The Unified Computation Laboratory. C M I Rattray & R G Clark (eds)

Linear Term rewriting systems are higher dimensional string rewriting systems

Michael Johnson

School of Maths and Computing
Macquarie University

Abstract

This paper illustrates a categorical approach to the theory of rewriting systems along the lines adumbrated by Buchberger [3]. It shows how 2-categories provide a simple framework for string rewriting systems and briefly reviews the theory of string rewriting to indicate its simplicity. In contrast the theory of term rewriting systems is complicated by the interaction of substitution and rewriting. We outline a 3-categorical framework for linear term rewriting systems and show by a detailed example how this separates substitution and rewriting and reduces linear term rewriting to the simplicity of string rewriting, but with one extra dimension.

1 Introduction

Rewriting systems have broad applications in Computer Science in areas as diverse as automated theorem proving [12], polynomial ideal theory [2], computational algebra [10], Petri nets, and general models of computation, especially using constructions of free categories with structure [16],[14],[4].

A rewriting rule is a "basic derivation" that allows one to pass from one "term" in an appropriate language to another, and the study of a rewriting system is the study of the compositions of such basic derivations. The mathematical theory which is usually used to analyse such a situation is category theory.

The description of rewriting systems in terms of category theory is rather complicated by the fact that certain rewrites commute (see for example the two rewrites $ab \to x$ in Example 1 below). In Section 2 we demonstrate that this difficulty disappears for string rewriting systems if they are modelled by 2-categories. The development of the usual theory of Church-Rosser string rewriting systems is carried out in Section 3 mainly to set notation and to demonstrate the simplicity of string rewriting.

The description of *term* rewriting systems is further complicated by the presence of substitution which is another operation which must interact correctly with composition of rewrites. Rydeheard and Stell [13] used 2-categories to model this situation, demonstrating that the laws of 2-category theory suffice to capture the interaction of substitution and rewriting. However, the chief complication caused by the presence of substitution (the need to prove that there are only finitely many critical pairs from which, using substitution, all other critical pairs can be obtained) remains.

In Section 4 we show that linear term rewriting systems are modelled well by 3-categories. Two dimensions of the 3-category are used for substitution and rewriting while the third dimension carries information about the structure of terms. The two dimensions satisfy the usual 2-categorical laws so substitution and rewriting work together correctly and the third dimension allows us to avoid the need for a proof of the existence of "most general critical pairs". This reduces the linear term rewriting systems to the simplicity of string rewriting systems but at one higher dimension.

Finally Section 5 makes several remarks on the utility of the 3-categorical approach and its relationship to earlier work.

2 String rewriting systems

This section reviews the well known theory of string rewriting systems and introduces a 2-categorical notation for such systems.

If Σ is a set, write Σ^* for the free monoid on Σ (the elements of Σ^* are words in the alphabet Σ).

A *string rewriting system* consists of an alphabet Σ and a set $R \subset \Sigma^* \times \Sigma^*$. We write an element $(s, s') \in R$ as $s \to s'$ and call it a *rewrite* of s to s'. An element $w \in \Sigma^*$ may be rewritten by finding a subword of w which matches the left hand side of a rewrite and replacing it with the corresponding right hand side to obtain some w'. We then abuse notation and write $w \to w'$ and we sometimes say that w and w' are $\to$-related. A sequence of such rewrites, $w \to w' \to \ldots \to w''$ is called a *derivation*.

Example 1 $\Sigma = \{a, b, c, x, y, z\}$ and $R = \{xyz \to a, ab \to x\}$. An example of a derivation beginning with $w = xyzbyzabc$ would be

$$xyzbyzabc \to abyzabc \to xyzabc \to xyzxc \to axc$$

Notice that the application of the rewrite $ab \to x$ in the context $abc \to xc$ could have been carried out at any position in the sequence of rewrites. Thus a distinct but essentially equivalent derivation would be

$$xyzbyzabc \to abyzabc \to abyzxc \to xyzxc \to axc.$$

Indeed the rewrite taking $abc \to xc$ may be thought of as occurring in parallel with the other rewrites. This can be made explicit if we take a 2-categorical perspective.

A 2-category is a one of the kinds of 2-dimensional categories. A 2-category consists of objects which we will indicate as points, arrows called 1-cells which go between objects and which compose in the usual way, and arrows called 2-cells which go between 1-cells and which compose both horizontally

$$\begin{tikzcd} \bullet \arrow[r, bend left=50] \arrow[r] \arrow[r, phantom, bend left=25, "\Downarrow"] & \bullet \arrow[r, bend left=50] \arrow[r] \arrow[r, phantom, bend left=25, "\Downarrow"] & \bullet \end{tikzcd}$$

and vertically

$$\begin{tikzcd} \bullet \arrow[r, bend left=50] \arrow[r] \arrow[r, bend right=50] \arrow[r, phantom, bend left=25, "\Downarrow"] \arrow[r, phantom, bend right=25, "\Downarrow"] & \bullet \end{tikzcd}$$

In a 2-category all compositions are associative, there are identity 1-cells and 2-cells and the horizontal and vertical compositions interact according to the "middle four interchange law" which says that in the situation

$$\begin{tikzcd} \bullet \arrow[r, bend left=50] \arrow[r] \arrow[r, bend right=50] \arrow[r, phantom, bend left=25, "\Downarrow"] \arrow[r, phantom, bend right=25, "\Downarrow"] & \bullet \arrow[r, bend left=50] \arrow[r] \arrow[r, bend right=50] \arrow[r, phantom, bend left=25, "\Downarrow"] \arrow[r, phantom, bend right=25, "\Downarrow"] & \bullet \end{tikzcd}$$

composing first horizontally and then vertically gives the same result as composing first vertically and then horizontally. Using these laws one can prove a *pasting theorem* which says that diagrams like the one in Example 2 below suffice to specify a unique 2-cell even though they can be interpreted as specifying many different sequences of horizontal and vertical compositions.

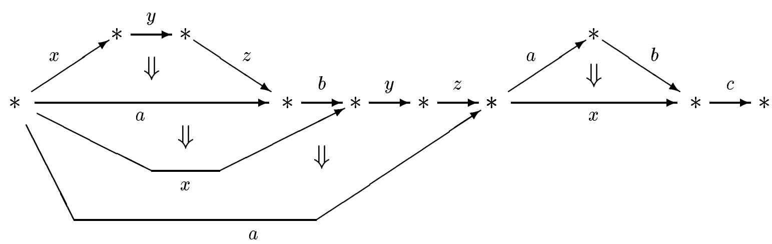
For an elementary introduction to 2-categories see [9] and for a proof of the pasting theorem and its generalization to higher dimensions see [7]. For readers with some acquaintance with category theory it may help to point out that a 2-category C is just a **Cat**-enriched category which means that if x and y are objects of C then the hom $C(x, y)$ is a category—its objects are the arrows from x to y and its arrows are 2-cells. Furthermore, the 'prototypic' 2-category has categories as objects, functors as 1-cells and natural transformations as 2-cells and is usually called **Cat** (despite the risk of confusion with the (ordinary) category **Cat** of categories and functors).

Example 2 As in Example 1 let $\Sigma = \{a, b, c, x, y, z\}$ and $R = \{xyz \to a, ab \to x\}$.

Consider the 2-category which has one object $*$, 1-cells generated by $a : * \to *$, $b : * \to *$, $c : * \to *$, $x : * \to *$, $y : * \to *$, $z : * \to *$, and 2-cells generated by

$$\begin{tikzcd} & * \arrow[dr, "b"] \arrow[d, phantom, "\Downarrow"] & \\ * \arrow[ur, "a"] \arrow[rr, "x"'] & & * \end{tikzcd} \qquad \text{and} \qquad \begin{tikzcd} & * \arrow[r, "y"] \arrow[dr, phantom, "\Downarrow"] & * \arrow[dr, "z"] & \\ * \arrow[ur, "x"] \arrow[rrr, "a"'] & & & * \end{tikzcd}$$

(where we repeat the single object $*$ for clarity). Then the 1-cells correspond to the elements of Σ^* and the 2-cells correspond to derivations except that derivations which are equivalent in the sense of Example 1 correspond to the same 2-cell. For example the derivations of Example 1 both correspond to the 2-cell



Given any string rewriting system (Σ, R) one can freely construct a 2-category $C_{\Sigma,R}$ as above using the techniques of [4]. Then the hom category $C_{\Sigma,R}(*, *)$ has as objects the elements of Σ^* and as arrows the derivations modulo the equivalence relation indicated in Example 1. It seems that this is the category in which the general theory of string rewriting systems belongs.

3 String rewriting systems' theory

The main question to ask about a string rewriting system (Σ, R) is: When does each word in Σ^* have a unique normal form under rewriting? This is because one of the principal applications of rewriting systems is the computation of congruence relations.

Explicitly, a rewriting system (Σ, R) generates a congruence $\sim_R$ by taking the reflexive symmetric transitive closure of the relation $\to$ on Σ^*. If each word in Σ^* has a unique normal form then two words are congruent if and only if they have the same normal form. Thus, to test for congruence it suffices to rewrite the words until their normal forms are found and then compare the normal forms.

The main results in the theory of string rewriting systems give sufficient conditions for the existence of unique normal forms. A general development of the theory can be found in [5]. Here we only review some basic results to establish notation and to illustrate the simplicity of the string rewriting theory.

Definition 3 Let (Σ, R) be a string rewriting system. A word $w \in \Sigma^*$ is called *irreducible* if w cannot be rewritten, that is, if the only arrow out of w in $C_{\Sigma,R}(*,*)$ is the identity. The word w is a *normal form* for the word w' if w is irreducible and there is an arrow (derivation) from w to w' in $C_{\Sigma,R}(*,*)$. We say that w' has *rank* n if the longest derivation starting at w' contains n rewrites and we call (Σ, R) *terminating* if each $w \in \Sigma^*$ has some finite rank.

Most of the theory of rewriting deals with terminating systems. Typically rewrites are chosen so that under some 'complexity measure' on the words, the rewrites are complexity reducing. Since the complexity measures are usually positive integers there is then a limit on the number of rewrites that can be sequentially applied to a given word. From now on we will deal with terminating systems.

Definition 4 A string rewriting system (Σ, R) is called *locally confluent* if for all $w, x, y \in \Sigma^*$ with $w \to x$ and $w \to y$ there is some $z \in \Sigma^*$ and derivations from x to z and from y to z. (This is sometimes called *the local confluence condition for w, x and y*.)

Lemma 5 *If (Σ, R) is locally confluent and terminating then each word in Σ^* has a normal form and it is unique.*

This Lemma would be computationally useful except that the check for local confluence would require infinitely many tests.

Definition 6 A string rewriting system (Σ, R) is called *locally confluent on overlaps* if, whenever the left hand sides of two rewrite rules can be superimposed to form a single word w, then w and the two words obtained by applying the rewrites to it, say x and y, satisfy the local confluence condition. In this case the two words x and y have been called a *critical pair*.

Notice that if R is finite then there are only finitely many critical pairs.

Lemma 7 *If (Σ, R) is locally confluent on overlaps and terminating then each word in Σ^* has a normal form and it is unique.*

(The fact that locally confluent on overlaps suffices for local confluence follows easily from the pasting theorem.)

Lemma 7 displays the simplicity of string rewriting systems—given a string rewriting system with finite R there are finitely many tests to check that it is locally confluent on overlaps. Then, provided that it is terminating, we know that normal forms exist and are unique and we can compute them by rewriting in any order.

4 Term rewriting systems

Definition 8 A *term rewriting system* consists of

1. A set of types
2. A set of typed variables
3. A set of typed functions (this much determines a *language*)
4. A set of pairs (α, β) of terms in the language such that the variables occurring in β are a subset of the variables occurring in α.

The elements of the set (4) are called *rewrite schema*. A term rewriting system is called *linear* if for each rewrite schema (α, β) each variable occurs at most once in α and at most once in β

Most of the string rewriting systems' theory lifts to term rewriting systems but there are some serious complications. These complications arise because the specified rewrites are only schema—in general, a single rewrite schema will specify infinitely many rewrites by substituting terms in place of one or more variables.

Example 9 Consider the term rewriting system whose types are given by the set $\{A\}$, variables by the set $\{u, v, w, x, y, z\}$ all of type A, functions by the set $\{f\}$ where f has type $A \times A \to A$, and rewrite schema are given by the set $\{(f(x, f(y, z)), f(f(x, y), z))\}$ (this is often called the associativity rewrite). Then the term

$$f(f(u, f(v, w)), f(x, f(y, z)))$$

can be rewritten in three distinct ways. By way of example, if we substitute $f(u, f(v, w))$ for x in the rewrite schema, x for y in the rewrite schema and $f(y, z)$ for z in the rewrite schema we can rewrite to obtain

$$f(f(f(u, f(v, w)), x), f(y, z)).$$

Complications 10 *1. Even if (4) is finite there are infinitely many distinct rewrites and so the test for local confluence is infinite.*

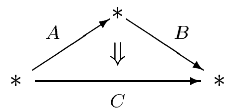
2. Rewriting is harder—one must match substituted instances of left hand sides of schema rather than just the left hand sides of the rewrite rules.

The first complication can be avoided by proving that for each pair of rewrite schema, if there is a term in which their left hand sides overlap, then there are infinitely many such terms but there are finitely many terms from which all the others can be obtained by substitution. These finitely many terms have been called *most general unifiers* (although this name has also been used in the more restrictive sense of the most general term which is a substituted instance of both of two given terms).

The proof of the existence of such a finite collection of terms played an important role in developing algorithms for term rewriting and critical pair completion in the 1960's because it permitted the lifting of the string rewriting systems' results of Section 3 to term rewriting systems. The advantage of the description of linear term rewriting systems outlined below is that it is immediately apparent that, when there are only finitely many rewrite schema, there are only finitely many possible overlaps and so there are only finitely many tests required to determine local confluence.

We will model linear term rewriting systems in 3-categories. A 3-category is a 2-**Cat**-enriched category. (We will say no more about the definition but rather trust the reader's intuition—there are arrows called 3-cells which go between 2-cells.)

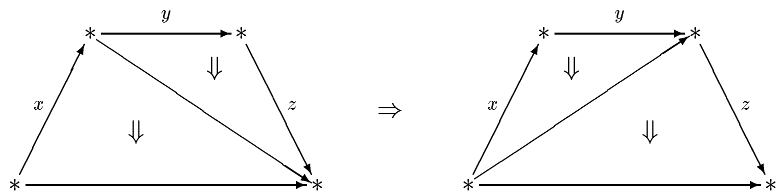
We will work in the 3-category which has a single object $*$, 1-cells generated by the types thought of as arrows $* \to *$, and 2-cells generated by the functions where a function $f : A \times B \to C$ is thought of as



Then the terms are 2-cells so it makes sense to ask that 3-cells be generated by the rewrite schema.

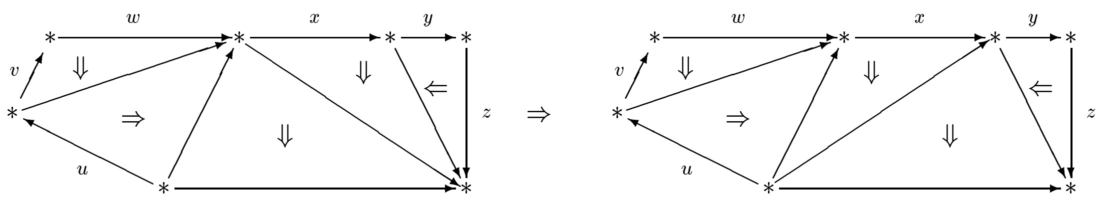
Example 11 The 3-category needed for Example 9 has one object, 1-cells corresponding to the natural numbers, 2-cells generated by triangles and 3-cells generated by

$$f(x, f(y, z)) \to f(f(x, y), z)$$



where the arrows, all of which correspond to the type A have been labelled with the variables whose place they hold.

The rewrite of Example 9 is then



Observation 12 *Notice that the only difference between the two 2-cells is in one quadrilateral and the difference is exactly the rewrite specified by the rewrite schema. There is no need to worry about substitution—the particular substituted instance of the rewrite schema that has been used is determined by what surrounds the rewrite quadrilateral and the actual rewrite which takes place is just the unsubstituted rewrite schema. Thus working in the 3-category we keep the substitution separate from the rewriting.*

Remark 13 The example that we have developed is single-sorted but there is nothing special about this. For example one can easily modify Example 9 to treat the two-sorted theory of a semigroup acting on a set (there will be two different f's, one will be the semigroup multiplication and the other the action, and there will be two different rewrite schema, one for the associativity of the multiplication and the other for the axiom of the action). The important point is that the rewites must be *linear* because we have given no indication of how to handle repeated variables.

We have indicated how linear term rewrite systems may be viewed as 2-dimensional string rewriting systems. The Complications 10 are no longer a problem: As in string rewriting systems one checks for local confluence on overlaps by simply overlapping the rewrites' left hand sides and if there are finitely many rewrite rules then there are finitely many tests, and we carry out rewriting by simply beginning with a 2-cell and matching sub-2-cells with left hand sides of rewrite schema.

5 Discussion

Remark 14

1. The development of the preceding section can be formalized using *pasting schemes* which are described in [7].
2. The reformulation in terms of category theory is intended to be of theoretical interest (this is as predicted by Buchberger in [3], page 25). It was observed by Paola Bonacina that this setting could provide an algorithm for computing most general unifiers. However, work of Smolka et al [15] has shown the existence of most general unifiers for systems with regular signatures and linear term rewriting systems can always be modified to guarantee regularity. Therefore a further proof would be largely redundant and Bonacina's suggestion has not been developed here.
3. The advantage of the reformulation can be seen by consulting Sections 3 and 5 of [10]. An example of the clarification obtained is the realization that Knuth and Bendix' Case 1 on page 274 of [10] is really just another instance of the disjoint rewrites that they had already treated (but they had to do it that way because of the terms as words approach).
4. A small modification of the well known theory of Section 3 to take account of when a diagram of derivations commutes leads to a new treatment of categorical coherence. I am currently preparing a paper describing this.

We end with a few words about related work.

The view of rewriting described here may be seen as a small part of a large movement: Theoretical Computer Science has long been founded in logic. This is appropriate because many of the important questions (termination, complexity, correctness, ...) are essentially logical questions. However, the recent rise of Categorical Computer Science (see e.g. [11]) has shown that much of Theoretical Computer Science is handled well by Category Theory. Indeed the development of Topos

Theory [1],[8] has to a large degree subsumed Logic under Category Theory—the logical is being replaced by the functional.

However, the work goes a step further because the use of higher dimensional categories permits the maintenance of the geometry of the terms (and this is why the need to prove the existence of a finite collection of critical pairs which generates all critical pairs disappears). In this respect the work follows Rydeheard and Stell [13] and Seely [14].

Finally the 3-categorical formulation of term rewriting systems suggests a framework in which Buchberger's suggestion [3] of an axiomatization of rewriting and critical pair completion in terms of *patterns* (certain specified 2-cells), *multipliers* (2-categorical composition) and *replacements* (certain specified 3-cells) can be carried out. A more detailed attempt at such an axiomatization can be found in Chapter 4 of [6] where it is used to prove coherence theorems in category theory.

6 Acknowledgements

The author gratefully acknowledges the assistance of the ARC and the advice of an anonymous referee.

References

[1] Barr M and C Wells, *Toposes, Triples and Theories*, Springer, 1985.

[2] Buchberger B, *An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal*, PhD Thesis, University of Innsbruck, 1965, (quoted in [3]).

[3] Buchberger B, History and Basic Features of the Critical-Pair/Completion Procedure, *J Symbolic Computation*, **3**, 3–38, 1987.

[4] Eilenberg S and R H Street, *Constructing free 2-categories*, manuscript in preparation.

[5] Huet G, Confluent reductions: Abstract properties and applications to term rewriting systems, *J Assoc Comp Mach*, **27**, 797–821, 1980.

[6] Johnson M, Pasting diagrams in n-categories with applications to coherence theorems and categories of paths, *Sydney Category Seminar Reports*, **88-14**, 1988.

[7] Johnson M, The combinatorics of n-categorical pasting, *J Pure and Applied Algebra*, **63**, 211–225, 1989.

[8] Johnstone P T, *Topos Theory*, Academic Press, 1977.

[9] Kelly G M and R H Street, Review of the elements of 2-categories, *Lecture Notes in Mathematics*, **420**, 75–103, 1974.

[10] Knuth D E and P B Bendix, Simple Word Problems in Universal Algebras, *Proc of the 1967 Conf on Computational Problems in Abstract Algebra*, Pergamon Press, 1970.

[11] Proceedings of the Category Theory and Computer Science Conferences, *Lecture Notes in Computer Science*, **240, 283, 389**, 1985,1987,1989.

[12] Robinson J A, A machine-oriented logic based on the resolution principle, *J Assoc Comp Mach*, **12**, 23–41, 1965.

[13] Rydeheard D E and J G Stell, Foundations of equational deduction: A categorical treatment of equational proofs and unification algorithms, *Lecture Notes in Computer Science*, **283**, 1987.

[14] Seely R A G, Modelling computations—a 2-categorical approach, *LICS Proceedings*, IEEE Publications, 1987.

[15] Smolka G, W Nutt, J A Goguen and J Meseguer, Order-sorted equational completion, *SEKI Report,* **SR-87-14**, Universitat Kauserslautern, 1987.

[16] Walters R F C, *Categories and Computer Science*, Carslaw Publications, 1991, to appear.

School of Maths and Computing
Macquarie University
N.S.W. 2109
AUSTRALIA