Semantic Labelling for Termination of Combinatory Reduction Systems

Makoto Hamana

Department of Computer Science, Gunma University, Japan
hamana@cs.gunma-u.ac.jp

Abstract. We give a method of proving termination of Klop's higher-order rewriting format, combinatory reduction system (CRS). Our method called higher-order semantic labelling is an extension of Zantema's semantic labelling for first-order term rewriting systems. We systematically define the labelling by using the complete algebraic semantics of CRS.

1 Introduction

Rewrite rules appear everywhere in computer science and logic. When reasoning with such rewrite rules, termination property is important, so we need some way to ensure termination of rewrite rules. This topic has been extensively investigated in the field of term rewriting [3]. Higher-order extension of term rewriting are known as several formats: major representatives are Klop's Combinatory Reduction Systems (CRSs) [13], Nipkow's Higher-order Rewrite Systems [15], and Blanqui, Okada and Jouannaud's systems [5, 4]. In this paper, we deal with Klop's CRSs. We give a method to prove termination (meaning strong normalisation) of a CRS, called *higher-order semantic labelling*. This is an extension of semantic labelling for first-order term rewriting systems (TRSs) given by Zantema [17].

Background. This paper is based on the algebraic semantics of CRS [11] by Σ-monoids. The notion of Σ-monoids was introduced by Fiore, Plotkin and Turi [9], then a higher-order abstract syntax for free Σ-monoids was developed by the author [10]. Full version paper is available from `http://www.keim.cs.gunma-u.ac.jp/~hamana/hslf.pdf`.

2 Preliminaries

CRS. We use the definition of the standard reference [14] of CRSs with a slight modification of syntax used in [8]. This is the exactly the same one used in [11]. Moreover, we treat only CRSs built from binding signatures, which we call *structural CRSs* [11]. Hereafter, we do not explicitly say "structural". A *binding signature* Σ is consisting of a set Σ of function symbols with an arity function $a : \Sigma \to \mathbb{N}^*$ ($\mathbb{N}^*$ denotes the set of all finite sequences of natural numbers). A function symbol of *binding arity* $\langle n_1, \ldots, n_l \rangle$, denoted by $f : \langle n_1, \ldots, n_l \rangle$, has l arguments and binds n_i variables in the i-th argument ($1 \le i \le l$). For a formal treatment of named variables modulo α-equivalence in CRSs, we assume the method of de Bruijn levels [9] for the naming convention of variables in CRSs. We also use the convention that $n \in \mathbb{N}$ denotes the set $\{1, \ldots, n\}$ (n is possibly 0). Under the method of de Bruijn levels, this n means the set of variables from 1 to n.

Fix an $\mathbb{N}$-indexed set Z of metavariables defined by $Z(l) \triangleq \{\mathrm{z} \mid \mathrm{z}$ has arity $l\}$ (z is called a *metavariable*). A *meta-term* t is obtained if $n \vdash t$ is derived from the following rules for some $n \in \mathbb{N}$.

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

A meta-term containing no metavariables is called a *term*. The $\mathbb{N}$-indexed set of all metaterms is defined by $M_\Sigma Z(n) = \{t \mid n \vdash t\}$ generated by an $\mathbb{N}$-indexed set of metavariables Z. A *rewrite rule*, written $l \to r$, consists of two meta-terms l and r with the following additional restrictions: (i) l and r are closed (w.r.t. variables) meta-terms, (ii) l must be a function term where all meta-applications have the form $\mathrm{z}[x_1, \ldots, x_n]$ with distinct variables x_i, (iii) r can only contain meta-applications with meta-variables occurring in the left-hand side. We may use the notation $Z|n \vdash s \to t$ for a rule or a rewrite step if metavariables and variables in s and t are included in Z and n respectively. We may also simply write $Z \vdash s \to t$ or $n \vdash s \to t$ if another part is not important. A *valuation* θ is a mapping that assigns to n-ary metavariable z a term t $\theta : \mathrm{z} \longmapsto \underline{\lambda}(x_1, \ldots, x_n).t$ where all variables in t are included in $\{x_1, \ldots, x_n\}$. Any valuation is naturally extended to a function on meta-terms. A set of rewrite rules under the signature Σ is called a CRS and denoted by $(\Sigma, \mathcal{R})$ or simply $\mathcal{R}$. The CRS *rewrite relation* $\to_{\mathcal{R}}$ is generated by context and safe valuation closure of a given CRS $\mathcal{R}$.

Binding algebras. Algebraic semantics of CRSs is given by binding algebras. Let $\mathbb{F}$ be the category which has finite cardinals $n = \{1, \ldots, n\}$ (n is possibly 0) as objects, and all functions between them as arrows $m \to n$. This is the category of object variables by the method of de Bruijn levels (i.e. natural numbers) and their renamings. We use the functor category $\mathbf{Set}^{\mathbb{F}}$. An object of $\mathbf{Set}^{\mathbb{F}}$ may be called a *presheaf*. The functor $\delta : \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}$ for "index extension" is defined by $(\delta L)(n) = L(n + 1)$ for $L \in \mathbf{Set}^{\mathbb{F}}$. To a binding signature Σ, we associate the *signature functor* $\Sigma : \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}$ given by $\Sigma A = \coprod_{f:\langle n_1,\ldots,n_l\rangle\in\Sigma} \prod_{1\le i\le l} \delta^{n_i} A$. A Σ-*algebra* is an algebra of this functor. The presheaf of variables $\mathrm{V} \in \mathbf{Set}^{\mathbb{F}}$ is given by $\mathrm{V}(n) = n$. The $\mathbb{N}$-indexed set of all metaterms $M_\Sigma Z$ generated by Z forms a free Σ-monoid [10]. Moreover, when $Z = 0$ (empty set), $M_\Sigma 0$ forms an initial $\mathrm{V} + \Sigma$-algebra and an initial Σ-monoid [9, 10].

Algebraic semantics of rewriting. For a presheaf A, we write $\geq_A$ for a family of preorders $\{\geq_{A(n)}\}_{n\in\mathbb{N}}$, where $\geq_{A(n)}$ is a preorder on a set $A(n)$ for each $n \in \mathbb{N}$. Let $(A_1, \geq_{A_1}), \ldots, (A_l, \geq_{A_l}), (B, \geq_B)$ be presheaves equipped with preorders. A map $f : A_1 \times \cdots \times A_l \longrightarrow B$ in $\mathbf{Set}^{\mathbb{F}}$ is *weakly monotone* if all $n \in \mathbb{N}$, all $a_1, b_1 \in A_1(n), \ldots, a_l, b_l \in A_l(n)$ with $a_k \geq_{A(n)} b_k$ for some k and $a_j = b_j$ for all $j \neq k$, then $f(n)(a_1, \ldots, a_l) \geq_{B(n)} f(n)(b_1, \ldots, b_l)$. A *weakly monotone* $\mathrm{V}{+}\Sigma$-*algebra* $(A, \geq_A)$ is a $\mathrm{V}{+}\Sigma$-algebra equipped with preorders such that every operation is weakly monotone. For a $\mathrm{V} + \Sigma$-algebra A, a *term-generated assignment* $\phi : Z \longrightarrow A$ is a morphism of $\mathbf{Set}^{\mathbb{F}}$ that is a composite $Z \longrightarrow M_\Sigma 0 \longrightarrow A$ when the second morphism is a homomorphism. A $\mathrm{V}{+}\Sigma$-algebra A *satisfies* a rewrite rule $Z \vdash \vec{n}.l \to \vec{n}.r$ if $\phi^*(n)(l) = \phi^*(n)(r)$ for all term-generated assignments $\phi : Z \longrightarrow A$. A *model* A for a CRS $(\Sigma, \mathcal{R})$ is a $\mathrm{V}{+}\Sigma$-algebra A that satisfies all rules in the weakening closure $\mathcal{R}^\circ$ (cf. [11]) which means rules allow free variables in their instances. A weakly monotone $\mathrm{V}{+}\Sigma$-algebra $(A, \geq_A)$ *satisfies* a rewrite rule $Z \vdash \vec{n}.l \to \vec{n}.r$ if $\phi^*(n)(l) \geq_{A(n)} \phi^*(n)(r)$ for all term-generated assignments

$\phi : Z \longrightarrow A$. A *quasi-model* A for $(\Sigma, \mathcal{R})$ is a weakly monotone V+Σ-algebra A that satisfies all rules in the weakening closure $\mathcal{R}^{\circ}$.

3 Higher-Order Semantic Labelling

We assume that Z is an $\mathbb{N}$-indexed set of metavariables, Σ is a binding signature and M is a V+Σ-algebra. We introduce labelling of functions symbols: choose for every $f \in \Sigma$ a corresponding non-empty set S_f of labels, called *sort set*. The binding signature $\overline{\Sigma}$ for labelled function symbols is defined by $\overline{\Sigma} = \{f_p \mid f \in \Sigma,\ p \in S_f\}$ where the binding arity of f_p is defined to be the binding arity of f. A function symbol is labelled if S_f contains more than one element. For unlabelled f, the set S_f containing only one element can be left implicit; in that case we write f instead of f_p. Choose for $f : \langle i_1, \ldots, i_l \rangle \in \Sigma$, a *sort map* that is a morphism of $\mathbf{Set}^{\mathbb{F}}$ defined by $\langle\!\langle - \rangle\!\rangle^f : \delta^{i_1} M \times \cdots \times \delta^{i_l} M \longrightarrow K_{S_f}$ where $K_{S_f} \in \mathbf{Set}^{\mathbb{F}}$ is the constant presheaf defined by $K_{S_f}(n) = S_f$. The sort map was originally called a projection, denoted by π_f in [17]. Then, as in the case of ordinary signature, we define $M_{\overline{\Sigma}} Z$ by the presheaf of all meta-terms generated by the labelled signature $\overline{\Sigma}$. Let $\phi : Z \longrightarrow M$ be an assignment. The *labelling map* $\phi^{\mathsf{L}} : M_{\Sigma} Z \longrightarrow M_{\overline{\Sigma}} Z$ is a morphism of $\mathbf{Set}^{\mathbb{F}}$ defined by $\phi^{\mathsf{L}}_n(x) = x,\ \phi^{\mathsf{L}}_n(\mathrm{z}[\vec{t}]) = \mathrm{z}[\phi^{\mathsf{L}}_n \vec{t}],\ \phi^{\mathsf{L}}_n(f(n{+}\vec{i_1}.t_1, \ldots, n{+}\vec{i_l}.t_l)) = f_{\langle\!\langle \phi^*_{n+i_1}(t_1), \ldots, \phi^*_{n+i_l}(t_l) \rangle\!\rangle^f_n}(n{+}\vec{i_1}.\phi^{\mathsf{L}}_{n+i_1} t_1, \ldots, n{+}\vec{i_l}.\phi^{\mathsf{L}}_{n+i_l} t_l)$. For a given CRS $(\Sigma, \mathcal{R})$, we define the labelled rules by $\overline{\mathcal{R}} = \{Z \vdash \vec{n}.\phi^{\mathsf{L}}_n l \to \vec{n}.\phi^{\mathsf{L}}_n r \mid Z \vdash \vec{n}.l \to \vec{n}.r \in \mathcal{R},\ \text{assignment}\ \phi : Z \longrightarrow M\}$. Thus $\overline{\mathcal{R}}$ is a set of rewrite rules on labelled terms in $M_{\overline{\Sigma}} Z(0)$. So, $(\overline{\Sigma}, \overline{\mathcal{R}})$ forms a CRS that gives rewriting on $\overline{\Sigma}$-terms. The labelling map ϕ^{L} preserves $\mathcal{R}$-rewrite structures, hence we have the main theorem of this paper.

Proposition 1. *Let M be a model of $(\Sigma, \mathcal{R})$. If we have CRS rewriting $n \vdash s \to_{\mathcal{R}} t$ on $M_{\Sigma} 0_n$, then for the assignment $\phi : 0 \longrightarrow M$, we have rewriting $n \vdash \phi^{\mathsf{L}}_n s \to_{\overline{\mathcal{R}}} \phi^{\mathsf{L}}_n t$ on $M_{\overline{\Sigma}} 0_n$.*

Theorem 2 (Higher-order semantic labelling). *Let M be a model of a CRS $(\Sigma, \mathcal{R})$. Then, $(\Sigma, \mathcal{R})$ is terminating if and only if $(\overline{\Sigma}, \overline{\mathcal{R}})$ is terminating.*

We need separately a way to prove termination of the labelled system. For this purpose, we use Blanqui's version of General Schema for "new definition of IDTS" ([6] Def. 1). Structural CRSs used in the present paper can be seen as a subclass of Blanqui's IDTSs. Hence we can use General Schema as a criteria of termination of structural CRSs. In our experience, this is the most powerful decidable method to prove termination of CRSs. General Schema uses a *precedence* which is a partial order on function symbols occurring in a CRS.

Example 3 (CRS for prefix sum). Consider the following CRS $\mathcal{P}$ for computing the prefix sum of a list i.e. the list with the sum of all prefixes of a given list using the higher-order function map [7].

$$
\begin{array}{llll}
\mathsf{map}(a.\mathrm{F}[a], \mathsf{nil}) & \to \mathsf{nil} & \mathsf{ps}(\mathsf{nil}) & \to \mathsf{nil} \\
\mathsf{map}(a.\mathrm{F}[a], \mathrm{x} : \mathrm{xs}) & \to \mathrm{F}[\mathrm{x}] : \mathsf{map}(a.\mathrm{F}[a], \mathrm{xs}) & \mathsf{ps}(\mathrm{x} : \mathrm{xs}) & \to \mathrm{x} : \mathsf{ps}(\mathsf{map}(a.\mathrm{x} + a, \mathrm{xs}))
\end{array}
$$

Unfortunately, General Schema *cannot* show termination of the CRS $\mathcal{P}$ because the argument of ps in the right-hand side of the last rule is not a subterm of the argument of ps in the left-hand side. So we use higher-order semantic labelling. The CRS $\mathcal{P}$ is formulated under the binding signature $\Sigma = \{\mathsf{map} : \langle 1,0\rangle, \mathsf{S}, \mathsf{ps} : \langle 0\rangle, 0, \mathsf{nil} : \langle\rangle, +, \text{“ : ”} : \langle 0,0\rangle\}$. We take the presheaf $\mathcal{M}_n \triangleq (\mathbb{N}^n \to \mathbb{N})$ of all functions on $\mathbb{N}$ for a model of $\mathcal{P}$. The operations are defined by $\mathsf{map}_{\mathcal{M}_0}(f,y) = y,\ \mathsf{ps}(x) = x,\ :_{\mathcal{M}_0}(x,y) = y+1,\ \mathsf{nil}_{\mathcal{M}_0} = 0,\ x +_{\mathcal{M}_0} y = 0$. The idea of this model is “to count the number of cons's”. We label the function symbol ps and assume that other function symbols are unlabelled. We use the natural numbers $\mathbb{N}$ as the sort set S_{ps}. The sort map is defined by $\langle\!\langle x\rangle\!\rangle_0^{\mathsf{ps}} = x$. Then, we have the labelled rules $\mathsf{ps}_0(\mathsf{nil}) \to \mathsf{nil},\ \mathsf{ps}_{i+1}(\mathrm{x} : \mathrm{xs}) \to \mathrm{x} : \mathsf{ps}_i(\mathsf{map}(a.\mathrm{x}+a, \mathrm{xs}))$ for all $i \in \mathbb{N}$. General Schema succeeds in showing termination of this labelled CRS with the precedence $\mathsf{ps}_i > \mathsf{ps}_j > \mathsf{map} > \mathsf{nil}, :$ for $i > j \in \mathbb{N}$.

The quasi-model version of the semantic labelling theorem is obtained similarly. Define the CRS Decr (called “decreasing rules”) over $\overline{\Sigma}$ by $f_p(\vec{i_1}.\mathrm{z}_1[\vec{i_1}], \ldots, \vec{i_1}.\mathrm{z}_l[\vec{i_l}]) \to f_q(\vec{i_1}.\mathrm{z}_1[\vec{i_1}], \ldots, \vec{i_1}.\mathrm{z}_l[\vec{i_l}])$ for all $f : \langle i_1, \ldots, i_l\rangle \in \Sigma$ and all $p >_S q \in S_f$, where $>_S$ denotes the strict part of $\geq_S$.

Theorem 4. *Let M be a quasi-model for a CRS $(\Sigma, \mathcal{R})$ and $(\overline{\Sigma}, \overline{\mathcal{R}})$ the labelled CRS with respect to M. Then $(\Sigma, \mathcal{R})$ is terminating if and only if $(\overline{\Sigma}, \overline{\mathcal{R}} \cup \mathsf{Decr})$ is terminating.*

Example 5 (Monad for the lambda calculus). Consider the following CRS $\mathcal{R}$ under the binding signature $\Sigma = \{\mathsf{lift}, \mathsf{bind} : \langle 1,0\rangle, \mathsf{app}\langle 0,0\rangle, \mathsf{abs} : \langle 1\rangle, \mathsf{var}, \mathsf{old} : \langle 0\rangle, \mathsf{new} : \langle\rangle\}$ and the metavariables $Z = \{\mathrm{F}^1, \mathrm{G}^1, \mathrm{X}^0, \mathrm{S}^0, \mathrm{T}^0\}$. This is the single base type version of the rewrite system of multiplication operation of the monad of λ-calculus in [1].

$$\mathsf{lift}(\mathrm{F}, \mathsf{new}) \to \mathsf{var}(\mathsf{new}) \qquad \mathsf{lift}(\mathrm{F}, \mathsf{old}(\mathrm{X})) \to \mathsf{bind}(a.\mathsf{var}(\mathsf{old}(a)), \mathrm{F}[\mathrm{X}])$$
$$\mathsf{bind}(\mathrm{F}, \mathsf{var}(\mathrm{X})) \to \mathrm{F}[\mathrm{X}] \qquad \mathsf{bind}(\mathrm{F}, \mathsf{app}(\mathrm{S}, \mathrm{T})) \to \mathsf{app}(\mathsf{bind}(\mathrm{F}, \mathrm{S}), \mathsf{bind}(\mathrm{F}, \mathrm{T}))$$
$$\mathsf{bind}(\mathrm{F}, \mathsf{abs}(a.\mathrm{G}[a])) \to \mathsf{abs}(\ a.\, \mathsf{bind}(b.\mathsf{lift}(\mathrm{F}, b), \mathrm{G}[a])\ )$$

It is not straightforward to show termination of this system using known criteria in higher-order rewriting. In the last rule, bind decomposes the abs construct, but in its right-hand side, the recursive call of bind happens with the argument $b.\mathsf{lift}(\mathrm{F}, b)$ which is structurally bigger than F in the left-hand side. Moreover, lift is defined by using bind, which seems to be circularity. Due to these reasons, most of RPO-like syntactical methods for termination of higher-order rewriting fail. When we try to show termination of $\mathcal{R}$ using General Schema, we cannot determine the precedence between the function symbols lift and bind. This is the same as for the corresponding rewrite rules written in other formats of higher-order rewriting and termination criteria for them. For the higher-order rewrite system corresponding to $\mathcal{R}$ in the format Inductive Data Type Systems [5], the higher-order RPO [12] fails to show termination. For the S-expression rewrite system corresponding to $\mathcal{R}$, the lexicographic path order described in [16] fails to show termination. For the simply-typed term rewriting system (STTRS) corresponding to $\mathcal{R}$, the dependency pair technique for STTRSs [2] fails to show termination.

Termination proof. We take the carrier $\mathcal{M}$ to be the presheaf of *strictly monotone functions* on $\mathbb{N}$ equipped with the usual order $\geq$ and its pointwise extension, i.e. $\mathcal{M}_n = [\mathbb{N}^n \Rightarrow \mathbb{N}]$ where $[- \Rightarrow -]$ denotes the set of all strictly monotone functions. This $\mathcal{M}$ also forms a monoid in $\mathbf{Set}^{\mathbb{F}}$ by taking the multiplication as the composition and the unit as the projections. The Σ-algebra structure on $\mathcal{M}$ is defined by $\mathsf{lift}_{\mathcal{M}_0}(f, x) = f(0)$, $\mathsf{bind}_{\mathcal{M}_0}(f, x) = f(x){+}1$, $\mathsf{new}_{\mathcal{M}_0} = 0$, $\mathsf{old}_{\mathcal{M}_0}(x) = 0$, $\mathsf{var}_{\mathcal{M}_0}(x) = x$, $\mathsf{app}_{\mathcal{M}_0}(x, y) = max(x, y)$, $\mathsf{abs}_{\mathcal{M}_0}(g) = 1$. One can easily check that $\mathcal{M}$ with these operations is indeed a quasi-model for $\mathcal{R}$. We only label the function symbol bind. We choose the sort set $S_{\mathsf{bind}} = \mathbb{N}$ and the weakly monotone sort map as $\langle\!\langle -, - \rangle\!\rangle_0^{\mathsf{bind}} : [\mathbb{N} \Rightarrow \mathbb{N}] \times \mathbb{N} \longrightarrow \mathbb{N}$, $\langle\!\langle f, x \rangle\!\rangle_0^{\mathsf{bind}} = f(x)$. Using this, we have the following labelled CRS $\overline{\mathcal{R}} \cup \mathsf{Decr}$ for all strictly monotone functions $f \in [\mathbb{N} \to \mathbb{N}]$.

$$
\begin{array}{lll}
\mathsf{lift}(\textsc{f}, \mathsf{old}(\textsc{x})) & \to \mathsf{bind}_0(a.\mathsf{var}(\mathsf{old}(a)), \textsc{f}[\textsc{x}]) & \\
\mathsf{bind}_{f(x)}(\textsc{f}, \mathsf{var}(\textsc{x})) & \to \textsc{f}[\textsc{x}] & \text{for all } x \in \mathbb{N} \\
\mathsf{bind}_{f(s)}(\textsc{f}, \mathsf{app}(\textsc{s}, \textsc{t})) & \to \mathsf{app}(\mathsf{bind}_{f(s)}(\textsc{f}, \textsc{s}), \mathsf{bind}_{f(t)}(\textsc{f}, \textsc{t})) & \text{for all } s \geq t \in \mathbb{N} \\
\mathsf{bind}_{f(t)}(\textsc{f}, \mathsf{app}(\textsc{s}, \textsc{t})) & \to \mathsf{app}(\mathsf{bind}_{f(s)}(\textsc{f}, \textsc{s}), \mathsf{bind}_{f(t)}(\textsc{f}, \textsc{t})) & \text{for all } s < t \in \mathbb{N} \\
\mathsf{bind}_{f(1)}(\textsc{f}, \mathsf{abs}(a.\textsc{g}[a])) & \to \mathsf{abs}(\ a.\,\mathsf{bind}_{f(0)}(b.\mathsf{lift}(\textsc{f}, b), \textsc{g}[a])\ ) & \\
\mathsf{bind}_i(\textsc{f}, \textsc{x}) & \to \mathsf{bind}_j(\textsc{f}, \textsc{x}) & \text{for all } i > j \in \mathbb{N}
\end{array}
$$

With the precedence $\mathsf{bind}_i > \mathsf{bind}_j > \mathsf{lift} > \mathsf{bind}_0 > \mathsf{var}, \mathsf{app}, \mathsf{abs}, \mathsf{old}, \mathsf{new}$ for all $i > j > 0$, General Schema succeeds in showing the termination of $\overline{\mathcal{R}} \cup \mathsf{Decr}$. Hence, by Thm. 2, we conclude termination of $\mathcal{R}$.

References

[1] T. Altenkirch and B. Reus. Monadic presentations of lambda terms using generalized inductive types. In *Proc. of 13th Int. Wksh. on Computer Science Logic, CSL '99*, LNCS 1683, pages 453–468, 1999.

[2] T. Aoto and T. Yamada. Dependency pairs for simply typed term rewriting. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA 2005)*, Lecture Notes in Computer Science 3467, pages 120–134, 2005.

[3] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[4] F. Blanqui, J.-P. Jouannaud, and M. Okada. The calculus of algebraic constructions. In *Rewriting Techniques and Applications (RTA 1999)*, LNCS 1631, pages 301–316. Springer, 1999.

[5] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive data type systems. *Theoretical Computer Science*, 272:41–68, 2002.

[6] Frederic Blanqui. Termination and confluence of higher-order rewrite systems. In *Rewriting Techniques and Application (RTA 2000)*, LNCS 1833, pages 47–61. Springer, 2000.

[7] Cristina Borralleras and Albert Rubio. A monotonic higher-order semantic path ordering. In *Procs. 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, LNCS 2250, pages 531–547, 2001.

[8] O. Danvy and K.H. Rose. Higher-order rewriting and partial evaluation. In *Rewriting Techniques and Applications, 9th International Conference, (RTA'98)*, LNCS 1379, 1998.

[9] M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proc. 14th Annual Symposium on Logic in Computer Science*, pages 193–202, 1999.

[10] M. Hamana. Free Σ-monoids: A higher-order syntax with metavariables. In *Asian Symposium on Programming Languages and Systems (APLAS 2004)*, LNCS 3302, pages 348–363, 2004.

[11] M. Hamana. Universal algebra for termination of higher-order rewriting,. In *Proceedings of 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, LNCS 3467, pages 135–149. Springer, 2005.

[12] J.-P. Jouannaud and A. Rubio. The higher-order recursive path ordering. In *Proc. LICS'99*, pages 402–411. IEEE, 1999.

[13] J.W. Klop. *Combinatory Reduction Systems*. PhD thesis, CWI, Amsterdam, 1980. volume 127 of Mathematical Centre Tracts.

[14] J.W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: Introduction and survey. *Theor. Comput. Sci.*, 121(1&2):279–308, 1993.

[15] Tobias Nipkow. Higher-order critical pairs. In *Proc. 6th IEEE Symp. Logic in Computer Science*, pages 342–349, 1991.

[16] Y. Toyama. Termination of S-expression rewriting systems: Lexicographic path ordering for higher-order terms. In *Proceedings of the 15th International Conference on Rewriting Techniques and Applications (RTA 2004)*, Lecture Notes in Computer Science 3091, pages 40–54, 2004.

[17] H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24(1/2):89–105, 1995.