Semantic Labelling for Proving Termination of Combinatory Reduction Systems

Makoto Hamana

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

Abstract. We give a novel transformation method for proving termination of higher-order rewrite rules in Klop's format called Combinatory Reduction System (CRS). The format CRS essentially covers the usual pure higher-order functional programs such as Haskell. Our method called higher-order semantic labelling is an extension of a method known in the theory of term rewriting. This attaches semantics of the arguments to each function symbol. We systematically define the labelling by using the complete algebraic semantics of CRS, Σ-monoids. We also examine the power of higher-order semantic labelling by several examples. This includes an interesting example from the viewpoint of functional programming.

1 Introduction

Rewrite rules appear everywhere in computer science. In programming language theory, we use often transformation of states, expressions, terms, or programs given by some form of rewrite rules. Functional programs such as Haskell can also be regarded as rewrite rules. When reasoning with such rewrite rules, termination is one of the most important property, because it is necessary for decidable equality checking. This topic has been extensively investigated in the field of term rewriting [BN98, Ter03].

In this paper, we deal with higher-order rewrite rules in Klop's format called combinatory reduction systems (CRSs) [Klo80, KOR93]. The format CRS is known as one of the most early detailed formulation of higher-order rewriting systems (i.e. rewriting systems having the feature of *variable binding* and meta-level substitutions) in the theory of term rewriting. A CRS is a set of rewrite rules on second-order terms. We give a method to prove termination, meaning *strong normalisation*, of a CRS by a translation called *higher-order semantic labelling*. This is an extension of a method for first-order term rewriting systems (TRSs) [Zan95].

Weakness of existing syntactic methods. Higher-order extensions of term rewriting systems [Ter03] are known as several formats: major representatives are CRSs, Higher-order Rewrite Systems [Nip91], and Inductive Data Type Systems [BJO02]. There exist several termination criteria: higher-order recursive path order (HORPO) [JR07], the General Schema [BJO02, Bla00], hereditary monotone functional interpretation [Pol94], binding algebra interpretation [Ham05]. Recently improvements of HORPO/General Schema are actively investigated [BR01, Raa01, JR06]. A recent survey on this field can be found in [BJR08].

S.Escobar (Ed.): WFLP 2009, LNCS 5979, pp. 62–78, 2010.

For CRSs, the General Schema is a decidable syntactic criteria of termination [Bla00]. The idea of the General Schema is to control the arguments of the right-hand side recursive calls in a rewrite rule by checking that they are smaller than the left-hand sides ones in the strict subterm order extended in a multiset or lexicographic manner.

The General Schema criteria is effective for rewrite rules defined *structural recursively* on term structures. However, there are many realistic rewrite rules which do not fit into this scheme. We use often rewrite rules defined *non-structural recursively* on term structures. The General Schema is *not* applicable to prove termination of such rewrite rules. What does this mean? This implies that rather than syntactic structures, *semantic structures* should be more explicit in many situations, and they can be a "hint" of complete termination proofs. Our approach to give termination proofs in this paper does not aim to be fully automatic. Finding appropriate semantics of rewrite rules automatically is hard in general. But in most cases, *one has the intended semantics for rewrite rules in one's mind that matches the intended application* (if not, why one could write such rewrite rules?) Hence, making such semantics explicit is merely a matter of formulation. Notice that our method is not fully semantical either. We combine both syntactic and semantical information. Below, we give two examples to illustrate this situation.

Example 1 (The prefix sum of a list). 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 (taken from [BR01]).

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

We want to prove termination of the CRS $\mathcal{P}$. Unfortunately, the CRS $\mathcal{P}$ does not follow the General Schema, hence the exiting syntactic method is not powerful enough. This is 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. However, we know that the map function does not change the length of a list, thus a shorter list than $x : xs$ is always used in the recursive call of ps. To prove termination of ps, this "semantic" information (rather than only syntactical structures) should be effectively used.

Higher-order semantic labelling developed in this paper solves this problem. It is a method to reflect such information in rewrite rules. In this case, we use the "length" of a list for ps as the semantics. Higher-order semantic labelling transforms the original CRS $\mathcal{P}$ to the following labelled CRS:

$$\begin{aligned}
\mathsf{ps}_0(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{ps}_{i+1}(\mathrm{x} : \mathrm{xs}) &\to \mathrm{x} : \mathsf{ps}_i(\mathsf{map}(a.\mathrm{x} + a, \mathrm{xs}))
\end{aligned}$$

where $i \in \mathbb{N}$. This i denotes the "semantics" of argument ps, i.e., the length of a list. This transformation to attach semantics to function symbols is systematically defined in the structure of Σ-*monoids*, which is abstract algebraic structures of higher-order terms. Then, this labelled CRS successfully follows the General Schema with the precedence

$\mathsf{ps}_i > \mathsf{ps}_j > \mathsf{map} >$: for $i > j \in \mathbb{N}$. This ordering is used to compare two term structures in a recursive manner. If two root function symbols can be compared by the ordering, thier arguments need not to be comparied. Hence, the order $\mathsf{ps}_{i+1} > \mathsf{ps}_i$ effectively solve the above mentioned problem on the mismatch between the term size and semantical size. Our main theorem (Thm. 10) of higher-order semantic labelling is that if the labelled CRS is terminating, then the original CRS is terminating. Hence, we can conclude termination of the CRS $\mathcal{P}$.

Example 2 (Haskell's rewrite rules). Glasgow Haskell Compiler (GHC) has a pragma called "rewrite rules" [JTH01] for an optimization purpose. GHC applies rewrite rules to the source program wherever it can. The following is an example that composes two `map`s in a program.

```
{-# RULES
  "map/map" forall f g xs.  map f (map g xs) = map (f.g) xs
#-}
```

Rewriting by this Haskell's rewrite rule are expected to be terminating at the compile time. But GHC makes no attempt to ensure that the rule is terminating because of complications of the combination of the rewrite rule and compiler's optimization rules [JTH01]. But ideally we should ensure termination of rewrite rules. To consider this problem in a formal setting, we model this Haskell's rewrite rule as the following CRS's rewrite rule:

$$\mathsf{map}(\mathrm{F}, \mathsf{map}(\mathrm{G}, \mathrm{xs})) \to \mathsf{map}(a.\mathrm{F}[\mathrm{G}[a]], \mathrm{xs})$$

A difficulty is that this rule is not defined structural recursively on a data structure. Moreover, the first argument of map in the right-hand side is bigger than ones in the left-hand side. Hence, this does not follow the General Schema.

Higher-order semantic labelling again solves this problem. We use "the number of maps" in the second argument of map as the semantics. Higher-order semantic labelling transforms the original rule to the following labelled rules:

$$\begin{array}{lll} \mathsf{map}_{i+1}(\mathrm{F}, \mathsf{map}_i(\mathrm{G}, \mathrm{xs})) & \to \mathsf{map}_i(a.\mathrm{F}[\mathrm{G}[a]], \mathrm{xs}) & \text{for all } i \in \mathbb{N} \\ \mathsf{map}_i(\mathrm{F}, \mathrm{xs}) & \to \mathsf{map}_j(\mathrm{F}, \mathrm{xs}) & \text{for all } i > j \in \mathbb{N} \end{array}$$

Each label denotes the number of maps in the second argument. The General Schema succeeds in showing termination of the rules with the precedence $\mathsf{map}_i > \mathsf{map}_j$ for all $i > j \in \mathbb{N}$. Hence by our main theorem, we conclude termination of the original map's rule..

These two examples use seemingly simple semantics, i.e., "the length of lists" and "the number of maps". But to compute them actually needs a sophisticated semantical account because the rewrite rules involve higher-order functions. Moreover, semantics need not to be numbers or "sizes". Arbitrary (higher-order) algebraic structures (e.g. λ-terms, domains, categories, etc.) can be semantics of a CRS. In other words, the semantic information for labels cannot be obtained by just syntactic counting of symbols. In this paper, using the complete algebraic semantics Σ-monoid, we systematically give higher-order semantic labelling for CRSs.

Contribution. The contribution of this paper is summarised as follows.

(i) Theoretical contribution.
- We generalised semantics labelling for TRSs [Zan95] to higher-order semantic labelling for CRSs in the framework of Σ-monoids. This also showed that Σ-monoids was certainly the right structure as the semantics of CRSs.
- We showed that semantic labelled meta-terms form a Σ-monoid.
- We identified the commutativity of the labelling operation with the substitutions appearing in formulation of CRSs is an essential property to establish semantic labelling.

(ii) Practical contribution. We demonstrate higher-order semantic labelling by several examples for which the General Schema alone fails.

Background. Semantic labelling on higher-order terms has been defined for Inductive Datatype Systems [Ham07]. The present paper much simplifies the labelling method to deal with CRSs. We also aim to apply it to examples taken from functional programming. The semantics used in this paper is based on the algebraic semantics of CRS Σ-monoids. The notion of Σ-monoids was introduced by Fiore, Plotkin and Turi [FPT99], then a higher-order abstract syntax for free Σ-monoids was developed by the author [Ham04]. The algebraic semantics for CRSs [Ham05] was an application of this Σ-monoid structure. The outline of semantic labelling for CRSs (without proofs) was presented at 13th International Conference on Logic for Programming Artificial Intelligence Reasoning (LPAR'06) as a short paper.

How to read the paper. Theories on term rewriting usually avoid the use of semantics as much as possible. In contrast to it, we rely on the semantics of higher-order terms and rewriting. The semantics structure Σ-monoid is a natural extension of the first-order universal algebra to the second-order setting by shifting the base category from **Set** to a presheaf category [FPT99]. It is systematically defined in the framework of categorical universal algebra. Why categorical notions are needed is to make definitions and discussions on higher-order rewriting mathematically simple, manageable and systematic. The seemingly "elementary" extension of first-order semantic structures to higher-order setting by hand (within ordinary set-theoretic setting) makes definitions and theories quite complex because of the combinations of ordinary first-order structures and higher-order structures. Category theory prevents this explosion by giving a right abstraction for algebraic and higher-order terms. Hence, this paper assumes basic knowledge of category theory for reading the development of the semantic labelling method, such as functor categories, monoidal categories, monoids and algebras (e.g. [Mac71] Chap. II, VII).

Future work. As a future work, we plan to make our method to be more accessible for users of proof assitants and dependently-typed programming languages. One of the most expected area that seriously needs termination proofs is proof assitants. Finding appropriate semantics fully automatic is impossible, but one's intended semantics might be directly mechanised within a proof assitant such as Coq or Agda. Combining it with syntactic methods will greatly reduce efforts to give full termination proofs in proof assitants. Hence, giving a convenient library and recipes for higher-order semantic labelling in a proof assitant might be interesting.

Organisation. This paper is organised as follows. We first review the definition of CRSs in Section 2 and the semantics of CRSs in Section 3. We give higher-order semantic labelling of CRSs in Section 4. In Section 5, we give the quasi-model version of higher-order semantic labelling and show several examples of termination proof using our method. All omitted proofs are given in Appendix.

2 Combinatory Reduction Systems

CRS. We review the definition of CRSs. We use the definition of the standard reference [KOR93] of CRSs with a slight modification of syntax used in [DR98]: $-.-$ and $-[-]$ instead of ordinary ones $[-]-$ and $-(-)$ in [KOR93].

Assume a signature Σ of function symbols f^l with arity, metavariables z^l with arity (in both cases the superscript $l \in \mathbb{N}$ is the arity).

(i) CRS *terms* have the form $t ::= x \mid x.t \mid f^l(t_1, \ldots, t_l)$. These forms are respectively called *variables*, *abstractions*, and *function terms*.

(ii) CRS *meta-terms* extend CRS terms to $t ::= x \mid x.t \mid f^l(t_1, \ldots, t_l) \mid \mathrm{z}^l[t_1, \ldots, t_l]$. The last form is called a *meta-application*.

(iii) A *valuation* θ is a mapping that assigns to n-ary metavariable z an n-ary *substitute* (a meta-level lambda notation, cf. [KOR93]) $\theta : \mathrm{z} \longmapsto \underline{\lambda}(x_1, \ldots, x_n).t$ where t is a term. Any valuation is extended to a function on meta-terms:

$$\begin{array}{ll} \theta(x) = x & \theta(f(t_1, \ldots, t_l)) = f(\theta(t_1), \ldots, \theta(t_l)) \\ \theta(x.t) = x.\theta(t) & \theta(\mathrm{z}[t_1, \ldots, t_l]) = \theta(\mathrm{z})\,(\theta(t_1), \ldots, \theta(t_l)) \end{array} \quad (1)$$

Note that the right-hand side of the equation (1) uses an application at the meta-level to the substitute. The valuation is *safe* if there are no two substitutes $\theta(\mathrm{z})$ and $\theta(\mathrm{z}')$ such that $\theta(\mathrm{z})$ contains a free variable x which appears also bound in $\theta(\mathrm{z}')$.

(iv) CRS *rules*, written $l \to r$, consist of two meta-terms l and r with the following additional restrictions:

(iv-a) l and r are closed (w.r.t. variables) meta-terms,

(iv-b) l must be a "pattern", i.e. a function term where all meta-applications have the form $\mathrm{z}[x_1, \ldots, x_n]$ with distinct variables x_i,

(iv-c) r can only contain meta-applications with meta-variables occurring in the left-hand side.

The rewrite rule $l \to r$ is *safe for* θ , if for all z in l and r, the substitute $\theta(\mathrm{z})$ does not have a free variable x occurring in an abstraction $x.-$ of l and r. A set of rewrite rules under the signature Σ is called a CRS and denoted by $(\Sigma, \mathcal{R})$ or simply $\mathcal{R}$.

(v) The CRS *rewrite relation* $\to_{\mathcal{R}}$ is generated by context and safe valuation closure of a given CRS $\mathcal{R}$:

$$\frac{l \to r \in \mathcal{R}}{\theta(l) \to_{\mathcal{R}} \theta(r)}\ \text{safe } \theta \qquad \frac{s \to_{\mathcal{R}} t}{x.s \to_{\mathcal{R}} x.t} \qquad \frac{s \to_{\mathcal{R}} t}{f(\ldots, s, \ldots) \to_{\mathcal{R}} f(\ldots, t, \ldots)}$$

where $l \to r$ must be safe for the safe valuation θ. The third rule means rewriting at the i-th argument of f. We say that $\mathcal{R}$ is *terminating* if $\to_{\mathcal{R}}$ is well-founded.

Structural CRSs. In this paper, we treat CRSs using (meta-)terms built from binding signatures, which we call *structural CRSs* (cf. Aczel's contraction schemes [Acz78]). A *binding signature* Σ consists of a set Σ of function symbols with an arity function $a : \Sigma \to \mathbb{N}^*$, where $\mathbb{N}^*$ 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 ($1 \leq i \leq l$). For a formal treatment of named variables modulo α-equivalence in CRSs, we assume the method of de Bruijn levels [dB72] for the naming convention of variables (N.B. not for metavariables) 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. *Structural meta-terms* are of the form $t ::= x \mid f(x_1 \cdots x_{i_1}.t_1, \ldots, x_1 \cdots x_{i_r}.t_l) \mid \mathrm{z}^l[t_1, \ldots, t_l]$ satisfying the restriction generated by the inference system given blow. Fix an $\mathbb{N}$-indexed set Z of metavariables defined by $Z(l) \triangleq \{\mathrm{z} \mid \mathrm{z} \text{ has arity } l\}$. A meta-term t is *structural* if $n \vdash t$ is derived from the following rules for some $n \in \mathbb{N}$.

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

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

By using these rules, we obtain meta-terms in the method of de Bruijn levels. A rewrite rule $1.\cdots n.l \to 1.\cdots n.r$ is called structural if l and r are structural, i.e. $n \vdash l$ and $n \vdash l$. A CRS is structural if all rules are structural. A valuation θ is structural if for any mapping by $\theta : \mathrm{z} \mapsto \underline{\lambda}(x_1, \ldots, x_n).t$, t is a structural term and all variables in t are included in $x_1, \ldots, x_n$. 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 the other part is not important.

3 Semantics of CRSs

3.1 Binding Algebras

The semantics of CRS is given by the notion of binding algebras and Σ-monoids. What are Σ-monoids? A Σ-monoid is an algebra equipped with *substitution operation* on (semantics of) terms. This substitution operation is called *multiplication*, typically denoted by β in this paper. Why this is a multiplication is that the substitution operation satisfies the monoid law (imagine compositions of substitutions with the identity substitution) in an abstract setting.

We review the notion of binding algebras and Σ-monoids. For detail, see [FPT99]. 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. 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 A of $\mathbf{Set}^{\mathbb{F}}$ is often called a *presheaf*. Subscripts may be used to denote parameters. 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 $\Sigma A = \coprod_{f:\langle n_1,\ldots,n_l\rangle \in \Sigma} \prod_{1 \leq i \leq l} \delta^{n_i} A$. A Σ-*algebra*

is a pair (A, α) consisting of a presheaf $A \in \mathbf{Set}^{\mathbb{F}}$ called a *carrier* and a map ([ ] denotes a copair of coproducts) $\alpha = [f_A]_{f\in\Sigma} : \Sigma A \longrightarrow A$ called an *algebra structure*, where f_A is an *operation* $f_A : \delta^{n_1}A \times \ldots \times \delta^{n_l}A \longrightarrow A$ defined for each function symbol $f : \langle n_1, \ldots, n_l\rangle \in \Sigma$. The "presheaf of variables" $\mathrm{V} \in \mathbf{Set}^{\mathbb{F}}$ is defined by $\mathrm{V}(n) = n$, $\mathrm{V}(\rho) = \rho\,(\rho : m \to n \text{ in } \mathbb{F})$. For presheaves A and B, $(A \bullet B)(n) \triangleq (\coprod_{m\in\mathbb{N}} A(m) \times B(n)^m)/\sim$ where $\sim$ is the equivalence relation generated by $(t; u_{\rho 1}, \ldots, u_{\rho m}) \sim (A(\rho)(t); u_1, \ldots, u_l)$ for $\rho : m \to l$ in $\mathbb{F}$. Then, $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathrm{V})$ forms a monoidal category [Mac71], where the "substitution" monoidal product is defined as follows. An element of $A(m) \times B(n)^m$ is denoted by $(t; u_1, \ldots, u_m)$ where $t \in A(m)$ and $u_1, \ldots, u_m \in B(m)$. A representative of an equivalence class in $A \bullet B(n)$ is also denoted by this notation. Let Σ be a signature functor with strength st defined by a binding signature. A Σ-*monoid* $M = (M, \alpha, \eta, \mu)$ consists of a *monoid* $(M, \eta : \mathrm{V} \to M, \mu : M \bullet M \to M)$ in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathrm{V})$ with a Σ algebra structure $\alpha : \Sigma M \to M$ satisfying $\mu \circ (\alpha \bullet \mathrm{id}_M) = \alpha \circ \Sigma\mu \circ \mathrm{st}$. A Σ-*monoid morphism* $M \longrightarrow M'$ is a morphism in $\mathbf{Set}^{\mathbb{F}}$ which is both Σ-algebra homomorphism and monoid morphism.

3.2 Algebra of Meta-terms

Let Z be an arbitrary $\mathbb{N}$-indexed set of metavariables (cf. Sec. 2). The *presheaf* $M_\Sigma Z$ *of meta-terms* is defined by $M_\Sigma Z(n) = \{t \mid n \vdash t\}$. We abbreviate $n{+}1, \ldots, n{+}k.t$ to $n{+}\vec{k}.t$. For every $f : \langle i_1, \ldots, i_l\rangle \in \Sigma$, we define the map $f_T : \delta^{i_1} M_\Sigma Z \times \cdots \times \delta^{i_l} M_\Sigma Z \longrightarrow M_\Sigma Z$ in $\mathbf{Set}^{\mathbb{F}}$ by $(t_1, \ldots, t_l) \longmapsto f(n{+}\vec{i_1}.t_1, \ldots, n{+}\vec{i_l}.t_l)$. The *multiplication* $\beta : M_\Sigma Z \bullet M_\Sigma Z \longrightarrow M_\Sigma Z$ is a map in $\mathbf{Set}^{\mathbb{F}}$ that performs a substitution of variables defined inductively as follows.

$$\beta(n)(i; \vec{t}) = t_i \qquad \beta(n)(\mathrm{z}[s_1, \ldots, s_l]; \vec{t}) = \mathrm{z}[\beta(n)(s_1; \vec{t}), \ldots, \beta(n)(s_l; \vec{t})]$$

$$\beta(n)(f(m{+}\vec{i_1}.s_1, \ldots, m{+}\vec{i_l}.s_l); \vec{t}) = f(m{+}\vec{i_1}.\beta(m{+}i_1)(s_1; \mathsf{up}_{i_1}(\vec{t}), m{+}1, \ldots, m{+}i_1), \ldots \\ m{+}\vec{i_l}.\beta(m{+}i_l)(s_l; \mathsf{up}_{i_l}(\vec{t}), m{+}1, \ldots, m{+}i_l)$$

where $f : \langle i_1, \ldots, i_l\rangle \in \Sigma$ and $\vec{t}$ denotes $t_1, \ldots, t_m$, and the weakening map from $M_\Sigma Z(m)$ to $M_\Sigma Z(m + i)$ is defined by $\mathsf{up}_i \triangleq M_\Sigma Z(\mathrm{id}_m + \mathsf{w}_i)$ where $\mathsf{w}_i : 0 \to i$. Then, the structural meta-terms $(M_\Sigma Z, [f_T]_{f\in\Sigma}, \nu, \beta)$ is a *free* Σ-*monoid* over a presheaf $\hat{Z}$, where $\nu : \mathrm{V} \longrightarrow M_\Sigma Z$ in $\mathbf{Set}^{\mathbb{F}}$ is defined by $x \longmapsto x$ and $\hat{Z}(n) = \coprod_{k\in\mathbb{N}} \mathbb{F}(k, n) \times Z(k)$ [Ham04]. Hereafter, given $\mathbb{N}$-indexed set Z, we abuse the notation to use Z to denote its presheaf version $\hat{Z} \in \mathbf{Set}^{\mathbb{F}}$ in an assignment.

Definition 3. We call an *assignment* a morphism $\phi : Z \longrightarrow A$ of $\mathbf{Set}^{\mathbb{F}}$ whose target A has a Σ-monoid structure (A, α, η, μ). By freeness, an assignment $\phi : Z \longrightarrow A$ is extended to the Σ-*monoid morphism* $\phi^* : M_\Sigma Z \longrightarrow A$ defined by

$$\phi^*_n(x) = \eta_n(x) \qquad (x \in n)$$
$$\phi^*_n(f(n{+}\vec{i_1}.t_1, \ldots, n{+}\vec{i_l}.t_l)) = f_A(\phi^*_{n+i_1}(t_1), \ldots, \phi^*_{n+i_l}(t_l))$$
$$\phi^*_n(\mathrm{z}[t_1, \ldots, t_l]) = \mu_n(\,\phi_l(\mathrm{z}); \,\phi^*_n(t_1), \ldots \phi^*_n(t_l)\,)$$

where $f : \langle i_1, \ldots, i_l\rangle \in \Sigma$.

When the $\mathbb{N}$-indexed set of metavariables $Z = 0$ (empty set), $M_\Sigma 0$ is the presheaf of all structural terms (written as $T_\Sigma \mathrm{V}$ in [Ham05]). Moreover, $M_\Sigma 0$ forms the initial Σ-monoid [FPT99, Ham04]. An assignment $\theta : Z \longrightarrow M_\Sigma 0$ gives a structural valuation, and $\theta^* : M_\Sigma Z \longrightarrow M_\Sigma 0$ gives its "homomorphic" extension on meta-terms. We also call a *valuation* an assignment $\theta : Z \longrightarrow M_\Sigma 0$.

3.3 Algebraic Semantics of Rewriting

Henceforth, in this paper we consider structural CRSs only. So we say "a CRS" for a structural CRS.

The notions of models and quasi-models for CRSs are defined as follows. 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 $A = (A, [\nu, [f_A]_{f\in\Sigma}])$, where $\nu : \mathrm{V} \longrightarrow A$, equipped with preorders $\{\geq_{A(n)}\}_{n\in\mathbb{N}}$, such that every operation f_A is weakly monotone. Let A be a $\mathrm{V} + \Sigma$-algebra. A *term-generated assignment* $\phi : Z \longrightarrow A$ is a morphism of $\mathbf{Set}^{\mathbb{F}}$ that is expressed as the composite $Z \xrightarrow{\theta} M_\Sigma 0 \xrightarrow{!_A} A$ for some valuation θ, where $!_A$ is the unique $\mathrm{V} + \Sigma$-algebra homomorphism from the initial $\mathrm{V} + \Sigma$-algebra $M_\Sigma 0$. 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. [Ham05]). 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 $\mathrm{V}+\Sigma$-algebra A that satisfies all rules in the weakening closure $\mathcal{R}^\circ$. An important fact is that any Σ-monoid (M, α, ν, μ) gives a $\mathrm{V}+\Sigma$-algebra $(M, [\nu, \alpha])$. Thus in this paper, we will basically work with Σ-monoids rather than V+Σ-algebras, which gives uniform semantic treatment of algebras with substitutions.

4 Higher-Order Semantic Labelling

We are now ready to give our semantic labelling for CRSs. We give an abstract formulation along the idea of initial algebra semantics in the framework of Σ-monoids.

4.1 Semantic Labelling for Meta-terms

Henceforth, we assume that Z is an $\mathbb{N}$-indexed set of metavariables, Σ is a binding signature and M is a Σ-monoid. We introduce labelling of functions symbols: choose for every $f \in \Sigma$ a corresponding non-empty set S_f of labels, called *semantic label 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 will often write f instead of f_p.

Choose for $f : \langle i_1, \ldots, i_l \rangle \in \Sigma$, a *semantic label 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 $K_{S_f}(n) = S_f$. If it is clear from the context, the superscript of $\langle\!\langle - \rangle\!\rangle^f$ will be omitted. The semantic label map was originally called a projection, denoted by π_f in [Zan95]. 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 $\overline{\Sigma}$.

Definition 4 (Labelling map). 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

$$\phi^{\mathsf{L}}_n : M_\Sigma Z_n \longrightarrow M_{\overline{\Sigma}}Z_n$$

$$\phi^{\mathsf{L}}_n(x) = x \qquad \phi^{\mathsf{L}}_n(\mathsf{z}[\vec{t}]) = \mathsf{z}[\phi^{\mathsf{L}}_n \vec{t}]$$

$$\phi^{\mathsf{L}}_n(f(n{+}\vec{i_1}.t_1, \ldots, n{+}\vec{i_l}.t_l)) = f_{\langle\!\langle \phi^*_{n+i_1}(t_1), \ldots, \phi^*_{n+i_l}(t_l) \rangle\!\rangle^f_n}(n{+}\vec{i_1}.\phi^{\mathsf{L}}_{n+i_1} t_1, \ldots, n{+}\vec{i_l}.\phi^{\mathsf{L}}_{n+i_l} t_l)$$

We state the following characterisation that clarifies what is the mathematical structure of semantic labelled meta-terms.

Theorem 5. *For each assignment $\phi : Z \longrightarrow M$, $(M_{\overline{\Sigma}}Z, [f_\phi]_{f\in\Sigma}, \nu_\phi, \beta_\phi)$ is a Σ-monoid.*

Corollary 6. *For each assignment $\phi : Z \to M$, the labelling map $\phi^{\mathsf{L}} : M_\Sigma Z \to M_{\overline{\Sigma}}Z$ is the unique Σ-monoid morphism $(M_\Sigma Z, [f_T]_{f\in\Sigma}, \nu, \beta) \to (M_{\overline{\Sigma}}Z, [f_\phi]_{f\in\Sigma}, \nu_\phi, \beta_\phi)$.*

Proof. Let $i_\phi : Z \to M_{\overline{\Sigma}}Z$ be the assignment into the Σ-monoid $(M_{\overline{\Sigma}}Z, [f_\phi]_{f\in\Sigma}, \nu_\phi, \beta_\phi)$ defined by $\mathsf{z} \mapsto \mathsf{z}$. It is clear that $i^*_\phi = \phi^{\mathsf{L}}$ by just comparing the definitions of ϕ^{L} and the Σ-monoid extension $(-)^*$. Hence ϕ^{L} gives a Σ-monoid morphism. □

Below we describes the Σ-monoid structure on $M_{\overline{\Sigma}}Z$ mentioned above for each assignment $\phi : Z \longrightarrow M$. Let $|-|$ be the function that erases all labels in a labeled meta-term for the ordinary signature Σ.

Unit. $\nu_\phi : \mathrm{V} \to M_{\overline{\Sigma}}Z$ is defined by $x \mapsto x$.

Operations. For $f : \langle i_1, \ldots, i_l \rangle \in \Sigma$, the corresponding operation $f_\phi : \delta^{i_1} M_{\overline{\Sigma}}Z \times \cdots \times \delta^{i_l} M_{\overline{\Sigma}}Z \longrightarrow M_{\overline{\Sigma}}Z$ is defined by

$$f_\phi(n)(s_1, \ldots, s_l) = f_{\langle\!\langle \phi^*_{n+i_1}(|s_1|), \ldots, \phi^*_{n+i_l}(|s_l|) \rangle\!\rangle_n}(n + i_1.s_1, \ldots, n + i_l.s_l).$$

Multiplication. $\beta_\phi : M_{\overline{\Sigma}}Z \bullet M_{\overline{\Sigma}}Z \longrightarrow M_{\overline{\Sigma}}Z$ is defined by

$$\beta_\phi(n)(x;\, \vec{t}) = t_x$$

$$\beta_\phi(n)(\mathsf{z}[s_1, \ldots, s_l];\, \vec{t}) = \mathsf{z}[\beta_\phi(n)(s_1; \vec{t}), \ldots, \beta_\phi(n)(s_l; \vec{t})]$$

$$\beta_\phi(n)(f_q(m{+}\vec{i_1}.s_1, \ldots, m{+}\vec{i_l}.s_l);\, \vec{t})$$

$$= \begin{cases} f_p(m{+}\vec{i_1}.\beta_\phi(m{+}i_1)(s_1;\, \mathsf{up}_{m+i_1}(\vec{t}), m{+}1, \ldots, m{+}i_1), \ldots) & \text{if } m+1 > n \\ f_p(n{+}\vec{i_1}.\beta_\phi(n{+}i_1)(s_1;\, \mathsf{up}_{n+i_1}(\vec{t}), n{+}1, \ldots, n{+}i_1), \ldots) & \text{if } m+1 \le n \end{cases}$$

where $p = \langle\!\langle \phi^*(n) | \beta_\phi(n+i_1)(s_1; \mathsf{up}_{i_1}(\vec{t}), n+1, \ldots, n+i_1) |, \ldots, \phi^*(n) | \beta_\phi(n+i_l)(s_l; \mathsf{up}_{i_l}(\vec{t}), n+1, \ldots, n+i_l) | \rangle\!\rangle_n$. For the third clause, we assume that m is the length of $\vec{t}$, and I is the maximum of $i_1, \ldots, i_l$, Note that the length of "$\mathsf{up}_{i_1}(\vec{t}), n+1, \ldots, n+i_1$" is $m + i_1$, and it renames $m + k$ by $n + k$ to make bound variables sense.

Laws. To check that $M_{\overline{\Sigma}}Z$ satisfies the monoid law is straightforward induction on meta-terms. To check the Σ-monoid law $\beta_\phi \circ ([f_\phi]_{f\in\Sigma} \bullet \mathrm{id}) = [f_\phi]_{f\in\Sigma} \circ \Sigma\beta_\phi \circ \mathrm{st}$, we instantiate this at $n \in \mathbb{F}$ and chase an element, this eventually becomes the equality

$$\beta_\phi(n)(f_r(m+\vec{i_1}.s_1, \ldots, m+\vec{i_l}.s_l); \vec{t}) = f_p(m+\vec{i_1}.\beta_\phi(m+i_1)(s_1; \mathsf{up}_{i_1}(\vec{t}), m+1, \ldots, m+i_1), \ldots \\ m+\vec{i_l}.\beta_\phi(m+i_l)(s_l; \mathsf{up}_{i_l}(\vec{t}), m+1, \ldots, m+i_l)$$

where $r = \langle\!\langle \phi^*_{n+i_1}(|s_1|), \ldots, \phi^*_{n+i_l}(|s_l|) \rangle\!\rangle_n$ and p is the one given above. This obviously holds by the definition of β_ϕ.

4.2 Commutativity

In CRSs, there are two kinds of variables, i.e. "variables" and "metavariables". Accordingly, there are two kinds of substitutions:

- substitution of variables (written as β in Lemma 7), to perform (essentially) the β-reduction of an instantiated meta-application, such as an instance of F[X].
- substitution of metavariables (written as θ in Lemma 8), used to instantiate rewrite rules, and formally called valuation (Def. 3).

The labelling map ϕ^{L} has to commute with these two substitutions. This is needed is that to establish higher-order semantic labelling. We translate a usual rewrite $s \to_{\mathcal{R}} t$ to the labelled rewrite $\phi^{\mathsf{L}}_n s \to_{\overline{\mathcal{R}}} \phi^{\mathsf{L}}_n t$ (Prop. 9). This process requires to push substitutions from inside to outside of an application of the labelling map in term structures in two levels (i.e. for variables and for metavariables). Mathematically, this is commutativity of labelling with substitutions.

Lemma 7. *Let $\phi : 0 \longrightarrow M$ be an assignment. Then, the following diagram commutes in $\boldsymbol{Set}^{\mathbb{F}}$:*

$$\begin{array}{ccc} M_\Sigma 0 \bullet M_\Sigma 0 & \xrightarrow{\beta} & M_\Sigma 0 \\ \downarrow{\scriptstyle \phi^{\mathsf{L}} \bullet \phi^{\mathsf{L}}} & & \downarrow{\scriptstyle \phi^{\mathsf{L}}} \\ M_{\overline{\Sigma}} 0 \bullet M_{\overline{\Sigma}} 0 & \xrightarrow[\beta_\phi]{} & M_{\overline{\Sigma}} 0 \end{array}$$

Proof. Since ϕ^{L} is a Σ-monoid morphism, it preserves the multiplication.

Lemma 8. *Let $\phi : 0 \longrightarrow M$ and $\theta : Z \longrightarrow M_\Sigma 0$ be assignments. Then, the following diagram commutes in $\boldsymbol{Set}^{\mathbb{F}}$:*

$$\begin{array}{ccc} M_\Sigma Z & \xrightarrow{\theta^*} & M_\Sigma 0 \\ \downarrow{\scriptstyle (\phi^* \theta)^{\mathsf{L}}} & & \downarrow{\scriptstyle \phi^{\mathsf{L}}} \\ M_{\overline{\Sigma}} Z & \xrightarrow[(\phi^{\mathsf{L}}\theta)^{\overline{*}}]{} & M_{\overline{\Sigma}} 0 \end{array}$$

Here $(-)^{\overline{*}}$ denotes the $\overline{\Sigma}$-monoid morphism extension $(-)^*$ (cf. Def. 3) for the case of the labelled signature $\overline{\Sigma}$.

4.3 Labelled System

For a given CRS $(\Sigma, \mathcal{R})$ and Σ-monoid M, we define the labelled rules by

$$\overline{\mathcal{R}} = \{Z \vdash \vec{n}.\phi_n^{\mathsf{L}} l \to \vec{n}.\phi_n^{\mathsf{L}} r \mid Z \vdash \vec{n}.l \to \vec{n}.r \in \mathcal{R},\ \text{assignment } \phi : Z \longrightarrow\!\!\!\!\!\rightarrow\ M\}.$$

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. We have seen that the labelling map ϕ^{L} is a Σ-monoid morphism, i.e., preserves Σ-meta-term structures. The following proposition states that ϕ^{L} moreover preserves $\mathcal{R}$-rewrite structures.

Proposition 9. *Let M be a model of $\mathcal{R}$. If we have CRS rewriting $n \vdash s \to_{\mathcal{R}} t$ on $M_\Sigma 0_n$, then for the assignment $\phi : 0 \longrightarrow\!\!\!\!\!\rightarrow\ M$, we have rewriting $n \vdash \phi_n^{\mathsf{L}} s \to_{\overline{\mathcal{R}}} \phi_n^{\mathsf{L}} t$ on $M_{\overline{\Sigma}} 0_n$.*

Theorem 10 (Higher-order semantic labelling). *Let M be a model of $\mathcal{R}$. A CRS $\mathcal{R}$ is terminating if and only if $\overline{\mathcal{R}}$ is terminating.*

Proof. For both directions, we prove contrapositions. [$\Leftarrow$]: By Prop. 9. [$\Rightarrow$]: By erasing all labels in rewrite steps. □

4.4 Example

We illustrate how to apply the higher-order semantic labelling method. Higher-order semantic labelling itself merely transforms a CRS into a labelled one. We need separately a way to prove termination of the labelled system. For this purpose, we use Blanqui's version of the General Schema for CRSs [Bla00] to prove termination of labelled CRSs because in our experience, this is the most powerful decidable method to prove termination of CRSs. The General Schema uses a *precedence* which is a partial order on function symbols occurring in a CRS. Using a precedence, if all rewrite rules of a given CRSs follows the General Schema, we conclude termination of it.

Example 11 (CRS for prefix sum). Consider the example of CRS $\mathcal{P}$ for computing prefix sum of lists given in Example 1. The CRS $\mathcal{P}$ is formulated under the binding signature $\Sigma = \{\mathsf{map} : \langle 1, 0\rangle, \mathsf{S}, \mathsf{ps} : \langle 0\rangle,\ 0, \mathsf{nil} : \langle\rangle, +, \text{“}:\text{”} : \langle 0, 0\rangle\}$.

To use higher-order semantic labelling, we need a model of $\mathcal{P}$. Here we take the presheaf $\mathcal{M}_n \triangleq (\mathbb{N}^n \to \mathbb{N})$ of all functions on $\mathbb{N}$. This $\mathcal{M}$ forms a monoid in the monoidal category $\mathbf{Set}^{\mathbb{F}}$ by taking the multiplication $\beta : \mathcal{M} \bullet \mathcal{M} \to \mathcal{M}$ as the composition "$\circ$", and the unit $\nu : \mathrm{V} \to \mathcal{M}$ as the projections of Cartesian products $i \mapsto \pi_i$. To construct a Σ-monoid $\mathcal{M}$, we define a Σ-algebra structure on $\mathcal{M}$. First, we define the operations at the stage 0 (here we call the component parameter of a natural transformation *stage*):

$$\mathsf{map}_{\mathcal{M}_0}(f, y) = y \quad \mathsf{ps}(x) = x \quad :_{\mathcal{M}_0}(x, y) = y + 1 \quad \mathsf{nil}_{\mathcal{M}_0} = 0 \quad x +_{\mathcal{M}_0} y = 0.$$

The idea of this model is to count the number of cons's. The definition of $:_{\mathcal{M}_0}$ reflects this idea and the definition of $\mathsf{map}_{\mathcal{M}_0}$ comes from the observation that map does not change the number of cons's. For each $f : \langle i_1, \ldots, i_l \rangle \in \Sigma$, the operation at stage $n \geq 1$ is given by using pairing of functions $f_{\mathcal{M}_n}(a_1, \ldots, a_l) \triangleq f_{\mathcal{M}_0} \circ \langle a_1, \ldots, a_l \rangle$, more concretely, $f_{\mathcal{M}_n}(a_1, \ldots, a_l)(\Gamma) = f_{\mathcal{M}_0}(a_1(\Gamma), \ldots, a_l(\Gamma))$ for $\Gamma \in \mathbb{N}^n$. This indeed gives a morphism of $\mathbf{Set}^{\mathbb{F}}$. We can straightforwardly check that this gives a model of $\mathcal{P}$. We label the function symbol ps and assume that other function symbols are unlabelled. We use the natural numbers $\mathbb{N}$ as the semantic label set S_{ps}. The semantic label map is defined by $\langle\!\langle x \rangle\!\rangle_0^{\mathsf{ps}} = x$. Then, we have the following 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 $i \in \mathbb{N}$. the General Schema succeeds in showing termination of this labelled CRS with the precedence $\mathsf{ps}_i > \mathsf{ps}_j > \mathsf{map} > \mathsf{nil}, :$ for $i > j \in \mathbb{N}$.

5 Labelling with Quasi-Models

Until now the model M was a presheaf and semantic label set S_f was a set. Here we require them to be equipped with well-founded partial orders. The operations $f_{\mathcal{M}}$ and semantic label map $\langle\!\langle - \rangle\!\rangle_f$ have to be weakly monotone morphisms in $\mathbf{Set}^{\mathbb{F}}$. Moreover, here M is only required to be a quasi-model for a CRS, meaning that the interpretation of the left-hand side of a rule is greater than or equal to ($\geq$) the corresponding right-hand side.

We define this labelling with quasi-models formally. For $f : \langle i_1, \ldots, i_l \rangle \in \Sigma$, we associate a well-founded poset $(S_f, \geq_S)$ of *semantic labels* and a *semantic label map* that is a weakly monotone morphism $\langle\!\langle - \rangle\!\rangle^f : \delta^{i_1} M \times \cdots \times \delta^{i_l} M \longrightarrow K_{S_f}$. The labelled signature $\overline{\Sigma}$ is defined by using the semantic label set S_f as in Sec. 4. Let $(M, \geq_M)$ be a quasi-model for a CRS $\mathcal{R}$. Using the semantic label map and the Σ-monoid M, the labelled CRS $\overline{\mathcal{R}}$ is also defined by the same as in Sec. 4.3. Moreover, we define the CRS Decr (called "decreasing rules") over $\overline{\Sigma}$ to consist of the rules

$$f_p(\vec{i_1}.\mathrm{z}_1[\vec{i_1}], \ldots, \vec{i_1}.\mathrm{z}_l[\vec{i_l}]) \;\to\; f_q(\vec{i_1}.\mathrm{z}_1[\vec{i_1}], \ldots, \vec{i_1}.\mathrm{z}_l[\vec{i_l}])$$

for all $f : \langle i_1, \ldots, i_l \rangle \in \Sigma$ and all $p >_S q \in S_f$. Here each metavariable z_k has arity i_k (for $1 \leq k \leq l$) and $>_S$ denotes the strict part of $\geq_S$.

Proposition 12. *Let $(M, \geq_M)$ be a quasi-model for $\mathcal{R}$. If we have rewriting $n \vdash s \to_{\mathcal{R}} t$ on $M_\Sigma 0_n$, then for the assignment $\phi : 0 \longrightarrow M$, $n \vdash \phi_n^{\mathsf{L}} s \to_{\mathsf{Decr}}^* ; \to_{\overline{\mathcal{R}}} \phi_n^{\mathsf{L}} t$ holds. Here ";" denotes the sequential composition of relations.*

Theorem 13. *Let M be a quasi-model for a CRS $\mathcal{R}$ and $\overline{\mathcal{R}}$ the labelled CRS with respect to M. Then $\mathcal{R}$ is terminating if and only if $\overline{\mathcal{R}} \cup \mathsf{Decr}$ is terminating.*

Proof. For both directions, we prove contrapositions. [$\Leftarrow$]: By Prop. 12. [$\Rightarrow$]: By erasing all labels in rewrite steps. □

Example 14 (CRS for quick sort). Quick sort algorithm on natural numbers can be implemented as the CRS $\mathcal{R}$ with the standard rewrite rules: if,+, filter, ">", "≤".

$$
\begin{array}{llll}
0 > \mathrm{Y} & \to \mathsf{false} & 0 \le \mathrm{Y} & \to \mathsf{true} \\
\mathrm{x} > 0 & \to \mathsf{true} & \mathsf{s}(\mathrm{x}) \le 0 & \to \mathsf{false} \\
\mathsf{s}(\mathrm{x}) > \mathsf{s}(\mathrm{Y}) & \to \mathrm{x} > \mathrm{Y} & \mathsf{s}(\mathrm{x}) \le \mathsf{s}(\mathrm{Y}) & \to \mathrm{x} \le \mathrm{Y} \\
\mathsf{if}(\mathsf{true}, \mathrm{x}, \mathrm{Y}) & \to \mathrm{x} & \mathsf{nil} +\!\!+ \mathrm{YS} & \to \mathrm{YS} \\
\mathsf{if}(\mathsf{false}, \mathrm{x}, \mathrm{Y}) & \to \mathrm{Y} & (\mathrm{x} : \mathrm{xs}) +\!\!+ \mathrm{YS} & \to \mathrm{x} : (\mathrm{xs} +\!\!+ \mathrm{YS}) \\
\mathsf{filter}(\mathrm{P}, \mathsf{nil}) & \to \mathsf{nil} & & \\
\mathsf{filter}(\mathrm{P}, \mathrm{x} : \mathrm{xs}) & \to \mathsf{if}(\mathrm{P}[\mathrm{x}], \mathrm{x} : \mathsf{filter}(\mathrm{P}, \mathrm{xs}), \mathsf{filter}(\mathrm{P}, \mathrm{xs})) & & \\
\mathsf{qsort}(\mathsf{nil}) & \to \mathsf{nil} & & \\
\mathsf{qsort}(\mathrm{x} : \mathrm{xs}) & \to \mathsf{qsort}(\mathsf{filter}(a.\ a \le \mathrm{x}, \mathrm{xs})) +\!\!+ ((\mathrm{x} : \mathsf{nil}) +\!\!+ & & \\
& \quad \mathsf{qsort}(\mathsf{filter}(a.\ a > \mathrm{x}, \mathrm{xs}))) & &
\end{array}
$$

Since the argument of qsort in the right-hand side of the last rule (filter$(\cdots)$) is structurally bigger than the argument of qsort in the left-hand side (x : xs), the General Schema is not applicable. The higher-order recursive path ordering for the corresponding rewrite system written in the format called Inductive Data Type Systems [BJO02] also fails [BR01].

Here, we use higher-order semantic labelling with a quasi-model. Let $D = \mathbb{N} \times \mathbb{N}^*$ with the order $\langle n, l\rangle \ge \langle n', l'\rangle \overset{def}{\iff} n \ge n'$. We use the carrier $\mathcal{M}_k \triangleq (D^k \to D)$. The operations at stage 0 are:

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

$$
>_{\mathcal{M}_0}(m,n) = \begin{cases} \langle 1, \epsilon\rangle & \text{if } m > n \\ \langle 0, \epsilon\rangle & \text{otherwise} \end{cases} \qquad \le_{\mathcal{M}_0}(m,n) = \begin{cases} \langle 1, \epsilon\rangle & \text{if } m \le n \\ \langle 0, \epsilon\rangle & \text{otherwise} \end{cases}
$$

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

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

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

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

The operations at stage $n > 0$ are defined similarly to Example 11. This is indeed a quasi-model and cannot be a model. We label the function symbol qsort only. The semantic label set S_{qsort} is $\mathbb{N}$ with the usual order. The semantic label map is $\langle\!\langle \langle n, l\rangle \rangle\!\rangle_0^{\mathsf{qsort}} = n$, which is weakly monotone. Then, we have the labelled rules:

$$
\begin{array}{rl}
\mathsf{qsort}_0(\mathsf{nil}) & \to \mathsf{nil} \\
\mathsf{qsort}_{i+1}(\mathrm{x} : \mathrm{xs}) & \to \mathsf{qsort}_j(\mathsf{filter}(a.a \le \mathrm{x}, \mathrm{xs})) +\!\!+ ((\mathrm{x} : \mathsf{nil}) +\!\!+ \\
& \quad \mathsf{qsort}_k(\mathsf{filter}(a.a > \mathrm{x}, \mathrm{xs}))) \quad \text{where } i+1 > j, k \\
\mathsf{qsort}_i(\mathrm{xs}) & \to \mathsf{qsort}_j(\mathrm{xs}) \qquad \text{for all } i > j \in \mathbb{N}
\end{array}
$$

the General Schema shows termination of the labelled CRS with the precedence $\mathsf{qsort}_i > \mathsf{qsort}_j >$ filter $>$ if, +, ">", "≤" $>$ nil, :, 0, S, true, false for $i > j \in \mathbb{N}$.

Example 15 (Haskell's rewrite rule for map/map). Consider the CRS's rewrite rule in Example 2. We take the same carrier $\mathcal{M}^n = (\mathbb{N}^n \to \mathbb{N})$ as in Example 14. Now the operation on $\mathcal{M}$ is $\mathsf{map}_{\mathcal{M}_0} : (\mathbb{N} \to \mathbb{N}) \times \mathbb{N} \longrightarrow\!\!\!\!\!\rightarrow \mathbb{N}$, $\mathsf{map}_{\mathcal{M}_0}(f, x) = x + 1$. The $\mathcal{M}$ is indeed a quasi-model. We use the semantic label set and semantic label map for map are $S_{\mathsf{map}} = \mathbb{N}$ with the usual order, and $\langle\!\langle f, x \rangle\!\rangle_0^{\mathsf{map}} = x$, which is weakly monotone. This gives the labelled rules given in Introduction, hence the original rule terminates.

6 Conclusion

We have given a method of proving termination of higher-order rewrite rules in Klop's format called combinatory reduction system (CRS). The method to prove termination, called higher-order semantic labelling, is an extension of a method known in the theory of term rewriting. This attaches semantics of the arguments to each function symbol. We systematically define the labelling by using the complete algebraic semantics of CRS, Σ-monoids. A key to establish the main theorem of semantic labelling was commutativity of labelling with two kinds of substitutions appearing in formulation of CRS. We have examined the power of higher-order semantic labelling by several examples taken from functional programming. This shows usefulness of higher-order semantic labelling in programming languages.

Acknowledgments. I am grateful to the anonymous referees for useful comments on improving the presentation of the paper. This work is supported by the JSPS Grant-in-Aid for Scientific Research (19700006).

References

[Acz78] P. Aczel. A general Church-Rosser theorem. Technical report, University of Manchester, 1978.

[BJO02] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive data type systems. *Theoretical Computer Science*, 272:41–68, 2002.

[BJR08] F. Blanqui, J.-P. Jouannaud, and A. Rubio. The computability path ordering: The end of a quest. In *Proc. of CSL'08*, pages 1–14, 2008.

[Bla00] Frederic Blanqui. Termination and confluence of higher-order rewrite systems. In *Rewriting Techniques and Application (RTA 2000)*, LNCS 1833, pages 47–61. Springer, 2000.

[BN98] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[BR01] 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.

[dB72] N. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae*, 34:381–391, 1972.

[DR98] 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.

[FPT99] 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.

[Ham04] 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.

[Ham05] 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.

[Ham07] M. Hamana. Higher-order semantic labelling for inductive datatype systems. In *Ninth ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP'07)*, pages 97–108, 2007.

[JR06] J.-P. Jouannaud and A. Rubio. Higher-order orderings for normal rewriting. In *Proc. of RTA'06*, LNCS 4098, pages 387–399, 2006.

[JR07] J.-P. Jouannaud and A. Rubio. Polymorphic higher-order recursive path orderings. *Journal of ACM*, 54(1), 2007.

[JTH01] S. P. Jones, A. Tolmach, and T. Hoare. Playing by the rules: rewriting as a practical optimisation technique in GHC. In *Haskell Workshop 2001*, 2001.

[Klo80] J.W. Klop. *Combinatory Reduction Systems*. PhD thesis, CWI, Amsterdam, 1980. volume 127 of Mathematical Centre Tracts.

[KOR93] 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.

[Mac71] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1971.

[Nip91] T. Nipkow. Higher-order critical pairs. In *Proc. 6th IEEE Symp. Logic in Computer Science*, pages 342–349, 1991.

[Pol94] J. van de Pol. Termination proofs for higher-order rewrite systems. In *the First International Workshop on Higher-Order Algebra, Logic and Term Rewriting (HOA'93)*, LNCS 816, pages 305–325, 1994.

[Raa01] F. van Raamsdonk. On termination of higher-order rewriting. In *Rewriting Techniques and Applications, 12th International Conference (RTA 2001)*, pages 261–275, 2001.

[Ter03] Terese. *Term Rewriting Systems*. Number 55 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.

[Zan95] H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24(1/2):89–105, 1995.

A Appendix

A.1 Proof of Lemma 8

By induction on meta-terms in $M_\Sigma Z$. The cases x and $f(\vec{s}) \in M_\Sigma Z_n$ are straightforward. For the case $\mathrm{z}[\vec{t}] \in M_\Sigma Z_n$, we have the following.

$$\begin{aligned}
\mathrm{lhs} &= \phi^{\mathsf{L}}\theta^*(\mathrm{z}[\vec{t}]) \\
&= \phi^{\mathsf{L}}\beta(\theta\mathrm{z}; \theta^*\vec{t}) = \beta_\phi(\phi^{\mathsf{L}}\theta\mathrm{z}; \phi^{\mathsf{L}}\theta^*\vec{t}) \qquad \text{(by Lemma 7)} \\
\mathrm{rhs} &= (\phi^{\mathsf{L}}\theta)^{\overline{*}}(\phi^*\theta)^{\mathsf{L}}\mathrm{z}[\vec{t}] \\
&= (\phi^{\mathsf{L}}\theta)^{\overline{*}}\mathrm{z}[(\phi^*\theta)^{\mathsf{L}}\vec{t}] \\
&= \beta_\phi(\phi^{\mathsf{L}}\theta\mathrm{z}; (\phi^{\mathsf{L}}\theta)^{\overline{*}}(\phi^*\theta)^{\mathsf{L}}\vec{t}) \\
&= \beta_\phi(\phi^{\mathsf{L}}\theta\mathrm{z}; \phi^{\mathsf{L}}\theta^*\vec{t}) = \mathrm{lhs} \qquad \text{(by I.H.)}
\end{aligned}$$

A.2 Proof of Proposition 9

By induction on proof trees of $\to_{\mathcal{R}}$. Since $\mathcal{R}$ is structural, it suffices to consider the following two cases [Ham05].

(i) Case $n \vdash \theta_n^* l \to_{\mathcal{R}} \theta_n^* r$.
This is derived from $Z \vdash \vec{n}.l \to \vec{n}.r \in \mathcal{R}$ where $\theta : Z \longrightarrow M_\Sigma 0$. Let $\phi : 0 \longrightarrow M$ be the assignment. Now we have a labeled rule

$$(\phi^*\theta)_n^{\mathsf{L}} l \to (\phi^*\theta)_n^{\mathsf{L}} r \in \overline{\mathcal{R}}.$$

By Lemma 8 and closedness of $\overline{\mathcal{R}}$-rewrite by the valuation $\phi^{\mathsf{L}}\theta : Z \longrightarrow M_{\overline{\Sigma}} 0$, we have

$$\phi_n^{\mathsf{L}}(\theta_n^* l) = (\phi^{\mathsf{L}}\theta)_n^{\overline{*}}(\phi^*\theta)_n^{\mathsf{L}} l \ \to_{\overline{\mathcal{R}}}\ (\phi^{\mathsf{L}}\theta)_n^{\overline{*}}(\phi^*\theta)_n^{\mathsf{L}} r = \phi_n^{\mathsf{L}}(\theta_n^* r)$$

(ii) Case $n \vdash f(\ldots, n + \vec{i}.s, \ldots) \to_{\mathcal{R}} f(\ldots, n + \vec{i}.t, \ldots)$.
This is derived from $n + i \vdash s \to_{\mathcal{R}} t$. Since M is a model, notice $\phi^*_{n+i} s = \phi^*_{n+i} t$. By induction hypothesis, we have $\phi^{\mathsf{L}}_{n+i} s \to_{\overline{\mathcal{R}}} \phi^{\mathsf{L}}_{n+i} t$. So,

$$\begin{aligned}
&\phi_n^{\mathsf{L}}(f(\ldots, n + \vec{i}.s, \ldots)) \\
&= f_{\langle\!\langle \ldots, \phi^*_{n+i} s, \ldots \rangle\!\rangle_n}(\ldots, n + \vec{i}.\phi^{\mathsf{L}}_{n+i} s, \ldots) \\
&= f_{\langle\!\langle \ldots, \phi^*_{n+i} t, \ldots \rangle\!\rangle_n}(\ldots, n + \vec{i}.\phi^{\mathsf{L}}_{n+i} s, \ldots) \\
&\to_{\overline{\mathcal{R}}} f_{\langle\!\langle \ldots, \phi^*_{n+i} t, \ldots \rangle\!\rangle_n}(\ldots, n + \vec{i}.\phi^{\mathsf{L}}_{n+i} t, \ldots) \\
&= \phi_n^{\mathsf{L}}(f(\ldots, n + \vec{i}.t, \ldots))
\end{aligned}$$

A.3 Proof of Proposition 12

By induction on proof trees of $\to_{\mathcal{R}}$.

(i) Case $n \vdash \theta_n^* l \to_{\mathcal{R}} \theta_n^* r$. This case is proved by the same as in the proof of Prop. 9.
(ii) Case $n \vdash f(\ldots, n + i.s, \ldots) \to_{\mathcal{R}} f(\ldots, n + i.t, \ldots)$
This is derived from $n + i \vdash s \to_{\mathcal{R}} t$. Since $(M, \geq_M)$ is a quasi-model, we have $\phi^*_{n+i} s \geq_{M(n+i)} \phi^*_{n+i} t$. By induction hypothesis, we have $\phi^{\mathsf{L}}_{n+i} s \to^*_{\mathsf{Decr}}; \to_{\overline{\mathcal{R}}} \phi^{\mathsf{L}}_{n+i} t$. Notice also that $\langle\!\langle - \rangle\!\rangle$ is weakly monotone. So,

$$\begin{aligned}
\phi_n^{\mathsf{L}}(f(\ldots, n + i.s, \ldots)) &= f_{\langle\!\langle \ldots, \phi^*_{n+i} s, \ldots \rangle\!\rangle_n}(\ldots, n + \vec{i}.\phi^{\mathsf{L}}_{n+i} s, \ldots) \\
&\to^*_{\mathsf{Decr}} f_{\langle\!\langle \ldots, \phi^*_{n+i} t, \ldots \rangle\!\rangle_n}(\ldots, n + \vec{i}.\phi^{\mathsf{L}}_{n+i} s, \ldots) \\
&\to^*_{\mathsf{Decr}}; \to_{\overline{\mathcal{R}}} f_{\langle\!\langle \ldots, \phi^*_{n+i} t, \ldots \rangle\!\rangle_n}(\ldots, n + \vec{i}.\phi^{\mathsf{L}}_{n+i} t, \ldots) \\
&= \phi_n^{\mathsf{L}}(f(\ldots, n + \vec{i}.t, \ldots))
\end{aligned}$$

A.4 Structural CRSs as Typed CRSs

In [Bla00], Blanqui defined a version of higher-order rewriting format Inductive Data Type Systems (IDTS), which he called "new definition of IDTS" ([Bla00] Def. 1). We call his "new definition of IDTS" *typed CRS* since as mentioned in his paper, it is a simply-typed version of CRS. Blow we show that our structural CRSs is a subclass of Blanqui's typed CRSs. Hence we can apply General Schema for typed CRSs given in [Bla00] to structural CRSs to show termination of structural CRSs.

To give a typed CRSs, the following alphabet $\mathcal{A}$ ([Bla00] Def. 1) is required. In typed CRSs, types are simple types generated by the base types. (i) a set of base types, (ii) type-indexed collection of variables, (iii) type-indexed collection of function symbols, (iv) type-indexed collection of metavariables. Then the set of all meta-terms of a typed CRS is constructed from $\mathcal{A}$, and a typed CRS is a set of pairs of meta-terms.

Suppose that a structural CRS $(\Sigma, \mathcal{R})$ using a $\mathbb{N}$-indexed set Z of metavariables is given. We show that this gives rise to the following alphabet $\mathcal{A}$ and typed CRS. We assume the only base type ι and all variables (now, natural numbers) have the base type. For each function symbol $f : \langle i_1, \ldots, i_l \rangle \in \Sigma$, we assign to the type $f : \iota^{i_1}, \ldots, \iota^{i_l} \to \iota$ where $\iota^i = (\iota \to \cdots \to \iota) \to \iota$ (the part $(\iota \to \cdots \to \iota)$ denotes i-times ι). For each metavariable z of arity n in Z, we associate a metavariable z in $\mathcal{A}$ of the type $\iota^n \to \iota$. Then, the set of all structural meta-terms $\bigcup_{k\in\mathbb{N}} M_\Sigma Z(k)$ is equal to the set of all meta-terms of typed CRS given in [Bla00] under this alphabet $\mathcal{A}$. Thus, the structural CRS $\mathcal{R}$ is a typed CRS. Valuations and generation of a rewrite relation for structural CRSs also fit into those of typed CRS version.