Proving termination

- How to prove termination of while loops?
- Show that at each iteration, *some quantity is decreasing*.
- This quantity should be defined as a function of the program state.
- Example: Why does this program terminates?
  
  $f : \text{Nat} ;$
  
  $\text{ifact (n : Nat)} =$
  
  $i : \text{Nat} ;$
  $f := 1 ;$
  $i := 0 ;$
  while $i \neq n$ 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
  
  $\forall F \subseteq E. F \neq \emptyset \Rightarrow (\exists e \in F. \forall e' \in F. e' \neq e)$

Well founded relations: Examples/Non examples

- $(\mathbb{N}, <)$ is a WFS.
- $(\mathbb{Z}, <)$ is not a WFS.
- $(\mathbb{Z}, \sqsubseteq)$, where $x \sqsubseteq y \Leftrightarrow |x| < |y|$, is a WFS.
- $(\mathbb{R}_{\geq 0}, <)$ is not a WFS.
- $(\text{List}[\ast], <_{\text{lgth}})$ is a WFS, where $\ell_1 <_{\text{lgth}} \ell_2 \Leftrightarrow |\ell_1| < |\ell_2|.$
- $(\text{List}[\ast], <_{\text{pref}})$ is a WFS, where
  
  $\ell <_{\text{pref}} \ell' \Leftrightarrow \exists \sigma \in \text{List}[\ast]. \sigma \neq [] \land \ell' = \ell @ \sigma.$
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 \(\prec \subseteq (E_1 \times \cdots \times E_n)^2\):
  \[ (e_1, \ldots, e_n) \prec (e_1', \ldots, e_n') \iff \forall i \in \{1, \ldots, n\}. \ e_i \prec_i e_i' \]

- Lexicographic Well Founded Relation \(\prec^\ell \subseteq (E_1 \times \cdots \times E_n)^2\):
  \[ (e_1, \ldots, e_n) \prec^\ell (e_1', \ldots, e_n') \iff \exists i \in \{1, \ldots, n\}. (e_i \prec_i e_i' \land (\forall j < i. \ e_j = e_j')) \]

Hoare logic: Proving total correctness

- Formulas of the form of the form:
  \(\{\phi\} S \{\psi\}\)

- Formal Semantics:
  \(\{\phi\} S \{\psi\}\) iff \(\forall \mu. (\mu \models \phi \Rightarrow \exists \mu'. (\mu \xrightarrow{S} \mu' \land \mu' \models \psi))\)

- Intuitive meaning:
  Starting from any state satisfying \(\phi\), the execution of \(S\) terminates and leads to a state satisfying \(\psi\).

Rules for total correctness

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

- Total Iteration Rule:
  \(\rho : D^n \to E (E, \prec)\) is a WFS
  \(\{\phi \land C \land \rho = e\} S \{\phi \land \rho < e\}\)
  \(\{\phi\} \text{ while } C \text{ do } S \{\phi \land \neg C\}\)
When does it fail

\begin{verbatim}
i, n, e : Nat;
while i ≠ n do
  skip;
\end{verbatim}

Prove:

\begin{verbatim}
{ i ≠ n \land n - i = e }
skip
{n - i < e}
\end{verbatim}

Termination proof: Example

\begin{verbatim}
f : Nat;
ifact (n : Nat) =
i : Nat ; e : Nat ;
f := 1 ;
i := 0 ;
while i ≠ n do
  i := i + 1 ;
f := i * f
\end{verbatim}

Prove:

\begin{verbatim}
{ i ≠ n \land n - i = e }
i := i + 1 ;
\end{verbatim}

By assignment:

\begin{verbatim}
{n - (i + 1) < e}
i := i + 1 ;
{n - i < e}
\end{verbatim}

Does i ≠ n \land n - i = e \implies n - (i + 1) < e ? No, need i < n
Assignment + Sequential composition rules:
\[
\begin{array}{l}
\{ \langle i + 1 \leq n \land n - i - 1 < e \rangle \} \\
i := i + 1 \\
\{ \langle i \leq n \land n - i < e \rangle \} \\
f := i \ast f \\
\{ \langle i \leq n \land n - i < e \rangle \}
\end{array}
\]

\[
(i \leq n \land i \neq n) \Rightarrow i + 1 \leq n
\]

\[
i < n \land n - i = e \Rightarrow 0 < e
\]

\[
n - i = e \land 0 < e \Rightarrow n - i - 1 < e
\]

Implication rule:
\[
\begin{array}{l}
\{ \langle i \leq n \land i \neq n \land n - i = e \rangle \} \\
i := i + 1; f := i \ast f \\
\{ \langle i \leq n \land n - i < e \rangle \}
\end{array}
\]

Prove:
\[
\{ \langle f = \text{fact}(i) \land 0 \leq i \leq n \land i \neq n \land n - i = e \rangle \} \\
i := i + 1; f := i \ast f \\
\{ \langle f = \text{fact}(i) \land 0 \leq i \leq n \land n - i < e \rangle \}
\]

Deduce:
\[
\{ \langle f = \text{fact}(i) \land 0 \leq i \leq n \rangle \} \\
\text{while } (i \neq n) \text{ do } \{ i := i + 1; f := i \ast f \} \\
\{ f = \text{fact}(i) \land 0 \leq i \leq n \land i = n \}
\]

A more complex example
\[
x, y : \text{Nat};
\]
\[
\text{while } x > 0 \text{ do }
\]
\[
\text{if even}(y) \text{ then } \\
x := x - 1; \\
y := y + 3
\]
\[
\text{else } \\
y := y - 1
\]
\[
(x = 4, y = 4) \xrightarrow{x-x-1\ y-y+3} (x = 3, y = 7) \xrightarrow{y-y-1} (x = 3, y = 6) \\
(x = 3, y = 6) \xrightarrow{x-x-1\ y-y+3} (x = 2, y = 9) \xrightarrow{y-y-1} (x = 2, y = 8) \\
(x = 2, y = 8) \xrightarrow{x-x-1\ y-y+3} (x = 1, y = 11) \xrightarrow{y-y-1} (x = 1, y = 10) \\
(x = 1, y = 10) \xrightarrow{x-x-1\ y-y+3} (x = 0, y = 13) \xrightarrow{y-y-1} (x = 0, y = 12)
\]

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

Well founded set: \( (\text{Nat} \times \text{Nat}, <_\ell) \)

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.