REACHABILITY AND VERIFICATION

- Let $T \subseteq S \times S$ be the set of transitions ($\hookrightarrow$) defined in the previous slides.
  - Is T finite?
  - Is T defined for a specific program c or for any program?
- Given a program c, a sequence of transitions $(\sigma_0, \mathsf{c}) \hookrightarrow (\sigma_1, \mathsf{c}_1) \ldots \hookrightarrow (\sigma_n, \mathsf{c}_n)$ is called an execution of c.
  - A program state σ is called reachable if there exists an execution $(\sigma_0, \mathsf{c}) \hookrightarrow \ldots \hookrightarrow (\sigma, \mathsf{c}_n)$ which ends in the state σ.
- Verification Problem: Is $(Error, \mathsf{c}')$ reachable for some c′?
  - Program c is called safe if the error state is not reachable.
  - What about the initial state?

EXAMPLE

```
assume(i = 0 ∧ n ≥ 0);
while(i < n) do
  i := i + 1;
assert(i = n);
```

- Is $(Error, c')$ reachable?

PRE/POST-CONDITIONS AND VERIFICATION

- Alternatively, we can express the Verification problem in terms of pre-conditions and post-conditions.
- A program c satisfies the specification $\{P\}\mathsf{c}\{Q\}$ if:
  - $\forall \sigma, \sigma' .\ \sigma \vDash P \wedge (\sigma, \mathsf{c}) \hookrightarrow^* (\sigma', \mathsf{skip}) \rightarrow \sigma' \vDash Q$
- $\{P\}\mathsf{c}\{Q\}$ is also called a 'Hoare Triple'.
- If c satisfies the specification $\{P\}\mathsf{c}\{Q\}$, then we also say that the Hoare Triple $\{P\}\mathsf{c}\{Q\}$ is valid.

TOTAL CORRECTNESS

- Both ways of specifying the verification problem deal with Partial Correctness
  - They only consider terminating executions. Non-terminating executions trivially satisfy both definitions.
- Total Correctness also requires all program executions to be of finite length.
- A program c satisfies the specification $[P]\mathsf{c}[Q]$ if
  - $\forall \sigma \,.\, \sigma \models P \Rightarrow \exists n, \sigma' \,.\, (\sigma, \mathsf{c}) \hookrightarrow^{n} (\sigma', \mathsf{skip}) \wedge \sigma' \models Q$

EXAMPLES OF HOARE TRIPLES

- What can be said about the following triples?
  - $\{true\}$ c $\{Q\}$
  - $\{false\}$ c $\{Q\}$
  - $\{P\}$ c $\{true\}$
  - $\{true\}$ c $\{false\}$
- Partial and total correctness
  - Is $\{x = 0\}$ while(x $\geq$ 0) do x:=x+1 $\{x = 1\}$ valid?
  - What about $[x = 0]$ while(x $\geq$ 0) do x:=x+1 $[x = 1]$?

AUTOMATED VERIFICATION

- We will reduce the verification problem to the satisfiability problem (modulo theories) in FOL.
  - First, we will consider the 'reachability of error states'-based definition of verification.
- Let us encode the semantics of every individual command in FOL.
- If V is the set of variables used in a program c, then an FOL formula $F[V]$ encodes a set of states of the program.
  - E.g. If $V = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$, then the formula $\mathsf{x} + \mathsf{y} > 0$ encodes the set of states $\{(\mathsf{x} \mapsto m, \mathsf{y} \mapsto n, \mathsf{z} \mapsto o) \mid m + n > 0\}$

AUTOMATED VERIFICATION

- If $(\sigma, \text{c}) \hookrightarrow (\sigma', \text{skip})$, then we will use the FOL formula $\rho(\text{c})[V, V']$ to encode the states σ and σ'.
- All states σ, σ', such that $(\sigma, \text{c}) \hookrightarrow (\sigma', \text{skip})$ are satisfying interpretations of formula $\rho(\text{c})[V, V']$ (with the domain of σ' being V').
  - E.g. $\rho(\text{x:=y+1}) \triangleq \text{x}' = \text{y} + 1 \wedge \text{y}' = \text{y}$
- We will a special variable error $\in V$ to indicate the $Error$ state (obtained after assertion failure). error $= 0$ indicates a non-error state.

SEMANTICS IN FOL

For $U \subseteq V$, we define frame(U) to be the formula $\bigwedge_{\mathrm{v} \in V \setminus U} \mathrm{v}' = \mathrm{v}$

- E.g. $V = \{x, y, z\}$, $\mathit{frame}(\mathrm{x}) \triangleq (\mathrm{y}' = \mathrm{y}) \wedge (\mathrm{z}' = \mathrm{z})$

Now, the semantics of commands in FOL can be defined as follows:

- $\rho(\mathrm{x{:=}e}) \triangleq \mathrm{x}' = \mathrm{e} \wedge \mathit{frame}(\mathrm{x})$
- $\rho(\mathrm{x{:=}havoc}) \triangleq \mathit{frame}(\mathrm{x})$
- $\rho(\mathrm{assume(F)}) \triangleq \mathrm{F} \wedge \mathit{frame}(\varnothing)$
- $\rho(\mathrm{assert(F)}) \triangleq \mathrm{F} \rightarrow \mathit{frame}(\varnothing)$

STRONGEST POST-CONDITION

SYMBOLIC EXECUTION IN THE FORWARD DIRECTION

- Given a set of states S and a command c, the strongest post-condition operator $sp(S, c)$ consists of all states that can be obtained after executing c on any state in S.

$$sp(S, c) \triangleq \{\sigma' \mid \exists \sigma \in S\,.\,(\sigma, c) \hookrightarrow^* (\sigma', \mathsf{skip})\}$$

- We can use a FOL formula F to represent a set of states.
- The symbolic strongest post-condition operator can be defined as:

$$\sigma' \vDash sp(F, \mathsf{c}) \Leftrightarrow \exists \sigma\,.\,\sigma \vDash F \wedge (\sigma, \mathsf{c}) \hookrightarrow (\sigma', \mathsf{skip})$$

- We can now use the semantics in FOL (ρ) to define symbolic sp:

$$sp(F, \mathsf{c}) \triangleq (\exists V\,.\,F \wedge \rho(\mathsf{c}))[V/V']$$

FIRST ELIMINATE EXISTENTIAL QUANTIFICATION ON V, THEN SUBSTITUTE V FOR V'