Software Systems Verification and Validation

Lecture 08 - Correctness - Dijkstra

Lect. dr. Andreea Vescan

Babeș-Bolyai University
Cluj-Napoca

2015-2016

1. Program verification
   - Floyd Method - Inductive assertions
   - Hoare Logic
2. Dijkstra's Language
   - Guarded commands, Nondeterminacy and Formal Derivation of Prog
   - Guarded commands, Nondeterminacy
   - Formal Derivation of Programs
3. Developing correct programs from specification
   - Refinement
   - Rules of Refinement
   - Examples
4. Static analysis
   - JML- Java Modeling Language
   - ESC/Java2- Extended Static Checker for Java
5. Next lecture
   - Next lecture

Program verification

- Program verification
  - proof-based, computer-assisted, program-verification approach, mainly used for programs which we expect to terminate and produce a result
  - model-based, automatic, property-verification approach, mainly used for concurrent, reactive systems (originally used in a post-development stage) - model checking

Floyd Method - Inductive assertions

Figure: Robert W Floyd (June 8, 1936 - September 25, 2001)

Hoare triples

Figure: Charles Antony Richard Hoare (11 January 1934, Colombo, Sri Lanka)

Edsger Wybe Dijkstra

Figure: Edsger Wybe Dijkstra (May 11, 1930 - August 6, 2002)

Guarded command

- "guarded command" - a statement list prefixed by a boolean expression: only when this boolean expression is initially true, is the statement list eligible for execution
- $< guarded\ command >::=< guard >\rightarrow< guarded\ list >$
- $< guard >::=< boolean\ expression >$
- $< guarded\ list >::=< statement > \{; < statement >\}$
- $< guarded\ command\ set >::=$
  $< guarded\ command > \{\square < guarded\ command >\}$
- $< alternative\ construct >::=$ **if** $< guarded\ command\ set >$ **fi**
- $< repetitive\ construct >::=$ **do** $< guarded\ command\ set >$ **do**
- $< statement >::=< alternative\ construct > \ |$
  $< repetitive\ construct > \ |$ "*other statements*"

Nondeterminacy

- Example 1

  if $x \geq y \rightarrow m := x$

  $\square y \geq x \rightarrow m := y$

  fi

- Example 2 - compute k s.t. for fixed value n and fixed function *f(i) (defined for* $0 \leq i < n$*), k will eventually satisfy* $0 \leq k < n$ *and* $(\forall i : 0 \leq i < n : \; f(k) \geq f(i))$.

  $k := 0;\; j := 1;$

  do $j \neq n \rightarrow$ **if** $f(j) \leq f(k) \rightarrow j := j + 1$

  $\qquad\qquad \square f(j) \geq f(k) \rightarrow k := j;\; j := j + 1$

  $\qquad\qquad$ **fi**

  od

Weakest pre-conditions

- Hoare - introduced sufficient pre-conditions such that the mechanism will no produce the wrong result
  but may fail to terminate.
- Dijkstra - introduced necessary and sufficient pre-conditions such that the mechanism are guaranteed to produce the right result.
  $=$ weakest pre-conditions
- $wp(S,\ R)$, where S denotes a statement list, R some condition on the state of the system.
- *wp* - called a " predicate transformer" - because it associates a pre-condition to any post-condition R.

Properties of *wp*

1. Law of the Excluded Miracle
   For any S, for all states: $wp(S, F) = F$
2. For any S and any two post-conditions, such that for all states $P \Rightarrow Q$, for all states:
   $$wp(S, P) \Rightarrow wp(S, Q)$$
3. For any S and any two post-conditions P and Q, for all states:
   $$wp(S, P) \textbf{ and } wp(S, Q) = wp(S, P \textbf{ and } Q)$$
4. For any deterministic S and any post-conditions P and Q, for all states:
   $$(wp(S,\ P) \textbf{ or } wp(S,\ Q)) \Rightarrow wp(S, P \textbf{ or } Q)$$

Assignment and concatenation operator

- Assignment
  The semantics of $x := E$ are given by:
  $wp(``x := E'', R) = R^x_E$, R^x_E -denotes a copy of the predicate defining R in which each occurrence of the variable x is replaced by E.
- Concatenation operator - ;
  The semantics of the ; operator are given by:
  $wp(``S1 \, ; S2'', R) = wp(S1, wp(S2, R))$.

The Alternative Construct

- Let IF denote: $\mathbf{if} B_1 \rightarrow SL_1 \square ... \square B_n \rightarrow SL_n \mathbf{fi}$
  Let BB denote: $(\exists i: \ 1 \leq i \leq n: \ B_i)$, then, by definition
  $wp(IF, R) = (BB \ and \ (\forall i: \ 1 \leq i \leq n: \ B_i \Rightarrow wp(SL_i, R)))$.
- Theorem 1
  From $(\forall i: 1 \leq i \leq n: \ (Q \ \mathbf{and} \ B_i) \Rightarrow wp(SL_i, R)$ for all states we can conclude that $(Q \ \mathbf{and} \ BB) \Rightarrow wp(IF, R)$ holds for all states.)
- Let t denote some integer function, and $wdec(S, t)$
- Theorem 2
  From $(\forall i: 1 \leq i \leq n: (Q \ \mathbf{and} \ B_i) \Rightarrow wdec(SL_i, t))$ for all states we can conclude that $(Q \ \mathbf{and} \ BB) \Rightarrow wdec(IF, t)$ hold for all states.
- By definition,
  $wdec(S, t) = (tmin(X) \leq t(X) - 1) = (tmin(X) < t(X))$.

The Alternative Construct - example

- The formal requirements for performing $m := max(x, y)$ is: $R: \ (m = x \textbf{ or } m = y) \textbf{ and } m \geq x \textbf{ and } m \geq y.$
- Assignment $m := x$ for $m = x$? $wp(\text{“}m := x\text{''}, R) = (x = x \textbf{ or } x = y) \textbf{ and } x \geq x \textbf{ and } x \geq y = x \geq y$
- Theorem 1: $\textbf{if} x \geq y \rightarrow m := x \textbf{fi}$
- But $B \neq T$, so we weakening BB means looking for alternatives which might introduce new guards.
- Alternative: “$m := y$'' that introduces the new guard
  $wp(\text{“}m\text{''} := y, R) = y \geq x$
  $\textbf{if} x \geq y \rightarrow m := x$
  $\square y \geq x \rightarrow m := y$
  $\textbf{fi}$

The Repetitive Construct

- Let DO denote: $\mathbf{do} B_1 \rightarrow SL_1 \Box ... \Box B_n \rightarrow SL_n \mathbf{do}$
  Let $H_0 = (R \textbf{ and non } BB)$
  and for $k > 0$, $H_k(R) = (wp(IF, H_{k-1}(R))) \textbf{ or } H_0(R)$
  then, by definition: $wp(DO, R) = (\exists k : k \geq 0 : H_k(R))$.
- Theorem 3
  If we have all the states
  $(P \textbf{ and } BB) \Rightarrow (wp(IF, P) \textbf{ and } wdec(IF, t) \textbf{ and } t \geq 0)$ we can conclude that we have for all states $P \Rightarrow wp(DO, P \textbf{ and non } BB)$
- T is the condition satisfied by all states, and $wp(S, T)$ is the weakest pre-condition guaranteeing proper termination of S.
- Theorem 4
  From $(P \textbf{ and } BB) \Rightarrow wp(IF, P)$ for all states, we can conclude that we have for all states
  $(P \textbf{ and } wp(DO, T) \Rightarrow wp(DO, P \textbf{ and non } BB))$

The Repetitive Construct - example

- The greatest common divisor: $x = gcd(X, Y)$
- Choose an invariant relation and variant function.
  establish the relation P to be kept invariant
  do "decrease t as long as possible under variance of P" **od**
- invariant relation (established by $x := X; y := Y$):
  $P : gcd(X, Y) = gcd(x, y)$ **and** $x > 0$ **and** $y > 0$
- $(P$ **and** $B) \Rightarrow wp(``x, y : E1, E2'', P))$
  $= (gcd(X, Y) = gcd(E1, E2)$ *and* $E1 > 0$ **and** $E2 > 0)$.
  - $gcd(X, Y) = gcd(E1, E2)$ is implied by P
  - invariant for (x, y) : $wp(``x := x - y, P) = (gcd(X, Y) = gcd(x - y, y)$ **and** $x - y > 0$ **and** $y > 0)$, and guard $x > y$
  - decrease of the variant function $t = x + y$
    $wp(``x := x - y'', t \leq t_0) = (x \leq t_0)$
    $tmin = x$, $wdec(''x := x - y'', t) = (x < x + y) = y > 0$

The Repetitive Construct - example

- x:=X; y:=Y
  do $x > y \rightarrow x := x - y$ **od**
- But P **and** BB - are not allowed to conclude $x = gcd(X, Y)$
  the alternative $y := y - x$ requires a guard $y > x$
- x:=X; y:=Y
  do $x > y \rightarrow$ x:=x-y
  $\square y > x \rightarrow y := y - x$
  od

Refinement

- Input data: X $\qquad \varphi(X)$
  Output data: Z $\qquad \psi(X, Z)$
- Abstract program
  $Z : [\varphi, \ \psi]$
- Refinement
  $P_1 \prec P_2 \prec ... \prec P_{n-1} \prec P_n$
- Rules of refinement
  - Assignment rule
  - Sequential composition rule
  - Alternation rule
  - Iteration rule

Refinement

- Assignment rule: $[\varphi(v/e), \psi] \prec v := e$
- Sequential composition rule ($\gamma - middlepredicate$)

$$
\begin{aligned}
[\eta_1, \eta_2] \prec\ & [\eta_1, \gamma] \\
& [\gamma, \eta_2]
\end{aligned}
$$

- Alternation rule, $G = g_1 \vee g_2 \vee \ldots \vee g_n$

$$
\begin{array}{l}
[\eta_1, \eta_2] \prec \\
\textbf{if } g_1 \rightarrow [\eta_1 \wedge g_1, \eta_2] \\
\quad \Box g_2 \rightarrow [\eta_1 \wedge g_2, \eta_2] \\
\quad \vdots \\
\quad \Box g_n \rightarrow [\eta_1 \wedge g_n, \eta_2] \\
\textbf{fi}
\end{array}
$$

Refinement

- Iteration rule $G = g_1 \vee g_2 \vee ... \vee g_n$
  $[\eta, \eta \wedge \neg G] \prec$
  do $g_1 \rightarrow [\eta \wedge g_1, \eta \wedge TC]$
  $\quad \Box g_2 \rightarrow [\eta \wedge g_2, \eta \wedge TC]$
  $\quad \vdots$
  $\quad \Box g_n \rightarrow [\eta \wedge g_n, \eta \wedge TC]$
  do

Examples

- See the file with the examples

JML

- Tutorial JML
- Demo JML

ESC/Java2

- Tutorial ESCJava2
- Demo ESCJava2

Next lecture

-
- Lecture 09 - Compulsory Attendance
- Date:22 April 2016 - FRIDAY
- Hours: 12:00-14:00
- Room: A2, FSEGA Building
- ISDC - presentation
- Quality