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

Makoto Hamana

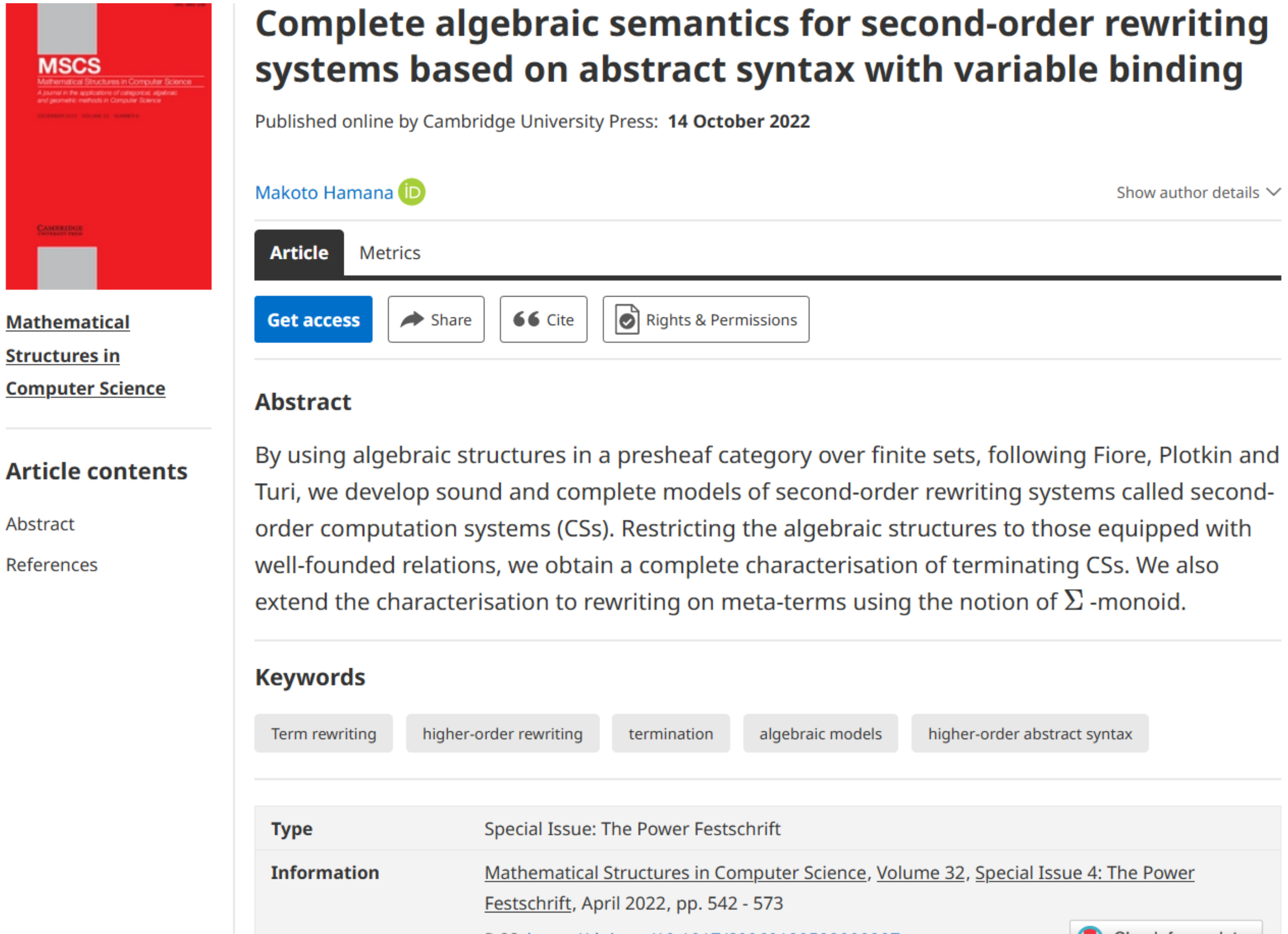
Faculty of Informatics, Gunma University, Japan

SYCO 10

19th December, 2022, Edinburgh

This Talk

- ▷ Complete algebraic semantics of second-order rewriting
- ▷ 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



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) $\mathcal{R}$:

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

Semantics: well-founded monotone $\boldsymbol{\Sigma}$-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

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

Hence $\mathcal{R}$ is terminating.

Aim: Sound and Complete Algebraic Characterisation

Thm. [Huet and Lankford'78]

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

$$\Leftrightarrow$$

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

- ▷ 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: $\neg, \wedge, \vee, \forall, \exists$

Second-Order Rewriting System is defined on
Second-Order Abstract Syntax [Hamana'04, Fiore LICS'06]

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

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

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

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

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

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

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

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

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

- 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: $\boldsymbol{\Sigma}$-Algebras in $\mathbf{Set}^{\mathbb{F}}$

Def. A binding signature $\boldsymbol{\Sigma}$ is a set of function symbols with binding arities:

$$\boldsymbol{f} : \langle \boldsymbol{n}_1, \ldots, \boldsymbol{n}_l \rangle$$

which has $\boldsymbol{l}$ arguments and binds $\boldsymbol{n}_i$ variables in the $\boldsymbol{i}$-th argument .

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

- ▷ carrier: a presheaf $\boldsymbol{A} \in \mathbf{Set}^{\mathbb{F}}$
- ▷ operations: arrows of $\mathbf{Set}^{\mathbb{F}}$

$$\boldsymbol{f}^{\boldsymbol{A}} : \boldsymbol{\delta}^{\boldsymbol{n}_1} \boldsymbol{A} \times \ldots \times \boldsymbol{\delta}^{\boldsymbol{n}_l} \boldsymbol{A} \longrightarrow \boldsymbol{A}$$

  corresponding to function symbols $\boldsymbol{f} : \langle \boldsymbol{n}_1, \ldots, \boldsymbol{n}_l \rangle \in \boldsymbol{\Sigma}$.

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

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

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

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

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

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

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

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

▷ Presheaf of variables: $\mathbf{V} \in \mathbf{Set}^{\mathbb{F}}; \mathbf{V}(n) = \{1, \ldots, n\}$

▷ Thm. $\boldsymbol{\Lambda}$ $(= \mathbf{T}_{\Sigma}\mathbf{V})$ is an initial $\mathbf{V} + \boldsymbol{\Sigma}_{\lambda}$-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

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

    in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V})$,
  - both are compatible.

- Idea
  - Unit $\boldsymbol{\nu}$ models the embedding of variables
  - Multiplication $\boldsymbol{\mu}$ models substitution for object variables

Algebraic Characterisation of Syntax with Binding

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

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

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

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

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

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

Algebraic Characterisation of Syntax with Binding

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

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

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

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

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

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

▷ Thm. [Fiore, Plotkin, Turi'99]

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

– $(\mathbf{T_\Sigma V}, \nu)$ 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

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

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

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

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

▷ Thm. [Fiore, Plotkin, Turi'99]

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

– $(\mathbf{T_\Sigma V}, \nu)$ 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 Σ

▷ Z is an $\mathbb{N}$-indexed set of metavariables parameterised by arities:

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

▷ Raw meta-terms generated by Z:

$$t ::= x \mid f(x_1 \cdots x_{i_1}.t_1, \ldots, x_1 \cdots x_{i_l}.t_l) \mid \mathrm{M}[t_1, \ldots, t_l]$$

▷ A meta-term t is a raw meta-term derived from:

$$\frac{x \in n}{n \vdash x} \qquad \frac{f : \langle i_1, \ldots, i_l \rangle \in \Sigma \quad n{+}i_1 \vdash t_1 \ \cdots \ n{+}i_l \vdash t_l}{n \vdash f(\, n{+}1 \ldots n{+}i_1.t_1,\ \ldots,\ n{+}1 \ldots n{+}i_l.t_l\,)}$$

$$\frac{\mathrm{M} \in Z(l) \quad n \vdash t_1 \ \cdots \ n \vdash t_l}{n \vdash \mathrm{M}[t_1, \ldots, t_l]}$$

Meta-terms: Terms with Metavariables

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

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

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

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

▷ 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 Σ-monoids: Syntax with Metavariables [Hamana, APLAS'04]

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

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

$$
\begin{array}{ccc}
Z & \xrightarrow{\eta_Z} & M_\Sigma Z \\
& {\scriptstyle\theta} \searrow & \downarrow {\scriptstyle \exists!\, \theta^\sharp} \\
& & A
\end{array}
$$

Σ-monoid morphism

Σ-monoid

▷ The unique Σ-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,

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

$\boldsymbol{\Sigma}$-monoid morphism

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

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

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

▷ 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 – $\boldsymbol{\theta}^\sharp$ is compositional interpretation

Second-Order Rewriting System

Eg. A transformation to prenex normal forms

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

Def.

Rewrite rules $\mathcal{R}$ $\quad l \rightarrow r$ on meta-terms $M_\Sigma Z$

(with some syntactic conditions)

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

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

- ▷ Substitution $\theta : Z \rightarrow \mathbf{T}_\Sigma \mathbf{V}$ 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. $(\mathbf{T}_{\mathbf{\Sigma}}\mathbf{V}, \to_{\mathcal{R}}\ )$ is a monotone $\mathbf{V}+\mathbf{\Sigma}$-algebra.

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

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

$$\begin{tikzcd}
Z \arrow[r, "\eta_Z"] \arrow[dr, "\theta"'] & M_\Sigma Z \arrow[d, "\theta^\sharp"] \\
\text{initial } \mathbf{V}+\mathbf{\Sigma}\text{-algebra} & \mathbf{T}_\Sigma \mathbf{V} \arrow[d, "!_A"] \\
\mathbf{V}+\mathbf{\Sigma}\text{-algebra} & A
\end{tikzcd}$$

$l \to r \in \mathcal{R}$ $\cdots$ a rule

a unique $\mathbf{\Sigma}$-monoid mor. extends θ

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

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

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

Soundness and Completeness of Models

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

$$\Leftrightarrow$$

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

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

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

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,

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

Contradiction.

$(\Rightarrow)$: When $\mathcal{R}$ is terminating, the $(\mathbf{V}+\mathbf{\Sigma}, \mathcal{R})$-algebra $(\mathbf{T}_{\Sigma}\mathbf{V}, \to_{\mathcal{R}})$ 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

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

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

$$
\begin{array}{ll}
\nu_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{i}) = \mathbf{0} & \qquad \wedge_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x}, \boldsymbol{y}) = \vee_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{2}\boldsymbol{x} + \mathbf{2}\boldsymbol{y} \\
\neg_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x}) = \mathbf{2}\boldsymbol{x} & \qquad \forall_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x}) = \exists_{\boldsymbol{n}}^{\boldsymbol{K}}(\boldsymbol{x}) = \boldsymbol{x} + \mathbf{1}.
\end{array}
$$

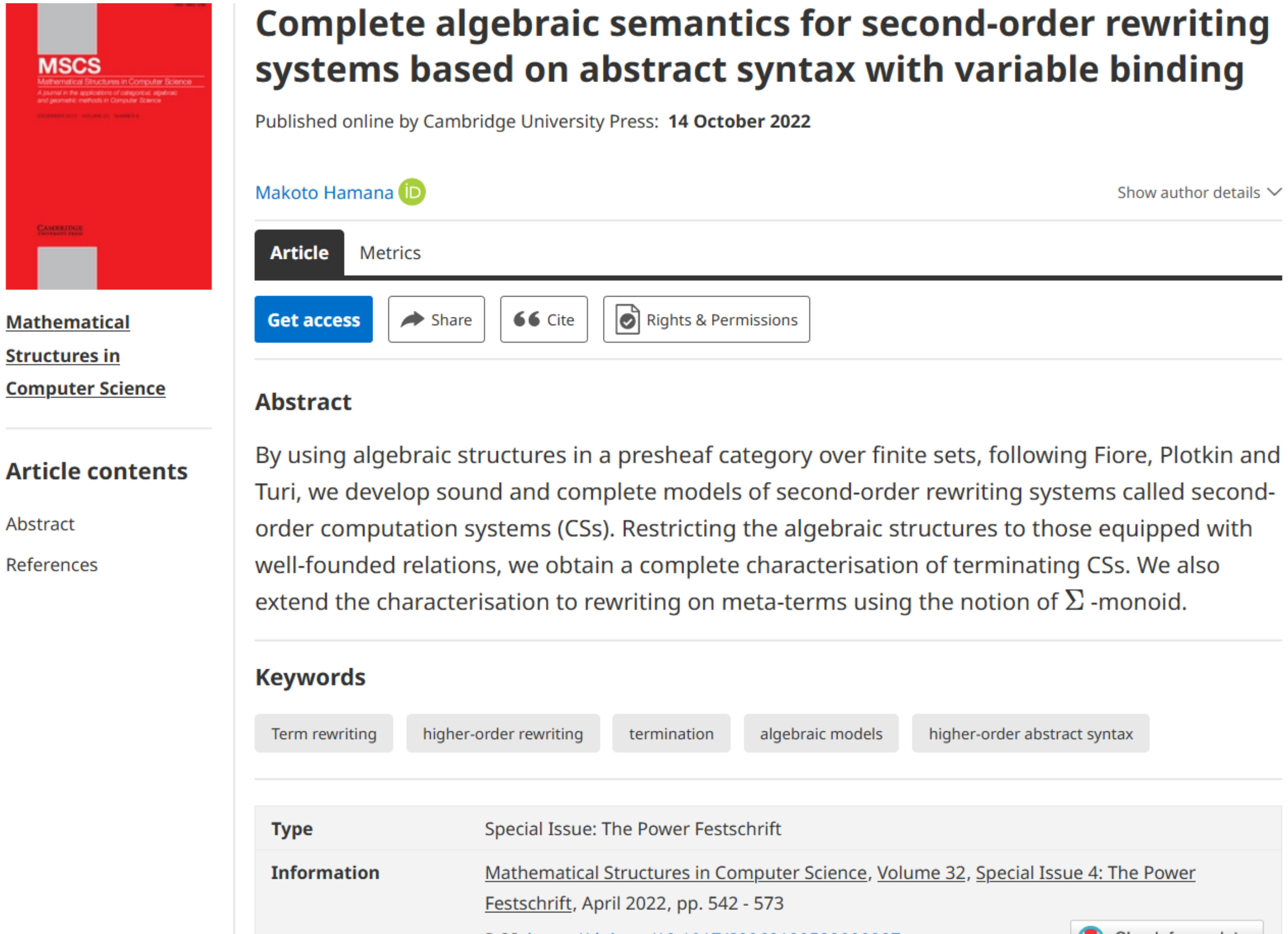
$(\mathbf{V}+\Sigma,\mathcal{R})$-algebra

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

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

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 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 Σ-monoids
- ▷ **Modularity** of Termination for Second-Order rewriting [H. LMCS'21]
  $\mathbf{A}$: terminating & $\mathbf{B}$ terminating $\quad\Rightarrow\quad$ $\mathbf{A} \uplus \mathbf{B}$: 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/webcui/sol/`

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