Lecture 5
Computing Postconditions and Preconditions
Loops and Recursion

Viktor Kuncak

Review of Key Definitions

Hoare triple:

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

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

Strongest postcondition:

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

Weakest precondition:

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

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:

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

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.

Helping Alice: Properties of the Relation

We believe Alice is wrong and that there exists r such that the property $(viii)$ from the previous slide is false. In other words, that there is relation r such that

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

We are thus looking for relation that is:

- on a non-empty set S
- **total**, because $dom(r) = S$ means that for every element $x \in S$ there exists $y \in S$ such that $(x, y) \in r$.
- **irreflexive**: there is no element $x \in S$ such that $(x, x) \in r$, otherwise we would have $\Delta \cap r = \emptyset$
- **transitive**: indeed, if B^c denotes complement of a set B, then $A \cap B^c = \emptyset$ is equivalent to $A \subseteq B$. Thus, the last conjunct above just says that $r \circ r \subseteq r$, which is stating transitivity of r.

Find a total irreflexive transitive relation on a non-empty set.

Counter-Example for Alice

Let $S = \{0, 1, 2, \ldots\}$ (non-negative integers)
Define $r = \{(x, y) \mid x < y\}$
S is non-empty, for every element there exists a larger, no element is strictly larger than itself, and the relation is transitive.

- r satisfies properties that make Alice's conjecture false

Counter-Example for Alice

Let $S = \{0, 1, 2, \ldots\}$ (non-negative integers)
Define $r = \{(x, y) \mid x < y\}$
S is non-empty, for every element there exists a larger, no element is strictly larger than itself, and the relation is transitive.

- r satisfies properties that make Alice's conjecture false

Is there a relation on a finite set as a counter-example? Perhaps Alice was trying finite counter-examples by hand, but if she tried to enumerate it fast with a computer program, she would find a different, finite, counter-example?

Counter-Example for Alice

Let $S = \{0, 1, 2, \ldots\}$ (non-negative integers)
Define $r = \{(x, y) \mid x < y\}$
S is non-empty, for every element there exists a larger, no element is strictly larger than itself, and the relation is transitive.

- r satisfies properties that make Alice's conjecture false

Is there a relation on a finite set as a counter-example? Perhaps Alice was trying finite counter-examples by hand, but if she tried to enumerate it fast with a computer program, she would find a different, finite, counter-example?

- No! All relations with these properties are infinite!

Total Irreflexive Transitive Relations are Infinite

It may be helpful to keep $<$ as an example in mind, but now r is arbitrary with the given properties.
We show by induction that for every non-negative integer k there exists a sequence $x_0, x_1, \ldots, x_k$ of elements inside S such that $(x_i, x_{i+1}) \in r$ for every $0 \leq i < k$.

- Let $x_0 \in S$ be an arbitrary element of our non-empty set S.
- Consider by inductive hypothesis elements $x_0, \ldots, x_k$ such that $(x_i, x_{i+1}) \in r$ for all $1 \leq i < k$. By totality of r, there exists element y such that $(x_i, y) \in r$; define x_{i+1} to be one such y. We obtain a longer sequence, which completes proof by induction.

In a sequence of elements related by r, all elements are distinct. Indeed, for $i < j$, by transitivity, $(x_i, x_j) \in r$, and r is irreflexive.
Now, if S were finite it would have some size given by natural number n. By our property there exists a sequence of $n + 1$ distinct elements inside S, which is a contradiction.

Formulas for Strongest Postconditions

Forward Verification Condition Generation

Computing Formulas for Strongest Postcondition

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

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

Denote the set of states satisfying a predicate by underscore s: let $\tilde{P}$ be the set of states that satisfies it: $\tilde{P} = \{\bar{x} | P\}$
We consider how to compute with *representations* of those sets and relations

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

We introduce sp_F on formulas. We look how to compute Q such that $sp_F(P, c) = Q$ implies $sp(\tilde{P}, \rho(c)) = \tilde{Q}$

Deriving sp_F

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

for $P_1 = \tilde{P}$, $r = \rho(c)$, this becomes

$$sp(\tilde{P}, \rho(c)) = \{\exists \bar{x}.\ P \land F_c\}$$

If we use convention that formulas range over $\bar{x}$ and not $\bar{x}'$, then $\boldsymbol{sp_F(P, c)}$ will be a formula logically equivalent to

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

$sp_F(P, c)$ is therefore the formula Q that describes the set of states that can result from executing c in a state satisfying P.

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 such as the one above

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 such as the one above
We next give rules to compute $sp_F(P, c)$ for our commands such that

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

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:

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

Strongest Postcondition of Havoc

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

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

General formula for postcondition is:

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

It becomes here

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

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: $\boldsymbol{(\exists x_i.P)}$.

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

$$\exists x_0.\; x_0 \geq 2 \land y \leq 5 \land x_0 \leq y$$

i.e.

$$\exists x_0.\; 2 \leq x_0 \leq y \land y \leq 5$$

i.e.

$$2 \leq y \land y \leq 5$$

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

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

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

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

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

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

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 $\rho(\mathit{assume}(F)) = \Delta_{\tilde{F}}$. 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:

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

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:

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

From here we obtain a justification to define:

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

Rules for Computing Weakest Preconditions

Havoc Statement

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

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 $wp_F(c, Q)$