Program Verification

Floyd 1967, Hoare 1969, many others

Textbook: Harel, Kozen, Tiuryn: Dynamic Logic (2000)

Specification ⇝ Program

- Specification says *what* has to be computed
- Program says *how* it is computed

Specification: (A, B) where $A, B \subseteq \Sigma$

A: precondition
B: postcondition

p partially correct for (A, B) if
$\forall (\sigma, \sigma') \in R_p : \sigma \in A \Rightarrow \sigma' \in B$

if p succeeds for a state in A, it yields a state in B

p totally correct for (A, B) if
p partially correct for (A, B) and
$A \subseteq \mathrm{Dom}(R_p)$

p succeeds for all states in A with states in B

Example

$$A = X \doteq X_0 \wedge X_0 \geq 0$$
$$B = Z \doteq X_0 \cdot Y \wedge Y \doteq 0$$

A, B are described by formulas

$A = \{\sigma \mid \sigma X = \sigma X_0 \wedge \sigma X_0 \geq 0\}$

$B = \{\sigma \mid \sigma Z = \sigma X_0 \cdot \sigma Y \wedge \sigma Y = 0\}$

- If p can only assign X and Z, then (A, B) determines R_p completely
- If p does not assign X_0 and Y, then (A, B) determines R_p on X, X_0, Y and Z

Weakest Preconditions

$\mathcal{N} \in \wp(\Sigma^2) \to \wp\Sigma \to \wp\Sigma$ necessity operator

$\mathcal{N} R B = \{\sigma \in \Sigma \mid \forall \sigma' \in \Sigma : (\sigma, \sigma') \in R \Rightarrow \sigma' \in B\}$

weakest precondition for R and B

$$p \text{ part. correct for } (A, B) \iff A \subseteq \mathcal{N}(R_p) B$$

$$A \subseteq \mathrm{Dom}(R_p) \iff A \cap \mathcal{N}(R_p)\emptyset = \emptyset$$

$\mathcal{N}(R_p)\emptyset$ is the set of all states for which p cannot succeed

Notations

$[p]B \stackrel{\text{def}}{=} \mathcal{N}(\mathcal{R}p)B$ necessity

$b \stackrel{\text{def}}{=} \{\sigma \in \Sigma \mid \mathcal{R}b\sigma = 1\}$

$A \wedge B \stackrel{\text{def}}{=} A \cap B$

$A \vee B \stackrel{\text{def}}{=} A \cup B$

$\bar{A} \stackrel{\text{def}}{=} \Sigma - A$

$A \to B \stackrel{\text{def}}{=} \bar{A} \vee B$

$\phi, \Sigma, \wedge, \vee, \neg$ satisfy Boolean laws

Assertions

$\mathcal{R}\mathcal{D} = \Sigma \to \mathcal{B}$ is an infinite Boolean algebra

- An assertion is a description of a subset $A \subseteq \Sigma$
- Assertions can be formally represented in S by extending the signature and standard interpretation for regular programs as follows:

$0, 1 : \mathcal{B}$ $\quad \mathcal{R}0\sigma = 0, \quad \mathcal{R}1\sigma = 1$

$\wedge, \vee : \mathcal{B} \to \mathcal{B} \to \mathcal{B}$ $\quad \mathcal{R}\wedge fg\sigma = f\sigma \wedge g\sigma, \quad \mathcal{R}\vee fg\sigma = f\sigma \vee g\sigma$

$[\,] : P \to \mathcal{B} \to \mathcal{B}$ $\quad \mathcal{R}[\,]Rf\sigma = (\forall \sigma' \in \Sigma : (\sigma,\sigma') \in R \Rightarrow f\sigma')$

$\{\sigma \mid \mathcal{R}b\sigma = 1\} \subseteq \{\sigma \mid \mathcal{R}b'\sigma = 1\} \iff \mathcal{R} \models b \to b' = 1$

Assignment ⟶ Substitution

$\mathcal{R} \models [X := a]b = b[X := a]$ eliminates necessity

provided b is an assertion not containing necessities and $b[X:=a]$ is obtained from b by replacing all occurrences of X with a.

Example

$[Z := Z + Y](X_0 \cdot Y \doteq Z + X \cdot Y) = X_0 \cdot Y \doteq (Z + Y) + X \cdot Y$

Equational Properties

Elimination rules for necessities

$[b?]B = b \to B$

$[p;p']B = [p]([p']B)$

$[p+p']B = [p]B \wedge [p']B$

$[p^*]B = \bigcup \{I \subseteq B \mid I \subseteq [p]I\}$

$[\text{if } b \text{ then } p \text{ else } p']B = b \wedge [p]B \vee \bar{b} \wedge [p']B$

Implicational Properties

I is referred to as invariant

$I \subseteq [p]I \Rightarrow I \subseteq [p^*]I$ — invariant rule for p^*

$I \wedge b \subseteq [p]I \Rightarrow I \subseteq [\text{while } b \text{ do } p](I \wedge \overline{b})$ — invariant rule for while

$A \subseteq [p]B \wedge A' \subseteq A \wedge B \subseteq B' \Rightarrow A' \subseteq [p]B'$ — Weakening

$A \subseteq [p]B \wedge B \subseteq [p']C \Rightarrow A \subseteq [p;p']C$ — Chaining

Verification Strategy

prove $A \subseteq [p]B$

algorithmic — eliminate loops

abstract verification conditions

creative — find invariants

concrete verification conditions

ordinary proofs — check

done

Example (abstract VCs and invariant)

$p =$ $z := 0$; while $y \geq 1$ do ($z := z + y$; $y := y - 1$)

$A \subseteq [p]B \iff \exists I \subseteq \Sigma:$ — abstract verification conditions

(1) $A \subseteq [z := 0]I \wedge$ — Chaining

(2) $I \wedge y \geq 1 \subseteq [z := z + y;\ y := y - 1]I \wedge$ — Invariant Rule

(3) $I \wedge y \leq 0 \subseteq B$ — Weakening

All 3 conditions are satisfied for

$A = y \doteq x_0 \wedge x_0 \geq 0$ — precondition

$B = z \doteq x_0 \cdot y \wedge y \doteq 0$ — postcondition

$I = x_0 \cdot y \doteq z + y \cdot y \wedge y \geq 0$ — invariant

Example (checking of concrete VCs)

$A \subseteq [z := 0]I$

$[z := 0]I$
$= x_0 \cdot y \doteq 0 + y \cdot y \wedge y \geq 0$
$\supseteq x = x_0 \wedge x_0 \geq 0$
$= A$

(1)

$I \wedge y \geq 1 \subseteq [z := z + y;\ y := y - 1]I$

$[z := z + y;\ y := y - 1]I$
$= x_0 \cdot y \doteq z + y + (y - 1)y \wedge y - 1 \geq 0$
$= x_0 \cdot y \doteq z + y \cdot y \wedge y \geq 1$
$= I \wedge y \geq 1$

(2)

substitution steps

$I \wedge y \leq 0 \subseteq B$

$I \wedge y \leq 0$
$= x_0 \cdot y \doteq z + y \cdot y \wedge y \doteq 0$
$= z \doteq x_0 \cdot y \wedge y \doteq 0$
$= B$

(3)

$A = y \doteq x_0 \wedge x_0 \geq 0$ $\quad B = z \doteq x_0 \cdot y \wedge y \doteq 0$

$I = x_0 \cdot y \doteq z + y \cdot y \wedge y \geq 0$

Combine Program Construction and Verification

Advocated by Hoare, Dijkstra, Gries, ...

$B = \boxed{z \doteq X_0 \cdot Y_0}$ postcondition

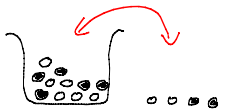
$I = \boxed{X_0 \cdot Y_0 \doteq z + Y \cdot Y \wedge Y \geq 0}$ invariant decrement Y until $Y = 0$

$A = \boxed{Y \doteq Y_0 \wedge Y_0 \geq 0}$ precondition

$P = \boxed{z := 0 \,;\ \text{while } Y \geq 1 \text{ do } (z := z + X_0 \,;\ Y := Y - 1)}$ program

proceed in this order

Coffee Beans Revisited



Given A pot with white and black beans

X_0: initial number of white beans

Y_0: initial number of black beans

$X_0 \geq 0 \wedge Y_0 \geq 0 \wedge X_0 + Y_0 \geq 1$

Wanted Algorithm that takes beans in and out such that

1) exactly one bean remains
2) remaining bean is black iff X_0 even
3) number of beans in pot is decreased by 1 by each round

Coffee Beans Continued

$A = X_0 \geq 0 \wedge Y_0 \geq 0 \wedge X_0 + Y_0 \geq 1$

$B = Y \doteq X_0 \bmod 2 \wedge Y + X = 1$

$I = X \bmod 2 \doteq X_0 \bmod 2 \wedge Y \geq 0 \wedge X \geq 0 \wedge X + Y \geq 1$

- decrease X in steps of 2
- decrease Y

$P =$ $X := X_0$; $Y := Y_0$;

do $X \geq 2 \Rightarrow X := X - 2$; $Y := Y + 1$

$\square$ $Y \geq 2 \Rightarrow Y := Y - 1$

$\square$ $X \geq 1 \wedge Y \geq 1 \Rightarrow Y := Y - 1$

od

Example: Fast Exponentiation

Task: Compute x^n in $O(\log n)$ where $x \in \mathbb{Z}$ and $n \in \mathbb{N}$

$A = N_0 \geq 0$

$B = z \doteq X_0^{N_0}$

$I = X_0^{N_0} \doteq z \cdot X^N$ decrease N until $N = 0$; $x^n = (x^2)^{n/2}$

$P =$ $N := N_0$; $X := X_0$; $z := 1$;

while $N \geq 1$ do

if $N \bmod 2 = 0$ then $X := X \cdot X$; $N := N \operatorname{div} 2$

else $z := z \cdot X$; $N := N - 1$