Proving Termination of Recursive Programs or How to Avoid Proving Termination

Nikolaj Popov and Tudor Jebelean

Research Institute for Symbolic Computation, Linz

popov@risc.uni-linz.ac.at

Outline

Preconditions and Postconditions. Total Correctness

Given the triple

$\{I\}F\{O\}$ (Input condition, Function definition, Output condition)

Total Correctness Formula

$(\forall n : I[n])\ (F[n]\downarrow \wedge O[n, F[n]])$

Example

$\{x \in \mathbb{R} \wedge n \in \mathbb{N}\}$

$pow[x, n] =$ **If** $n = 0$ **then** 1 **else** $x * pow[x, n-1]$

$\{x^n = pow[x, n]\}$

$(\forall x : \mathbb{R})(\forall n : \mathbb{N})\ (pow[x, n]\downarrow \wedge\ x^n = pow[x, n])$

Preconditions and Postconditions. Total Correctness

Given the triple

$\{I\}F\{O\}$ (Input condition, Function definition, Output condition)

Total Correctness Formula

$(\forall n : I[n])\ (F[n]\downarrow \wedge O[n, F[n]])$

Example

$\{x \in \mathbb{R} \wedge n \in \mathbb{N}\}$

$pow[x, n] =$ **If** $n = 0$ **then** 1 **else** $x * pow[x, n-1]$

$\{x^n = pow[x, n]\}$

$(\forall x : \mathbb{R})(\forall n : \mathbb{N})\ (pow[x, n]\downarrow \wedge\ x^n = pow[x, n])$

Preconditions and Postconditions. Total Correctness

Given the triple

$\{I\}F\{O\}$ (Input condition, Function definition, Output condition)

Total Correctness Formula

$(\forall n : I[n])\ (F[n] \downarrow \wedge O[n, F[n]])$

Example

$\{x \in \mathbb{R} \wedge n \in \mathbb{N}\}$

$pow[x, n] =$ **If** $n = 0$ **then** 1 **else** $x * pow[x, n-1]$

$\{x^n = pow[x, n]\}$

$(\forall x : \mathbb{R})(\forall n : \mathbb{N})\ (pow[x, n] \downarrow \wedge\ x^n = pow[x, n])$

Building up Correct Programs

Basic Functions e.g. +, -, *, etc.

New Functions in Terms of Already Known Functions

- Input and output predicates;
- Prove total correctness;

Modularity. After proving correctness, use only the specification.

$\{x \in \mathbb{R} \wedge n \in \mathbb{N}\}$ *Input condition*

$pow[x, n] = \ldots$

$\{x^n = pow[x, n]\}$ *Output condition*

Building up Correct Programs

Basic Functions e.g. +, -, *, etc.

New Functions in Terms of Already Known Functions

- Input and output predicates;
- Prove total correctness;

Modularity. After proving correctness, use only the specification.

$\{x \in \mathbb{R} \wedge n \in \mathbb{N}\}$ *Input condition*

$pow[x, n] = \ldots$

$\{x^n = pow[x, n]\}$ *Output condition*

Building up Correct Programs

Basic Functions e.g. +, -, *, etc.

New Functions in Terms of Already Known Functions

- Input and output predicates;
- Prove total correctness;

Modularity. After proving correctness, use only the specification.

$\{x \in \mathbb{R} \wedge n \in \mathbb{N}\}$ *Input condition*

$pow[x, n] = \ldots$

$\{x^n = pow[x, n]\}$ *Output condition*

Coherence

Appropriate values for the auxiliary functions

No input condition of an auxiliary function will be violated

Example

$F[x] =$ **If** $Q[x]$ **then** $H[x]$ **else** $G[x]$

- $(\forall x : I_F[x])\ (Q[x] \Rightarrow I_H[x])$
- $(\forall x : I_F[x])\ (\neg Q[x] \Rightarrow I_G[x])$

We call these programs *Coherent*

Coherence

Appropriate values for the auxiliary functions

No input condition of an auxiliary function will be violated

Example

$F[x] =$ **If** $Q[x]$ **then** $H[x]$ **else** $G[x]$

- $(\forall x: \; I_F[x]) \; (Q[x] \Rightarrow I_H[x])$
- $(\forall x: \; I_F[x]) \; (\neg Q[x] \Rightarrow I_G[x])$

We call these programs *Coherent*

Coherence

Appropriate values for the auxiliary functions

No input condition of an auxiliary function will be violated

Example

$F[x] =$ **If** $Q[x]$ **then** $H[x]$ **else** $G[x]$

- $(\forall x : I_F[x]) \ (Q[x] \Rightarrow I_H[x])$
- $(\forall x : I_F[x]) \ (\neg Q[x] \Rightarrow I_G[x])$

We call these programs *Coherent*

Coherent Recursive Programs

Double (Multiple) Recursion Program Scheme

$F[x] =$ **If** $Q[x]$ **then** $S[x]$ **else** $C[x, F[R_1[x]], F[R_2[x]]]$

Conditions for coherence

- $(\forall x : I_F[x])\ (Q[x] \Rightarrow I_S[x])$

- $(\forall x : I_F[x])\ (\neg Q[x] \Rightarrow I_F[R_1[x]])$
- $(\forall x : I_F[x])\ (\neg Q[x] \Rightarrow I_F[R_2[x]])$

- $(\forall x : I_F[x])\ (\neg Q[x] \Rightarrow I_{R_1}[x])$
- $(\forall x : I_F[x])\ (\neg Q[x] \Rightarrow I_{R_2}[x])$

- $(\forall x, y, z : I_F[x])\ (\neg Q[x] \wedge O_F[R_1[x], y] \wedge O_F[R_2[x], z] \Rightarrow I_C[x, y, z])$

Coherent Recursive Programs

Double (Multiple) Recursion Program Scheme

$F[x] = $ **If** $Q[x]$ **then** $S[x]$ **else** $C[x, F[R_1[x]], F[R_2[x]]]$

Conditions for coherence

- $(\forall x : I_F[x])\ (Q[x] \Rightarrow I_S[x])$

- $(\forall x : I_F[x])\ (\neg Q[x] \Rightarrow I_F[R_1[x]])$
- $(\forall x : I_F[x])\ (\neg Q[x] \Rightarrow I_F[R_2[x]])$

- $(\forall x : I_F[x])\ (\neg Q[x] \Rightarrow I_{R_1}[x])$
- $(\forall x : I_F[x])\ (\neg Q[x] \Rightarrow I_{R_2}[x])$

- $(\forall x, y, z : I_F[x])\ (\neg Q[x] \wedge O_F[R_1[x], y] \wedge O_F[R_2[x], z] \Rightarrow I_C[x, y, z])$

Coherent Recursive Programs

Double (Multiple) Recursion Program Scheme

$F[x] =$ **If** $Q[x]$ **then** $S[x]$ **else** $C[x, F[R_1[x]], F[R_2[x]]]$

Conditions for Partial Correctness

- $(\forall x : \ I_F[x]) \ (Q[x] \Rightarrow O_F[x, S[x]])$
- $(\forall x, y, z : \ I_F[x]) \ (\neg Q[x] \wedge O_F[R_1[x], y] \wedge O_F[R_2[x], z] \Rightarrow O_F[x, C[x, y, z]])$

Coherent Recursive Programs

Double (Multiple) Recursion Program Scheme

$F[x] =$ **If** $Q[x]$ **then** $S[x]$ **else** $C[x, F[R_1[x]], F[R_2[x]]]$

Condition for Termination

- $(\forall x : \ I_F[x]) \ (F'[x] = \mathbf{T})$
- where:

$$F'[x] = \textbf{ If } Q[x] \textbf{ then T else } F'[R_1[x]] \wedge F'[R_2[x]]$$

Coherent Recursive Programs

Double (Multiple) Recursion Program Scheme

$F[x] = $ **If** $Q[x]$ **then** $S[x]$ **else** $C[x, F[R_1[x]], F[R_2[x]]]$

Condition for Termination

- $(\forall x: \ I_F[x]) \ (F'[x] = \mathbf{T})$
- where:

$$F'[x] = \textbf{ If } Q[x] \textbf{ then T else } F'[R_1[x]] \wedge F'[R_2[x]]$$

Example Factorial

Fact $(\forall n : \mathbb{N})\ (Fact[n] = n!)$

$$Fact[n] = \textbf{If } n = 0 \textbf{ then } 1 \quad \textbf{else } n * Fact[n-1].$$

is coherent if

- $(\forall n : \mathbb{N})\ (n = 0 \ \Rightarrow \mathbf{T})$
- $(\forall n : \mathbb{N})\ (n \neq 0 \ \Rightarrow n - 1 \in \mathbb{N})$
- $(\forall n : \mathbb{N})\ (n \neq 0 \ \Rightarrow \mathbf{T})$

Example Factorial

Fact $(\forall n : \mathbb{N})\ (Fact[n] = n!)$

$$Fact[n] = \quad \textbf{If } n = 0 \textbf{ then } 1 \quad \textbf{else } n * Fact[n-1].$$

is coherent if

- $(\forall n : \mathbb{N})\ (n = 0 \ \Rightarrow \mathbf{T})$
- $(\forall n : \mathbb{N})\ (n \neq 0 \ \Rightarrow n - 1 \in \mathbb{N})$
- $(\forall n : \mathbb{N})\ (n \neq 0 \ \Rightarrow \mathbf{T})$

Example Factorial

Fact $(\forall n : \mathbb{N})\ (Fact[n] = n!)$

$$Fact[n] = \textbf{If } n = 0 \textbf{ then } 1 \quad \textbf{else } n * Fact[n-1].$$

is partially correct if

- $(\forall n : \mathbb{N})\ (n = 0 \Rightarrow 1 = n!)$
- $(\forall n, m : \mathbb{N})\ (n \neq 0 \wedge m = (n-1)! \Rightarrow n * m = n!)$

Example Factorial

Fact $(\forall n : \mathbb{N})\ (Fact[n] = n!)$

$$Fact[n] = \begin{array}{l}\textbf{If } n = 0 \textbf{ then } 1\\ \textbf{else } n * Fact[n-1].\end{array}$$

is partially correct if

- $(\forall n : \mathbb{N})\ (n = 0 \ \Rightarrow 1 = n!)$
- $(\forall n, m : \mathbb{N})\ (n \neq 0 \wedge m = (n-1)! \ \Rightarrow n * m = n!)$

Example Factorial

Fact $(\forall n : \mathbb{N})\ (Fact[n] = n!)$

$$Fact[n] = \begin{array}{l}\textbf{If } n = 0 \textbf{ then } 1\\ \textbf{else } n * Fact[n-1].\end{array}$$

terminates if

- $(\forall n : \mathbb{N})\ (Fact'[n] = \mathbf{T})$
- where:

$$Fact'[n] = \begin{array}{l}\textbf{If } n = 0 \textbf{ then } \mathbf{T}\\ \textbf{else } Fact'[n-1].\end{array}$$

Example Factorial

Fact $(\forall n : \mathbb{N})\ (Fact[n] = n!)$

$$Fact[n] = \quad \textbf{If } n = 0 \textbf{ then } 1$$
$$\textbf{else } n * Fact[n-1].$$

terminates if

- $(\forall n : \mathbb{N})\ (Fact'[n] = \mathbf{T})$
- where:

$$Fact'[n] = \quad \textbf{If } n = 0 \textbf{ then } \mathbf{T}$$
$$\textbf{else } Fact'[n-1].$$

Example Sum

Sum $(\forall n : \mathbb{N})\ (Sum[n] = \frac{n(n+1)}{2})$

$$Sum[n] = \textbf{If } n = 0 \textbf{ then } 0 \quad \textbf{else } n + Sum[n-1].$$

is coherent if

- $(\forall n : \mathbb{N})\ (n = 0 \Rightarrow \mathbf{T})$
- $(\forall n : \mathbb{N})\ (n \neq 0 \Rightarrow n - 1 \in \mathbb{N})$
- $(\forall n : \mathbb{N})\ (n \neq 0 \Rightarrow \mathbf{T})$

Example Sum

Sum $(\forall n : \mathbb{N})\ (Sum[n] = \frac{n(n+1)}{2})$

$$Sum[n] = \quad \textbf{If } n = 0 \textbf{ then } 0 \quad \textbf{else } n + Sum[n-1].$$

is coherent if

- $(\forall n : \mathbb{N})\ (n = 0 \ \Rightarrow \mathbf{T})$
- $(\forall n : \mathbb{N})\ (n \neq 0 \ \Rightarrow n - 1 \in \mathbb{N})$
- $(\forall n : \mathbb{N})\ (n \neq 0 \ \Rightarrow \mathbf{T})$

Example Sum

Sum $(\forall n : \mathbb{N})\ (Sum[n] = \frac{n(n+1)}{2})$

$$Sum[n] = \textbf{If } n = 0 \textbf{ then } 0 \quad \textbf{else } n + Sum[n-1].$$

is partially correct if

- $(\forall n : \mathbb{N})\ (n = 0 \Rightarrow 0 = \frac{n(n+1)}{2})$
- $(\forall n, m : \mathbb{N})\ (n \neq 0 \wedge m = \frac{(n-1)((n-1)+1)}{2} \Rightarrow n + m = \frac{n(n+1)}{2})$

Example Sum

Sum $(\forall n : \mathbb{N})\ (Sum[n] = \frac{n(n+1)}{2})$

$$Sum[n] = \quad \textbf{If } n = 0 \textbf{ then } 0$$
$$\textbf{else } n + Sum[n-1].$$

is partially correct if

- $(\forall n : \mathbb{N})\ (n = 0 \ \Rightarrow 0 = \frac{n(n+1)}{2})$
- $(\forall n, m : \mathbb{N})\ (n \neq 0 \wedge m = \frac{(n-1)((n-1)+1)}{2} \ \Rightarrow n + m = \frac{n(n+1)}{2})$

Example Sum

Sum $(\forall n : \mathbb{N})\ (Sum[n] = \frac{n(n+1)}{2})$

$$Sum[n] = \textbf{If } n = 0 \textbf{ then } 0 \quad \textbf{else } n + Sum[n-1].$$

terminates if

- $(\forall n : \mathbb{N})\ (Sum'[n] = \mathbf{T})$
- where:

$$Sum'[n] = \textbf{If } n = 0 \textbf{ then } \mathbf{T} \quad \textbf{else } Sum'[n-1].$$

Example Sum

Sum $(\forall n : \mathbb{N})\ (Sum[n] = \frac{n(n+1)}{2})$

$$Sum[n] = \quad \textbf{If } n = 0 \textbf{ then } 0$$
$$\textbf{else } n + Sum[n-1].$$

terminates if

- $(\forall n : \mathbb{N})\ (Sum'[n] = \mathbf{T})$
- where:

$$Sum'[n] = \quad \textbf{If } n = 0 \textbf{ then } \mathbf{T}$$
$$\textbf{else } Sum'[n-1].$$

Neville's Algorithm

Specification

Given a field K, two non-empty tuples x, a over K of same length n, s.t. $(\forall i, j)(i, j = 1, \ldots, n \wedge i \neq j \Rightarrow x_i \neq x_j)$

Find a polynomial $p \in \mathcal{P}[K]$, s.t. $deg[p] \leq n - 1$ and $(\forall i)(i = 1, \ldots, n \Rightarrow Eval[p, x_i] = a_i)$

Algorithm

$p[x, a] =$ **If** $\|a\| \leq 1$ **then** $First[a]$

else $\dfrac{(\mathcal{X} - First[x])(p[Tail[x], Tail[a]]) - (\mathcal{X} - Last[x])(p[Bgn[x], Bgn[a]])}{Last[x] - First[x]}.$

Neville's Algorithm

Specification

Given a field K, two non-empty tuples x, a over K of same length n, s.t. $(\forall i, j)(i, j = 1, \ldots, n \wedge i \neq j \Rightarrow x_i \neq x_j)$

Find a polynomial $p \in \mathcal{P}[K]$, s.t. $deg[p] \leq n - 1$ and $(\forall i)(i = 1, \ldots, n \Rightarrow Eval[p, x_i] = a_i)$

Algorithm

$p[x, a] =$ **If** $\|a\| \leq 1$ **then** $First[a]$

$$\textbf{else } \frac{(\mathcal{X} - First[x])(p[Tail[x], Tail[a]]) - (\mathcal{X} - Last[x])(p[Bgn[x], Bgn[a]])}{Last[x] - First[x]}.$$

Neville's Algorithm

is coherent if

- $(\forall x, a)(IsTuple[a] \wedge IsTuple[x] \wedge \|a\| \geq 1 \wedge$

  $(\forall i, j)(i, j = 1 \ldots \|a\| \wedge i \neq j \Rightarrow x_i \neq x_j) \wedge \|a\| \leq 1 \;\Rightarrow IsTuple[a] \wedge \|a\| \geq 1)$
- $(\forall x, a)(IsTuple[a] \wedge IsTuple[x] \wedge \|a\| \geq 1 \wedge$

  $(\forall i, j)(i, j = 1 \ldots \|a\| \wedge i \neq j \Rightarrow x_i \neq x_j) \wedge \neg(\|a\| \leq 1) \;\Rightarrow$

  $IsTuple[Tail[x]] \wedge IsTuple[Tail[a]] \wedge \|Tail[a]\| = \|Tail[x]\| \wedge \|Tail[a]\| \geq 1$

  $\wedge (\forall i, j)(i, j = 1 \ldots \|Tail[a]\| \wedge i \neq j \Rightarrow Tail[x]_i \neq Tail[x]_j)$
- ...
- ...

Neville's Algorithm

is coherent if

- $(\forall x, a)(IsTuple[a] \wedge IsTuple[x] \wedge \|a\| \geq 1 \wedge$

  $(\forall i, j)(i, j = 1 \ldots \|a\| \wedge i \neq j \Rightarrow x_i \neq x_j) \wedge \|a\| \leq 1 \ \Rightarrow IsTuple[a] \wedge \|a\| \geq 1)$

- $(\forall x, a)(IsTuple[a] \wedge IsTuple[x] \wedge \|a\| \geq 1 \wedge$

  $(\forall i, j)(i, j = 1 \ldots \|a\| \wedge i \neq j \Rightarrow x_i \neq x_j) \wedge \neg(\|a\| \leq 1) \ \Rightarrow$

  $IsTuple[Tail[x]] \wedge IsTuple[Tail[a]] \wedge \|Tail[a]\| = \|Tail[x]\| \wedge \|Tail[a]\| \geq 1$

  $\wedge (\forall i, j)(i, j = 1 \ldots \|Tail[a]\| \wedge i \neq j \Rightarrow Tail[x]_i \neq Tail[x]_j)$

- ...
- ...

Neville's Algorithm

is partially correct if and only if

- $\dots \wedge IsPoly[p_1] \wedge IsPoly[p_2] \Rightarrow IsPoly[\frac{(\mathcal{X}-First[x])p_1-(\mathcal{X}-Last[x])p_2}{Last[x]-First[x]}]$

- $\dots \wedge (\forall i)(i = 1 \dots \|Tail[x]\|)(Eval[p_1, Tail[x]_i]) = Tail[a]_i$

  $$\wedge(\forall i)(i = 1 \dots \|Bgn[x]\|)(Eval[p_2, Bgn[x]_i]) = Tail[a]_i$$

  $$\Rightarrow (\forall i)(i = 1 \dots \|a\|)(Eval[\frac{(\mathcal{X} - First[x])p_1 - (\mathcal{X} - Last[x])p_2}{Last[x] - First[x]}, x_i] = a_i)$$

- $\dots \wedge deg[p_1] \leq \|Tail[a]\| - 1 \wedge deg[p_2] \leq \|Bgn[a]\| - 1$

  $$\Rightarrow deg[\frac{(\mathcal{X} - First[x])p_1 - (\mathcal{X} - Last[x])p_2}{Last[x] - First[x]} \leq \|a\| - 1$$

- $\dots$

Neville's Algorithm

is partially correct if and only if

- $\cdots \wedge IsPoly[p_1] \wedge IsPoly[p_2] \Rightarrow IsPoly[\frac{(\mathcal{X}-First[x])p_1-(\mathcal{X}-Last[x])p_2}{Last[x]-First[x]}]$
- $\cdots \wedge (\forall i)(i = 1 \ldots \|Tail[x]\|)(Eval[p_1, Tail[x]_i]) = Tail[a]_i$

  $$\wedge(\forall i)(i = 1 \ldots \|Bgn[x]\|)(Eval[p_2, Bgn[x]_i]) = Tail[a]_i$$

  $$\Rightarrow (\forall i)(i = 1 \ldots \|a\|)(Eval[\frac{(\mathcal{X} - First[x])p_1 - (\mathcal{X} - Last[x])p_2}{Last[x] - First[x]}, x_i] = a_i)$$
- $\cdots \wedge deg[p_1] \leq \|Tail[a]\| - 1 \wedge deg[p_2] \leq \|Bgn[a]\| - 1$

  $$\Rightarrow deg[\frac{(\mathcal{X} - First[x])p_1 - (\mathcal{X} - Last[x])p_2}{Last[x] - First[x]} \leq \|a\| - 1$$
- $\ldots$

Neville's Algorithm

terminates if and only if

- $(\forall x, a : \mathit{IsTuple}[a] \wedge \mathit{IsTuple}[x] \wedge \|a\| = \|x\|) \quad p'[x, a] = \mathbf{T}$
- Where:

$$p'[x, a] = \quad \textbf{If } \|a\| \leq 1 \textbf{ then T}$$
$$\textbf{else } p'[\mathit{Tail}[x], \mathit{Tail}[a]] \wedge p'[\mathit{Bgn}[x], \mathit{Bgn}[a]].$$

Neville's Algorithm

terminates if and only if

- $(\forall x, a : \mathit{IsTuple}[a] \wedge \mathit{IsTuple}[x] \wedge \|a\| = \|x\|) \quad p'[x, a] = \mathbf{T}$
- Where:

$$p'[x, a] = \quad \textbf{If } \|a\| \leq 1 \textbf{ then T}$$
$$\textbf{else } p'[\mathit{Tail}[x], \mathit{Tail}[a]] \wedge p'[\mathit{Bgn}[x], \mathit{Bgn}[a]].$$

Neville's Algorithm

terminates if and only if

- $(\forall x, a : \mathit{IsTuple}[a] \wedge \mathit{IsTuple}[x] \wedge \|a\| = \|x\|) \quad p'[x, a] = \mathbf{T}$
- Where:

$$p'[x, a] = \quad \textbf{If } \|a\| \leq 1 \textbf{ then T} \\ \textbf{else } p'[\mathit{Tail}[x], \mathit{Tail}[a]] \wedge p'[\mathit{Bgn}[x], \mathit{Bgn}[a]].$$

Outline

Further Work

- Implementation;
- Test on many examples;
- Special induction provers for the termination conditions;
- Use of other termination results.