Higher-Order Semantic Labelling for Inductive Datatype Systems

Makoto Hamana

Gunma University/University of Tokyo
Japan

PPDP'07
14th July, 2007.

Intro: Termination Proof by Syntactic Method

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

$$fact(s(x)) \to fact(x) * s(x)$$

▷ Recursive path ordering (RPO) [Dershowitz TCS'82] proves termination (= SN) by using the precedence

$$fact > * > s > 0$$

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!

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

This Work

- About higher-order term rewriting
- Inductive Data Type Systems (IDTSs)
  & Termination criteria: the General Schema
  [Blanqui,Jouannaud,Okada TCS'02, RTA'00]
- Difficulty: what is a suitable semantic structure for labelling higher-order rewriting systems?
- Contribution:
  i. Answer
  ii. Higher-order semantics labelling
  iii. Applications

Inductive Data Type Systems

[Blanqui,Jouannaud,Okada RTA'00, TCS'02]

Features:

- ▷ Rewrite rules on higher-order terms
- ▷ Simple types (up to 2nd-order in this work)
- ▷ Inductive types (by conditions of types of constructors)
- ▷ Metavariables with arities and substitutions, e.g.

$$\mathrm{ap}(\boldsymbol{\lambda}(\boldsymbol{x}.\boldsymbol{M}(\boldsymbol{x})), \boldsymbol{N}) \longrightarrow \boldsymbol{M}(\boldsymbol{N})$$

Idea: Attach Semantics of Arguments in Rewrite Rules

Original $\mathcal{R}$ $\qquad f(l) \;\longrightarrow g(\cdots f(t)\cdots)$

$$\Downarrow$$

Labelled $\overline{\mathcal{R}}$ $\qquad f_{[\![l]\!]\rho}(l') \;\longrightarrow g(\cdots f_{[\![t]\!]\rho}(t')\cdots)$

$(M, \geq)$: quasi-model $\quad (\forall (l \longrightarrow r) \in \mathcal{R}.\ [\![l]\!]\rho \geq [\![r]\!]\rho)$

$\rho : X \longrightarrow M$ $\quad$ valuation

$[\![-]\!] : T_\Sigma X \longrightarrow M$

What Kind of Semantics for Higher-Order Labelling?

- TRS: (1st-order) Universal algebra
- IDTS: ?? Higher-order version of universal algebra
- Semantics of Higher-order Rewrite Systems by van de Pol [HOA'93]: hereditary monotone functionals, but not complete for termination. The term model is not a model.
- Need to satisfy several requirements

What Kind of Semantics for Higher-Order Labelling?

$$s \longrightarrow_{\mathcal{R}} t$$

Semantic labels must reflect:

- contexts

$$g_{[\![s]\!]}(\ s'\ ) \longrightarrow_{\overline{\mathcal{R}}} g_{[\![t]\!]}(\ t'\ )$$

- binders

$$\lambda_{[\![s]\!]_x}(x.\ s'\ ) \longrightarrow_{\overline{\mathcal{R}}} \lambda_{[\![t]\!]_x}(x.\ t'\ )$$

- substitutions

$$\mathsf{map}(x.F(x), \mathsf{cons}(M, N)) \ \longrightarrow_{\mathcal{R}} \ \mathsf{cons}(F(M), \cdots)$$

$$\Downarrow$$

$$\cdots \ \longrightarrow_{\overline{\mathcal{R}}} \ \mathsf{cons}_{[\![F]\!][\![M]\!]}(F(M), \cdots)$$

What Kind of Semantics for Higher-Order Labelling?

- Models of $\boldsymbol{\lambda}$-calculus?
- But $\boldsymbol{\lambda}$-algebra doesn't satisfy $\boldsymbol{\xi}$-rule in general [Plotkin JSL'74]

$$\frac{\boldsymbol{M = N}}{\boldsymbol{\lambda x.M = \lambda x.N}}$$

- Right framework: binding algebras and $\boldsymbol{\Sigma}$-monoids [Fiore,Plotkin,Turi LICS'99] with order structure
- $\boldsymbol{\Sigma}$-monoid $=$ $\boldsymbol{\Sigma}$-algebra $+$ monoid
- Free $\boldsymbol{\Sigma}$-monoid $=$ higher-order syntax with metavariables [Hamana APLAS'04]
- Algebraic semantics of higher-order rewriting [Hamana RTA'05]
- Typed binding algebra [Fiore PPDP'02]

Semantic Labelling

- An assignment $\phi : Z \longrightarrow M$ into a quasi-model
- Labelling map $\phi^{\mathsf{L}} : M_{\Sigma} Z \longrightarrow M_{\overline{\Sigma}} Z$

$$\phi^{\mathsf{L}}(x) = x$$
$$\phi^{\mathsf{L}}(\mathrm{z}(\vec{t})) = \mathrm{z}(\phi^{\mathsf{L}}\vec{t})$$
$$\phi^{\mathsf{L}}(f(t_1, \ldots, t_l)) = f_{\langle\!\langle \phi^*(t_1), \ldots, \phi^*(t_l) \rangle\!\rangle^f}(\phi^{\mathsf{L}} t_1, \ldots, \phi^{\mathsf{L}} t_l)$$

- Labelling

Free Σ-monoid	terms	$M_{\Sigma} Z$	$l \to r \in \mathcal{R}$
Σ-m.m.	labelling map	$\phi^{\mathsf{L}} \downarrow$	$\downarrow$
Σ-monoid	labelled terms	$M_{\overline{\Sigma}} Z$	$\phi^{\mathsf{L}}(l) \to \phi^{\mathsf{L}}(r) \in \overline{\mathcal{R}}$
			labelled IDTS

Higher-Order Semantic Labelling

▷ **Proposition**

terms	$M_\Sigma 0$	$s \longrightarrow_{\mathcal{R}} t$
labelling map ϕ^{L}	$\downarrow$	$\mapsto$
labelled terms	$M_{\overline{\Sigma}} 0$	$\phi^{\mathsf{L}}(s) \longrightarrow^{*}_{\mathsf{Decr}} \cdot \longrightarrow_{\overline{\mathcal{R}}} \phi^{\mathsf{L}}(t)$

▷ **Th. [Higher-order semantic labelling]**

IDTS $\overline{\mathcal{R}} \cup \mathsf{Decr}$ is terminating $\quad \Rightarrow \quad$ $\mathcal{R}$ is terminating.

▷ "Decreasing rules" Decr

$$f_p(z_1, \ldots, z_l) \;\longrightarrow\; f_q(z_1, \ldots, z_l)$$

for all labels $p > q$

Application 1: Simply-Typed $\boldsymbol{\lambda}\mathrm{x}$-calculus [Bloo,Rose'95]

$$
\begin{aligned}
(\boldsymbol{\lambda} x.M)N &\longrightarrow M\langle x := N\rangle \\
(MN)\langle x := K\rangle &\longrightarrow M\langle x := K\rangle\ N\langle x := K\rangle \\
(\boldsymbol{\lambda} y.M)\langle x := K\rangle &\longrightarrow \boldsymbol{\lambda} y.M\langle x := K\rangle && \text{if } x \neq y \\
x\langle x := K\rangle &\longrightarrow K \\
M\langle x := K\rangle &\longrightarrow M && \text{if } x \notin \mathsf{FV}(M)
\end{aligned}
$$

▷ This doesn't follow the General Schema:

$$?? \qquad @ > -\langle - := -\rangle \qquad\qquad - \langle - := -\rangle > @$$

▷ Labels help!

Semantics $\cdots$ simply typed $\boldsymbol{\lambda}$-terms evaluating all ex. subst.

Application 1: Simply-Typed $\boldsymbol{\lambda}\mathrm{x}$-calculus

$$
\begin{aligned}
(\boldsymbol{\lambda} x.M)N &\to M\langle x := N\rangle \\
(MN)\langle x := K\rangle &\to M\langle x := K\rangle\ N\langle x := K\rangle \\
(\boldsymbol{\lambda} y.M)\langle x := K\rangle &\to \boldsymbol{\lambda} y.M\langle x := K\rangle \qquad \text{if } x \neq y \\
x\langle x := K\rangle &\to K \\
M\langle x := K\rangle &\to M \qquad \text{if } x \notin \mathsf{FV}(M)
\end{aligned}
$$

- ▷ Labelled rules

$$
\begin{aligned}
\mathsf{ap}_{(\boldsymbol{\lambda} x.s)t}(\boldsymbol{\lambda}(x.M(x)), N) &\to M(x)\langle x := N\rangle_{s[x:=t]} \\
(\mathsf{ap}_{s\,t}(M(x), N(x)))\langle x := K\rangle_{st[x:=u]} &\to \mathsf{ap}_{st[x:=u]}(\cdots)
\end{aligned}
$$

Precedence: $\mathsf{ap}_s > -\langle - := -\rangle_t > \mathsf{ap}_t > \boldsymbol{\lambda}$ for $s\ (\to_\beta \cup \rhd)^*\ t$

- ▷ This follows the General Schema, hence SN
- ▷ Point: $\boldsymbol{\lambda}$-terms form a quasi-model of $\boldsymbol{\lambda}\mathrm{x}$-calculus
- ▷ NB. Not a termination model (i.e. not giving '>') but useful

Application 2: Labelling with Term Model

- ▷ Example: λx-calculus
- ▷ (Restricted) term model is a typical model
- ▷ Take the full term model $(T_\Sigma V, (\longrightarrow_{\mathcal{R}} \cup \rhd)^*)$

Def. [Middeldorp,Ohsaki,Zantema CADE'96]
A 1st-order TRS $\mathcal{R}$ is precedence terminating
if ∃ well-founded order ("precedence") s.t.
$f(\vec{t}) \longrightarrow r \in \mathcal{R}, \quad f > \forall g \in \mathrm{Fun}(r)$

Prop. $\mathcal{R}$ SN $\Leftrightarrow$ term labelled $\overline{\mathcal{R}} \cup$ Decr precedence terminating

- ▷ TRS ok [MOZ'96]
- ▷ IDTS fails – subterm property is not closed under substitutions
- ▷ Solid IDTS ok   new notion

Solid IDTS

Def. A term $\boldsymbol{t}$ is solid if for each $\mathrm{z}(\boldsymbol{s_1}, \ldots, \boldsymbol{s_n})$ in $\boldsymbol{t}$,
. all $\boldsymbol{s_i}$ do not contain function symbols

Def. IDTS $\boldsymbol{\mathcal{R}}$ is solid if

i. $\boldsymbol{\mathcal{R}}$ consists of solid terms only

ii. (about strictly positive inductive types and accessibility of variables)

Example

i. $\mathrm{ap}(\boldsymbol{\lambda}(\boldsymbol{x}.\boldsymbol{M}(\boldsymbol{x})), \boldsymbol{N}) \longrightarrow \boldsymbol{M}(\boldsymbol{N})$

ii. $\boldsymbol{\lambda}\mathrm{x}$-calculus

Prop. Solid IDTS $\boldsymbol{\mathcal{R}}$ SN $\Leftrightarrow$ $\overline{\boldsymbol{\mathcal{R}}} \cup \mathrm{Decr}$ precedence terminating

Application 3: Modularity with HO-RPS

HO Recursive Program Schema (RPS)

$$\boldsymbol{f}(\boldsymbol{x_1} \ldots \boldsymbol{x_{i_1}}.\mathrm{Z}_1(\boldsymbol{x_1}, \ldots, \boldsymbol{x_{i_1}}),\ \ldots) \ \longrightarrow\ \boldsymbol{t}$$

Th. Termination is modular for the disjoint union of solid IDTS and solid HO-RPS.

Proof Labelling with a term model given by normal forms of HO-RPS and show precedence termination.

Summary

- Higher-order semantic labelling for IDTSs using $\boldsymbol{\Sigma}$-monoids
- Applications: $\boldsymbol{\lambda}\mathrm{x}$, modularity
- Introduction of solid property
  - Reasonable extension of FO case

Note

- Almost no property is modular for HO rewriting [van Oostrom'05]
- Signature extension doesn't preserve SN for HO rewriting
- But solid case is ok

Why fails?

- Consider labelling with a term model
- Need to establish the property

$$\frac{f(l) \longrightarrow r \in \mathcal{R}}{f_{f(l)\theta}(l) \longrightarrow r \in \overline{\mathcal{R}}}$$

$$f(l)\theta \longrightarrow_{\mathcal{R}} r\theta \trianglerighteq t\theta \qquad \text{for } r \trianglerighteq t$$

Take the order on labels: $(\longrightarrow_{\mathcal{R}} \cup \rhd)^*$

- TRS ok
- IDTS NG, since

$$\mathrm{z}(f) \rhd f \quad \Rightarrow \quad c \not\rhd f \qquad \text{by } \mathrm{z} \mapsto c$$

Σ-monoids [Fiore,Plotkin,Turi'99]

A Σ-monoid consists of

▷ a monoid object $M = (M, \eta, \mu)$
in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V})$ ("substitution prod.") with

▷ a Σ-binding algebra $\alpha : \Sigma M \longrightarrow M$ such that

$$
\begin{array}{ccccc}
\Sigma(M) \bullet M & \xrightarrow{st} & \Sigma(M \bullet M) & \xrightarrow{\Sigma\mu} & \Sigma M \\
{\scriptstyle \alpha \bullet M}\downarrow & & & & \downarrow{\scriptstyle \alpha} \\
M \bullet M & & \xrightarrow[\mu]{\qquad\qquad\qquad} & & M
\end{array}
$$

commutes.