Journal of Algebra 321 (2009) 2187–2205

Contents lists available at ScienceDirect

Journal of Algebra

www.elsevier.com/locate/jalgebra

Small overlap monoids I: The word problem

Mark Kambites

School of Mathematics, University of Manchester, Alan Turing Building, Oxford Road, Manchester M13 9PL, England, United Kingdom

ARTICLE INFO

Article history:
Received 24 June 2008
Available online 11 November 2008
Communicated by Derek Holt

Keywords:
Monoid
Semigroup
Word problem
Finite presentation
Small overlap
Small cancellation

ABSTRACT

We develop a combinatorial approach to the study of semigroups and monoids with finite presentations satisfying small overlap conditions. In contrast to existing geometric methods, our approach facilitates a sequential left–right analysis of words which lends itself to the development of practical, efficient computational algorithms. In particular, we obtain a highly practical linear time solution to the word problem for monoids and semigroups with finite presentations satisfying the condition $C(4)$, and a polynomial time solution to the uniform word problem for presentations satisfying the same condition.

Small overlap conditions are simple and natural combinatorial conditions on semigroup and monoid presentations, which serve to limit the complexity of derivation sequences between equivalent words in the generators. They form a natural semigroup-theoretic analogue of the *small cancellation* conditions which are extensively used in combinatorial and computational group theory [8].

It is well known that every group admitting a finite presentation satisfying suitable small cancellation conditions is *word hyperbolic* in the sense of Gromov [2], and in particular has word problem solvable in linear time. In the 1970s, Remmers [11,12] developed an elegant geometric theory of small overlap semigroups, using the natural semigroup-theoretic analogue of the *van Kampen diagrams* extensively employed in combinatorial group theory (see for example [8]). He applied his methods to show that semigroups satisfying sufficiently small overlap conditions have what would now be called *linear Dehn function*, that is, that the minimum length of a derivation sequence between any two equivalent words is bounded above by a linear function of the word lengths. In theory, it follows immediately that one can test if two words in the generators for such a semigroup are equivalent, by exhaustively searching the (finite) space of all applicable derivation sequences of the given length, to see if any of them transforms one word to the other. However, the number of possible derivation sequences, and hence the time complexity of this algorithm, is exponential in the word length. More

E-mail address: Mark.Kambites@manchester.ac.uk.

0021-8693/$ – see front matter
doi:10.1016/j.jalgebra.2008.09.038

sophisticated techniques (such as applications of graph reachability algorithms) are of course applicable, but the problem remains one of searching a space of exponential size, and so we cannot really hope that this approach will lead to a tractable solution for the word problem. The question naturally arises, then, of how hard the word problem really is in these semigroups.

In this paper, we develop a new approach to the study of this important class of semigroups and monoids, along purely combinatorial lines. While our work lacks some of the mathematical elegance of Remmers' approach—indeed our foundational results are of a rather technical nature and our proofs mainly by case analysis—it has the advantage of permitting a *sequential* (left–right) analysis of elements, which for computational purposes seems more relevant than a geometric viewpoint. Two computational consequences of the theory we develop are of particular interest. The first is a linear time (on a two-tape Turing machine) algorithm to solve the word problem in any semigroup with a presentation satisfying Remmers' condition $C(4)$. The second is a polynomial time (more precisely, in the RAM model, quadratic in the presentation length and linear in the word length) solution to the uniform word problem for presentations satisfying the same condition. While the proofs of correctness and of the time complexity bounds for these algorithms are quite difficult, the algorithms themselves are quite straightforward to describe and eminently suitable for practical implementation. A demonstration implementation [7] is available as a package for the GAP computer algebra system [1].

In a future article [6] we shall establish some further consequences of the theory and methods developed here, including the facts that every monoid admitting a $C(4)$ presentation is *asynchronous automatic*, *rational* in the sense of Sakarovitch [10,13] and *word hyperbolic* in the sense of Duncan and Gilman. It will follow also that such monoids admit analogues of Kleene's theorem, and allow straightforward solutions to important algorithmic problems, such as the rational subset and submonoid membership problems.

Like word hyperbolicity in groups [9], small overlap conditions are also "generic" properties of finitely presented monoids and semigroups, in the sense that a finite presentation chosen uniformly at random from amongst those with a given alphabet, number of relations and total length, will satisfy any given small overlap condition with probability which approaches 1 as the size of the presentation approaches infinity [5]. This means that our results are potentially relevant to the emerging field of cryptography based on non-commutative algebraic structures. Several authors have proposed cryptosystems based on the difficulty of algorithmic problems in combinatorial algebra (see [14] for a survey), for some of which security is dependent upon the difficulty of computational problems relating to a randomly chosen finite presentation. The genericity of small overlap conditions, combined with the ease of performing computation in monoids satisfying these conditions, suggests that if any such system is to be secure, it will have to employ relatively sophisticated methods of random generation which avoid an overwhelming majority of presentations for which computation is too easy.

In addition to this introduction, this paper comprises five sections. In Section 1 we briefly recall the definitions of small overlap semigroups and monoids, together with some of their properties, and introduce some notation and terminology which will be used in the rest of the paper. Section 2 establishes some technical combinatorial properties of small overlap monoids, which are then used in Section 3 to give a sequential characterisation of equivalence for two words in the generators of a $C(4)$ presentation. Section 4 shows how this characterisation can be used to develop a linear time algorithm for the solution of the word problem of a fixed small overlap presentation. Finally, in Section 5 we apply our techniques to the solution of the uniform word problem for $C(4)$ presentations; we also observe that one can test efficiently whether an arbitrary presentation satisfies the condition $C(4)$.

The relationship of this work to the geometric approach developed by Remmers [11] perhaps deserves a further comment. As already mentioned, our approach to small overlap semigroups is entirely combinatorial and, in its finished state, makes no direct use of Remmers' geometric machinery. However, the author would most likely never have arrived at this viewpoint without the insight and intuition afforded by Remmers' approach, and the reader interested in fully understanding the present paper may find it helpful to study also Remmers' work in parallel. Some of his results have been given a very accessible treatment by Higgins [3], but unfortunately the only complete source still seems to be his thesis [11].

1. Preliminaries

We assume familiarity with basic notions of combinatorial semigroup theory, including free semigroups and monoids, and semigroup and monoid presentations. In all but Section 5 of the paper, which is devoted to uniform decision problems, we assume we have a fixed finite presentation for a monoid (or semigroup—we shall see shortly that the difference is unimportant). Words are assumed to be drawn from the free monoid on the generating alphabet unless otherwise stated. We write $u = v$ to indicate that two words are equal in the free monoid or semigroup, and $u \equiv v$ to indicate that they represent the same element of the monoid or semigroup presented. We say that a word p is a *possible prefix* of u if there exists a (possibly empty) word w with $pw \equiv u$, that is, if the element represented by u lies in the right ideal generated by the element represented by p. The empty word is denoted ϵ.

A *relation word* is a word which occurs as one side of a relation in the presentation. A *piece* is a word in the generators which occurs as a factor in sides of two different relations, or as a factor of both sides of a relation, or in two different (possibly overlapping) places within one side of a relation. To ensure a uniform treatment for free semigroups and monoids, we make the convention that the empty word ϵ is always a piece, even if the presentation has no relations.

The presentation is said to *satisfy the condition* $C(n)$, where n is a positive integer, if no relation word can be written as the product of *strictly fewer than* n pieces. Thus for each n, $C(n+1)$ is a strictly stronger condition than $C(n)$. We briefly mention another related condition. The presentation *satisfies the condition* $C'(x)$, where $0 \leqslant x \leqslant 1$, if whenever a piece p occurs as a factor of a relation word R we have $|p| < x|R|$. Notice that if n is a positive integer, then a semigroup satisfying $C'(1/n)$ will certainly satisfy $C(n+1)$.

The weakest meaningful small overlap condition, $C(1)$, says that no relation word is a product of zero pieces, that is, that ϵ is not a relation word. From this we see that in a small overlap monoid presentation, no non-empty word can be equivalent to the empty word, that is, no non-empty word can represent the identity. It follows that every small overlap monoid presentation is also interpretable as a semigroup presentation, and that the monoid presented is isomorphic to the semigroup presented with an adjoined identity element. For simplicity in what follows we shall focus upon small overlap monoids, but from each of our results one can immediately deduce a corresponding result for small overlap semigroups.

For each relation word R, let X_R and Z_R denote respectively the longest prefix of R which is a piece, and the longest suffix of R which is a piece. If the presentation satisfies $C(3)$ then R cannot be written as a product of two pieces, so this prefix and suffix cannot meet; thus, R admits a factorisation $X_R Y_R Z_R$ for some non-empty word Y_R. If moreover the presentation satisfies the stronger condition $C(4)$ then R cannot be written as a product of three pieces, so Y_R is not a piece. The converse also holds: a $C(3)$ presentation such that no Y_R is a piece is a $C(4)$ presentation. We call X_R, Y_R and Z_R the *maximal piece prefix*, the *middle word* and the *maximal piece suffix* respectively of R.

Assuming now that the presentation satisfies at least the condition $C(3)$, we shall use the letters X, Y and Z (sometimes with adornments or subscripts) exclusively to represent maximal piece prefixes, middle words and maximal piece suffixes respectively of relation words; two such letters with the same subscript or adornment (or with none) will be assumed to stand for the appropriate factors of the same relation word.

If R is a relation word we write $\overline{R}$ for the (necessarily unique, as a result of the small overlap condition) word such that $(R, \overline{R})$ or $(\overline{R}, R)$ is a relation in the presentation. We write $\overline{X_R}$, $\overline{Y_R}$ and $\overline{Z_R}$ for $X_{\overline{R}}$, $Y_{\overline{R}}$ and $Z_{\overline{R}}$ respectively. (This is an abuse of notation since, for example, the word X_R may be a maximal piece prefix of two distinct relation words, but we shall be careful to ensure that the meaning is clear from the context.)

2. Weak cancellativity properties

To perform efficient computations with words, it is very helpful to be able to process them in a sequential, left–right manner. To facilitate this in the case of the word problem for small overlap monoids, we need to know what can be deduced about the equivalence (or non-equivalence) of two

words from prefixes of those words. This section develops a theory with this end in mind, including a number of technical properties which can be viewed as weak cancellativity conditions satisfied by small overlap monoids. We assume throughout a fixed monoid presentation satisfying the small overlap condition $C(4)$.

We first introduce some terminology. A *relation prefix* of a word is a prefix which admits a (necessarily unique, as a consequence of the small overlap condition) factorisation of the form aXY where X and Y are the maximal piece prefix and middle word respectively of some relation word XYZ. An *overlap prefix (of length n)* of a word u is a relation prefix which admits an (again necessarily unique) factorisation of the form $bX_1Y_1'X_2Y_2'\dots X_nY_n$ where

- $n \geqslant 1$;
- $bX_1Y_1'X_2Y_2'\dots X_nY_n$ has no factor of the form X_0Y_0, where X_0 and Y_0 are the maximal piece prefix and middle word respectively of some relation word, beginning before the end of the prefix b;
- for each $1 \leqslant i \leqslant n$, $R_i = X_iY_iZ_i$ is a relation word with X_i and Z_i the maximal piece prefix and suffix respectively; and
- for each $1 \leqslant i < n$, Y_i' is a proper, non-empty prefix of Y_i.

Notice that if a word has a relation prefix, then the shortest such must be an overlap prefix. A relation prefix aXY of a word u is called *clean* if u does *not* have a prefix

$$aXY'X_1Y_1$$

where X_1 and Y_1 are the maximal piece prefix and middle word respectively of some relation word, and Y' is a proper, non-empty prefix of Y. Clean overlap prefixes, in particular, will play a crucial role in what follows.

Proposition 1. *Let $aX_1Y_1'X_2Y_2'\dots X_nY_n$ be an overlap prefix of some word. Then this prefix contains no relation word as a factor, except possibly the suffix X_nY_n in the case that $Z_n = \epsilon$.*

Proof. Suppose that the given overlap prefix contains a relation word R as a factor. By the definition of an overlap prefix, no occurrence of R can begin before the end of the prefix a, so we may assume that R is a factor of $X_1Y_1'X_2Y_2'\dots X_nY_n$. It follows that either R contains X_iY_i' as a factor for some i, or else R is a proper factor of $X_iY_i'X_{i+1}Y_{i+1}'$ for some i (where $Y_{i+1}' = Y_n$ if $i+1 = n$) and we may assume without loss of generality that the occurrence of R overlaps non-trivially with the prefix X_iY_i'.

In the former case, since X_i is a maximal piece prefix of $X_iY_iZ_i$ and Y_i' is non-empty, X_iY_i' cannot be a piece; it follows then that we must have $R = X_iY_iZ_i$ with the occurrence in the obvious place. In the latter case, R is the product of a non-empty factor of $X_iY_iZ_i$ with a factor of the $X_{i+1}Y_{i+1}Z_{i+1}$; but by the small overlap assumption, R cannot be written as a product of two pieces, so it must again be that $R = X_iY_iZ_i$ with the occurrence in the obvious place.

Now if $i = n$ then, since R is a factor of the given relation prefix, we must clearly have $R = X_iY_iZ_i = X_iY_i$ so that $Z_i = \epsilon$. On the other hand, if $i < n$ then either $X_iY_iZ_i$ contains $X_{i+1}Y_{i+1}'$ as a factor which is not a prefix, which contradicts the fact that X_{i+1} is a maximal piece prefix of $X_iY_iZ_i$, or else (recalling that Y_i' is a proper prefix of Y_i) we see that $X_{i+1}Y_{i+1}'$ contains a non-empty suffix of Y_i followed by Z_i, which contradicts the fact that Z_i is a maximal piece suffix of $X_iY_iZ_i$. □

Proposition 2. *Let u be a word. Every overlap prefix of u is contained in a clean overlap prefix of u.*

Proof. If u has an overlap prefix, then it is contained in the longest overlap prefix of u. Let $aX_1Y_1'\dots X_nY_n$ be the longest overlap prefix, and suppose for a contradiction that this is not clean. Then by definition there exist words X and Y, being the maximal piece prefix and the middle word respectively of some relation word, and a proper non-empty prefix Y_n' of Y_n such that $aX_1Y_1'\dots X_nY_n'XY$ is a prefix of u. But clearly this is also an overlap prefix of u which is strictly

longer than the original one, thus contradicting the assumption that the original one is of maximal length. □

Corollary 1. *If a word u has no clean overlap prefix, then it contains no relation word as a factor, and so if $u \equiv v$ then $u = v$.*

Proof. Suppose u has no clean overlap prefix. If u contained a relation word as a factor then clearly it would have a relation prefix, that is, a prefix of the form aX_RY_R for some relation word R. But by our observations above, the shortest relation prefix of u would be an overlap prefix, and so by Proposition 2, is contained in a clean overlap prefix of u. Thus, u contains no relation word as a factor. It follows easily that no relations can be applied to u, so the only word equivalent to u is u itself. □

Lemma 1. *If $u = wXYZu'$ with wXY a clean overlap prefix then $w\overline{XY}$ is a clean overlap prefix of $w\overline{XYZ}u'$.*

Proof. Let

$$wXY = aX_1Y_1' \dots X_nY_n'XY \tag{1}$$

be the factorisation arising from definition of a clean overlap prefix. Then $w\overline{XYZ}u'$ has a prefix

$$w\overline{XY} = aX_1Y_1' \dots X_nY_n'\overline{X}\overline{Y}. \tag{2}$$

We claim first that $w\overline{XY}$ is an overlap prefix; to show this, the only condition we need to check is that there is no factor of the form X_0Y_0 beginning before the end of the prefix a. If $n \geqslant 1$ then, since such a factor cannot contain X_1Y_1' which is not a piece, any such factor would also arise in the corresponding place in wXY, contradicting the fact that wXY is an overlap prefix. Suppose, then, that $n = 0$. Recalling that Y_0 is not a piece, and so cannot be a factor of $\overline{XY}$, we see that $a\overline{XY}$ admits a factorisation

$$a\overline{XY} = bX_0Y_0'\overline{XY} \tag{3}$$

for some non-empty prefix Y_0' of Y_0. Moreover, Y_0' must be a proper prefix of Y_0, or else a would have a factor X_0Y_0, contradicting the fact that wXY was a clean overlap prefix of u. This completes the proof that $w\overline{X}\overline{Y} = bX_0Y_0'\overline{XY}$ is an overlap prefix of $w\overline{XYZ}u'$.

It remains to show that the given overlap prefix is clean. Suppose for a contradiction that it is not. Then by definition, there is a factor of the form $\hat{X}\hat{Y}$ overlapping the end of the prefix $a\overline{XY}$; but this factor is either by contained in $\overline{XYZ}$ (contradicting the supposition that $\hat{X}$ is a maximal piece prefix of the relation word $\hat{X}\hat{Y}\hat{Z}$) or contains a non-empty suffix of $\overline{Y}$ followed by $\overline{Z}$ (contradicting the assumption that $\overline{Z}$ is a maximal piece suffix of $\overline{XYZ}$). □

The following lemma is fundamental to our approach to $C(4)$ monoids. With careful application it seems to permit a comparable understanding to that resulting from Remmers' geometric theory, but in a purely combinatorial (and hence more computationally orientated) way.

Lemma 2. *Suppose a word u has clean overlap prefix wXY. If $u \equiv v$ then v has overlap prefix either wXY or $w\overline{XY}$, and no relation word occurring as a factor of v overlaps this prefix, unless it is XYZ or $\overline{XYZ}$ as appropriate.*

Note that the lemma does *not* assert that wXY or $w\overline{XY}$ is a *clean* overlap prefix of v; if this were always the case then the theory of small overlap monoids would be substantially simpler.

Proof. Since wXY is an overlap prefix of u, it has by definition a factorisation

$$wXY = aX_1Y_1' \dots X_nY_n'XY$$

for some $n \geqslant 0$. We use this fact to prove the claim by induction on the length r of a rewrite sequence (using the defining relations) from u to v.

In the case $r = 0$, we have $u = v$, so v certainly has (clean) overlap prefix vXY. By Proposition 1, no relation word factor can occur entirely within this prefix, unless it is the suffix XY and $Z = \epsilon$. If a relation word factor of v overlaps the end of the given overlap prefix and entirely contains XY then, since XY is not a piece, that relation word must clearly be XYZ. Finally, a relation word cannot overlap the end of the given overlap prefix but not contain the suffix XY, since this would clearly contradicts the fact that the given overlap prefix is clean.

Suppose now for induction that the lemma holds for all values less than r, and that there is a rewrite sequence from u to v of length r. Let u_1 be the second term in the sequence, so that u_1 is obtained from u by a single rewrite using the defining relations, and v from u_1 by $r-1$ rewrites.

Consider the relation word in u which is to be rewritten in order to obtain u_1, and in particular its position in u. By Proposition 1, this relation word cannot be contained in the clean overlap prefix wXY, unless it is XY where $Z = \epsilon$.

Suppose first that the relation word to be rewritten contains the final factor Y of the given clean overlap prefix. (Note that this covers in particular the case that the relation word is XY and $Z = \epsilon$.) From the $C(4)$ assumption we know that Y is not a piece, so we may deduce that the relation word is XYZ contained in the obvious place. In this case, applying the rewrite clearly leaves u_1 with a prefix $w\overline{XY}$, and by Lemma 1, this is a clean overlap prefix. Now v can be obtained from u_1 by $r-1$ rewrite steps, so it follows from the inductive hypothesis that v has overlap prefix either $w\overline{XY}$ or $w\overline{\overline{XY}} = wXY$, and that no relation word occurring as a factor of v overlaps this prefix, unless it is XYZ or $\overline{XYZ}$ as appropriate; this completes the proof in this case.

Next, we consider the case in which the relation word factor in u to be rewritten does not contain the final factor Y of the clean overlap prefix, but does overlap with the end of the clean overlap prefix. Then u has a factor of the form $\hat{X}\hat{Y}$, where $\hat{X}$ is the maximal piece prefix and $\hat{Y}$ the middle word of a relation word, which overlaps XY, beginning after the start of Y. This clearly contradicts the assumption that the overlap prefix is clean.

Finally, we consider the case in which the relation word factor in u which is to be rewritten does not overlap the given clean overlap prefix at all. Then obviously, the given clean overlap prefix of u remains an overlap prefix of u_1. If this overlap prefix is clean, then a simple application of the inductive hypothesis again suffices to prove that v has the required property.

There remains, then, only the case in which the given overlap prefix is no longer clean in u_1. Then by definition there exist words $\hat{X}$ and $\hat{Y}$, being a maximal piece prefix and middle word respectively of some relation word, such that u_1 has the prefix

$$aX_1Y_1' \dots X_nY_n'XY'\hat{X}\hat{Y}$$

for some proper, non-empty prefix Y' of Y. Now certainly this is not a prefix of u, since this would contradict the assumption that $aX_1Y_1' \dots X_nY_n'XY$ is a clean overlap prefix of u. So we deduce that u_1 must contain a relation word overlapping the final $\hat{X}\hat{Y}$. This relation word factor cannot contain the entire of this factor $\hat{X}\hat{Y}$, since then it would overlap with the prefix $aX_1Y_1' \dots X_nY_nXY$, which would again contradict the assumption that this prefix is a clean overlap prefix of u. Nor can the relation word contain the final factor $\hat{Y}$, since $\hat{Y}$ is not a piece. Hence, u_1 must have a prefix

$$aX_1Y_1' \dots X_{n-1}Y_{n-1}'X_nY_n'XY'\hat{X}\hat{Y}'R$$

for some relation word and proper, non-empty prefix $\hat{Y}'$ of $\hat{Y}$ and some relation word R. Suppose $R = X_RY_RZ_R$ where X_R and Z_R are the maximal piece prefix and suffix respectively. Then it is readily

verified that

$$aX_1Y_1'\dots X_{n-1}Y_{n-1}'X_nY_n'XY'\hat{X}\hat{Y}'X_RY_R$$

is a clean overlap prefix of u_1. But now by the inductive hypothesis, v has prefix either

$$aX_1Y_1'\dots X_{n-1}Y_{n-1}'X_nY_n'XY'\hat{X}\hat{Y}'X_RY_R \tag{4}$$

or

$$aX_1Y_1'\dots X_{n-1}Y_{n-1}'X_nY_n'XY'\hat{X}\hat{Y}'\overline{X_RY_R}. \tag{5}$$

In both cases v has prefix

$$aX_1Y_1'\dots X_{n-1}Y_{n-1}'X_nY_n'XY'\hat{X}\hat{Y}'$$

which in turn has prefix

$$aX_1Y_1'\dots X_{n-1}Y_{n-1}'X_nY_n'XY. \tag{6}$$

Moreover, by Proposition 1, the prefix (4) or (5) of v contains no relation word as a factor, unless it is the final factor X_RY_R [$\overline{X_RY_R}$] and $Z_R=\epsilon$ [respectively, $\overline{Z_R}=\epsilon$], and it follows easily that no relation word factor overlaps the prefix (6) of v. □

The lemma has the following easy corollary.

Corollary 2. *Suppose a word u has (not necessarily clean) overlap prefix wXY. If $u\equiv v$ then v has a prefix w and contains no relation word overlapping this prefix.*

Proof. By Proposition 2 the overlap prefix wXY of u is contained in a clean overlap prefix $w'X'Y'$ of u. Now by Lemma 2, v has a prefix w' and contains no relation word overlapping this prefix. But it is easily seen that w' must be at least as long as w, so that v has prefix w and contains no relation word overlapping this prefix, as required. □

The following proposition, which can be viewed as giving a very weak left cancellativity property of small overlap monoids, will allow us to restrict attention to words with a prefix of the form XY where X and Y are the maximal piece prefix and middle word respectively of some relation word.

Proposition 3. *Suppose a word u has an overlap prefix aXY and that $u=aXYu''$. Then $u\equiv v$ if and only if $v=av'$ where $v'\equiv XYu''$.*

Proof. Clearly if $v=av'$ with $v'\equiv XYu''$ then it is immediate that $v=av'\equiv aXYu''=v$.

Conversely, suppose $u\equiv v$. Since aXY is an overlap prefix, by Corollary 2 (applied with u in place of v) it cannot contain a relation word starting before the end of a. By Corollary 2 again, v has prefix a, say $v=av'$. Now consider a rewrite sequence, using the defining relations, from u to v. Again using Corollary 2, every term in this sequence will have prefix a, and contain no relation word overlapping this prefix. It follows that the same sequence of rewriting rules can be applied to take XYu'' to v', so that $v'\equiv XYu''$ as required. □

We now introduce some more terminology. Let u be a word and p be a piece. We say that u is *p-active* if pu has a relation prefix aXY with $|a|<|p|$, and *p-inactive* otherwise. The following proposition can be seen as describing another weak cancellativity property of small overlap monoids.

Proposition 4. *Let u be a word and p a piece. If u is p-inactive then $pu \equiv v$ if and only if $v = pw$ for some w with $u \equiv w$.*

Proof. If pu contains no relation prefix then it contains no relation word, so $pu \equiv v$ if and only if $pu = v$ and the claim certainly holds. Otherwise suppose pu has shortest relation prefix $paXY$, so that u has shortest relation prefix aXY. Let $u = aXYu''$. If $pu \equiv v$ then by Proposition 3 (since the shortest relation prefix is clearly an overlap prefix), we have $v = pav'$ where $v' \equiv XYu''$. Now setting $w = av'$ we have $v = pw$ and $u = aXYu' \equiv av' = aw$. The converse implication is obvious. □

Proposition 5. *Let p_1 and p_2 be pieces and suppose u is p_1-active and p_2-active. Then p_1 and p_2 have a common non-empty suffix, and if z is their maximal common suffix then*

(i) *u is z-active*;
(ii) *$p_1 u \equiv v$ if and only if $v = z_1 v'$ where $z_1 z = p_1$ and $v' \equiv zu$; and*
(iii) *$p_2 u \equiv v$ if and only if $v = z_2 v'$ where $z_2 z = p_2$; and $v' \equiv zu$.*

Proof. Let bX_1Y_1 and cX_2Y_2 be the shortest relation prefixes of p_1u and p_2v respectively. Since u is p_1-active and p_2-active, we must have $|b| < |p_1|$ and $|c| < |p_2|$. Moreover, since p_1 is a piece and X_1 is a maximal piece prefix of the relation word $X_1Y_1Z_1$ we must have $|p_1| \leqslant |bX_1|$, and similarly $|p_2| \leqslant |cX_2|$.

It follows that u has prefixes $X_1'Y_1$ and $X_2'Y_2$ where X_1' and X_2' are proper (perhaps empty) suffixes of X_1 and X_2 respectively. Thus, one of $X_1'Y_1$ and $X_2'Y_2$ is a prefix of the other, and so either Y_1 is a factor of $X_2'Y_2$ and hence of $X_2Y_2Z_2$ or Y_2 is a factor of $X_1'Y_1$ and hence of $X_1Y_1Z_1$. But by the $C(4)$ assumption, neither Y_1 nor Y_2 is a piece so the only possible explanation is that $X_1Y_1Z_1$ and $X_2Y_2Z_2$ are the same relation word, and moreover $X_1' = X_2'$.

Now let p be such that $pX_1' = X_1$. We have already observed that X_1' is a proper prefix of X_1, so p is non-empty. Now $p_1 = bp$, and also

$$pX_2' = pX_1' = X_1 = X_2$$

so by symmetry we have $p_2 = cp$. Hence, p is a common non-empty suffix of p_1 and p_2.

Now let z be the maximal common suffix of p_1 and p_2. Let y, z_1 and z_2 be such that $z = yp$, $p_1 = z_1 z$ and $p_2 = z_2 z$. Then clearly $b = z_1 y$ and $c = z_2 y$. Now $zu = ypu$ has a relation prefix yX_1Y_1, from which it is immediate that u is z-active so that (i) holds.

To show that (ii) holds, let u' be such that $u = X_1'Y_1u'$, and suppose $p_1 u \equiv v$. Now

$$p_1 u = z_1 z X_1' Y_1 u' = z_1 ypX_1' Y_1 u' = z_1 yX_1 Y_1 u'$$

where $z_1 yX_1Y_1$ is the shortest relation prefix, and hence is an overlap prefix. Hence, by Proposition 3 we have $v = z_1 yv''$ where $v'' \equiv X_1Y_1u'$. But now setting $v' = yv''$ we have $v = z_1 v'$, $z_1 z = p_1$ and

$$v' = yv'' \equiv yX_1Y_1u' = ypX_1'Y_1u' = zX_1'Y_1u' = zu$$

as required. Conversely, if $v = z_1 v'$ where $z_1 z = p_1$ and $v' \equiv zu$ then we have

$$p_1 u = z_1 zu \equiv z_1 v' = v.$$

This completes the proof that (ii) holds, and an entirely symmetric argument shows that (iii) holds. □

We remark that the second paragraph of the proof of Proposition 5 is probably the only place in which we make fundamental use of the fact that our monoid satisfies $C(4)$, rather than the weaker condition $C(3)$ assumed in the work of Remmers [11].

Corollary 3. *Let p_1 and p_2 be pieces. Suppose $p_1u \equiv p_1v$ and u is p_2-active. Then $p_2u \equiv p_2v$.*

Proof. If u is p_1-inactive then by Proposition 4 we have $u \equiv v$, and so certainly $p_2u \equiv p_2v$.

Suppose now that u is p_1-active. Let z be the maximal common suffix of p_1 and p_2 and let z_1 and z_2 be such that $z_1z = p_1$ and $z_2z = p_2$. Then by Proposition 5(ii), since $p_1u \equiv p_1v$ we have $p_1v = z_1v'$ where $v' \equiv zu$. But from $z_1zv = p_1v = z_1v'$ we deduce that $v' = zv$, so now we have

$$p_2u = z_2zu \equiv z_2v' = z_2zv = p_2v. \qquad \square$$

Corollary 4. *Let u and v be words and p_1 and p_2 be pieces. Suppose there exist words $u = u_1, \ldots, u_n = v$ such that*

$$p_1u_1 \equiv p_1u_2, \quad p_2u_2 \equiv p_2u_3, \quad p_1u_3 \equiv p_1u_4, \quad \ldots$$
$$\ldots, \quad \begin{cases} p_1u_{n-1} \equiv p_1u_n & \text{if } n \text{ is even,} \\ p_2u_{n-1} \equiv p_2u_n & \text{if } n \text{ is odd.} \end{cases}$$

Then either $p_1u \equiv p_1v$ or $p_1u \equiv p_2v$.

Proof. Fix u and v, and suppose n is minimal (allowing exchanging p_1 and p_2 if necessary) such that a sequence of equivalences as above exists. Suppose further for a contradiction that $n > 2$.

If u_2 was p_2-inactive then by Proposition 4 we would have $u_2 \equiv u_3$ so that $p_1u_1 \equiv p_1u_2 \equiv p_1u_3$ which contradicts the minimality assumption on n. Thus, u_2 is p_2-active. But now since $p_1u_1 \equiv p_1u_2$, we apply Corollary 3 to see that $p_2u_1 \equiv p_2u_2 \equiv p_2u_3$, again providing the required contradiction. $\square$

3. Sequential characterisation of equality

In this section we use the theory developed in Section 2 to provide a new characterisation of when two words in the generators of a small overlap presentation represent the same element of the monoid presented. In Section 4 we shall use this characterisation to develop an efficient algorithm to solve the word problem.

We first present a lemma which gives a set of mutually exclusive combinatorial conditions, the disjunction of which is necessary and sufficient for two words of a certain form to represent the same element.

Lemma 3. *Suppose $u = XYu'$ where XY is a clean overlap prefix of u. Then $u \equiv v$ if and only if one of the following mutually exclusive conditions holds*:

(1) $u = XYZu''$ *and* $v = XYZv''$ *and either* $Zu'' \equiv Zv''$ *or* $\overline{Z}u'' \equiv \overline{Z}v''$;
(2) $u = XYu'$, $v = XYv'$, *and Z fails to be a prefix of at least one of u' and v', and* $u' \equiv v'$;
(3) $u = XYZu''$, $v = \overline{X}\overline{Y}\overline{Z}v''$ *and either* $Zu'' \equiv Zv''$ *or* $\overline{Z}u'' \equiv \overline{Z}v''$;
(4) $u = XYu'$, $v = \overline{X}\overline{Y}\overline{Z}v''$ *but Z is not a prefix of u' and* $u' \equiv Zv''$;
(5) $u = XYZu''$, $v = \overline{X}\overline{Y}v'$ *but $\overline{Z}$ is not a prefix of v' and* $\overline{Z}u'' \equiv v'$;
(6) $u = XYu'$, $v = \overline{X}\overline{Y}v'$, *$Z$ is not a prefix of u' and $\overline{Z}$ is not a prefix of v', but $Z = z_1z$, $\overline{Z} = z_2z$, $u' = z_1u''$, $v' = z_2v''$ where $u'' \equiv v''$ and z is the maximal common suffix of Z and $\overline{Z}$, z is non-empty, and z is a possible prefix of u''.*

Proof. First we treat the claim that the conditions (1)–(6) are mutually exclusive. Since X is a maximal piece prefix of XYZ and Y is non-empty, XY is not a piece. An entirely similar argument shows that $\overline{XY}$ is not a piece. In particular, neither of XY and $\overline{XY}$ is a prefix of the other, and so v can have at most one of them as a prefix. Thus, conditions (1)–(2) are not consistent with conditions (3)–(6). The mutual exclusivity of (1) and (2) is self-evident from the definitions, and likewise that of (3)–(6).

It is easily verified that each of the conditions (1)–(5) imply that $u \equiv v$. We show next that (6) implies that $u \equiv v$. Since z is a possible prefix of u'' and $u'' \equiv v''$, we may write $u'' \equiv zx \equiv v''$ for some word x. Now we have

$$\begin{aligned} u = XYu' = XYz_1u'' \equiv XYz_1zx = XYZx \\ \equiv \overline{XYZ}x = \overline{XY}z_2zx \equiv \overline{XY}z_2v'' = \overline{XY}v' = v. \end{aligned}$$

What remains, which is the main burden of the proof, is to prove that $u \equiv v$ implies that at least one of the conditions (1)–(6) holds. To this end, suppose $u \equiv v$; then there is a rewriting sequence taking u to v. By Lemma 2, every term in this sequence will have prefix either XY or $\overline{XY}$ and this prefix can only be modified by the application of the relation $(XYZ, \overline{XYZ})$ in the obvious place. We now prove the claim by case analysis.

By Lemma 2, v begins either with XY or with $\overline{XY}$. Consider first the case in which v begins with XY; we split this into two further cases depending on whether u and v both begin with the full relation word XYZ; these will correspond respectively to conditions (1) and (2) in the statement of the lemma.

Case (1). Suppose $u = XYZu''$ and $v = XYZv''$. Then clearly there is a rewriting sequence taking u to v which by Lemma 2 can be broken up as:

$$\begin{aligned} u = XYZu'' \to^* XYZu_1 \to \overline{XYZ}u_1 \to^* \overline{XYZ}u_2 \\ \to XYZu_2 \to^* \cdots \to XYZu_n \to^* XYZv'' = v \end{aligned}$$

where none of the steps in the sequences indicated by $\to^*$ involves rewriting a relation word overlapping with the prefix XY or $\overline{XY}$ as appropriate. It follows that there are rewriting sequences.

$$Zu'' \to^* Zu_1, \quad \overline{Z}u_1 \to^* \overline{Z}u_2, \quad Zu_2 \to^* Zu_3, \quad \ldots, \quad Zu_n \to^* Zv''.$$

Now by Corollary 4, either $Zu'' \equiv Zv''$ or $\overline{Z}u'' \equiv \overline{Z}v''$ as required to show that condition (1) holds.

Case (2). Suppose now that $u = XYu'$, $v = XYv'$ and Z fails to be a prefix of at least one of u' and v'. We must show that $u' \equiv v'$; suppose for a contradiction that this does not hold. We consider only the case that Z is not a prefix of u'; the case that Z is not a prefix of v' is symmetric. We consider rewriting sequences from $u = XYu'$ to $v = XYv'$. Again using Lemma 2, we see that there is either (i) such a sequence taking u to v containing no rewrites of relation words overlapping the prefix XY, or (ii) such a sequence taking u to v which can be broken up as:

$$\begin{aligned} u = XYu' \to^* XYZu_1 \to \overline{XYZ}u_1 \to^* \overline{XYZ}u_2 \\ \to XYZu_2 \to^* \cdots \to XYZu_n \to^* XYv' = v \end{aligned}$$

where none of the intermediate words in the sequences indicated by $\to^*$ contains a relation word overlapping with the prefix XY or $\overline{XY}$ as appropriate. In case (i) there is clearly a rewrite sequence taking u' to v' so that $u' \equiv v'$ as required. In case (ii), there are rewriting sequences.

$$u' \to^* Zu_1, \quad \overline{Z}u_1 \to^* \overline{Z}u_2, \quad Zu_2 \to^* Zu_3, \quad \ldots, \quad Zu_n \to^* v'.$$

Notice that, since u' does not begin with Z, we can deduce from Proposition 4 that u_1 is Z-active. By Corollary 4, either $Zu_1 \equiv Zu_n$ or $\overline{Z}u_1 \equiv \overline{Z}u_n$. In the latter case, since u_1 is Z-active, Corollary 3 tells us that we also have $Zu_1 \equiv Zu_n$ in any case. But now

$$u' \equiv Zu_1 \equiv Zu_n \equiv v'$$

so condition (2) holds and we are done.

We have now shown that if v begins with XY then either condition (1) or condition (2) holds. It remains to consider the case in which v begins with $\overline{XY}$, and show that one of conditions (1)–(6) must be satisfied. We split the analysis here into four cases depending on whether u begins with the full relation word XYZ, and whether v begins with the full relation word $\overline{XYZ}$; these four cases will correspond respectively to conditions (3)–(6) in the statement of the lemma.

Case (3). Suppose $u = XYZu''$ and $v = \overline{XYZ}v''$. Then $u = XYZu'' \equiv v \equiv XYZv''$, so by the same argument as in case (1) we have either $Zu'' \equiv Zv''$ or $\overline{Z}u'' \equiv \overline{Z}v''$ as required to show that condition (3) holds.

Case (4). Suppose $u = XYu'$ and $v = \overline{XYZ}v''$ but Z is not a prefix of u'. Then $u = XYu' \equiv v \equiv XYZv''$. Now applying the same argument as in case (2) (with $XYZv''$ in place of v and setting $v' = Zv''$) we have $u' \equiv v' = Zv''$ so that condition (4) holds.

Case (5). Suppose $u = XYZu''$, $v = \overline{XY}v'$ but $\overline{Z}$ is not a prefix of v'. Then we have $\overline{XYZ}u'' \equiv u \equiv v = \overline{XY}v'$. Now applying the same argument as in case (1) (but with $\overline{XYZ}u''$ in place of u and setting $u' = \overline{Z}u''$) we obtain $u' \equiv v' = \overline{Z}u''$ so that condition (5) holds.

Case (6). Suppose $u = XYu'$, $v = \overline{XY}v'$ and that Z is not a prefix of u' and $\overline{Z}$ is not a prefix of v'. It follows this time there is a rewriting sequence taking u to v of the form

$$\begin{aligned} u = XYu' &\to^* XYZu_1 \to \overline{XYZ}u_1 \to^* \overline{XYZ}u_2 \to XYZu_2 \\ &\to^* \cdots \to \overline{XYZ}u_n \to^* \overline{XY}v' = v \end{aligned}$$

where once more by Lemma 2 none of the intermediate words in the sequences indicated by $\to^*$ contains a relation word overlapping with the prefix XY or $\overline{XY}$ as appropriate. Now there are rewriting sequences.

$$u' \to^* Zu_1, \quad \overline{Z}u_1 \to^* \overline{Z}u_2, \quad Zu_2 \to^* Zu_3, \quad \ldots, \quad Zu_{n-1} \to^* Zu_n, \quad \overline{Z}u_n \to^* v'.$$

Notice that, since u' does not begin with Z, we may deduce from Proposition 4 that u_1 is Z-active. By Corollary 4, either $Zu_1 \equiv Zu_n$ or $\overline{Z}u_1 \equiv \overline{Z}u_n$. In the latter case, since u_1 is Z-active, Corollary 3 tells us that we also have $Zu_1 \equiv Zu_n$ anyway. But now

$$u' \equiv Zu_1 \equiv Zu_n$$

where u' does not begin with Z, and also $v' \equiv \overline{Z}u_n$ were v' does not begin with $\overline{Z}$. By applying Proposition 4 twice, we deduce that u_n is both Z-active and $\overline{Z}$-active.

Let z be the maximal common suffix of Z and $\overline{Z}$. Then applying Proposition 5 (with $p_1 = Z$ and $p_2 = \overline{Z}$), we see that z is non-empty and

- $u' = z_1u''$ where $Z = z_1z$ and $u'' \equiv zu_n$; and
- $v' = z_2v''$ where $\overline{Z} = z_2z$ and $v'' \equiv zu_n$.

But then we have $u'' \equiv zu_n \equiv v''$ and also z is a possible prefix of u'' as required to show that condition (6) holds. □

Lemma 3 gives a first clue as to how one might solve the word problem for a small overlap monoid by analysing words sequentially from left to right. The natural strategy is as follows. First, use Proposition 3 to reduce to the case in which the words both have clean relation prefixes of the form XY or $\overline{XY}$. Now by examining short prefixes, one can clearly always rule out at least five of the six mutually exclusive conditions of the lemma. The remaining condition will involve equivalence of

words derived from suffixes of u and v, so apply the same approach recursively to test whether this condition is satisfied.

This approach meets with several apparent obstacles. Firstly, it is not clear that the words derived from the suffixes of u and v, which must be tested for equivalence in the recursive call, are shorter than the original words u and v; for example, a relation word XYZ may be shorter than the maximal piece suffix $\overline{Z}$ of the word on the other side of the relation. In fact the recursive call will not always involve shorter words, but it will involve words which are simpler in a more subtle sense, so that the algorithm still terminates rapidly. Secondly, some of the conditions involve a disjunction of equivalence of *two* pairs of words derived from the suffixes; testing both would require two recursive calls, potentially leading to exponential time complexity. It transpires, though, that the theory of activity and inactivity developed in Section 2, and in particular Corollary 3, guarantees that making just one of these recursive calls will is always sufficient. Finally, condition (6) requires us to check whether a certain piece is a possible prefix of a given word, and it is not obvious how this can be efficiently achieved; this problem is solved by the following development of Lemma 3, which gives simultaneous conditions for two words to be equal, and to admit a given piece as a possible prefix.

Lemma 4. *Suppose $u = XYu'$ where XY is a clean overlap prefix, and suppose p is a piece. Then $u \equiv v$ and p is a possible prefix of u if and only if one of the following mutually exclusive conditions holds*:

(1′) $u = XYZu''$ *and* $v = XYZv''$, *either* $Zu'' \equiv Zv''$ *or* $\overline{Z}u'' \equiv \overline{Z}v''$, *and also p is a prefix of either X or $\overline{X}$*;
(2′) $u = XYu'$, $v = XYv'$, *and Z fails to be a prefix of at least one of u' and v', and $u' \equiv v'$, and also either*
  – *p is a prefix of X; or*
  – *p is a prefix of $\overline{X}$ and Z is a possible prefix of u'.*
(3′) $u = XYZu''$, $v = \overline{X}\overline{Y}\overline{Z}v''$ *and either* $Zu'' \equiv Zv''$ *or* $\overline{Z}u'' \equiv \overline{Z}v''$, *and also p is a prefix of X or $\overline{X}$*;
(4′) $u = XYu'$, $v = \overline{X}\overline{Y}\overline{Z}v''$ *but Z is not a prefix of u' and $u' \equiv Zv''$, and also p is a prefix of X or $\overline{X}$*;
(5′) $u = XYZu''$, $v = \overline{X}\overline{Y}v'$ *but $\overline{Z}$ is not a prefix of v' and $\overline{Z}u'' \equiv v'$, and also p is a prefix of X or $\overline{X}$*;
(6′) $u = XYu'$, $v = \overline{X}\overline{Y}v'$, *$Z$ is not a prefix of u' and $\overline{Z}$ is not a prefix of v', but $Z = z_1z$, $\overline{Z} = z_2z$, $u' = z_1u''$, $v' = z_2v''$ where $u'' \equiv v''$, z is the maximal common suffix of Z and $\overline{Z}$, z in non-empty, z is a possible prefix of u'', and also p is a prefix of X or $\overline{X}$.*

Proof. Mutual exclusivity of the six conditions is proved exactly as for Lemma 3.

Suppose now that one of the six conditions above applies. Each condition clearly implies the corresponding condition from Lemma 3, so we deduce immediately that $u \equiv v$. We must show, using the fact that p is a prefix of X or of $\overline{X}$, that p is a possible prefix of u, or equivalently of v.

In case (1′), if p is a prefix of X then it is a prefix of u, while if p is a prefix of $\overline{X}$ then it is a prefix of $\overline{XYZ}u''$ which is clearly equivalent to u. In case (2′), if p is a prefix of X then it is again a prefix of u, while if p is a prefix of $\overline{X}$ and Z is a possible prefix of u', say $u' \equiv Zw$, then

$$u = XYu' \equiv XYZw \equiv \overline{XYZ}w$$

where the latter has p as a prefix. In the remaining cases u begins with X and v begins with $\overline{X}$, so p is a prefix of either u or v, and hence a possible prefix of u.

Conversely, suppose $u \equiv v$ and p is a possible prefix of u. Then exactly one of the six conditions in Lemma 3 applies. By Lemma 2, every word equivalent to u begins with either XY or $\overline{XY}$. Since p is a piece, X is the maximal piece prefix of XYZ, and $\overline{X}$ is the maximal piece prefix of $\overline{XYZ}$ it follows that p is a prefix of either X or $\overline{X}$. If any but condition (2) of Lemma 3 is satisfied, this suffices to show that the corresponding condition from the statement of Lemma 4 holds.

If condition (2) from Lemma 3 applies, we must show additionally that either p is a prefix of X, or p is a prefix of $\overline{X}$ and Z is a possible prefix of u'. Suppose p is not a prefix of X. Then by the above, p is a prefix of $\overline{X}$. It follows from Lemma 2, that the only way the prefix XY of the word u can be changed using the defining relations is by application of the relation $(XYZ, \overline{XYZ})$. In order for this to happen, one must clearly be able to rewrite $u = XYu'$ to a word of the form $XYZw$; consider the shortest possible rewriting sequence which achieves this. By Lemma 2, no term in the sequence

```
WP-Prefix(u, v, p)
 1  if u = ϵ or v = ϵ
 2    then if u = ϵ and v = ϵ and p = ϵ
 3           then return Yes
 4           else return No
 5  elseif u does not have the form XYu′ with XY a clean overlap prefix
 6    then if u and v begin with different letters
 7           then return No
 8         elseif p ≠ ϵ and u and p begin with different letters
 9           then return No
10         else
11              u ← u with first letter deleted
12              v ← v with first letter deleted
13              if p ≠ ϵ
14                then p ← p with first letter deleted
15              return WP-Prefix(u, v, p)
16  else
17       let X, Y, u′ be such that u = XYu′
18       if p is a prefix of neither X nor X̄
19         then return No
20       elseif v does not begin either with XY or with X̄Ȳ
21         then return No
22       elseif u = XYZu″ and v = XYZv″
23         then if u″ is Z-active
24                then return WP-Prefix(Zu″, Zv″, ϵ)
25                else return WP-Prefix(Z̄u″, Z̄v″, ϵ)
26       elseif u = XYu′ and v = XYv′
27         then if p is a prefix of X
28                then return WP-Prefix(u′, v′, ϵ)
29                else return WP-Prefix(u′, v′, Z)
30       elseif u = XYZu″ and v = X̄Ȳ Z̄v″
31         then if u″ is Z-active
32                then return WP-Prefix(Zu″, Zv″, ϵ)
33                else return WP-Prefix(Z̄u″, Z̄v″, ϵ)
34       elseif u = XYu′ and v = X̄Ȳ Z̄v″
35         then return WP-Prefix(u′, Zv″, ϵ)
36       elseif u = XYZu″ and v = X̄Ȳ v′
37         then return WP-Prefix(Z̄u″, v′, ϵ)
38       elseif u = XYu′ and v = X̄Ȳ v′
39         then let z be the maximal common suffix of Z and Z̄
40           let z1 be such that Z = z1z
41           let z2 be such that Z̄ = z2z
42           if u′ does not begin with z1 or v′ does not begin with z2;
43             then return NO
44             else let u″ be such that u′ := z1u″
45               let v″ be such that v′ := z2v″;
46               return WP-Prefix(u″, v″, z)
```

Fig. 1. Algorithm for the word problem.

except for the last term will contain a relation word overlapping the initial XY. It follows that the same rewriting steps rewrite u' to Zw, so that Z is a possible prefix of u', as required. □

4. The algorithm

In this section we present an algorithm, for a fixed monoid presentation satisfying $C(4)$, which takes as input arbitrary words u and v and a piece p, and decides whether $u \equiv v$ and p is a possible prefix of u. It will transpire that this algorithm can be implemented (on a 2-tape Turing machine) to run in time in linear in the shorter of u and v. In particular, by setting $p = \epsilon$ we obtain an algorithm to solve the word problem in time linear in the smaller of the input words. The algorithm is shown (in recursive/functional pseudocode) in Fig. 1. Our first objective is to prove the correctness of the algorithm, that is, that whenever the algorithm terminates, the output it gives is correct.

Lemma 5. *Suppose u and v are words and p a piece. Then the algorithm* WP-PREFIX(u, v, p)

- *outputs* YES *only if $u \equiv v$ and p is a possible prefix of u; and*
- *outputs* NO *only if $u \not\equiv v$ or p is not a possible prefix of u.*

Proof. We prove correctness using induction on the number n of recursive calls.

Consider first the base case $n = 0$, that is, where the algorithm terminates without a recursive call. Suppose u, v and p are such that this happens. We consider each of the possible lines at which termination may occur, establishing in each case that the output produced is correct.

Line 3. If $u = \epsilon$, $v = \epsilon$ and $p = \epsilon$ then clearly $u \equiv v$ and p is a possible prefix of u, so the output YES is correct.

Line 4. If $u = \epsilon$ [respectively, $v = \epsilon$] then it follows easily from the small overlap condition $C(4)$ that no relations can be applied to u [v]; indeed a relation which could be applied to u [v] would have to have ϵ as one side, but ϵ is a piece and hence cannot be a relation word. Hence, we can have that $u \equiv v$ and p is a possible prefix of u only if $u = v = p = \epsilon$. In this case, this condition is not satisfied, so the output NO is correct.

Line 7. In this case, u does not begin with a clean overlap prefix of the form XY. So by Proposition 3, every word equivalent to u must begin with the same letter as u. Hence, if u and v do not begin with the same letter then we cannot have $u \equiv v$, so the output NO is correct.

Line 9. Again, u does not begin with a clean overlap prefix. If p is non-empty and begins with a different letter to u, then again by Proposition 3, p cannot be a possible prefix of u, so the output NO is correct.

Line 19. We are now in the case that u has a clean overlap prefix XY. If p is not a prefix of X or $\overline{X}$ then by Lemma 4 we see that p is not a possible prefix of u, so the output NO is correct.

Line 21. Once again, we are in the case that u has a clean overlap prefix XY. If v does not begin with either XY or $\overline{XY}$ then by Lemma 3 we cannot have $u \equiv v$ so the output NO is correct.

Line 43. We are now in the case that $u = XYu'$ and $v = \overline{XY}v'$ where Z is not a prefix of u' and $\overline{Z}$ is not a prefix of v'. We know also that z is the maximal common suffix of Z and $\overline{Z}$ and z_1 and z_2 are such that $Z = z_1 z$ and $\overline{Z} = z_2 z$. By Lemma 4 we cannot have $u \equiv v$ unless u' and v' have the form $z_1 u''$ and $z_2 v''$ respectively, so if this is not the case, the output NO is correct.

Now let $n > 0$ and suppose for induction that the algorithm produces the correct output whenever it terminates after strictly fewer than n recursive calls. Let u, v, p be such that the algorithm terminates after n recursive calls. This time, we consider each of the possible places at which the first recursive call can be made, establishing in each case that the output produced is correct.

Line 15. In this case u does not begin with a clean overlap prefix of the form XY and we have $u = au'$. It follows by Proposition 3 that every word equivalent to u has the form aw where $w \equiv u'$. In particular, $u \equiv v = av'$ if and only if $u' \equiv v'$, p is a possible prefix exactly if either $p = \epsilon$ or $p = ap'$ where p' is a possible prefix of u'. By the inductive hypothesis, the recursive call correctly establishes whether these conditions hold.

Line 24. We know that $u = XYZu''$, that $v = XYZv''$ and that p is a prefix of X or $\overline{X}$. By Lemma 4, it follows that $u \equiv v$ and p is a possible prefix of u if and only if $Zu'' \equiv Zv''$ or $\overline{Z}u'' \equiv \overline{Z}v''$. We also know that u'' is Z-active, so by Corollary 3, this is true if and only if $Zu'' \equiv Zv''$.

Line 25. This is the same as the previous case, except that u'' is not Z-active. In this case, by Proposition 4 we have that $Zu'' \equiv Zv''$ implies $u'' \equiv v''$ which in turn implies $\overline{Z}u'' \equiv \overline{Z}v''$, so it suffices to test the latter.

Line 28. Here we know that $u = XYu'$, $v = XYv'$, that Z is not a prefix of u' or v' and that p is a prefix of X. It follows by Lemma 4 that $u \equiv v$ and p is a possible prefix of u if and only if $u' \equiv v'$.

Line 29. This time we know that $u = XYu'$, $v = XYv'$ and that p is a prefix of $\overline{X}$ but not of X. It follows by Lemma 4 that $u \equiv v$ and p is a possible prefix of u if and only if $u' \equiv v'$ and Z is a possible prefix of u'.

Line 32. Here we have $u = XYZu''$ and $v = \overline{XYZ}v''$, and p is a prefix of X or $\overline{X}$. It follows by Lemma 4 that $u \equiv v$ and p is a possible prefix of u if and only if either $Zu'' \equiv Zv''$ or $\overline{Z}u'' \equiv \overline{Z}v''$. We also know that u'' is Z-active, so by Corollary 3, this is true if and only if $Zu'' \equiv Zv''$.

Line 33. This is the same as the previous case, except that u'' is not Z-active. In this case, by Proposition 4 we have that $Zu'' \equiv Zv''$ implies $u'' \equiv v''$ which in turn implies $\overline{Z}u'' \equiv \overline{Z}v''$, so it suffices to test the latter.

Line 35. If we get here, we know that $u = XYu'$, that $v = \overline{XYZ}v''$, that Z is not a prefix of u' and that p is a prefix of X or $\overline{X}$; it follows that $u \equiv v$ and p is a possible prefix of u if and only if condition $(4')$ of Lemma 4 holds, that is, if and only if $u' \equiv Zv''$. By the inductive hypothesis, the recursive call will correctly establish if this is the case.

Line 37. The argument here is symmetric to that for termination at line 4.

Line 46. Having got here, we know that p is a prefix of X or $\overline{X}$, that $u = XYu'$ and $v = \overline{XY}v'$ where Z is not a prefix of u' and $\overline{Z}$ is not a prefix of v'. We know also that z is the maximal common suffix of Z and $\overline{Z}$ and z_1 and z_2 are such that $Z = z_1 z$ and $\overline{Z} = z_2 z$. Finally, we know that $u' = z_1 u''$ and $v' = z_2 v''$. It follows by Lemma 4 that $u \equiv v$ and p is a possible prefix of z if and only if $u'' \equiv v''$ and z is a possible prefix of u''. By the inductive hypothesis, the recursive call correctly establishes whether this holds. □

We have now shown that our algorithm produces the correct output whenever it terminates, but we have not yet proved that it always terminates. In fact, the following lemma shows that it does so after a number of recursive calls bounded above by a linear function of the length of the first input word.

Lemma 6. *Let k be the length of the longest maximal piece suffix of a relation word. The number of recursive calls during execution of a call to* WP-PREFIX(u, v, p) *is bounded above by $(k+2)|u|$.*

Proof. For clarity in our analysis, we let u_i, v_i and p_i denote the parameters to the ith recursive call in the execution. We make the convention that the original (non-recursive) call is the "0th" recursive call, setting in particular $u_0 = u$, $v_0 = v$ and $p_0 = p$. Each call to the function involves executing exactly one of the sections 2–4, 6–15 and 17–46; we call these calls of type A, B and C respectively. We shall show that the number of calls of each of these types is bounded above by a linear function of $|u|$ so that, the total number of calls is also bounded above by a linear function of $|u|$.

First, notice that a call of type A cannot make a recursive call, so there is only at most one type A call in the execution.

Now for a word x we define $r(x) = 0$ if x does not have a clean overlap prefix, and $r(x)$ to be the length of the part of x which follows the shortest clean overlap prefix, that is, $|x'|$ where $x = aXYx'$ with aXY the shortest clean overlap prefix, otherwise.

It is readily verified that if the ith recursive call is of type B and itself makes a recursive call then we have $r(u_{i+1}) = r(u_i)$, while if the ith recursive call is of type C and itself makes a recursive call then we have $r(u_{i+1}) < r(u)$. Since $r(u_i)$ can never be negative, it follows that the total number of calls of type C is bounded above by $r(u_0) + 1$, which clearly is no more than $|u_0|$.

Now note that if the ith recursive call is of type B and itself makes a recursive call then we have $|u_{i+1}| = |u_i| - 1$, while if the ith recursive call is of type C and itself makes a recursive call then we have $r(u_{i+1}) \leqslant |u_i| + k$.

We have seen that the entire execution cannot feature more than $|u_0|$ calls of type C or more than 1 call of type A. Hence, if the execution involves i recursive calls (so $i + 1$ calls in total), it must include at most $|u_0|$ calls of type C, and at least $(i + 1) - |u_0| - 1 = i - |u_0|$ calls of type B. It follows that, if execution involves i recursive calls, we must have

$$|u_i| \leqslant |u_0| + |u_0|k - \big(i - |u_0|\big) = (k + 2)|u| - i.$$

Since the length of u_i cannot be negative, we must have $i \leqslant (k + 2)|u|$, that is, the execution cannot feature more than $(k + 2)|u|$ recursive calls. □

It remains to justify our claim that this algorithm can be implemented in linear time. Since the concept of linear time is highly dependent upon model of computation, it is necessary to be precise about the particular model under consideration. We consider a Turing machine with two-way-infinite read–write storage tapes, using a tape alphabet including the generators for our monoid and a separator symbol #. (Recall that a two-way-infinite tape can be simulated using a one-way-infinite tape in linear time [4, Section 7.5], so the assumption of a two-way-infinite tape is essentially immaterial.) If we assume that the input words u, v and p are initially encoded on one of the tapes in the form $\#u\#v\#p\#$, then it is easily seen that, with a linear amount of preprocessing, we can store the piece p in the finite state control, and arrange for $\#u\#$ and $\#v\#$ to be the content of the first and second tape respectively.

We shall need the following lemma.

Lemma 7. *Let $u \in A^*$, and let u' be a prefix of u of length at least twice the length of the longest relation word in the presentation. Then*

(i) *for any piece p, u is p-active if and only if u' is p-active*;
(ii) *for any relation word R, u has clean overlap prefix $X_R Y_R$ if and only if u' has clean overlap prefix $X_R Y_R$.*

Proof. (i) If u is p-active then pu has a factor of the form $X_R Y_R$ for some relation word R, which begins before the end of p. But $|u'| \geqslant |R| \geqslant |X_R Y_R|$, so this factor must end before the end of the prefix pu', and so is also factor of pu'. Thus, u' is Z-active. The converse is obvious.

(ii) Suppose u has clean overlap prefix $X_R Y_R$. We know that $|u'| \geqslant |R| \geqslant |X_R Y_R|$ so certainly $X_R Y_R$ is a prefix of u. Moreover, since u has no factor of the form $X'Y'$ beginning before the end of the prefix $X_R Y_R$, it is clear that u' also does not, so u' has clean overlap prefix $X_R Y_R$.

Conversely, if u' has a clean overlap prefix $X_R Y_R$ then certainly u has $X_R Y_R$ as a prefix. If this overlap prefix were not clean then u would have a factor of the form $X_{R'} Y_{R'}$ for some relation word R' beginning before the end of this prefix. But this would have to end before position $|X_R Y_R| + |X_{R'} Y_{R'}| \leqslant |R| + |R'| \leqslant |u'|$, and so would also occur in u', giving a contradiction. □

By Lemma 7 we can, given a word u, check whether u has a clean overlap prefix of the form XY, and if so find X, Y and the corresponding Z, by analysing a prefix of u of bounded length. Similarly, for a given maximal piece suffix Z, we can check whether u is Z-active by analysing a prefix of u of bounded length. It follows that each recursive step of our algorithm involves analysing prefixes of u and v of bounded length, before possibly making a recursive call, with u and v modified only by changing prefixes of bounded length. Clearly any analysis of a bounded length prefix can be performed in constant time; moreover, if a recursive call is required then the tape contents and finite

state control can be modified to contain the parameters for that call, again in constant time. It follows that the algorithm can be implemented with execution time bounded above by a linear function of the number of recursive calls in the execution, which by Lemma 6 is bounded above by a linear function of the length of u. Thus we obtain the following.

Theorem 1. *For every monoid presentation satisfying $C(4)$, there exists a two-tape Turing machine which solves the corresponding word problem in time linear in the lengths the input words.*

Note that, aside from the preprocessing required to place the input words on different tapes, the time taken to solve the word problem is linear in the length of the *first* input word u. Moreover, by applying a dual version of the algorithm if necessary, one may clearly make the time linear in the length of the *shorter* input word. The reader may initially be surprised by the idea of testing equivalence of two words in time bounded by a function of the shorter of the two—indeed, this bound potentially does not even afford time to fully read the longer word. The explanation can be found in the work of Remmers, where it is shown that, for a fixed $C(3)$ presentation, the length of the longer of two equivalent words is bounded by a linear function of the length of the shorter ([3, Theorem 5.2.14] and [11, Theorem 4.12]). Thus, if the difference in lengths of two words is too great, one may conclude without further analysis that the words are not equivalent. In fact Remmers' result is the only possible explanation for this phenomenon, so the fact that this property holds for $C(4)$ presentations can also be deduced from Theorem 1.

5. Uniform decision problems

In Section 4 we developed a linear time algorithm to solve the word problem for a fixed small overlap presentation. Since our method of describing the algorithm was entirely constructive, one might reasonably expect that it also gives rise to a solution for the uniform word problem for $C(4)$ presentations, that is the algorithmic problem of, given a $C(4)$ presentation and two words, deciding whether the words represent the same element of the monoid presented. In this section, we shall see that this is indeed the case, and show that the resulting algorithm remains fast.

To avoid unnecessary technicalities, we describe and analyse the algorithms using the RAM model of computation; in particular this allows us to assume that elementary operations involving generators from the presentation (such as comparing two generators) are single steps performable in constant time. The exact time complexity of a Turing machine implementation would depend upon the number of tapes and the precise encoding of the input, but would certainly remain polynomial of low degree in the input size.

We begin with some simple results describing the complexity of some elementary computations with a finite monoid presentation. If $\langle \mathscr{A} \mid \mathscr{R} \rangle$ is a finite presentation we denote by $|\mathscr{A}|$ the cardinality of the alphabet $\mathscr{A}$, and by $|\mathscr{R}|$ the sum length of the relation words in $\mathscr{R}$. Where the meaning is clear, we shall abuse notation by using $\mathscr{R}$ also to denote the set of relation words in the presentation.

Proposition 6. *There is a RAM algorithm which, given a presentation $\langle \mathscr{A} \mid \mathscr{R} \rangle$ and a word w, computes the maximum piece prefix (and/or maximum piece suffix) of w in time $O(|w||\mathscr{R}|)$. In particular, there is a RAM algorithm to decide, given the same input, whether the word w is a piece in time $O(|w||\mathscr{R}|)$.*

Proof. For each relation word $R \in \mathscr{R}$ and position $1 < i < |R|$ in that word we can compute in time $O(|w|)$ the length n of the longest common prefix of w and $R_i \dots R_{|R|}$ (where R_j represents the jth letter of R). Our machine does this for each relation word and each position in that relation word in turn, recording as it goes along (i) the maximum value of n attained so far, and (ii) the maximum value of n which has been attained or exceeded at least twice. The latter, upon completion, is clearly the length of the longest piece prefix of w, and the total time taken for execution is

$$O\left(\sum_{R\in\mathscr{R}}\sum_{i=1}^{|R|}|w|\right) = O\left(|w||\mathscr{R}|\right)$$

as claimed. An obvious dual algorithm can be used to find the longest piece suffix of w. □

Corollary 5. *There is a RAM algorithm which, given as input a presentation $\langle \mathscr{A} \mid \mathscr{R} \rangle$, decides in time $O(|\mathscr{R}|^2)$ whether the presentation satisfies the condition $C(4)$.*

Proof. Our machine begins by computing the maximum piece prefix X_R and maximum piece suffix Z_R for each relation word $R \in \mathscr{R}$; by Proposition 6 this can be done in time

$$O\Big(\sum_{R\in\mathscr{R}} |R||\mathscr{R}|\Big) = O\big(|\mathscr{R}|^2\big).$$

It then tests, in time $O(|\mathscr{R}|)$, whether for any of the relation words R we have $|X_R| + |Z_R| \geqslant |R|$. If so then some relation word is a product of two pieces, so the presentation does not even satisfy the weaker condition $C(3)$ and we are done.

Otherwise, the machine computes, again in time $O(|\mathscr{R}|)$, the middle word Y_R of each relation word. By our remarks in Section 1, the presentation satisfies $C(4)$ if and only if none of the words Y_R is a piece. Using Proposition 6 again, this condition can be tested in time

$$O\Big(\sum_{R\in\mathscr{R}} |Y_R||\mathscr{R}|\Big) = O\big(|\mathscr{R}|^2\big).$$

Thus, we have described a RAM algorithm to test a presentation $\langle \mathscr{A} \mid \mathscr{R} \rangle$ for the $C(4)$ condition in time $O(|\mathscr{R}|^2)$. □

Theorem 2. *There is a RAM algorithm which, given as input a $C(4)$ presentation $\langle \mathscr{A} \mid \mathscr{R} \rangle$ and two words $u, v \in \mathscr{A}^*$, decides whether u and v represent the same element of the semigroup presented in time*

$$O\big(|\mathscr{R}|^2 \min\big(|u|, |v|\big)\big).$$

Proof. Suppose we are given a $C(4)$ presentation $\langle \mathscr{A} \mid \mathscr{R} \rangle$ and two words $u, v \in \mathscr{A}^*$. Just as in the proof of Proposition 6, the machine begins by finding for every relation R the maximum piece prefix X_R, the maximum piece suffix Z_R and the middle word Y_R, in time $O(|\mathscr{R}|^2)$.

It now has the information required to apply the algorithm WP-PREFIX given above. A simple line-by-line analysis shows that each line, and hence each recursive call, can be executed in time $O(|\mathscr{R}|)$. By Lemma 6, the number of recursive calls is bounded above by $(k+2)|u|$ where k, being the length of the longest maximum piece suffix of a relation word, is less than $|\mathscr{R}|$. Thus, this part of the algorithm terminates in time $O(|\mathscr{R}|^2|u|)$.

As above we may assume, by exchanging u and v at the start of the computation if necessary, that $|u| < |v|$ so that $\min(|u|, |v|) = |u|$. It follows that the uniform word problem can be solved in time $O(|\mathscr{R}|^2 \min(|u|, |v|))$ as claimed. □

Acknowledgments

This research was supported by an RCUK Academic Fellowship. The author is grateful to V.N. Remeslennikov, whose questions prompted this line of research and who shared many helpful ideas. He would also like to thank A.V. Borovik for some helpful conversations, J.B. Fountain and V.A.R. Gould for facilitating access to some of the relevant literature, and Kirsty for all her support and encouragement.

References

[1] The GAP-Group, GAP—Groups, Algorithms, and Programming, Version 4.4, http://www.gap-system.org/, 2005.
[2] M. Gromov, Hyperbolic groups, in: Essays in Group Theory, in: Math. Sci. Res. Inst. Publ., vol. 8, Springer-Verlag, New York, 1987, pp. 75–263.

[3] P.M. Higgins, Techniques of Semigroup Theory, Oxford Sci. Publ., The Clarendon Press, Oxford University Press, New York, 1992, with a foreword by G.B. Preston.
[4] J.E. Hopcroft, J.D. Ullman, Formal Languages and their Relation to Automata, Addison–Wesley, 1969.
[5] M. Kambites, On generic properties of finitely presented monoids and semigroups, arXiv: 0807.1190 [math.RA], 2008.
[6] M. Kambites, Small overlap monoids II: Automatic structures and normal forms, arXiv: 0806.3891 [math.RA], 2008.
[7] M. Kambites, SmallOverlap—A GAP 4 package for small overlap monoids and semigroups, http://www.maths.manchester.ac.uk/~mkambites/, 2008.
[8] R.C. Lyndon, P.E. Schupp, Combinatorial Group Theory, Springer-Verlag, 1977.
[9] A.Yu. Ol'shanskiĭ, Almost every group is hyperbolic, Internat. J. Algebra Comput. 2 (1) (1992) 1–17.
[10] M. Pelletier, J. Sakarovitch, Easy multiplications II. Extensions of rational semigroups, Inform. and Comput. 88 (1990) 18–59.
[11] J.H. Remmers, Some algorithmic problems for semigroups: A geometric approach, PhD thesis, University of Michigan, 1971.
[12] J.H. Remmers, On the geometry of semigroup presentations, Adv. Math. 36 (3) (1980) 283–296.
[13] J. Sakarovitch, Easy multiplications I. The realm of Kleene's theorem, Inform. and Comput. 74 (1987) 173–197.
[14] V. Shpilrain, G. Zapata, Combinatorial group theory and public key cryptography, Appl. Algebra Engrg. Comm. Comput. 17 (3–4) (2006) 291–302.