Rewriting procedures generalise to Kan extensions of actions of categories

Anne Heyworth
University of Wales, Bangor

1 Introduction

This is a brief account of work of Brown and Heyworth [1] on extensions of rewriting methods.

The standard expression of such methods is in terms of words w in a free monoid Δ^* on a set Δ. This may be extended to terms $x|w$ where x belongs to a set X and the link between x and w is in terms of an action. More precisely, we suppose a monoid A acts on the set X on the right, and there is given a morphism of monoids F: $A \to B$ where B is given by a presentation with generating set Δ. The result of the rewriting will then be normal forms for the *induced action* of B on $F_*(X)$. This gives an important extension of rewrite methods.

In fact monoids may be replaced by categories, and sets by directed graphs. This gives a formulation in terms of Kan extensions, or induced actions of categories, which we now explain.

2 Presentations of Kan Extensions

Let A be a category. A **category action** X of A is a functor $X : \mathsf{A} \to \mathsf{Sets}$. Let Bbe a second category and let $F : \mathsf{A} \to \mathsf{B}$ be a functor. Then an**extension of the action** X **along** F is a pair (K, ε) where $K : \mathsf{B} \to \mathsf{Sets}$ is a functor and $\varepsilon : X \to K \circ F$ is a natural transformation. The **Kan extension of the action** X **along** F is an extension of the action (K, ε) with the universal property that for any other extension of the action (K', ε') there exists a unique natural transformation $\alpha : K \to K'$ such that $\varepsilon' = \alpha \circ \varepsilon$.

The problem that has been introduced is that of "computing a Kan extension". In order to keep the analogy with computation and rewriting for presentations of monoids we propose a definition of a presentation of a Kan extension. The papers [2, 4, 5, 7] were very influential on the current work.

A **Kan extension data** (X', F') consists of small categories A, B and functors $X' : \mathsf{A} \to \mathsf{Sets}$ and $F' : \mathsf{A} \to \mathsf{B}$. A **Kan extension presentation** is a quintuple $\mathcal{P} := kan\langle \Gamma | \Delta | RelB | X | F \rangle$ where

1. Γ and Δ are (directed) graphs;
2. $X : \Gamma \to \mathsf{Sets}$ and $F : \Gamma \to P\Delta$ are graph morphisms to the category of sets and the free category on Δ respectively;
3. and $RelB$ is a set of relations on the free category $P\Delta$.

We say $\mathcal{P}$ **presents** the Kan extension (K, ε) of the Kan extension data (X', F') where $X' : \mathsf{A} \to \mathsf{Sets}$ and $F' : \mathsf{A} \to \mathsf{B}$ if

1. Γ is a generating graph for A and $X : \Gamma \to \mathsf{Sets}$ is the restriction of $X' : \mathsf{A} \to \mathsf{Sets}$
2. $cat\langle \Delta | RelB \rangle$ is a category presentation for B.
3. $F : \Gamma \to P\Delta$ induces $F' : \mathsf{A} \to \mathsf{B}$.

We expect that a Kan extension (K, ε) is given by a set KB for each $B \in \mathrm{Ob}\Delta$ and a function $Kb : KB_1 \to KB_2$ for each $b : B_1 \to B_2 \in \mathsf{B}$ (defining the functor K) together with a function $\varepsilon_A : XA \to KFA$ for each $A \in \mathrm{Ob}\mathsf{A}$ (the natural transformation). This information can be given in four parts:

- the set $\bigsqcup_B KB$;
- a function $\overline{\tau} : \bigsqcup_B KB \to \mathrm{Ob}\mathsf{B}$;
- a partial function (action) $\bigsqcup_B KB \times \mathrm{Arr}\mathsf{P} \to \bigsqcup_B KB$;
- and a function $\varepsilon : \bigsqcup_A XA \to \bigsqcup_B KB$.

Here $\bigsqcup_B KB$ and $\bigsqcup_A XA$ are the disjoint unions of the sets KB, XA over $\mathrm{Ob}\mathsf{B}$, $\mathrm{Ob}\mathsf{A}$ respectively; if $z \in KB$ then $\overline{\tau}(z) = B$ and if further $src(p) = B$ for $p \in \mathrm{Arr}\mathsf{P}$ then $z \cdot p$ is defined.

3 Rewriting for Kan Extensions

The main result of the paper defines rewriting procedures on the $P\Delta$-set

$$T := \bigsqcup_{B \in \mathrm{Ob}\Delta} \bigsqcup_{A \in \mathrm{Ob}\Gamma} XA \times P\Delta(FA, B).$$

Two kinds of rewriting are involved here. The first is the familiar $x|ulv \to x|urv$ given by a relation (l, r). The second derives from a given action of certain words on elements, so allowing rewriting $x|F(a)v \to x \cdot a|v$. Further,the elements x and $x \cdot a$ may belong to different sets. When such rewritingprocedures complete, the associated normal form gives in effect acomputation of what we call the *Kan extension defined by the presentation.*

Theorem 3.1 *Let $\mathcal{P} = kan\langle \Gamma | \Delta | RelB | XF \rangle$ be a Kan extension presentation, and let P, T, $R = (R_\varepsilon, R_K)$ be defined as above. Then the Kan extension (K, ε) presented by $\mathcal{P}$ may be given by the following data:*

i) the set $\bigsqcup_B KB = T / \overset{*}{\leftrightarrow}_R$,

ii) the function $\overline{\tau} : \bigsqcup_B KB \to \mathrm{Ob}\mathsf{B}$ induced by $\tau : T \to \mathrm{Ob}\mathsf{P}$, *iii) the action of* B *on* $\bigsqcup_B KB$ *induced by the action of* P *on* T*,*

iv) the natural transformation ε *determined by* $x \mapsto [x|id_{FA}]$ *for* $x \in XA,\ A \in \mathrm{Ob}\mathsf{A}$.

3.1 Reduction and critical pairs

To work with a rewrite system R on T we will require certain concepts of order on T. We give properties of orderings $>_X$ on $\bigsqcup_A XA$ and $>_P$ on $\mathrm{Arr}\mathsf{P}$ to enable us to construct an ordering $>_T$ on T with the properties needed for the rewriting procedures.

Given an admissible well-ordering $>_T$ on T it is possible to discuss when a reduction relation generated by a rewrite system is compatible with this ordering. It is a standard result that if a reduction relation is compatible with an admissible well-ordering, then it is Noetherian. A Noetherian reduction relation on a set is confluent if it is locally confluent(Newman's Lemma[16]). By standard abuse of notation the rewrite system R will be called complete when $\to_R$ is complete. Hence, if R is compatible with an admissible well-ordering on T and $\to_R$ is locally confluent then $\to_R$ is complete. By orienting the pairs of R with respect to the chosen ordering $>_T$ on T, R is made to be Noetherian. The problem remaining is testing for local confluence of $\to_R$ and changing R in order to obtain an equivalent confluent reduction relation.

We explain the notion of critical pair for a rewrite system for T, extending the traditional notion to our situation. In particular the overlaps involve either just R_T, or just R_P or an interaction between R_T and R_P.

A term $crit \in T$ is called **critical** if it may be reduced by two or more different rules. A pair $(crit1, crit2)$ of distinct terms resulting from two single-step reductions of the same term is called a **critical pair**. A critical pair for a reduction relation $\to_R$ is said to **resolve** if there exists a term res such that both $crit1$ and $crit2$ reduce to a res, i.e. $crit1 \xrightarrow{*}_R res$, $crit2 \xrightarrow{*}_R res$. If $t = x|b_1 \cdots b_n$, then a **part** of t is either a term $x|b_1 \cdots b_i$ for some $1 \leqslant i \leqslant n$ or a word $b_i b_{i+1} \cdots b_j$ for some $1 \leqslant i \leqslant j \leqslant n$. Let $(rule1, rule2)$ be a pair of rules of the rewrite system $R = (R_T, R_P)$ where $R_T \subseteq T \times T$ and $R_P \subseteq \mathrm{Arr}\mathsf{P} \times \mathrm{Arr}\mathsf{P}$. Then the rules are said to **overlap** when $rule1$ and $rule2$ may both be applied to the same term t in such a way that there is a part c of the term that is affected by both the rules.

There are five types of overlap for this kind of rewrite system, as shown in the following table:

#	rule1	in	rule2	in	overlap	critical pair
(i)	(s_1, u_1)	R_T	(s_2, u_2)	R_T	$s_2 = s_1 \cdot q$ for some $q \in \mathrm{Arr}\mathsf{P}$	$(u_1 \cdot q, u_2)$
(ii)	(l_1, r_1)	R_P	(l_2, r_2)	R_P	$l_1 = pl_2q$ for some $p, q \in \mathrm{Arr}\mathsf{P}$	(r_1, pr_2q)
(iii)					$l_1q = pl_2$ for some $p, q \in \mathrm{Arr}\mathsf{P}$	(r_1q, pr_2)
(iv)	(s_1, u_1)	R_T	(l_1, r_1)	R_P	$s_1 \cdot q = s \cdot l_1$ for some $s \in T, q \in \mathrm{Arr}\mathsf{P}$	$(u_1 \cdot q, s \cdot r_1)$
(v)					$s_1 = s \cdot (l_1q)$ for some $s \in T, q \in \mathrm{Arr}\mathsf{P}$	$(u_1, s \cdot r_1q)$

Overlap table

3.2 Completion procedure

We show: (i) how to find overlaps between rules of R; (ii) how to test whetherthe resulting critical pairs resolve; (iii) that if all the critical pairs resolve then this implies $\rightarrow_R$ is confluent; and (iv) that critical pairswhich do not resolve may be added to R without affecting theequivalence relation R defines on T. We have now set up and proved everything necessary for a variant of the Knuth-Bendix procedure, which will add rules to a rewrite system R resulting from a presentation of a Kan extension, to attempt to find an equivalent complete rewrite system R^C. The benefit of such a system is that $\rightarrow_{R^C}$ then acts as a normal form function for $\overset{*}{\leftrightarrow}_{R^C}$ on T.

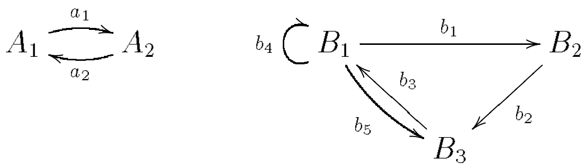
Theorem 3.2 *Let $\mathcal{P} = \langle \Gamma | \Delta | RelB | X | F \rangle$ be a finite presentation of a Kan extension (K, ε). Let $P := P\Delta$, $T := \bigsqcup_{\mathrm{Ob}\Delta} \bigsqcup_{\mathrm{Ob}\Gamma} XA \times \mathsf{P}(FA, B)$, and let R be the initial rewrite system for $\mathcal{P}$ on T. Let $>_T$ be an admissible well-ordering on T. Then there exists a procedure which, if it terminates, will return a rewrite system R^C which is complete with respect to the ordering $>_T$ and such that the equivalence relations $\overset{*}{\leftrightarrow}_R$, $\overset{*}{\leftrightarrow}_{R^C}$ coincide.*

The above procedure which attempts completion of a presentation of a

Kan extension has been implemented in GAP3.

4 Example of a GAP Session on the Rewriting Procedure

Here we give an example to show the use of the implementation. Let A and B be the categories generated by the graphs below, where B has the relation $b_1 b_2 b_3 = b_4$.



Let $X : \mathsf{A} \rightarrow \mathsf{Sets}$ be defined by $XA_1 = \{x_1, x_2, x_3\}$, $XA_2 = \{y_1, y_2\}$ with
$Xa_1 : XA_1 \rightarrow XA_2 : x_1 \mapsto y_1, x_2 \mapsto y_2, x_3 \mapsto y_1$,
$Xa_2 : XA_1 \rightarrow XA_2 : y_1 \mapsto x_1, y_2 \mapsto x_2$,
and let $F : \mathsf{A} \rightarrow \mathsf{B}$ be defined by $FA_1 = B_1$, $FA_2 = B_2$, $Fa_1 = b_1$ and $Fa_2 = b_3 b_2$. The input to the computer program takes the following form.

First read in the program and set up the variables:

```
gap> RequirePackage("kan");
gap> F:=FreeGroup("b1","b2","b3","b4","b5","x1","x2","x3","y1","y2");;
gap> b1:=F.1;;b2:=F.2;;b3:=F.3;;b4:=F.4;;b5:=F.5;;
gap> x1:=F.6;;x2:=F.7;;x3:=F.8;;y1:=F.9;;y2:=F.10;;
```

Then we input the data (choice of names is unimportant):

```
gap> OBJa:=[1,2];;
gap> ARRa:=[[1,2],[2,1]];;
gap> OBJb:=[1,2,3];;
gap> ARRb:=[[b1,1,2],[b2,2,3],[b3,3,1],[b4,1,1],[b5,1,3]];;
gap> RELb:=[[b1*b2*b3,b4]];;
gap> fOBa:=[1,2];;
gap> fARRa:=[b1,b2*b3];;
gap> xOBa:=[[x1,x2,x3],[y1,y2]];;
gap> xARRa:=[[y1,y2,y1],[x1,x2]];;
```

To combine all this data in one record (the field names are important):

```
gap> KAN:=rec( ObA:=OBJa, ArrA:=ARRa,  ObB:=OBJb, ArrB:=ARRb, RelB:=RELb,
              FObA:=fOBa, FArrA:=fARRa, XObA:=xOBa, XArrA:=xARRa );;
```

To calculate the initial rules do:

```
gap> InitialRules( KAN );
```

The output will be:

```
i= 1, XA= [ x1, x2, x3 ], Ax= x1, rule= [ x1*b1, y1 ]
i= 1, XA= [ x1, x2, x3 ], Ax= x2, rule= [ x2*b1, y2 ]
i= 1, XA= [ x1, x2, x3 ], Ax= x3, rule= [ x3*b1, y1 ]
i= 2, XA= [ y1, y2 ], Ax= y1, rule= [ y1*b2*b3, x1 ]
i= 2, XA= [ y1, y2 ], Ax= y2, rule= [ y2*b2*b3, x2 ]
[ [ b1*b2*b3, b4 ], [ x1*b1, y1 ], [ x2*b1, y2 ], [ x3*b1, y1 ],
  [ y1*b2*b3, x1 ], [ y2*b2*b3, x2 ] ]
```

This means that there are five initial ε-rules:
$(x_1|Fa_1, x_1.a_1|id_{FA_2})$, $(x_2|Fa_1, x_2.a_1|id_{FA_2})$, $(x_3|Fa_1, x_3.a_1|id_{FA_2})$,
$(y_1|Fa_2, y_1.a_1|id_{FA_1})$, $(y_2|Fa_2, y_2.|a_1 id_{FA_1})$,
i.e. $x_1|b_1 \to y_1|id_{B_2}$, $x_2|b_1 \to y_2|id_{B_2}$, $x_3|b_1 \to y_1|id_{B_2}$, $y_1|b_2b_3 \to x_1|id_{B_1}, y_2|b_2b_3 \to x_2|id_{B_1}$

and one initial K-rule: $b_1b_2b_3 \to b_4$.

To attempt to complete the Kan extension presentation do:

```
gap> KB( InitialRules(KAN) );
```

The output is:

```
[ [ x1*b1, y1 ], [ x1*b4, x1 ], [ x2*b1, y2 ], [ x2*b4, x2 ], [ x3*b1, y1 ],
  [ x3*b4, x1 ], [ b1*b2*b3, b4 ], [ y1*b2*b3, x1 ], [ y2*b2*b3, x2 ] ]
```

In other words to complete the system we have to add the rules

$$x_1|b_4 \to x_1, \quad x_2|b_4 \to x_2, \text{ and } x_3|b_4 \to x_1.$$

The result of attempting to compute the sets by doing:

```
gap> Kan(KAN);
```

is a long list and then:

```
enumeration limit exceeded: complete rewrite system is:
[ [ x1*b1, y1 ], [ x1*b4, x1 ], [ x2*b1, y2 ], [ x2*b4, x2 ], [ x3*b1, y1 ],
  [ x3*b4, x1 ], [ b1*b2*b3, b4 ], [ y1*b2*b3, x1 ], [ y2*b2*b3, x2 ] ]
```

This means that the sets KB for B in B are too large. The limit set in the program is 1000. (To change this the user should type `EnumerationLimit:= 5000` – or whatever, after reading in the program.) In fact the above example is infinite. The complete rewrite system is output instead of the sets. We can in fact use this to obtain regular expressions for the sets. In this case the regular expressions are:

$$\begin{array}{lcl}
KB_1 & := & (x_1 + x_2 + x_3)|(b_5(b_3{b_4}^*b_5)^*b_3{b_4}^* + id_{B_1}). \\
KB_2 & := & (x_1 + x_2 + x_3)|b_5(b_3{b_4}^*b_5)^*b_3{b_4}^*(b_1) + (y_1 + y_2)|id_{B_2}. \\
KB_3 & := & (x_1 + x_2 + x_3)|b_5(b_3{b_4}^*b_5)^*(b_3{b_4}^*b_1b_2 + id_{B_3}) + (y_1 + y_2)|b_2.
\end{array}$$

The actions of the arrows are defined by concatenation followed by reduction.
For example $x_1|b_5b_3b_4b_4b_5$ is an element of KB_3, so b_3 acts on it to give $x_1|b_5b_3b_4b_4b_5b_3$ which is irreducible, and an element of KB_1.

The general method of obtaining regular expressions for these computations will be given in a separate paper (see Chapter 4 of [8]).

5 Applications

Mac Lane wrote that "the notion of Kan extensions subsumes all the other fundamental concepts of category theory" in section 10.7 of [12] (entitled "All Concepts are Kan Extensions"). So the power of rewriting theory may now be brought to bear on a much wider range of combinatorial enumeration problems. Traditionally rewriting is used for solving the word problem for monoids. It has also been used for coset enumeration problems [15, 10]. It may now also be used in the specification of

i) equivalence classes and equivariant equivalence classes,

ii) arrows of a category or groupoid,

iii) right congruence classes given by a relation on a monoid,

iv) orbits of an action of a group or monoid.

v) conjugacy classes of a group,

vi) coequalisers, pushouts and colimits of sets,

vii) induced permutation representations of a group or monoid.

and many others.

In this paper we are concerned with the description of the theory and the implementation of the procedure with respect to one ordering. It is hoped to consider implementation of efficiency strategies and other orderings on another occasion. The advantages of our abstraction should then become even clearer, since one efficient implementation will be able to apply to a variety of situations, including some not yet apparent.

6 Further work, questions

6.1 Iteration

One of the pleasant features of the procedure we describe is that the input and the output are of a similar form. The consequence of this is that if the action K, given by (X', F'), has been defined on Δ, then given a second functor $G' : \mathsf{B} \to \mathsf{C}$ and a presentation $cat\langle\Lambda|RelC\rangle$ for C, it is straightforward to consider a presentation for the Kan extension data (K, G'). This new extension is in fact the Kan extension with data $(X', G' \circ F')$.

6.2 Kan Extensions and Noncommutative Gröbner Bases

It is well-known that rewrite systems are a special case of noncommutative Gröbner bases. It is possible to express a K-algebra presentation as an example of a Kan extension over K-categories but it is not clear how to apply Gröbner basis procedures to general Kan extensions of actions of K-categories.

6.3 Orderings on P-sets

In our paper we put stronger conditions on the ordering than may be necessary. Weaker conditions may or may not have an advantage. The only ordering we have implemented is the standard length-lexicographical. The choice of orderings may be wider than with ordinary rewriting, and this has not been investigated.

6.4 Language Theory

The actions in question are category actions on sets. Thus for each object of B we wish to specify a set, and for each arrow of B we require a function defined on the sets. In theory it is fine to specify sets by equivalence classes of a larger set, with a normal form function. In practice we may wish to get hold of an expression for all the normal forms. When the sets are finite we can use a basic enumeration procedure, but when the sets are infinite, enumeration is not an answer. In this case an automaton can be constructed from the complete rewrite system and language equations can be obtained and manipulated to obtain a *regular expression* for the normal forms of the elements of each set. It would be nice to program this!

6.5 Automatic Kan Extensions

Given the existing and current work on automatic groups, semigroups and coset systems it is natural to ask: what does the concept of automatic mean in terms of a Kan extension? An automatic coset system consists of "a finite state automaton that provides a name for each coset, and a set of finite state automata that allow these cosets to be multiplied by the group generators" [15]. We would expect therefore that an automatic Kan extension system would consist of a finite state automata for each set KB that provides a name for each element of the set, and a finite state automaton for each arrow on Δ that allows the sets to be acted upon by the arrows of B.

References

[1] R.Brown and A.Heyworth: "Using Rewrite Systems to Compute Kan Extensions and Induced Actions of Categories", `http://xxx.soton.ac.uk/abs/math.CO/9903032` *UWB Math Preprint 98.14*

[2] M.R.Bush, M.Leeming and R.F.C.Walters : "Computing Left Kan Extensions", *Journal of Symbolic Computation, 11 p11-20* (1997).

[3] R.V.Book and F.Otto : "String-Rewriting Systems", *Springer-Verlag, New York*, (1993).

[4] S.Carmody and R.F.C.Walters : "The Todd-Coxeter Procedure and Left Kan Extensions", *Research Reports of the School of Mathematics and Statistics, The University of Sydney p90-19*, (1990). with M.Leeming : *Journal of Symbolic Computation, 19 p459-488* (1995).

[5] S.Carmody and R.F.C.Walters : "Computing Quotients of Actions on a Free Category", *in A.Carboni, M.C.Pedicchio, G.Rosolini (eds), Category Theory, Proceedings of the Int. Conf. Como, Italy 22-28 July 1990, Springer-Verlag*, (1991).

[6] D.B.A.Epstein, J.W.Cannon et al: "Word Processing in Groups",
Boston : Jones and Bartlett Publishers, (1992).

[7] M.Fleming, R.Gunther and R.Rosebrugh : "User Guide for the Categories Database and Manual", *anonymous* `ftp://sun1.mta.ca/pub/papers/rosebrugh/catdsalg.dvi,tex` *and* `/catuser.dvi,tex` (1996).

[8] A.Heyworth: "Applications of Rewriting Systems and Gröbner Bases to Computing Kan Extensions and Identities Among Relations", *PhD thesis, UWB Math Preprint 98.23*, (1998).
`http://xxx.soton.ac.uk/abs/math.CT/9812097`

[9] D.F.Holt : "Knuth-Bendix in Monoids, and Automatic Groups", *Mathematics Institute, University of Warwick* (1996).

[10] D.F.Holt and D.F.Hurt : "Computing Automatic Coset Systems and Subgroup Presentations", *Journal of Symbolic Computation* (1996).

[11] J.Hopcroft and J.Ullman: "Introduction to Automata Theory, Languages and Computation", *Addison-Wesley, Reading, MA*, (1979).

[12] S.Mac Lane : "Categories for the Working Mathematician", *Springer-Verlag* (1971).

[13] B.Mitchell : "Rings with many objects", *Academic Press vol.8 no.1* (1972).

[14] T.Mora : "Gröbner Bases and the Word Problem", *University of Genova* (1987).

[15] I.D.Redfern : "Automatic Coset Systems", *PhD thesis, University of Warwick*, (1993).

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