Constructing a Group with Linear Word Problem and Unsolvable Conjugacy Problem

Patrick D. Bangert, Andreas Kolling*

September 6, 2004

Abstract

We give a general method of constructing a group with unsolvable conjugacy problem and solvable word problem together with an algorithm to solve the word problem in linear-time. The starting point for the construction is a recursive function $\phi(x, y) \in \mathbb{N}$ with $x, y \in \mathbb{N}$ such that the predicate $A(x) \equiv (\exists y)(\phi(x, y) = 0)$ is recursively undecidable. By definition, many such groups may be constructed and find applications principally in cryptography.

Note to the editor and referee: The method is general, rather lengthy and complex. A real calculation suffers from two ails: A suitable predicate must be selected and a lengthy calculation engaged in. We believe that it is more constructive to separate this large project into two papers. The first giving the general method and the second giving some results. We are still working on the second paper and believe that feedback from the first could significantly aid the development of the second — finding good predicates is a search over all of mathematics and we would like to alert a large readership to this possibility. Hence we submit the paper in its current form at this time.

1 Introduction

Combinatorial methods in group theory are very powerful in mathematics and its applications [12]. A recent and very prominent example is the creation of a public-key cryptosystem based upon solving the word and conjugacy problems in a certain group [1]. The two communicating parties must be able to solve a word problem in order to decode each others' messages. A potential intruder must however solve a conjugacy problem. The details of the protocol are not important here (for background on the protocol, please refer to [1]); the message is that we need a group in which solving the word problem is easy and the conjugacy problem is hard.

It was thought that the braid groups were a suitable candidate, however the conjugacy problem is solvable in polynomial-time [3, 4, 5, 6]. It stands to reason to look for a group in which the word problem is solvable in linear-time (this is the minimum possible as the cryptographic key needs to be at least input) and the conjugacy problem fundamentally unsolvable (that would also protect it against possible proofs that $NP = P$). However, it seems that no example of such a group is known. We provide a framework in which an example may be obtained. The framework is much more general in that it allows straightforward calculation (possibly by a computer) of a group with any given Turing degree of the word and conjugacy problems independently from each other.

It is clear that unsolvability of the conjugacy problem as a whole does not mean that it is not solvable (and perhaps *easily* solvable) in a certain restricted subclass of words in the group at hand. Before such a group could thus be used as a platform group for cryptography, further analysis into the status of the conjugacy problem for the elusive average case would have to be conducted. This information must be taken under advisement in constructing possible secret keys. It would seem however, that such a group has better chances at being a practical group for cryptography than others.

The rest of the paper constitutes the proof of our fundamental result (with exception of the final section that outlines some generalizations):

*School of Engineering and Science, International University Bremen, P.O. Box 750 561, 28725 Bremen, Germany, p.bangert@iu-bremen.de, a.kolling@iu-bremen.de

Theorem 1 *Given a recursive function $\phi(x,y) \in \mathbb{N}$ with $x,y \in \mathbb{N}$ and a predicate $A(x) \equiv (\exists y)(\phi(x,y)=0)$ such that $A(x)$ is recursively undecidable, we construct a finitely presented group G with unsolvable conjugacy problem and linear-time word problem.*

The method of construction is explicitly spelled out and completely mechanizable.

We begin with showing how to construct a single register machine with unsolvable confluence problem from $A(x)$. It is clear that many such predicates exist and it is this element that makes the theory general. We proceed to show how to translate this machine into a rewrite system with unsolvable word problem. This is subsequently transformed into a group with unsolvable conjugacy problem. Then, we give a linear-time algorithm to solve the word problem in all groups constructed in this way. Finally, we point out some possible generalizations to this work.

2 Single-Register Machine with Unsolvable Confluence Problem

The first technical step will be to create a certain register machine. We assume familiarity with these machines here and give a colloquial definition; for details on these machines please see [13, 14].

Definition 2 (Register Machine) *A* register machine *M is a quadruple $M=(C,S,I,R)$ where S is a fixed and finite set of states, $C \in S$ is the current state, I is a set of instructions for operating upon the states and R is a set of registers (natural numbers).*

Various degrees of difficulty are usually ranked using the concept of Turing degree.

Definition 3 (Turing degree) *Two problems are of the same* Turing degree *if each problem is solvable using an oracle of the other.*

Three immediate questions may be asked: (1) Given two machines $M=(C,S,I,R)$ and $M'=(C',S,I,R)$ the *confluence problem* asks whether the instruction set I is such that $M=M'$ after a finite number of state transitions of either or both machines, (2) the *word problem* asks whether $M=M'$ after a finite number of state transitions of M only, (3) the *halting problem* asks whether there exists a natural number m such that the number of state transitions until the machine stops for all possible starting states is bounded from above by m.

Let us define a machine M by the following two-line program on states $S=\{1,2\}$ that give the line number, four natural number registers $R=(r_1,r_2,r_3,r_4)$ and an arbitrary recursive function $\phi(r_1,r_2)$:

1. (a) $r_2 \leftarrow \mu$ (b) if $\phi(r_1,r_2) \neq 0$, $r_3 \leftarrow r_3+1$ (c) go to step 1.
2. (a) $r_4 \leftarrow 0$ (b) go to step 2.

where

$$\mu = \begin{cases} \min\{r_2 : r_2 \leq r_3\} & \text{if } \{r_2 : r_2 \leq r_3\} \neq \emptyset \\ r_3 & \text{if } \{r_2 : r_2 \leq r_3\} = \emptyset \end{cases} \tag{1}$$

and arbitrary finite initial values. Upon inspection, it is clear that this machine never halts and that we can solve the confluence or word problems if and only if we decide the predicate $A(x) \equiv (\exists y)(\phi(x,y)=0)$.

To be very precise about our machine, we desire to write it in such a way that it can be computed on a word written explicitly in the binary language. Suppose that the multiple register machine has the registers $(x_1,x_2,\cdots,x_n)$. We construct the binary sequence $1^{x_1}01^{x_2}0\cdots01^{x_n}$ from these registers where the powers indicate repetition. This allows us to convert a multiple register machine into a single register machine (SRM).

The SRM operates upon a word w written in the letters 1 and 0; this word is the content of the single register of the machine. We take N as the length of w in letters. We define our language in the usual way by providing a small set of primitive operations and then defining subroutines for the more complex tasks. The primitive operations are (the subscript indicates the current length of the word; it will be needed later):

1. $\texttt{Print}_N(0)$: Prints 0 at the end of w.
2. $\texttt{Print}_N(1)$: Prints 1 at the end of w.
3. $\texttt{ScanDelete}_N(k,l)$: Scan and delete the first letter of w. If it is 0 (or w is empty) go to line k of the program, otherwise go to line l.

Now, we reduce the confluence problem to the solution of the predicate.

Lemma 4 *Any two configurations C_1 and C_2 are confluent if and only if $A(x)$ can be solved.*

Proof. If $A(x)$ is solvable, it is obvious whether C_1 and C_2 are confluent. The fact that if we can decide confluence, then we may compute $A(x)$ is demonstrated by analyzing the four possible cases:

1. Let $C_1 = (2, x, y, z, u)$ and $C_2 = (2, x, y, z, u')$ for any $x, y, z, u \neq u'$, then C_1 and C_2 are confluent trivially.
2. Let $C_1 = (1, x, y, z, u)$ and $C_2 = (1, x, y', z', u')$ for any $x, y \neq y'$, $z \neq z'$ and $u \neq u'$, then C_1 and C_2 are confluent if and only if $A(x)$ is true.
3. Let $C_1 = (1, x, y, z, u)$ and $C_2 = (1, x, y', z', u)$ for any $x, y \neq y'$, $z \neq z'$ and u, then C_1 and C_2 are confluent trivially.
4. Let $C_1 = (1, x, y, z, u)$ and $C_2 = (2, x, y', z', u')$ for any $x, y \neq y'$, $z \neq z'$ and $u \neq u'$, then C_1 and C_2 are confluent if and only if $A(x)$ is true and $C_1 = (2, x, y', z', 0)$. □

Thus writing M in the above language results in an SRM with unsolvable confluence problem.

3 Rewrite Systems with Unsolvable Word Problem

We assume that the reader is familiar with the basics of rewriting systems that may be found in [2]. Given a standard one-way rewrite system R, we may take the reflexive-transitive-symmetric closure of its rules to obtain the associated two-way rewrite system $R^\dagger$; note that $R^\dagger$ is a monoid.

Given the SRM M constructed above, we write our rewrite system $R(M)$ in the alphabet $\{\lhd, \rhd, 0, 1, \overline{0}, \overline{1}, q_i, R^{(j)}_{k,i}\}$ with $i = 1, 2, \cdots, s$; $k = 1, 2, \cdots, s-1$ and $j = 0, 1, \cdots, N_i$. The symbols $\rhd$ and $\lhd$ will indicate the start and end of the word to be rewritten. The letters 0 and 1 are the binary symbols of the single-register of M whereas the letters $\overline{0}$ and $\overline{1}$ are the operated-upon versions of the binary letters. The tokens q_i indicate operations akin to the lines of the SRM program and the $R^{(j)}_{k,i}$ symbolize the results of these operations.

The system $R(M)$ is actually constructed by adding some relations for every primitive instruction in the SRM program:

1. $\texttt{Print}_{N_i}(0) \Rightarrow \{\rhd q_i \rightarrow \rhd R^{(1)}_{i,i+1};\ \ R^{(N_i)}_{i,i+1} \lhd \rightarrow q_{i+1}0 \lhd\}$
2. $\texttt{Print}_{N_i}(1) \Rightarrow \{\rhd q_i \rightarrow \rhd R^{(1)}_{i,i+1};\ \ R^{(N_i)}_{i,i+1} \lhd \rightarrow R^{(0)}_{1,i+1}1 \lhd\}$
3. $\texttt{ScanDelete}_{N_i}(k,l) \Rightarrow \{\rhd q_i 0 \rightarrow \rhd R^{(2)}_{i,k};\ \ \rhd q_i \lhd \rightarrow \rhd R^{(1)}_{i,k} \lhd;\ \ \rhd q_i 1 \rightarrow \rhd R^{(1)}_{i,l};\ \ R^{(N_i)}_{i,k} \lhd \rightarrow R^{(0)}_{1,k} \lhd;\ \ R^{(N_i)}_{i,l} \lhd \rightarrow R^{(0)}_{1,l} \lhd\}$

In addition to these instruction-specific relations, we also add $\overline{0}R^{(0)}_{1,i} \rightarrow R^{(0)}_{1,i}0$; $\overline{1}R^{(0)}_{1,i} \rightarrow R^{(0)}_{1,i}1$ and $\rhd R^{(0)}_{1,i} \rightarrow \rhd q_i$ for every q_i and the relations $R^{(j)}_{k,i}0 \rightarrow \overline{0}R^{(j+1)}_{k,i}$ and $R^{(j)}_{k,i}1 \rightarrow \overline{1}R^{(j)}_{k,i}$ for every $R^{(j)}_{k,i}$ with $j > 0$.

We would like a rewrite system that has the same Turing degree for its halting, word and confluence problems as M. Currently, $R(M)$ has more degrees of freedom than M and thus must be restricted by reducing the number of configurations such that the configuration set remains closed. This restriction process preserves the Turing degree of all three problems of the system [15]. Every word in $R(M)$ either contains a q_i or a $R^{(j)}_{k,i}$ (q-words) or does not (q-free words). As

all rules of $R(M)$ contain q-words, we may safely ignore all q-free words. Let us call any word terminal if no relation of $R(M)$ may be applied to it; these words may also be ignored for the three decision problems. By restricting $R(M)$ to non-terminal q-words only, we achieve a rewrite system whose halting, word and confluence problems are of the same Turing degree as the SRM M (see [15] for a proof).

By the construction method, we know that two words w and v in the letters of $R(M)$ are equal in $R(M)^{\dagger}$ if and only if they are confluent in the restricted $R(M)$ [15]. Thus the restricted $R(M)$ is a rewrite system with unsolvable word problem.

4 Group with Unsolvable Conjugacy Problem

Suppose that $R(M)$ has n_g generators and n_r relations, then we may simplify the presentation by rewriting it in the form $R(M) = \left\langle \left\{ s_1, s_2, \cdots, s_{n_g}, q \right\} : F_j q \rightarrow q K_j \right\rangle$ with $1 \leq j \leq n_r$, where the generators s_i denote all the symbols in $R(M)$, where F_j and K_j denote the left and right sides of the rules obtained before and q is added to this presentation for technical reasons only (to become apparent in the next step). Note that the letters making up F_j and K_j are now only s_i's and they are thus q-free words (with q being the new technical generator). We then make use of the following theorem.

Theorem 5 (Collins [10]) *Let $R(M)$ be a rewrite system of the above form and G a group with presentation*

$$G = \left\langle \begin{array}{l} \left\{ s_1, s_2, \cdots, s_{n_g}, r_1, r_2, \cdots, r_{n_r}, x, q, k, t \right\} : \quad x s_i = s_i x x; \quad r_i s_j = s_j x r_i x; \\ r_i \overline{F}_i q = q K_i r_i; \quad tx = xt; tr_i = r_i t; \quad kx = xk; \quad kr_i = r_i k \end{array} \right\rangle \tag{2}$$

where the $\overline{F}_i$ are obtained from the F_i by replacing all s_i by their inverses (note that the F_i initially do not contain any inverses). Then G has solvable word problem and the conjugacy problem in G is Turing equivalent to the word problem for q-words in $R(M)$.

Finally, we are in possession of a group with solvable word problem and unsolvable conjugacy problem constructed from a suitable logical predicate. It now remains to demonstrate that the word problem can be solved in linear-time in the letter length of the word.

5 Algorithm to Solve Word Problem in Linear-Time

A number of technicalities are necessary to derive a word problem algorithm for G. We will need to make use of four auxiliary groups that are obtained from G by successively deleting generators and relations,

$$G_1 = \left\langle \left\{ s_1, \cdots, s_{n_g}, x, r_1, \cdots, r_{n_r}, q \right\} : \quad x s_i = s_i x x; \quad r_i s_j = s_j x r_i x; \quad r_i \overline{F}_i q = q K_i r_i \right\rangle \tag{3}$$

$$G_2 = \left\langle \left\{ s_1, \cdots, s_{n_g}, x, r_1, \cdots, r_{n_r} \right\} : \quad x s_i = s_i x x; \quad r_i s_j = s_j x r_i x \right\rangle \tag{4}$$

$$G_3 = \left\langle \left\{ s_1, \cdots, s_{n_g}, x \right\} : \quad x s_i = s_i x x \right\rangle \tag{5}$$

$$G_4 = \langle \{x\} : \emptyset \rangle \tag{6}$$

Furthermore, we need two technical definitions and lemmas.

Definition 6 (Stable letters, basis, Britton condition) *Let $E = \langle S : D \rangle$ be any group presentation with alphabet S and relation set D. Let $E^{\star} = E \bigcup \langle \{p_v\} : F_i p_{v_i} G_i = H_i p_{v_i} K_i \rangle$ with $i \in I$ and $v, v_i \in V$ for I and V two index sets and F_i, G_i, H_i and K_i words over S such that at least one of the relations involves p_v for every $v \in V$. Put $A_i = H_i^{-1} F_i$ and $B_i = K_i G_i^{-1}$ and denote all words on the A_i or B_i for a particular v by $A(v)$ or $B(v)$ respectively. Under these conditions, we will say that $E^{\star}$ is a presentation with* stable letters p_v *and* basis E*. Furthermore, if $A(v)$ and $B(v)$ are isomorphic for every $v \in V$ we say that the* Britton condition *holds and denote this by $\mathcal{B}(E^{\star}, E, p_v, V)$.*

It may be shown that the following four Britton conditions hold [10]: $\mathcal{B}(G_3, G_4, s_i, \{1, \cdots, n_g\})$, $\mathcal{B}(G_2, G_3, r_i, \{1, \cdots, n_r\})$, $\mathcal{B}(G_1, G_2, q, \emptyset)$, $\mathcal{B}(G, G_1, \{t, k\}, \emptyset)$.

Definition 7 (p-reduction) *Suppose that* $\mathcal{B}(E^\star, E, p_v, V)$ *and* $W \in E^\star$. *If* W *contains a subword of the form* $p_v^{-1}Cp_v$ *where* $C \in A(v)$ *is p-free (where* $A = A_{i_1}^{e_1}A_{i_2}^{e_2}\cdots A_{i_r}^{e_r}$ *with* $e_j = \pm 1$ *and* $i_j \in I$*), then the word obtained by replacing* $p_v^{-1}Cp_v$ *with* $B_{i_1}^{e_1}B_{i_2}^{e_2}\cdots B_{i_r}^{e_r}$ *is a* primitive p-reduction. *Another primitive p-reduction is defined analogously if* W *is of the form* $p_vCp_v^{-1}$ *where* $C \in B(v)$. *A word* W *is called* p-reducible *if at least one primitive p-reduction may be applied to* W *and* p-reduced *if no primitive p-reductions can be applied to* W*; the p-reduced form of* W *is denoted by* $p[W]$.

A somewhat simpler condition for p-reducibility is expressed in the following lemma.

Lemma 8 (Britton [9]) *If* $\mathcal{B}(E^\star, E, p_v, V)$, $W =_{E^\star} 1$ *and* W *is not p-free, then* W *is p-reducible.*

It is possible that a word may be p-reduced in several different ways at the same time. The fact that all these reduction paths yield the same answer (i.e. that the p-reduction process is confluent) is shown by the last lemma.

Lemma 9 (Boone and Collins [7, 8, 10]) *If* $\mathcal{B}(E^\star, E, p_v, V)$ *and* $W =_{E^\star} U$ *for two words* $W, U \in E^\star$, *then* $p[W]$ *and* $p[U]$ *have identical p's in identical positions with the number of letter in between p's also the same. Thus the two final forms are identical with respect to p-reduction and so all possible p-reduction paths are confluent.*

This enables us to write down an algorithm to solve the word problem in G:

Data : $W \in G$.
Result : True if $W =_G 1$ and false otherwise.
$W \leftarrow t[k[W]]$
if W *is not t-free and k-free* **then** return false
$W \leftarrow q[W]$
if W *is not q-free* **then** return false
$W \leftarrow r[W]$
if W *is not r-free* **then** return false
$W \leftarrow s[W]$
if W *is not s-free* **then** return false
if $W \neq_{G_4} 1$ **then** return false
return true

Algorithm 1: Word Problem in G.

The assignment statements are computable because of the Britton conditions and lemma 8 and are simply p-reductions that can be made in linear-time (matching the pattern is linear and there are at most a linear number of reductions possible as each reduces the number of letters in W by two). The various statements of the form "if W is not x-free" must be answered in the negative because W would then not be an element in the next higher group in our hierarchy. We cannot combine any of the reductions and checks into single lines in this algorithm as we must make sure that the reductions can in fact be carried out, i.e. the words must be elements of the required groups. We have thus reduced the word problem in G to that in the free group G_4 by successive p-reductions and freedom checks. This final word problem is trivial of course and thus gives us a linear-time algorithm for the word problem in G.

6 Generalizations

The object of the paper thus far was the construction of a group such that the word and conjugacy problems had specific Turing degrees. This approach can be generalized to constructing a group with an arbitrary Turing degree for these problems. The construction is similar to the one above and the connecting results are the same. It thus suffices to state the two theorems that allow the generalization to be made. The details of the construction may be found together with the proofs of these theorems.

Theorem 10 (Shepherdson [15]) *If $\{D_i\}$ is a recursively enumerable (r.e.) set of r.e. Turing degrees and D is any r.e. Turing degree such that $D \geq D_i$ for all i, then there exists a recursive construction of a rewrite system T such that: (1) For every D_i, there exists a word of T such that its individual word problem has degree D_i. (2) The word problem in T has degree D.*

Theorem 11 (Collins and Miller [11]) *Under the same conditions as theorem 10 we may construct a group G such that: (1) G has solvable word problem. (2) For every D_i, there exists a word in G such that its individual conjugacy problem has degree D_i. (3) The conjugacy problem in G has degree D.*

References

[1] Anshel, M., Goldfeld, D., Anshel, I. (1999): An Algebraic Method for Public-Key Cryptography. Math. Res. Lett. **6**, 1–5.

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

[3] Bangert, P.D., Berger, M.A. and Prandi, R. (2002): In Search of Minimal Random Braid Configurations. J. Phys. A: Math. Gen. **35**, 43–59.

[4] Bangert, P.D. (2004): Raid Braid: Fast Conjugacy Disassembly in Braid and Other Groups. Proceedings of Applications of Computer Algebra 2004.

[5] Bangert, P.D.(2003): The Word and Conjugacy Problems in B_n. To appear.

[6] Bangert, P.D. (2004): The Conjugacy Problem in B_n. in Anshel, I., Ashel, M., Goldfeld, D. (2003): Contributions to Contemporary Cryptography. World Scientific Pub. Co., Singapore.

[7] Boone, W.W. (1966): Word problems and recursively enumerable degrees of unsolvability. A first paper on Thue systems. Ann. Math. **83**, 520–571.

[8] Boone, W.W. (1966): Word problems and recursively enumerable degrees of unsolvability. A sequel on finitely presented groups. Ann. Math. **84**, 49–84.

[9] Britton, J.L. (1963): The word problem. Ann. Math. **77**, 16–32.

[10] Collins, D.J. (1969): Recursively enumerable degrees and the conjugacy problem. Acta Math. **122**, 115–160.

[11] Collins, D.J., Miller, C.F. III (1977): The Conjugacy Problem and Subgroups of Finite Index. Proc. London Math. Soc. **34**, 535–556.

[12] Lyndon, R.C., Schupp, P.E. (1977): Combinatorial group theory. (Springer).

[13] Minsky, M.L. (1961): Recursive Unsolvability of Post's Problem of 'Tag' and Other Topics in Theory of Turing Machines. Ann. Math. 74, 437–455.

[14] Shepherdson, J.C. and Sturgis, H.E. (1963): Computability of Recursive Functions. J. Assoc. Comput. Mach. 10, 217–255.

[15] Shepherdson, J.C. (1965): Machine Configuration and Word Problems of given Degree of Unsolvability. Z. Math. Logik Grundlagen Math., 149–175.