Using semantic labelling with natural numbers for proving termination automatically.

Adam Koprowski

29th June 2005

Outline

- Overview of TPA

Outline

- Overview of TPA
- General idea of semantic labelling with natural numbers.

Outline

- Overview of TPA
- General idea of semantic labelling with natural numbers.
- Why SLnat was considered not to be suitable for automatic termination provers.

Outline

- Overview of TPA
- General idea of semantic labelling with natural numbers.
- Why SLnat was considered not to be suitable for automatic termination provers.
- Implementing SLnat

Outline

- Overview of TPA
- General idea of semantic labelling with natural numbers.
- Why SLnat was considered not to be suitable for automatic termination provers.
- Implementing SLnat
  - SLnat and polynomial interpretations

Outline

- Overview of TPA
- General idea of semantic labelling with natural numbers.
- Why SLnat was considered not to be suitable for automatic termination provers.
- Implementing SLnat
  - SLnat and polynomial interpretations
  - SLnat and recursive path ordering (RPO)

Outline

- Overview of TPA
- General idea of semantic labelling with natural numbers.
- Why SLnat was considered not to be suitable for automatic termination provers.
- Implementing SLnat
  - SLnat and polynomial interpretations
  - SLnat and recursive path ordering (RPO)
- Example

Outline

- Overview of TPA
- General idea of semantic labelling with natural numbers.
- Why SLnat was considered not to be suitable for automatic termination provers.
- Implementing SLnat
  - SLnat and polynomial interpretations
  - SLnat and recursive path ordering (RPO)
- Example
- Conclusions

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

- TPA supports relative termination.

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

- TPA supports relative termination.
- TPA uses SLnat technique.

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

- TPA supports relative termination.
- TPA uses SLnat technique.
- TPA aims at certified termination proofs.

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

- TPA supports relative termination.
- TPA uses SLnat technique.
- TPA aims at certified termination proofs.

Some facts about TPA:

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

- TPA supports relative termination.
- TPA uses SLnat technique.
- TPA aims at certified termination proofs.

Some facts about TPA:

- Developed at TU/e by author since Nov 2004.

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

- TPA supports relative termination.
- TPA uses SLnat technique.
- TPA aims at certified termination proofs.

Some facts about TPA:

- Developed at TU/e by author since Nov 2004.
- 3rd place (out of 6 participants) in international termination competition (Nara 2005).

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

- TPA supports relative termination.
- TPA uses SLnat technique.
- TPA aims at certified termination proofs.

Some facts about TPA:

- Developed at TU/e by author since Nov 2004.
- 3rd place (out of 6 participants) in international termination competition (Nara 2005).
- Written in Ocaml (strong type checking, lazy evaluation).

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

- TPA supports relative termination.
- TPA uses SLnat technique.
- TPA aims at certified termination proofs.

Some facts about TPA:

- Developed at TU/e by author since Nov 2004.
- 3rd place (out of 6 participants) in international termination competition (Nara 2005).
- Written in Ocaml (strong type checking, lazy evaluation).
- 9000 lines of code at the moment (and growing!).

TPA (Termination Proved Automatically)

TPA is a tool for proving termination of TRSs fully automatically.
Why making yet another such a tool?

- TPA supports relative termination.
- TPA uses SLnat technique.
- TPA aims at certified termination proofs.

Some facts about TPA:

- Developed at TU/e by author since Nov 2004.
- 3rd place (out of 6 participants) in international termination competition (Nara 2005).
- Written in Ocaml (strong type checking, lazy evaluation).
- 9000 lines of code at the moment (and growing!).

http://www.win.tue.nl/tpa

Termination techniques used by TPA

- Polynomial interpretations,

Termination techniques used by TPA

- Polynomial interpretations,
- Recursive path ordering (RPO),

Termination techniques used by TPA

- Polynomial interpretations,
- Recursive path ordering (RPO),
- Semantic labelling with booleans,

Termination techniques used by TPA

- Polynomial interpretations,
- Recursive path ordering (RPO),
- Semantic labelling with booleans,
- Semantic labelling with natural numbers (SLnat),

Termination techniques used by TPA

- Polynomial interpretations,
- Recursive path ordering (RPO),
- Semantic labelling with booleans,
- Semantic labelling with natural numbers (SLnat),
- Simple version of dependency pairs transformation (DP),

Termination techniques used by TPA

- Polynomial interpretations,
- Recursive path ordering (RPO),
- Semantic labelling with booleans,
- Semantic labelling with natural numbers (SLnat),
- Simple version of dependency pairs transformation (DP),
- Dummy elimination,

Termination techniques used by TPA

- Polynomial interpretations,
- Recursive path ordering (RPO),
- Semantic labelling with booleans,
- Semantic labelling with natural numbers (SLnat),
- Simple version of dependency pairs transformation (DP),
- Dummy elimination,
- ... more to come?

CoLoR (Coq library on rewriting and termination)

Goal: formalization of theoretical results concerning termination of term rewriting.

CoLoR (Coq library on rewriting and termination)

Goal: formalization of theoretical results concerning termination of term rewriting.

Ultimate goal: certification of proof candidates produced by tools for proving termination automatically.

CoLoR (Coq library on rewriting and termination)

Goal: formalization of theoretical results concerning termination of term rewriting.

Ultimate goal: certification of proof candidates produced by tools for proving termination automatically.

Content of CoLoR :

- Basic dependency pairs transformation and arguments filtering (Frédéric Blanqui)

CoLoR (Coq library on rewriting and termination)

Goal: formalization of theoretical results concerning termination of term rewriting.

Ultimate goal: certification of proof candidates produced by tools for proving termination automatically.

Content of CoLoR :

- Basic dependency pairs transformation and arguments filtering (Frédéric Blanqui)
- Polynomial interpretations (Sébastien Hinderer)

CoLoR (Coq library on rewriting and termination)

Goal: formalization of theoretical results concerning termination of term rewriting.

Ultimate goal: certification of proof candidates produced by tools for proving termination automatically.

Content of CoLoR :

- Basic dependency pairs transformation and arguments filtering (Frédéric Blanqui)
- Polynomial interpretations (Sébastien Hinderer)
- RPO (William Delobel and Solange Coupet-Grimal)

CoLoR (Coq library on rewriting and termination)

Goal: formalization of theoretical results concerning termination of term rewriting.

Ultimate goal: certification of proof candidates produced by tools for proving termination automatically.

Content of CoLoR :

- Basic dependency pairs transformation and arguments filtering (Frédéric Blanqui)
- Polynomial interpretations (Sébastien Hinderer)
- RPO (William Delobel and Solange Coupet-Grimal)
- HORPO (Adam Koprowski and Femke van Raamsdonk)

CoLoR (Coq library on rewriting and termination)

Goal: formalization of theoretical results concerning termination of term rewriting.

Ultimate goal: certification of proof candidates produced by tools for proving termination automatically.

Content of CoLoR :

- Basic dependency pairs transformation and arguments filtering (Frédéric Blanqui)
- Polynomial interpretations (Sébastien Hinderer)
- RPO (William Delobel and Solange Coupet-Grimal)
- HORPO (Adam Koprowski and Femke van Raamsdonk)

Contributions are very welcome!

http://color.loria.fr/

SLnat - general idea

Is the following system terminating?

$$
\begin{array}{lrcl}
(1) & \mathrm{f}(x, \mathrm{c}) & \rightarrow & \mathrm{c} \\
(2) & \mathrm{f}(\mathrm{c}, y) & \rightarrow & \mathrm{c} \\
(3) & \mathrm{f}(\mathrm{p}(x), \mathrm{p}(y)) & \rightarrow & \mathrm{p}(\mathrm{f}(x, y)) \\
(5) & \mathrm{g}(x, \mathrm{c}) & \rightarrow & x \\
(4) & \mathrm{g}(\mathrm{c}, y) & \rightarrow & y \\
(6) & \mathrm{g}(\mathrm{p}(x), \mathrm{p}(y)) & \rightarrow & \mathrm{p}(\mathrm{g}(x, y)) \\
(7) & \mathrm{w}(x, \mathrm{c}) & \rightarrow & x \\
(8) & \mathrm{w}(\mathrm{p}(x), \mathrm{p}(y)) & \rightarrow & \mathrm{w}(x, y) \\
(9) & \mathrm{h}(\mathrm{p}(x), \mathrm{p}(y)) & \rightarrow & \mathrm{h}(\mathrm{w}(\mathrm{p}(\mathrm{g}(x, y)), \mathrm{p}(\mathrm{f}(x, y))), \mathrm{p}(\mathrm{f}(x, y)))
\end{array}
$$

SLnat - general idea (continued)

And how about this one?

$$
\begin{array}{lrcl}
(1) & \min(x, 0) & \to & 0 \\
(2) & \min(0, y) & \to & 0 \\
(3) & \min(\mathrm{s}(x), \mathrm{s}(y)) & \to & \mathrm{s}(\min(x, y)) \\
(5) & \max(x, 0) & \to & x \\
(4) & \max(0, y) & \to & y \\
(6) & \max(\mathrm{s}(x), \mathrm{s}(y)) & \to & \mathrm{s}(\max(x, y)) \\
(7) & x - 0 & \to & x \\
(8) & \mathrm{s}(x) - \mathrm{s}(y) & \to & x - y \\
(9) & \gcd(\mathrm{s}(x), \mathrm{s}(y)) & \to & \gcd(\mathrm{s}(\max(x, y)) - \mathrm{s}(\min(x, y)), \mathrm{s}(\min(x, y)))
\end{array}
$$

Natural interpretation for function symbols:

$$
\begin{array}{rclcrcl}
[0] & = & 0 & \qquad & [\mathrm{min}(x,y)] & = & \mathrm{min}(x,y) \\
[\mathrm{s}(x)] & = & x+1 & & [\mathrm{max}(x,y)] & = & \mathrm{max}(x,y) \\
[x-y] & = & x-y & & [\mathrm{gcd}(x,y)] & = & \mathrm{gcd}(x,y)
\end{array}
$$

Natural interpretation for function symbols:

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
[0] & = & 0 & [\mathrm{min}(x,y)] & = & \mathrm{min}(x,y) \\
[\mathrm{s}(x)] & = & x+1 & [\mathrm{max}(x,y)] & = & \mathrm{max}(x,y) \\
[x-y] & = & x-y & [\mathrm{gcd}(x,y)] & = & \mathrm{gcd}(x,y)
\end{array}
$$

$$
\begin{array}{rcll}
\mathrm{min}_{i,0}(x,0) & \to & 0 & \\
\mathrm{min}_{0,j}(0,y) & \to & 0 & \\
\mathrm{min}_{i+1,j+1}(\mathrm{s}_i(x),\mathrm{s}_j(y)) & \to & \mathrm{s}_j(\mathrm{min}_{i,j}(x,y)) & \text{if } i >= j \\
\mathrm{min}_{i+1,j+1}(\mathrm{s}_i(x),\mathrm{s}_j(y)) & \to & \mathrm{s}_i(\mathrm{min}_{i,j}(x,y)) & \text{if } i < j \\
\mathrm{max}_{i,0}(x,0) & \to & x & \\
\mathrm{max}_{0,j}(0,y) & \to & y & \\
\mathrm{max}_{i+1,j+1}(\mathrm{s}_i(x),\mathrm{s}_j(y)) & \to & \mathrm{s}_i(\mathrm{max}_{i,j}(x,y)) & \text{if } i >= j \\
\mathrm{max}_{i+1,j+1}(\mathrm{s}_i(x),\mathrm{s}_j(y)) & \to & \mathrm{s}_j(\mathrm{max}_{i,j}(x,y)) & \text{if } i < j \\
x -_{i,0} 0 & \to & x & \\
\mathrm{s}_i(x) -_{i+1,j+1} \mathrm{s}_j(y)) & \to & x -_{i,j} y & \\
\mathrm{gcd}_{i+1,j+1}(\mathrm{s}_i(x),\mathrm{s}_j(y)) & \to & \mathrm{gcd}_{i-j,j+1}(\mathrm{s}_i(\mathrm{max}_{i,j}(x,y)) -_{i+1,j+1} \mathrm{s}_j(\mathrm{min}_{i,j}(x,y)), & \text{if } i >= j \\
 & & \mathrm{min}_{i,j}(x,y))) & \\
\mathrm{gcd}_{i+1,j+1}(\mathrm{s}_i(x),\mathrm{s}_j(y)) & \to & \mathrm{gcd}_{j-i,i+1}(\mathrm{s}_j(\mathrm{max}_{i,j}(x,y)) -_{j+1,i+1} \mathrm{s}_i(\mathrm{min}_{i,j}(x,y)), & \text{if } i < j \\
 & & \mathrm{min}_{i,j}(x,y))) &
\end{array}
$$

If we label only gcd symbol with $\pi_{\text{gcd}}(x, y) = x + y$ then we get the following system:

$$
\begin{array}{lrcl}
(1) & \min(x, 0) & \rightarrow & 0 \\
(2) & \min(0, y) & \rightarrow & 0 \\
(3) & \min(\mathrm{s}(x), \mathrm{s}(y)) & \rightarrow & \mathrm{s}(\min(x, y)) \\
(5) & \max(x, 0) & \rightarrow & x \\
(4) & \max(0, y) & \rightarrow & y \\
(6) & \max(\mathrm{s}(x), \mathrm{s}(y)) & \rightarrow & \mathrm{s}(\max(x, y)) \\
(7) & x - 0 & \rightarrow & x \\
(8) & \mathrm{s}(x) - \mathrm{s}(y) & \rightarrow & x - y \\
(9) & \mathrm{gcd}_i(\mathrm{s}(x), \mathrm{s}(y)) & \rightarrow & \mathrm{gcd}_j(\mathrm{s}(\max(x, y)) - \mathrm{s}(\min(x, y)), \mathrm{s}(\min(x, y))), \ \text{for } i > j
\end{array}
$$

If we label only gcd symbol with $\pi_{\text{gcd}}(x, y) = x + y$ then we get the following system:

$$
\begin{array}{lrcl}
(1) & \min(x, 0) & \to & 0 \\
(2) & \min(0, y) & \to & 0 \\
(3) & \min(\mathrm{s}(x), \mathrm{s}(y)) & \to & \mathrm{s}(\min(x, y)) \\
(5) & \max(x, 0) & \to & x \\
(4) & \max(0, y) & \to & y \\
(6) & \max(\mathrm{s}(x), \mathrm{s}(y)) & \to & \mathrm{s}(\max(x, y)) \\
(7) & x - 0 & \to & x \\
(8) & \mathrm{s}(x) - \mathrm{s}(y) & \to & x - y \\
(9) & \mathrm{gcd}_i(\mathrm{s}(x), \mathrm{s}(y)) & \to & \mathrm{gcd}_j(\mathrm{s}(\max(x, y)) - \mathrm{s}(\min(x, y)), \mathrm{s}(\min(x, y))), \text{ for } i > j
\end{array}
$$

which can easily be proved to be terminating by RPO with the following precedence:

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
\min & > & \mathrm{s} & \mathrm{gcd}_i & > & \min \\
\max & > & \mathrm{s} & \mathrm{gcd}_i & > & \max \\
\mathrm{gcd}_i & > & - & \mathrm{gcd}_i & > & \mathrm{s} \\
\mathrm{gcd}_i & > & \mathrm{gcd}_j, \text{ for } i > j & & &
\end{array}
$$

What are the problems with using SLnat in automatic termination provers?

What are the problems with using SLnat in automatic termination provers?

- How to find interpretations that give raise to a model?

What are the problems with using SLnat in automatic termination provers?

- How to find interpretations that give raise to a model?
- Labelled system has an infinite signature and infinitely many rules.

What are the problems with using SLnat in automatic termination provers?

- How to find interpretations that give raise to a model?
- Labelled system has an infinite signature and infinitely many rules.
- If we are to use min and max functions then we need to deal with side conditions.

What are the problems with using SLnat in automatic termination provers?

- How to find interpretations that give raise to a model?
- Labelled system has an infinite signature and infinitely many rules.
- If we are to use min and max functions then we need to deal with side conditions.

TPA does the following transformation to reduce any TRS to a TRS containing only constants, unary and binary symbols.

$$\boxed{\mathrm{f}(x_1, \ldots, x_n) \equiv \mathrm{f}'(x_1, \mathrm{f}''(x_2, \ldots))}$$

Proving termination using semantic labelling.

For every arity (constants, unary and binary symbols) there is a predefined set of basic functions to be tried as interpretations.

Proving termination using semantic labelling.

For every arity (constants, unary and binary symbols) there is a predefined set of basic functions to be tried as interpretations.

Combinations of those functions are tried in the search for a (quasi-) model. The search space is finite.

Proving termination using semantic labelling.

For every arity (constants, unary and binary symbols) there is a predefined set of basic functions to be tried as interpretations.

Combinations of those functions are tried in the search for a (quasi-) model. The search space is finite.

For every obtained (quasi-)model some techniques are applied to the labelled system in order to try to prove its termination.

SLnat and polynomial interpretations

TPA has a basic sets of polynomial interpretations for constants (I_c), unary (I_u) and binary symbols (I_b).

SLnat and polynomial interpretations

TPA has a basic sets of polynomial interpretations for constants (I_c), unary (I_u) and binary symbols (I_b).

For labelled system those sets are extended in the following way:

$$\begin{array}{rcl} [c] & \in & I_c \\ [s_m(x)] & \in & \{i_u,\ i_u + m \mid i_u \in I_u\} \\ [f_{m,n}(x,y)] & \in & \{i_b,\ i_b + m,\ i_b + n,\ i_b + m + n \mid i_b \in I_b\} \end{array}$$

$$\begin{array}{rcl} f(I(x),y) & \to & f(x,y) \\ f(x,y) & \to= & f(x,I(y)) \end{array}$$

SLnat and polynomial interpretations

TPA has a basic sets of polynomial interpretations for constants (I_c), unary (I_u) and binary symbols (I_b).

For labelled system those sets are extended in the following way:

$$\begin{array}{rcl} [c] & \in & I_c \\ [s_m(x)] & \in & \{i_u,\ i_u + m \mid i_u \in I_u\} \\ [f_{m,n}(x,y)] & \in & \{i_b,\ i_b + m,\ i_b + n,\ i_b + m + n \mid i_b \in I_b\} \end{array}$$

$$\begin{array}{rcl} f(I(x),y) & \rightarrow & f(x,y) \\ f(x,y) & \rightarrow= & f(x,I(y)) \end{array}$$

$$\begin{array}{rcl} [f(x,y)] & = & 2 \\ [I(x)] & = & x+1 \end{array}$$

SLnat and polynomial interpretations

TPA has a basic sets of polynomial interpretations for constants (I_c), unary (I_u) and binary symbols (I_b).

For labelled system those sets are extended in the following way:

$$\begin{array}{rcl} [c] & \in & I_c \\ [s_m(x)] & \in & \{i_u,\ i_u + m \mid i_u \in I_u\} \\ [f_{m,n}(x,y)] & \in & \{i_b,\ i_b + m,\ i_b + n,\ i_b + m + n \mid i_b \in I_b\} \end{array}$$

$$\begin{array}{rcl} f(I(x),y) & \to & f(x,y) \\ f(x,y) & \to= & f(x,I(y)) \end{array}$$

$$\begin{array}{rcl} [f(x,y)] & = & 2 \\ [I(x)] & = & x+1 \end{array}$$

$$\begin{array}{rcl} f_{i+1,j}(I(x),y) & \to & f_{i,j}(x,y) \\ f_{i,j}(x,y) & \to= & f_{i,j+1}(x,I(y)) \end{array}$$

SLnat and polynomial interpretations

TPA has a basic sets of polynomial interpretations for constants (I_c), unary (I_u) and binary symbols (I_b).

For labelled system those sets are extended in the following way:

$$\begin{array}{rcl} [c] & \in & I_c \\ [s_m(x)] & \in & \{i_u,\ i_u+m \mid i_u \in I_u\} \\ [f_{m,n}(x,y)] & \in & \{i_b,\ i_b+m,\ i_b+n,\ i_b+m+n \mid i_b \in I_b\} \end{array}$$

$$\begin{array}{rcl} f(I(x),y) & \to & f(x,y) \\ f(x,y) & \to= & f(x,I(y)) \end{array}$$

$$\begin{array}{rcl} [f(x,y)] & = & 2 \\ [I(x)] & = & x+1 \end{array}$$

$$\begin{array}{rcl} f_{i+1,j}(I(x),y) & \to & f_{i,j}(x,y) \\ f_{i,j}(x,y) & \to= & f_{i,j+1}(x,I(y)) \end{array}$$

$$\begin{array}{rcl} [f_{i,j}(x,y)] & = & x+y+i \\ [I(x)] & = & x \end{array}$$

TPA meets RPO

Precedence is an well-founded ordering on function symbols.

TPA meets RPO

Precedence is an well-founded ordering on function symbols.

- Orient rules of TRS collecting requirements on precedence.

TPA meets RPO

Precedence is an well-founded ordering on function symbols.

- Orient rules of TRS collecting requirements on precedence.
- Check whether the requirements can be combined in order to form a precedence.

TPA meets RPO

Precedence is an well-founded ordering on function symbols.

- Orient rules of TRS collecting requirements on precedence.
- Check whether the requirements can be combined in order to form a precedence.

Precedence is well-founded if the corresponding directed graph is acyclic.

$$
\begin{array}{lrcl}
(1) & 0 + x & \rightarrow & x \\
(2) & \mathrm{s}(x) + y & \rightarrow & \mathrm{s}(x + y) \\
(3) & 0 * x & \rightarrow & 0 \\
(4) & \mathrm{s}(x) * y & \rightarrow & y + (x * y)
\end{array}
$$

TPA meets RPO

Precedence is an well-founded ordering on function symbols.

- Orient rules of TRS collecting requirements on precedence.
- Check whether the requirements can be combined in order to form a precedence.

Precedence is well-founded if the corresponding directed graph is acyclic.

$$\begin{array}{lrcl} (1) & 0 + x & \rightarrow & x \\ (2) & \mathrm{s}(x) + y & \rightarrow & \mathrm{s}(x + y) \\ (3) & 0 * x & \rightarrow & 0 \\ (4) & \mathrm{s}(x) * y & \rightarrow & y + (x * y) \end{array}$$

$$\begin{array}{ccc} * & > & + \\ + & > & \mathrm{s} \end{array}$$

TPA meets RPO

Precedence is an well-founded ordering on function symbols.

- Orient rules of TRS collecting requirements on precedence.
- Check whether the requirements can be combined in order to form a precedence.

Precedence is well-founded if the corresponding directed graph is acyclic.

$$\begin{array}{lrcl} (1) & 0+x & \rightarrow & x \\ (2) & \mathrm{s}(x)+y & \rightarrow & \mathrm{s}(x+y) \\ (3) & 0*x & \rightarrow & 0 \\ (4) & \mathrm{s}(x)*y & \rightarrow & y+(x*y) \end{array}$$

$$\begin{array}{ccc} * & > & + \\ + & > & \mathrm{s} \end{array}$$

$$* \longrightarrow + \longrightarrow \mathrm{s}$$

$$\mathrm{compare} : \mathrm{funS} \times \mathrm{funS} \rightarrow \{<, =, >, ?\}$$

RPO with SLnat

- In RPO we want to take advantage of having labels.

RPO with SLnat

- In RPO we want to take advantage of having labels.
- Goal: capture infinite ordering in a finite description:

RPO with SLnat

- In RPO we want to take advantage of having labels.
- Goal: capture infinite ordering in a finite description:
  - Expressiveness

RPO with SLnat

- In RPO we want to take advantage of having labels.
- Goal: capture infinite ordering in a finite description:
  - Expressiveness
  - Easy handling (checking well-foundedness of an ordering)

RPO with SLnat

- In RPO we want to take advantage of having labels.
- Goal: capture infinite ordering in a finite description:
  - Expressiveness
  - Easy handling (checking well-foundedness of an ordering)

$$\begin{array}{l} \text{compare} : \text{funS} \times \text{funS} \rightarrow \text{argsF} \times \text{argsF} \times \text{cmpRes} \times \text{cmpRes} \times \text{cmpRes} \\ \text{cmpRes} : \{<, >, ?\} \\ \text{argsF} : \{\leftarrow, \rightarrow, \leftrightarrow\} \end{array}$$

if $\text{compare}(f, g) = (\Gamma_f, \Gamma_g, \Delta_<, \Delta_=, \Delta_>)$

$$\begin{array}{l} \text{args}_{\leftarrow}(l, r) = l \\ \text{args}_{\rightarrow}(l, r) = r \\ \text{args}_{\leftrightarrow}(l, r) = l + r \end{array}$$

$$f_{i,j} \otimes g_{k,l} \quad \text{if} \quad \text{args}_{\Gamma_f}(i, j) \begin{array}{c} < \\ = \\ > \end{array} \text{args}_{\Gamma_g}(k, l) \wedge \begin{array}{c} \Delta_< \\ \Delta_= \\ \Delta_> \end{array} = \otimes$$

Criterion for well-foundedness of an ordering

Now a relation induced by compare function corresponds to a multigraph with labelled edges.

Criterion for well-foundedness of an ordering

Now a relation induced by compare function corresponds to a multigraph with labelled edges.

Edges:

$f \xrightarrow{\text{cond}} g$ meaning $f > g$ under given condition cond

Criterion for well-foundedness of an ordering

Now a relation induced by compare function corresponds to a multigraph with labelled edges.

Edges:

$$f \xrightarrow{\text{cond}} g \quad \text{meaning } f > g \text{ under given condition cond}$$

Where condition is of the shape:

$$X \otimes Y$$
$$X, Y \in \{\leftarrow, \rightarrow, \leftrightarrow\}$$
$$\otimes \in \{<, =, >\}$$

RPO with SLnat - examples of precedence

Expected result	Precedence function	Edges in a multigraph
$f_{i,j} > g_{k,l}$ for all i, j, k, l	$\Omega(f, g) = (\leftrightarrow, \leftrightarrow, >, >, >)$	$f \overset{\leftrightarrow>\leftrightarrow}{\underset{\leftrightarrow<\leftrightarrow}{\xrightarrow{\leftrightarrow=\leftrightarrow}}} g$

RPO with SLnat - examples of precedence

Expected result	Precedence function	Edges in a multigraph
$f_{i,j} > g_{k,l}$ for all i, j, k, l	$\Omega(f, g) = (\leftrightarrow, \leftrightarrow, >, >, >)$	$f \xrightarrow{\leftrightarrow>\leftrightarrow} g$, $f \xrightarrow{\leftrightarrow=\leftrightarrow} g$, $f \xrightarrow{\leftrightarrow<\leftrightarrow} g$
$s_1 < s_2 < \ldots$	$\Omega(s, s) = (\leftrightarrow, \leftrightarrow, <, =, >)$	$s \xrightarrow{\leftrightarrow>\leftrightarrow} s$

RPO with SLnat - examples of precedence

Expected result	Precedence function	Edges in a multigraph
$f_{i,j} > g_{k,l}$ for all i, j, k, l	$\Omega(f, g) = (\leftrightarrow, \leftrightarrow, >, >, >)$	$\begin{tikzcd} f \arrow[r, bend left, "\leftrightarrow>\leftrightarrow"] \arrow[r, "\leftrightarrow=\leftrightarrow"] \arrow[r, bend right, "\leftrightarrow<\leftrightarrow"'] & g \end{tikzcd}$
$s_1 < s_2 < \ldots$	$\Omega(s, s) = (\leftrightarrow, \leftrightarrow, <, =, >)$	$\begin{tikzcd} s \arrow[loop above, "\leftrightarrow>\leftrightarrow"] \end{tikzcd}$
$s_1 < t_1 < s_2 < t_2 < \ldots$	$\Omega(s, t) = (\leftrightarrow, \leftrightarrow, <, <, >)$	$\begin{tikzcd} s \arrow[r, bend left, "\leftrightarrow>\leftrightarrow"] & t \arrow[l, bend left, "\leftrightarrow\leq\leftrightarrow"] \end{tikzcd}$

Criterion for well-foundedness of an ordering (cont.)

We call a cycle in a multigraph safe, if:

Criterion for well-foundedness of an ordering (cont.)

We call a cycle in a multigraph <u>safe</u>, if:

- for every condition $\circ \xrightarrow{X \otimes Y} \circ$, $\otimes \in \{>, =\}$,

Criterion for well-foundedness of an ordering (cont.)

We call a cycle in a multigraph <u>safe</u>, if:

- for every condition $\circ \xrightarrow{X \otimes Y} \circ$, $\otimes \in \{>, =\}$,
- at least for one condition $\circ \xrightarrow{X \otimes Y} \circ$, $\otimes = >$ and

Criterion for well-foundedness of an ordering (cont.)

We call a cycle in a multigraph safe, if:

- for every condition $\circ \xrightarrow{X \otimes Y} \circ$, $\otimes \in \{>, =\}$,
- at least for one condition $\circ \xrightarrow{X \otimes Y} \circ$, $\otimes = >$ and
- for every two consecutive edges $\circ \xrightarrow{W \otimes_1 X} \circ \xrightarrow{Y \otimes_2 Z} \circ$, $X = Y$.

Criterion for well-foundedness of an ordering (cont.)

We call a cycle in a multigraph safe, if:

- for every condition $\circ \xrightarrow{X \otimes Y} \circ$, $\otimes \in \{>, =\}$,
- at least for one condition $\circ \xrightarrow{X \otimes Y} \circ$, $\otimes = >$ and
- for every two consecutive edges $\circ \xrightarrow{W \otimes_1 X} \circ \xrightarrow{Y \otimes_2 Z} \circ$, $X = Y$.

  Those three conditions assure that some weight-function on indexes decrease along the cycle.

$$\cdots \xrightarrow{X \geq Y} f \xrightarrow{Y \geq Z} \cdots \qquad X \geq Y = Y \geq Z$$

Conjecture:
Relation is well-founded, and hence can be extended to precedence, if every cycle in its multigraph is safe.

Motivating example

Consider the following TRS (SUBST) which describes the process of substitution in combinatory categorical logic.

$$
\begin{array}{llcl}
(1) & \lambda(x) \circ y & \to & \lambda(x \circ (1 \cdot (y \circ \uparrow))) \\
(2) & (x \cdot y) \circ z & \to & (x \circ z) \cdot (y \circ z) \\
(3) & (x \circ y) \circ z & \to & x \circ (y \circ z) \\
(4) & \mathrm{id} \circ x & \to & x \\
(5) & 1 \circ \mathrm{id} & \to & 1 \\
(6) & \uparrow \circ\, \mathrm{id} & \to & \uparrow \\
(7) & 1 \circ (x \cdot y) & \to & x \\
(8) & \uparrow \circ\, (x \cdot y) & \to & y
\end{array}
$$

- Termination of this system (implying termination of the process of explicit substitution in untyped λ-calculus) was the main result of two publications: Hardin and Lavills [1986], Curien et al. [1992].

Motivating example

Consider the following TRS (SUBST) which describes the process of substitution in combinatory categorical logic.

$$
\begin{array}{lrcl}
(1) & \lambda(x) \circ y & \to & \lambda(x \circ (1 \cdot (y \circ \uparrow))) \\
(2) & (x \cdot y) \circ z & \to & (x \circ z) \cdot (y \circ z) \\
(3) & (x \circ y) \circ z & \to & x \circ (y \circ z) \\
(4) & \mathrm{id} \circ x & \to & x \\
(5) & 1 \circ \mathrm{id} & \to & 1 \\
(6) & \uparrow \circ\, \mathrm{id} & \to & \uparrow \\
(7) & 1 \circ (x \cdot y) & \to & x \\
(8) & \uparrow \circ\, (x \cdot y) & \to & y
\end{array}
$$

- Termination of this system (implying termination of the process of explicit substitution in untyped λ-calculus) was the main result of two publications: Hardin and Lavills [1986], Curien et al. [1992].
- Using technique of SLnat a very concise proof can be given.

Motivating example

Consider the following TRS (SUBST) which describes the process of substitution in combinatory categorical logic.

$$
\begin{array}{llcl}
(1) & \lambda(x) \circ y & \to & \lambda(x \circ (1 \cdot (y \circ \uparrow))) \\
(2) & (x \cdot y) \circ z & \to & (x \circ z) \cdot (y \circ z) \\
(3) & (x \circ y) \circ z & \to & x \circ (y \circ z) \\
(4) & \mathrm{id} \circ x & \to & x \\
(5) & 1 \circ \mathrm{id} & \to & 1 \\
(6) & \uparrow \circ\, \mathrm{id} & \to & \uparrow \\
(7) & 1 \circ (x \cdot y) & \to & x \\
(8) & \uparrow \circ\, (x \cdot y) & \to & y
\end{array}
$$

- Termination of this system (implying termination of the process of explicit substitution in untyped λ-calculus) was the main result of two publications: Hardin and Lavills [1986], Curien et al. [1992].
- Using technique of SLnat a very concise proof can be given.
- Now it can be given by TPA.

Termination proof of SUBST as given by TPA

```
  TPA v.1.0b

Result: TRS is terminating

Default interpretations for symbols are not printed. For polynomial interpretations
and semantic labelling over N\{0,1} defaults are 2 for constants, identity for unary
symbols and x+y-2 for binary symbols. For semantic labelling over {0,1} (booleans)
defaults are 0 for constants, identity for unary symbols and disjunction for binary
symbols.

[1]  TRS as loaded from the input file:
(1) o(lambda(x),y) -> lambda(o(x,d(1,o(y,p))))
(2) o(d(x,y),z) -> d(o(x,z),o(y,z))
(3) o(o(x,y),z) -> o(x,o(y,z))
(4) lambda(x) -> x
(5) o(x,y) -> x
(6) o(x,y) -> y
(7) d(x,y) -> x
(8) d(x,y) -> y

[2]  Label this TRS using following interpretation over N\{0,1}:
[lambda(x)] = x + 1
[d(x,y)] = max(x, y)
rest default
```

Termination proof of SUBST as given by TPA (cont.)

```
  This interpretation is a quasi-model and yields following TRS:
(D1) lambda{i + 1}(x) ->= lambda{i}(x)
(D2) o{i + 1,j}(x,y) ->= o{i,j}(x,y)
(D3) o{i,j + 1}(x,y) ->= o{i,j}(x,y)
(D4) d{i + 1,j}(x,y) ->= d{i,j}(x,y)
(D5) d{i,j + 1}(x,y) ->= d{i,j}(x,y)
(1) o{i + 1,j}(lambda{i}(x),y) -> lambda{j + i - 2}(o{i,j}(x,d{2,j}(1,o{j,2}(y,p))))
(2<) o{i,k}(d{i,j}(x,y),z) -> d{k + i - 2,k + j - 2}(o{i,k}(x,z),o{j,k}(y,z)) for i >= j
(2>) o{j,k}(d{i,j}(x,y),z) -> d{k + i - 2,k + j - 2}(o{i,k}(x,z),o{j,k}(y,z)) for j >= i
(3) o{j + i - 2,k}(o{i,j}(x,y),z) -> o{i,k + j - 2}(x,o{j,k}(y,z))
(4) lambda{i}(x) -> x
(5) o{i,j}(x,y) -> x
(6) o{i,j}(x,y) -> y
(7<) d{i,j}(x,y) -> x for i >= j
(7>) d{i,j}(x,y) -> x for j >= i
(8>) d{i,j}(x,y) -> y for j >= i
(8<) d{i,j}(x,y) -> y for i >= j

[3]  All the rules of this TRS can be oriented with RPO with the following precedence:
Status: o: Lex-LR,
Precedence:
o{i,j} > o{k,l} for i+j > k+l
o{i,j} > lambda{k} for all i, j, k
o{i,j} > d{k,l} for all i, j, k, l
o{i,j} > 1 for all i, j
o{i,j} > p for all i, j
lambda{i} > lambda{k} for i > k
d{i,j} > d{k,l} for i+j > k+l
```

Conclusions

- SLnat can be successfully used for proving termination automatically.

Conclusions

- SLnat can be successfully used for proving termination automatically.
- It is possible to achieve quite satisfactory results with a limited set of termination techniques and without using dependency pairs transformation (CiME vs TPA).

Conclusions

- SLnat can be successfully used for proving termination automatically.
- It is possible to achieve quite satisfactory results with a limited set of termination techniques and without using dependency pairs transformation (CiME vs TPA).

Thank you for your attention!

Conclusions

- SLnat can be successfully used for proving termination automatically.
- It is possible to achieve quite satisfactory results with a limited set of termination techniques and without using dependency pairs transformation (CiME vs TPA).

Thank you for your attention!

Questions?