Semantic Labelling for Proving Termination of Combinatory Reduction Systems

Makoto Hamana

Department of Computer Science,
Gunma University, Japan

WFLP'09
June, 2009.

Intro: Termination Proof by RPO

Term Rewriting System (TRS) $\mathcal{R}$:

$$\boldsymbol{fact(0) \longrightarrow s(0)}$$
$$\boldsymbol{fact(s(x)) \longrightarrow fact(x) * s(x)}$$

Recursive path ordering (RPO) [Dershowitz TCS'82]
proves termination (= SN)

Intro: Termination Proof by RPO

Term Rewriting System (TRS) $\mathcal{R}$:

$$\boldsymbol{fact(0) \to s(0)}$$
$$\boldsymbol{fact(s(x)) \to fact(x) * s(x)}$$

Proof method:

1. Give a precedence:

   $$\boldsymbol{fact > * > s > 0}$$

   Thm. RPO $>_{\mathbf{RPO}}$ well-founded & closed under subst., contexts

2. Check for each $\boldsymbol{l \to r \in \mathcal{R}}$, $\boldsymbol{l >_{\mathbf{RPO}} r}$
   (i.e. $\to \subseteq >_{\mathbf{RPO}}$)
   Then: $\to_{\mathcal{R}} \subseteq >_{\mathbf{RPO}}$

Hence $\mathcal{R}$ is SN.

Intro: Semantic Labelling for TRSs [Zantema'95]

Original TRS $\mathcal{R}$:

$$\begin{aligned} fact(s(x)) &\to fact(p(s(x))) * s(x) \\ p(s(0)) &\to 0 \\ p(s(s(x))) &\to s(p(s(x))) \end{aligned}$$

- RPO doesn't work

Semantics: Σ-algebra $(\mathbb{N}, \{fact_{\mathbb{N}}, s_{\mathbb{N}}, p_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}, \cdots\})$

Labelled TRS $\overline{\mathcal{R}}$:

$$\begin{aligned} fact_{i+1}(s(x)) &\to fact_i(p(s(x))) * s(x) \\ p(s(0)) &\to 0 \\ p(s(s(x))) &\to s(p(s(x))) \end{aligned}$$

- RPO works! $fact_{i+1} > fact_i$

Th. [Zantema'95] TRS $\overline{\mathcal{R}}$ is terminating $\Rightarrow$ $\mathcal{R}$ is terminating.

This Work

- ▷ Extends semantic labelling to CRSs
- ▷ Klop's format of higher-order rewriting:
  Combinatory Reduction System (CRS) [Klop'80]
  - Second-order rewriting systems
  - TRS + variable binding, metavariables, meta-terms

Example of CRS: Quick Sort

$$
\begin{array}{rcl}
\mathsf{if}(\mathsf{true}, \mathrm{X}, \mathrm{Y}) & \longrightarrow & \mathrm{X} \\
\mathsf{if}(\mathsf{false}, \mathrm{X}, \mathrm{Y}) & \longrightarrow & \mathrm{Y} \\
\mathsf{filter}(\mathrm{P}, \mathsf{nil}) & \longrightarrow & \mathsf{nil} \\
\mathsf{filter}(\mathrm{P}, \mathrm{X} : \mathrm{XS}) & \longrightarrow & \mathsf{if}(\mathrm{P}[\mathrm{X}], \mathrm{X} : \mathsf{filter}(\mathrm{P}, \mathrm{XS}), \mathsf{filter}(\mathrm{P}, \mathrm{XS})) \\
\mathsf{qsort}(\mathsf{nil}) & \longrightarrow & \mathsf{nil} \\
\mathsf{qsort}(\mathrm{X} : \mathrm{XS}) & \longrightarrow & \mathsf{qsort}(\mathsf{filter}(\boldsymbol{a}.\ \boldsymbol{a} \leq \mathrm{X}, \mathrm{XS})) +\!\!+ ((\mathrm{X} : \mathsf{nil}) +\!\!+ \\
 & & \quad \mathsf{qsort}(\mathsf{filter}(\boldsymbol{a}.\ \boldsymbol{a} > \mathrm{X}, \mathrm{XS}))
\end{array}
$$

$$
\begin{array}{rcl}
\mathsf{nil} +\!\!+ \mathrm{XS} & \longrightarrow & \mathrm{XS} \\
(\mathrm{X} : \mathrm{XS}) +\!\!+ \mathrm{YS} & \longrightarrow & \mathrm{X} : (\mathrm{XS} +\!\!+ \mathrm{YS})
\end{array}
$$

This Work

- Difficulty: what is a suitable semantic structure for labelling CRSs?
- Contribution:
  1. Answer
  2. Second-order version of semantics labelling
  3. Applications to functional programs – so useful
- How to tackle?
  - TRSs
    - syntactic proof method: RPO
    - semantics: universal algebra
  - CRSs
    - syntactic proof method: the General Schema
    - semantics: ?

Termination Proof by the General Schema

The General Schema
[Blanqui,Jouannaud,Okada TCS'02, Blanqui RTA'00]
can be used as a higher-order version of RPO method

$\mathcal{R}$ higher-order rewrite rules (map, filter, etc.)

Proof method:

1. Give a precedence $>$ on fun. symbols.
2. Check for each $l \longrightarrow r \in \mathcal{R}$,
   r is in the computable closure (wrt. $>$) of l
   (the "General Schema").

Then $\mathcal{R}$ is SN.

Semantics Labelling for CRS: Theory

CRS's rewrite: $s \to_{\mathcal{R}} t$

Semantic labels must be consistent with:

- functional contexts

$$g_{[\![s]\!]}(\ \overline{s}\ ) \to_{\overline{\mathcal{R}}} g_{[\![t]\!]}(\ \overline{t}\ )$$

- binders

$$\lambda_{[\![s]\!]_x}(x.\ \overline{s}\ ) \to_{\overline{\mathcal{R}}} \lambda_{[\![t]\!]_x}(x.\ \overline{t}\ )$$

- function applications

$$\begin{array}{rcl} \mathsf{map}(x.F(x), \mathsf{cons}(M,N)) & \to_{\mathcal{R}} & \mathsf{cons}(F(M),\cdots) \\ & \Downarrow & \\ \cdots & \to_{\overline{\mathcal{R}}} & \mathsf{cons}_{[\![F]\!][\![M]\!]}(F(M),\cdots) \end{array}$$

- Σ-monoids [Fiore,Plotkin,Turi LICS'99] with order structure [Hamana RTA'05]

Semantics Labelling for CRS: Theory

- Signatures Σ, $\overline{\Sigma}$ (labelled)
- Interpretation by a Σ-monoid $(M, \geq)$

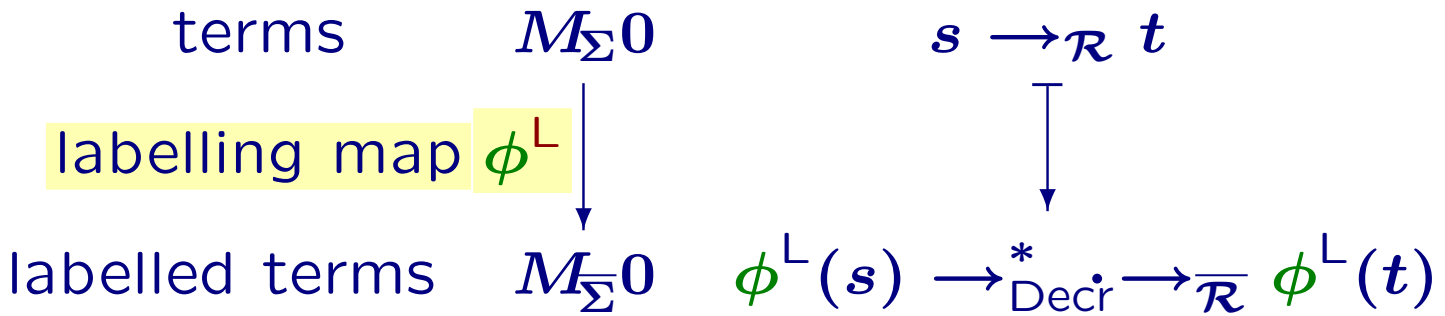
meta-terms	$M_\Sigma Z$	$l \longrightarrow r \in \mathcal{R}$	CRS
homomorphism ϕ	$\downarrow$	$\downarrow$	
quasi-model	M	$\phi(l) \geq \phi(r)$	

- Labelling

meta-terms	$M_\Sigma Z$	$l \longrightarrow r \in \mathcal{R}$	CRS
labelling map	ϕ^{L} $\downarrow$	$\downarrow$	
labelled meta-terms	$M_{\overline{\Sigma}} Z$	$\phi^{\mathsf{L}}(l) \longrightarrow \phi^{\mathsf{L}}(r) \in \overline{\mathcal{R}}$	labelled CRS

Semantics Labelling for CRS: Theory

▷ **Proposition**



[Proof]

- $M_{\underline{\Sigma}} 0$ forms a Σ-monoid
- commutativity of labelling map ϕ^{L} with meta and object substitution operations

▷ **Thm. [Semantic labelling for CRS]**
CRS $\overline{\mathcal{R}} \cup$ Decr is terminating $\Rightarrow$ $\mathcal{R}$ is terminating.

▷ "Decreasing rules" Decr, for all $f \in \Sigma$, labels $p > q$,

$$f_p(\mathrm{x}_1, \ldots, \mathrm{x}_l) \to f_q(\mathrm{x}_1, \ldots, \mathrm{x}_l)$$

Termination Proof by Semantic Labelling for CRS

CRS $\mathcal{R}$ given

Proof method

1. Give a quasi-model for $\mathcal{R}$: Σ-monoid $(\mathcal{M}, \geq)$
2. Generate labelled CRS $\overline{\mathcal{R}}$ by attaching semantic labels (wrt $\mathcal{M}$) to chosen fun. symbols
3. Give a precedence on semantically-labelled fun. symbols in $\overline{\mathcal{R}}$
4. Check that $\overline{\mathcal{R}} \cup$ Decr satisfies the General Schema

Then $\mathcal{R}$ is SN.

Example: Quick Sort

$$
\begin{aligned}
\mathsf{if}(\mathsf{true}, \mathrm{X}, \mathrm{Y}) &\longrightarrow \mathrm{X} & \mathsf{nil} +\!\!+ \mathrm{XS} &\longrightarrow \mathrm{XS} \\
\mathsf{if}(\mathsf{false}, \mathrm{X}, \mathrm{Y}) &\longrightarrow \mathrm{Y} & (\mathrm{X} : \mathrm{XS}) +\!\!+ \mathrm{YS} &\longrightarrow \mathrm{X} : (\mathrm{XS} +\!\!+ \mathrm{YS}) \\
\mathsf{filter}(\mathrm{P}, \mathsf{nil}) &\longrightarrow \mathsf{nil} & & \\
\mathsf{filter}(\mathrm{P}, \mathrm{X} : \mathrm{XS}) &\longrightarrow \mathsf{if}(\mathrm{P}[\mathrm{X}], \mathrm{X} : \mathsf{filter}(\mathrm{P}, \mathrm{XS}), \mathsf{filter}(\mathrm{P}, \mathrm{XS})) & & \\
\mathsf{qsort}(\mathsf{nil}) &\longrightarrow \mathsf{nil} & & \\
\mathsf{qsort}(\mathrm{X} : \mathrm{XS}) &\longrightarrow \mathsf{qsort}(\mathsf{filter}(\boldsymbol{a}.\ \boldsymbol{a} \leq \mathrm{X}, \mathrm{XS})) +\!\!+ ((\mathrm{X} : \mathsf{nil}) +\!\!+ & & \\
&\qquad\quad \mathsf{qsort}(\mathsf{filter}(\boldsymbol{a}.\ \boldsymbol{a} > \mathrm{X}, \mathrm{XS})) & &
\end{aligned}
$$

The General Schema doesn't satisfy $\boldsymbol{\mathcal{R}}$.

Problem:

$\mathrm{X} : \mathrm{XS} \prec$ arguments of recursive calls in the red parts

(i.e. structurally bigger).

Example: Quick Sort

Semantics $D = \mathbb{N} \times \mathbb{N}^*$ with the order

$$\langle n, l\rangle \geq \langle n', l'\rangle \overset{def}{\iff} n \geq n'$$

Carrier $\mathcal{M}_k \triangleq (D^k \to D)$

$$\mathsf{true}_{\mathcal{M}_0} = \langle 1, \epsilon\rangle \qquad \mathsf{false}_{\mathcal{M}_0} = \langle 0, \epsilon\rangle$$

$$0_{\mathcal{M}_0} = \langle 0, \epsilon\rangle \qquad \mathsf{s}_{\mathcal{M}_0}(\langle n, l\rangle) = \langle n+1, l\rangle$$

$$\mathsf{qsort}_{\mathcal{M}_0}(\langle n, l\rangle) = \langle n, \epsilon\rangle \qquad \mathsf{if}_{\mathcal{M}_0}(b, y, z) = \begin{cases} y & \text{if } b = \langle 1, \epsilon\rangle \\ z & \text{if } b = \langle 0, \epsilon\rangle \\ \langle 0, \epsilon\rangle & \text{otherwise} \end{cases}$$

$$\mathsf{filter}_{\mathcal{M}_0}(p, \langle n, l\rangle) = \langle \text{the number of } p(\langle i, \epsilon\rangle) = \langle 1, \epsilon\rangle \text{ for } i \text{ in } l, \epsilon\rangle$$

$$+\!+_{\mathcal{M}_0}(\langle n, l\rangle, \langle n', l'\rangle) = \langle n + n', l \cdot l'\rangle$$

$$\mathsf{nil}_{\mathcal{M}_0} = \langle 0, \epsilon\rangle \qquad :_{\mathcal{M}_0}(\langle a, s\rangle, \langle n, l\rangle) = \langle n+1, a \cdot l\rangle$$

This gives a quasi-model for $\mathcal{R}$.

Example: Quick Sort

Take semantic labels as the lengths $\boldsymbol{n}$ from $\langle \boldsymbol{n}, \boldsymbol{l} \rangle$.

We have the labelled rules $\overline{\boldsymbol{\mathcal{R}}}$:

$$
\begin{array}{rcl}
\mathsf{qsort}_{\mathbf{0}}(\mathsf{nil}) & \longrightarrow & \mathsf{nil} \\
\mathsf{qsort}_{i+1}(\mathrm{x} : \mathrm{xs}) & \longrightarrow & \mathsf{qsort}_{j}(\mathsf{filter}(\boldsymbol{a.a} \leq \mathrm{x}, \mathrm{xs})) +\!\!+ ((\mathrm{x} : \mathsf{nil}) +\!\!+ \\
 & & \mathsf{qsort}_{k}(\mathsf{filter}(\boldsymbol{a.a} > \mathrm{x}, \mathrm{xs}))) \quad \text{where } \boldsymbol{i+1 > j, k} \\
\mathsf{qsort}_{i}(\mathrm{xs}) & \longrightarrow & \mathsf{qsort}_{j}(\mathrm{xs}) \qquad \text{for all } \boldsymbol{i > j} \in \mathbb{N}
\end{array}
$$

With the precedence (for $\boldsymbol{i > j}$)

$$\mathsf{qsort}_i > \mathsf{qsort}_j > \mathsf{filter} > \mathsf{if}, +\!\!+, \text{“}>\text{”}, \text{“}\leq\text{”} > \mathsf{nil}, :, \mathsf{0}, \mathsf{S}, \mathsf{true}, \mathsf{false}$$

$\overline{\boldsymbol{\mathcal{R}}}$ satisfies the General Schema, hence $\boldsymbol{\mathcal{R}}$ is SN.

Summary

- ▷ Semantic labelling for CRS enhances the power of existing proof methods (the General Schema) for termination of CRS
- ▷ Semantic labelling for CRS is established uniformly in the framework of Σ-monoids
- ▷ Useful to prove termination of higher-order functional programs

Related Work

Hamana, Higher-Order Semantic Labelling for Inductive Datatype Systems, PPDP'07.

Roux and Blanqui, On the relation between size-based termination and semantic labelling, CSL'09.

Summary

- ▷ Semantic labelling for CRS enhances the power of existing proof methods (the General Schema) of termination of CRS
- ▷ Semantic labelling for CRS is established uniformly in the framework of $\mathbf{\Sigma}$-monoids
- ▷ Useful to prove termination of higher-order functional programs

Future Work

Practical termination checker
for proof assistances + functional programming (e.g. Agda, Coq)
by formalizing semantics and then using semantic labelling