Software Verification
Exercise Solution: Software Model Checking

The routine we consider is:

\[
\text{always\_positive} \ (x: \text{INTEGER}): \text{INTEGER} \\
\text{if } x > 0 \text{ then} \\
\quad \text{Result} := x + x \\
\text{else} \\
\quad \text{if } x = 0 \text{ then} \\
\quad \quad \text{Result} := 1 \\
\quad \text{else} \\
\quad \quad \text{Result} := x \times x \\
\end{align*}
\]

\[
\text{end} \\
\text{ensure Result} > 0 \text{ end}
\]

(a) We build the predicate abstraction of \text{always\_positive} in an incremental fashion.
1. Normalize the conditions appearing in conditionals and loops. We get

\[
\text{always\_positive\_1} \ (x: \text{INTEGER}): \text{INTEGER} \\
\text{if } ? \text{ then} \\
\quad \text{assume } x > 0 \\
\quad \text{Result} := x + x \\
\text{else} \\
\quad \text{assume } x \leq 0 \\
\quad \text{if } ? \text{ then} \\
\quad \quad \text{assume } x = 0 \\
\quad \quad \text{Result} := 1 \\
\quad \text{else} \\
\quad \quad \text{assume } x \neq 0 \\
\quad \quad \text{Result} := x \times x \\
\end{align*}
\]

\[
\text{end} \\
\text{ensure Result} > 0 \text{ end}
\]

2. We rewrite the assume statements, and apply common simplifications to the logical formulae as well as peephole optimizations.
   i. For \text{assume } x > 0:
\[ \neg \text{Pred}(\neg x > 0) = \neg \text{Pred}(x <= 0) = \neg \neg \text{pos} = \text{pos} \]

So we will add \textbf{assume} \ pos.

We must also take into account the effect of the assume statement on pos and Rpos:

For pos:
\[
\text{wp(assume } x > 0, \ x > 0) = (x > 0 => x > 0) = \text{True} \\
\text{Pred(True) = True} \\
\text{So we must include the update} \\
\text{if True then pos := True else if ... else ... end},
\]
which simplifies to \text{pos := True}.

For Rpos:
\[
\text{wp(assume } x > 0, \ \text{Result} > 0) = (x > 0 => \text{Result} > 0) \\
\text{Pred(x > 0 => Result > 0) = (pos => Rpos).} \\
\text{Similarly, wp(assume } x > 0, \ \text{Result} <= 0) = (x > 0 => \text{Result} <= 0) \\
\text{Pred(x > 0 => Result <= 0) = (pos => \neg Rpos)} \\
\text{So we get} \\
\text{if pos => Rpos then} \\
\quad \text{Rpos := True} \\
\text{else if pos => \neg Rpos then} \\
\quad \text{Rpos := False} \\
\text{else Rpos := ?} \\
\text{end}
\]
Since we just assumed pos in the code, we can apply the peephole optimization and remove this update, since it will have no effect on the value of Rpos.

Hence \textbf{assume} \ x > 0 becomes \textbf{assume} \ pos; \text{pos := True} in the abstraction.

\[ \cdot \]

\textbf{ii. For assume } x <= 0:
\[ \neg \text{Pred}(\neg x <= 0) = \neg \text{Pred}(x > 0) = \neg \text{pos} \]
So we will add \textbf{assume} \ \neg \text{pos} in the abstraction.

The effect on pos:
\[
\text{wp(assume } x <= 0, \ x > 0) = (x <= 0 => x > 0) = x > 0 \\
\text{Pred(x > 0) = pos.} \\
\text{Similarly, wp(assume } x <= 0, \ x <= 0) = (x <= 0 => x <= 0) = \text{True} \\
\text{Pred(True) = True.} \\
\text{So we have} \\
\text{if pos then} \\
\quad \text{pos := True} \\
\text{else if True then} \\
\quad \text{pos := False} \]
else pos := ? end
Since we just assumed \( \neg \text{pos} \) in the abstraction, we can simplify this update to pos := False.

The effect on Rpos:
\[
\text{wp(assume } x \leq 0, \text{ Result} > 0) = (x <= 0 \Rightarrow \text{Result} > 0) \\
\text{Pred}(x <= 0 \Rightarrow \text{Result} > 0) = (\neg \text{pos} \Rightarrow \text{Rpos}) = \text{Rpos} \text{ because of a peephole optimization.}
\]
\[
\text{wp(assume } x <= 0, \text{ Result} <= 0) = (x <= 0 \Rightarrow \text{Result} <= 0) \\
\text{Pred}(x <= 0 \Rightarrow \text{Result} <= 0) = (\neg \text{pos} \Rightarrow \neg \text{Rpos}) = \neg \text{Rpos} \text{ because of the same peephole optimization.}
\]
So we have
\[
\text{if Rpos then} \\
\quad \text{Rpos := True} \\
\text{else if } \neg \text{Rpos then} \\
\quad \text{Rpos := False} \\
\text{else Rpos := ? end}
\]
which can be eliminated.

Hence assume \( x \leq 0 \) becomes assume \( \neg \text{pos} \); pos := False.

iii. For assume \( x = 0 \):
\[
\neg \text{Pred}(\neg x = 0) = \neg \text{pos}
\]
So we will add assume \( \neg \text{pos} \) in the abstraction.

The effect on pos:
\[
\text{wp(assume } x = 0, x > 0) = (x = 0 \Rightarrow x > 0) = (x /= 0) \\
\text{Pred}(x /= 0) = \text{pos}
\]
Similarly, \[
\text{wp(assume } x = 0, x \leq 0) = (x = 0 \Rightarrow x \leq 0) = \text{True} \\
\text{Pred(} \text{True} \text{) = True}
\]
So we have the update:
\[
\text{if pos then} \\
\quad \text{pos := True} \\
\text{else if } \text{True then} \\
\quad \text{pos := False} \\
\text{else pos := ? end}
\]
which becomes pos := False when we do a peephole simplification.

The effect on Rpos:
\[
\text{wp(assume } x = 0, \text{ Result} > 0) = (x = 0 \Rightarrow \text{Result} > 0) \\
\text{Pred}(x = 0 \Rightarrow \text{Result} > 0) = \text{pos} \lor \text{Rpos}
\]
Similarly, \[
\text{wp(assume } x = 0, \text{ Result} \leq 0) = (x = 0 \Rightarrow \text{Result} \leq 0) \\
\text{Pred}(x = 0 \Rightarrow \text{Result} \leq 0) = \text{pos} \lor \neg \text{Rpos}
\]
Applying peephole optimization, we see that the update will not have any effect and can be removed.
Hence assume $x = 0$ becomes assume $\neg$ pos; pos := False.

iv. For assume $x \neq 0$:
   $\neg$ Pred($\neg x = 0$) = $\neg$ Pred(x = 0) = $\neg$ False = True
So we do not need to add an assume statement to the abstraction.

The effect on pos:
wp(assume $x \neq 0$, $x > 0$) = ($x \neq 0 => x > 0$) = ($x \geq 0$)
Pred($x \geq 0$) = pos
wp(assume $x \neq 0$, $x \leq 0$) = ($x \neq 0 => x \leq 0$) = ($x \leq 0$)
Pred($x \leq 0$) = $\neg$ pos
So the assume has no effect on the value of pos.

The effect on Rpos:
wp(assume $x \neq 0$, Result > 0) = ($x \neq 0 => Result > 0$)
Pred($x \neq 0 => Result > 0$) = Rpos
wp(assume $x \neq 0$, Result <= 0) = ($x \neq 0 => Result <= 0$)
Pred($x \neq 0 => Result <= 0$) = $\neg$ Rpos
So assume $x \neq 0$ has no effect on the value of Rpos.

Hence assume $x \neq 0$ becomes skip.

After transforming the assume statements, we also abstract the postcondition and get:

always_positive_2 (x: INTEGER): INTEGER
if ? then
   assume pos; pos := True
   Result := x + x
else
   assume $\neg$ pos; pos := False
   if ? then
      assume $\neg$ pos; pos := False
      Result := 1
   else
      skip
      Result := x * x
   end
end
ensure Rpos end

3. Lastly, we transform the assignment statements.
   i. The assignment Result := $x + x$.
The effect on pos:
\[\text{wp(}\text{Result} := \text{x} + \text{x}, \ \text{x} > 0\text{)} = (\text{x} > 0)\]
Pred(\(x > 0\)) = pos
\[\text{wp(}\text{Result} := \text{x} + \text{x}, \ \text{x} \leq 0\text{)} = (\text{x} \leq 0)\]
Pred(\(x \leq 0\)) = \neg pos
So the assignment has no effect on pos.

The effect on Rpos:
\[\text{wp(}\text{Result} := \text{x} + \text{x}, \ \text{Result} > 0\text{)} = (\text{x} + \text{x} > 0)\]
Pred(\(x + x > 0\)) = pos
\[\text{wp(}\text{Result} := \text{x} + \text{x}, \ \text{Result} \leq 0\text{)} = (\text{x} + \text{x} \leq 0)\]
Pred(\(x + x \leq 0\)) = \neg pos
So we have the update:
\[\text{if} \ pos \ \text{then} \quad \text{Rpos := True}\]
\[\text{else if} \ \neg pos \ \text{then} \quad \text{Rpos := False}\]
\[\text{else} \ \text{Rpos := ?}\]
\[\text{end}\]
which can be simplified by a peephole optimization to Rpos := True.

Hence Result := x + x is transformed into Rpos := True.

ii. The assignment Result := 1.

The effect on pos:
\[\text{wp(}\text{Result} := 1, \ \text{x} > 0\text{)} = (1 > 0)\]
Pred(\(1 > 0\)) = True
\[\text{wp(}\text{Result} := 1, \ \text{x} \leq 0\text{)} = (1 \leq 0)\]
Pred(\(1 \leq 0\)) = \neg pos
So Result := 1 has no effect on pos.

The effect on Rpos:
\[\text{wp(}\text{Result} := 1, \ \text{Result} > 0\text{)} = (1 > 0) = True\]
Pred(True) = True
So Result := 1 has the effect Rpos := True.

Hence Result := 1 is transformed into Rpos := True.

iii. The assignment Result := x * x.

The effect on pos:
\[\text{wp(}\text{Result} := \text{x} \ast \text{x}, \ \text{x} > 0\text{)} = (\text{x} > 0)\]
Pred(\(x > 0\)) = pos
\[\text{wp(}\text{Result} := \text{x} \ast \text{x}, \ \text{x} \leq 0\text{)} = (\text{x} \leq 0)\]
Pred(x <= 0) = ¬ pos
So Result := x * x has no effect on pos.

The effect on Rpos:
wp(Result := x * x, Result > 0) = (x * x > 0) = (x /= 0)
Pred(x /= 0) = pos
wp(Result := x * x, Result <= 0) = (x * x <= 0) = (x = 0)
Pred(x = 0) = False
The update:
if pos then
   Rpos := True
else if False then
   Rpos := False
else Rpos := ? end

can be simplified with a peephole optimization to become Rpos := ?.

Hence Result := x * x is transformed into Rpos := ?.

The resulting abstraction looks as follows:

always_positive_3 (x: INTEGER): INTEGER
if ? then
   assume pos; pos := True
   Rpos := True
else
   assume ¬ pos; pos := False
   if ? then
      assume ¬ pos; pos := False
      Rpos := True
   else
      skip
   end
end
ensure Rpos end

(b) No, we cannot verify the abstraction always_positive_3.

Here is a counterexample run:
{¬ pos, ¬ Rpos}
[¬ ?]
{¬ pos, ¬ Rpos}
assume ¬ pos; pos := False
{¬ pos, ¬ Rpos}
[¬ ?]
\{\neg \text{pos}, \neg \text{Rpos}\}
Rpos := ?
\{\neg \text{pos}, \neg \text{Rpos}\}

It corresponds to the following concrete run, for which we computed the weakest precondition with respect to \(x \leq 0 \land \text{Result} \leq 0\):
\[
\{x \leq 0 \land x^2 \leq 0 \land \neg x = 0 \land \neg x > 0\} \quad \text{// Equivalent to False.}
\]
\[
[\neg x > 0]
\{x \leq 0 \land x^2 \leq 0 \land \neg x = 0\} \quad \text{// Equivalent to False.}
\]
\[
[\neg x = 0]
\{x \leq 0 \land x^2 \leq 0\} \quad \text{// Equivalent to } x = 0
\]
\textbf{Result} := x \cdot x
\{x \leq 0 /\text{ Result} \leq 0\}

Since two of the concrete assertions are not satisfiable (equivalent to \textbf{False}), we know that the conjunctions of these assertions will their abstract counterparts will not be satisfiable either. Hence the abstract run is spurious, and calls for abstraction refinement.

Cheat sheet for the translation of assume statements and assignments

Assume statements
A statement \texttt{assume ex end}
gets translated into \texttt{assume }\neg(\texttt{Pred}(\neg\texttt{ex}))\texttt{ end}
plus a parallel assignment such that:
For each p(i),
if \(ex \Rightarrow e(i)\), add \(p(i) := \text{True}\) to the parallel assignment
if \(ex \Rightarrow \neg e(i)\), add \(p(i) := \text{False}\) to the parallel assignment
else omit \(p(i)\) from the parallel assignment.
This parallel assignment is equivalent to the sequence of individual assignments, because all of them assign constants to different variables.

Assignments
If an assignment statement does not modify any of \(e(i)\)'s free variables, then the assignment will leave \(p(i)\) unchanged.