Lecturecise 9
Hoare Logic

2013

Introduction

We have seen how to translate programs into relations. We will use these relations in a proof system called Hoare logic. Hoare logic is a way of inserting annotations into code to make proofs about (imperative) program behavior simpler.

Example proof:

```
//{0 <= y}
i = y;
//{0 <= y & i = y}
r = 0;
//{0 <= y & i = y & r = 0}
while //{r = (y−i)*x & 0 <= i}
 (i > 0) (
  //{r = (y−i)*x & 0 < i}
  r = r + x;
  //{r = (y−i+1)*x & 0 < i}
  i = i − 1
  //{r = (y−i)*x & 0 <= i}
)
//{r = x * y}
```

Hoare triples for Sets and Relations

When $P, Q \subseteq S$ (sets of states) and $r \subseteq S \times S$ (relation on states, command semantics) then the Hoare triple

$$\{P\}\ r\ \{Q\}$$

means

$$\forall s, s' \in S. \left(s \in P \land (s, s') \in r \to s' \in Q\right)$$

We call P precondition and Q postcondition.

The Hoare triple provides only a *partial correctness* guarantee, i.e. if P holds initially, and r executes and terminates, then Q must hold. If r does not terminate, then no guarantees on Q are provided.

Exercise: Which Hoare triples are valid?

Assume all variables to be over integers.

1. $\{j = a\}$ j :=j+1 $\{a = j + 1\}$
2. $\{i = j\}$ i:=j+i $\{i > j\}$
3. $\{j = a + b\}$ i:=b; j:=a $\{j = 2 * a\}$
4. $\{i > j\}$ j:=i+1; i:=j+1 $\{i > j\}$
5. $\{i \mathrel{!}= j\}$ **if** i>j then m:=i−j **else** m:=j−i $\{m > 0\}$
6. $\{i = 3*j\}$ **if** i>j then m:=i−j **else** m:=j−i $\{m-2*j=0\}$
7. $\{x = b\}$ **while** x>a **do** x:=x−1 $\{b = a\}$

More postconditions

What is the relationship between these postconditions?

$$\{x = 5\} \quad x := x + 2 \quad \{\mathbf{x > 0}\}$$

$$\{x = 5\} \quad x := x + 2 \quad \{\mathbf{x = 7}\}$$

- weakest conditions (predicates) correspond to largest sets
- strongest conditions (predicates) correspond to smallest sets

that satisfy a given property.

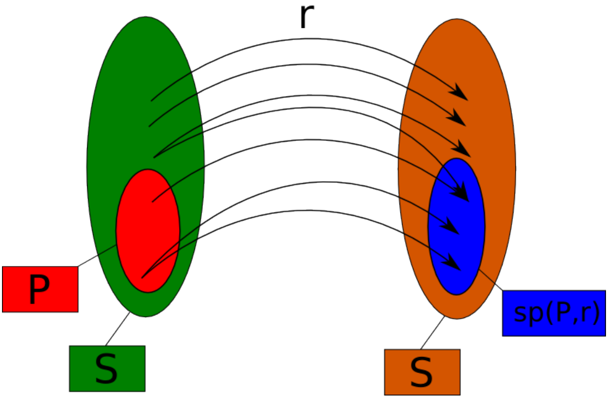
(Graphically, a stronger condition $x > 0 \wedge y > 0$ denotes one quadrant in plane, whereas a weaker condition $x > 0$ denotes the entire half-plane.)

Strongest postcondition

Definition: For $P \subseteq S$, $r \subseteq S \times S$,

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

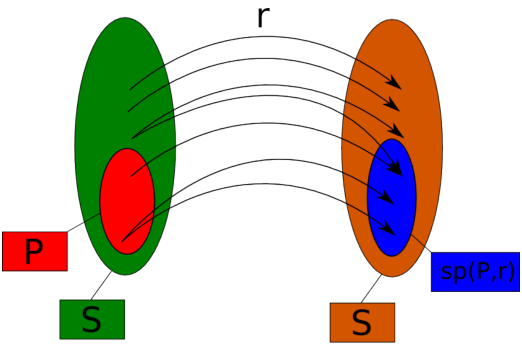
This is simply the relation image of a set.



Lemma: Characterization of sp

$sp(P, r)$ is the the smallest set Q such that $\{P\}r\{Q\}$, that is:

- $\{P\}r\{sp(P, r)\}$
- $\forall Q \subseteq S.\ \{P\}r\{Q\} \rightarrow sp(P, r) \subseteq Q$



$$\{P\}\ r\ \{Q\} \Leftrightarrow \forall s, s' \in S.\ (s \in P \wedge (s, s') \in r \rightarrow s' \in Q)$$
$$sp(P, r) = \{s' \mid \exists s. s \in P \wedge (s, s') \in r\}$$

Backward Propagation of Errors

If we have a relation r and a set of errors E, we can check if a program meets its specification by checking:

$$sp(P,r) \cap E = \emptyset$$

$$\forall y. \neg(y \in sp(P,r) \land y \in E)$$

$$\forall y.\ \neg((\exists x. P(x) \land (x,y) \in r) \land y \in E)$$

$$\forall y.\ \neg\exists x. (P(x) \land (x,y) \in r \land y \in E)$$

$$\forall x, y.\ \neg(x \in P \land (x,y) \in r \land y \in E)$$

$$\forall x, y.\ \neg(x \in P \land (y,x) \in r^{-1} \land y \in E)$$

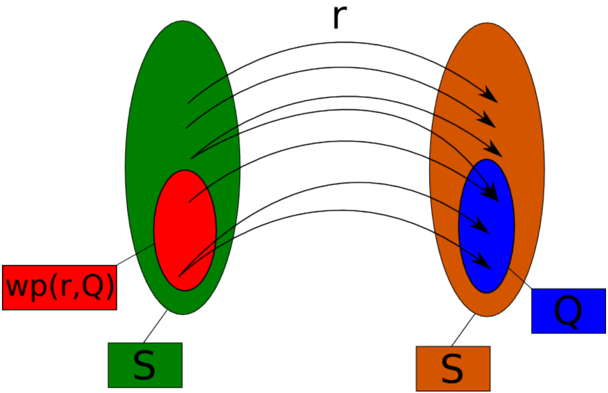
$$\forall x, y.\ \neg(y \in E \land (y,x) \in r^{-1} \land x \in P)$$

$$sp(E, r^{-1}) \cap P = \emptyset$$

$$P \subseteq sp(E, r^{-1})^c$$

In other words, we obtain an upper bound on the set of states P from which we do not reach error. We next introduce the notion of weakest precondition, which allows us to express $sp(E, r^{-1})$ from Q given as complement of error states E.

Weakest precondition

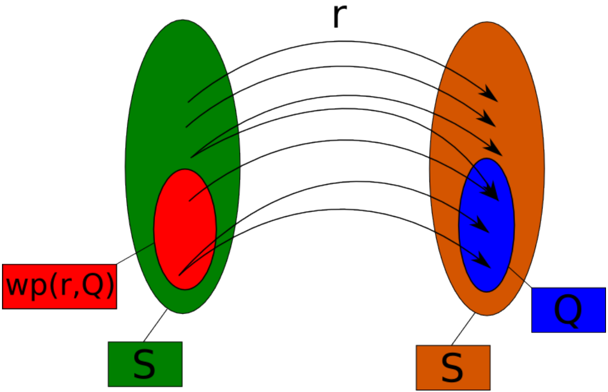


Weakest precondition

Definition: for $Q \subseteq S$, $r \subseteq S \times S$,

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

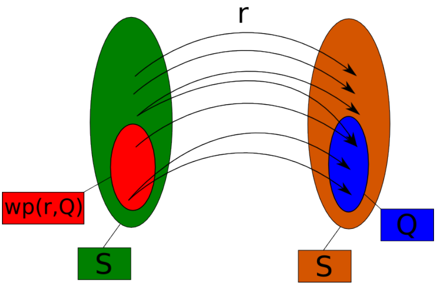
Note that this is in general not the same as $sp(Q, r^{-1})$ when then relation is non-deterministic or partial.



Lemma: Characterization of wp

$wp(r, Q)$ is the largest set P such that $\{P\}r\{Q\}$, that is:

- $\{wp(r, Q)\}r\{Q\}$
- $\forall P \subseteq S.\ \{P\}r\{Q\} \rightarrow P \subseteq wp(r, Q)$



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

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

Exercise: Postcondition of inverse versus wp

Using definitions of Hoare triple, sp, wp in Hoare logic, prove the following: If instead of good states we look at the completement set of "error states", then *wp* corresponds to doing *sp* backwards. In other words, we have the following:

$$S \setminus wp(r, Q) = sp(S \setminus Q, r^{-1})$$

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

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

Exercise: Three Forms of Hoare Triple

Show the following:
The following three conditions are equivalent:

- $\{P\}r\{Q\}$
- $P \subseteq wp(r, Q)$
- $sp(P, r) \subseteq Q$

Hoare Logic for Loop-free Code

Expanding Paths

The condition

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

is equivalent to

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

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:

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

Hoare Logic for Loops

The following inference rule holds:

$$\frac{\{P\}s\{P\}, \quad n \geq 0}{\{P\}s^n\{P\}}$$

Proof is by transitivity.
By Expanding Paths condition, we then have:

$$\frac{\{P\}s\{P\}}{\{P\}\bigcup_{n\geq 0} s^n\{P\}}$$

In fact, $\bigcup_{n\geq 0} s^n = s^*$, so we have

$$\frac{\{P\}s\{P\}}{\{P\}s^*\{P\}}$$

This is the rule for non-deterministic loops.

Exercise

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

(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 the following conjecture: For all sets S and relations $r \subseteq S \times S$ it holds:

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

She tried many sets and relations and did not find any counterexample. Is her conjecture true?
If so, prove it, otherwise provide a counterexample for which S is smallest.

Forward VCG

Some notation

If P is a formula on state and c a command, let $sp_F(P, c)$ be 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

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

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 P_s be the set of states that satisfies it:

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

Forward VCG: Using Strongest Postcondition

We can use the sp_F operator to compute verification conditions: for a triple $\{P\}c\{Q\}$ we can generate the verification condition $sp_F(P, c) \rightarrow Q$.

Assume Statement

Define:

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

Then

$$\begin{aligned}
& sp(P_s, \rho(assume(F))) \\
& = sp(P_s, \Delta_{F_s}) \\
& = \{\bar{x}' \mid \exists \bar{x} \in P_s.\ ((\bar{x}, \bar{x}') \in \Delta_{F_s})\} \\
& = \{\bar{x}' \mid \exists \bar{x} \in P_s.\ (\bar{x} = \bar{x}' \wedge \bar{x} \in F_s)\} \\
& = \{\bar{x}' \mid \bar{x}' \in P_s,\ \bar{x}' \in F_s\} \\
& = P_s \cap F_s.
\end{aligned}$$

Rules for Computing Strongest Postcondition

Havoc Statement

Define:

$$sp_F(P, \text{havoc}(x)) = \exists x_0 . P[x := x_0]$$

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:

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

Exercise

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

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

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

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

Rules for Computing Strongest Postcondition

Sequential Composition
For relations we proved

$$sp(P_s, r_1 \circ r_2) = sp(sp(P_s, r_1), r_2)$$

Therefore, define

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

Nondeterministic Choice (Branches)
We had $sp(P_s, r_1 \cup r_2) = sp(P_s, r_1) \cup sp(P_s, r_2)$. Therefore define:

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

Correctness

Show by induction on c_1 that for all P:

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

Size of Generated Formulas

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

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

Reducing sp to Relation Composition

The following identity holds for relations:

$$sp(P_s, r) = ran(\Delta_P \circ r)$$

Based on this, we can compute $sp(P_s, \rho(c_1))$ in two steps:

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

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

$$r_1 = \{(\vec{x}, \vec{x}\,'). \; F_1(\vec{x}, \vec{x}\,')\}$$

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

$$ran(r_1) = \{\vec{x}\,'. \; \exists \vec{x}. F_1(\vec{x}, \vec{x}\,')\}$$

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