USING COMPLEXITY FUNCTIONS TO GENERATE NORMAL FORMS FOR FREE ALGEBRA TERMS

ALEXANDRU PELIN
COMPUTER AND INFORMATION SCIENCES
TEMPLE UNIVERSITY

1. INTRODUCTION

Multisorted algebras were first introduced by Birkhoff and Lipson [1] as a generalization of the concept of universal algebra to more than one sort.

An important problem for a free multisorted algebra F with a set of equations E is the word problem, i.e. the problem of determining, for any terms t and u in F if t and u fall in the same congruence class under E.

Knuth and Bendix [7] present a method of solving the word problem for certain classes of equations by transforming the equations into reduction rules. This is done by considering one side ρ of an equation (e) $\lambda \approx \rho$ to be "simpler" than λ. Knuth and Bendix define a well ordering $>$ on the set of terms of the algebra and interpret "simpler" to mean that $\lambda > \rho$. Such a reduction is written $\lambda \to \rho$.

A term t reduces in one step to a term t^1 if there is a reduction rule (r) $\lambda \to \rho$ and an address u in the domain of the term t such that the subterm of t at address u is an instance of λ and t^1 is the term obtained from t by replacing the subterm at address u by the corresponding instance of ρ. We write $t \xrightarrow[r,u]{} t^1$ or $t \to t^1$ for such a reduction.

A term t reduces to a term t_n if there is a sequence $t_0 \to t_1 \to t_2 \to \ldots \to t_{n-1} \to t_n$ such that for all i, $0 \le i \le n-1$, t_i reduces in one step to t_{i+1}. We write $t_0 \xrightarrow{*} t_n$ for the multiple step reduction.

In order to solve the word problem, Knuth and Bendix [7] require that the system of reductions be confluent i.e. for all pairs of reductions $t_2 \xleftarrow{*} t_1 \xrightarrow{*} t_3$ there is a term t_4 such that $t_2 \xrightarrow{*} t_4 \xleftarrow{*} t_3$.

A sufficient condition for a system of reductions to be confluent is for the system to be confluent for all critical pairs. A critical pair is a pair $< \ell_1, \ell_2 >$ where (1) $\ell_1 \to r_1$ and (2) $\ell_2 \to r_2$ are reductions and ℓ_1 has an address u such that the subterm of ℓ_1 at address u is non-trivial (i.e. it is not a variable) and it is ϕ-unifiable with ℓ_2. We say that two terms t_1 and t_2 are ϕ-unifiable if there are substitutions s and s^1 such that $s(t_1) = s^1(t_2)$.

For all critical pairs $<\ell_1, \ell_2>$ the pair of reductions $\ell_1[u \leftarrow r_2] \xleftarrow[2,u]{} \ell_1 \xrightarrow[1,\varepsilon]{} 1$, must be confluent. By $\ell_1[u \leftarrow r_2]$ we mean the term obtained from ℓ_1 by replacing its subterm at address by r_2.

Based on the fact that a system of reductions is confluent if it is confluent for the set of critical pairs, Knuth and Bendix [7] give an algorithm for generating a complete set of reductions.

This is done by defining a well ordering on the set of ground terms of the algebra F. This well ordering transform the equations in E into reductions. By imposing the condition that the critical pairs of these reductions are confluent we obtain new reductions. If these reductions cannot be obtained from the existing set of reductions they are added to the set.

The algorithm stops when all reductions generated by the critical pairs are already in the set.

Our approach extends Knuth and Bendix's work in several ways. We start with a finite set of axioms A_0 over a free algebra F. Unlike Knuth and Bendix [7] we do allow conditional equations.

Let L_0 be the set of ground terms in F. Our object is to construct a representative function $Rep: L_0 \to L_0$ such that $t_1 \approx_{A_0} t_2$ if and only if $Rep(t_1) = Rep(t_2)$.

We differ from Knuth and Bendix since we do not require that Rep deal with all axioms in the same step. We define Rep as a composition of steps $S_1 \cdot S_2 \cdot \ldots \cdot S_n$. Each S_i takes as input a pair (L_{i-1}, A_{i-1}) where L_{i-1} is a set of terms from L_0 and A_{i-1} is a set of conditional equations. S_i selects some set of (conditional) equations that it wants to eliminate. These equations can be axioms from A_{i-1} or theorems obtained from the axioms in A_{i-1}. Once we select the equations to be eliminated we must find a well ordering on the set of terms in L_{i-1} which transforms the conditional equations into meta-reductions. This is done by defining a function $\delta_i : L_{i-1} \to N^{k_i}$ where k_i is some natural number. On N^{k_i} we define a well ordering $>$. Knuth and Bendix [7] require that k=2 and the relation $t_1 > t_2$ holds if either t_1 has weight greater than t_2 or the weights are equal and t_2 precedes t_1 in the lexiographical ordering introduced by the symbols of the free algebra.

They define the weight of a term as the sum of the weights of the symbols in the term, each symbol having a fixed weight.

The only requirements that we impose on δ are that δ is recursive, δ has the subterm property i.e. if t_1 is a subterm of t_2 then $\delta(t_2) > \delta(t_1)$ and the well ordering $>$ on N^{k_i} is recursive.

We call δ_i a complexity function for L_{i-1} over $(N_i^{k_i}, >)$. We say that a complexity function δ strongly suits an equation $\ell \simeq r$ if for all ground substitutions s, $\delta(s(\ell)) > \delta(s(r))$ or for all ground substitutions s, $\delta(s(r)) > \delta(s(\ell))$. A complexity function δ strongly suits a conditional equation $e_1, e_2, \ldots, e_n \to e$ if δ strongly suits e and for all equations ℓ_i, $1 \le i \le n$, for all ground substitutions s, $\delta(s(r)) > \delta(s(\ell_i))$. The complexity of an instance of an equation $s(\ell) \simeq s(r)$ is defined as $\max\{\delta(s(\ell)), \delta(s(r))\}$.

A weaker version of the suitability concept which is useful in treating conditional axioms is the concept of weak suitability defined below.

A complexity function δ weakly suits an equation $\ell \simeq r$ if for all ground substitutions S, $\delta(s(\ell)) = \delta(s(r))$ implies that $s(\ell) = s(r)$, that is, $s(\ell)$ and $s(r)$ are identical.

Both strong and weak suitability are used to generate meta-reductions. A meta-reduction has the form $(C) \Rightarrow \ell \to r$, where C is a recursive predicate in a meta-language which contains names for the well order $(N^{k_i}, >)$, the complexity functions and ℓ and r are terms in L_{i-1}. For all such meta-reductions, $\delta(s(\ell)) > \delta(s(r))$ if the condition C evaluates to true.

Once we have selected the set of equations E_i and we found a complexity function that suits each equation in E_i we define S_i to be the top-down reduction extension of the set R_i of reductions generated by E_i and δ.

The top-down reduction extension α of a set of meta rules R_i is defined as follows:

For every term t in L_{i-1}, we have the following cases:

(1) If for a meta-rule $(C) \Rightarrow \ell \to r$ and a ground substitution s, $s(\ell) \equiv t$ and C evaluates to true, then $\alpha(t) = \alpha(s(r))$.

(2) If case 1 does not apply and t has the form $g(t_1, \ldots, t_n)$, then compute recursively $\alpha(t_1), \ldots, \alpha(t_n)$. If for any i, $1 \le i \le n$, $\alpha(t_i) \ne t_i$ then $\alpha(t) = \alpha(g(\alpha(t_1), \ldots, \alpha(t_n)))$.

(3) If neither of the above cases applies, then $\alpha(t) = t$. In case 3, t is called an atom.

We define L_i to be the range of S_i and A_i to be the system of axioms obtained by applying S_i to the axioms in A_{i-1}.

If the set E_i is well chosen some axioms from A_{i-1} become identities in A_{i-1} and thus are eliminated. If this is not possible a proper choice for E_i will give a simpler form for L_i or for the axioms in A_i. This way we can see S_i as a function from (L_{i-1}, A_{i-1}) onto (L_i, A_i). The last step function is $R_n : (L_{n-1}, A_{n-1}) \to (L_n, \phi)$, L_n being the set of normal forms. Thus Rep can be seen as the composition of the function given by the sequence:

$$(4) \quad \langle L_0, A_0 \rangle \xrightarrow{S_1} \langle L_1, A_1 \rangle \to \ldots \xrightarrow{S_n} \langle L_n, \phi \rangle .$$

We say that a top-down reduction extension $S_i : \langle L_{i-1}, A_{i-1} \rangle \to \langle L_i, A_i \rangle$ has the α-property for L_{i-1} if for every operator f of rank n and for every n terms $t_1, t_2, \ldots, t_n$ in L_{i-1}, for every j, $1 \le j \le n$, if $f(t_1, \ldots, t_n) \in L_{i-1}$ then $S_i(f(t_1, \ldots, t_j, \ldots, t_n)) = S_i(f(t_1, \ldots, S_i(t_j), \ldots, t_n))$. It can be shown that if the composition $S_1 \cdot S_2 \cdot \ldots S_n$ has the α-property, then it has the representation property for $\langle L_0, A_0 \rangle$.

We will give criteria for obtaining "useful" theorems in E_i. There are basically two methods.

The first method is to force confluence in the set of rules associated with $\langle L_{i-1}, A_{i-1} \rangle$ using the method of critical pairs of Knuth and Bendix [7]. However, the resulting equation is not transformed into a rewrite rule but added to E_i.

The second method is to use conditional meta-rules. The complexity functions guarantee that the reductions in R_i terminate.

Using conditional meta-rules allows us to deal with system rules in which rules which increase the complexity are allowed, provided that the number of applications of such rules is finite.

We list below some of the advantages of our method over the Knuth-Bendix [7] approach.

1) The method is a stepwise refinement technique and, at each step, the complexity functions may be different.

2) The method allows one to characterize both the normal forms and the intermediate forms.

3) A problem can be reduced to an already solved problem.

For example, if one wants to compute the prenex conjunctive normal form of a first-order sentence, one can proceed as follows:

(1) Relabel variables which appear more than once.
(2) Push negations all the way inward.
(3) Push quantifiers in front.
(4) Distribute over .
(5) Eliminate duplicates and order the conjuncts.

If one attempted to perform these 5 steps in a single transformation, it would be very complicated. Also, if one already knows how to compute the conjunctive normal form for propositional logic, after step 3, the problem is reduced to one whose solution is known.

The complexity functions serve a double role. They can be used as the "control part" of the algorithm for computing normal forms, and they can be used to prove termination. Since the complexity functions that suit an equation are not unique we have flexibility in choosing the desired normal form.

2. THE FORMALISM

We will follow the notation and definitions found in Huet and Oppen [6]. Given a finite signature (S, Σ, τ), the initial algebra T_Σ is defined in the usual way ([6]). Terms in T_Σ will be represented in prefix form. Then, the set T_Σ^s of terms of sort s is a deterministic context free language.

Given an S-sorted set of variables V, the free algebra over V is denoted as $T_\Sigma(V)$ and consists of terms with variables.

A Σ-equation of sort s is a pair $\langle M,N \rangle$ of terms in $T_\Sigma(V)^s$, and will also be denoted as $M \approx N$. A conditional equation is an expression of the form $e_1, e_2, \ldots, e_n \Rightarrow \ell$, where $e_1, e_2, \ldots, e_n, e$ are equations.

A presentation is a triple $P = \langle \Sigma, V, E \rangle$, where Σ is a finite signature, V an S-indexed set of variables and E a finite set of conditional equations. Substitutions and E-unification are defined as in Huet and Oppen [6]. The concept of complexity function was defined in Introduction and will not be repeated here.

Given a complexity function h, for an operator f such that $\tau(f) = s_1 \times s_2 \times \ldots \times s_n \to s$, we say that h is monotone in f if for every i, $1 \le i \le n$, and all terms $t_1 \in T_\Sigma^{s_1}, \ldots, t_n \in T_\Sigma^{s_n}$, the following condition holds:

(1) $h(t_i^1) > h(t_i)$ implies $h(f(t_1, \ldots, t_i^1, \ldots, t_n)) > h(f(t_1, \ldots, t_i, \ldots, t_n))$

We say that a complexity function is monotone if it is monotone for every operator in Σ.

The notions of strong and weak suitability were introduced in the Introduction and are not repeated here. A complexity function h strongly (weakly) suits a set of (conditional) equations if it strongly (weakly) suits every (conditional) equation in the set. Given a set of (conditional) equations E and a complexity function h which suits E (weakly or strongly we define the set of rules associated with E under h, denoted as $R(E,h)$, or for short R, as follows:

(2) If $R \approx r \in E$ and h strongly suits $\ell \approx r$ then
 (i) If for all s, $h((\ell)) > h(s(r))$, then $\ell \to r$ is in $R(E,h)$
 (ii) otherwise $r \to \ell$ is in $R(E,h)$

(3) If $\ell \approx r \in E$ and h weakly suits $\ell \approx r$ then both metareductions $(h(\ell) > h(r)) \Rightarrow \ell \to r$ and $(h(r) > h(\ell)) \Rightarrow r \to \ell$ are in $R(E,h)$

(4) If $e: e_1, \ldots, e_n \to \ell \approx r \in E$ and h strongly suits e then $(\alpha(\ell_1) = \alpha(r_1) \wedge \ldots \alpha(\ell_n) = \alpha(r_n)) \Rightarrow \ell \to r$ is in $R(E,h)$ if $h(s(\ell)) > h(\alpha(r))$ for all ground substitutions s, or $(\alpha(\ell_1) = \alpha(r_1) \ldots \alpha(\ell_n) = \alpha(r_n)) \Rightarrow r \to \ell \in R(E,h)$ otherwise. We assume that the equations ℓ_i have the form $\ell_i \approx r_i$.

(5) If $e: e_1, \ldots, e_n \Rightarrow \ell \approx r \in E$ and h weakly suits e then both meta rules $(h(\ell) > h(r)) \wedge > \alpha(\ell_1) = \alpha(r_1) \wedge \ldots \wedge \alpha(\ell_n) = \alpha(r_n)$ $\alpha\, \ell \to r$ and $(h(r) > h(\ell) \wedge \alpha(\ell_1) = \alpha(r_1) \ldots \wedge \alpha(\ell_n) = \alpha(r_n) \Rightarrow r \to \ell$ are in $R(E,h)$

We say that $R(E,h)$ is functional if for all terms $t \in T_\Sigma$ and pairs of meta-rules (C_1) $\ell_1 \to r_1$ and $(C_2$ $\ell_2 \to r_2$ in $R(E,h)$, if there exists substitutions s_1 and s_2 such that $s_1(\ell_1) = s_2(\ell_2)$ and $C_1(t)=(2(t)$=true, then $s_1(r_1) = s_2(r_2)$

The set $R(E,h)$ is linear if for any meta-rule (C) $\ell \to r$, every variable occurs at most once in ℓ. The top-down reduction extension α of $R(E,h)$ has been defined in the Introduction.

3. TECHNICAL RESULTS

In Lemmes 1-7, E is assumed to be a finite set of Σ-equations, h is a complexity function that weakly suits E, R is the set of meta-rules associated with E under h, and β is the top-down reduction extension of R to T_Σ.

Lemma 1
If h is monotone then β is a recursive relation from T_Σ to T_Σ.

Lemma 2
If h is monotone and R is functional then β is a recursive function from T_Σ to T_Σ.

Lemma 3
If h is monotone then for all terms t in T_Σ, $\beta(\beta(t)) = \beta(t)$.

Lemma 4
If h is monotone then for all terms t in T_Σ, $h(t)= h(\beta(t))$ if and only if t is an atom.

Lemma 5
If h is monotone, f is an operator of type $\tau(f)=s_1 X\, s_2 X \ldots X\, s_n \to s$, if $f(v_1,\ldots,v_n)$ and ℓ are not unifiable for any variables $v_1,\ldots,v_n$ (with each v_i of sort s_i) and meta-rule (C) $\ell \to r$, then $\beta(f(t_1,\ldots,t_n)) = f(\beta(t_1),\ldots,\beta(t_n))$.

If the number of variables of each sort is unbounded, the unification condition can be replaced by:

$f(v_1,\ldots,v_n)$ and ℓ are not unifiable for an n-tuple of distinct variables $v_1,\ldots,v_n$, each v_i of sort s_i.

A set R of meta-reductions is said to be pure if it does not contain conditional meta-rules and h strongly suits E.

Lemma 6
Let R be pure and linear, h be monotone and L a subset of T_Σ such that $\beta(L)$ is a subset of L. If L is accepted by a (deterministic) bottom-up finite tre automaton then $\beta(L)$ is also accepted by a (deterministic) bottom-up finite tree automaton.

Lemma 7
If the terms of the language L in lemma 6 are represented in prefix notation then $\beta(L)$ is accepted by a deterministic push down automaton.

The proofs of lemmas 1-7 can be found in Pelin and Gallier [9].

If R is not linear or contains conditional equations, $\beta(L)$ is not necessarily deterministic or even context-free.

Lemma 8
Let h be monotone. The top-down reduction extension β is a function with the α-property if and only if R is locally confluent.

Next, we present examples illustrating the above techniques and results.

Example 1 (Stacks of natural numbers with errors)
The set of sorts is S={ nat,stack },
Σ={pop, push, top, 0, succ, $\wedge$, e,E }, and the typing function is:

τ (o) = τ(e) = $\to$ nat
τ ($\wedge$) = τ(E) = $\to$ stack
τ (push) = stack x nat $\to$ stack
τ (pop) = sack $\to$ stack
τ (top) = stack $\to$ nat
τ (succ) = nat $\to$ nat

Variables of sort nat will be denoted as n_k and variables of sort stack as s_k. The axioms for the data type stack of natural numbers are:

1. push E n $\simeq$ E
2. push s e $\simeq$ E
3. pop E $\simeq$ E
4. pop $\wedge$ $\simeq$ E
5. top $\wedge$ $\simeq$ E e
6. top E $\simeq$ e
7. pop push so $\simeq$ s
8. pop push sn $\simeq$ s $\to$ pop push s succ n $\simeq$ s
9. top push n $\simeq$ n
10. top push s n $\simeq$ n, top push sm $\simeq$ m $\to$ top push s n m $\simeq$ m
11. succ e $\simeq$ e.

The symbol $\wedge$ stands for the empty stack, E for the error stack, o is the natural number zero, succ is the successor function, e is the error natural number, push is the push function, pop pops the top of the stack and top returns the top of the stack (without altering the stack). Axioms 1,2,3,6 and 11 state that once an error occurred any subsequent operation will yield an error as result. Axioms 4 and 5 state that poping or retrieving the top of the empty stack yields an error. Axioms 7 and 8 are weaker versions of pop push sn $\simeq$ which is not valid for stacks with errors. Axioms 9 and 10 are weaker versions of top push sn $\simeq$ n which holds for s $\neq$ E. However, s $\neq$ E $\to$ top push sn $\simeq$ n is not accepted by our formalism which requires that all formulas be either equations or implications $e_1,\ldots,e_n \rightarrow e$ where $e_1,\ldots,e_n,e$ are equations.

It can be verified that the complexity function length (length of a string) strongly suits (A). It can also be verified that the set of reductions obtained from (A) is functional. Hence, by Lemma 2, the top down reduction β is recursive. In order to determine the range of β, Lemma 9 can be proved.

Lemma 9

For all stacks s and natural numbers n:

$$\beta(\text{pop push s n}) \neq \beta(s) \Rightarrow \beta(n) = e \wedge$$
$$\beta(\text{top push s n}) \neq \beta(n) \Rightarrow \beta(s) = E \wedge$$
$$\text{length}\ (\beta(\text{pop s})) < \text{length}\ (\text{pop s}) \wedge$$
$$\text{length}\ (\beta(\text{top s})) < \text{length}\ (\text{top s})$$

Lemma 10

The range of β is a context-free language.

Lemma 11

β has the representation property for $\langle L_0, A\rangle$ where L_0 is the language generated by the context free grammar given below:

$$S \rightarrow \Lambda \mid E \mid \text{push SN} \mid \text{pop S}$$
$$N \rightarrow e \mid o \mid \text{top S} \mid \text{succ N}$$

Example 2 (Free commutative semigroup with an infinite number of generators)

The set sorts is $S = \{N\}$, $\Sigma = \{g, I, +\}$, and the typing function is:

$$\tau(g) = \rightarrow N$$
$$\tau(I) = N \rightarrow N$$
$$\tau(+) = N \times N \rightarrow N$$

The set L_0 of terms in prefix form in the initial algebra T_Σ is given by the following context-free grammar:

$$S^1 \rightarrow G \mid S$$
$$G \rightarrow g \mid IG$$
$$S \rightarrow +SS$$

The set of axioms (C) is given below:

1. $+ xy \simeq + yx$
2. $+ x + yz \simeq + + xyz$

First, we pick

$E_1 = \{ + x = yz \simeq ++ xyz \}$. We can check that the complexity function $h_1: L_0 \rightarrow N$ defined below strongly suits E_1.

$$h_1(g) = 1$$
$$h_1(Ix) = 1$$
$$h_1(+ xy) = 1 + h_1(x) + 2h_1(y)$$

The complexity function h_1 transforms E_1 into the reduction

$R_1 = \{ + x + yz \rightarrow ++ xyz \}$. Let β_1 be the top-down reduction extension of R_1.

$A_1 = \beta_1(A_1)$ has the form $\{\beta_1(+xy) \simeq \beta_1(+yx) \mid$ for all ground terms x and y in $L_0\}$.

In order to eliminate the set of axioms A_2, we use reductions obtained by forcing confluence on the set of rules obtained by orienting the axioms of A from left to right.

Since $+ x + yz \Rightarrow ++ xyz$, $+ x + yz \Rightarrow + x + zy \Rightarrow ++ xzy$, we use the equation $++ xyz \simeq ++ xzy$. Let us call the terms of the form g or $I^n g$ literals. We choose $E_2 = \{++ xyz \simeq ++ xzy \mid$ for all literals $y,z\ L_1\} \cup \{ + yz \simeq + zy \mid$ for all literals $y,z\ L_1\}$. Let $h_2: L_1 \rightarrow N$ be

$$h_2(g) = 1$$
$$h_2(I^n g) = n + 1$$
$$h_2(+ xy) = 1 + 2h_2(x) + h_2(y).$$

We can show that h_2 weakly suits E_2. The system of meta-rules obtained from h_2 is $R_2 = \{ (h_2(y) > h_2(z) \Rightarrow ++ xyz \rightarrow ++ xzy,\ (h_2(y) > h_2(z)) \Rightarrow + yz \rightarrow + zy \}$. Let β_2 be the top-down reduction extension of R_2 to L_1. We can show that $\beta_2(\beta_1(+xy)) = \beta_2(\beta_1(+yx))$ for all $x, y \in L_0$. Hence, $A_2 = \beta_2(A_1) = \emptyset$. We can also show that $\beta_1 \cdot \beta_2$ has the α property. Thus $\beta_1 \cdot \beta_2$ has the representation property for $\langle L_0, C\rangle$.

More examples can be found in Pelin and Gallier [9].

4. CONCLUSIONS

Complexity functions play an important role in computing normal forms. Various authors such as Book [2], Gallier and Book [4], Huet [5], Lankford and Ballantyne [8], Knuth and Bendix [7] and others have used complexity functions for generating reductions. Dershowitz [3] and Plaisted [10] use particular complexity functions to prove termination of rewriting systems. In our approach, complexity functions serve both to prove termination and generate reductions.

The definition of a complexity function as presented in this paper is very general and it would be useful to identify which classes of complexity functions suit particular types of axioms. For example, we have shown that linear functions (functions of the form $(ft_1,\ldots,t_n)) = C_0 + C_1\ (t_1) + \ldots + C_n\ (t_n))$ can be used for associativity and commutativity axioms.

However, the distributivity axiom $* x + yz \simeq + * xy * xz$ require a quadratic function to obtain the reduction rule $* x + yz \rightarrow + * xy * xz$. The following function does the job:

$$h(*\ xy) = 2h(x)h(y)$$
$$h(+\ xy) = 1 + h(x) + h(y)$$

Also, some axioms, such as the commutativity axiom for a groupoid with an infinite number of generators, require complexity functions over N^k, for $k > 1$.

Another area of research is to investigate classes of axioms for which the set of normal forms can be characterized by context free languages.

5. BIBLIOGRAPHY

[1] Birkhoff, G. and Lipson, J.D., Heterogeneous Algebras, Journal of Combinatorial Theory 8 (1970), 115-133.

[2] Book, R., Confluent and Other Types of True Systems, JACM 29 (1982), 171-182.

[3] Deshowitz, N., Orderings for Term-Rewriting Systems, Theoretical Computer Science 17 (1982), 279-301.

[4] Gallier, J. H. and Book, R., Reductions in Tree-Rewriting Systems, submitted for publication to Theoretical Computer Science (1983).

[5] Huet, G., Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems, JACM 27 (4) (1980).

[6] Huet, G. and Oppen, D., Equations and Rewrite Rules, in Formal Languages: Perspective and Open Problems, R. V. Book Ed., Academic Press (1980), 349-405.

[7] Knuth, D. and Bendix, P., Simple Word Problems in Universal Algebras, in Computational Problems in Abstract Algebra, Leach, J., Ed., Pergamon Press (1970), 263-297.

[8] Lankford, D. S. and Ballantyne, A. M., Decision Procedures for Simple Equational Theories with Permutative Reductions, Report ATP-37, Department of Mathematics and Computer Science, University of Texas, Austin (1977).

[9] Pelin, A. and Gallier, J. H., Computing Normal Forms Using Complexity Functions over N^k, in preparation.

[10] Plaisted, D., Well-Founded Orderings for Proving Termination of Systems of Rewrite Rules, Report R-78-932, Department of Computer Science, University of Illinois, Urbana, IL (1978).