Theme 2: Proving Correct Imperative Sequential Programs

Lecture 6: Proving Program Termination

Ahmed Bouajjani

Paris Diderot University, Paris 7

January 2014

Proving termination

- How to prove termination of while loops?

- Example: Why this program terminates?

$f : Nat$;

$$
\begin{aligned}
&\mathrm{ifact}\,(\mathrm{n} : \mathrm{Nat}) \;=\\
&\quad \mathrm{i} : \mathrm{Nat}\ ;\\
&\quad \mathrm{f} := 1\ ;\\
&\quad \mathrm{i} := 0\ ;\\
&\quad \mathrm{while}\ \mathrm{i} \neq \mathrm{n}\ \ \mathrm{do}\\
&\qquad\quad \mathrm{i} := \mathrm{i} + 1\ ;\\
&\qquad\quad \mathrm{f} := \mathrm{i} * \mathrm{f}
\end{aligned}
$$

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 this program terminates?

```
f : Nat ;

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

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 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.

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.

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.

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.
- $(\mathbb{R}_{\geq 0}, <_\epsilon)$ is a WFS, where $\epsilon > 0$, and $x <_\epsilon y \Leftrightarrow x \leq y - \epsilon$.

Well founded relations: Examples/Non examples

- $(\mathbb{N}, <)$ is a WFS.
- $(\mathbb{Z}, <)$ is not a WFS.
- $(\mathbb{Z}, \sqsubset)$, where $\mathsf{x} \sqsubset \mathsf{y} \Leftrightarrow |\mathsf{x}| < |\mathsf{y}|$, is a WFS.
- $(\mathbb{R}_{\geq 0}, <)$ is not a WFS.
- $(\mathbb{R}_{\geq 0}, <_\epsilon)$ is a WFS, where $\epsilon > 0$, and $\mathsf{x} <_\epsilon \mathsf{y} \Leftrightarrow \mathsf{x} \leq \mathsf{y} - \epsilon$.
- $(\mathsf{List}[\star], <_{\mathsf{lgth}})$ is a WFS, where $\ell_1 <_{\mathsf{lgth}} \ell_2 \Leftrightarrow |\ell_1| < |\ell_2|$.
- $(\mathsf{List}[\star], <_{\mathsf{pref}})$ is a WFS, where $\ell <_{\mathsf{pref}} \ell' \Leftrightarrow \exists \sigma \in \mathsf{List}[\star].\ \ell' = \ell @ \sigma$.

Product and Lexicographic Well Founded Relations

Let $(\mathsf{E}_1, \prec_1), (\mathsf{E}_2, \prec_2), \ldots, (\mathsf{E}_n, \prec_n)$ be n WFS's.

Product and Lexicographic Well Founded Relations

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

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

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

Product and Lexicographic Well Founded Relations

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

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

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

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

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

Ranking functions

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

$$\forall \mu, \mu'. \ (\mu \models \phi \wedge C \text{ and } \mu \xrightarrow{S} \mu') \Rightarrow \mu' \models \phi$$

Ranking functions

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

$$\forall \mu, \mu'.\ (\mu \models \phi \wedge C \text{ and } \mu \xrightarrow{S} \mu') \Rightarrow \mu' \models \phi$$

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

$$\forall \mu, \mu'.\ (\mu \models \phi \wedge C \text{ and } \mu \xrightarrow{S} \mu') \Rightarrow \rho(\mu) \prec \rho(\mu')$$

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

Ranking functions

- Let $\mathsf{X} = \{\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}\}$ be the set of program variables.
- Consider a while loop: `while` C `do` S.
- Let ϕ be an invariant of the loop, i.e.,

$$\forall \mu, \mu'. \, (\mu \models \phi \wedge \mathsf{C} \text{ and } \mu \xrightarrow{\mathsf{S}} \mu') \Rightarrow \mu' \models \phi$$

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

$$\forall \mu, \mu'. \, (\mu \models \phi \wedge \mathsf{C} \text{ and } \mu \xrightarrow{\mathsf{S}} \mu') \Rightarrow \rho(\mu) \prec \rho(\mu')$$

  where $(\mathsf{E}, \prec)$ 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:

$$\{\!|\phi|\!\}\ \mathsf{S}\ \{\!|\psi|\!\}$$

Hoare logic: Proving total correctness

- Formulas of the form of the form:

$$\{\!|\phi|\!\}\ \mathsf{S}\ \{\!|\psi|\!\}$$

- Formal Semantics:

$$\{\!|\phi|\!\}\ \mathsf{S}\ \{\!|\psi|\!\} \ \text{ iff } \ \forall \mu.\ (\mu \models \phi \Rightarrow \exists \mu'.\ (\mu \xrightarrow{\mathsf{S}} \mu' \wedge \mu' \models \psi))$$

- Intuitive meaning:

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

Rules for total correctness

- Same rules as for partial correctness, except the case of while loops.

Rules for total correctness

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

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

Termination proof: Example

$\mathsf{f} : \mathsf{Nat}\ ;$
$\mathsf{ifact}\,(\mathsf{n} : \mathsf{Nat})\ =$
$\quad \mathsf{i} : \mathsf{Nat}\ ;$
$\quad \mathsf{f} := 1\ ;$
$\quad \mathsf{i} := 0\ ;$
$\quad$ `while` $\mathsf{i} \neq \mathsf{n}$ `do`
$\qquad \mathsf{i} := \mathsf{i} + 1\ ;$
$\qquad \mathsf{f} := \mathsf{i} * \mathsf{f}$

Termination proof: Example

$$
\begin{array}{l}
\mathsf{f} : \mathsf{Nat}\ ; \\
\mathsf{ifact}\,(\mathsf{n} : \mathsf{Nat}) \ = \\
\quad \mathsf{i} : \mathsf{Nat}\ ;\ \mathsf{e} : \mathsf{Nat}\ ; \\
\quad \mathsf{f} := 1\ ; \\
\quad \mathsf{i} := 0\ ; \\
\quad \texttt{while}\ \mathsf{i} \neq \mathsf{n}\ \texttt{do} \\
\qquad\quad \mathsf{i} := \mathsf{i} + 1\ ; \\
\qquad\quad \mathsf{f} := \mathsf{i} * \mathsf{f}
\end{array}
$$

- Prove:

$$
\begin{array}{c}
\{\!|\ \phi\ \wedge \mathsf{i} \neq \mathsf{n} \wedge \mathsf{n} - \mathsf{i} = \mathsf{e}\ |\!\} \\
\mathsf{i} := \mathsf{i} + 1; \mathsf{f} := \mathsf{i} * \mathsf{f} \\
\{\!|\ \phi \wedge \mathsf{n} - \mathsf{i} < \mathsf{e}\ |\!\}
\end{array}
$$

for some supporting invariant ϕ.

Termination proof: Example

$\mathsf{f} : \mathsf{Nat}\ ;$
$\mathsf{ifact}\,(\mathsf{n} : \mathsf{Nat}) \ =$
$\quad \mathsf{i} : \mathsf{Nat}\ ;\ \mathsf{e} : \mathsf{Nat}\ ;$
$\quad \mathsf{f} := 1\ ;$
$\quad \mathsf{i} := 0\ ;$
$\quad \texttt{while}\ \mathsf{i} \neq \mathsf{n}\ \texttt{do}$
$\qquad \mathsf{i} := \mathsf{i} + 1\ ;$
$\qquad \mathsf{f} := \mathsf{i} * \mathsf{f}$

- Prove:

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

  for some supporting invariant ϕ.

- $\phi = \mathsf{true}$?

Termination proof: Example

$\mathsf{f} : \mathsf{Nat}\ ;$
$\mathsf{ifact}\,(\mathsf{n} : \mathsf{Nat}) \ =$
$\quad \mathsf{i} : \mathsf{Nat}\ ;\ \mathsf{e} : \mathsf{Nat}\ ;$
$\quad \mathsf{f} := 1\ ;$
$\quad \mathsf{i} := 0\ ;$
$\quad \texttt{while}\ \mathsf{i} \neq \mathsf{n}\ \texttt{do}$
$\qquad \mathsf{i} := \mathsf{i} + 1\ ;$
$\qquad \mathsf{f} := \mathsf{i} * \mathsf{f}$

- Prove:

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

  for some supporting invariant ϕ.

- $\phi = \mathsf{true}$?
- We must use the fact that $\mathsf{i} \leq \mathsf{n}$.

Termination proof: Example (cont.)

- Prove:

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

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:

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

Termination proof: Example (cont.)

- Assignment + Sequential composition rules:

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

Termination proof: Example (cont.)

- Assignment + Sequential composition rules:

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

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

Termination proof: Example (cont.)

- Assignment + Sequential composition rules:

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

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

$$
\begin{array}{l}
\mathsf{f} : \mathsf{Nat}\ ; \\
\mathsf{ifact}\,(\mathsf{n} : \mathsf{Nat}) \ = \\
\quad \mathsf{i} : \mathsf{Nat}\ ; \\
\quad \mathsf{f} := 1\ ; \\
\quad \mathsf{i} := 0\ ; \\
\quad \mathsf{while}\ \mathsf{i} \neq \mathsf{n}\ \ \mathsf{do} \\
\qquad\quad \mathsf{i} := \mathsf{i} + 1\ ; \\
\qquad\quad \mathsf{f} := \mathsf{i} * \mathsf{f}
\end{array}
$$

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:

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

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
```

A more complex example

$$
\begin{array}{l}
x, y : \ Nat \ ; \\
\texttt{while } x > 0 \texttt{ do} \\
\qquad \texttt{if } \ even(y) \ \texttt{then} \\
\qquad\qquad\qquad x := x - 1 \ ; \\
\qquad\qquad\qquad y := y + 3 \\
\qquad \texttt{else} \\
\qquad\qquad\qquad y := y - 1
\end{array}
$$

$$(x = 4, y = 4) \xrightarrow{x:=x-1;y:=y+3} (x = 3, y = 7) \xrightarrow{y:=y-1} (x = 3, y = 6)$$

A more complex example

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

$$(x = 4, y = 4) \xrightarrow{x:=x-1;y:=y+3} (x = 3, y = 7) \xrightarrow{y:=y-1} (x = 3, y = 6)$$
$$(x = 3, y = 6) \xrightarrow{x:=x-1;y:=y+3} (x = 2, y = 9) \xrightarrow{y:=y-1} (x = 2, y = 8)$$

A more complex example

$x, y : \ Nat \ ;$
$\texttt{while}\ x > 0\ \texttt{do}$
$\quad \texttt{if}\ \ even(y)\ \ \texttt{then}$
$\quad\quad x := x - 1\ ;$
$\quad\quad y := y + 3$
$\quad \texttt{else}$
$\quad\quad y := y - 1$

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

A more complex example

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

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

A more complex example

$x, y : \ Nat$;
$\texttt{while}\ x > 0\ \texttt{do}$
$\quad \texttt{if}\ \ even(y)\ \texttt{then}$
$\qquad x := x - 1$;
$\qquad y := y + 3$
$\quad \texttt{else}$
$\qquad y := y - 1$

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

- We need to consider the lexicographic order over pairs of integers.

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: $\rho(x, y) = (x, y)$

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 well-founded sets can be considered, in particular, lexicographic well-founded relations are needed in some cases.
- Proving termination cannot be automatized in general. (Halting problem of Turing machines.) But there are (uncomplete) techniques that can be used in some cases.