Theme 2: Correctness Proofs of Imperative Sequential Programs

Lecture 5: Proving Program Termination

Matthieu Sozeau

Inria & Paris Diderot University, Paris 7

October 2014

Proving termination

- How to prove termination of while loops?
- Show that at each iteration, *some quantity is decreasing*.
- This quantity should be a defined as a function of the program state.
- Example: Why does this program terminates?

  f : Nat ;

  ```
  ifact (n : Nat) =
      i : Nat ;
      f := 1 ;
      i := 0 ;
      while i ≠ n do
          i := i + 1 ;
          f := i * f
  ```

- Because $n - i$ decreases at each iteration: $n; n - 1; \cdots ; 0$.

Well founded relations (reminder)

- Let E be a set, and let $\prec \subseteq E \times E$ a binary relation over E.
- The relation $\prec$ is well founded if it has no infinite descending chains, i.e., no sequences of the form

$$e_0 \succ e_1 \succ \cdots \succ e_i \succ \cdots$$

- $(E; \prec)$ is said to be a well founded set (WFS for short).
- Thm: $\prec$ is well founded iff

$$\forall F \subseteq E{:}\ F \neq \emptyset \Rightarrow (\exists e \in F{:}\ \forall e' \in F{:}\ e' \not\prec e)$$

Well founded relations: Examples/Non examples

- $(\mathbb{N}; <)$ is a WFS.
- $(\mathbb{Z}; <)$ is not a WFS.
- $(\mathbb{Z}; \sqsubset)$, where $x \sqsubset y \Leftrightarrow |x| < |y|$, is a WFS.
- $(\mathbb{R}_{\geq 0}; <)$ is not a WFS.
- $(List[?]; <_{lgth})$ is a WFS, where $\text{`}_1 <_{lgth} \text{`}_2 \Leftrightarrow |\text{`}_1| < |\text{`}_2|$.
- $(List[?]; <_{pref})$ is a WFS, where
  $\text{`} <_{pref} \text{`}' \Leftrightarrow \exists \ \in List[?]{:} \ \neq [] \wedge \text{`}' = \text{`}@ \ .$

Product and Lexicographic Well Founded Relations

Let $(E_1; \prec_1); (E_2; \prec_2); :::; (E_n; \prec_n)$ be n WFS's.

- Product Well Founded Relation $\prec_\times \subseteq (E_1 \times \cdots \times E_n)^2$:

  $$(e_1; :::; e_n) \prec_\times (e'_1; :::; e'_n) \iff \forall i \in \{1; :::; n\}: e_i \prec_i e'_i$$

- Lexicographic Well Founded Relation $\prec_\ell \subseteq (E_1 \times \cdots \times E_n)^2$:

  $$(e_1; :::; e_n) \prec_\ell (e'_1; :::; e'_n) \iff$$
  $$\exists i \in \{1; :::; n\}: \; e_i \prec_i e'_i \wedge (\forall j < i: e_j = e'_j)$$

Ranking functions

- Let $X = \{x_1; :::; x_n\}$ be the set of program variables.
- Consider a while loop: `while` C `do` S.
- Let   be an invariant of the loop, i.e.,

  $$\forall \; ; \; ': (\; \models \; \wedge C \text{ and } \; \xrightarrow{S} \; ') \Rightarrow \; ' \models$$

- Ranking function of the loop: $: D^n \to E$ such that

  $$\forall \; ; \; ': (\; \models \; \wedge C \text{ and } \; \xrightarrow{S} \; ') \Rightarrow \; (\;) \prec \; (\; ')$$

  where $(E; \prec)$ is a well founded set.

- Termination:

  The while loop terminates if S *is a terminating statement, and the loop has a ranking function.*

Hoare logic: Proving total correctness

- Formulas of the form of the form:

  $$\{\!| \; |\!\} \, S \, \{\!| \; |\!\}$$

- Formal Semantics:

  $$\{\!| \; |\!\} \, S \, \{\!| \; |\!\} \text{ iff } \forall \; : (\; \models \; \Rightarrow \exists \; ': (\; \xrightarrow{S} \; ' \wedge \; ' \models \;))$$

- Intuitive meaning:

  Starting from any state satisfying , the execution of S *terminates and leads to a state satisfying .*

Rules for total correctness

- Same rules as for partial correctness, except the case of while loops.
- Total Iteration Rule:

  $$\frac{: D^n \to E \qquad (E; \prec) \text{ is a WFS} \qquad \{\!| \; \wedge C \wedge \; = e |\!\} \, S \, \{\!| \; \wedge \; \prec e |\!\}}{\{\!| \; |\!\} \; \texttt{while} \; C \; \texttt{do} \; S \; \{\!| \; \wedge \neg C |\!\}}$$

When does it fail

```
i; n; e :  Nat ;
while i ≠ n  do
       skip ;
```

Prove:

$$\{\!|\ i \neq n \wedge n - i = e\ |\!\}$$
$$\text{skip}$$
$$\{\!|\ n - i < e\ |\!\}$$

When does it fail

```
n; i; e :  Nat ;
i := 0 ;
while i ≠ n  do
       i := i + 1
```

Prove:

$$\{\!|\ i \neq n \wedge n - i = e\ |\!\}$$
$$i := i + 1$$
$$\{\!|\ n - i < e\ |\!\}$$

By assignment:

$$\{\!|\ n - (i + 1) < e\ |\!\}$$
$$i := i + 1\ ;$$
$$\{\!|\ n - i < e\ |\!\}$$

Does $i \neq n \wedge n - i = e \implies n - (i + 1) < e$? No, need $i < n$

Termination proof: Example

```
f : Nat ;
ifact (n : Nat)  =
    i : Nat ; e : Nat ;
    f := 1 ;
    i := 0 ;
    while i ≠ n  do
           i := i + 1 ;
           f := i * f
```

- Prove:

$$\{\!|\quad \wedge\ i \neq n \wedge n - i = e\ |\!\}$$
$$i := i + 1; f := i * f$$
$$\{\!|\quad \wedge\ n - i < e\ |\!\}$$

  for some supporting invariant   .
-    = true ?
- We must use the fact that $i \leq n$.

Termination proof: Example (cont.)

- Prove:

$$\{\!|\ i \leq n \wedge i \neq n \wedge n - i = e\ |\!\}$$
$$i := i + 1; f := i * f$$
$$\{\!|\ i \leq n \wedge n - i < e\ |\!\}$$

- Deduce:

$$\{\!|\ i \leq n\ |\!\}$$
$$\texttt{while}\ i \neq n\ \text{do}\ \{i := i + 1; f := i * f\}$$
$$\{\!|\ i \leq n \wedge i = n\ |\!\}$$

Termination proof: Example (cont.)

- Assignment + Sequential composition rules:

$$\{\!|\, i+1 \leq n \wedge n-i-1 < e \,|\!\}$$
$$i := i+1\,;$$
$$\{\!|\, i \leq n \wedge n-i < e \,|\!\}$$
$$f := i * f$$
$$\{\!|\, i \leq n \wedge n-i < e \,|\!\}$$

- $(i \leq n \wedge i \neq n) \Rightarrow i+1 \leq n$
- $i < n \wedge n-i = e \Rightarrow 0 < e$
- $n-i = e \wedge 0 < e \Rightarrow n-i-1 < e$
- Implication rule:

$$\{\!|\, i \leq n \wedge i \neq n \wedge n-i = e \,|\!\}$$
$$i := i+1; f := i * f$$
$$\{\!|\, i \leq n \wedge n-i < e \,|\!\}$$

Total correctness proof

```
f : Nat ;
ifact (n : Nat) =
    i : Nat ; e : Nat ;
    f := 1 ;
    i := 0 ;
    while i ≠ n do
        i := i + 1 ;
        f := i * f
```

- Prove:

$$\{\!|\, f = fact(i) \wedge 0 \leq i \leq n \wedge i \neq n \wedge n-i = e \,|\!\}$$
$$i := i+1; f := i * f$$
$$\{\!|\, f = fact(i) \wedge 0 \leq i \leq n \wedge n-i < e \,|\!\}$$

- Deduce:

$$\{\!|\, f = fact(i) \wedge 0 \leq i \leq n \,|\!\}$$
$$\texttt{while}\ (i \neq n)\ \texttt{do}\ \{i := i+1; f := i * f\}$$
$$\{\!|\, f = fact(i) \wedge 0 \leq i \leq n \wedge i = n \,|\!\}$$

A more complex example

```
x; y : Nat ;
while x > 0 do
    if even(y) then
            x := x − 1 ;
            y := y + 3
    else
            y := y − 1
```

$$(x=4; y=4) \xrightarrow{x:=x-1; y:=y+3} (x=3; y=7) \xrightarrow{y:=y-1} (x=3; y=6)$$
$$(x=3; y=6) \xrightarrow{x:=x-1; y:=y+3} (x=2; y=9) \xrightarrow{y:=y-1} (x=2; y=8)$$
$$(x=2; y=8) \xrightarrow{x:=x-1; y:=y+3} (x=1; y=11) \xrightarrow{y:=y-1} (x=1; y=10)$$
$$(x=1; y=10) \xrightarrow{x:=x-1; y:=y+3} (x=0; y=13) \xrightarrow{y:=y-1} (x=0; y=12)$$

- We need to consider the lexicographic order over pairs of integers.
- Well founded set: $(Nat \times Nat; <_\ell)$
- Ranking function: $(x; y) = (x; y)$

Summary

- Total correctness = Partial correctness + Termination.
- Partial correctness ensures that the programs provides the expected results if it terminates.
- Proving termination needs reasoning about "well-foundedness" of computations.
- It amounts in finding ranking functions for while loops mapping states to elements of well-founded sets.
- Various ious ious

Proving terminationabgeneral.t: Haltovins