An Efficient Strategy for Word-Cycle Completion in Finitely Presented Groups

Luiz M. R. Gadelha Jr*
Departmento de Ciência de Computação
Universidade de Brasília
70910-900 Brasília, DF - Brasil
gadelha@cic.unb.br

Mauricio Ayala-Rincón
Departamento de Matemática
Universidade de Brasília
70910-900 Brasília, DF - Brasil
ayala@mat.unb.br

Abstract

We present an efficient strategy for the application of the inference rules of a completion procedure for finitely presented groups. This procedure has been proposed by Cremanns and Otto and uses a combinatorial structure called word-cycle. Our strategy is complete in the sense that a set of persistent word-cycles can be used to solve the reduced word problem of the finitely presented group given as input. This procedure has been implemented in order to experiment with various examples of groups.

Keywords: *String-Rewriting Systems, Knuth-Bendix Completion, Word Problem in Finitely Presented Groups*

1 Introduction

The analysis of combinatorial properties of algebraic structures such as monoids and groups has been of fundamental relevance in the development of the theory of computation. Two examples that illustrate this fact are

- the nice relations between formal language theory, the theory of automata, the word problem for groups and string-rewriting systems and
- the classical connections between decidability of problems over algebraic structures and of fundamental computational problems.

More specifically,

- the first kind of relations could be illustrated with well-known results such as the regularity of the language given by the word problem of finitely presented groups that are finite and the polynomial construction of finite-state acceptors for the set of reducible and irreducible strings using a finite string-rewriting system [6, 9].
- A relevant illustration of the second kind of connections is the well-known proof of the undecidability of the Post Correspondence Problem, that is based on the undecidability of the word problem in monoids as presented in many fundamental books on formal language, automata and computational theory (see for example [10]).

Knuth-Bendix completion has been used to compute, from a (string-)rewriting system R and a *reduction ordering* $>$, a rewriting system R_∞ that defines the same equational theory as R being compatible with the reduction ordering, and hence noetherian or terminating, and that is confluent [5]. Additionally, if the reduction ordering is total, then this procedure terminates successfully if and only if a finite system R_∞ with these properties exists. For a group G presented by $\langle\Sigma; R\rangle$, where Σ is a finite alphabet and R a set of equational relations between words over these alphabet, the word problem is effectively reducible to the membership problem for the congruence class $[\epsilon]_R$ of the empty string, ϵ. Even for small systems R that result in small systems R_∞ intermediate sets of rules can become arbitrarily large, and in particular in the setting of group completion, each intermediate rule induces several other rules based on the group axioms. This makes space efficient methods for group completion extremely relevant. In the setting of presentations of groups from each rule its cyclic permutations, its inverse and the cyclic permutations thereof are induced by this *symmetrization* process. To save space this information can be stored in a *word-cycle*.

In [3] a completion procedure that uses word-cycles was introduced. This procedure is presented as a set of four inference rules that are applied to a set of word-cycles. When the procedure terminates for a given input R, it generates a finite set of word-cycles Z^∞ that describes the same congruence as R. This set of word-cycles can be converted into

*Supported by M.Sc. fellowships from CNPq and CAPES Brazilian Foundations. Current address: CertiSign Certificadora Digital, Rua Lauro Müller, 116 - Gr. 2705, 22290-160 Rio de Janeiro, RJ - Brasil. lgadelha@certisign.com.br.

a confluent string-rewriting system and consequently, if it is also terminating, it corresponds to a (*convergent* system or an effective) solution for the *word problem* of the group. Although this method represents an important improvement on space-complexity its practical time performance is very bad for simple groups as the dihedral and Fibonacci groups.

Here we present an efficient strategy for applying the inference rules of the Cremanns-Otto procedure, that is complete in the sense that the set of persistent word-cycles corresponds to a (possibly infinite) solution for the reduced word problem of the given group.

The results presented here are part of the first author master's thesis [4].

2 Background

We assume familiarity with basic notions of abstract algebra and of string rewriting theory as in [2, 1].

Let M be a set and $\rightarrow$ a binary relation over M. The pair $(M, \rightarrow)$ is called an **abstract rewriting system**. $\rightarrow$ is called an **abstract rewriting relation**. By $\leftarrow$, $\rightarrow^+$, $\rightarrow^*$, $\leftrightarrow$, and $\leftrightarrow^*$ we denote, respectively, the inverse of $\rightarrow$, its transitive closure, its reflexive and transitive closure its symmetric closure and its reflexive, symmetric and transitive closure. From the syntactic identity, denoted either as $=$ or $\rightarrow^0$, we can define inductively $\rightarrow^*$ as the union of the relations $\rightarrow^{n+1} = \rightarrow^n \circ \rightarrow$, for all $n \in \mathbb{N}$.

Given a rewriting system $(M, \rightarrow)$ we denote the equivalence class of $x \in M$ in the relation $\leftrightarrow^*$ by $[x]$, that is $[x] = \{y \in M \mid y \leftrightarrow^* x\}$. x is said to be *irreducible* if there is no $y \in M$ such that $x \rightarrow y$. Let $u, v \in M$ if $u \rightarrow^* v$ and v is irreducible then we say that v is a *normal form* for u. u and v are *joinable*, denoted as $u \downarrow v$, if there exists $w \in M$ such that $u \rightarrow^* w$ and $v \rightarrow^* w$.

$(M, \rightarrow)$ is *noetherian* if there are no infinite reduction sequences of the form $u_0 \rightarrow u_1 \rightarrow \cdots$. $(M, \rightarrow)$ is said to be *confluent* whenever for all $u, v \in M$, $u \leftrightarrow^* v$ implies $u \downarrow v$. A rewriting system is *convergent* if it is confluent and noetherian.

Well-known and important computational properties of the rewriting systems are related with the unicity of normal forms for confluent and convergent rewriting systems:

- if $(M, \rightarrow)$ is confluent and $u \in M$ has a normal form then this normal form is unique.
- if $(M, \rightarrow)$ is convergent then each $u \in M$ has a unique normal form.

From the computational point of view existence and unicity of normal forms corresponds to the uniqueness of outputs for non-deterministic algorithmic processes that are specified via convergent rewriting systems.

The study of rewriting theory is focused in two main classes of systems:

- term rewriting systems and
- string-rewriting systems.

The first class has successfully been used for algebraic specification and formal implementation of first and higher-order programming languages and proof assistants [1] and the second class, that is the one of interest in this paper, for applications related with computation in algebraic structures as for instance the treatment of the word problem [2].

A **string-rewriting system** R over a finite alphabet Σ is a subset of $\Sigma^* \times \Sigma^*$, where as usual Σ^* denotes the set of words over the alphabet Σ. The reduction relation $\rightarrow_R$ over Σ^* induced by R is defined as: for all $x, y \in \Sigma^*$, $x \rightarrow_R y$ if and only if there are two words $u, v \in \Sigma^*$ and a *rule* $(l, r) \in R$ such that $x = ulv$ and $y = urv$. Observe that if R is a string-rewriting system then $\langle \Sigma^*, \rightarrow_R \rangle$ is an abstract rewriting system as defined previously.

A major application of (string-)rewriting theory to combinatorial group theory is given by the use of the Knuth-Bendix procedure in the completion of group presentations. Knuth-Bendix is called a *completion* procedure in the sense that it takes as input a finite set of string-rewriting rules and an *ordering* over strings and if it halts then it generates as output a finite string-rewriting system that has the property of being convergent. The procedure iterates deducing from rules of the form $uv \rightarrow x$ and $vw \rightarrow y$ *critical pairs* of the form $\langle xw, uy \rangle$. Then the procedure tries to orient these critical pairs into new rules either of the form $xw \rightarrow uy$ or $uy \rightarrow xw$ depending on the ordering used. If in a stage of the computation all of these rules (or their normalized counterparts) are already present in the current rewriting system then the procedure stops. And in this case the current rewriting system (that is given as output) provides a straightforward solution to the word problem of the input group since a word will be congruent to the group identity, ϵ, if and only if its normal form is equal to ϵ. A detailed treatment of the use of the Knuth-Bendix procedure in the context of groups can be found in [8].

In the sequel we present the specific notation and algebraic concepts needed for our completion strategy.

Let $^{-1} : \Sigma \rightarrow \Sigma$ be a function such that for all $a \in \Sigma$ $(a^{-1})^{-1} = a$. The function $^{-1}$ is extended to Σ^* by defining $\epsilon^{-1} = \epsilon$, for the empty word, denoted as ϵ, and $(wa)^{-1} = a^{-1}w^{-1}$, for all $a \in \Sigma, w \in \Sigma^*$. Σ^+ denotes $\Sigma \setminus \{\epsilon\}$. For $u, v \in \Sigma^*$, $\subset (u, v)$ and $\supset (u, v)$ denote, respectively, the longest common prefix and suffix of u and v. Let $\sim$ be a congruence over Σ^*, such that $aa^{-1} = \epsilon$, for all $a \in \Sigma$. Thus the class of equivalences of $[\epsilon]_\sim$ is closed under inversion and cyclic permutations. Let $\approx$ be the smallest equivalence relation over Σ^*, such that $u \approx u^{-1}$ and $uv \approx vu$, for all $u, v \in \Sigma^*$. The equivalence classes of $\approx$, denoted as $[u]_\approx$, are called *word-cycles* and $[\epsilon]_\approx$ is called the *empty word-cycle*. A word-cycle $[w]_\approx$ is said to be *freely reduced* if all $z \in [w]_\approx$ is freely reduced, that means that z does not contain sub-words of the form uu^{-1}. For a set Z of word-cycles, we define the congruence $\sim_Z$, as the one generated by

$$L(Z) = \{x \in \Sigma^* \mid [w]_\approx \in Z\}$$

that is, the congruence generated by the string-rewriting system

$$R_{L(Z)} = \begin{cases} \{(w \to \epsilon) \mid w \in \Sigma^*, [w]_\approx \in Z\} \\ \bigcup \\ \{(aa^{-1} \to \epsilon) \mid a \in \Sigma\} \end{cases}$$

Let $\sim$ be a congruence over Σ^* such that $aa^{-1} \sim \epsilon$ for all $a \in \Sigma$. The *word problem*, $\mathrm{WP}_\sim$, is the set of all word-cycles $[w]_\approx$ such that $w \sim \epsilon$. The *reduced word problem*, $\mathrm{RWP}_\sim$, is the set of all word-cycles $[w]_\approx$ such that w is freely reduced and $w \sim \epsilon$ and there exists no sub-word of w, w', such that $w' \sim \epsilon$.

Cremanns-Otto completion procedure is based on the system $\mathcal{P}$ of four inference rules presented in the Table 1 that transform a set of word-cycles, Z.

$$\mathcal{P}1.\ \frac{[\epsilon]_\approx \in Z}{Z \setminus \{[\epsilon]_\approx\}}$$

$$\mathcal{P}2.\ \frac{[uaa^{-1}]_\approx \in Z}{Z \setminus \{[uaa^{-1}]_\approx\} \cup \{[u]_\approx\}}, u \in \Sigma^*, a \in \Sigma$$

$$\mathcal{P}3.\ \frac{[u]_\approx, [uv]_\approx \in Z}{Z \setminus \{[uv]_\approx\} \cup \{[v]_\approx\}}, u, v \in \Sigma^+$$

$$\mathcal{P}4.\ \frac{[ux]_\approx, [uy]_\approx \in Z}{Z \cup \{[xy^{-1}]_\approx\}}, u, x, y \in \Sigma^+, \subset(x, y) = \epsilon \text{ and } \supset(x, y) = \epsilon$$

Table 1. Set $\mathcal{P}$ of inference rules

$Z \vdash_{\mathcal{P}} Z'$ denotes that the set of word-cycles Z' is obtained from Z by application of one of the previous inference rules. A $\mathcal{P}$-derivation is a finite or infinite sequence $(Z_i)_{i \in I}$ of sets of word-cycles, where $I = \mathbb{N}$ or $I = \{k \mid k \in \mathbb{N}, k \le n \in \mathbb{N}\}$, such that for all $i \in I \setminus \{0\}$, $Z_{i-1} \vdash_{\mathcal{P}} Z_i$. A word-cycle is called persistent if, for some $i \in I$, it belongs to any Z_j with $j \in I$ and $j \ge i$. For a $\mathcal{P}$-derivation $(Z_i)_{i \in I}$, we define $Z = \cup_{i \in I} Z_i$ and Z^∞ as the set of all persistent word-cycles.

A set of word-cycles is said to be *interreduced* if rules $\mathcal{P}1, \mathcal{P}2, \mathcal{P}3$ don't apply on it.

The notion of *fairness* is defined as usual. We say that a $\mathcal{P}$-derivation $(Z_i)_{i \in I}$ is *fair* if Z^∞ is interreduced and for all $u, x, y \in \Sigma^+$, if $[ux]_\approx, [uy]_\approx \in Z^\infty$ and $\epsilon = \subset(x, y) = \supset(x, y)$, then $[xy^{-1}]_\approx \in Z^\infty$.

A *completion procedure* has as input a finite set of word-cycles Z_0 and generates a fair $\mathcal{P}$-derivation $(Z_i)_{i \in I}$ such that, whenever Z^∞ is finite, the $\mathcal{P}$-derivation generated is finite as well, and the procedure terminates and returns the set Z^∞ as output.

```
INPUT: Z_0, a finite set of word-cycles
OUTPUT: A
A := Z_0;
Interreduce(A);
REPEAT
        B := A;
        C := the set of all word-cycles that are
               obtained from word-cycles in A
               by an application of rule P4;
        A := A ∪ C;
        Interreduce(A);
UNTIL A = B;
```

Table 2. Cremanns-Otto word-cycle completion procedure

By $\mathcal{I}nterreduce$ we denote a procedure that given as input a finite set of word-cycles applies rules $\mathcal{P}1, \mathcal{P}2, \mathcal{P}3$ as long as possible. Observe that the procedure $\mathcal{I}nterreduce$ always terminates.

Cremanns-Otto completion procedure is given in the Table 2 and consists essentially of an initial application of $\mathcal{I}nterreduce$ and then alternative applications of rule $\mathcal{P}4$ and $\mathcal{I}nterreduce$ until the current set of word-cycles no longer changes.

Cremanns and Otto proved that their procedure is in fact a *completion procedure* and that for every fair $\mathcal{P}$-derivation, $\mathrm{RWP}_\sim = Z^\infty$. The second result is not straightforward and is proved using properties of group diagrams [3].

The previous result has the very interesting consequence that the Cremanns-Otto procedure, and in general every *completion procedure*, will always terminate when the reduced word problem of a group is finite, because in this case Z^∞, being equal to $\mathrm{RWP}_\sim$, will also be finite. And, since the Cremanns-Otto procedure is indeed a *completion procedure*, the $\mathcal{P}$-derivation will be finite and will stop with Z^∞ as output.

3 An efficient completion strategy

The beautiful Knuth-Bendix style of the Cremanns-Otto novel procedure and its completeness proof are doubtless very relevant results, but unfortunately their completion procedure doesn't work as one could expect in practice. In fact Cremanns-Otto procedure was neither implemented nor analyzed at all before, we do it [7] and for our early prototype implementations of their method the results were extremely bad in comparison with the direct application of Knuth-Bendix completion.

For instance, consider the very simple example of the group of symmetries of the square, presented by the set of equations:

$$\begin{array}{ll} aA = \epsilon & Aa = \epsilon \\ bB = \epsilon & Bb = \epsilon \\ a^4 = \epsilon & b^2 = \epsilon \\ Baba = \epsilon & \end{array}$$

where for simplicity A and B denote the inverses of a and b (a^{-1} and b^{-1}, respectively).

This example is completed by any Knuth-Bendix completion system in seconds, while by direct application of the Cremanns-Otto procedure it takes, starting with the set of word-cycles

$$Z_0 = \{[a^4], [b^2], [Baba]\}$$

more than thirty minutes.

All our experiments were done on Linux based PCs with processors having less than 300 MHz. Our implementations were specified in C [4].

In the sequel we present a novel and efficient strategy for applying the four inference rules that makes Cremanns-Otto theoretical results about general *completion procedures* over word-cycles applicable and competitive for many important examples with respect to the direct application of the classical Knuth-Bendix completion over strings. Fairness of our strategy is proved and as conclusion we obtain that this is a *completion procedure* in the sense of the previous section.

Our strategy is based on four respective algorithms, denoted by $\mathcal{R}_1$ until $\mathcal{R}_4$, for the inference rules $\mathcal{P}1$ until $\mathcal{P}4$ that work on sets of word-cycles avoiding re-application of the inference rules on the same word-cycles and maintaining, as long as possible, the current set of word-cycles reduced with respect to the precedent inference rules (or respective algorithms).

Firstly, we define an algorithm called $\mathcal{S}$*implify*-$\mathcal{I}$*nverses* that given a word-cycle simplifies it by eliminating all cyclic consecutive inverses (aa^{-1}). As result we obtain a word-cycle freely reduced.

Afterward, we define four algorithms to control the application of our inference rules as in the Table 3.

- $\mathcal{R}1$ applies the rule $\mathcal{P}1$ for a whole set of word-cycles.
- $\mathcal{R}2$ applies as long as possible the rule $\mathcal{P}2$ for a set of word-cycles taking out all generated empty word-cycles.
- $\mathcal{R}3$ generates all possible word-cycles by applying the rule $\mathcal{P}3$ and immediately simplifies them according to the first two inference rules.
- $\mathcal{R}4$ generates all possible new word-cycles by applying the rule $\mathcal{P}4$ from a set of word-cycles simplifying them immediately in a directed way according to the first three inference rules.

A key of the efficiency of our strategy is that both $\mathcal{R}3$ and $\mathcal{R}4$ avoid re-application of the same inference rules on the same (pairs of) word-cycles.

By simple induction on the number of application of inference rules, three easy preservation lemmas for these algorithms can be proved [4]:

Lemma 1 The algorithms $\mathcal{R}2, \mathcal{R}3$ and $\mathcal{R}4$ preserve any set of word-cycles that is irreducible with respect to the rules $\mathcal{P}1$ and $\mathcal{P}2$ irreducible with respect to these rules.

Lemma 2 After an initial application of the algorithm $\mathcal{R}1$ and a subsequent application of the algorithm $\mathcal{R}2$, the exhaustive application of the algorithm $\mathcal{R}3$ interreduces any input set of word-cycles.

Lemma 3 For any interreduced input Z, the algorithm $\mathcal{R}4$ generates a super-set Z' of word-cycles of Z, that is irreducible for $\mathcal{P}4$ and such that for any subsequent possible application of the inference rule $\mathcal{P}3$ on pairs of word-cycles of the form $[uv]_\approx, [u]_\approx \in Z'$, that simplifies $[uv]_\approx$ as $[v]_\approx$, $[u]_\approx \in Z' \setminus Z$ holds.

Finally, our strategy is presented as a procedure that applies the previous four algorithms as given in the Table 4.

We prove that our completion strategy is fair and as consequence that it is in fact a *completion procedure*.

Theorem 1 ([4]) Our completion strategy is a *completion procedure*.

Proof (Sketch). Our completion strategy generates a $\mathcal{P}$-derivation, $(Z_i)_{i \in I}$, for an input Z. According to L2, before the main "REPEAT" Z' is interreduced. Thus between this "REPEAT", because of L1 and L3, after each application of $\mathcal{R}4$ Z' is irreducible with respect to the rules $\mathcal{P}1$ and $\mathcal{P}2$ and then, by L2, a further exhaustive application of $\mathcal{R}3$ gives an interreduced output. Then the set of persistent word-cycles, Z^∞, is interreduced. Suppose that $[ux]_\approx, [uy]_\approx \in Z^\infty$ for $u, x, y \in \Sigma^+$ with $\epsilon = \subset(x, y) = \supset(x, y)$. Then by application of $\mathcal{R}4$ $[xy^{-1}]_\approx$ will be generated and consequently the $\mathcal{P}$-derivation is fair.

Suppose that Z^∞ is finite. Then after a finite number of iterations of the main "REPEAT" this set will be contained in Z'. By contradiction suppose that there exists $[w]_\approx \in Z' \setminus Z^\infty$. Then there is a sub word w' of w such that $[w']_\approx \in$

ALGORITHM $\mathcal{R}1(Z)$
FOR all $[u]_\approx \in Z$ DO
  apply $\mathcal{P}1$ whenever possible;
RETURN Z.

ALGORITHM $\mathcal{R}2(Z)$
FOR all $[u]_\approx \in Z$ DO
  $u := \mathcal{S}implify\text{-}\mathcal{I}nverses(u)$;
  IF $u = \epsilon$ THEN $Z := Z \setminus \{[u]_\approx\}$;
RETURN Z.

ALGORITHM $\mathcal{R}3(Z)$
FOR all previously unconsidered different pairs
$[u]_\approx, [v]_\approx \in Z$
DO
  LET u_M and u_m be their representants,
    being u_M larger than u_m;
  WHILE ($[u_M]_\approx = [u_m x]_\approx$, for some $x \in \Sigma^+$)
  DO
    $u_M := \mathcal{S}implify\text{-}\mathcal{I}nverses(x)$;
    Replace the corresponding old
    word-cycle in Z with $[u_M]_\approx$;
    IF $u_M = \epsilon$ THEN $Z := Z \setminus \{[u_M]_\approx\}$
    LET u_M and u_m be the new
      representants, being u_M
      larger than u_m;
  END WHILE;
RETURN Z.

ALGORITHM $\mathcal{R}4(Z)$
$Z' := Z$;
FOR all previously unconsidered different pairs
$[u]_\approx, [v]_\approx \in Z$
DO
  FOR all $w, x, y \in \Sigma^+$ such that
    $\epsilon = \subset(x, y) = \supset(x, y)$ and
    $[u]_\approx = [wx]_\approx,\ [v]_\approx = [wy]_\approx$
  DO
   $t := xy^{-1}$;
   IF $[t]_\approx \notin Z'$ THEN
    FOR all $[s]_\approx \in Z'$ DO
     IF $|t| \geq s$ THEN
      WHILE $[t]_\approx = [xs]_\approx$, for some $x \in \Sigma^*$
      DO
        $t := \mathcal{S}implify\text{-}\mathcal{I}nverses(t)$;
   IF $t \neq \epsilon$ THEN $Z' := Z' \cup \{[t]_\approx\}$;
RETURN Z'.

Table 3. Algorithms for controlled application of the inference rules

INPUT: Z, a finite set of word-cycles
OUTPUT: Z
$Z' := \mathcal{R}1(Z)$;
$Z' := \mathcal{R}2(Z')$;
REPEAT
  $Z' := \mathcal{R}3(Z')$
UNTIL Z' does not change;
REPEAT
  $Z := Z'$;
  $Z' := \mathcal{R}4(Z')$;
  REPEAT
    $Z' := \mathcal{R}3(Z')$
  UNTIL Z' does not change;
UNTIL $Z' = Z$;

Table 4. Efficient completion strategy

Z^∞. That contradicts the fact that Z' is interreduced. Then $Z' = Z^\infty$. ∎

As consequence we have that the interesting fact mentioned at the end of the previous section applies: always when the reduced word problem of a group is finite our strategy will terminate given as output $Z^\infty =$RWP$_\sim$.

4 Conclusions

To our knowledge [7], our implementation of the Cremanns-Otto procedure for completing finitely presented groups was the first one given and our experimental work pointed out practical problems in the straightforward application of the inference rules introduced in [3]. In particular, we detected that uncontrolled application of the rules gives rise to an explosion on the running time making inefficient the procedure and therefore non competitive with the classical application of Knuth-Bendix completion over strings. This was checked for very simple but relevant examples as the ones of the dihedral groups, generated by $D_{2n} = \langle a, b \mid a^n, b^2, b^{-1}aba\rangle$ for which the input set of word-cycles corresponds to $Z_0 = \{[a^n]_\approx, [b^2]_\approx, [Baba]_\approx\}$ (D_8, for instance, is the group of symmetries of the square). Mainly, three inefficiency problems were detected and solved in our completion strategy:

- non-deterministic application of the inference rules $\mathcal{P}1, \mathcal{P}2$ and $\mathcal{P}3$ (that is, of the procedure $\mathcal{I}nterreduce$) gives rise to redundant application of these rules on the same word-cycles;

- uncontrolled application of the inference rules $\mathcal{P}3$ and $\mathcal{P}4$ gives rise to repetitive and redundant application of these rules on the same pairs of word-cycles and

- inference of new word-cycles by applying rule $\mathcal{P}4$ and further application of these inference rule on the non simplified new word-cycles propagates inefficiently generation of redundant non interreduced word-cycles.

Our completion strategy was proved to be fair and in fact a *completion procedure*. From the practical point of view, its implementation results competitive with the application of Knuth-Bendix completion over strings for very important examples as the dihedral groups. But for simple examples as the Fibonacci groups Knuth-Bendix completion over strings results too much better than completion over word-cycles. The Fibonacci groups are defined as $F(2,k) = \langle x_0, ..., x_{k-1} \mid x_{(i \bmod k)} x_{(i+1 \bmod k)} = x_{(i+2 \bmod k)},$ for $0 \leq i \leq k-1 \rangle$. Although for the very small Fibonacci groups as $F(2,3) = \langle a, b, c \mid ab = c,\ bc = a,\ ca = b \rangle$, the implementation runs for hours before stopping giving a correct answer.

As further work, additional improvements could be incorporated to our strategy. For example, since word-cycles are combinatorial data structures, matching and unification could be specialized. Well-known linear string pattern matching algorithms can be adapted for detecting all possible applications of the inference rules $\mathcal{P}3$ and $\mathcal{P}4$ on a pair of word-cycles in linear time. This improvement was not crucial for our experiments since, for our input groups, the size of the persistent word-cycles was relatively small.

Additionally, to validate this kind of completion procedures, the determination of particular groups for which the computational space is critical and consequently Cremanns-Otto style of Knuth-Bendix completion over word-cycles is better than Knuth-Bendix completion over strings is an essential work to be done.

Acknowledgments

We would like to thank Friedrich Otto for his gently and detailed answers to all our questions about the Cremanns-Otto completion method.

References

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

[2] R. Book and F. Otto. *String-Rewriting Systems*. Springer-Verlag, 1993.

[3] R. Cremanns and F. Otto. A completion procedure for finitely presented groups that is based on word cycles. Technical report, Universität Gesamthochschule Kassel, 1998.

[4] L. M. R. Gadelha Jr. Aplicações de Técnicas de Reescrita ao Problema da Palavra em Teoria dos Grupos. Master's thesis, Departamento de Ciência da Computação, Universidade de Brasília, December 2000. In Portuguese.

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

[6] F. Otto. On the connections between rewriting and formal language theory. In *10th International Conference on Rewriting Techniques and Applications*, volume 1631 of *Lecture Notes in Computer Science*, pages 332–355, Trento, Italy, 1999. Springer-Verlag.

[7] F. Otto. Personal communication. 2000.

[8] C. Sims. *Computation with Finitely Presented Groups*, volume 48 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994.

[9] I. A. Stewart and R. M. Thomas. Formal Languages and the Word Problem for Groups. In *Groups St Andrews 1997 in Bath, II*, volume 2 of *London Mathematical Society Lecture Note Series*, pages 689–700. Cambridge University Press, 1999.

[10] T. Sudkamp. *Languages and Machines: An Introduction to the Theory of Computer Science*. Addison-Wesley, 1997.