TRANSACTIONS OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 276, Number 2, April 1983

EFFICIENT COMPUTATION IN GROUPS AND SIMPLICIAL COMPLEXES

BY

JOHN C. STILLWELL

ABSTRACT. Using HNN extensions of the Boone-Britton group, a group E is obtained which simulates Turing machine computation in linear space and cubic time. Space in E is measured by the length of words, and time by the number of substitutions of defining relators and conjugations by generators required to convert one word to another. The space bound is used to derive a PSPACE-complete problem for a topological model of computation previously used to characterize NP-completeness and RE-completeness.

Introduction. The ability of mathematical systems to simulate computation has often been used to prove unsolvability results. The first, and most instructive, example was Post's simulation of Turing machines by finitely presented semigroups [**10**]. For each deterministic Turing machine M, Post constructs a semigroup $T(M)$ on generators we shall call q_a, s_b, where a and b range over certain finite sets. An *instantaneous description* (ID for short) of the state of computation at any moment can be written as a word Σ in these generators, there is a special q_a generator called q, and the defining relations of $T(M)$ are such that $\Sigma = q$ is derivable if and only if the ID Σ leads M to halt. Thus the halting problem for M is reduced to the word problem for $T(M)$, and by choosing an M with unsolvable halting problem Post proved the unsolvability of the word problem for semigroups.

The simulation of M by $T(M)$ shows that "semigroups can compute". What is interesting from the viewpoint of computational complexity is that the derivation in $T(M)$ which reflects a given computation of M has length bounded by a linear function of the length of computation. Thus "time" in $T(M)$ (measured by the number of steps in a derivation) is slower than M's time by no more than a linear function, so the simulation is *efficient* with respect to time (and, a fortiori, with respect to space) and, in particular, the semigroups $T(M)$ can do polynomial time computation.

We now briefly review Post's argument, because with a little elaboration of it one can also obtain nondeterministic polynomial time (NP) computation in semigroups.

An ID Σ of M consists of the sequence of symbols s_b on M's tape at the given moment, bounded to left and right by end markers s_0, and with a symbol q_a,

Received by the editors August 15, 1980 and, in revised form, March 24, 1982.

1980 *Mathematics Subject Classification.* Primary 20F10, 68C25; Secondary 57M20.

Key words and phrases. Word problem for groups, Turing machines, two-dimensional complexes, polynomial time computation.

0002-9947/82/0000-1254/$04.00

representing M's current internal state, inserted to the left of the symbol representing the current scanned square. A *step* of computation changes the ID in a region 2 or 3 symbols to left or right of the q_a symbol, yielding an *M-successor*, Σ', of Σ. Σ' can be obtained by substituting a word Γ_i, containing a symbol $q_{a'}$ for the state which follows q_a, for a suitable subword Σ_i of Σ containing q_a. Thus a *computation* is a sequence $(\Sigma, \Sigma', \ldots)$ of ID's, each of which results from its predecessor by one of the above transformations $\Sigma_i \to \Gamma_i$. Additional transformations, which we also denote by $\Sigma_i \to \Gamma_i$, are introduced for the purpose of simplifying any ID which occurs at the end of a halting computation. They introduce the special symbol q which then "eats up" all other symbols so that

$$\Sigma \to q \Leftrightarrow \Sigma \text{ leads } M \text{ to halt},$$

where $\to$ denotes convertibility by means of the transformations $\Sigma_i \to \Gamma_i$. A word Σ' which results from Σ by one of the new transformations $\Sigma_i \to \Gamma_i$ will also be called an M-successor of Σ, and likewise any M-successor of an ID will be called an ID.

Post now adds the inverse transformations and considers the semigroup $T = T(M)$ which results with generators $\{q_a\}$, $\{s_b\}$ and defining relations $\Sigma_i = \Gamma_i$. To show that

$$\Sigma = q \text{ in } T(M) \Leftrightarrow \Sigma \text{ leads } M \text{ to halt}$$

it suffices to show that the set $\text{Halt}(M) = \{\text{ID's } \Sigma \mid \Sigma \to q\}$ is closed under M-successors and M-predecessors. If $\Sigma \in \text{Halt}(M)$ and Σ' is an M-predecessor of Σ, then $\Sigma' \to \Sigma$ and it is immediate that $\Sigma' \in \text{Halt}(M)$. If Σ' is an M-successor of Σ, and if Σ' contains q, then $\Sigma' \to q$ trivially. If Σ' does not contain q, then Σ and Σ' are successive ID's in a computation and we can use Post's argument when M is deterministic. Namely, Σ' is then the *unique* M-successor of Σ and hence $\Sigma' \to q$ as part of the process $\Sigma \to q$, so $\Sigma' \in \text{Halt}(M)$. If M is nondeterministic we have to assume that M "preserves its past" in a suitable way, so that if Σ is an ID which can lead M to a halt then *any* M-successor Σ' of Σ can also lead M to a halt, in effect by recovering Σ. Any nondeterministic Turing machine N can be simulated in linear time by an M with this special property. For example, M can begin by duplicating the input of N, replacing each symbol s_b by a double symbol $s_b s_b$ in a single square of tape and thereafter behaving as N does, but only on the right-hand half of each double symbol (thus preserving the input as the sequence of left-hand halves), except that at any time M has the option of erasing the right half of each square and starting afresh by copying the input onto the now blank right halves of the squares.

Thus, whether or not M is deterministic, $T(M)$ simulates M in the sense that $\Sigma = q$ in $T(M) \Leftrightarrow \Sigma$ leads M to halt and the simulation is efficient, in fact linear time, because a computation $(\Sigma, \Sigma', \ldots, \Sigma^{(n)})$ is represented by a derivation of no more than $(2n + \text{length}(\Sigma))$ steps. The first n steps produce exactly the words $\Sigma', \ldots, \Sigma^{(n)}$, and if halting occurs the number of steps needed to reduce $\Sigma^{(n)}$ to q is at most $\text{length}(\Sigma) + n$, the maximum possible length of $\Sigma^{(n)}$.

DEFINITION. A system $\mathfrak{S}(M)$ simulates a Turing machine M efficiently if:

(1) There is a polynomial time computable mapping $\Sigma \to W(\Sigma)$ from ID's to words of $\mathfrak{S}(M)$, and a distinguished word Ω such that Ω is derivable from $W(\Sigma) \Leftrightarrow \Sigma$ leads M to halt.

(2) There is a polynomial time computable *derivation function* $D(\Sigma, \Sigma')$, whose value is a derivation of $W(\Sigma')$ from $W(\Sigma)$ when Σ' is an M-successor of Σ, and a derivation of Ω from $W(\Sigma)$ when $\Sigma = \Sigma'$ is a halting ID.

Since the length of output to a polynomial time computation is bounded by a polynomial function of the length of input, (2) implies a polynomial bound $\sigma(l)$ on the lengths of words required to derive $W(\Sigma')$ or Ω from $W(\Sigma)$, where $l = \text{length}(\Sigma)$. It likewise implies a polynomial bound $\sigma'(l)$ on the number of words in $D(\Sigma, \Sigma')$. Now if M takes s steps from Σ to halt, any ID in the computation has length $\leqslant l + s$, so (2) implies a derivation of Ω from $W(\Sigma)$ in $\leqslant s \cdot \sigma'(l + s)$ steps. Thus

$$\tau(l, s) = s \cdot \sigma'(l + s)$$

is a polynomial bound on the length of derivations in $\mathfrak{S}(M)$, relative to the length of computation in M and the size of the initial ID.

Of course, the lower the degree of σ and τ the better one feels the simulation to be, and Post's construction shows that linear σ and τ can be attained for semigroups. However, a simulation within polynomial bounds deserves to be considered efficient, since Turing machines themselves are the most efficient model of computation only modulo polynomial bounds.

A careful study of existing proofs of the unsolvability of the word problem for groups [**1–4**, **6**, **8**, **9**, **11**] shows that all require exponential time, and hence they do not efficiently simulate Turing machines in the above sense. It turns out, however, that a modification E of the Boone-Britton group of [**2**] can simulate a Turing machine efficiently, in fact in linear space (i.e. σ linear) and cubic time (i.e. τ cubic), hence E joins Post's semigroup T among the efficient models of computation. The present paper describes the modifications needed, and deduces an efficient topological model of computation. We use the notation of [**2**], except that we write presentations with angle brackets, and we extend the presentation notation slightly by writing

$$H = G \cup \langle \{h_i\}; \{U_j = V_j\} \rangle$$

to indicate that H results from G by adding new generators h_i and relations $U_j = V_j$ (which may involve generators of G).

The derivational steps we need are not only cancellations and substitutions according to the defining relations of E, but also conjugations by generators of E. This amounts to the same thing as considering words to be circular, then allowing only cancellations and substitutions. It can be justified on the grounds that

$$W \text{ equals } 1 \Leftrightarrow \text{all conjugates of } W \text{ equal } 1$$

and the fact that 1 is the word Ω which simulates "halt" in the Boone-Britton group. Conjugation is also a natural operation in the topological context, where words are represented by closed curves.

(ADDED June 6, 1982). It should be pointed out that the present work is quite different in content from the work of Trakhtenbrot [**15**] and Valiev [**16**, **17**] on the efficient reduction of word problems to halting problems. Both these authors are concerned with efficiently deciding, *by ordinary computation*, whether a given word

$W = 1$. They do not confine themselves to operations on W which have a group theoretic meaning. The purpose of our restriction to "derivational steps" which have a group theoretic meaning is to obtain results about the complexity of group theoretic processes and related topological processes. Some such results are given in the last section of this paper. Another consequence, which will be published elsewhere, is NP-completeness of the problem of finding bounded solutions to equations in free groups.

In fact, I should claim only to have observed the *existence* of efficient computation in groups, and some of its consequences, since it now appears that the group E constructed below is *not* the first which simulates computation efficiently. The referee has pointed out that the original construction of Boone [**0**] is not the same as the one in [**1**], and it appears to simulate Turing machines in polynomial time. The construction of E, if not actually necessary, nevertheless presents a "data compression" technique which is useful in work with the Boone-Britton group or the Cohen-Aanderaa group of [**4**] and [**14**], and at least it allows one to avoid reading the formidable details of [**0**]. Ideally, some dedicated group theorist should reexamine [**0**] and simplify it in the light of present day knowledge, since it appears capable of yielding a simulation in linear space and quadratic time, improving the results of the present paper.

Simulation of a Turing machine in the Boone-Britton group. The Boone-Britton group $G = G(M)$ has generators $\{q_a\}$, $\{s_b\}$, k, t, x, y, $\{l_i\}$, $\{r_i\}$, of which the q_a and s_b are the generators of Post's semigroup T. Any *special* word Σ of T, i.e. a word consisting of s_b's and one q_a symbol (such as an ID), is encoded by

$$W(\Sigma) = k^{-1}\Sigma^{-1}t\Sigma k\Sigma^{-1}t^{-1}\Sigma \tag{1}$$

and the defining relations are

$$\begin{aligned}
&s_b y = y^2 s_b, && xs_b = s_b x^2,\\
&s_b l_i = y l_i y s_b, && r_i s_b = s_b x r_i x,\\
&\Sigma_i = l_i \Gamma_i r_i && \text{(for each } \Sigma_i = \Gamma_i \text{ in } T),\\
&t l_i = l_i t, && r_i k = k r_i,\\
&ty = yt, && xk = kx,\\
& && k^{-1}q^{-1}tqkq^{-1}t^{-1}q = 1.
\end{aligned} \tag{2}$$

LEMMA 1. *If Σ' is an M-successor of a special word Σ of length l, then one can convert $W(\Sigma)$ to $W(\Sigma')$ by derivational steps of G, producing words which, except for the presence of two subwords of each of the forms*

$$y^{\pm(2^m-1)} l_i^{\pm 1} y^{\pm(2^m-1)}, \qquad x^{\pm(2^n-1)} r_i^{\pm 1} x^{\pm(2^n-1)},$$

where $m, n \leqslant l$, have lengths bounded by a linear function of l and involve only $\{q_a\}$, $\{s_b\}$, k, t.

PROOF. Since Σ is a special word we may assume $\Sigma = U\Sigma_i V$, $\Sigma' = U\Gamma_i V$, where U, V consist only of s_b's. First use the relation $\Sigma_i = l_i\Gamma_i r_i$ of (2) to convert

$$W(\Sigma) \equiv k^{-1}\Sigma^{-1}t\Sigma k\Sigma^{-1}t^{-1}\Sigma$$

to

$$k^{-1}(Ul_i\Gamma_i r_i V)^{-1}t(Ul_i\Gamma_i r_i V)k(Ul_i\Gamma_i r_i V)^{-1}t^{-1}(Ul_i\Gamma_i r_i V).$$

Now use the relations $s_b l_i = y l_i y s_b$ and $s_b y = y^2 s_b$ to move l_i (flanked by lengthening strings of y's) to the other side of U. We have

$$\begin{aligned} s_{b_m}\cdots s_{b_2}s_{b_1}l_i &= s_{b_m}\cdots s_{b_2}yl_iys_{b_1} \\ &= s_{b_m}\cdots y^2\cdot yl_iy\cdot y^2s_{b_2}s_{b_1} \\ &\vdots \\ &= y^{2^m-1}l_iy^{2^m-1}s_{b_m}\cdots s_{b_2}s_{b_1} \end{aligned}$$

as is easily proved by induction; hence if length$(U) = c$ we get

$$Ul_i = y^{2^c-1}l_iy^{2^c-1}U.$$

Similarly, if length$(V) = d$ we get

$$r_iV = Vx^{2^d-1}r_ix^{2^d-1},$$

and if we transport l_i, y's and r_i, x's simultaneously across the two occurrences of $U^{\pm 1}$, $V^{\pm 1}$ then at intermediate stages of the derivation we have two subwords of the forms

$$y^{\pm(2^m-1)}l_i^{\pm 1}y^{\pm(2^m-1)}, \qquad x^{\pm(2^n-1)}r_i^{\pm 1}x^{\pm(2^n-1)}, \qquad m, n \leqslant l.$$

Once $y^{-(2^m-1)}l_i^{-1}y^{-(2^m-1)}$ and $y^{2^m-1}l_iy^{2^m-1}$ are facing each other across the t and t^{-1}, they can be commuted across $t^{\pm 1}$ into mutual annihilation. The r_i's and x's can be annihilated similarly by commuting them across k and k^{-1} (remembering that we regard the word as circular), and we will then have

$$k^{-1}(U\Gamma_iV)^{-1}t(U\Gamma_iV)k(U\Gamma_iV)^{-1}t^{-1}(U\Gamma_iV),$$

that is, $W(\Sigma')$. □

The referee has pointed out that the conversion of $W(\Sigma)$ to $W(\Sigma')$ can actually be managed in linear space, essentially by moving only the outermost y and x in each of the four special subwords. The conversion still takes exponential time, however, since the same (exponential) number of y's and x's have to be moved.

Lemma 1 is essentially a sharpened form of the easy direction of the Boone-Britton theorem: if $\Sigma \to q$ in M then $W(\Sigma) = 1$ in G. For if we apply Lemma 1 to the sequence of M-successors of Σ which ends in q we will derive the word

$$W(q) \equiv k^{-1}q^{-1}tqkq^{-1}t^{-1}q$$

which equals 1 by the last relation of (2).

We shall re-enact the proof of Lemma 1 later (Theorem 2), when we have found a way to compress the powers of y and x to words whose lengths are of order l, where $l =$ length(Σ). The number of atomic operations in the derivation will then be of order l^2.

Power compression. If y is an element of infinite order in any group A then $y \mapsto y^3$ defines an isomorphism between subgroups of A and we can make an HNN extension to

$$H = A \cup \langle v; v^{-1}yv = y^3 \rangle.$$

The new element v can be used to "compress" powers of y because the new relation

$$v^{-1}yv = y^3 \tag{3}$$

implies $v^{-m}yv^m = y^{3^m}$, as follows easily from the equations

$$v^{-(m+1)}yv^{m+1} = v^{-m} \cdot v^{-1}yv \cdot v^m = v^{-m} \cdot y^3 \cdot v^m = v^{-m}yv^m \cdot v^{-m}yv^m \cdot v^{-m}yv^m$$

and induction on m. Life would be easier if we could use the relation

$$v^{-1}yv = y^2, \tag{3}'$$

which implies $v^{-m}yv^m = y^{2^m}$, as this gives a more convenient way to shorten the words $y^{2^m-1}l_iy^{2^m-1}$ in the derivation shown in Lemma 1. However, to move the v's past the s_b's and t, without spending exponential time converting them back to y's, we need the relations

$$s_bv = vs_b \quad \text{or} \quad v^{-1}s_bv = s_b, \tag{4}$$

$$tv = vt \quad \text{or} \quad v^{-1}tv = t, \tag{5}$$

and it so happens that (3)′, (4), (5) do not define an HNN extension of G, whereas (3), (4), (5) do. The latter fact will follow if we can prove:

(a) y is of infinite order in G;

(b) the subgroup of G generated by y, t, $\{s_b\}$ is $A_4 = \langle y, t, \{s_b\}; ty = yt, \{s_by = y^2s_b\}\rangle$;

(c) the homomorphism of A_4 determined by $y \mapsto y^3$, $t \mapsto t$, $s_b \mapsto s_b$ is an isomorphism.

We prove (a) and (b) in Lemma 2, (c) in Lemma 3. The notation $A \hookrightarrow B$ is used to express "A embeds in B". By the fundamental theorem on HNN extensions [7], $A \hookrightarrow H$ if H is an HNN extension of A.

LEMMA 2. *Let* $A_5 = \langle y; \rangle$; $A_4 = \langle y, t, \{s_b\}; ty = yt, \{s_by = y^2s_b\}\rangle$. *Then there are groups* A_3, A_2, A_1 *such that*

$$A_5 \hookrightarrow A_4 \hookrightarrow A_3 \hookrightarrow A_2 \hookrightarrow A_1 \hookrightarrow G.$$

PROOF. $A_5 \hookrightarrow A_4$ because $y \mapsto y$, $y \mapsto y^2$ are isomorphisms in A_5, hence A_5 embeds in the first, and all subsequent, HNN extensions of A_5 obtained by adding $\langle t; t^{-1}yt = y\rangle$; then all the $\langle s_b; s_bys_b^{-1} = y^2\rangle$ until we get A_4.

$$A_4 \hookrightarrow A_3 = A_4 \cup \langle x; \{xs_b = s_bx^2\}\rangle$$

because A_4 is the quotient of A_3 by the relation $x = 1$.

$$A_3 \hookrightarrow A_2 = A_3 \cup \langle \{r_i\}; \{r_i(s_bx^{-1})r_i^{-1} = s_bx\}\rangle$$
$$\cup \langle \{l_i\}; \{l_i^{-1}(y^{-1}s_b)l_i = ys_b\}, \{l_i^{-1}tl_i = t\}\rangle$$

because A_3 is obtained from A_3 by a series of HNN extensions. Namely, setting $x = y = 1$ we see that t, $\{s_b\}$ form a free subset of A_3. Then $s_bx^{-1} \mapsto s_bx$ defines an

isomorphism between free subgroups of A_3 and hence we have HNN extensions by the r_i. Likewise, $y^{-1}s_b \mapsto ys_b$, $t \mapsto t$ defines an isomorphism between free subgroups, so the l_i also give HNN extensions.

A_2 has all the generators of G except $\{q_a\}$, k, and all the relations except $\{\Sigma_i = l_i\Gamma_i r_i\}$, $\{r_i k = kr_i\}$, $xk = kx$ and $k^{-1}(q^{-1}tq)k = q^{-1}tq$. We now use a trick of Britton [**2**, part (z)] to show

$$A_2 \hookrightarrow A_1 = A_2 \cup \langle \{q_a\}; \{\Sigma_i = l_i\Gamma_i r_i\} \rangle.$$

It will suffice to show $A_2 \hookrightarrow H_1 = A_1 \cup \langle z; \rangle$, since A_2 then embeds in any subgroup of H containing the generators of A_2, in particular A_1. Now

$$H_1 = A_2 \cup \langle z, \{q_a\}; \{\Sigma_i = l_i\Gamma_i r_i\} \rangle$$

and if we eliminate the generators q_a in favour of generators $p_a = q_a z^{-1}$ we can write $\Sigma_i = F_i p_{\alpha_i} z G_i$, $\Gamma_i = H_i p_{\beta_i} z K_i$, giving

$$H_1 = A_2 \cup \left\langle z, \{p_a\}; \left\{ z^{-1}\left(p_{\beta_i}^{-1} H_i^{-1} l_i^{-1} p_{\alpha_i} \right) z = K_i r_i G_i^{-1} \right\} \right\rangle$$

which is an HNN extension of $A_2 \cup \langle \{p_a\}; \rangle$, the isomorphism being between free subgroups since the $\{l_i\}$ and $\{r_i\}$ are free. But $A_2 \hookrightarrow A_2 \cup \langle \{p_a\}; \rangle$ (as quotient by relations $p_a = 1$), hence $A_2 \hookrightarrow A_2 \cup \langle \{p_a\}; \rangle \hookrightarrow H_1$ as required.

Finally,

$$G = A_1 \cup \left\langle k; k^{-1}xk = x, k^{-1}\left(q^{-1}tq\right)k = q^{-1}tq, \left\{k^{-1}r_i k = r_i\right\} \right\rangle,$$

which is clearly an HNN extension of A_1; hence $A_1 \hookrightarrow G$. □

LEMMA 3. *The homomorphism* ϕ: $A_4 \to A_4$, *defined by* $\phi(y) = y^3$, $\phi(t) = t$, $\phi(s_b) = s_b$, *is an isomorphism.*

PROOF. We have to show that the kernel of ϕ is $\{1\}$.

The relation $s_b y = y^2 s_b$ in A_4 implies that any power of y can be moved to the left across a positive power of s_b (doubling its exponent in the process). Thus if any word W in A_4 contains a subword $s_b y^j s_b^{-1}$ we can cancel s_b and s_b^{-1} by first moving y^j to the left across s_b. Moreover, $s_b^{-1}y^{2j}s_b = s_b^{-1}s_b y^j$, hence s_b^{-1}, s_b can also be cancelled in this circumstance, and in the single case where we fail to cancel a pair s_b^{-1}, s_b with only y's between them, $s_b^{-1}y^{2j+1}s_b = s_b^{-1}ys_b y^j$. More trivially, the relation $ty = yt$ allows us to cancel any pair t^{-1}, t with only y's between them.

Let us call the result of deleting all y's in W the (s, t)-*part* of W. Applying the above processes repeatedly, we obtain a (nonunique) normal form of W in which the (s, t)-part contains no subwords of the form tt^{-1}, $t^{-1}t$ or $s_b s_b^{-1}$, and such that any occurrence of $s_b^{-1}s_b$ in the (s, t)-part corresponds to an occurrence of $s_b^{-1}ys_b$ in the normal form itself. Now suppose we compute $\phi(W)$ from the normal form of W, then reduce $\phi(W)$ to normal form as above. Any $s_b^{-1}ys_b$ in W becomes $s_b^{-1}y^3s_b = s_b^{-1}ys_b y$ in $\phi(W)$, hence $\phi(W)$ has a normal form with the same (s, t)-part as W.

If the $\phi(W)$ so obtained is equal to 1, we can apply Britton's lemma [**2**, Lemma 4] to conclude that $\phi(W)$ contains a subword of the form $s_b^{-1}Cs_b$ with $C \in \langle y^2; \rangle$, or $s_b C s_b^{-1}$ with $C \in \langle y; \rangle$, if it involves s_b at all. The normal form rules out such subwords, hence no $s_b^{\pm 1}$ occurs. Similarly there is no $t^{\pm 1}$, so the (s, t)-part of $\phi(W)$,

and hence of W, is empty. But then

$$\phi(W) = y^{3j} = 1 \Leftrightarrow j = 0 \Leftrightarrow W = y^j = 1$$

since y is of infinite order by Lemma 2. □

The fact that ϕ sends y to an odd power of y is crucial for the above proof, for if $\phi'(y) = y^{2j}$ then ϕ' does not send every normal form to a normal form with the same (s, t)-part. In particular, if ϕ': $A_4 \to A_4$ is defined by $\phi'(y) = y^2$, $\phi'(t) = t$, $\phi'(s_b) = s_b$ and $W = y^{-1}s_c s_b^{-1} y s_b s_c^{-1}$, we have $W \neq 1$ by Britton's lemma, but

$$\phi'(W) = y^{-2}s_c s_b^{-1} y^2 s_b s_c^{-1} = y^{-2}s_c y s_c^{-1} = y^{-2}y^2 s_c s_c^{-1} = 1.$$

Thus ϕ' is not an isomorphism, which explains why (3)′, (4), (5) do not define an HNN extension of G.

We can now prove that (3), (4), (5) do, namely

THEOREM 1. *$F = G \cup \langle v;\ v^{-1}yv = y^3,\ v^{-1}tv = t,\ \{v^{-1}s_b v = s_b\}\rangle$ is an HNN extension of G, and*

$$E = F \cup \left\langle u;\ u^{-1}xu = x^3,\ u^{-1}ku = k,\ \left\{u^{-1}s_b u = s_b\right\}\right\rangle$$

is an HNN extension of F.

PROOF. If is immediate from Lemmas 2 and 3 that F is an HNN extension of G, and hence $G \hookrightarrow F$.

Now using the following sequence of groups:

$B_5 = \langle x;\ \rangle$,

$B_4 = B_5 \cup \langle k, \{s_b\};\ k^{-1}xk = x,\ \{s_b^{-1}xs_b = x^2\}\rangle$,

$B_3 = B_4 \cup \langle y;\ \{s_b y = y^2 s_b\}\rangle$,

$B_2 = B_3 \cup \langle \{l_i\}; \{l_i^{-1}(y^{-1}s_b)l_i = ys_b\}\rangle \cup \langle \{r_i\}; \{r_i(s_b x^{-1})r_i^{-1} = s_b x\}, \{r_i^{-1}kr_i = k\}\rangle$,

$B_1 = B_2 \cup \langle \{q_a\};\ \{\Sigma_i = l_i \Gamma_i r_i\}\rangle$,

$I_1 = B_1 \cup \langle z;\ \rangle$,

$G = B_1 \cup \langle t;\ t^{-1}yt = y,\ t^{-1}(qk^{-1}q^{-1})t = qk^{-1}q^{-1},\ \{t^{-1}l_i t = l_i\}\rangle$,

one proves exactly as in Lemma 2 that B_4 is the subgroup of G generated by x, k, $\{s_b\}$. Then exactly as in Lemma 3 one shows that $\psi(x) = x^3$, $\psi(k) = k$, $\psi(s_b) = s_b$ determine an isomorphism in G, and hence in F, since $G \hookrightarrow F$. Thus the isomorphism condition is satisfied and E is an HNN extension of F. □

Theorem 1 gives $G \hookrightarrow E$ and in particular the hard direction of the Boone-Britton theorem remains true: if Σ is a special word

$$W(\Sigma) = 1 \quad \text{in } E \Rightarrow \Sigma \to q \quad \text{in } M.$$

In the next section we rederive the easy direction, using the fact that

$$y^{3^m} = v^{-m}yv^m \quad \text{and} \quad x^{3^n} = u^{-n}xu^n \quad \text{in } E$$

to economise on space and time.

Calculation with compressed powers. To avoid irrelevant details of polynomial expressions in this section, we shall say that a quantity X is of order l^p, or $O(l^p)$ for short, if X is bounded by a pth degree polynomial function of l. Whenever we have a sequence of $\leqslant l$ quantities $X(j)$ depending on a parameter j, and such that $X(j)$ is

$O(j^p)$, it will be true that each $X(j)$ can be bounded by the *same* pth degree polynomial in j, hence we can conclude that $X(1) + X(2) + \cdots$ is $O(l^{p+1})$.

We begin by finding convenient expressions for the terms $y^{2^m-1}l_i y^{2^m-1}$ which occur in the derivation in Lemma 1. Let $2^m - 1 = [\varepsilon_e \cdots \varepsilon_1 \varepsilon_0]$, where each $\varepsilon_h = 0$, 1 or 2, be the base 3 numeral for $2^m - 1$. Now we have

$$2^m - 1 = \varepsilon_e \cdot 3^e + \cdots + \varepsilon_1 \cdot 3 + \varepsilon_0$$

by definition of base 3 numerals, and

$$y^{\varepsilon_h \cdot 3^h} = \left(y^{3^h}\right)^{\varepsilon_h} = \left(v^{-h} y v^h\right)^{\varepsilon_h} = v^{-h} y^{\varepsilon_h} v^h.$$

Hence

$$y^{2^m-1} = v^{-e} y^{\varepsilon_e} v^e \cdots v^{-1} y^{\varepsilon_1} v y^{\varepsilon_0},$$

and also

$$= y^{\varepsilon_0} \cdot v^{-1} y^{\varepsilon_1} v \cdots v^{-e} y^{\varepsilon_e} v^e.$$

Following a suggestion of the referee, we cancel v's as far as possible in these two expressions for y^{2^m-1}, obtaining

$$v^{-e} y^{\varepsilon_e} v y^{\varepsilon_{e-1}} v \cdots v y^{\varepsilon_1} v y^{\varepsilon_0}, \tag{6}$$

and

$$y^{\varepsilon_0} v^{-1} y^{\varepsilon_1} v^{-1} \cdots v^{-1} y^{\varepsilon_e} v^e, \tag{7}$$

each of which has length $O(e) = O(m)$. Hence the expression

$$v^{-e} y^{\varepsilon_e} v \cdots v y^{\varepsilon_1} v y^{\varepsilon_0} \cdot l_i \cdot y^{\varepsilon_0} v^{-1} y^{\varepsilon_1} v^{-1} \cdots v^{-1} y^{\varepsilon_e} v^e \tag{8}$$

for $y^{2^m-1}l_i y^{2^m-1}$ has length $O(m)$.

If (6) and (7) are viewed as base 3 words for $2^m - 1$, then they should turn into the corresponding words for $2^{m+1} - 1$ as (8) is passed to the left across an s_b, since $y^{2^m-1}l_i y^{2^m-1}$ turns into $y^{2^{m+1}-1}l_i y^{2^{m+1}-1}$. In fact, this conversion can be made in linear time by imitating the ordinary method of computation with base 3 numerals, as the following lemma shows.

LEMMA 4. (i) *If* $[\varepsilon_e \cdots \varepsilon_1 \varepsilon_0] + 1 = [\varepsilon'_{e+1} \cdots \varepsilon'_1 \varepsilon'_0]$ *then* $v^{-e} y^{\varepsilon_e} v \cdots v y^{\varepsilon_1} v y^{\varepsilon_0} y$ *can be converted to* $v^{-(e+1)} y^{\varepsilon'_{e+1}} v \cdots v y^{\varepsilon'_1} v y^{\varepsilon'_0}$ *in* $O(e)$ *steps.*

(ii) *If* $[\varepsilon_e \cdots \varepsilon_1 \varepsilon_0] \times 2 = [\varepsilon''_{e+1} \cdots \varepsilon''_1 \varepsilon''_0]$ *then* $v^{-e} t^{2\varepsilon_e} v \cdots v y^{2\varepsilon_1} v y^{2\varepsilon_0}$ *can be converted to* $v^{-(e+1)} y^{\varepsilon''_{e+1}} v \cdots v y^{\varepsilon''_1} v y^{\varepsilon''_0}$ *in* $O(e)$ *steps.*

There are analogous results for words of the form (7).

PROOF. (i) $v^{-e} y^{\varepsilon_e} v \cdots v y^{\varepsilon_1} v y^{\varepsilon_0} \cdot y = v^{-e} y^{\varepsilon_e} v \cdots v y^{\varepsilon_1} v y^{\delta_0 + \varepsilon'_0}$ where $\delta_0 = 0$ or 3 is the *carry*. If $\delta_0 = 0$ then $\varepsilon_1 = \varepsilon'_1, \ldots, \varepsilon_e = \varepsilon'_e$, $\varepsilon'_{e+1} = 0$ and there is nothing to do but insert $v^{-1}v$ after v^{-e} to get $v^{-(e+1)} y^{\varepsilon'_{e+1}} v \cdots v y^{\varepsilon'_1} v y^{\varepsilon'_0}$. If $\delta_0 = 3$ we pass y^{δ_0} to the left using the relation $vy^3 = yv$ from (2), obtaining

$$v^{-e} y^{\varepsilon_e} v \cdots v y^{\varepsilon_1} y v y^{\varepsilon'_0} = v^{-e} y^{\varepsilon_e} v \cdots v y^{\delta_1 + \varepsilon'_1} v y^{\varepsilon'_0}$$

where $\delta_1 = 0$ or 3 is the new carry. Then proceed as before, propagating the carry to the left, until one has

$$v^{-e} y^{\delta_e + \varepsilon'_e} v \cdots v y^{\varepsilon'_1} v y^{\varepsilon'_0} = v^{-(e+1)} v y^{\delta_e + \varepsilon'_e} v \cdots v y^{\varepsilon'_1} v y^{\varepsilon'_0},$$

inserting $v^{-1}v$ if $\delta_e = 3$

$$= v^{-(e+1)}y^{\varepsilon'_{e+1}}vy^{\varepsilon'_e}v \cdots vy^{\varepsilon'_1}vy^{\varepsilon'_0}$$

using $vy^3 = yv$.

The whole process takes $O(e)$ steps.

(ii) $v^{-e}y^{2\varepsilon_e}v \cdots vy^{2\varepsilon_1}vy^{2\varepsilon_0} = v^{-e}y^{2\varepsilon_e}v \cdots vy^{2\varepsilon_1}vy^{\gamma_0+\varepsilon''_0}$ where $\gamma_0 = 0$ or 3 is the carry. If $\gamma_0 = 3$ then we pass y^{γ_0} to the left using $vy^3 = yv$, obtaining

$$v^{-e}y^{2\varepsilon_e}v \cdots vy^{2\varepsilon_1} \cdot yvy^{\varepsilon''_0} = v^{-e}y^{2\varepsilon_e}v \cdots vy^{\gamma_1+\varepsilon''_1}vy^{\varepsilon''_0}$$

where $\gamma_1 = 0$ or 3 is the new carry. If $\gamma_0 = 0$ then γ_1 is defined by $2\varepsilon_1 = \gamma_1 + \varepsilon''_1$. In either case we send y^{γ_1} to the left as we did y^{γ_0}, and so on, arriving at $v^{-(e+1)}y^{\varepsilon''_{e+1}}v \cdots vy^{\varepsilon''_1}vy^{\varepsilon''_0}$ in $O(e)$ steps, much as in (i).

Completely analogous derivations are used for words of the form (7). □

Our main theorem now follows.

THEOREM 2. *E simulates Turing machine computation in linear space and cubic time.*

PROOF. We first use Lemma 4 to show that the word (8) for $y^{2^m-1}l_iy^{2^m-1}$ can be passed to the left across an s_b and turned into the corresponding word for $y^{2^{m+1}-1}l_iy^{2^{m+1}-1}$ in $O(m)$ steps. The y^{2^m-1} on the left of l_i is expressed by a base 3 word

$$v^{-e}y^{\varepsilon_e}v \cdots vy^{\varepsilon_1}vy^{\varepsilon_0}. \tag{6}$$

Since $s_bv = vs_b$ and $s_by = y^2s_b$ by (2), (6) can be converted to $v^{-e}y^{2\varepsilon_e}v \cdots vy^{2\varepsilon_1}vy^{2\varepsilon_0}$, on the other side of s_b, in $O(e) = O(m)$ steps. Lemma 4(ii) says that the latter can be converted to $v^{-(e+1)}y^{\varepsilon''_{e+1}}v \cdots vy^{\varepsilon''_1}vy^{\varepsilon''_0}$ in $O(e)$ steps, where $[\varepsilon''_{e+1},\ldots,\varepsilon''_1] = [\varepsilon_e,\ldots,\varepsilon_1,\varepsilon_0] \times 2 = 2^{m+1} - 2$. Next we pass l_i to the left by the relation $s_bl_i = yl_iys_b$ of (2), so on the left of l_i we now have

$$v^{-(e+1)}y^{\varepsilon''_{e+1}}v \cdots vy^{\varepsilon''_1}vy^{\varepsilon''_0} \cdot y,$$

which, by Lemma 4(i), can be converted in $O(e)$ steps to

$$v^{-(e+1)}y^{\varepsilon'_{e+1}}v \cdots vy^{\varepsilon'_1}vy^{\varepsilon'_0}$$

where

$$[\varepsilon'_{e+1},\ldots,\varepsilon'_1,\varepsilon'_0] = [\varepsilon''_{e+1},\ldots,\varepsilon''_1,\varepsilon''_0] + 1 = 2^{m+1} - 1,$$

as required. The word (7) on the right of l_i in (8) can be handled in a completely analogous way, hence the whole word (8) can be passed leftwards across an s_b, with appropriate changes, in $O(e) = O(m)$ steps.

We can now reorganize the derivation of $W(\Sigma')$ from $W(\Sigma)$ in Lemma 1 to operate in linear space and cubic time.

Let $\Sigma = U\Sigma_iV$, $\Sigma_i = U\Gamma_iV$ as before, and first consider what happens to the subwords $\Sigma^{\pm 1}$ of $W(\Sigma)$. We convert $U\Sigma_iV$ to $Ul_i\Gamma_ir_iV$ and then move l_i (flanked by lengthening words (6) and (7)) to the other side of U, and r_i (accompanied similarly by a lengthening entourage of u's and x's) to the other side of V.

The movement across U divides into c (= length(U)) stages, in each of which the word (8) passes a single letter s_b in U. By the argument above, each such stage can be

completed in $O(l)$ steps, where $l = \text{length}(\Sigma) \geqslant m$. The movement across V divides into d ($= \text{length}(V)$) similar stages, and c, $d \leqslant l$, hence the movements across U and V can be completed in $O(l^2)$ steps, after which the entourages of l_i and r_i have length $O(l)$. The remaining movements across t and k, and subsequent annihilations, take time of the same order as the length of the entourages, and there is no further increase in word length, hence the whole derivation from $W(\Sigma)$ to $W(\Sigma')$ takes space $O(l)$ and time $O(l^2)$. This gives the linear space bound on E's simulation.

To find the time bound, suppose there are s steps in the computation by Turing machine M. Then there are $O(s)$ steps in the simulation by T, and if l is the length of the initial ID, any ID Σ occurring in the simulation by T has length $\leqslant l + s$. The simulation by E from $W(\Sigma)$ to $W(\Sigma')$ therefore takes $\leqslant \rho(l + s)$ steps, where ρ is some quadratic polynomial, hence there are $\leqslant s \cdot p(l + s)$ steps in E's simulation of the whole computation. Thus we have a time bound $s \cdot \rho(l + s)$ which is cubic in s (and quadratic in l). □

Computation in surface complexes. Computation in surface complexes was discussed in [**12** and **13**]. A brief recapitulation of the relevant parts is as follows. Given a group $H = \langle \{a_i\}; \{r_j = 1\} \rangle$, one constructs a 2-dimensional cell complex $C = C(H)$ by taking a bouquet of circles, corresponding to the a_i, and attaching a disc d_j, with boundary corresponding to r_j, for each relator. C can be given a triangulation $S = S(H)$ to produce a 2-dimensional simplicial complex (C, S) whose description is roughly proportional in size to the presentation chosen for H.

A word W of H corresponds to a closed edge path w of S, and an atomic operation on W is simulated by a series of simplicial homotopies on w—insertion or deletion of r_j by pulling w from one side of d_j to the other, and insertion or deletion of $a_i a_i^{-1}$ or $a_i^{-1} a_i$ by insertion or deletion of *spurs*, where a spur is a path which runs out and back along an edge of S. Since there are constant bounds on the number of homotopies required to pull a path across a d_j one triangle at a time, or to pull out a path $a_i a_i^{-1}$ one spur at a time, this system simulates computation in H in linear space and time. In particular, if we take H to be the group $E = E(M)$, then M's computation can be simulated by simplicial homotopies in $(C(E), S(E))$ in linear space and cubic time by Theorem 2. Halting of M's computation corresponds to the contraction of w to a point.

In [**13**] we gave a modification of the above construction which allows the curve w to remain simple at all times. The circles a_i are replaced by annuli $A_i = I \times a_i$ with a single transverse segment I in common. Any word W in the generators a_i can be represented by a simple arc $\mathring{w}$ winding round the annuli, with its ends at different points of I. A square $I \times J$ which meets the annuli only along I provides space for $\mathring{w}$ to be closed to a simple curve $\bar{w}$. Disjoint interior subintervals I_j of I yield disjoint subannuli $I_j \times a_i$ which are used to carry arcs $\mathring{r}_j$ representing the defining relators of H. We take the ends of $\mathring{r}_j$ to be ends of I_j; then spanning each $\mathring{r}_j \cup I_j$ by a disc d_j' gives a complex $C'(H)$ which deformation retracts to $C(H)$, hence it has the same fundamental group. Thus the same relations hold in $C'(H)$, and computation can still be simulated.

To simulate conversion of $W = UV$ to Ur_jV one isotopes $\bar{w}$ into the form uI_jv, where u, v are arcs disjoint from $\mathring{r}_j$ representing U and V, then pulls its subarc I_j across d_j' to position $\mathring{r}_j$. Deletion of r_j is done by the reverse process, and insertion or deletion of $a_ia_i^{-1}$ or $a_i^{-1}a_i$ is simulated even more trivially by isotopies. Thus the curve remains simple at all times, and it can clearly be represented by an edge path in some triangulation, although one needs increasingly fine triangulations of $C'(H)$ as longer words have to be handled.

Assuming a bound l on the lengths of words is given in advance, a suitable triangulation $S = S(l)$ can be constructed as follows. To represent a word of length l one needs l "tracks" on each annulus to carry up to l turns of an arc. The arc must also be able to move sideways enough for any part of it to be isotoped onto an I_j, say by having l tracks between any two adjacent subannuli $I_j \times A_i$. Such tracks, and transverse connections between them to allow the curve to jump from one track to the next, can be provided by $O(l)$ triangles. The extra triangulation needed for the $\mathring{r}_j$ and d_j' is only a constant amount, and the amount of triangulation of the square $I \times J$ needed to carry the arcs which close the $\mathring{w}$ to $\bar{w}$ is also $O(l)$. Thus the simulation of computation still remains in linear space, though the degree of the time bound goes up by two—it takes $O(l^2)$ simplicial isotopies to manipulate $\bar{w}$ into the form uI_jr ready to simulate insertion of the relator r_j, since each of the $\leqslant l$ turns of $\bar{w}$ has to be moved across $O(l)$ tracks.

With a fixed triangulation S of C', the problem of deciding whether a simple curve w reduces to the boundary of a triangle by simplicial isotopies of S (that is, whether w is *simplicially unknotted in C' with respect to S*) is a PSPACE problem, as was noted in [**13**]. We are now able to prove that this problem is PSPACE-complete.

We refer to [**5**] for a discussion of PSPACE and PSPACE-completeness. In particular, we shall appeal to the PSPACE-completeness of the following problem [**5**, p. 175].

$\mathfrak{L}$: Given a Turing machine M which works in space equal in length to the initial ID, and an ID Σ, decide whether Σ leads M to halt.

THEOREM 3. *The problem $\mathfrak{K}$ of deciding, given a finite 2-dimensional simplicial complex (C, S) and a simple edge path w of S, whether w is simplicially unknotted in C with respect to S, is* PSPACE*-complete.*

PROOF. Since $\mathfrak{K}$ is in PSPACE, it will suffice to reduce $\mathfrak{L}$ to it in polynomial time.

To do this we construct $E(M)$, then $C(E)$, and a path $w(\Sigma)$ corresponding to $W(\Sigma)$. It takes quadratic time to write down the presentation of $E(M)$ (because of the relations with two sets of indices) and essentially the same time to describe $C(E)$ and $w(\Sigma)$. Since the ID's of M being simulated have lengths $\leqslant$ length(Σ), Theorem 2 tells us that the lengths of words W used when $E(M)$ simulates M can be bounded by a linear function of $l =$ length(Σ), and hence in time polynomial in l we can construct a triangulation $S(E, \Sigma)$ of $C(E)$ in which the operations of E can be simulated by simplicial isotopies.

Then

$$\Sigma \text{ leads } M \text{ to halt} \Leftrightarrow W(\Sigma) = 1 \text{ in } E \Leftrightarrow w(\Sigma) \text{ is simplicially unknotted in } C(E) \text{ with respect to } S(E, \Sigma).$$

It is clear that $C(E(M))$, $S(E(M), \Sigma)$, $w(\Sigma)$ can be uniformly computed from M, Σ; hence we have a polynomial time reduction of $\mathfrak{L}$ to $\mathfrak{K}$. □

Theorem 3 complements the results on computation in surface complexes in [**12**]. The general framework is the following:

"machine" = 2-dimensional simplicial complex (C, S);

"input" = simple edge path w of S;

"atomic operations" = simplicial homotopies of S;

"halting" = reduction of w to the boundary of a triangle of S. If the simplicial homotopies are

(a) unrestricted, we get general recursive computation [**12**],

(b) restricted to isotopies, we get PSPACE computation (Theorem 3),

(c) restricted to isotopies which never return the path to an edge it has previously left, we get NP computation [**12**].

References

0. W. W. Boone, *Certain simple unsolvable problems of group theory*. I–VI, Nederl. Akad. Wetensch. Indag. Math. **57** (1954), 231–237, 492–497; **58** (1955), 252–256, 571–577; **60** (1957), 22–27, 227–232.

1. _____, *The word problem*, Ann. of Math. (2) **70** (1959), 207–265.

2. J. L. Britton, *The word problem*, Ann. of Math. (2) **77** (1963), 16–32.

3. _____, *The word problem for groups*, Proc. London Math. Soc. **8** (1958), 493–506.

4. D. E. Cohen and S. Aanderaa, *Modular machines, the word problem for finitely presented groups, and Collins' theorem*, Word Problems. II (S. I. Adian, W. W. Boone and G. Higman, editors), North-Holland, Amsterdam, 1980, pp. 1–16.

5. M. R. Garey and D. S. Johnson, *Computers and intractibility*, Freeman, San Francisco, Calif., 1979.

6. G. Higman, *Subgroups of finitely presented groups*, Proc. Roy. Soc. London Ser. A **262** (1961), 455–475.

7. G. Higman, B. H. Neumann and H. Neumann, *Embedding theorems for groups*, J. London Math. Soc. **24** (1949), 247–254.

8. R. McKenzie and R. J. Thompson, *An elementary construction of unsolvable word problems in group theory*, Word Problems (W. W. Boone, F. B. Cannonito and R. C. Lyndon, editors), North-Holland, Amsterdam, 1973, pp. 457–478.

9. P. S. Novikov, *On the algorithmic unsolvability of the word problem in group theory*, Trudy Mat. Inst. Steklov **44** (1955), (Russian)

10. E. L. Post, *Recursive unsolvability of a problem of Thue*, J. Symbolic Logic **12** (1947), 1–11.

11. J. R. Shoenfield, *Mathematical logic*, Addison-Wesley, Reading, Mass., 1967.

12. J. C. Stillwell, *Isotopy in surface complexes from the computational viewpoint*, Bull. Austral. Math. Soc. **20** (1979), 1–6.

13. _____, *Unsolvability of the knot problem for surface complexes*, Bull. Austral. Math. Soc. **20** (1979), 131–137.

14. _____, *The word problem and the isomorphism problem for groups*, Bull. Amer. Math. Soc. **6** (1982), 33–56.

15. B. A. Trakhtenbrot, *On the complexity of reduction algorithms in Novikov-Boone constructions*, Algebra and Logic **8** (1969), 93–128.

16. M. K. Valiev, *On the complexity of the identity problem for finitely defined groups*, Algebra and Logic **8** (1969), 5–43.

17. _____, *On polynomial reducibiliity of word problem under embedding of recursively presented groups in finitely presented groups*, Lecture Notes in Computer Science, vol. 32, Springer-Verlag, Berlin and New York, 1975, pp. 432–438.

Department of Mathematics, Monash University, Clayton, Victoria, Australia 3168