1

An Introduction to Categorical Rewriting

Christoph Lüth
(FB 3 — Mathematik und Informatik, Universität Bremen
cxl@informatik.uni-bremen.de)

Neil Ghani
(Department of Mathematics and Computer Science, University of Leicester
ng13@mcs.le.ac.uk)

Abstract: Term rewriting systems (TRSs) are widely used throughout computer science as they form an abstract model of computation. However, the concrete syntactic nature of term rewriting often obscures the underlying structure, and the usual semantics based upon relations lacks the structure to adequately model key concepts arising in more complex problems.
A more abstract semantics at an intermediate level of abstraction between the syntax and the relational model is therefore needed. Generalising the categorical treatment of universal algebra, we model TRSs by monads. This way, we have for the first time been able to give categorical proofs of state of the art results in rewriting. This paper forms an introduction to our research, highlighting the new insights afforded by the categorical perspective and sketching the application of this semantics to modular rewriting.

1 Introduction

Term rewriting systems combine the flexibility and generality of universal algebra with the computational notion of reduction. They originally arose in the study of the λ-calculus and combinatory logic. More recently, they have played a prominent rôle in the design and implementation of logical and functional programming languages, the theory of abstract data types, and automated theorem proving. They have also found applications in recursion theory and algebra, as they are a Turing-complete model of computation which can be used to provide decision procedures.

The concreteness of TRSs has lead to a tendency to concentrate on the syntactical details of specific problems to the detriment of a wider understanding of the subject. Abstract models of TRSs have been developed to address this problem with the simplest being *Abstract Reduction Systems* (ARSs) based upon

[1] Technical Report No. 2000/35, Department of Mathematics & Computer Science, University of Leicester, University Road, Leicester LE1 7RH, England.

relations. While useful for certain results, ARSs lack sufficient structure to adequately model key concepts such as substitution, context, layer structure etc, and hence ARSs are mainly used as an organisational tool with the difficult results proved directly at the syntactic level.

Category theory has been used to provide a semantics for term rewriting systems at an intermediate level of abstraction between the actual syntax and the relational model. Research originally focused on structures such as *2-categories* [Rydeheard and Stell, 1987, Seely, 1987], *Sesqui-categories* [Stell, 1994] and *ordered categories* [Jay, 1990, Ghani, 1995a] as models of TRSs. However, despite some one-off results [Ghani, 1995b, Rydeheard and Stell, 1987], these approaches have yet to make a lasting impact on term rewriting. Part of the problem seems to be that these semantics are for the most part term-generated, and we often end up translating the problem back into the syntax!

Our alternative approach starts from the observation that term rewriting systems can be regarded as generalisations of algebraic theories. Syntactically, algebraic theories declare term constructors, while TRSs declare term constructors and rewrite rules, thus both are presented via signatures. Semantically, algebraic theories are modelled by *sets* of terms while TRSs are modelled by *pre-orders* of terms. Categorically, algebraic theories are modelled as *monads* on the category of sets and so it is natural to model a term rewriting system by a monad over a more structured base category.

This work combines techniques from two rather distant areas, and few readers will be experts in both of them. Hence, we here give an introductory exposition of our research, suitable to a general audience of theoretical computer scientists. We assume no familiarity with term rewriting, and merely a passing knowledge of the basics of category theory, concepts such as categories, functors, natural transformations, limits and colimits, and adjunctions, which can be found e.g. in Chapter I–V of [Mac Lane, 1971].

The main contribution of this paper is an overview of the monadic semantics for TRSs, with an emphasis on examples and motivation, giving basic definitions and results while eliding the technical proofs which can be found in the references. Instead, we shall concentrate on where the categorical perspective gives new insights into rewriting and demonstrate how our ideas can be applied to modularity problems.

The structure of this paper is as follows: in the next section, we will present the modelling of algebraic theories by monads. [Section 3] will introduce the basics of TRSs and ARSs. In [Section 4], we will present the monadic semantics, and demonstrate how to use this semantics in modularity problems in [Section 5]. [Section 6] closes with a summary and an outlook.

2 Algebraic Theories and Monads

Since our approach generalises the categorical treatment of universal algebra, where algebraic theories are equivalent to monads on the category of sets, we will give a self-contained presentation of this equivalence here. However, since this standard material, we omit proofs and instead refer the reader to [Manes, 1976] or Chapter VI of [Mac Lane, 1971].

Definition 1 (Signature). A (single-sorted) *signature* consists of a function $\Sigma : \mathit{Nat} \to \mathbf{Set}$. The set of *$n$-ary operators* of Σ is defined $\Sigma_n \stackrel{def}{=} \Sigma(n)$.

Definition 2 (Term Algebra). Given a signature Σ and a set of variables X, the *term algebra* $T_\Sigma(X)$ is defined inductively:

$$\frac{x \in X}{\mathtt{'}x \in T_\Sigma(X)} \qquad \frac{f \in \Sigma_n \quad t_1, \dots t_n \in T_\Sigma(X)}{f(t_1, \dots, t_n) \in T_\Sigma(X)}$$

Quotes are used to distinguish a variable $x \in X$ from the term $\mathtt{'}x \in T_\Sigma(X)$. This will become important when analysing the layer structure on terms formed by the disjoint union of signatures.

Definition 3 (Equations and Algebraic Theories). Given a signature Σ,a Σ*-equation* is of the form $X \vdash t = s$ where X is a set and $t, s \in T_\Sigma(X)$. An *algebraic theory* $\langle \Sigma, E \rangle$ consists of a signature Σ and a set E of Σ-equations.

The term algebra construction generalises from signatures to algebraic theories by mapping a set X to the term algebra quotiented by the equivalence relation $\sim$ generated from the equations

$$T_{\mathcal{A}}(X) = T_\Sigma(X)/\!\sim$$

The key idea of the categorical semantics is that the free term algebra construction, which for every set X gives us the set of terms of term $T_{\mathcal{A}}(X)$, extends to a *functor* $T_{\mathcal{A}} : \mathbf{Set} \to \mathbf{Set}$ over the category $\mathbf{Set}$ of sets. Further, every variable $x \in X$ gives a term $\mathtt{'}x \in T_{\mathcal{A}}(X)$, so the variables are given by a function $\eta_X : X \to T_{\mathcal{A}}(X)$. Lastly, substitution takes terms built over terms and flattens them, as described by a function $\mu_X : T_{\mathcal{A}}(T_{\mathcal{A}}(X)) \to T_{\mathcal{A}}(X)$. These three pieces of data define a monad:

Definition 4 (Monad). A *monad* $\mathsf{T} = \langle T, \eta, \mu \rangle$ on a category $\mathcal{C}$ is given by an endofunctor $T : \mathcal{C} \to \mathcal{C}$, called the *action*, and two natural transformations, $\eta : 1 \Rightarrow T$, called the *unit*, and $\mu : TT \Rightarrow T$, called the *multiplication* of the monad, satisfying the *monad laws*: $\mu{\cdot}T\eta = 1 = \mu{\cdot}\eta_T$, and $\mu{\cdot}T\mu = \mu{\cdot}\mu_T$.

Lemma 5. *For every algebraic theory $\mathcal{A}$, $\mathsf{T}_{\mathcal{A}} = \langle T_{\mathcal{A}}, \eta, \mu \rangle$ is a monad.*

The data from Def. 4 are actually enough to characterise term construction. Further, the category of algebras of this monad is equivalent to the category of models of $\mathcal{A}$, justifying the claim that "universal algebra is the study of monads over **Set**" ([Manes, 1976], pg. 42).

In summary, monads provide an abstract calculus for algebraic theories where *variables, substitution* and *term algebra* (represented by the unit, multiplication and action of the monad) are taken as primitive concepts.

3 Term Rewriting Systems and Abstract Reduction Systems

This section reviews the basic concepts of term rewriting systems and abstract reduction systems. Further details may be found in [Baader and Nipkow, 1998, Klop, 1992].

Definition 6 (Rewrite Rules and TRSs). Given a signature Σ, a *Σ-rewrite rule* is of the form $(Y \vdash l \to r)$ where Y is a set of variables and $l, r \in T_\Sigma(Y)$. A *term rewriting system* $\mathcal{R} = \langle \Sigma, R \rangle$ consists of a signature Σ and a set R of Σ-rewrite rules.

Usually one requires that l is not a variable and that the free variables of r are contained in the free variables of l, but semantically these restrictions are not necessary and hence omitted here. In the following examples, we denote the arity of an operation by subscripts; so $f \in \Sigma(n)$ is denoted by f_n.We also use infix notation for binary operators:

Example 1 (Combinatory Logic). The TRS $CL = \langle \Sigma_{CL}, R_{CL} \rangle$ has $\Sigma_{CL} = \{S_0, K_0, \cdot_2\}$ and $R_{CL} = \{\{x, y\} \vdash (K \cdot x) \cdot y \to x, \{x, y, z\} \vdash ((S \cdot x) \cdot y) \cdot z \to (x \cdot z) \cdot (y \cdot z)\}$.

Example 2 (Addition). The TRS $Add = \langle \Sigma_{Add}, R_{Add} \rangle$ implements addition on the natural numbers and has $\Sigma_{Add} = \{0_0, S_1, +_2\}$ and $R_{Add} = \{r_1, r_2\}$ where

$$\begin{aligned} r_1 &= \{x\} \vdash 0 + x \to x \\ r_2 &= \{x, y\} \vdash Sx + y \to S(x + y) \end{aligned}$$

A term rewriting system $\mathcal{R}$ induces a *one-step reduction relation* $\to_{\mathcal{R}}$ on the terms. Formally, this is obtained by closing the rewrite rules under contexts and substitutions — for the exact definitions, see [Klop, 1992]. The *many-step* *reduction relation* $\twoheadrightarrow_{\mathcal{R}}$ is the reflexive-transitive closure of the one-step reduction relation. For example, the addition TRS supports the following rewrites

$$\begin{array}{ccccc} S(0+0)+0 & \twoheadrightarrow & S((0+0)+0) & & \\ \downarrow\!\!\!\downarrow & & \downarrow\!\!\!\downarrow & & \\ S0+0 & \twoheadrightarrow & S(0+0) & \twoheadrightarrow & S0 \end{array}$$

Notice that although the term $S(0+0)+0$ can be reduced to two different terms, this choice is illusory as both reduce to $S(0+0)$. Further, there are no infinite reduction sequences starting from the term $S(0+0)+0$. These examples illustrate the two key properties of TRSs:

Definition 7 (Confluence and SN). A term rewriting system $\mathcal{R} = \langle \Sigma, R \rangle$ is *confluent* iff $\forall t, s_1, s_2.\, t \twoheadrightarrow_R s_1 \wedge t \twoheadrightarrow_R s_2\ \exists u.\, s_1 \twoheadrightarrow_R u \wedge s_2 \twoheadrightarrow_R u$. It is *strongly normalising* (SN) iff there is no infinite sequence $t_1 \to_R t_2 \to_R t_3 \to_R \ldots$.

Example 1 is confluent, but not strongly normalising while Example 2 is confluent and strongly normalising. This may not be apparent at first sight, and indeed in general these properties are undecidable. Consequently a variety of proof tools have been developed which are helpful in certain cases. Some of these methods keep track of the *redexes*, the places in the term where a rewrite can be applied. Possibly the most famous example of redex tracking is the *minimal developments* technique used in λ-calculus [Hindley and Seldin, 1986]. Other, more ad-hoc methods include *Knuth-Bendix Completion* [Knuth and Bendix, 1970], *reducibility*, and ordering the operations or terms (*reduction orders*) to show termination. These methods are rather technical, often involving copious amounts of induction on the structure of terms, and tend to impose syntactic preconditions on term rewriting systems, such as left-linearity or orthogonality.

In order to remedy this situation, we turn to the *semantics* of term rewriting systems hoping to obtain a new ontology for rewriting. The simplest models of TRSs are the *abstract reduction systems* based upon relations; the advantages of this approach are its simplicity and versatility.

Definition 8 (Abstract Reduction Systems). An *abstract reduction system* (ARS) is a pair $\mathcal{A} = \langle A, \to_A \rangle$ consisting of a carrier set A and a binary relation $\to_A \subseteq A \times A$, called the reduction or rewrite relation.

Every term rewriting system $\mathcal{R} = \langle \Sigma, R \rangle$ defines an ARS whose carrier set is the set of terms $T_\Sigma(V)$ where V is an infinite set of variables, and whose relation is the one-step reduction relation. Properties like confluence and strong normalisation can be (and usually are) defined on the level of ARSs. We can then prove lemmas and theorems about these properties at the level of relations, such as *Newman's Lemma* (if an ARS is locally confluent and strongly normalising then it is confluent), or the *Hindley-Rosen Lemma* (if two ARS are confluent and commute, then their union is confluent).

The key point about these lemmas and their proofs is that they are formulated and proven at the level of relations. Nowhere do we need the fact that the carrier set consists of terms and that, for example, results may be proven by induction on term structure. Thus we may contrast the *relational methodology* of the ARS model to the *combinatorial* methodology inherent in direct reasoning.

While these lemmas are very useful, relations cannot express the key concepts of term rewriting like substitution, redex, context, layer and everything else relating to the term structure. Hence, ARSs are mainly used as an organisational tool, while the difficult results are proven directly at the syntactic level. Thus, the very simplicity which is the advantage of relational models is also its main limitation. What is needed is a semantics at an intermediate level of abstraction between the actual syntax and the relational model.

4 Categorical Models of Term Rewriting Systems

The key observation underlying our semantics for TRSs is that TRSs can be viewed as generalised algebraic theories. This generalisation occurs at both a syntactic and a semantic level. Syntactically, algebraic theories and TRSs are both presented via signatures — the former declares term constructors while TRSs declare term constructors and rewrite rules. Semantically, algebraic theories are modelled by sets with extra structure while TRSs are modelled by pre-orders with extra structure.

Given the satisfactory categorical treatment of algebraic theories by monads on **Set** presented in [Section 2], it is natural to model a term rewriting system by a monad over a more structured base category. In this section we formalise these comments, first with a concrete construction and then with a more abstract, and in many ways more illuminating, construction.

4.1 A Monadic Semantics for TRSs

We will now generalise the monadic semantics from algebraic theories to TRSs. Our aim is to model a TRS $\mathcal{R} = \langle \Sigma, R \rangle$ by a monad $\mathsf{T}_{\mathcal{R}}$.

First, we have to decide on the base category of the monad. Since the main focus of rewriting is to prove properties of the many-step reduction relation, we choose to use the category **Pre** of pre-orders and monotone functions. Other possibilities are the category **Cat** of all small categories and functors, if we were interested in labelled many-step reduction or the category **Rel** of relations and monotone maps if one wanted to model the one-step reduction relation.

Next we consider the action of the monad $\mathsf{T}_{\mathcal{R}}$. Recall that if $\mathcal{A}$ is an algebraic theory, then $T_{\mathcal{A}}$ maps a set of variables to the free algebra over it. In general, the action is an endofunctor, and variables and theory must belong to the same category which in our case is **Pre**. Thus, the action $T_{\mathcal{R}}$ should map a *pre-order* of variables P to the pre-order of terms and rewrites built over P. The order structure on the variables can be thought of as reduction assumptions: if $a \to b$, then whatever we substitute for a has to reduce to what we substitute for b.

Thus, we have two main differences from the traditional semantics: a TRSs is modelled by a particular kind of *function* rather than a relation; and we endow variables with an order structure. A third categorically inspired idea is that there is no fixed collection of variables, but that we have a term reduction algebra defined for *every* pre-order of variables.

Table 1: The Term Reduction Algebra for a TRS $\mathcal{R} = \langle \Sigma, R \rangle$

Given a term rewriting system $\mathcal{R} = \langle \Sigma, R \rangle$ and a pre-order X, the *term reduction algebra* $T_{\mathcal{R}}(X)$ is the smallest pre-order $\to$ on the terms $T_{\Sigma}(X)$ satisfying the following rules:

$$[\textsc{Var}]\ \frac{x \to y \text{ in } X}{{}'x \to {}'y} \qquad [\textsc{Cong}]\ \frac{f \in \Sigma_n, t_1 \to s_1, \ldots, t_n \to s_n}{f(t_1, \ldots, t_n) \to f(s_1, \ldots, s_n)}$$

$$[\textsc{Inst}]\ \frac{(Y \vdash l \to r) \in R,\ \sigma : Y \to T_{\mathcal{R}}(X)}{l\sigma \to r\sigma}$$

These considerations are summed in the definition of the *term reduction algebra* in [Tab. 4.1]. As with algebraic theories, this gives rise to a monad:

Lemma 9. *Let $\mathcal{R}$ be a TRS. The map $X \mapsto T_{\mathcal{R}}(X)$ extends to a monad $\mathsf{T}_{\mathcal{R}} = \langle T_{\mathcal{R}}, \eta, \mu \rangle$ on* **Pre**.

Proof. The proof is the obvious generalisation of Lemma 5. Essentially, we have to show that the action, unit and substitution preserve the reduction order. Note how the clause [Var] guarantees that the unit is a monotone function.

Lemma 9 provides us with a categorical semantics for TRSs. This semantics is powerful enough to prove non-trivial modularity results, but in a number of ways, it is still unsatisfactory and can be improved on. Firstly, the semantics is still term-generated; proofs about it will still use induction on the term structure; secondly, the semantics has been constructed based on a number of choices, like that of the base category, and if we change any of these, we have to start the construction from scratch again; and thirdly, the construction does not generalise easily to other forms of rewriting. These deficiencies are answered by the theory of enriched monads.

4.2 TRSs and Enriched Monads

As we have seen, algebraic theories are modelled by monads over **Set** while TRSs are modelled by monads over **Pre**. These results are instances of a general theory developed by Kelly and Power [Kelly, 1982, Kelly and Power, 1993] which shows how monads over categories other than **Set** give rise to a generalised notion of algebraic theories. For any such category $\mathcal{C}$, this theory provides us with suitably generalised notions of *signature*, *term*, and *equations* (Def. 10, 11 and 12 below). Technically, we *enrich* the monads [Kelly, 1982], and this imposes certain conditions on $\mathcal{C}$, but since this paper is tutorial in nature, we omit these here; we also make several simplifications to keep technicalities at a minimum. [Robinson, 1994] gives a gentle introduction but more exact introduction into this theory. [Section 5] dealing with the application of our semantics to modular rewriting is independent of this material and hence may be read first.

Recall from definition 1 that a signature is a function $\Sigma : Nat \rightarrow \mathbf{Set}$ which maps each arity n to the set of operations $\Sigma(n)$ of that arity. To define signatures for a category $\mathcal{C}$ we replace a *set of operations* by a *$\mathcal{C}$-object of operations*, i.e. change the codomain of Σ to $\mathcal{C}$. This will be the general theme: we systematically replace everything specific to **Set** by $\mathcal{C}$. As for the domain of Σ, the key point is that the natural numbers represent the isomorphism classes of finite sets (i.e. objects of the base category). Hence, we replace Nat by the *finitely presentable* objects of $\mathcal{C}$:

Definition 10 (Generalised Signatures). If $\mathcal{C}$ is a category, then a $\mathcal{C}$-signature is a map $\Sigma : |\mathcal{C}_{\mathrm{fp}}| \rightarrow \mathcal{C}$.

A pre-order is finitely presentable iff its carrier set is finite. Hence if we present a TRS as a generalised signature Σ, the signature will map a finite pre-order P to a pre-order $\Sigma(P)$ of operations of that arity. The carrier set of $\Sigma(P)$ can be thought of as term constructors and the order relation as rewrite rules. This is another instance where the categorical approach differs from the traditional one where rewrite rules are more often viewed as oriented equations rather than constructors in their own right.

We write $\mathbf{1}, \mathbf{2}, \ldots$ for the finite discrete pre-orders, the elements of which we denote as $x_1, x_2, \ldots$.

Example 3 (Addition cont'd). In the generalised signature $\mathcal{R}$ for the addition TRS, the term constructors retain their arities, while the two rewrite constructors have arity $\mathbf{1}$ and $\mathbf{2}$, respectively. Hence, we have

$$\begin{array}{lcl} \mathcal{R}(\emptyset) & = & (0) \\ \mathcal{R}(\mathbf{1}) & = & (S \quad s_1 \xrightarrow{r_1} t_2) \\ \mathcal{R}(\mathbf{2}) & = & (+ \quad s_2 \xrightarrow{r_2} t_2) \\ \mathcal{R}(X) & = & () \quad \text{otherwise} \end{array}$$

Since the arity of the rewrite rule $0 + x_1 \to x_1$ is the discrete 1-element pre-order, we have represented it by an arrow in $\mathcal{R}(\mathbf{1})$. But of course arrows need sources and targets, and hence the carrier set of $\mathcal{R}(\mathbf{1})$ has three elements and thus declares three term constructors. To model a TRS as a generalised signature we extend the signature of the TRS with a term constructor for the source and target of every rewrite rule r_i (which above are denoted by s_i and t_i). We shall later use equations to enforce that e.g. the target of rule r_2 equals $S(x_1 + x_2)$.

Importantly, arities can be non-discrete. Syntactically, this means we allow *generalised rewrite rules*, given as $(Y \vdash l \to r)$ where Y is a *pre-order* and $l, r \in T_{\mathcal{A}}(Y)$.

The next step is to generalise the term algebra construction from Def. 2. The term algebra over a set X can be seen as the limit of the sets $T_n(X)$ of terms of depth at most n. The terms of depth 0 are just the variables while terms of depth $n+1$ are formed by applying operations of arity m to m-tuples of terms of depth n. Thus, the terms of depth $n+1$ are pairs of $f \in \Sigma(m)$ and maps $m \to T_n(X)$. Equivalently, the terms of depth $n+1$ are given by $\mathbf{Set}(m, T_n(X)) \times \Sigma(m)$, where $\mathbf{Set}(m, T_n(X))$ is the set of maps from m to $T_n(X)$. Abstracting away from the category $\mathbf{Set}$, we obtain the following:

Definition 11 (Generalised Term Algebra). For a $\mathcal{C}$-signature $\Sigma : |\mathcal{C}_{\mathrm{fp}}| \to \mathcal{C}$, the term algebra $T_\Sigma(X)$ over an object X of $\mathcal{C}$ is where

$$\begin{aligned} T_\Sigma(X) &\stackrel{\text{def}}{=} \mathop{colim}_{n<\omega} T_n(X) \\ T_0(X) &\stackrel{\text{def}}{=} X \\ T_{n+1}(X) &\stackrel{\text{def}}{=} X + \sum_{c \in \mathcal{C}_{\mathrm{fp}}} \mathcal{C}(c, T_n(X)) \times \Sigma(c) \end{aligned}$$

Example 4 (Addition cont'd). Let P be the pre-order $(a \to b)$. Leaving away those arities c for which $\Sigma(c)$ is empty, we get

$$T_1(P) = P + \mathbf{Pre}(\emptyset, P) \times \Sigma(\emptyset) + \mathbf{Pre}(1, P) \times \Sigma(1) + \mathbf{Pre}(2, P) \times \Sigma(2)$$

Now $\mathbf{Pre}(\emptyset, P) = 1$, $\mathbf{Pre}(1, P) = P$ and $\mathbf{Pre}(2, P) = P \times P$. Hence $T_1(P)$ is the preorder with the following elements and order structure:

$$\begin{tikzcd} a \arrow[d] \\ b \end{tikzcd} \qquad 0 \qquad \begin{tikzcd} S(a) \arrow[d] \\ S(b) \end{tikzcd} \qquad \begin{tikzcd} s_1(a) \arrow[r] \arrow[d] & t_1(a) \arrow[d] \\ s_1(b) \arrow[r] & t_1(b) \end{tikzcd} \qquad \begin{tikzcd} a+a \arrow[r] \arrow[d] & a+b \arrow[d] \\ b+a \arrow[r] & b+b \end{tikzcd}$$

$$\begin{tikzcd} s_1(a,a) \arrow[rr] \arrow[dr] \arrow[dd] & & s_1(a,b) \arrow[dr] \arrow[dd] & \\ & t_1(a,a) \arrow[rr] \arrow[dd] & & t_1(a,b) \arrow[dd] \\ s_1(b,a) \arrow[rr] \arrow[dr] & & s_1(b,b) \arrow[dr] & \\ & t_1(b,a) \arrow[rr] & & t_1(b,b) \end{tikzcd}$$

Finally, we need to replace the term constructors representing the source and target of rewrites with the intended terms. Intuitively we would like to write

$$s_1(x) = 0 + x \quad t_1(x) = x \quad s_2(x,y) = Sx + y \quad t_2(x,y) = S(x+y)$$

Thus, we have to move from generalised signatures to generalised algebraic theories. As with signatures, equations have arities and the collection of all equations of a given arity are required to form an object of $\mathcal{C}$.

Definition 12 (Generalised Algebraic Theory). A *generalised algebraic theory* $\mathcal{A} = \langle \Sigma, E \rangle$ is given by a generalised signature Σ, and a set of equations, which are given by a function $E : |\mathcal{C}_{\mathrm{fp}}| \to \mathcal{C}$ together with a family of pairs of maps

$$\left\{ E(c) \underset{\tau_c}{\overset{\sigma_c}{\rightrightarrows}} T_\Sigma(c) \right\}_{c \in |\mathcal{C}_{\mathrm{fp}}|}$$

The idea is that E specifies the shape of the equations while the maps σ and τ can be thought of as exhibiting a subobject of $T_\Sigma(c) \times T_\Sigma(c)$, i.e. pairs of terms, just like usual equations.

In the addition example there are two equations between terms of arity $\mathbf{1}$ and two equations between terms of arity $\mathbf{2}$, hence $E(\mathbf{1}) = E(\mathbf{2}) = (\circ\ \circ)$. The maps σ_1 and τ_1 now pick out a pair of elements of $T_\Sigma(1)$, and similarly σ_2 and τ_2. Therefore we define:

$$\begin{aligned}
\sigma_1(\circ\ \circ) &= (s_1(x_1) \quad\quad t_1(x_1))\\
\tau_1(\circ\ \circ) &= (0 + x_1 \quad\quad x_1)\\
\sigma_2(\circ\ \circ) &= (s_2(x_1,x_2) \quad t_2(x_1,x_2))\\
\tau_2(\circ\ \circ) &= (Sx_1 + x_2 \quad S(x_1 + x_2))
\end{aligned}$$

Just as for usual algebraic theories, the semantics of such a generalised algebraic theory is constructed as the quotient of the term algebra by the equations. Categorically, quotients are defined by *coequalisers* and hence the monad $T_{\mathcal{A}}$ is *presented* as the *coequaliser* of the two maps σ and τ or, in other words the smallest monad which makes these maps equal.

We finish this section with a crucial observation about our semantics, namely that it is constructed *categorically*, and hence we can reason about it categorically. This contrasts, for example, with the 2-categorical semantics of a TRS where the theory of a TRS is constructed syntactically. It is also more amenable to generalisations, e.g. the equations do not have to be between a source or target and a term, we can have equations between any two terms, allowing for *equational rewriting*.

5 Monads and Modularity

Modularity seeks criteria under which structured TRSs inherit properties from smaller (and hence easier to reason about) subsystems. Formally, a property P

is *modular* for a structuring operation $\oplus$ iff

$$\mathcal{R}_1 \oplus \mathcal{R}_2 \models P \text{ iff } (\mathcal{R}_1 \models P \text{ and } \mathcal{R}_2 \models P)$$

There is a host of results in the literature about modularity. For example, confluence is modular with respect to the disjoint union, but strong normalisation is only modular under strong syntactic conditions. Neither property is modular for the non-disjoint union of TRSs (see [Baader and Nipkow, 1998] for a survey of results and further references).

To attack the modularity of P for $\oplus$ categorically, we have to

1. Give a semantics of rewriting by representing $\mathcal{R}$ as a monad $T_{\mathcal{R}}$;
2. Find a monadic version of the property P as a property P' on monads;
3. Represent $\oplus$ by an operation on monads;
4. Show that the operation on monads preserves P'.

Along the way, we have to show that each of these steps is correct. Then we have reduced the syntactic modularity to a semantic modularity problem.

5.1 Semantics of Rewriting

This paper has sketched a semantics for TRSs as monads over the category **Pre** of pre-orders. However there are numerous generalisations of TRSs which are widely studied in the literature. The Kelly-Power framework also allows the modelling of equational rewriting and conditional rewriting. A rather more distant area, which is currently being investigated, is infinitary rewriting via a co-algebraic semantics.

To show correctness of this step, we show that for a discrete pre-order X, the term reduction algebra $T_{\mathcal{A}}(X)$ is isomorphic to the many-step reduction algebra $\twoheadrightarrow_{\mathcal{R}}$ [Lüth, 1998, Lüth and Ghani, 1997].

5.2 Monadic Version of Rewriting Properties

Properties such as confluence and strong normalisation are easily formalised for pre-orders. Then a monad T satisfies a property P if its action T preserves it, i.e. $\mathsf{T} \models P$ iff ($X \models P$ implies $T(X) \models P$). To show correctness of this definition we have to show it is equivalent to the usual formulation of these properties, i.e. $\mathsf{T}_{\mathcal{R}} \models P$ iff $\mathcal{R} \models P$. This may not be totally straightforward, and depends upon showing that the variable rewrites do not interfere with the rewrites generated by the rewrite rules. For confluence and strong normalisation, this equivalence has already been established in [Lüth, 1998, Lüth and Ghani, 1997].

5.3 Combining Monads

A semantics is called *compositional* if it is compatible with the structuring operations — the semantics of the whole can be built from the semantics of its parts. Here, the semantics for TRSs is a left adjoint, with the right adjoint given by the *internal language.* Most structuring operations (such as the disjoint and non-disjoint union) are colimits, and since left adjoints preserve colimits, our semantics is always compositional.

Our research has so far concentrated upon the simplest structuring operation, namely the disjoint union of TRSs which categorically is the coproduct. Hence we need a concrete construction of the coproduct of two monads. As motivation, consider the terms built in the disjoint union of two (ordinary) signatures Ω, Σ. Such terms have an inherent notion of layer, that is one can decompose a term constructed from symbols in the union of two disjoint signatures into a term constructed from symbols in only one signature and strictly smaller subterms whose head symbols are from the other signature. Thus terms built from operations of $\Omega + \Sigma$ are contained in

$$\begin{aligned} T_{\Omega+\Sigma}(X) = \; & X + T_\Omega(X) + T_\Sigma(X) + T_\Omega T_\Sigma(X) + T_\Sigma T_\Omega(X) + \\ & T_\Omega T_\Sigma T_\Omega(X) + T_\Sigma T_\Omega T_\Sigma(X) + \ldots \end{aligned} \tag{1}$$

However this disjoint union is too large as e.g. each component of the sum in equation (1) contains a separate copy of the variables X. Therefore, this sum is quotiented by taking the colimit of a diagram D_X; abstractly, D_X can be seen as the description of a way in which rewrites from the component systems are combined.

For this construction it is crucial that the monadic semantics allows to build terms over other terms. In this way, the concept of a monad precisely captures the notion of layers which is central to modularity proofs.

5.4 Monadic Modularity Proofs

We have now reduced modularity P for a structuring operation $\oplus$ on term rewriting systems to the equivalent problem of modularity of a property P' for a structuring operation of monads. For the coproduct of monads, we know how to express this coproduct pointwise as the colimit of a diagram D_X at every pre-order at X.

By reducing a syntactic problem to a semantic one, we can apply the methods of category theory. Since the monadic properties are defined in terms of preservation of a property of the objects of the base category, we know by induction that all the objects of the diagram D_X — all summands of (1) — satisfy P'. It remains to be shown that it is preserved by passing to the colimit. Since any colimit can be given by coproducts and coequalisers, this means that proving that e.g. the colimit of D_X preserves confluence can be reduced to proving that coproducts and coequalisers (in **Pre**) preserve confluence. For coproducts this is easy, while for coequalisers this requires some work. Here, the variable rewrites come in very useful as they allow to specify the rewrites in the lower layers, and thus abolish the distinction between inner and outer reductions [Klop et al., 1994]. This way, categorical modularity proofs for confluence and SN have been given in [Lüth, 1998, Lüth and Ghani, 1997].

To sum up this section, we have demonstrated how the monadic approach can be used for modularity problems. The semantics gives us a clear road-map how to approach the problem, by reducing the syntactic modularity problem to a semantic one.

6 Conclusion and Outlook

This paper has started with the claim that term rewriting needs a semantics at an intermediate level of abstraction between the reductions generated by the syntax, and abstract reduction systems based on relations. By following through the analogy between universal algebra and term rewriting systems, we gave a semantics for term rewriting systems, both in an explicit form (the term reduction algebra) and a more general form (generalised algebraic theory) based on enriched monads.

This semantics has also provided us with new insights, and different perspectives of rewriting. Some of these have turned out to be useful in proofs, such as the functionality of the semantics, the variable rewrites and the compositionality; some of these remain mere observations at the moment, such as considering rewrite rules as constructors rather than oriented equations, and the rôle of the variables and the arities. Finally, the semantics also gives us new syntactical insights, such as the generalised rewrite rules and the quotes which distinguish a term from a variable and considerably simplify syntactic proofs of modularity.

For future work, the flexibility of the framework allows many generalisations, some of which (conditional and equational rewriting) have already been intimated above; others include formalising graph rewriting and persistence problems. As for modularity, structuring operations other than the coproduct can be considered.

Of course, the monadic semantics has its limitations. Presently variable bindings and higher-order rewriting systems fall out of their scope, though there are indications how to cover the former [Fiore et al., 1999].

References

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

[Fiore et al., 1999] Fiore, M., Plotkin, G., and Turi, D. (1999). Abstract syntax and variable binding. In *Proc. LICS'99*. To appear.

[Ghani, 1995a] Ghani, N. (1995a). *Adjoint Rewriting.* PhD thesis, University of Edinburgh.

[Ghani, 1995b] Ghani, N. (1995b). $\beta\eta$-equality for coproducts. In *Second Conference on Typed Lambda Calculus and its Applications*, Edinburgh.

[Hindley and Seldin, 1986] Hindley, J. R. and Seldin, J. P. (1986). *Introduction to Combinators and λ-Calculus.* Number 1 in London Mathematical Society Student Texts. Cambridge University Press.

[Jay, 1990] Jay, C. B. (1990). Modelling reductions in confluent categories. In *Proceedings of the Durham Symposium on Applications of Categories in Computer Science.*

[Kelly, 1982] Kelly, G. M. (1982). *Basic Concepts of Enriched Category Theory*, volume 64 of *London Mathematical Society Lecture Note Series.* Cambridge University Press.

[Kelly and Power, 1993] Kelly, G. M. and Power, A. J. (1993). Adjunctions whose counits are coequalizers, and presentations of finitary monads. *Journal for Pure and Applied Algebra*, 89:163– 179.

[Klop, 1992] Klop, J. W. (1992). Term rewriting systems. In Abramsky, S., Gabbay, D. M., and Maibaum, T. S. E., editors, *Handbook of Logic in Computer Science*, volume 2 (Background: Computational Structures), pages 1–116. Oxford University Press.

[Klop et al., 1994] Klop, J. W., Middeldorp, A., Toyama, Y., and de Vrijer, R. (1994). A simplified proof of Toyama's theorem. *Information Processing Letters*, 49:101–109.

[Knuth and Bendix, 1970] Knuth, D. and Bendix, P. (1970). Simple word problems in universal algebra. In Leech, J., editor, *Computational Problems in Universal Algebras*, pages 263–297. Pergamon Press.

[Lüth, 1998] Lüth, C. (1998). *Categorical Term Rewriting: Monads and Modularity.* PhD thesis, University of Edinburgh.

[Lüth and Ghani, 1997] Lüth, C. and Ghani, N. (1997). Monads and modular term rewriting. In Moggi, E. and Rosolini, G., editors, *Category Theory and Computer Science*, number 1290 in Lecture Notes in Computer Science, pages 69– 86, Santa Margherita Ligure, Italy. Springer Verlag.

[Mac Lane, 1971] Mac Lane, S. (1971). *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics.* Springer Verlag.

[Manes, 1976] Manes, E. G. (1976). *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics.* Springer Verlag.

[Robinson, 1994] Robinson, E. (1994). Variations on algebra: monadicity and generalisations of equational theories. Technical Report 6/94, Sussex Computer Science Technical Report.

[Rydeheard and Stell, 1987] Rydeheard, D. E. and Stell, J. G. (1987). Foundations of equational deduction: A categorical treatment of equational proofs and unification algorithms. In *Category Theory and Computer Science*, number 283 in Lecture Notes in Computer Science, pages 114– 139. Springer Verlag.

[Seely, 1987] Seely, R. A. G. (1987). Modelling computations: A 2-categorical framework. In *Proceedings of the Second Annual Symposium on Logic in Computer Science*, pages 65–71.

[Stell, 1994] Stell, J. G. (1994). Modelling term rewriting systems by Sesqui-categories. Technical Report TR94-02, Keele Unversity.