{x} := \bar{x}'])\}$$

Thus, we will be defining wp_F as equivalent to

$$\forall \bar{x}'.\ (F \wedge Q[\bar{x} := \bar{x}'])$$

Assume Statement

Suppose we have one variable x, and identify the state with that variable. Note that $\rho(assume(F)) = \Delta_{\tilde{F}}$. By definition

$$
\begin{aligned}
wp(\Delta_{\tilde{F}}, \tilde{Q}) &= \{x \mid \forall x'.(x, x') \in \Delta_{\tilde{F}} \rightarrow x' \in \tilde{Q}\} \\
&= \{x \mid \forall x'.(x \in \tilde{F} \wedge x = x') \rightarrow x' \in \tilde{Q}\} \\
&= \{x \mid x \in \tilde{F} \rightarrow x \in \tilde{Q}\} = \{x \mid F \rightarrow Q\}
\end{aligned}
$$

Changing from sets to formulas, we obtain the rule for *wp* on formulas:

$$wp_F(\text{assume(F)}, Q) = (F \rightarrow Q)$$

Rules for Computing Weakest Preconditions

Assignment Statement

Consider the case of two variables. Recall that the relation associated with the assignment $x = e$ is

$$x' = e \wedge y' = y$$

Then we have, for formula Q containing x and y:

$$\begin{aligned} wp(\rho(x = e), \{(x, y) \mid Q\}) &= \{(x, y) \mid \quad \forall x'.\forall y'.\ x' = e \wedge y' = y \rightarrow \\ &\qquad\qquad\qquad\quad Q[x := x', y := y']\} \\ &= \{(x, y) \mid \quad Q[x := e]\} \end{aligned}$$

From here we obtain a justification to define:

$$wp_F(x = e, Q) = Q[x := e]$$

Rules for Computing Weakest Preconditions

Havoc Statement

$$wp_F(\mathrm{havoc}(\mathrm{x}), Q) = \forall x. Q$$

Sequential Composition

$$wp(r_1 \circ r_2, \tilde{Q}) = wp(r_1, wp(r_2, \tilde{Q}))$$

Same for formulas:

$$wp_F(c_1 \ ; \ c_2, Q) = wp_F(c_1, wp_F(c_2, Q))$$

Nondeterministic Choice (Branches)

In terms of sets and relations

$$wp(r_1 \cup r_2, \tilde{Q}) = wp(r_1, \tilde{Q}) \cap wp(r_2, \tilde{Q})$$

In terms of formulas

$$wp_F(c_1 \square c_2, Q) = wp_F(c_1, Q) \wedge wp_F(c_2, Q)$$

Summary of Weakest Precondition Rules

c	$wp(c, Q)$
$x = e$	$Q[x := e]$
$havoc(x)$	$\forall x. Q$
$assume(F)$	$F \rightarrow Q$
$c_1 \square c_2$	$wp(c_1, Q) \wedge wp(c_2, Q)$
$c_1 ; c_2$	$wp(c_1, wp(c_2, Q))$

Size of Generated Verification Conditions

Because of the rule

$$wp_F(c_1 \square c_2, Q) = wp_F(c_1, Q) \wedge wp_F(c_2, Q)$$

which duplicates Q, the size can be exponential.

$wp_F((c_1 \square c_2); (c_3 \square c_4), Q) =$

Avoiding Exponential Blowup

Propose an algorithm that, given an arbitrary program c and a formula Q, computes in polynomial time formula equivalent to $wp_F(c, Q)$