Complete Algebraic Semantics for Second-Order Rewriting Systems based on Abstract Syntax with Variable Binding

Makoto Hamana

Faculty of Informatics, Gunma University, Japan

SYCO 10

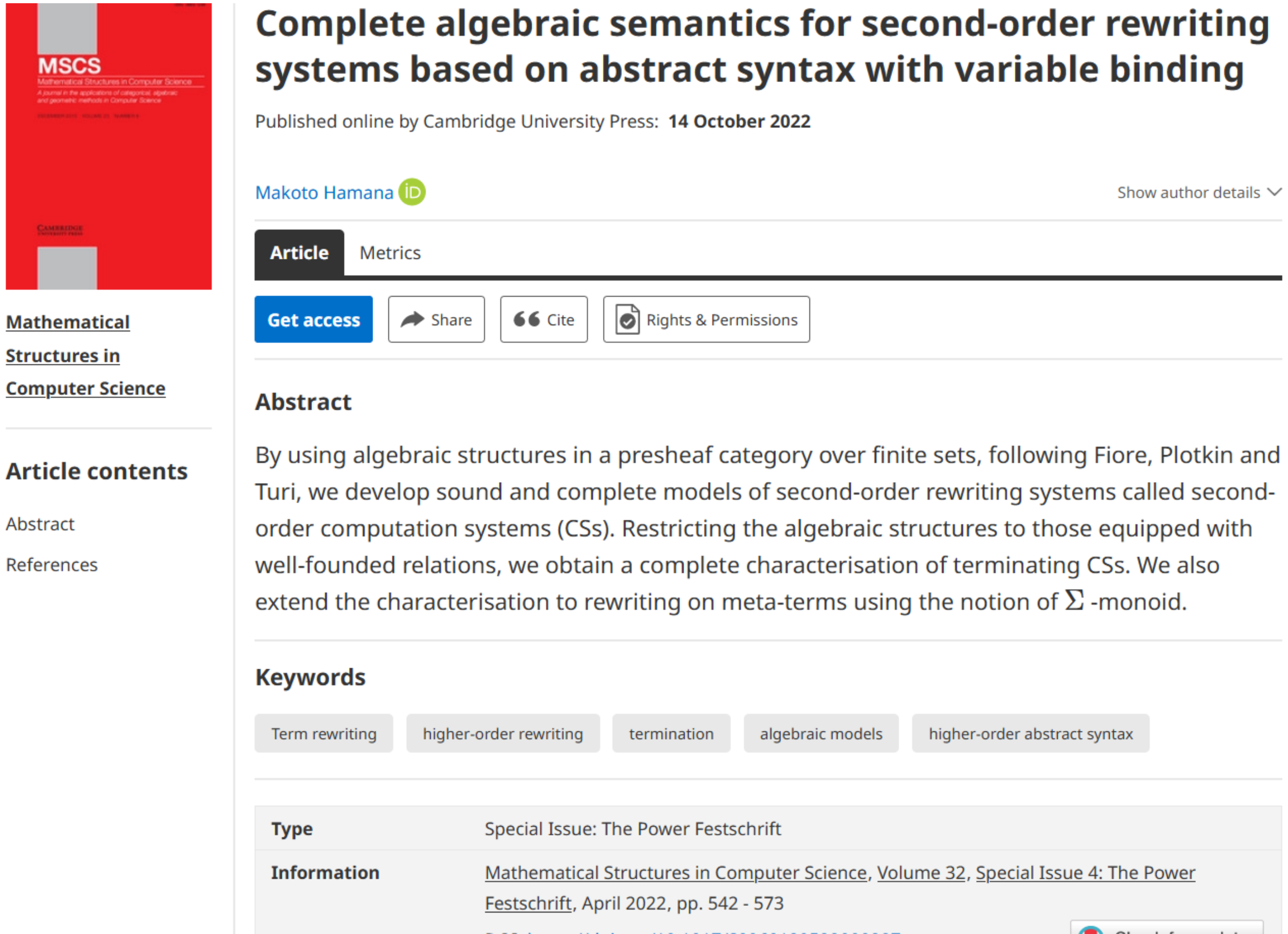
19th December, 2022, Edinburgh

This Talk

▷ Complete algebraic semantics of second-order rewriting

This Talk

- ▷ Complete algebraic semantics of second-order rewriting
- ▷ Based on my paper
  - – Complete Algebraic Semantics for Second-Order Rewriting Systems based on Abstract Syntax with Variable Binding
  - – MSCS, CUP, 2022,
    Special Issue of John Power Festschrift



MSCS

Mathematical Structures in Computer Science

Article contents

Abstract

References

Complete algebraic semantics for second-order rewriting systems based on abstract syntax with variable binding

Published online by Cambridge University Press: **14 October 2022**

Makoto Hamana

Show author details

Article Metrics

Get access | Share | Cite | Rights & Permissions

Abstract

By using algebraic structures in a presheaf category over finite sets, following Fiore, Plotkin and Turi, we develop sound and complete models of second-order rewriting systems called second-order computation systems (CSs). Restricting the algebraic structures to those equipped with well-founded relations, we obtain a complete characterisation of terminating CSs. We also extend the characterisation to rewriting on meta-terms using the notion of Σ-monoid.

Keywords

Term rewriting | higher-order rewriting | termination | algebraic models | higher-order abstract syntax

Type	Special Issue: The Power Festschrift
Information	Mathematical Structures in Computer Science, Volume 32, Special Issue 4: The Power Festschrift, April 2022, pp. 542 - 573

First-order Rewriting: Review

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

$$fact(0) \longrightarrow S(0)$$
$$fact(S(x)) \longrightarrow fact(x) * S(x)$$

Rewrite steps:

```
fact(S(S(0)))  =>  fact(S(0)) * S(S(0))  =>  (fact(0) * S(0)) * S(S(0))
 =>  (S(0) * S(0)) * S(S(0))  ==> S(S(0))   (normal form)
```

Fundametal problem

▷ Termination   (Strong Normalisation)

▷ How can we prove the termination of $\mathcal{R}$?

TRS: Sound and Complete Algebraic Characterisation

Thm. [Huet and Lankford'78]

A first-order term rewriting system $\mathcal{R}$ is terminating

$$\Leftrightarrow$$

there exists a well-founded monotone $\boldsymbol{\Sigma}$-algebra $(\boldsymbol{A}, >_{\boldsymbol{A}})$ that is compatible with $\mathcal{R}$.

Termination proof method

[$\Leftarrow$] Find a well-founded monotone $\boldsymbol{\Sigma}$-algebra that is compatible with $\mathcal{R}$.

First-order Rewriting: Review

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

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

Semantics: well-founded monotone Σ-algebra $(\mathbb{N}, >)$ given by

$$fact^{\mathbb{N}}(x) = 2x + 2 \qquad x *^{\mathbb{N}} y = x + y \qquad S^{\mathbb{N}}(x) = 2x + 1 \qquad 0^{\mathbb{N}} = 1$$

Then it is compatible with $\mathcal{R}$ as

$$\begin{array}{llll} fact^{\mathbb{N}}(0^{\mathbb{N}}) & = 2 + 2 & > \quad 2 + 1 & = S^{\mathbb{N}}(0^{\mathbb{N}}) \\ fact^{\mathbb{N}}(S^{\mathbb{N}}(x)) & = 2(2x + 1) + 2 & > \quad 2x + 2x + 1 & = fact^{\mathbb{N}}(x) * S^{\mathbb{N}}(x) \end{array}$$

Hence $\mathcal{R}$ is terminating.

Aim: Sound and Complete Algebraic Characterisation

Thm. [Huet and Lankford'78]

A first-order term rewriting system $\mathcal{R}$ is terminating

$$\Leftrightarrow$$

there exists a well-founded monotone $\mathbf{\Sigma}$-algebra $\boldsymbol{A}$ that
is compatible with $\mathcal{R}$.

- ▷ Aim: Extend this to second-order rewriting
- ▷ Give: Complete algebraic semantics of second-order rewriting

Example of Second-Order Rewriting : Prenex normal forms

$$\mathrm{P}\wedge\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) \longrightarrow \forall(\boldsymbol{x}.\mathrm{P}\wedge\mathrm{Q}[\boldsymbol{x}]) \qquad \neg\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) \longrightarrow \exists(\boldsymbol{x}.\neg(\mathrm{Q}[\boldsymbol{x}]))$$

$$\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}])\wedge\mathrm{P} \longrightarrow \forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]\wedge\mathrm{P}) \qquad \neg\exists(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) \longrightarrow \forall(\boldsymbol{x}.\neg(\mathrm{Q}[\boldsymbol{x}]))$$

Signature: $\neg, \wedge, \vee, \forall, \exists$

Example of Second-Order Rewriting : Prenex normal forms

$$
\begin{array}{llcll}
\mathrm{P}\wedge\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) & \longrightarrow \forall(\boldsymbol{x}.\mathrm{P}\wedge\mathrm{Q}[\boldsymbol{x}]) & \quad & \neg\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) & \longrightarrow \exists(\boldsymbol{x}.\neg(\mathrm{Q}[\boldsymbol{x}])) \\
\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}])\wedge\mathrm{P} & \longrightarrow \forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]\wedge\mathrm{P}) & \quad & \neg\exists(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) & \longrightarrow \forall(\boldsymbol{x}.\neg(\mathrm{Q}[\boldsymbol{x}]))
\end{array}
$$

Signature: $\neg, \wedge, \vee, \forall, \exists$

Second-Order Rewriting System is defined on
Second-Order Abstract Syntax

- ▷ Abstract syntax with variable binding [Fiore, Plotkin, Turi '99]
- ▷ Metavariables with arities (e.g. P,Q)
- ▷ Substitutions (Metavars, object vars)

Example: the $\boldsymbol{\lambda}$-calculus as a Second-Order Rewriting System

$$\boldsymbol{\lambda}(\boldsymbol{x}.\mathrm{M}[\boldsymbol{x}]) \mathbin{\boldsymbol{@}} \mathrm{N} \longrightarrow \mathrm{M}[\mathrm{N}]$$

$$\boldsymbol{\lambda}(\boldsymbol{x}.\mathrm{M} \mathbin{\boldsymbol{@}} \boldsymbol{x}) \longrightarrow \mathrm{M}$$

▷ Signature: $\boldsymbol{\lambda}, \boldsymbol{@}$

Abstract Syntax and Variable Binding [Fiore,Plotkin,Turi LICS'99]

- Aim: To model syntax with variable binding, e.g.

$$\frac{}{x_1, \dots, x_n \vdash x_i} \qquad \frac{x_1, \dots, x_n \vdash t \quad x_1, \dots, x_n \vdash s}{x_1, \dots, x_n \vdash t@s}$$

$$\frac{x_1, \dots, x_n, x_{n+1} \vdash t}{x_1, \dots, x_n \vdash \lambda(x_{n+1}.t)}$$

- Syntax generated by 3 constructors

- λ is a special unary function symbol:
  it decreases the context

Abstract Syntax and Variable Binding [Fiore,Plotkin,Turi LICS'99]

▷ Aim: model syntax with variable binding, e.g.

$$\frac{}{\boldsymbol{n} \vdash \boldsymbol{i}} \qquad \frac{\boldsymbol{n} \vdash \boldsymbol{t} \quad \boldsymbol{n} \vdash \boldsymbol{s}}{\boldsymbol{n} \vdash \boldsymbol{t@s}}$$

$$\frac{\boldsymbol{n+1} \vdash \boldsymbol{t}}{\boldsymbol{n} \vdash \boldsymbol{\lambda}(\boldsymbol{n+1.t})}$$

▷ Category $\mathbb{F}$ for variable contexts

objects: $\boldsymbol{n} = \{\mathbf{1}, \ldots, \boldsymbol{n}\}$ (variable contexts)

arrows: all functions $\boldsymbol{n} \longrightarrow \boldsymbol{n}'$ (renamings)

▷ Presheaf category $\mathbf{Set}^{\mathbb{F}}$

Models of Syntax with Binding: Σ-Algebras in $\mathbf{Set}^{\mathbb{F}}$

Def. A binding signature Σ consists of a set Σ of function symbols with binding arities:

$$f : \langle n_1, \ldots, n_l \rangle$$

which has l arguments and binds n_i variables in the i-th argument .

Def. A Σ-algebra $A = (A, [f^A]_{f \in \Sigma})$ in $\mathbf{Set}^{\mathbb{F}}$ consists of

▷ carrier: a presheaf $A \in \mathbf{Set}^{\mathbb{F}}$

▷ operations: arrows of $\mathbf{Set}^{\mathbb{F}}$

$$f^A : \delta^{n_1} A \times \ldots \times \delta^{n_l} A \longrightarrow A$$

corresponding to function symbols $f : \langle n_1, \ldots, n_l \rangle \in \Sigma$.

▷ Context extension: $\delta A \in \mathbf{Set}^{\mathbb{F}}$; $(\delta A)(n) = A(n+1)$

Example: $\boldsymbol{\lambda}$-terms

- Binding signature $\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}$ for $\boldsymbol{\lambda}$-terms

$$\boldsymbol{\lambda} \quad : \langle 1 \rangle, \qquad\qquad @ \quad : \langle 0, 0 \rangle$$

- Carrier: the presheaf $\boldsymbol{\Lambda}$ of all $\boldsymbol{\lambda}$-terms

$$\boldsymbol{\Lambda}(n) = \{ t \mid n \vdash t \}$$

$$\boldsymbol{\Lambda}(\rho) : \boldsymbol{\Lambda}(m) \to \boldsymbol{\Lambda}(n) \qquad \text{renaming on } \boldsymbol{\lambda}\text{-terms for } \rho : m \to n \text{ in } \mathbb{F}.$$

- Forms a $\mathbf{V}+\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}$-algebra

$$\begin{array}{llllll}
\mathsf{var}^{\boldsymbol{\Lambda}} : \mathbf{V} \to \boldsymbol{\Lambda} & @^{\boldsymbol{\Lambda}} : \boldsymbol{\Lambda} \times \boldsymbol{\Lambda} \to \boldsymbol{\Lambda} & \boldsymbol{\lambda}^{\boldsymbol{\Lambda}} & : \delta\boldsymbol{\Lambda} & \to \boldsymbol{\Lambda} \\
\quad i \mapsto i & \quad s\ ,\ t \mapsto s@t & \boldsymbol{\lambda}^{\boldsymbol{\Lambda}}(n) & : \boldsymbol{\Lambda}(n+1) & \to \boldsymbol{\Lambda}(n) \\
 & & & t & \mapsto \boldsymbol{\lambda} n{+}1.t
\end{array}$$

- Presheaf of variables: $\mathbf{V} \in \mathbf{Set}^{\mathbb{F}}; \mathbf{V}(n) = \{1, \ldots, n\}$
- Thm. $\boldsymbol{\Lambda}$ $(= \mathbf{T}_{\boldsymbol{\Sigma}}\mathbf{V})$ is an initial $\mathbf{V} + \boldsymbol{\Sigma}_{\boldsymbol{\lambda}}$-algebra.

Second-Order Abstract Syntax

- Abstract syntax with variable binding
- Metavariables with arities
- Substitutions (Metavars, object vars)

Models of Secound-Order Abstract Syntax: $\boldsymbol{\Sigma}$-monoids

- A $\boldsymbol{\Sigma}$-monoid [Fiore, Plotkin, Turi'99] is
  - a $\boldsymbol{\Sigma}$-algebra $\boldsymbol{A}$ with
  - a monoid structure

$$\mathbf{V} \xrightarrow{\nu} \boldsymbol{A} \xleftarrow{\mu} \boldsymbol{A} \bullet \boldsymbol{A}$$

    in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V})$,
  - both are compatible.
- Idea
  - Unit ν models the embedding of variables
  - Multiplication μ models substitution for object variables

Algebraic Characterisation of Syntax with Binding

Given a binding signature $\boldsymbol{\Sigma}$

- The presheaf of all $\boldsymbol{\Sigma}$-terms

$$\mathbf{T_{\Sigma}V}(\boldsymbol{n}) = \{\boldsymbol{t} \mid \boldsymbol{n} \vdash \boldsymbol{t}\}$$

- Multiplication $\boldsymbol{\mu} : \mathbf{T_{\Sigma}V} \bullet \mathbf{T_{\Sigma}V} \longrightarrow \mathbf{T_{\Sigma}V}$

$$\boldsymbol{\mu}_{\boldsymbol{n}}^{(\boldsymbol{m})}(\boldsymbol{t};\ \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{\boldsymbol{m}}) \triangleq \boldsymbol{t}[\mathbf{1} := \boldsymbol{s}_1, \ldots, \boldsymbol{n} := \boldsymbol{s}_{\boldsymbol{m}}]$$

(the substitution of $\boldsymbol{\Sigma}$-terms for de Bruijn variables)

Algebraic Characterisation of Syntax with Binding

Given a binding signature $\mathbf{\Sigma}$

▷ The presheaf of all $\mathbf{\Sigma}$-terms

$$\mathbf{T}_{\Sigma}\mathbf{V}(n) = \{t \mid n \vdash t\}$$

▷ Multiplication $\mu : \mathbf{T}_{\Sigma}\mathbf{V} \bullet \mathbf{T}_{\Sigma}\mathbf{V} \longrightarrow \mathbf{T}_{\Sigma}\mathbf{V}$

$$\mu_n^{(m)}(t;\ s_1, \ldots, s_m) \triangleq t[1 := s_1, \ldots, n := s_m]$$

(the substitution of $\mathbf{\Sigma}$-terms for de Bruijn variables)

▷ Thm. [Fiore, Plotkin, Turi'99]

– $(\mathbf{T}_{\Sigma}\mathbf{V}, \nu, \mu)$ is an initial $\mathbf{\Sigma}$-monoid.

– $(\mathbf{T}_{\Sigma}\mathbf{V}, \nu)$ is an initial $\mathbf{V} + \mathbf{\Sigma}$-algebra.

▶ How to model metavariables and substitutions for metavariables?

Algebraic Characterisation of Syntax with Binding

Given a binding signature $\mathbf{\Sigma}$

▷ The presheaf of all $\mathbf{\Sigma}$-terms

$$\mathbf{T}_\Sigma\mathbf{V}(n) = \{t \mid n \vdash t\}$$

▷ Multiplication $\mu : \mathbf{T}_\Sigma\mathbf{V} \bullet \mathbf{T}_\Sigma\mathbf{V} \longrightarrow \mathbf{T}_\Sigma\mathbf{V}$

$$\mu_n^{(m)}(t;\ s_1, \ldots, s_m) \triangleq t[1 := s_1, \ldots, n := s_m]$$

(the substitution of $\mathbf{\Sigma}$-terms for de Bruijn variables)

▷ Thm. [Fiore, Plotkin, Turi'99]

– $(\mathbf{T}_\Sigma\mathbf{V}, \nu, \mu)$ is an initial $\mathbf{\Sigma}$-monoid.

– $(\mathbf{T}_\Sigma\mathbf{V}, \nu)$ is an initial $\mathbf{V} + \mathbf{\Sigma}$-algebra.

▶ How to model metavariables and substitutions for metavariables?

▶ Free $\mathbf{\Sigma}$-monoids [Hamana, APLAS'04]

Meta-terms: Terms with Metavariables [Aczel '78]

- A binding signature $\boldsymbol{\Sigma}$
- $\boldsymbol{Z}$ is an $\mathbb{N}$-indexed set of metavariables parameterised by arities:

$$\boldsymbol{Z}(\boldsymbol{l}) \triangleq \{\mathrm{M} \mid \mathrm{M}^{\boldsymbol{l}}, \text{ where } \boldsymbol{l} \in \mathbb{N}\}.$$

- Raw meta-terms generated by $\boldsymbol{Z}$:

$$\boldsymbol{t} ::= \boldsymbol{x} \mid \boldsymbol{f}(\boldsymbol{x}_1 \cdots \boldsymbol{x}_{i_1}.\boldsymbol{t}_1, \ldots, \boldsymbol{x}_1 \cdots \boldsymbol{x}_{i_l}.\boldsymbol{t}_l) \mid \mathrm{M}[\boldsymbol{t}_1, \ldots, \boldsymbol{t}_l]$$

- A meta-term $\boldsymbol{t}$ is a raw meta-term derived from:

$$\frac{\boldsymbol{x} \in \boldsymbol{n}}{\boldsymbol{n} \vdash \boldsymbol{x}} \qquad \frac{\boldsymbol{f} : \langle \boldsymbol{i}_1, \ldots, \boldsymbol{i}_l \rangle \in \boldsymbol{\Sigma} \quad \boldsymbol{n}+\boldsymbol{i}_1 \vdash \boldsymbol{t}_1 \ \cdots \ \boldsymbol{n}+\boldsymbol{i}_l \vdash \boldsymbol{t}_l}{\boldsymbol{n} \vdash \boldsymbol{f}(\boldsymbol{n}+\boldsymbol{1} \ldots \boldsymbol{n}+\boldsymbol{i}_1.\boldsymbol{t}_1, \ \ldots, \ \boldsymbol{n}+\boldsymbol{1} \ldots \boldsymbol{n}+\boldsymbol{i}_l.\boldsymbol{t}_l)}$$

$$\frac{\mathrm{M} \in \boldsymbol{Z}(\boldsymbol{l}) \quad \boldsymbol{n} \vdash \boldsymbol{t}_1 \ \cdots \ \boldsymbol{n} \vdash \boldsymbol{t}_l}{\boldsymbol{n} \vdash \mathrm{M}[\boldsymbol{t}_1, \ldots, \boldsymbol{t}_l]}$$

Meta-terms: Terms with Metavariables

- Presheaf $M_\Sigma Z \in \mathbf{Set}^{\mathbb{F}}$

$$M_\Sigma Z(n) = \{t \mid n \vdash t\}$$

- $\mathbf{V}+\boldsymbol{\Sigma}$-algebra $(M_\Sigma Z, [\nu, f_T]_{f\in\Sigma})$

$$\begin{aligned} \nu(n) : \mathrm{V}(n) &\longrightarrow M_\Sigma Z(n), \\ x &\longmapsto x \end{aligned}$$

$$\begin{aligned} f^T : \delta^{i_1} M_\Sigma Z \times \cdots \times \delta^{i_l} M_\Sigma Z &\longrightarrow M_\Sigma Z \\ (t_1, \ldots, t_l) &\longmapsto f(n{+}\overline{i_1}.t_1, \ldots, n{+}\overline{i_l}.t_l). \end{aligned}$$

- Multiplication $\mu : M_\Sigma Z \bullet M_\Sigma Z \to M_\Sigma Z$

$$t, \quad \overline{s} \longmapsto t[1 := s_1, \ldots, n := s_n]$$

$\cdots$ substitution of meta-terms for object variables

Free $\boldsymbol{\Sigma}$-monoids: Syntax with Metavariables [Hamana, APLAS'04]

Thm. $(M_\Sigma Z, \nu, \mu)$ forms a free $\boldsymbol{\Sigma}$-monoid over Z.

▷ Freeness of $M_\Sigma Z$: in $\mathbf{Set}^{\mathbb{F}}$, given assignment θ

$$\begin{tikzcd} Z \arrow[r, "\eta_Z"] \arrow[dr, "\theta"'] & M_\Sigma Z \arrow[d, "\exists!\, \theta^\sharp"] \\ & A \end{tikzcd}$$

$\boldsymbol{\Sigma}$-monoid morphism

▷ The unique $\boldsymbol{\Sigma}$-monoid morphism $\theta^\sharp$ that extends θ.

Instance: Substitution for Metavariables

Case $\boldsymbol{A} = \mathbf{T}_{\Sigma}\mathbf{V}$ $\cdots$ a $\boldsymbol{\Sigma}$-monoid of terms,

$$\begin{tikzcd} \boldsymbol{Z} \arrow[r, "\eta_Z"] \arrow[dr, "\theta"'] & \boldsymbol{M}_{\Sigma}\boldsymbol{Z} \arrow[d, "\exists!\,\theta^{\sharp}"] \\ & \mathbf{T}_{\Sigma}\mathbf{V} \end{tikzcd}$$

Σ-monoid morphism

▷ $\theta^{\sharp}$ is a substitution of terms for metavariables $\boldsymbol{Z}$

▷ E.g. $\boldsymbol{\Sigma}$: signature for $\boldsymbol{\lambda}$-terms, for $\theta(\mathrm{M}^{(1)}) = a@a$

$$\theta^{\sharp}(\ \lambda(x.\mathrm{M}[x]@y)\ ) = \lambda(\ x.(x@x)@y\ )$$

▷ Other examples of $\boldsymbol{\Sigma}$-monoid $\boldsymbol{A}$:

- $\boldsymbol{M}_{\Sigma}\boldsymbol{Z}$: meta-substitution: substitution of meta-terms for metavars
- Any $\boldsymbol{\Sigma}$-monoid as a model – $\theta^{\sharp}$ is compositional interpretation

Second-Order Rewriting System

Eg. A transformation to prenex normal forms

$$\mathrm{P} \wedge \forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) \;\longrightarrow\; \forall(\boldsymbol{x}.\mathrm{P} \wedge \mathrm{Q}[\boldsymbol{x}]) \qquad \neg\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) \;\longrightarrow\; \exists(\boldsymbol{x}.\neg(\mathrm{Q}[\boldsymbol{x}]))$$

Def.

Rewrite rules $\mathcal{R}$ $\quad \boldsymbol{l} \longrightarrow \boldsymbol{r}$ $\quad$ on meta-terms $\boldsymbol{M}_{\Sigma}\boldsymbol{Z}$

(with some syntactic conditions)

Rewrite relation $\quad \longrightarrow_{\mathcal{R}}$ $\quad$ on terms $\mathbf{T}_{\Sigma}\mathbf{V}$

$$\frac{\boldsymbol{l} \longrightarrow \boldsymbol{r} \in \mathcal{R}}{\boldsymbol{\theta}^{\sharp}(\boldsymbol{l}) \;\longrightarrow_{\mathcal{R}}\; \boldsymbol{\theta}^{\sharp}(\boldsymbol{r})} \qquad \frac{\boldsymbol{s} \;\longrightarrow_{\mathcal{R}}\; \boldsymbol{t}}{\boldsymbol{f}(\ldots,\overline{\boldsymbol{x}}.\boldsymbol{s},\ldots) \;\longrightarrow_{\mathcal{R}}\; \boldsymbol{f}(\ldots,\overline{\boldsymbol{x}}.\boldsymbol{t},\ldots)}$$

- ▷ Substitution $\boldsymbol{\theta} : \boldsymbol{Z} \longrightarrow \mathbf{T}_{\Sigma}\mathbf{V}$ maps metavariables to terms
- ▷ NB. rewriting is defined on terms (without metavars)

Presheaf with relation $(A, >_A)$

Def. A presheaf $A \in \mathbf{Set}^{\mathbb{F}}$ is equipped with a binary relation $>_A$, if

1. $>_A$ is a family $\{>_{A(n)}\}_{n \in \mathbb{F}}$,
2. which is compatible with presheaf action.

(for all $a, b \in A(m)$ and $\rho : m \longrightarrow n$ in $\mathbb{F}$,
if $a >_{A(m)} b$, then $A(\rho)(a) >_{A(n)} A(\rho)(b)$.)

Monotone Algebra

Def. A monotone $\mathbf{V}+\mathbf{\Sigma}$-algebra $(\boldsymbol{A}, >_{\boldsymbol{A}})$ is a $\mathbf{V}+\mathbf{\Sigma}$-algebra $(\boldsymbol{A}, [\boldsymbol{\nu}, \boldsymbol{f}^{\boldsymbol{A}}]_{\boldsymbol{f}\in\mathbf{\Sigma}})$

- ▷ equipped with a relation $>_{\boldsymbol{A}}$ such that
- ▷ every operation $\boldsymbol{f}^{\boldsymbol{A}}$ is monotone.

Thm. $(\mathbf{T}_{\mathbf{\Sigma}}\mathbf{V}, \rightarrow_{\mathcal{R}}\ )$ is a monotone $\mathbf{V}+\mathbf{\Sigma}$-algebra.

Models of Rewrite System $\mathcal{R}$: $(\mathbf{V}+\Sigma, \mathcal{R})$-algebras

A $(\mathbf{V}+\Sigma, \mathcal{R})$-algebra $(A, >_A)$ is a monotone $\mathbf{V}+\Sigma$-algebra satisfying all rules in $\mathcal{R}$ as:

$$\begin{tikzcd}
Z \arrow[rr, "\eta_Z"] \arrow[drr, "\theta"'] & & M_\Sigma Z \arrow[d, "\theta^\sharp"] \\
& & \mathbf{T}_\Sigma \mathbf{V} \arrow[d, "!_A"] \\
& & A
\end{tikzcd}$$

$l \to r \in \mathcal{R}$ ··· a rule

$\theta^\sharp$ a unique Σ-monoid mor. extends θ

initial $\mathbf{V}+\Sigma$-algebra $\mathbf{T}_\Sigma\mathbf{V}$

$\theta^\sharp_n(l) \to_{\mathcal{R}} \theta^\sharp_n(r)$ ··· a rewrite

$!_A$ a unique $\mathbf{V}+\Sigma$-algebra homomor.

$\mathbf{V}+\Sigma$-algebra A

$!_A\theta^\sharp_n(l) >_{A(n)} !_A\theta^\sharp_n(l)$ ··· an interpretation

Soundness and Completeness of Models

Prop. $\quad s \longrightarrow_{\mathcal{R}} t$

$$\Leftrightarrow$$

$$!_A\theta^\sharp(s) \ >_A \ !_A\theta^\sharp(t) \qquad \text{for all } (\mathbf{V}+\boldsymbol{\Sigma}, \boldsymbol{\mathcal{R}})\text{-algebras } \boldsymbol{A}, \text{ assignments } \boldsymbol{\theta}.$$

Proof. [$\Rightarrow$]: By induction of the proof of rewrite.

[$\Leftarrow$]: Take $(\boldsymbol{A}, >_A) = (\mathbf{T}_\Sigma \mathbf{V}, \longrightarrow_{\mathcal{R}}\ )$.

Complete Characterisation of Terminating Second-Order Rewriting

Thm. A second-order rewriting system $\mathcal{R}$ is terminating iff there is a well-founded $(\mathbf{V}+\mathbf{\Sigma}, \mathcal{R})$-algebra $(A, >_A)$.

Proof. ($\Leftarrow$): Suppose a well-founded $(\mathbf{V}+\mathbf{\Sigma}, \mathcal{R})$-algebra $(A, >_A)$.

Assume $\mathcal{R}$ is non-terminating:

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots .$$

By soundness,

$$!_A\theta^\sharp(t_1) >_{A(n)} !_A\theta^\sharp(t_2) >_A \cdots .$$

Contradiction.

($\Rightarrow$): When $\mathcal{R}$ is terminating, the $(\mathbf{V}+\mathbf{\Sigma}, \mathcal{R})$-algebra $(\mathbf{T_\Sigma V}, \to_{\mathcal{R}})$ is a well-founded algebra.

▶ Because of the algebraic chatersiations of abstract sytanx with binding [FPT'99] and meta-terms [H.04]

Application: Termination by Interpretation

$$\mathrm{P}\wedge\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) \;\longrightarrow\; \forall(\boldsymbol{x}.\mathrm{P}\wedge\mathrm{Q}[\boldsymbol{x}]) \qquad \neg\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) \;\longrightarrow\; \exists(\boldsymbol{x}.\neg(\mathrm{Q}[\boldsymbol{x}]))$$

$$\forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}])\wedge\mathrm{P} \;\longrightarrow\; \forall(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]\wedge\mathrm{P}) \qquad \neg\exists(\boldsymbol{x}.\mathrm{Q}[\boldsymbol{x}]) \;\longrightarrow\; \forall(\boldsymbol{x}.\neg(\mathrm{Q}[\boldsymbol{x}]))$$

Take a well-founded monotone $\mathbf{V}+\boldsymbol{\Sigma}$-algebra $(\boldsymbol{K}, >_{\boldsymbol{K}})$
where $\boldsymbol{K}(\boldsymbol{n}) = \mathbb{N}$ with $>_{\boldsymbol{K}(\boldsymbol{n})} = >$ on $\mathbb{N}$.

Operations

$$\boldsymbol{\nu}_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{i}) = \mathbf{0} \qquad \wedge_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x},\boldsymbol{y}) = \vee_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x},\boldsymbol{y}) = \mathbf{2}\boldsymbol{x}+\mathbf{2}\boldsymbol{y}$$

$$\neg_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x}) = \mathbf{2}\boldsymbol{x} \qquad \forall_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x}) = \exists_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x}) = \boldsymbol{x}+\mathbf{1}.$$

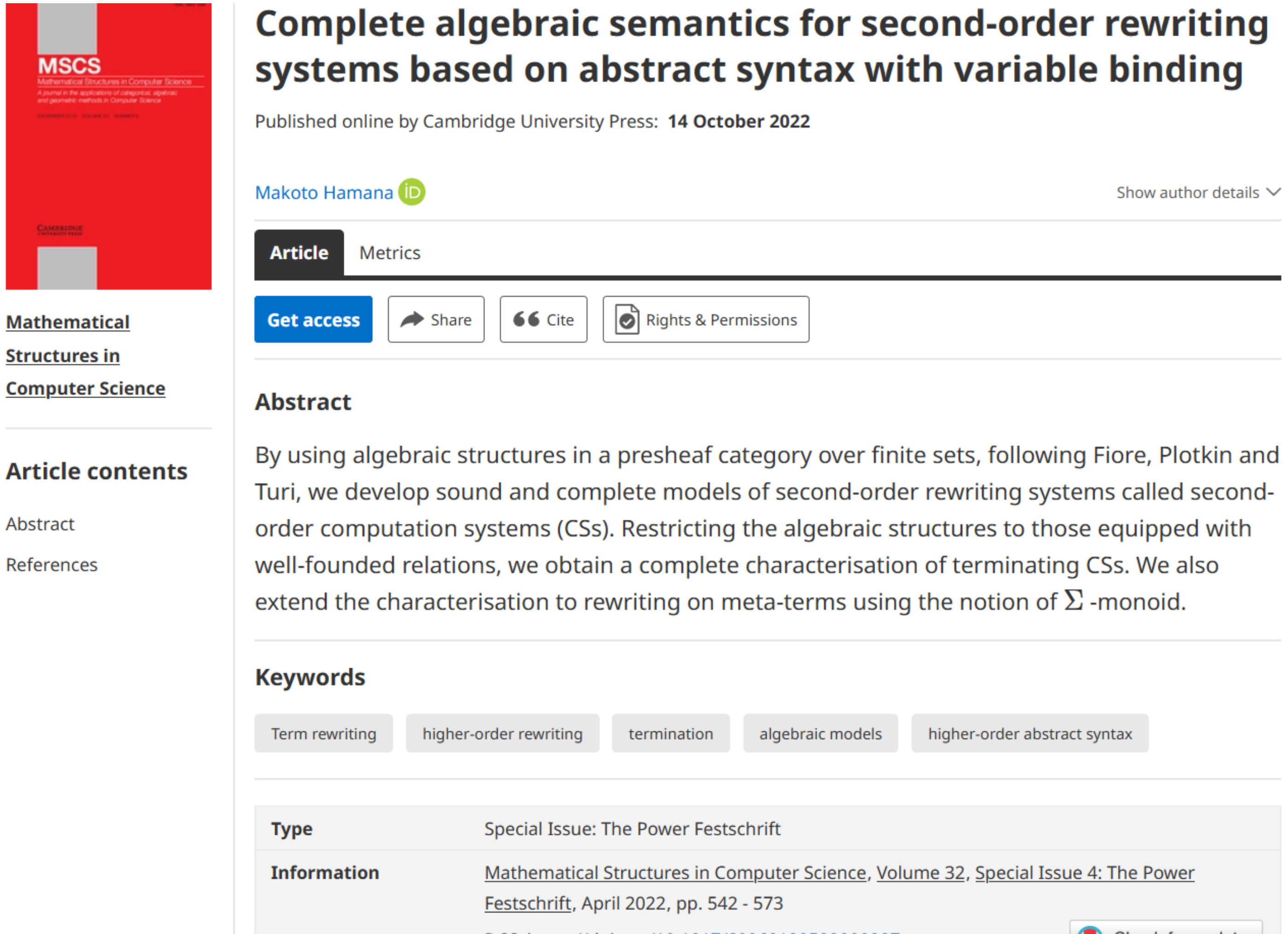
$(\mathbf{V}+\Sigma,\mathcal{R})$-algebra

$$!\boldsymbol{\theta}_0^{\sharp}(\mathrm{P}\wedge\forall(\mathbf{1}.\mathrm{Q}[\mathbf{1}])) = \mathbf{2}\boldsymbol{x}+\mathbf{2}(\boldsymbol{y}+\mathbf{1}) >_{\boldsymbol{K}(0)} (\mathbf{2}\boldsymbol{x}+\mathbf{2}\boldsymbol{y})+\mathbf{1} = !\boldsymbol{\theta}_0^{\sharp}(\forall(\mathbf{1}.\mathrm{P}\wedge\mathrm{Q}[\mathbf{1}]))$$

$$!\boldsymbol{\theta}_0^{\sharp}(\neg\exists(\mathbf{1}.\mathrm{Q}[\mathbf{1}])) = \mathbf{2}(\boldsymbol{y}+\mathbf{1}) >_{\boldsymbol{K}(0)} \mathbf{2}\boldsymbol{y}+\mathbf{1} = !\boldsymbol{\theta}_0^{\sharp}(\forall(\mathbf{1}.\neg(\mathrm{Q}[\mathbf{1}]))).$$

Summary

- ▷ Complete algebraic semantics of second-order rewriting systems
- ▷ Based on my paper
  - – Complete Algebraic Semantics for Second-Order Rewriting Systems based on Abstract Syntax with Variable Binding
  - – MSCS, CUP, 2022,
    Special Issue of John Power Festschrift


MSCS

Mathematical Structures in Computer Science

Article contents

Abstract

References

Complete algebraic semantics for second-order rewriting systems based on abstract syntax with variable binding

Published online by Cambridge University Press: **14 October 2022**

Makoto Hamana

Show author details

Article | Metrics

Get access | Share | Cite | Rights & Permissions

Abstract

By using algebraic structures in a presheaf category over finite sets, following Fiore, Plotkin and Turi, we develop sound and complete models of second-order rewriting systems called second-order computation systems (CSs). Restricting the algebraic structures to those equipped with well-founded relations, we obtain a complete characterisation of terminating CSs. We also extend the characterisation to rewriting on meta-terms using the notion of Σ-monoid.

Keywords

Term rewriting | higher-order rewriting | termination | algebraic models | higher-order abstract syntax

Type	Special Issue: The Power Festschrift
Information	Mathematical Structures in Computer Science, Volume 32, Special Issue 4: The Power Festschrift, April 2022, pp. 542 - 573

Summary

- ▷ Complete algebraic semantics of second-order rewriting systems
- ▷ Based on my paper
  - – Complete Algebraic Semantics for Second-Order Rewriting Systems based on Abstract Syntax with Variable Binding
  - – MSCS, CUP, 2022,
    Special Issue of John Power Festschrift
- ▷ Short history: I visted LFCS, Edinburgh in 1999-2000 as a JSPS postdoc.
- ▷ Thanks to John Power, Gordon Plotkin

Mathematical Structures in Computer Science

Article contents

Abstract

References

Complete algebraic semantics for second-order rewriting systems based on abstract syntax with variable binding

Published online by Cambridge University Press: **14 October 2022**

Makoto Hamana

Show author details

Article | Metrics

Get access | Share | Cite | Rights & Permissions

Abstract

By using algebraic structures in a presheaf category over finite sets, following Fiore, Plotkin and Turi, we develop sound and complete models of second-order rewriting systems called second-order computation systems (CSs). Restricting the algebraic structures to those equipped with well-founded relations, we obtain a complete characterisation of terminating CSs. We also extend the characterisation to rewriting on meta-terms using the notion of Σ-monoid.

Keywords

Term rewriting | higher-order rewriting | termination | algebraic models | higher-order abstract syntax

Type	Special Issue: The Power Festschrift
Information	Mathematical Structures in Computer Science, Volume 32, Special Issue 4: The Power Festschrift, April 2022, pp. 542 - 573

Summary

▷ Complete algebraic characterisation of second-order rewriting systems

▷ using algebraic models of second-order abstrax syntax

Further Topics and Applications

▷ Meta-rewriting: rewriting on meta-terms using monotone $\mathbf{\Sigma}$-monoids

▷ **Modularity** of Termination for Second-Order rewriting [H. LMCS'21]
$\mathbf{A}$: terminating & $\mathbf{B}$ terminating $\Rightarrow$ $\mathbf{A} \uplus \mathbf{B}$: terminating
with several conditions

▷ **Tool** SOL for termination and confluence checking
1st places in the Higher-order Category of

- International Confluence Competition 2020
- Termination Competition 2022

`http://solweb.mydns.jp/sol/`

Appendix

Summary

- ▷ Complete algebraic semantics of second-order rewriting
- ▷ Based on my paper
  - – Complete Algebraic Semantics for Second-Order Rewriting Systems based on Abstract Syntax with Variable Binding
  - – MSCS, CUP, 2022,
    Special Issue of John Power Festschrift
- ▷ Short history: I visted LFCS, Edinburgh
  in 1999-2000 as a JSPS postdoc.
- ▷ Thanks to John Power, Gordon Plotkin

Application: Benefit of Completeness

Binding signature $\mathbf{\Sigma} = \{\boldsymbol{c} : \langle \mathbf{0} \rangle\}$. Second-order rewriting system $\boldsymbol{\mathcal{R}}$

$$\boldsymbol{c}(\mathrm{F}[\mathrm{F}[\mathrm{X}[\boldsymbol{x}]]]) \longrightarrow \mathrm{F}[\mathrm{X}[\boldsymbol{x}]].$$

- No functional modesl – ordinary models of higher-order rewriting [van de Pol '93]
- Our semantics: take the monotone $\mathbf{V}+\mathbf{\Sigma}$-algebra $(\mathbf{T_\Sigma V}, \succ_{\mathbf{T_\Sigma V}})$

$$\boldsymbol{s} \succ_{\mathbf{T_\Sigma V}(\boldsymbol{n})} \boldsymbol{t}$$

  if the numbers of $\boldsymbol{c}$-symbols decreases in $\boldsymbol{s}$ and $\boldsymbol{t}$
- Any assignment into $\mathbf{T_\Sigma V}$ is of the form $\mathrm{F} \mapsto \boldsymbol{c}^{\boldsymbol{k}}(\boldsymbol{x})$, $\mathrm{X} \mapsto \boldsymbol{c}^{\boldsymbol{m}}(\boldsymbol{x})$
- This gives a well-founded $(\mathbf{V}+\mathbf{\Sigma}, \boldsymbol{\mathcal{R}})$-algebra.

Monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V})$

▷ unit: $\mathbf{V} : \mathbb{F} \longrightarrow \mathbf{Set}$; $\mathbf{V}(n) = n$

▷ monoidal product $\bullet$ for any $A, B \in \mathbf{Set}^{\mathbb{F}}$

$$(A \bullet B)(n) \triangleq (\coprod_{m \in \mathbb{N}} A(m) \times B(n)^m)/ \sim$$

where the equivalence relation $\sim$

$$(t; u_{\rho 1}, \ldots, u_{\rho m}) \sim (A(\rho)(t); u_1, \ldots, u_l)$$