Termination of Narrowing in Left-Linear Constructor Systems

Germán Vidal

Technical University of Valencia, Spain

9th Int'l Symposium on Functional and Logic Programming
FLOPS 2008

April 14-16, 2008
Ise, Japan

What is narrowing?

Standard definition of addition (TRS)

$$
\begin{array}{rcll}
\mathrm{add}(\mathrm{z}, y) & \rightarrow & y & (R_1) \\
\mathrm{add}(\mathrm{s}(x), y) & \rightarrow & \mathrm{s}(\mathrm{add}(x, y)) & (R_2)
\end{array}
$$

With **rewriting**: $\mathrm{add}(\mathrm{s}(\mathrm{z}), \mathrm{z}) \rightarrow_{R_2} \mathrm{s}(\mathrm{add}(\mathrm{z}, \mathrm{z})) \rightarrow_{R_1} \mathrm{s}(\mathrm{z})$

With **narrowing**: $\mathrm{add}(\mathrm{s}(\mathrm{z}), \mathrm{z}) \leadsto_{R_2} \mathrm{s}(\mathrm{add}(\mathrm{z}, \mathrm{z})) \leadsto_{R_1} \mathrm{s}(\mathrm{z})$

but also:

$$
\begin{tikzcd}
\mathrm{add}(x, \mathrm{z}) \arrow[dr, "\text{“guess”}"'] & & \mathrm{s}(\mathrm{add}(y, \mathrm{z})) \arrow[dr, "\text{“guess”}"'] & & \mathrm{s}(\mathrm{z}) \\
& \mathrm{add}(\mathrm{s}(y), \mathrm{z}) \arrow[ur, "R_2"'] & & \mathrm{s}(\mathrm{add}(\mathrm{z}, \mathrm{z})) \arrow[ur, "R_1"'] &
\end{tikzcd}
$$

(many other non-deterministic reductions possible. . .)

What is narrowing?

Standard definition of addition (TRS)

$$\begin{array}{rcll} \mathsf{add}(\mathsf{z}, y) & \to & y & (R_1) \\ \mathsf{add}(\mathsf{s}(x), y) & \to & \mathsf{s}(\mathsf{add}(x, y)) & (R_2) \end{array}$$

With **rewriting**: $\mathsf{add}(\mathsf{s}(\mathsf{z}), \mathsf{z}) \to_{R_2} \mathsf{s}(\mathsf{add}(\mathsf{z}, \mathsf{z})) \to_{R_1} \mathsf{s}(\mathsf{z})$

With **narrowing**: $\mathsf{add}(\mathsf{s}(\mathsf{z}), \mathsf{z}) \leadsto_{R_2} \mathsf{s}(\mathsf{add}(\mathsf{z}, \mathsf{z})) \leadsto_{R_1} \mathsf{s}(\mathsf{z})$

but also:

$$\begin{tikzcd} \mathsf{add}(x, \mathsf{z}) \arrow[dr, "\text{``guess''}"'] & & \mathsf{s}(\mathsf{add}(y, \mathsf{z})) \arrow[dr, "\text{``guess''}"'] & & \mathsf{s}(\mathsf{z}) \\ & \mathsf{add}(\mathsf{s}(y), \mathsf{z}) \arrow[ur, "R_2"'] & & \mathsf{s}(\mathsf{add}(\mathsf{z}, \mathsf{z})) \arrow[ur, "R_1"'] & \end{tikzcd}$$

(many other non-deterministic reductions possible. . .)

What is narrowing?

Standard definition of addition (TRS)

$$
\begin{array}{rcll}
\mathsf{add}(\mathsf{z}, y) & \rightarrow & y & (R_1) \\
\mathsf{add}(\mathsf{s}(x), y) & \rightarrow & \mathsf{s}(\mathsf{add}(x, y)) & (R_2)
\end{array}
$$

With **rewriting**: $\mathsf{add}(\mathsf{s}(\mathsf{z}), \mathsf{z}) \rightarrow_{R_2} \mathsf{s}(\mathsf{add}(\mathsf{z}, \mathsf{z})) \rightarrow_{R_1} \mathsf{s}(\mathsf{z})$

With **narrowing**: $\mathsf{add}(\mathsf{s}(\mathsf{z}), \mathsf{z}) \rightsquigarrow_{R_2} \mathsf{s}(\mathsf{add}(\mathsf{z}, \mathsf{z})) \rightsquigarrow_{R_1} \mathsf{s}(\mathsf{z})$

but also:

$$
\begin{tikzcd}
\mathsf{add}(x, \mathsf{z}) \arrow[dr, "\text{``guess''}"'] & & \mathsf{s}(\mathsf{add}(y, \mathsf{z})) \arrow[dr, "\text{``guess''}"'] & & \mathsf{s}(\mathsf{z}) \\
& \mathsf{add}(\mathsf{s}(y), \mathsf{z}) \arrow[ur, "R_2"'] & & \mathsf{s}(\mathsf{add}(\mathsf{z}, \mathsf{z})) \arrow[ur, "R_1"'] &
\end{tikzcd}
$$

(many other non-deterministic reductions possible. . .)

What is narrowing?

Standard definition of addition (TRS)

$$\begin{array}{rcll} \mathsf{add}(\mathsf{z}, y) & \to & y & (R_1) \\ \mathsf{add}(\mathsf{s}(x), y) & \to & \mathsf{s}(\mathsf{add}(x, y)) & (R_2) \end{array}$$

With **rewriting**: $\mathsf{add}(\mathsf{s}(\mathsf{z}), \mathsf{z}) \to_{R_2} \mathsf{s}(\mathsf{add}(\mathsf{z}, \mathsf{z})) \to_{R_1} \mathsf{s}(\mathsf{z})$

With **narrowing**: $\mathsf{add}(\mathsf{s}(\mathsf{z}), \mathsf{z}) \leadsto_{R_2} \mathsf{s}(\mathsf{add}(\mathsf{z}, \mathsf{z})) \leadsto_{R_1} \mathsf{s}(\mathsf{z})$

but also: $\mathsf{add}(x, \mathsf{z}) \leadsto_{R_2, \{x \mapsto \mathsf{s}(y)\}} \mathsf{s}(\mathsf{add}(y, \mathsf{z})) \leadsto_{\{R_1, y \mapsto \mathsf{z}\}} \mathsf{s}(\mathsf{z})$

$$\begin{tikzcd}
\mathsf{add}(x, \mathsf{z}) \arrow[dr, "\text{``guess''}"'] & & \mathsf{s}(\mathsf{add}(y, \mathsf{z})) \arrow[dr, "\text{``guess''}"'] & & \mathsf{s}(\mathsf{z}) \\
& \mathsf{add}(\mathsf{s}(y), \mathsf{z}) \arrow[ur, "R_2"'] & & \mathsf{s}(\mathsf{add}(\mathsf{z}, \mathsf{z})) \arrow[ur, "R_1"'] &
\end{tikzcd}$$

(many other non-deterministic reductions possible. . .)

Formal definition

Definition (rewriting)

$\boxed{s \to_{p,R} s[r\sigma]_p}$ if there are $\begin{cases} \bullet \text{ a position } p \text{ of } s \\ \bullet \text{ a rule } R = (l \to r) \text{ in } \mathcal{R} \\ \bullet \text{ a substitution } \sigma \text{ such that } s|_p = l\sigma \end{cases}$

$\Downarrow$

Definition (narrowing)

$\boxed{s \leadsto_{p,R,\sigma} (s[r]_p)\sigma}$ if there are $\begin{cases} \bullet \text{ a nonvariable position } p \text{ of } s \\ \bullet \text{ a variant } R = (l \to r) \text{ of a rule in } \mathcal{R} \\ \bullet \text{ a substitution } \sigma \text{ such that } s|_p\sigma = l\sigma \\ \quad [\sigma = \mathrm{mgu}(s|_p, l)] \end{cases}$

Some motivation

We want to analyze the termination of narrowing

Why?

- narrowing is relevant in a number of areas: functional logic languages, partial evaluation, protocol verification, type inference, etc
- no termination prover for narrowing

We want to analyze the termination of narrowing by analyzing the termination of rewriting

Why?

- many techniques and tools for rewriting!

Main ideas

- replace logic variables by data generators
- analyze the termination of rewriting with data generators
- adapt direct and transformational approaches

Some motivation

We want to analyze the termination of narrowing

Why?

- narrowing is relevant in a number of areas: functional logic languages, partial evaluation, protocol verification, type inference, etc
- no termination prover for narrowing

We want to analyze the termination of narrowing by analyzing the termination of rewriting

Why?

- many techniques and tools for rewriting!

Main ideas

- replace logic variables by data generators
- analyze the termination of rewriting with data generators
- adapt direct and transformational approaches

Some motivation

We want to analyze the termination of narrowing

Why?

- narrowing is relevant in a number of areas: functional logic languages, partial evaluation, protocol verification, type inference, etc
- no termination prover for narrowing

We want to analyze the termination of narrowing by analyzing the termination of rewriting

Why?

- many techniques and tools for rewriting!

Main ideas

- replace logic variables by data generators
- analyze the termination of rewriting with data generators
- adapt direct and transformational approaches

Termination of narrowing

The termination problem

- given a TRS, are all possible narrowing derivations finite?

Too strong!

$$\mathsf{add}(x,y) \leadsto_{R_2,\{x\mapsto \mathsf{s}(x')\}} \mathsf{add}(x',y) \leadsto_{R_2,\{x'\mapsto \mathsf{s}(x'')\}} \cdots$$

In this work

- given a TRS $\mathcal{R}$ and a set of terms T, are all possible narrowing derivations $t_1 \leadsto t_2 \leadsto \ldots$ for $t_1 \in T$ finite? (in symbols: T is $\leadsto_{\mathcal{R}}$-terminating)

For instance, $\{\ \mathsf{add}(s,t) \mid s \text{ is ground}\ \}$ is $\leadsto_{\mathcal{R}}$-terminating

Termination of narrowing

The termination problem

- given a TRS, are all possible narrowing derivations finite?

Too strong!

$$\mathsf{add}(x, y) \leadsto_{R_2,\{x \mapsto \mathsf{s}(x')\}} \mathsf{add}(x', y) \leadsto_{R_2,\{x' \mapsto \mathsf{s}(x'')\}} \cdots$$

In this work

- given a TRS $\mathcal{R}$ and a set of terms T, are all possible narrowing derivations $t_1 \leadsto t_2 \leadsto \ldots$ for $t_1 \in T$ finite? (in symbols: T is $\leadsto_{\mathcal{R}}$-terminating)

For instance, $\{\ \mathsf{add}(s, t) \mid s \text{ is ground}\ \}$ is $\leadsto_{\mathcal{R}}$-terminating

Termination of narrowing

The termination problem

- given a TRS, are all possible narrowing derivations finite?

Too strong!

$$\mathsf{add}(x,y) \leadsto_{R_2,\{x \mapsto \mathsf{s}(x')\}} \mathsf{add}(x',y) \leadsto_{R_2,\{x' \mapsto \mathsf{s}(x'')\}} \cdots$$

In this work

- given a TRS $\mathcal{R}$ and a set of terms T,
  are all possible narrowing derivations $t_1 \leadsto t_2 \leadsto \ldots$ for $t_1 \in T$ finite?
  (in symbols: T is $\leadsto_{\mathcal{R}}$-terminating)

For instance, $\{ \; \mathsf{add}(s,t) \mid s \text{ is ground } \}$ is $\leadsto_{\mathcal{R}}$-terminating

Termination of narrowing via termination of rewriting

We consider **left-linear** **constructor** TRSs:

$$\begin{array}{rcl} f_1(t_{11},\ldots,t_{1m_1}) & \to & r_1 \\ & \ldots & \\ f_n(t_{n1},\ldots,t_{nm_n}) & \to & r_n \end{array}$$

with

- $f_i(t_{i1},\ldots,t_{in_i})$ linear (no multiple occurrences of the same variable)
- $t_{i1},\ldots,t_{in_i}$ constructor terms (no occurrence of $f_1,\ldots,f_n$)

$\boxed{\textbf{Property}}$ variables are bound to (irreducible) constructor terms

$\Downarrow$

$\boxed{\textbf{Our approach}}$ we can replace variables by "data generators" that only produce constructor terms

Data generators [Antoy, Hanus, 2006; de Dios-Castro, López-Fraguas 2006]

For every TRS $\mathcal{R}$, we define $\mathcal{R}_{\mathsf{gen}}$ as $\mathcal{R}$ augmented with

$$\boxed{\mathsf{gen} \quad \to \quad \mathsf{c}(\overbrace{\mathsf{gen},\ldots,\mathsf{gen}}^{n \ \textit{times}})}$$

for all constructor $\mathsf{c}/n \in \mathcal{C},\ \ n \geqslant 0$

E.g., for $\mathcal{C} = \{\mathsf{z}/0, \mathsf{s}/1\}$, we have

$$\mathcal{R}_{\mathsf{gen}} = \mathcal{R} \cup \left\{ \begin{array}{lcl} \mathsf{gen} & \to & \mathsf{z} \\ \mathsf{gen} & \to & \mathsf{s}(\mathsf{gen}) \end{array} \right\}$$

Some notation: $\widehat{t} = t\sigma$, with $\sigma = \{x \mapsto \mathsf{gen} \mid x \in \mathcal{V}ar(t)\}$

Theorem (correctness)

$$t \leadsto_{\mathcal{R}} \ldots \leadsto_{\mathcal{R}} t' \quad \textit{iff} \quad \widehat{t} \to_{\mathcal{R}_{\mathsf{gen}}} \ldots \to_{\mathcal{R}_{\mathsf{gen}}} \widehat{t'}$$

Data generators [Antoy, Hanus, 2006; de Dios-Castro, López-Fraguas 2006]

For every TRS $\mathcal{R}$, we define $\mathcal{R}_{\mathsf{gen}}$ as $\mathcal{R}$ augmented with

$$\boxed{\mathsf{gen} \quad \rightarrow \quad \mathsf{c}(\overbrace{\mathsf{gen},\ldots,\mathsf{gen}}^{n \ \textit{times}})}$$

for all constructor $\mathsf{c}/n \in \mathcal{C}$, $n \geqslant 0$

E.g., for $\mathcal{C} = \{\mathsf{z}/0, \mathsf{s}/1\}$, we have

$$\mathcal{R}_{\mathsf{gen}} = \mathcal{R} \cup \left\{ \begin{array}{lcl} \mathsf{gen} & \rightarrow & \mathsf{z} \\ \mathsf{gen} & \rightarrow & \mathsf{s}(\mathsf{gen}) \end{array} \right\}$$

Some notation: $\widehat{t} = t\sigma$, with $\sigma = \{x \mapsto \mathsf{gen} \mid x \in \mathcal{V}\mathsf{ar}(t)\}$

Theorem (correctness)

$$t \leadsto_{\mathcal{R}} \ldots \leadsto_{\mathcal{R}} t' \quad \textit{iff} \quad \widehat{t} \rightarrow_{\mathcal{R}_{\mathsf{gen}}} \ldots \rightarrow_{\mathcal{R}_{\mathsf{gen}}} \widehat{t'}$$

What about termination in $\mathcal{R}_{\text{gen}}$?

Clearly, no term with occurrences of gen terminates!

Fortunately, **relative termination** of $\mathcal{R}_{\text{gen}}$ suffices:

- T is relatively $\mathcal{R}_{\text{gen}}$-terminating to $\mathcal{R}$ if every derivation $t_1 \to t_2 \ldots$ for $t_1 \in T$ contains finitely many $\to_{\mathcal{R}}$ steps

Theorem (termination of narrowing via termination of rewriting)

Let $\mathcal{R}$ be a left-linear constructor TRS
T is $\leadsto_{\mathcal{R}}$-terminating
iff
$\widehat{T}$ is relatively $\to_{\mathcal{R}_{\text{gen}}}$-terminating to $\mathcal{R}$

What about termination in $\mathcal{R}_{gen}$?

Clearly, no term with occurrences of gen terminates!

Fortunately, **relative termination** of $\mathcal{R}_{gen}$ suffices:

- T is relatively $\mathcal{R}_{gen}$-terminating to $\mathcal{R}$ if every derivation $t_1 \to t_2 \ldots$ for $t_1 \in T$ contains finitely many $\to_{\mathcal{R}}$ steps

Theorem (termination of narrowing via termination of rewriting)

Let $\mathcal{R}$ be a left-linear constructor TRS
T is $\leadsto_{\mathcal{R}}$-terminating
iff
$\widehat{T}$ is relatively $\to_{\mathcal{R}_{gen}}$-terminating to $\mathcal{R}$

What about termination in $\mathcal{R}_{\text{gen}}$?

Clearly, no term with occurrences of gen terminates!

Fortunately, **relative termination** of $\mathcal{R}_{\text{gen}}$ suffices:

- T is relatively $\mathcal{R}_{\text{gen}}$-terminating to $\mathcal{R}$ if every derivation $t_1 \to t_2 \ldots$ for $t_1 \in T$ contains finitely many $\to_{\mathcal{R}}$ steps

Theorem (termination of narrowing via termination of rewriting)

Let $\mathcal{R}$ be a left-linear constructor TRS
T is $\leadsto_{\mathcal{R}}$-terminating
iff
$\widehat{T}$ is relatively $\to_{\mathcal{R}_{\text{gen}}}$-terminating to $\mathcal{R}$

Proving termination automatically

The problem

Given $\mathcal{R}$ and T,

T is $\leadsto_{\mathcal{R}}$-terminating if $\widehat{T}$ is relatively $\rightarrow_{\mathcal{R}_{\text{gen}}}$-terminating to $\mathcal{R}$

Drawback

- the set T is generally infinite

Solution: use abstract terms

- similar to modes in logic programming
- E.g., $\text{add}(g, v)$ denotes the set of terms $\text{add}(t_1, t_2)$ with
  - t_1 (definitely) **g**round
  - t_2 (possibly) **v**ariable
- concretization funcion γ,

  e.g., $\gamma(\text{add}(g, v)) = \{\text{add}(z, x), \text{add}(z, z), \text{add}(s(z), x), \text{add}(s(z), z), \ldots\}$

Proving termination automatically

The problem

Given $\mathcal{R}$ and T,

T is $\leadsto_{\mathcal{R}}$-terminating if $\widehat{T}$ is relatively $\rightarrow_{\mathcal{R}_{\text{gen}}}$-terminating to $\mathcal{R}$

Drawback

- the set T is generally infinite

Solution: use abstract terms

- similar to modes in logic programming
- E.g., $\text{add}(g, v)$ denotes the set of terms $\text{add}(t_1, t_2)$ with
  - t_1 (definitely) **g**round
  - t_2 (possibly) **v**ariable
- concretization funcion γ,

  e.g., $\gamma(\text{add}(g, v)) = \{\text{add}(z, x), \text{add}(z, z), \text{add}(s(z), x), \text{add}(s(z), z), \ldots\}$

Proving termination automatically

The problem

Given $\mathcal{R}$ and T,

T is $\leadsto_{\mathcal{R}}$-terminating if $\widehat{T}$ is relatively $\rightarrow_{\mathcal{R}_{\mathrm{gen}}}$-terminating to $\mathcal{R}$

Drawback

- the set T is generally infinite

Solution: use abstract terms

- similar to modes in logic programming
- E.g., $\mathsf{add}(g, v)$ denotes the set of terms $\mathsf{add}(t_1, t_2)$ with
  - t_1 (definitely) **g**round
  - t_2 (possibly) **v**ariable
- concretization funcion γ,

  e.g., $\gamma(\mathsf{add}(g, v)) = \{\mathsf{add}(\mathsf{z}, x), \mathsf{add}(\mathsf{z}, \mathsf{z}), \mathsf{add}(\mathsf{s}(\mathsf{z}), x), \mathsf{add}(\mathsf{s}(\mathsf{z}), \mathsf{z}), \ldots\}$

Proving termination automatically

The problem

Given $\mathcal{R}$ and t^α,

$\gamma(t^\alpha)$ is $\leadsto_\mathcal{R}$-terminating if $\widehat{\gamma(t^\alpha)}$ is relatively $\rightarrow_{\mathcal{R}_{\mathsf{gen}}}$-terminating to $\mathcal{R}$

Drawback

- checking relative termination requires non-standard techniques

Solution: use argument filterings

- to filter away non-ground arguments of terms
  (equivalently, to filter away occurrences of gen)

Proving termination automatically

The problem

Given $\mathcal{R}$ and t^α,

$\gamma(t^\alpha)$ is $\leadsto_{\mathcal{R}}$-terminating if $\widehat{\gamma(t^\alpha)}$ is relatively $\rightarrow_{\mathcal{R}_{\mathsf{gen}}}$-terminating to $\mathcal{R}$

Drawback

- checking relative termination requires non-standard techniques

Solution: use argument filterings

- to filter away non-ground arguments of terms
  (equivalently, to filter away occurrences of gen)

Proving termination automatically

The problem

Given $\mathcal{R}$ and t^α,

$\gamma(t^\alpha)$ is $\rightsquigarrow_{\mathcal{R}}$-terminating if $\widehat{\gamma(t^\alpha)}$ is relatively $\rightarrow_{\mathcal{R}_{\mathsf{gen}}}$-terminating to $\mathcal{R}$

Drawback

- checking relative termination requires non-standard techniques

Solution: use argument filterings

- to filter away non-ground arguments of terms
  (equivalently, to filter away occurrences of gen)

Argument filterings [Kusakari, Nakamura, Toyama 1999]

$\boxed{\pi(\mathsf{f}) \subseteq \{1, \ldots, n\} \text{ for every defined function } \mathsf{f}/n}$

Argument filterings over terms:

$$\pi(t) = \begin{cases} x & \text{if } t = x \\ \mathsf{c}(\pi(t_1), \ldots, \pi(t_n)) & \text{if } t = \mathsf{c}(t_1, \ldots, t_n) \\ \mathsf{f}(\pi(t_{i_1}), \ldots, \pi(t_{i_m})) & \text{if } t = \mathsf{f}(t_1, \ldots, t_n) \text{ and } \pi(\mathsf{f}) = \{i_1, \ldots, i_m\} \end{cases}$$

From t^α we infer a safe argument filtering π for t^α

- $\pi(t^\alpha) = \mathsf{f}(g, g, \ldots, g)$
- for all $s \rightsquigarrow t$, if $\pi(s|_p)$ are ground then $\pi(t|_q)$ are ground too

Argument filterings [Kusakari, Nakamura, Toyama 1999]

$\pi(\mathsf{f}) \subseteq \{1, \ldots, n\}$ for every defined function f/n

Argument filterings over terms:

$$\pi(t) = \begin{cases} x & \text{if } t = x \\ \mathsf{c}(\pi(t_1), \ldots, \pi(t_n)) & \text{if } t = \mathsf{c}(t_1, \ldots, t_n) \\ \mathsf{f}(\pi(t_{i_1}), \ldots, \pi(t_{i_m})) & \text{if } t = \mathsf{f}(t_1, \ldots, t_n) \text{ and } \pi(\mathsf{f}) = \{i_1, \ldots, i_m\} \end{cases}$$

From t^α we infer a safe argument filtering π for t^α

- $\pi(t^\alpha) = \mathsf{f}(g, g, \ldots, g)$
- for all $s \rightsquigarrow t$, if $\pi(s|_p)$ are ground then $\pi(t|_q)$ are ground too

Proving termination automatically: approaches

A direct approach

- based on dependency pairs [Arts, Giesl 2000]
- only a slight extension needed

A transformational approach

- based on argument filtering transformation [Kusakari, Nakamura, Toyama 1999]
- no significant extension required

Dependency pairs approach: differences

Definition (chain)

A (possibly infinite) sequence of dependency pairs $s_1 \to t_1$, $s_2 \to t_2, \ldots$ from $DP(\mathcal{R})$ is a $(DP(\mathcal{R}), \mathcal{R}, \pi)$-chain if

- $\exists$ (constructor) substitution σ such that $\widehat{t_i\sigma} \to^*_{\mathcal{R}_{\text{gen}}} \widehat{s_{i+1}\sigma}$ for $i \geqslant 1$
- $\pi(\widehat{s_i\sigma}), \pi(\widehat{t_i\sigma})$ contain no occurrences of gen

Three main extensions w.r.t. the standard notion:

- it is parameterized by π
- variables are replaced by gen and reductions w.r.t. $\mathcal{R}_{\text{gen}}$
- π should filter away all occurrences of gen

Theorem

Let π be a safe argument filtering for t^α in $\mathcal{R}$
If there is no infinite $(DP(\mathcal{R}), \mathcal{R}, \pi)$-chain, then $\gamma(t^\alpha)$ is $\leadsto_{\mathcal{R}}$-terminating

Dependency pairs approach: differences

Definition (chain)

A (possibly infinite) sequence of dependency pairs $s_1 \to t_1$, $s_2 \to t_2, \ldots$ from $DP(\mathcal{R})$ is a $(DP(\mathcal{R}), \mathcal{R}, \pi)$-chain if

- $\exists$ (constructor) substitution σ such that $\widehat{t_i\sigma} \to^*_{\mathcal{R}_{\mathsf{gen}}} \widehat{s_{i+1}\sigma}$ for $i \geqslant 1$
- $\pi(\widehat{s_i\sigma}), \pi(\widehat{t_i\sigma})$ contain no occurrences of gen

Three main extensions w.r.t. the standard notion:

- it is parameterized by π
- variables are replaced by gen and reductions w.r.t. $\mathcal{R}_{\mathsf{gen}}$
- π should filter away all occurrences of gen

Theorem

Let π be a safe argument filtering for t^α in $\mathcal{R}$
If there is no infinite $(DP(\mathcal{R}), \mathcal{R}, \pi)$-chain, then $\gamma(t^\alpha)$ is $\leadsto_{\mathcal{R}}$-terminating

Dependency pairs approach: differences

Definition (chain)

A (possibly infinite) sequence of dependency pairs $s_1 \to t_1$, $s_2 \to t_2, \ldots$ from $DP(\mathcal{R})$ is a $(DP(\mathcal{R}), \mathcal{R}, \pi)$-chain if

- $\exists$ (constructor) substitution σ such that $\widehat{t_i\sigma} \to^*_{\mathcal{R}_{\mathsf{gen}}} \widehat{s_{i+1}\sigma}$ for $i \geqslant 1$
- $\pi(\widehat{s_i\sigma}), \pi(\widehat{t_i\sigma})$ contain no occurrences of gen

Three main extensions w.r.t. the standard notion:

- it is parameterized by π
- variables are replaced by gen and reductions w.r.t. $\mathcal{R}_{\mathsf{gen}}$
- π should filter away all occurrences of gen

Theorem

Let π be a safe argument filtering for t^α in $\mathcal{R}$
If there is no infinite $(DP(\mathcal{R}), \mathcal{R}, \pi)$-chain, then $\gamma(t^\alpha)$ is $\leadsto_{\mathcal{R}}$-terminating

A transformational approach

Our aim

- transform the original TRS $\mathcal{R}$ into a new TRS $\mathcal{R}'$
- narrowing terminates in $\mathcal{R}$ if rewriting terminates in $\mathcal{R}'$

Our transformation is a simplification of the *argument filtering transformation* (AFT) of [Kusakari, Nakamura, Toyama 1999]

Theorem

Let π be a safe argument filtering for t^α in $\mathcal{R}$
$\gamma(t^\alpha)$ *is* $\leadsto_{\mathcal{R}}$*-terminating if* $\mathsf{AFT}_\pi(\mathcal{R})$ *is terminating*

- $\mathsf{AFT}_\pi(\mathcal{R})$ can be analyzed using standard techniques and tools for proving the termination of TRSs

  (no data generator is involved in the derivations of $\mathsf{AFT}_\pi(\mathcal{R})$)

A transformational approach

Our aim

- transform the original TRS $\mathcal{R}$ into a new TRS $\mathcal{R}'$
- narrowing terminates in $\mathcal{R}$ if rewriting terminates in $\mathcal{R}'$

Our transformation is a simplification of the *argument filtering transformation* (AFT) of [Kusakari, Nakamura, Toyama 1999]

Theorem

Let π be a safe argument filtering for t^α in $\mathcal{R}$
$\gamma(t^\alpha)$ is $\leadsto_{\mathcal{R}}$-terminating if $\mathsf{AFT}_\pi(\mathcal{R})$ *is terminating*

- $\mathsf{AFT}_\pi(\mathcal{R})$ can be analyzed using standard techniques and tools for proving the termination of TRSs

  (no data generator is involved in the derivations of $\mathsf{AFT}_\pi(\mathcal{R})$)

Example

$$\begin{aligned} \mathsf{append}(\mathsf{nil}, y) &\to y \\ \mathsf{append}(\mathsf{cons}(x, xs), y) &\to \mathsf{cons}(x, \mathsf{append}(xs, y)) \\ \mathsf{reverse}(\mathsf{nil}) &\to \mathsf{nil} \\ \mathsf{reverse}(\mathsf{cons}(x, xs)) &\to \mathsf{append}(\mathsf{reverse}(xs), \mathsf{cons}(x, \mathsf{nil})) \end{aligned}$$

$t^\alpha = \mathsf{append}(g, v)$
$\pi = \{\mathsf{append} \mapsto \{1\},\ \mathsf{reverse} \mapsto \{1\}\}$ (π is safe for t^α)

The transformation $\mathsf{AFT}_\pi(\mathcal{R})$ returns

$$\begin{aligned} \mathsf{append}(\mathsf{nil}) &\to y \qquad\qquad (y \text{ is an extra variable}) \\ \mathsf{append}(\mathsf{cons}(x, xs)) &\to \mathsf{cons}(x, \mathsf{append}(xs)) \\ \mathsf{reverse}(\mathsf{nil}) &\to \mathsf{nil} \\ \mathsf{reverse}(\mathsf{cons}(x, xs)) &\to \mathsf{append}(\mathsf{reverse}(xs)) \end{aligned}$$

which is clearly **not** terminating

Example

$$\begin{aligned} \mathsf{append}(\mathsf{nil}, y) &\to y \\ \mathsf{append}(\mathsf{cons}(x, xs), y) &\to \mathsf{cons}(x, \mathsf{append}(xs, y)) \\ \mathsf{reverse}(\mathsf{nil}) &\to \mathsf{nil} \\ \mathsf{reverse}(\mathsf{cons}(x, xs)) &\to \mathsf{append}(\mathsf{reverse}(xs), \mathsf{cons}(x, \mathsf{nil})) \end{aligned}$$

$t^\alpha = \mathsf{append}(g, v)$
$\pi = \{\mathsf{append} \mapsto \{1\},\ \mathsf{reverse} \mapsto \{1\}\}$ (π is safe for t^α)

The transformation $\mathsf{AFT}_\pi(\mathcal{R})$ returns

$$\begin{aligned} \mathsf{append}(\mathsf{nil}) &\to \cancel{y}\ \ \bot && (\bot \text{ is a fresh constant}) \\ \mathsf{append}(\mathsf{cons}(x, xs)) &\to \mathsf{cons}(x, \mathsf{append}(xs)) \\ \mathsf{reverse}(\mathsf{nil}) &\to \mathsf{nil} \\ \mathsf{reverse}(\mathsf{cons}(x, xs)) &\to \mathsf{append}(\mathsf{reverse}(xs)) \end{aligned}$$

which is clearly not terminating

The termination tool TNT

It takes as input

- a left-linear constructor TRS
- an abstract term

and proceeds as follows:

- infers a safe argument filtering for the abstract term (a binding-time analysis)
- returns a transformed TRS using AFT_π

Website: http://german.dsic.upv.es/filtering.html

The termination of the transformed TRS can be checked with APROVE

Conclusions

Conclusions

- new techniques for proving the termination of narrowing in left-linear constructor systems
- good potential for reusing existing techniques and tools for rewriting
- first tool for proving the termination of narrowing

Future work

- extension to deal with extra-variables
- application to (offline) partial evaluation

Related work

Schneider-Kamp *et al* [SKGST07] presented an automated termination analysis for logic programs:

- logic programs are first translated to TRSs
- logic variables are simulated by infinite terms

Main differences:

- data generators (reuse of results relating narrowing and rewriting)
- no transformational approach in [SKGST07]

Nishida and Miura [NM06] adapted the dependency pair method for proving the termination of narrowing:

- direct approach (not based on using generators & rewriting)
- allow extra variables in TRSs
- not comparable

S. Antoy and M. Hanus.
Compiling Multi-Paradigm Declarative Programs into Prolog.
In *Proc. of the Int'l Workshop on Frontiers of Combining Systems (FroCoS'2000)*, pages 171–185. Springer LNCS 1794, 2000.

S. Antoy and M. Hanus.
Overlapping Rules and Logic Variables in Functional Logic Programs.
In *Proc. of the 22snd Int'l Conf. on Logic Programming (ICLP'06)*, pages 87–101. Springer LNCS 4079, 2006.

G. Arroyo, J.G. Ramos, J. Silva, and G. Vidal.
Improving Offline Narrowing-Driven Partial Evaluation using Size-Change Graphs.
In *Proc. of LOPSTR'06*, pages 60–76. Springer LNCS 4407, 2007.

J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke.
Mechanizing and Improving Dependency Pairs.
Journal of Automated Reasoning, 37(3):155–203, 2006.

K. Kusakari, M. Nakamura, and Y. Toyama.

Argument Filtering Transformation.
In *Proc. of PPDP'99*, pages 48–62. Springer LNCS 1702, 1999.

N. Nishida and K. Miura.
Dependency Graph Method for Proving Termination of Narrowing.
In *Proc. of WST'06*, pages 12–16, 2006.

J.G. Ramos, J. Silva, and G. Vidal.
Fast Narrowing-Driven Partial Evaluation for Inductively Sequential Systems.
In *Proc. of the 10th ACM SIGPLAN Int'l Conf. on Functional Programming (ICFP'05)*, pages 228–239. ACM Press, 2005.

P. Schneider-Kamp, J. Giesl, A. Serebrenik, and R. Thiemann.
Automated Termination Analysis for Logic Programs by Term Rewriting.
In *Proc. of LOPSTR'06*, pages 177–193. Springer LNCS 4407, 2007.