AAA528: Computational Logic

Lecture 9 — Total Correctness Proof

Hakjoo Oh
2025 Spring

Total Correctness

- Total correctness = Partial correctness + Termination
- Total correctness of a function asserts that if the precondition holds on entry, then the function eventually halts and the postcondition holds.

Well-Founded Relations

- Termination proof is based on well-founded relations.
- A binary relation $\prec$ over a set S is well-founded iff there does not exist an infinite sequence $s_1, s_2, \ldots$ of elements of S such that

$$s_1 \succ s_2 \succ \cdots .$$

- For example, the relation $<$ is well-founded over the natural numbers, because any sequence of natural numbers decreasing according to $<$ is finite: e.g.,

$$1023 > 39 > 30 > 29 > 8 > 3 > 0.$$

  However, the relation $<$ is not well-founded over the rationals or reals.

Lexicographic Relations

- A useful class of well-founded relations.
- From a set of pairs of sets and well-founded relations:

$$(S_1, \prec_1), \ldots, (S_m, \prec_m)$$

construct the set

$$S = S_1 \times \cdots \times S_m$$

and define the relation $\prec$:

$$(s_1, \ldots, s_m) \prec (t_1, \ldots, t_m) \iff \bigvee_{i=1}^{m} \left(s_i \prec_i t_i \wedge \bigwedge_{j=1}^{i-1} s_j = t_j \right)$$

- For example, let $S = \mathbb{N}^3$ and $<_3$ be triples of natural numbers and the natural lexicographic extension of $<$ to such triples, respectively:

$$(11, 9, 104) <_3 (11, 13, 3)$$

Proving Termination

- Define a set S with a well-founded relation $\prec$.
  - We usually choose as S the set of n-tuples of natural numbers and as $\prec_n$ the lexicographic extension $<_n$[1] of $<$, where n varies according to the application.
- Find a *ranking function* δ mapping program states to S such that δ decreases according to $\prec$ along every basic path.
- Then, since $\prec$ is well-founded, there cannot exist an infinite sequence of program states.

[1] When $n = 2$, $(a, b) <_2 (a', b') \iff a < a' \vee (a = a' \wedge b < b')$

Example: Linear Search

For each loop, annotate a ranking function:

```
@pre : 0 ≤ l ∧ u < |a|
@post : ⊤
bool LinearSearch (int[] a, int l, int u, int e) {
  int i := l;
  while
  @L : u < |a|
  ↓ (|a| − i)
  (i ≤ u) {
    if (a[i] = e) return true
    i := i + 1;
  }
  return false
}
```

Example: Linear Search

$$
\begin{array}{l}
@L : u < |a| \\
\downarrow (|a| - i) \\
\textbf{assume } i \leq u; \\
\textbf{assume } a[i] \neq e; \\
i := i + 1; \\
\downarrow (|a| - i)
\end{array}
$$

Other basic paths are not relevant to proving termination.

Example: Bubble Sort

For each loop, annotate a ranking function:

```
@pre : ⊤
@post : ⊤
int[] BubbleSort (int[] a_0) {
  int[] a := a_0
  @L_1 : i + 1 ≥ 0
  ↓ (i + 1, i + 1)
  for (int i := |a| − 1; i > 0; i := i − 1) {
    @L_2 : i + 1 ≥ 0 ∧ i − j ≥ 0 ∧ j ≥ 0
    ↓ (i + 1, i − j)
    for (int j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]) {
        int t := a[j];
        int a[j] := a[j + 1];
        int a[j + 1] := t;
      }
    }
  }
  return a;
}
```

Basic Paths

Prove that the ranking functions decrease along each basic paths.

$$
\begin{array}{ll}
(1) & @L_1 : i+1 \geq 0 \\
 & \downarrow (i+1, i+1) \\
 & \textbf{assume } i > 0; \\
 & j := 0; \\
 & \downarrow L_2 : (i+1, i-j) \\
(2) & L_2 : i+1 \geq 0 \wedge i-j \geq 0 \wedge j \geq 0 \\
 & \downarrow (i+1, i-j) \\
 & \textbf{assume } j < i; \\
 & \textbf{assume } a[j] > a[j+1]; \\
 & t := a[j]; \\
 & a[j] := a[j+1]; \\
 & a[j+1] := t; \\
 & j := j+1; \\
 & \downarrow L_2 : (i+1, i-j)
\end{array}
$$

Basic Paths

(3) $L_2 : i + 1 \geq 0 \wedge i - j \geq 0 \wedge j \geq 0$
$\downarrow (i + 1, i - j)$
assume $j < i$;
assume $a[j] \leq a[j + 1]$;
$j := j + 1$;
$\downarrow L_2 : (i + 1, i - j)$

(4) $i + 1 \geq 0 \wedge i - j \geq 0 \wedge j \geq 0$
$\downarrow L_2 : (i + 1, i - j)$
assume $j \geq i$;
$i := i - 1$;
$\downarrow L_1 : (i + 1, i + 1)$

Other basic paths are not relevant to proving termination.

Verification Conditions

The verification condition of basic path

$$
\begin{array}{l}
@\boldsymbol{F} \\
\downarrow \boldsymbol{\delta}[\bar{\boldsymbol{x}}] \\
\boldsymbol{S_1;} \\
\vdots \\
\boldsymbol{S_n;} \\
\downarrow \boldsymbol{\kappa}[\bar{\boldsymbol{x}}]
\end{array}
$$

is

$$\boldsymbol{F} \rightarrow \mathbf{wp}(\boldsymbol{\kappa} \prec \boldsymbol{\delta}[\bar{\boldsymbol{x}}_0], \boldsymbol{S_1}; \ldots; \boldsymbol{S_n})\{\bar{\boldsymbol{x}}_0 \mapsto \bar{\boldsymbol{x}}\}$$

The value of $\boldsymbol{\kappa}$ after executing the statements is less than the value of $\boldsymbol{\delta}$ before executing the statements. The annotation $\boldsymbol{F}$ can provide extra invariant to prove the relation.

Example

To derive the VC for the path

$$
\begin{aligned}
(4)\quad & L_2 : i+1 \geq 0 \wedge i-j \geq 0 \wedge j \geq 0 \\
& \downarrow L_2 : (i+1, i-j) \\
& \textbf{assume } j \geq i; \\
& i := i-1; \\
& \downarrow L_1 : (i+1, i+1)
\end{aligned}
$$

compute

$$
\begin{aligned}
& \textbf{wp}((i+1, i+1) \prec_2 (i_0+1, i_0-j_0), \textbf{assume } j \geq i; i := i-1) \\
\iff & \textbf{wp}(((i_0-1)+1, (i_0-1)+1) <_2 (i_0+1, i_0-j_0), \textbf{assume } j \geq i) \\
\iff & j \geq i \rightarrow (i, i) <_2 (i_0+1, i_0-j_0)
\end{aligned}
$$

Then, replace the variables:

$$j \geq i \rightarrow (i, i) <_2 (i+1, i-j).$$

The VC:

$$i+1 \geq 0 \wedge i-j \geq 0 \wedge j \geq 0 \wedge j \geq i \rightarrow (i, i) <_2 (i+1, i-j).$$

Example: Binary Search

$@\text{pre} : u - l + 1 \geq 0$
$@\text{post} : \top$
$\downarrow u - l + 1$

```
bool BinarySearch (int a[], int l, int u, int e) {
  if (l > u) return false;
  else {
    int m := (l + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) return BinarySearch (a, m + 1, u, e)
    else return BinarySearch (a, l, m − 1, e)
  }
}
```

Basic Paths

$$
\begin{array}{ll}
(1) & \texttt{@pre} : u - l + 1 \geq 0 \\
 & \downarrow u - l + 1 \\
 & \textbf{assume } l \leq u; \\
 & m := (l + u) \textbf{ div } 2; \\
 & \textbf{assume } a[m] \neq e \\
 & \textbf{assume } a[m] < e \\
 & \downarrow u - (m + 1) + 1
\end{array}
$$

VC:

$$u - l + 1 \geq 0 \wedge l \leq u \wedge \cdots \rightarrow u - (((l + u) \textbf{ div } 2) + 1) + 1 < u - l + 1$$

Basic Paths

$$
\begin{array}{ll}
\textbf{(2)} & \texttt{@pre} : u - l + 1 \geq 0 \\
& \downarrow u - l + 1 \\
& \textbf{assume } l \leq u; \\
& m := (l + u) \textbf{ div } 2; \\
& \textbf{assume } a[m] \neq e \\
& \textbf{assume } a[m] \geq e \\
& \downarrow (m - 1) - l + 1
\end{array}
$$

VC:

$$u - l + 1 \geq 0 \wedge l \leq u \wedge \cdots \rightarrow (((l + u) \textbf{ div } 2) - 1) - l + 1 < u - l + 1$$

Summary

Inductive assertion method for proving partial correctness (termination):

1. Derive verification conditions (VCs) from a function.
2. Check the validity of VCs by an SMT solver.
3. If all of VCs are valid, the function is partially correct (terminating)

The method can be automated, if proper loop invariants (ranking functions) are given. Automatically generating loop invariants (ranking functions), however, is not an easy task and remains a research problem.