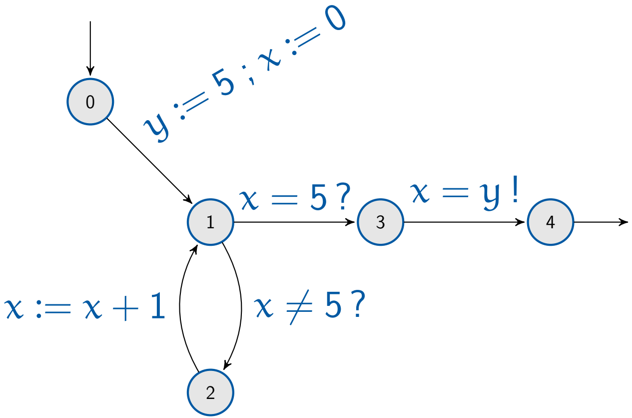
VCG: Abstraction of Loops

Vesal Vojdani

Department of Computer Science
University of Tartu

Formal Methods (2014)

WP computation was stuck in this loop



Havoc (wrong!)

- Concrete semantics:

$$[\![\text{havoc}\, x]\!]\, S = \{\sigma[x \mapsto z] \mid \sigma \in S,\ z \in \mathbb{Z}\}$$

- WP for havoc:

$$\text{WP}\, [\![\text{havoc}\, x]\!]\, \psi = \exists x : \psi$$

- Practically, all information about x is lost, except indirect relations remain:

$$\text{WP}\, [\![\text{havoc}\, x]\!]\, (y = x \wedge x = z) \implies (y = z)$$

Havoc (for post-conditions!)

- Concrete semantics:

$$[\![\mathsf{havoc}\, x]\!]\, S = \{\sigma[x \mapsto z] \mid \sigma \in S,\ z \in \mathbb{Z}\}$$

- WP for havoc:

$$\text{WP}\ [\![\mathsf{havoc}\, x]\!]\ \psi = \exists x : \psi$$

- Practically, all information about x is lost, except indirect relations remain (after the assignment):

$$\text{WP}\ [\![\mathsf{havoc}\, x]\!]\ (y = x \wedge x = z) \implies (y = z)$$

Pre-Condition of Havoc

- Concrete semantics:

$$[\![\text{havoc}\, x]\!]\, S = \{\sigma[x \mapsto z] \mid \sigma \in S,\ z \in \mathbb{Z}\}$$

- WP for havoc:

$$\text{WP}\, [\![\text{havoc}\, x]\!]\, \psi = \forall x : \psi$$

- We need ψ to hold for all values of x. Usually, we have assumes after havoc, so a typical example is

$$\text{WP}\, [\![\text{havoc}\, x]\!]\, ((y = x) \rightarrow (x = z)) \implies (y = z)$$

Pre-Condition of Havoc

- Concrete semantics:

$$[\![\text{havoc}\, x]\!]\, S = \{\sigma[x \mapsto z] \mid \sigma \in S,\, z \in \mathbb{Z}\}$$

- WP for havoc:

$$\text{WP}\, [\![\text{havoc}\, x]\!]\, \psi = \psi[x'/x] \qquad x' \text{ is fresh!}$$

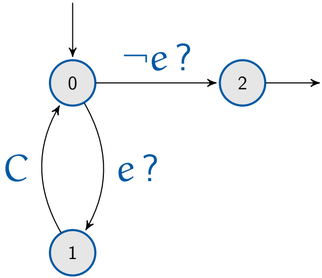
- We need ψ to hold for all values of x. Usually, we have assumes after havoc, so a typical example is

$$\text{WP}\, [\![\text{havoc}\, x]\!]\, ((y = x) \to (x = z)) \implies (y = z)$$

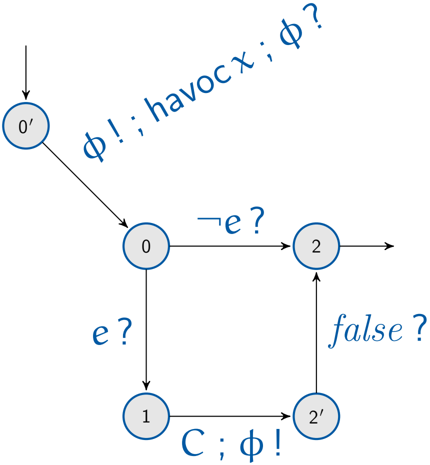
A simple assumption

- We should havoc all variables that are assigned to in the loop body.
- For simplicity, we assume this is only x.
- (You may think of x as a vector.)

Normal While Loop



Abstraction using invariant ϕ



Why can we do this?

- The construction guarantees that if

$$\perp \notin S_2$$

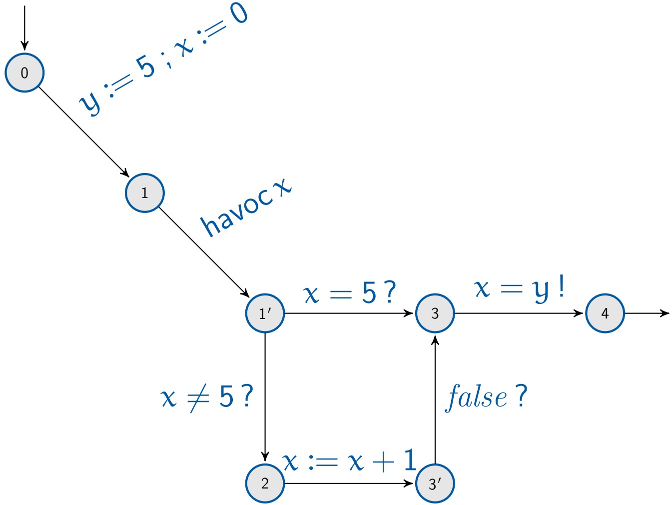
  we have

$$S_2' \subseteq S_2$$

  where S_i' are the sets computed for the original while loop.

- Note: it follows very closely the proof rules of Hoare logic.

Now we really can compute a VC



What happened?

- Well, there was no invariant to check.
- That's good because the invariant was trivial.
- The homework requires making this construction with an invariant.

- Just a note on procedure, and then we prove the soundness of the construction.

Procedure Calls

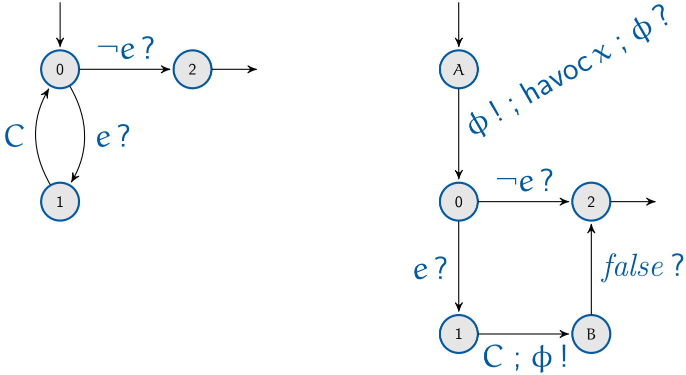
- Given a function P with parameter p and result r and contract

$$(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$$

- We produce the following translation for a call $x = P(e)$.

$$\begin{aligned} & p := e \\ & \phi\,! \\ & \psi\,? \\ & x := r \end{aligned}$$

Soundness of the transformation

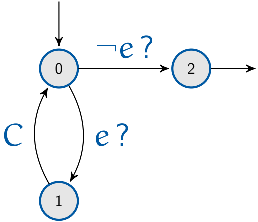


Proof Plan

1. Write down constraint systems S and S'.
2. Separate assertions into
   - the conditions they impose
   - constraint system for values
3. Show that the value system satisfies the constraints of S.
4. This implies that any solution of S' is greater than the least solution of S.

Constraint System S

$$S_0 \supseteq S$$
$$S_0 \supseteq [\![C]\!]\, S_1$$
$$S_1 \supseteq [\![e\,?]\!]\, S_0$$
$$S_2 \supseteq [\![\neg e\,?]\!]\, S_0$$



Constraint System S'

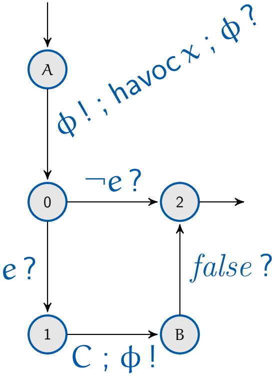
$$S'_A \supseteq S$$

$$S'_0 \supseteq [\![\phi\,?]\!]\,\{\sigma[x \mapsto z] \mid z \in \mathbb{Z},\ \sigma \in [\![\phi\,!]\!]\,S'_A\}$$

$$S'_1 \supseteq [\![e\,?]\!]\,S'_0$$

$$S'_B \supseteq [\![\phi\,!]\!]\,([\![C]\!]\,S'_1)$$

$$S'_2 \supseteq [\![\neg e\,?]\!]\,S'_0 \cup \{\bot \mid \bot \in S'_B\}$$



Splitting S' based on $\bot \in S'_2$

- We can be sure $\bot \notin S'_2$ if we have

$$S \models \phi$$
$$[\![C]\!]\, S'_1 \models \phi$$

- Letting $S_x = \{\sigma[x \mapsto z] \mid z \in \mathbb{Z}, \sigma \in S\}$, the following constraints remain:

$$S'_0 \supseteq [\![\phi\ ?]\!]\, S_x$$
$$S'_1 \supseteq [\![e\ ?]\!]\, S'_0$$
$$S'_2 \supseteq [\![\neg e\ ?]\!]\, S'_0$$

Splitting S' based on $\bot \in S'_2$

- We can be sure $\bot \notin S'_2$ if we have

$$S \models \phi$$
$$[\![C]\!]\, S'_1 \models \phi$$

- Letting $S_x = \{\sigma[x \mapsto z] \mid z \in \mathbb{Z}, \sigma \in S\}$, we obtain the following solution:

$$S'_0 = \{\sigma \in S_x \mid \sigma \models \phi\}$$
$$S'_1 = \{\sigma \in S_x \mid \sigma \models \phi \wedge e\}$$
$$S'_2 = \{\sigma \in S_x \mid \sigma \models \phi \wedge \neg e\}$$

Solution to original system?

- Given the solution and conditions:

$$S'_0 = \{\sigma \in S_x \mid \sigma \models \phi\} \qquad S \models \phi$$
$$S'_1 = \{\sigma \in S_x \mid \sigma \models \phi \wedge e\} \qquad [\![C]\!]\, S'_1 \models \phi$$
$$S'_2 = \{\sigma \in S_x \mid \sigma \models \phi \wedge \neg e\}$$

- We check if the original constraints are satisfied:

$$S'_0 \supseteq S \qquad S'_0 \supseteq [\![C]\!]\, S'_1$$
$$S'_1 \supseteq [\![e\,?]\!]\, S'_0 \qquad S'_2 \supseteq [\![\neg e\,?]\!]\, S'_0$$

What did we just do?

- We had two systems:

$$X \supseteq F(X)$$
$$X \supseteq F'(X)$$

- We showed that for any Y

$$Y \supseteq F'(Y) \implies Y \supseteq F(Y)$$

- What did we conclude?