Word problems of groups: formal languages, characterizations and decidability

Sam A. M. Jones and Richard M. Thomas

Department of Informatics, University of Leicester, Leicester LE1 7RH, UK

Abstract

Let G be a group and let $\varphi : \Sigma^* \to G$ be a monoid homomorphism from the set of all strings Σ^* over some finite alphabet Σ onto the group G. The set Σ is then called a generating set for G and the language $\{1\}\varphi^{-1} \subseteq \Sigma^*$ is called the word problem of G with respect to the generating set Σ (via the homomorphism φ) and is denoted by $W(G, \Sigma)$.

We consider nine conditions that hold in each such language of the form $W(G, \Sigma)$ and determine which combinations of these conditions are equivalent to the property of the language in question being the word problem of a group. We show that each of these nine conditions is decidable for the family of regular languages but that each is undecidable for the family of one-counter languages (the languages accepted by one-counter pushdown automata). We also show that the property of a language being the word problem of a group is undecidable for the family of one-counter languages but is decidable for the family of deterministic context-free languages (the languages accepted by deterministic pushdown automata).

1. Introduction

The word problem of a finitely generated group G is a fundamental notion in group theory. Whilst it has been traditionally thought of as the problem of determining whether or not a word in the generators of the group represents the identity element of G, it can also be defined as the set W of all the words in the generators of G that represent the identity element (so that the previous problem about representing the identity becomes the question of determining membership of the set W). This alternative approach allows us to consider such a word problem W as a formal language and a rich topic of research has been the connection betweens the complexity of this language W and the algebraic structure of the corresponding group G. For example, the groups with a regular word problem were classified in [1] and those with a context-free word problem in [23] (modulo a subsequent result in [7]).

We will focus on these two families of languages (the regular and context-free languages) in this paper along with the family of one-counter languages

(see Section 2 for a definition). This last family is particularly interesting when considering word problems of groups for the following reason. Herbst showed in [10] that, if $\mathcal{F}$ is a subfamily of the family of context-free languages which forms a cone (see Section 3), then the finitely generated groups whose word problem lies in the family $\mathcal{F}$ are either those with a regular word problem, those with a one-counter word problem or those with a context-free word problem. Herbst also classified [10] the groups with a one-counter word problem; see Section 4 for further details.

Another interesting related problem is that of characterizing which languages are word problems of groups. A simple necessary and sufficient criterion for a language to be such a word problem was established in [27]. This involves the conjunction of two conditions, universal prefix closure and deletion closure (see Definition 12 and Proposition 13); a natural question is then what other such characterizations there are. We investigate this problem, using sentences expressed in first-order logic where the only relations are membership of the language in question and concatenation of words. We choose some natural conditions on languages that hold in all word problems (see Definition 12) and then characterize which sets of these conditions are sufficient to guarantee that the language really is the word problem of a group (see Theorem 45). We also consider the related problem of asking when a language can represent a subgroup (or a normal subgroup) of a group, as opposed to just representing the identity element; see Section 7.

There are many interesting questions here. One general observation is that different families of languages can give rise to the same class of groups: this means that imposing certain very natural conditions on a language in one family forces it to lie in some specified proper subfamily. For example, any context-free language that satisfies the conditions for being a word problem of a group is necessarily deterministic context-free and even an NTS-language (see Section 4). Such results are not always easy to see directly (i.e. without using group theory) and this situation does seem worthy of further investigation.

Given that these properties are all natural ones for a language to potentially satisfy (many of which have been extensively studied elsewhere), in addition to being important when characterizing word problems of groups, we turn our attention to decidability results in Section 9. We see that the properties we study here are easily seen to be all decidable for the family of regular languages but they all turn out to be undecidable if we generalize to the family of one-counter languages (and hence to the context-free languages as well).

Having established the undecidability of these conditions for the family of one-counter languages, we turn our attention to the question of deciding whether or not a given language is the word problem of a group. Whilst this is easily seen to be decidable for the regular languages we build on the work in [22] to show that this is undecidable for one-counter languages (and hence for context-free languages as well); see Theorem 60. However, we know that any context-free language that is the word problem of a group is deterministic context-free and we show that the problem of deciding whether a deterministic context-free language is the word problem of a group is actually decidable (see Theorem 65).

2. Background from formal language theory

In this section and the next we will recall some concepts, notation and results we need from formal language theory and from group theory. For the background material on formal language theory the reader is referred to (for example) [3, 16, 19] and, for group theory, to [20, 30].

As usual, we let Σ^* denote the set of all *words*, including the *empty word* ϵ, and Σ^+ denote the set of all non-empty words over the alphabet Σ. (Our alphabets will always be finite here.) If $\alpha \in \Sigma^*$ and $x \in \Sigma$ we let $|\alpha|$ denote the length of α and $|\alpha|_x$ denote the number of occurrences of the symbol x in α. If $n \in \mathbb{N}$ then $\Sigma^{\leqslant n}$ is the set of words in Σ^* of length at most n and $\Sigma^{\geqslant n}$ is the set of words of length at least n (we will take the set $\mathbb{N}$ of natural numbers to include 0 in this paper).

If $\alpha = \beta\gamma$ for some $\beta, \gamma \in \Sigma^*$ then β is said to be a *prefix* of α and γ is said to be a *suffix* of α; if $\alpha = \beta\gamma\delta$ for some $\beta, \gamma, \delta \in \Sigma^*$ then γ is said to be a *factor* of α. If α is the word $a_1 a_2 \dots a_{n-1} a_n$ with $n \geqslant 1$ and $a_i \in \Sigma$ for each i, then the *reversal* α^{rev} of α is the word $a_n a_{n-1} \dots a_2 a_1$ (and ϵ^{rev} is defined to be ϵ). For any language L we define L^{rev} to be the language $\{\alpha^{rev} : \alpha \in L\}$.

Given a language L over an alphabet Σ we define the *syntactic congruence* $\approx_L$ to be the congruence on Σ^* defined by:

$$\alpha \approx_L \beta \iff (\gamma\alpha\delta \in L \Leftrightarrow \gamma\beta\delta \in L \text{ for all } \gamma, \delta \in \Sigma^*),$$

and then the *syntactic monoid* $Synt(L)$ of L to be the quotient $\Sigma^*/\approx_L$. If φ is the natural map from Σ^* onto $Synt(L)$ then $L = S\varphi^{-1}$ for some subset S of $Synt(L)$.

Remark 1. We will need the following elementary observation later in this paper. If L is a language over an alphabet Σ then the monoid $Synt(L)$ is trivial if and only if $\approx_L$ consists of a single congruence class, i.e. if and only if the only congruence class of $\approx_L$ is Σ^*. As L is a union of congruence classes of $\approx_L$, if L is non-empty then L must be Σ^*. So the only languages with a trivial syntactic monoid are $\emptyset$ and Σ^*. □

Throughout this paper we will be discussing the families of regular and context-free languages accepted by finite automata and (non-deterministic) pushdown automata respectively. Alternatively one could define these families using regular and context-free grammars, and we will need this approach as well in this paper. We will also be discussing *one-counter languages* which are those languages accepted by a *one-counter pushdown automaton*, i.e. a pushdown automaton where we have only a single stack symbol (apart from a special symbol marking the bottom of the stack); these automata are non-deterministic and accept by final state. We will use some standard definitions and properties of families of languages (such as their closure properties under certain operations and decidability results); see [3, 16, 19] for example. We will make some further comments about this in the next section.

3. Background from group theory

Here we recall some standard terminology and definitions from the theory of groups; we start with semigroups and monoids (as we will also use these concepts in this paper). A *semigroup* is a set S together with a closed binary operation $*$ which is associative. A *monoid* is a semigroup M where there is an identity element $1 = 1_M$ for $*$. A *group* is a monoid G where each g in G has an inverse g^{-1}. We usually suppress the reference to $*$, simply referring to the group (say) as G and writing gh for $g * h$.

A subset H of G that also forms a group under $*$ is called a *subgroup* of G (and is a *proper subgroup* of G if $H \neq G$). If H is a subgroup of G and if we have that $g^{-1}Hg = H$ for all $g \in G$, then H is said to be a *normal* subgroup of G. For any subgroup H of G there is a unique maximal normal subgroup of G contained in H, namely $\bigcap\{g^{-1}Hg : g \in G\}$.

If H is a subgroup of a group G then the distinct subsets of the form Hg (where $g \in G$) are known as the (right) *cosets* of H in G and these subsets form a partition of G. If there are only finitely many such cosets then we say that the subgroup H has *finite index* in G.

Bound up with these concepts is the idea of a homomorphism. A *semigroup homomorphism* is a mapping $\varphi : S \to T$ (where S and T are semigroups) such that $(st)\varphi = (s\varphi)(t\varphi)$ for all $s, t \in S$. If S and T are monoids then we have a *monoid homomorphism* φ if φ is a semigroup homomorphism such that $1_S\varphi = 1_T$. For a *group homomorphism* between groups we need to also have that $(g\varphi)^{-1} = (g^{-1})\varphi$ for all $g \in G$ (although this follows from the fact that φ is a semigroup homomorphism, as does the fact that φ preserves the identity element in the group case).

We mentioned in Section 2 that we would be assuming facts about closure properties of families of languages; amongst the properties we will need are the following. Let Σ and Ω be alphabets. We say that a family of languages $\mathcal{F}$ is *closed under homomorphism* if

$$L \in \mathcal{F},\ L \subseteq \Sigma^*,\ \varphi : \Sigma^* \to \Omega^* \text{ a monoid homomorphism } \implies L\varphi \in \mathcal{F},$$

and that $\mathcal{F}$ is *closed under inverse homomorphism* if

$$L \in \mathcal{F},\ L \subseteq \Omega^*,\ \varphi : \Sigma^* \to \Omega^* \text{ a monoid homomorphism } \implies L\varphi^{-1} \in \mathcal{F}.$$

We also say that a family $\mathcal{F}$ of languages is *closed under intersection with regular languages* if

$$L_1 \subseteq \Sigma^*,\ L_2 \subseteq \Sigma^*,\ L_1 \in \mathcal{F},\ L_2 \text{ regular } \implies L_1 \cap L_2 \in \mathcal{F}.$$

Following [3], we define a *cone* to be a family of languages closed under homomorphism, inverse homomorphism and intersection with regular languages. There are other names in use for this concept such as a *full trio* (see [16] for example). The families of languages we will be most concerned with in this paper, namely the regular, one-counter and context-free languages, are all cones.

With regards to decision problems, it is well known (see [16] for example) that one cannot decide whether or not a context-free language $L \subseteq \Sigma^*$ is equal to Σ^* (the so called *universe problem*). In fact, this problem remains undecidable if one restricts oneself to the certain subfamilies of the context-free languages such as the one-counter languages [18]. We will need a slight strengthening of this last fact here where we restrict to the special case where the alphabet has size 2:

Theorem 2. *The following decision problem is undecidable:*

Input: *a one-counter pushdown automaton M with input alphabet Ω of size 2.*

Output: *"yes" if $L(M) = \Omega^*$;*
"no" otherwise.

PROOF. We will show that, if we had an algorithm $\mathfrak{A}$ solving this decision problem, then we would have an algorithm solving the universe problem for one-counter pushdown automata with input alphabets of arbitrary size. So suppose that we have such an algorithm $\mathfrak{A}$. Let $\Sigma = \{x_1, x_2, \ldots, x_n\}$ be an arbitrary alphabet and let $\Omega = \{a, b\}$. Let $M = (Q, \Sigma, \Gamma, \tau, s, A)$ be a one-counter pushdown automaton and let $L = L(M)$. We want to determine whether or not $L(M) = \Sigma^*$.

Define $\varphi : \Sigma^* \to \Omega^*$ by $x_1 \mapsto ab$, $x_2 \mapsto a^2b$, $\ldots$, $x_n \mapsto a^nb$, and let $K = \Sigma^*\varphi$. Since K is regular, $R = \Omega^* - K$ is regular. Since φ is injective we have that $L = \Sigma^*$ if and only if $L\varphi = \Sigma^*\varphi = K$ which is equivalent to saying that $L\varphi \cup R = K \cup R = \Omega^*$. Since $L\varphi \cup R$ is a one-counter language over Ω we may use the algorithm $\mathfrak{A}$ to decide this problem, and so we could determine whether or not $L(M) = \Sigma^*$, a contradiction. □

Remark 3. In Theorem 2 all we have used about the family $\mathcal{F}$ of one-counter languages are the facts that the universe problem is undecidable for $\mathcal{F}$ and that the family $\mathcal{F}$ is closed under homomorphism and union with regular languages (and that we can effectively construct the necessary one-counter pushdown automaton given the specified homomorphism or regular language); so Theorem 2 applies to any family of languages with such properties. □

If G is a group and Σ is a finite set of symbols such that there is a surjective (monoid) homomorphism $\varphi : \Sigma^* \to G$, then we say that Σ is a *generating set* for G (via φ). Note that Σ is then a *monoid generating set* for G as opposed to a *group generating set*; in the latter case, we would have a set of symbols X and then let $\Sigma = X \cup X^{-1}$ where X^{-1} is a set of symbols in a (1-1) correspondence with X and where we insist that $x^{-1}\varphi = (x\varphi)^{-1}$ for each $x \in X$.

In either case the *word problem* of the group G is the set of all words in Σ^* that represent the identity element of G (i.e. the set of all words $\alpha \in \Sigma^*$ such that $\alpha\varphi = 1_G$).

For each $a \in \Sigma$ let $\overline{a}$ be an element of Σ^* such that $\overline{a}\varphi = (a\varphi)^{-1}$. We then have that $a_1a_2 \ldots a_n = b_1b_2 \ldots b_m$ in G (where $a_i, b_j \in \Sigma$ for each i and j) if and only if the word

$$a_1a_2 \ldots a_n\overline{b_m}\,\overline{b_{m-1}} \ldots \overline{b_1}$$

represents 1_G; so we can focus on the set of the words in Σ^* representing the identity of G and we refer to this language as the *word problem* $W(G, \Sigma)$ of G with respect to the generating set Σ (via the homomorphism φ).

Remark 4. A group is the syntactic monoid of its word problem; see [11] for example. We will need a more general result here. Let Σ be a finite set, G be a group and $\varphi : \Sigma^* \to G$ be a surjective homomorphism. Suppose that H is a subgroup of G, such that there is no non-trivial normal subgroup of G contained in H, i.e. such that

$$\bigcap \{g^{-1}Hg : g \in G\} = \{1\},$$

and let $L = H\varphi^{-1}$; then G is the syntactic monoid of L (see [26] for example).

Even more generally, suppose we drop the assumption that there is no non-trivial normal subgroup of G contained in H and let N be the maximal normal subgroup of G contained in H. Then $\varphi : \Sigma^* \to G$ induces a surjective homomorphism $\overline{\varphi} : \Sigma^* \to \overline{G} = G/N$ and there is no non-trivial normal subgroup of G/N contained in H/N. We now have that $L = H\overline{\varphi}^{-1}$ and that G/N is the syntactic monoid of L. □

A *presentation* for a group G is an expression of the form $\langle A : R \rangle$ where A is a generating set for G and R is a set of relations of the form $\alpha = \beta$. If A is a monoid generating set for G and $R \subseteq A^* \times A^*$, then we have a *monoid presentation* for G (we obviously also have monoid presentations for monoids that are not groups) and, if A is a group generating set for G, $\Sigma = A \cup A^{-1}$ as above, and $R \subseteq \Sigma^* \times \Sigma^*$, then we have a *group presentation* for G. In each case the set R must be a set of *defining relations* for G in the following sense: if $\approx$ is the congruence generated by R (together with all pairs of the form $(x^{-1}x, \epsilon)$ or (xx^{-1}, ϵ) with $x \in X$ in the case of a group generating set, where ϵ again denotes the empty word), then G is isomorphic to $\Sigma^*/\approx$, i.e. $\alpha \approx \beta$ if and only if $\alpha\varphi = \beta\varphi$. The *free group* on a set X has the (group) presentation $\langle X : \emptyset \rangle$. If we consider $\langle X : \emptyset \rangle$ as a monoid presentation then we get the *free monoid* on X which is naturally isomorphic to X^*.

Remark 5. If G is a group and $\wp = \langle A : R \rangle$ is a monoid presentation for G, then $\wp$ is also a group presentation for G. On the other hand, if $\wp = \langle A : R \rangle$ is a group presentation for G, we may easily obtain a monoid presentation for G as follows: for every generator $a \in A$ we add a new generator $\overline{a}$ and we then add the relations $a\overline{a} = \overline{a}a = 1$; the resulting presentation is now a monoid presentation for G. Given that we can effectively transform a monoid presentation into a group presentation and vice-versa, there is no need in many instances to specify which type of presentation we have, and we will adopt this convention throughout this paper where appropriate (as opposed to duplicating results, specifying them for group and monoid presentations separately). □

If P is any property of groups, then we say that a group is *virtually* P if it has a subgroup of finite index with property P. We will need the following fact (where we are adopting the convention from Remark 5):

Theorem 6. *The following problem is decidable:*

Input: *a finite presentation $\wp = \langle X : R \rangle$ for a virtually free group G;*
Output: *"yes" if G is trivial;*
"no" otherwise.

PROOF. We start two processes running. The first enumerates the consequences of the set R of relations, terminating if all the pairs (x, ϵ) with $x \in X$ have been output; this terminates if and only if G is trivial. The second process enumerates the subgroups of finite index in G (one can do this for any finitely presented group; see Section 5.6 of [32] or Section 5.4 of [13] for example) and terminates if it finds a proper subgroup (any non-trivial virtually free group must possess such a subgroup), which shows that the group is non-trivial. Eventually one of these two processes must terminate. □

4. Word problems and formal languages

When examining groups based on their word problem as a formal language it is quite common to try to classify groups based on what family of languages their word problem lies in. Whilst we will concentrate on the results most pertinent to the current paper here, the reader is referred to [4, 9, 15, 21, 33] (for example) for more information.

The first issue is that there is no guarantee that the word problem will lie in the same family $\mathcal{F}$ of languages for different finite generating sets. The following result (see [11] for example) shows that, under certain mild assumptions on $\mathcal{F}$, this is not a problem:

Theorem 7. *If a family of languages $\mathcal{F}$ is closed under inverse homomorphism and if the word problem of a group G lies in $\mathcal{F}$ with respect to some finite generating set then the word problem of G will lie in $\mathcal{F}$ for all finite generating sets.*

Theorem 7 applies to all the families of languages we will be most concerned with here, namely the families of regular, one-counter and context-free languages, as all these families are closed under inverse homomorphism (indeed, as previously mentioned, they are all cones). With regards to the first of these three families, Anisimov [1] classified the groups with a regular word problem:

Theorem 8. *A finitely generated group has a regular word problem if and only if it is a finite group.*

Further work was done by Muller and Schupp [23] which, along with a result of Dunwoody [7] concerning a property of groups known as accessibility, characterised the groups with a context-free word problem:

Theorem 9. *A finitely generated group G has a context-free word problem if and only if it is a virtually free group.*

There are several other interesting results related to Theorem 9; for example, it is not difficult to show that the word problem of a finitely generated virtually free group must be deterministic context-free (we pick up on this point in Section 11 below). One can go further: for example it was shown in [2] that such a word problem must even be an NTS language (i.e. a language generated by a context-free grammar where the set of sentential forms is unchanged when the production rules are used both ways). This phenomenon (different natural families of languages giving rise to the same class of groups) does not seem to be unusual. Another interesting reference here is [6] which presents a self-contained proof of Theorem 9 without using either the structure theorem of Stallings (which is important in the original proof in [23]) or the result of Dunwoody in [7].

One might ask what families of languages $\mathcal{F}$ contained in the family of context-free languages give rise to interesting classes of groups other than the finite and virtually free groups. Herbst [10] showed that, if the family $\mathcal{F}$ satisfies the natural closure conditions we have introduced, then there are not many possibilities for the corresponding class of groups:

Theorem 10. *If $\mathcal{F}$ is a subset of the context-free languages which is a cone then the class of finitely generated groups whose word problem lies in $\mathcal{F}$ is the class of groups with a regular word problem, the class of groups with a one-counter word problem or the class of groups with a context-free word problem.*

In the light of Theorems 8, 9 and 10, it is natural to ask which groups have a one-counter word problem. Herbst characterised these groups in [10] (see also [11] and [14]):

Theorem 11. *A finitely generated group G has a one-counter word problem if and only if it is a virtually cyclic group.*

Given Theorem 10 it is natural to ask if one can decide if a language lying in one of these three families of languages is a word problem of a group and we answer this question in Section 10.

5. Properties of word problems

As we said in the introduction, we are interested in determining which sets of properties of languages are sufficient to ensure that a language must be the word problem of a group. Obviously such properties must be ones that are satisfied by word problems of groups; the particular properties we will consider here are listed in the following definition:

Definition 12. We define some potential properties of a language L over an alphabet Σ:

(UPP) for all $\alpha \in \Sigma^*$ there exists $\beta \in \Sigma^*$ such that $\alpha\beta \in L$;
if L satisfies (UPP) we say that L has the *universal prefix property*;

(USP) for all $\alpha \in \Sigma^*$ there exists $\beta \in \Sigma^*$ such that $\beta\alpha \in L$;
if L satisfies (USP) we say that L has the *universal suffix property*;

(UFP) for all $\alpha \in \Sigma^*$ there exist $\beta, \gamma \in \Sigma^*$ such that $\beta\alpha\gamma \in L$;
if L satisfies (UFP) we say that L has the *universal factor property*;

(DC) $\alpha \in \Sigma^*, \beta \in L, \gamma \in \Sigma^*, \alpha\beta\gamma \in L \Longrightarrow \alpha\gamma \in L$;
if L satisfies (DC) we say that L is *deletion closed*;

(CRD) $\alpha \in \Sigma^*, \beta \in L, \alpha\beta \in L \Longrightarrow \alpha \in L$;
if L satisfies (CRD) we say that L is *closed under right deletions*;

(CLD) $\alpha \in L, \beta \in \Sigma^*, \alpha\beta \in L \Longrightarrow \beta \in L$;
if L satisfies (CLD) we say that L is *closed under left deletions*;

(IC) $\alpha \in \Sigma^*, \beta \in \Sigma^*, \gamma \in L, \alpha\beta \in L \Longrightarrow \alpha\gamma\beta \in L$;
if L satisfies (IC) we say that L is *insertion closed*;

(CCS) $\alpha \in \Sigma^*, \beta \in \Sigma^*, \alpha\beta \in L \Longrightarrow \beta\alpha \in L$;
if L satisfies the (CCS) we say that L is *closed under cyclic shift*;

(CC) $\alpha \in L, \beta \in L \Longrightarrow \alpha\beta \in L$;
if L satisfies (CC) we say that L is *closed under concatenation.*

It is easy to check that that all the properties in Definition 12 are satisfied by word problems of groups; we will use this fact from now on without further comment. These are all natural conditions on families of languages and many of them have been extensively studied; it is intriguing that we have such connections between word problems of groups and natural formal language conditions such as these.

The following result from [27] will be the starting point for our investigations here:

Proposition 13. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (UPP) and (DC).*

The proof of Proposition 13 in [27] shows that L being a word problem of a group is equivalent to the conditions (UPP), (IC) and (DC), and then that (IC) is a consequence of (UPP) and (DC). We will explore further such connections between the properties from Definition 12 in Section 6. However we also note the following result from [21] (see Proposition 5.4 there):

Proposition 14. *If Σ is an alphabet and $\emptyset \neq L \subseteq \Sigma^*$, then the following are equivalent:*

(i) *there is a monoid M and a monoid homomorphism $\varphi : \Sigma^* \to M$ such that $L = \{1\}\varphi^{-1}$;*

(ii) *L satisfies (DC) and (IC).*

Remark 15. In the situation described in Proposition 14, the language L must also satisfy (CLD) and (CRD), as these are consequences of (DC), and L must also satisfy (CC), as that is a consequence of (IC).

In general L does not need to satisfy any of the other conditions. For (UPP), (USP) and (UFP) this is clear as we can take a monoid (such as Σ^*) where the identity is not the product of two other elements; in this situation $\{1\}\varphi^{-1}$ must be $\{\epsilon\}$ and these three conditions clearly do not hold in this case.

Of interest later will be an example where L satisfies (UFP) but not (UPP) or (USP); this particular L will also not satisfy (CCS) and we will describe the language in question here.

Consider the *bicyclic monoid* B; this is the monoid defined by the (monoid) presentation $\langle a, b : ab = 1\rangle$. Let $\Sigma = \{a, b\}$; we have the natural (monoid) homomorphism $\varphi : \Sigma^* \to B$ and we let $L = \{1\}\varphi^{-1}$.

Since $ab = 1$, each element of B is represented by a word of the form $b^i a^j$ (where $i, j \geqslant 0$) and we have that $(b^i a^j)\theta = (b^k a^\ell)\theta$ if and only if $i = k$ and $j = \ell$. Indeed, if we consider the complete (i.e. the confluent and terminating) string rewriting system $\mathcal{R}$ over Σ where the only rewriting rule is $ab \to \epsilon$, we see that $\mathcal{R}$ reduces any word α in Σ^* to the unique word β of the form $b^i a^j$ $(i, j \geqslant 0)$ that represents the same element of B as α (i.e. to the word β of this form such that $\beta\varphi = \alpha\varphi$).

We have that L satisfies (UFP): if $\alpha = a^{i_1} b^{j_1} \ldots\ a^{i_n} b^{j_n}$ let

$$J = i_1 + \ldots + i_n \quad \text{and} \quad I = j_1 + \ldots + j_n;$$

then $a^I \alpha b^J$ reduces in $\mathcal{R}$ to ϵ and so $a^I \alpha b^J \in L$. However, L does not satisfy (UPP): if we let $\alpha = b$ then there is no word β such that $\alpha\beta \in L$ as any word in L can be reduced to ϵ through repeated uses of the rewriting rule $ab \to \epsilon$ and no word starting in b can be so reduced. A similar argument shows that no word ending in a can be so reduced and so L does not satisfy (USP) either. The fact that no word starting in b can belong to L also shows that L does not satisfy (CCS) (since $ab \in L$ but $ba \notin L$). □

We will note here a fact about deletion closed languages that may be of some independent interest. To do this we will use the following result from [12], known as *Higman's Lemma*:

Theorem 16. *The set of finite words over a finite alphabet, as partially ordered by the subsequence relation, is well-quasi-ordered. This, in particular, implies that there does not exist an infinite sequence where the elements of the sequence are all pairwise incomparable or, equivalently, any set containing only pairwise incomparable finite words is finite.*

We will write $\alpha \prec \beta$ if α can be obtained by deleting some symbols in β, i.e. if α is a proper subsequence of β; if α is a (not necessarily proper) subsequence of β, then we will write $\alpha \preceq \beta$. We then have:

Proposition 17. *A language $L \subseteq \Sigma^*$ which is deletion closed and which contains Σ is regular.*

Proof. Given that L is deletion closed and contains Σ, deleting any symbols from a word in L always results in another word in L; so, if $\alpha \in L$ and $\beta \prec \alpha$,

then $\beta \in L$. If $L = \Sigma^*$ then the result is clearly true; so we will assume that $L \neq \Sigma^*$ in what follows.

First, consider words $\beta \notin L$ such that

$$\alpha \prec \beta \Longrightarrow \alpha \in L$$

(such words are guaranteed to exist since $\emptyset \neq L \neq \Sigma^*$). Given two such words γ and β we must have that $\gamma \not\prec \beta$ and that $\beta \not\prec \gamma$; so, by Theorem 16, the set U of all such words must be finite.

Consider the language

$$V = \{\alpha \in \Sigma^* : \text{there exists } \ \beta \in U \text{ such that } \beta \preceq \alpha\}.$$

This language V is regular (as all we are doing is checking that some subsequence lies in a specified finite set).

If $\alpha \in V$ then there exists $\beta \in U$ such that $\beta \preceq \alpha$ and so $\alpha \notin L$ (as $\beta \notin L$). Conversely, if $\alpha \notin L$, then choose β minimal such that $\beta \preceq \alpha$ and $\beta \notin L$. If $\gamma \prec \beta$, then $\gamma \in L$ by the minimality of β; so $\beta \in U$ and hence $\alpha \in V$.

Given this we see that $\Sigma^* - L = V$ is regular and hence that L is regular. □

Remark 18. The hypothesis that L contains Σ in Proposition 17 is necessary; for example, the language $\{a^n b^n : n \geqslant 0\}$ is deletion closed but not regular. Indeed, since the word problem of any finitely generated group is deletion closed and there are finitely generated groups with unsolvable word problem, there exist deletion closed languages that are not even recursively enumerable. □

We now introduce a concept that we will call *duality*.

Remark 19. Suppose we have (as in Definition 12) a sentence σ in first-order logic where the only relations in σ are membership of the language L in question and concatenation of words. We can obtain a new sentence σ' by reversing the order of the words in any concatenation in σ (but leaving everything else in the sentence σ fixed). For example, if we take the sentence

$$\forall \alpha \in \Sigma^*, \exists \beta \in \Sigma^* : \alpha\beta \in L$$

representing the property (UPP), then the only concatenation in the sentence is $\alpha\beta \in L$; we reverse this to get $\beta\alpha \in L$ and we now have the sentence

$$\forall \alpha \in \Sigma^*, \exists \beta \in \Sigma^* : \beta\alpha \in L$$

representing (USP). In this sense we say that (USP) is the *dual* of (UPP) (and that (UPP) is the dual of (USP)).

In a similar vein we see that (CRD) is the dual of (CLD) and that the other properties listed in Definition 12 are all self-dual. We sum these facts up in the following tables:

Dual properties	
(UPP)	(USP)
(CLD)	(CRD)

Self dual properties
(UFP) (DC) (IC)
(CCS) (CC)

□

The motivation for introducing this concept is that, when characterizing word problems of groups, we will make extensive use of the following result:

Proposition 20. *If L is a language over some alphabet Σ, $S = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ is a subset of the properties listed in Definition 12, σ_i' is the dual of σ_i for each i and $S' = \{\sigma_1', \sigma_2', \ldots, \sigma_n'\}$, then the following statements are equivalent:*

(i) L is the word problem of a group if and only if it satisfies S.

(ii) L is the word problem of a group if and only if it satisfies S'.

PROOF. We will first show that L is a word problem of a group if and only if L^{rev} is a word problem of a group.

If $\Sigma = \{a_1, a_2, \ldots, a_n\}$ and $\varphi : \Sigma^* \to G$ is a surjective homomorphism from Σ^* onto a group G then we define a new homomorphism $\theta : \Sigma^* \to G$ by $a\theta = (a\varphi)^{-1}$ for all $a \in \Sigma$. If L is the word problem of G then, since

$$(g_1 \ldots \ g_n)^{-1} = g_n^{-1} \ldots \ g_1^{-1}$$

in G and since we also have that $\alpha\varphi = 1$ if and only if $(\alpha\varphi)^{-1} = 1$, we see that L^{rev} is also the word problem of G via the homomorphism θ. Applying the argument again shows that, if L^{rev} is the word problem of a group, then $L = (L^{rev})^{rev}$ is also the word problem of a group.

The result now follows from the observation that L satisfies the properties in S if and only if L^{rev} satisfies the properties in S'. □

Remark 21. When we are considering the word problem of a group G we are looking at the preimage of the subgroup $\{1\}$ of G under a natural homomorphism $\varphi : \Sigma^* \to G$. We can also widen the scope and consider languages that are preimages of arbitrary subgroups of G under such a homomorphism. The argument in Proposition 20 goes over to this more general situation and we have that, with the same notation as in Proposition 20, if S is a necessary and sufficient set of conditions for L to be the preimage of a subgroup of G, then S' is also such a set of conditions. We will explore languages which are preimages of arbitrary subgroups of a group in Section 7. □

6. Characterizing word problems

We now take Proposition 13 and derive some other similar characterizations of languages that are word problems of groups. There are quite a few results in this section but, for the convenience of the reader, we summarize them in Remark 38 below.

First of all, using Remark 19 and Proposition 20, we immediately have:

Corollary 22. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (USP) and (DC).*

We then note the following:

Proposition 23. *If a language L over an alphabet Σ satisfies properties (CCS) and (UFP) then it satisfies property (UPP).*

PROOF. If $\alpha \in \Sigma^*$ then there exist $\beta, \gamma \in \Sigma^*$ such that $\beta\alpha\gamma \in L$ by (UFP). Then $\alpha\gamma\beta \in L$ by (CCS) and so there exists $\delta = \gamma\beta$ with $\alpha\delta \in L$ as required. □

Using Remark 19 we immediately have:

Corollary 24. *If a language L over an alphabet Σ satisfies properties (CCS) and (UFP) then it satisfies property (USP).*

Given Propositions 13 and 23, we have the following immediate consequence:

Corollary 25. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (DC), (CCS) and (UFP).*

We next note the following:

Proposition 26. *If a language L over an alphabet Σ satisfies properties (CCS) and (CRD) then it satisfies property (DC).*

PROOF. If $\alpha\beta\gamma \in L$ and $\beta \in L$ then we can apply (CCS), (CRD) and (CCS) in turn to get that $\gamma\alpha\beta \in L$, that $\gamma\alpha \in L$, and then that $\alpha\gamma \in L$ as required. □

Given Propositions 13 and 26, we have the following:

Corollary 27. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (UPP), (CCS) and (CRD).*

By Remark 19 and Proposition 26, we have the following:

Corollary 28. *If a language L over an alphabet Σ satisfies properties (CCS) and (CLD) then it satisfies property (DC).*

Given Proposition 13 and Corollary 28 we have:

Corollary 29. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (UPP), (CCS) and (CLD).*

Given Propositions 13, 23 and 26 we have:

Corollary 30. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (UFP), (CCS) and (CRD).*

In a similar vein, Propositions 13, 23 and 28 give:

Corollary 31. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (UFP), (CCS) and (CLD).*

Given Corollaries 22 and 28 we have:

Corollary 32. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (CCS), (USP) and (CLD).*

In a similar way, Corollaries 22 and 26 give:

Corollary 33. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (CCS), (USP) and (CRD).*

Another such result is the following:

Proposition 34. *If a language L over an alphabet Σ satisfies properties (UPP), (IC) and and (CRD) then it satisfies property (DC).*

Proof. Assume that L satisfies (UPP), (IC) and (CRD); we want to show that L must then satisfy (DC) as well.

So assume that $\alpha\beta\gamma \in L$ (for some $\alpha, \gamma \in \Sigma^*$) and that $\beta \in L$. By (UPP) there exists $\delta \in \Sigma^*$ such that $\alpha\gamma\delta \in L$. Since $\beta \in L$ we have by (IC) that $\alpha\beta\gamma\delta \in L$. Since $\alpha\gamma\delta \in L$ and $\alpha\beta\gamma \in L$, (IC) also gives us that $\alpha\gamma(\alpha\beta\gamma)\delta \in L$. Since $\alpha\gamma\alpha\beta\gamma\delta \in L$ and $\alpha\beta\gamma\delta \in L$, (CRD) gives that $\alpha\gamma \in L$ as required. □

Given Propositions 13 and 34 we have another characterization of word problems as follows:

Corollary 35. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (UPP), (IC) and (CRD).*

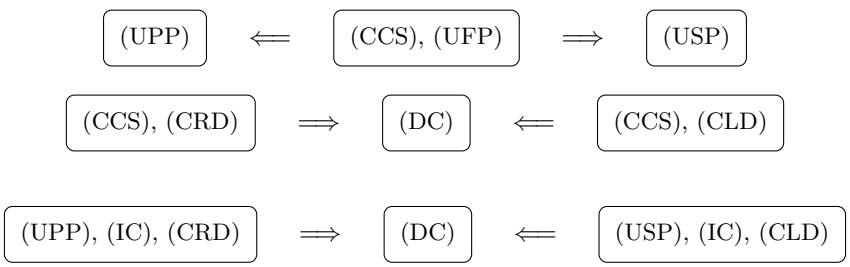
Given Proposition 34 we can apply Remark 19 to deduce:

Proposition 36. *If a language L over an alphabet Σ satisfies properties (USP), (IC) and (CLD) then it satisfies property (DC).*

Given Propositions 22 and 36 we have another characterization as follows:

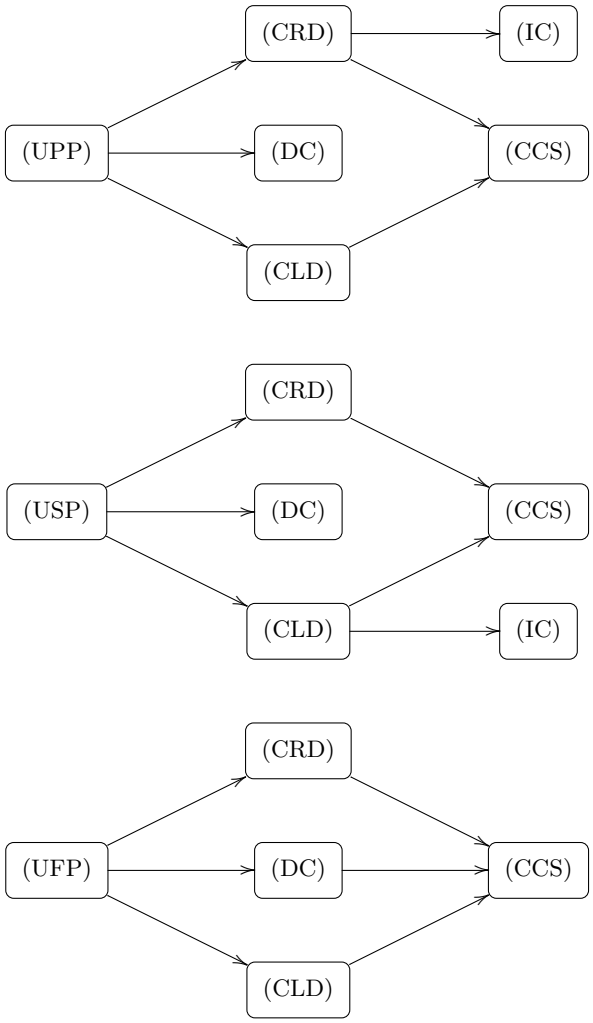
Corollary 37. *A language L over an alphabet Σ is the word problem of a group if and only if it satisfies properties (USP), (IC) and (CLD).*

Remark 38. For the convenience of the reader we summarize the implications between the properties listed in Definition 12 which we have established in this section by means of the following diagrams:



These various implications will be important in what follows.

We also summarize below the sets of conditions on a language which we have established are sufficient for the language in question to be the word problem of a group. The convention here is that one starts at one of the three positionson the left and then follows arrows to the right until one can go no further; the eleven paths one can trace doing this give the eleven such sets of conditions.



□

7. Preimages of subgroups

When we are considering the word problem of a group G we are looking at the preimage of the subgroup $\{1\}$ of G under a natural homomorphism $\varphi : \Sigma^* \to G$. In Section 5 we gave various characterizations of such a language. In this section we widen the scope of the languages considered and look at languages that are preimages of arbitrary subgroups of G.

We first make the following observation:

Proposition 39. *Let Σ be an alphabet, G be a group and $\varphi : \Sigma^* \to G$ be a surjective monoid homomorphism. If $\emptyset \neq S \subseteq G$ and $L = S\varphi^{-1}$ then the language L satisfies (UPP).*

Proof. We choose any element $x \in S$ and then let $\delta \in \Sigma^*$ be such that $\delta\varphi = x$; having done this, x and δ will be fixed in what follows.

Given an arbitrary $\alpha \in \Sigma^*$ we want to show that there exists $\beta \in \Sigma^*$ such that $\alpha\beta \in L$.

Let $\alpha \in \Sigma^*$ and then let $\gamma \in \Sigma^*$ be such that $\gamma\varphi = (\alpha\varphi)^{-1}$. Let $\beta = \gamma\delta$. Then

$$(\alpha\beta)\varphi = (\alpha\varphi)(\gamma\varphi)(\delta\varphi) = x \in S,$$

so that $\alpha\beta \in S\varphi^{-1} = L$ as required. □

Given this we can now establish the following:

Proposition 40. *Let Σ be an alphabet, G be a group and $\varphi : \Sigma^* \to G$ be a surjective monoid homomorphism. Let $S \subseteq G$ and then let $L = S\varphi^{-1}$. Then the following are equivalent:*

(i) S is a subgroup of G.
(ii) L satisfies (UPP), (CC) and (CRD).

Proof. First suppose that S is a subgroup of G; we have that L satisfies (UPP) by Proposition 39.

If $\alpha, \beta \in L$ then $\alpha\varphi, \beta\varphi \in S$ and so

$$(\alpha\beta)\varphi = (\alpha\varphi)(\beta\varphi) \in S,$$

so that $\alpha\beta \in L$; so L satisfies (CC).

Lastly, if $\alpha\beta \in L$, $\beta \in L$ then $(\alpha\beta)\varphi \in S$, $\beta\varphi \in S$, so that $(\beta\varphi)^{-1} \in S$ and then $\alpha\varphi = (\alpha\beta)\varphi(\beta\varphi)^{-1} \in S$, so that $\alpha \in L$; so L satisfies (CRD).

Conversely suppose that L satisfies (UPP), (CC) and (CRD). Note that $L\varphi\varphi^{-1} = L$. We also have that $L \neq \emptyset$, as L satisfies (UPP), and so $S \neq \emptyset$.

Since $L \neq \emptyset$ and L satisfies (CRD), we have that $\epsilon \in L$ and so $1 \in S$.

If $s, t \in S$ let $\alpha, \beta \in L$ be such that $\alpha\varphi = s$ and $\beta\varphi = t$; then $\alpha\beta \in L$, as L satisfies (CC) and $st = (\alpha\beta)\varphi$, so that $st \in S$.

Lastly, let $s \in S$ and then let $\alpha, \beta \in L$ be such that $\alpha\varphi = s$ and $\beta\varphi = s^{-1}$; then $(\beta\alpha)\varphi = 1 \in S$, so that $\beta\alpha \in L$, $\alpha \in L$, giving that $\beta \in L$ by (CRD) and then that $s^{-1} \in S$.

So S is a subgroup of G as required. □

Given Remark 21, we can use duality to deduce:

Corollary 41. *Let Σ be an alphabet, G be a group and $\varphi : \Sigma^* \to G$ be a surjective monoid homomorphism. Let $S \subseteq G$ and $L = S\varphi^{-1}$. Then the following are equivalent:*

(i) *S is a subgroup of G.*
(ii) *L satisfies (USP), (CC) and (CLD).*

We now turn our attention to normal subgroups and establish the following:

Proposition 42. *Let Σ be an alphabet, G be a group and $\varphi : \Sigma^* \to G$ be a surjective monoid homomorphism. Let H be a subgroup of G and then let $L = H\varphi^{-1}$. Then the following are equivalent:*

(i) *H is a normal subgroup of G.*
(ii) *L satisfies (IC).*
(iii) *L satisfies (DC).*
(iv) *L satisfies (CCS).*

Proof. If H is a normal subgroup of G then L is the word problem of the group G/H, and so L satisfies (IC), (DC) and (CCS). So (i) implies each of (ii), (iii) and (iv).

We will now complete the proof by showing that each of (ii), (iii) and (iv) implies (i).

Now the fact that H is a subgroup of G means that L satisfies (UPP), (USP), (CC), (CRD) and (CLD) by Proposition 40 and Corollary 41. Let N be the largest normal subgroup of G contained in H so that G/N is the syntactic monoid of L by Remark 4.

If L satisfies any of (IC), (DC) or (CCS) then L is the word problem of a group (by Corollary 35, Proposition 13 or Corollary 29 respectively). As in Remark 4, L is then the word problem of its syntactic monoid. Since G/N is the syntactic monoid of L it follows that every word in L must represent the identity of G/N, giving that $H = N$, and hence that H is a normal subgroup of G as required. □

8. Minimality results

In this section we build on the results in Section 6 and investigate the minimality of sets of conditions from Definition 12 on a language L that are sufficient for L to be a word problem of a group.

We first establish a result that will be crucial in establishing the minimality of certain such sets of conditions from Definition 12 when characterizing word problems of groups:

Proposition 43. *There are languages L_1, L_2, L_3, L_4, L_5 and L_6 that satisfy respectively the following specified subsets of the set of the properties listed in Definition 12:*

	(UPP)	*(DC)*	*(CCS)*	*(UFP)*	*(CRD)*	*(IC)*	*(CC)*	*(CLD)*	*(USP)*
L_1	$\surd$	X	$\surd$	$\surd$	X	$\surd$	$\surd$	X	$\surd$
L_2	X	$\surd$	$\surd$	X	$\surd$	X	X	$\surd$	X
L_3	$\surd$	X	X	$\surd$	$\surd$	X	$\surd$	$\surd$	$\surd$
L_4	X	$\surd$	X	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	X
L_5	X	$\surd$	$\surd$	X	$\surd$	$\surd$	$\surd$	$\surd$	X
L_6	$\surd$	X	X	$\surd$	X	$\surd$	$\surd$	$\surd$	X

Proof. Let $\Sigma = \{a, b\}$ and $n \geqslant 1$, and then let L_1 be the language $\Sigma^{\geqslant n}$ and L_2 be the language $\Sigma^{\leqslant n}$.

We see that the language L_1 satisfies the conditions (UPP), (USP), (UFP), (CCS), (CC) and (IC) but not (DC), (CLD) or (CRD).

On the other hand, the language L_2 satisfies the conditions (DC), (CRD), (CLD) and (CCS) but not (UPP), (USP), (UFP), (CC) or (IC).

Now let Ω be a finite set, G be a group, $\varphi : \Omega^* \to G$ be a surjective homomorphism, H be a non-trivial subgroup of G such that there is no non-trivial normal subgroup of G contained in H (in particular, H is itself not a normal subgroup of G) and then let L_3 be the language $H\varphi^{-1}$.

By Remark 4, we see that G is the syntactic monoid of L_3. Using Proposition 40, Corollary 41 and Proposition 42, we see that L_3 satisfies the conditions (UPP) (and hence (UFP) as well), (USP), (CC), (CRD) and (CLD), but not the conditions (IC), (DC) or (CCS).

For our next language L_4 we take the language L from Remark 15 consisting of the words representing the identity element of the bicyclic monoid B defined by the (monoid) presentation $\langle a, b : ab = 1\rangle$. We saw in Remark 15 that L satisfies the conditions (DC), (CRD), (CLD), (IC), (CC) and (UFP), but not the conditions (UPP), (USP) or (CCS).

We next consider L_5 be the empty language $\emptyset$. It is clear that L_5 satisfies the conditions (DC), (CCS), (CRD), (CLD), (IC) and (CC), but not the conditions (UPP), (USP) or (UFP).

Lastly we let $\Sigma = \{a, b\}$ and then let L_6 be the language $\{\epsilon\} \cup \Sigma^*\{a\}$. It is clear that L_6 satisfies the conditions (UPP) and (UFP) but not (USP). L_6 also satisfies (IC) and (CC) but not (CCS). Lastly L_6 satisfies (CLD) but not (CRD) or (DC). □

As we said above, the six languages specified in Proposition 43 will be useful in establishing the minimality of certain sets of conditions from Definition 12 which characterize word problems of groups. Indeed, we can now show that the eleven characterizations we have obtained so far are all minimal, in the sense that no proper subset S of any of the specified eleven sets of properties is sufficient to ensure that a language satisfying all the properties in S is the word problem of a group:

Proposition 44. *For any non-empty proper subset S of any of the sets*

$\{(UPP), (DC)\}$,	$\{(DC), (CCS), (UFP)\}$,
$\{(UPP), (CCS), (CRD)\}$,	$\{(CCS), (CRD), (UFP)\}$
$\{(UPP), (IC), (CRD)\}$,	$\{(USP), (DC)\}$,
$\{(IC), (CLD), (USP)\}$,	$\{(UPP), (CCS), (CLD)\}$,
$\{(USP), (CCS), (CLD)\}$,	$\{(USP), (CCS), (CRD)\}$,
or	$\{(UFP), (CCS), (CLD)\}$

there is a language satisfying all the conditions in S which is not a word problem of a group.

Proof. We will refer to the six languages L_1, L_2, L_3, L_4, L_5 and L_6 introduced in Proposition 43.

To eliminate proper subsets of ...	... we consider ...
$\{(\text{UPP}), (\text{DC})\}$	L_1 and L_2.
$\{(\text{DC}), (\text{CCS}), (\text{UFP})\}$	L_1, L_2 and L_4.
$\{(\text{UPP}), (\text{CCS}), (\text{CRD})\}$	L_1, L_2 and L_3.
$\{(\text{CCS}), (\text{CRD}), (\text{UFP})\}$	L_1, L_2 and L_3.
$\{(\text{UPP}), (\text{IC}), (\text{CRD})\}$	L_1, L_3 and L_4.
$\{(\text{USP}), (\text{DC})\}$	L_1 and L_2.
$\{(\text{IC}), (\text{CLD}), (\text{USP})\}$	L_1, L_3 and L_4.
$\{(\text{UPP}), (\text{CCS}), (\text{CLD})\}$	L_1, L_2 and L_3.
$\{(\text{USP}), (\text{CCS}), (\text{CLD})\}$	L_1, L_2 and L_3.
$\{(\text{USP}), (\text{CCS}), (\text{CRD})\}$	L_1, L_2 and L_3.
$\{(\text{UFP}), (\text{CCS}), (\text{CLD})\}$	L_1, L_2 and L_4.

For each maximal proper subset S of one the eleven sets we have given an example of a language satisfying all the properties in S which is not the word problem of a group. □

Theorem 45. *The eleven sets of properties listed in Proposition 44 are precisely those subsets S of the set of properties listed in Definition 12 such that satisfying the conditions in S is sufficient for a language L to be the word problem of a group but such that no proper subset of S has this property.*

Proof. Again, we will refer to the languages L_1, L_2, L_3, L_4, L_5 and L_6 specified in Proposition 43.

To start with, notice that the empty set L_5 is not a characterization and satisfies all of the properties except (UPP), (USP) and (UFP); so any characterization must contain at least one of these three properties. Next we note that, if a language satisfies (CCS) and any one of (UPP), (USP) and (UFP), then it satisfies all of them; so, to begin with, we will consider languages which do not satisfy (CCS).

Note, also, that each of (USP) and (UPP) implies (UFP) so that, when considering languages which satisfy two of (USP), (UPP) and (UFP), there

is therefore really only one pair to consider (taking minimality into account), namely (UPP) and (USP).

The result of these observations is that we have five cases which we will need to consider (with respect to minimal characterizations):

Case 1. We specify (UPP) but not (CCS).
Case 2. We specify (USP) but not (CCS).
Case 3. We specify (UFP) but not (CCS).
Case 4. We specify (UPP) and (USP) but not (CCS).
Case 5. We specify (CCS) and one of (UPP), (USP) and (UFP).

Let us consider Case 1 where we specify (UPP) but not (CCS), (USP) or (UFP). Since (UPP) and (DC) is already a characterization by Proposition 13 there is no minimal characterization properly containing both of these properties; so we will exclude (DC). Since (IC) implies (CC) we do not include both of these; so we are looking at subsets of (UPP), (CLD), (CRD) and (CC) or of (UPP), (CLD), (CRD) and (IC).

With regards to the set consisting of (UPP), (CLD), (CRD) and (CC), the language L_3 satisfies all these conditions, and so no subset of this set is sufficient for a characterization.

Let us now consider (UPP), (CLD), (CRD) and (IC). Considering L_3 again we see that (IC) must be included. If we only have (UPP) and (IC) then this is not sufficient as is demonstrated by L_1. If we add (CLD) to (UPP) and (IC) we see that this is not a characterization as witnessed by L_6. If we add (CRD) to (UPP) and (IC) we have a characterization by Corollary 35, and this is minimal by Proposition 44.

Case 2 is the dual of Case 1 (in the sense of Remark 19). Using Proposition 20 we see that the only minimal sets of conditions here are $\{(\text{USP}), (\text{DC})\}$ and $\{(\text{USP}), (\text{IC}), (\text{CLD})\}$.

Case 3 cannot give rise to any characterizations as witnessed by L_4 which satisfies all the properties listed in Definition 12 except (UPP), (USP) and (CCS).

Let us now consider Case 4 where we specify (UPP) and (USP) but not (CCS) or (UFP). Again, using Proposition 13, we can exclude (DC) if we are considering minimal characterizations. Since (IC) implies (CC), we do not include both of these properties; so we are looking at subsets of (UPP), (USP), (CLD), (CRD) and (CC) or of (UPP), (USP), (CLD), (CRD) and (IC). With regards to (UPP), (USP), (CLD), (CRD) and (CC), the language L_3 satisfies all these conditions, and so no subset of this particular set is sufficient for a characterization.

Now consider (UPP), (USP), (CLD), (CRD) and (IC). Given L_3 we see that (IC) must be included. If we only have (UPP), (USP) and (IC) this is not sufficient as demonstrated by L_1. If we add (CRD) to (UPP), (USP) and (IC) then we have a proper superset of $\{(\text{UPP}), (\text{IC}), (\text{CRD})\}$ which is a characterization as above, and, if we add (CLD) to (UPP), (USP) and (IC) then we have a proper superset of $\{(\text{USP}), (\text{IC}), (\text{CLD})\}$ which is also a characterization; so no new minimal characterizations arise here.

Lastly consider Case 5. We first consider the case where we have (CCS) and (UPP). Again, by minimality, we can assume that (DC) is excluded.

Given Proposition 44, if we include (CRD), then we have a minimal characterization by Corollary 27 and, if we include (CLD), then we have a minimal characterization by Corollary 31. We must include one of these, however, as L_1 satisfies (UPP), (CCS), (IC) and (CC).

We next turn to the case where we specify (CCS) and (USP). This is the dual of the case where we specify (CCS) and (UPP) and so we get the minimal characterizations $\{(\text{CCS}), (\text{USP}), (\text{CRD})\}$ and $\{(\text{CCS}), (\text{USP}), (\text{CLD})\}$ here.

Lastly we look at the case where we specify (CCS) and (UFP). Given that (UPP), (USP) and (UFP) are all equivalent in the presence of (CCS), we get (using Proposition 44) the minimal characterizations $\{(\text{CCS}), (\text{UFP}), (\text{CRD})\}$ and $\{(\text{CCS}), (\text{UFP}), (\text{CLD})\}$. The only other possibility would be to include (DC) as, unlike (UPP) and (USP), (DC) is not sufficient to guarantee a word problem when taken in conjunction with (UFP) as witnessed by L_4. The set $\{(\text{CCS}), (\text{UFP}), (\text{DC})\}$ is a characterization by Corollary 25 and is minimal by Proposition 44; so this is our last possibility (as we clearly cannot take any set properly containing it and preserve minimality). □

9. Decidability results

We now investigate the decidability of the properties listed in Definition 12. First of all we point out that it is reasonably clear that these are all decidable for the family of regular languages, i.e. given a finite automaton M we can decide whether or not $L(M)$ satisfies the property in question.

Proposition 46. *All the properties listed in Definition 12 are decidable for the family of regular languages.*

Proof. One possible approach for regular languages involves considering the syntactic monoid of $L(M)$. If $L = L(M) \subseteq \Sigma^*$ then we know that $Synt(L)$ is finite, that $L = S\varphi^{-1}$ for some $S \subseteq M$ (where φ is the natural map from Σ^* onto $Synt(L)$) and that we can explicitly construct $Synt(L)$ and S from M. Given this, the condition (UPP) (for example) is equivalent to the sentence

$$\forall x \in Synt(L),\, \exists y \in Synt(L) : xy \in S,$$

which is decidable as $Synt(L)$ is finite. The decidability of the other properties listed in Definition 12 for regular languages can all be established in the same way. □

Remark 47. We are only interested in decidability questions in this paper and not with computational complexity. In our one non-trivial result which establishes decidability (see Theorem 65) we do not have an easily computable time complexity (see Remark 66). For more information about the complexity of problems related to that described in Proposition 46 see [28] for example. □

When we consider the corresponding questions for one-counter languages then we will need the idea of a *counter machine*; see [16] for example. There are several ways of defining these machines and we give one possibility here.

In [16] counter machines are taken to be deterministic Turing machines which have a read-only input tape and a number of stacks each of which can contain only one symbol apart from a bottom of stack marker (which is present at the start of the computation and which is never erased). As the contents of each stack can be completely specified in terms of the number of instances of the symbol present, the stacks are usually referred to as *counters*. The one-counter pushdown automaton we introduced in Section 2 is essentially an example of a non-deterministic version of such a machine where we only have one counter (although, as usual, one has to be careful about allowing for the presence of a halt state in a counter machine as opposed to having accept states in a one-counter pushdown automaton).

In our model of a Turing machine (and, in particular, in our version of a counter machine) the input tape will be read-only and the read head will only move to the right. This is not a restriction on the power of the class of machines, as the input could always be copied to a work tape if necessary, but this convention will simplify our discussions here.

It is known that having two counters in a counter machine means that one can simulate an arbitrary Turing machine; see [16] for example. The argument proceeds by first showing that having four such counters is sufficient (in that this allows one to simulate two general stacks) and then showing how the four counters can be simulated by two counters. This is done by allowing one counter to contain $n = 2^b3^c5^d7^e$ symbols (there is no particular significance in the choice of 2, 3, 5 and 7, in that any four pair-wise coprime natural numbers would suffice here). We have that $b, c, d, e \geqslant 0$ (so that $n \geqslant 1$); in this way the counter with $2^b3^c5^d7^e$ symbols represents four counters with b, c, d and e symbols respectively. To increment b, c, d or e by 1, we multiply n by 2, 3, 5 or 7 respectively and, to decrement b, c, d or e by 1, we divide n by 2, 3, 5 or 7 respectively.

Of course, we can only decrement b if $b > 0$, i.e. if n is divisible by 2, with a similar restriction applying to c, d and e as well. The other counter in the two-counter machine (as described in [16]) is used to test whether or not each of b, c, d and e is equal to 0 and also to perform the necessary multiplications and divisions. We note that, if the original Turing machine never moves right on its input tape, then none of the successive machines we have mentioned here performing the simulations will do so either.

For our purposes we will take a slight variation of this model. We will have a single counter containing $n = 2^b3^c5^d7^e$ symbols but, rather that having a second counter to allow testing equality of b, c, d and e to 0 and performing the multiplications and divisions, we will have these facilities built into the machine M, i.e. M knows at each stage whether or not each of b, c, d and e is equal to 0, and M can also multiply and (if possible) divide the number n by 2, 3, 5 or 7 in one move. This is sufficient to simulate the four-counter model (and hence any Turing machine), as per the discussion in [16], but having just one counter will make the resulting discussion here much simpler.

We can think of the knowledge of whether or not each of b, c, d and e is equal to 0 as somehow being stored in the state of M (and updated after every move), so that M knows whether or not each of the four counters in the four-counter machine it is simulating is empty or not. Given that the four-counter machine that M is simulating will have (at most) one possible move for any given state, input symbol and stack symbol on each stack, for any state of M and any input symbol there will be at most one possible move (which will be multiplying the counter by some specified value of k if possible, otherwise leaving the counter unchanged, with specified states to go to in each case).

Given all this, for our purposes a counter machine M will be a particular type of two-tape Turing machine. The first tape is the input tape; it is read only and the head can only move to the right. The second tape is a stack so that, whenever we move left, M erases the symbol it moved away from. There is only one stack symbol, a say; as discussed above, this means that M can effectively only store a natural number, and so we can think of M as having an input tape and a counter. As we will see, the stack is never empty (and so, in this model, we will not need the presence of a bottom marker for the stack).

More formally, a *counter machine* is a sextuple $M = (Q, \Sigma, a, \delta, q_0, q_h)$ where Q is a finite set of states containing two distinguished states, q_0, the *start state*, and q_h, the *halt state*. The input alphabet Σ is a finite set of symbols such that $a \notin Q \cup \Sigma$ and $Q \cap \Sigma = \emptyset$. We have a designated special symbol $\blacktriangledown$ (which is not an element of Σ) to mark the right-hand end of the input tape (as we never move left on the input tape, there is no need to mark the left-hand end of that tape). We let $\overline{\Sigma}$ denote the set $\Sigma \cup \{\blacktriangledown\}$.

A *configuration* of M is a word of the form $\theta q a^n$ where $\theta\blacktriangledown \in \Sigma^*\blacktriangledown$ is the input remaining to be read (with the convention that $\theta = \epsilon$ if the current symbol on the input tape is $\blacktriangledown$), $q \in Q$ is the current state of M and the current stack contents of M are a^n ($n > 0$). Given our assumptions that $a \notin Q \cup \Sigma$ and that $Q \cap \Sigma = \emptyset$, there is no ambiguity here.

We take C to be $\{1, 2, 3, 5, 7, \frac{1}{2}, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}\}$, i.e. the set of values which can be used to multiply the current number stored in the counter. The transition relation δ is then a subset of

$$(Q - \{q_h\}) \times \overline{\Sigma} \times C \ \times \ (Q - \{q_0\}) \times \{N, R\} \times (Q - \{q_0)\} \times \{N, R\},$$

so that, once the computation has started, M never re-enters the start state q_0 (and there is no move out of the halt state q_f).

As per the above the discussion, for any $q \in Q$ and any $u \in \overline{\Sigma}$, there is at most one septuple of the form $(q, u, -, -, -, -, -)$ in δ. We specify that M starts in state q_0 with just a on its stack (i.e. with the counter set to 1) and, insist that, when accepting, M must have read all its input (i.e. it must be scanning the symbol $\blacktriangledown$) and must have set its counter to 1 again before entering the halt state q_h.

A transition $(q, u, k, p, D_1, r, D_2)$ in δ is interpreted as follows. If M is in state q reading an input u and if the result of multiplying the current value n of the counter (i.e. we currently have a^n on the stack) by the value k is an integer,

then we set the counter to kn, move to state p and move in direction D_1 on the input tape (i.e. we don't move if $D_1 = N$ and we move one cell right if $D_1 = R$); if kn is not an integer then the counter remains set at n, M moves to state r and we move in direction D_2 on the input tape. We write $\theta q a^n \vdash \zeta p a^{kn}$ (where $\theta = \zeta$ if $D_1 = N$ and $\theta = u\zeta$ if $D_1 = R$) or $\theta q a^n \vdash \zeta r a^n$ (again where $\theta = \zeta$ if $D_2 = N$ and $\theta = u\zeta$ if $D_2 = R$) as appropriate.

We obviously do not have transitions $(q, \blacktriangledown, k, p, D_1, r, D_2)$ with $D_1 = R$ or $D_2 = R$. If C and C' are configurations of M such that we do not have $C \vdash C'$, then we will write $C \nvdash C'$. Note that, given the way we have set up our machine, given any configuration C there is at most one configuration C' such that $C \vdash C'$.

In a counter machine any configuration $\theta q a^n$ will have $n = 2^b 3^c 5^d 7^e$ for some $b, c, d, e \geqslant 0$. Multiplying n by 2, 3, 5 or 7 increases b, c, d or e respectively by 1 and multiplying n by $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{5}$ or $\frac{1}{7}$ (if possible) decreases b, c, d or e by 1; so, as discussed above, we effectively have four counters each of which can be increased or decreased by 1. The fact that we can only multiply by k if kn is an integer effectively says that we can test each of the four counters individually for zero (for example, if $n = 2^b 3^c 5^d 7^e$ and we want to multiply by $\frac{1}{2}$, then we must have that $b > 0$); in the four-counter machine this is achieved by having the special symbol marking the bottom of each stack and, as we have said, in our version of a counter machine the machine knows at each stage which (if any) of b, c, d and e are equal to zero.

Given all this, given a Turing Machine T, one can effectively construct a counter machine (as defined here) accepting the same language as T.

We now turn to the notion of a valid computation of a counter machine:

Definition 48. Let M be a counter machine with input ψ. A *valid computation* of M is a word $C_0 C_1 \dots C_n \in (\Sigma \cup Q \cup \{a\})^*$ where the C_i are configurations of M and

$$C_0 = \psi q_0 a \vdash C_1 \vdash \dots \vdash C_n = q_h a;$$

other elements of $(\overline{\Sigma} \cup Q \cup \{a\})^*$ are said to be *invalid computations*.

Note that, given the way we have defined counter machines, for any input ψ to a counter machine M, there is at most one valid computation of M.

Our aim here is to show that, for any of the properties listed in Definition 12, the problem of deciding whether or not a language has that property is undecidable for the family of one-counter languages. In order to do this we will relate the set of invalid computations of a counter machine M to a one-counter language (i.e. a language accepted by a one-counter pushdown automaton as defined in Section 2). There have been other similar approaches to such problems (see [34] for example).

We start with the following technical result which essentially says that we can verify that a transition from one given configuration to another in a counter machine is not valid using a one-counter pushdown automaton; this will be the main tool in the proof of undecidability that follows (see Theorem 52).

Proposition 49. *If $M = (Q, \Sigma, a, \delta, q_0, q_h)$ is a counter machine then the language*

$$\{CC' : C \text{ and } C' \text{ are configurations of } M,\ C \nvdash C'\}$$

is a one-counter language.

PROOF. Let $L = \{CC' : C \text{ and } C' \text{ are configurations of } M\}$ and let

$$K = \{CC' : C \text{ and } C' \text{ are configurations of } M,\ C \nvdash C'\}.$$

For any $u \in \overline{\Sigma}$ and $q \in Q$ let

$$L_{u,q} = \{\ CC' : C \text{ and } C' \text{ are configurations of } M, \\ C = \theta q a^n \text{ for some } \theta \in \Sigma^* \text{ and } n > 0, \\ \text{where either the first symbol in } \theta \text{ is } u \text{ or } \theta = \epsilon \ \&\ u = \blacktriangledown\ \}.$$

We let $K_{u,q} = K \cap L_{u,q}$. We have that

$$K = \bigcup \{K_{u,q} : u \in \overline{\Sigma}, q \in Q\}.$$

Since there are only finitely many pairs $(u, q) \in \overline{\Sigma} \times Q$ and given that the family of one-counter languages is closed under union, it is sufficient to show that each of the sets $K_{u,q}$ is a one-counter language.

Let us therefore fix $u \in \overline{\Sigma}$ and $q \in Q$. If there is no septuple of the form $(u, q, -, -, -, -, -)$ in δ then $K_{u,q} = L_{u,q}$ is a regular language, and so is certainly one-counter. So we will assume that there is such a septuple. By the above comments, there is only one such septuple.

Let $CC' \in L_{u,q}$ where $C = \theta q a^n$ and $C' = \zeta p a^j$. In order for $C \vdash C'$ we must have that the septuple is of the form $(u, q, k, p, D_1, r, D_2)$ or $(u, q, k, r, D_1, p, D_2)$ for some k, r, D_1 and D_2 (with the possibility that $p = r$ here). We will show that, in all such cases, the language $K_{u,q}$ is a one-counter language by constructing a one-counter pushdown accepting it.

It is important to note that, as we read the a^n portion of the input, we can determine whether or not $kn \in \mathbb{N}$ without using the stack. If $k \in \mathbb{N}$ then the condition $kn \in \mathbb{N}$ is automatically satisfied; if $k \notin \mathbb{N}$, then the condition $kn \in \mathbb{N}$ is equivalent to $n \bmod \frac{1}{k}$ being zero. As we read the a^n portion of the input we update, in the state, the value of $i \bmod \frac{1}{k}$ at each stage (where a^i is the part of the a^n read so far). So, at the end of reading the a^n portion of the input, we will know whether or not $kn \in \mathbb{N}$. We will this that as read in what follows.

Let us first consider the possibility that we have a septuple of the form $(u, q, k, p, D_1, r, D_2)$ with $r \neq p$. In this case we have that

$$K_{u,q} = \{\theta q a^n \zeta p a^j : \ (i)\ \ \theta \text{ is of the form } u\eta \text{ (or } \theta = \epsilon \text{ if } u = \blacktriangledown); \\ (ii)\ \ \text{either } kn \notin \mathbb{N}; \\ \text{or } D_1 = N \text{ and } \theta \neq \zeta; \\ \text{or } D_1 = R \text{ and } \theta \neq u\zeta; \\ \text{or } kn \neq j\}.$$

The set of words of the form $\theta q a^n \zeta p a^j$, where θ is of the form $u\eta$ (or $\theta = \epsilon$ if $u = \blacktriangledown$), forms a regular language. If we can show that the set of words satisfying condition (ii) is a one-counter language then, as the intersection of a regular language and a one-counter language is a one-counter language, we will have established our result.

As we only have to verify that (at least) one of the four conditions in (ii) is satisfied, we can non-deterministically choose which one to check. Remember that we can check whether or not $kn \in \mathbb{N}$ without using the stack, and so we can focus on the remaining three conditions.

Let us assume that $D_1 = N$ and that we want to verify the condition that $\theta \neq \zeta$. If $\theta = \epsilon$ then we must have that $\zeta \neq \epsilon$ which is easily verified (just using the states of the machine); so we assume that $\theta \neq \epsilon$. When reading θ we put an a on the stack of our one-counter pushdown automaton for each symbol read in θ until we (non-deterministically) choose a symbol which will not match the corresponding symbol in ζ. We remember that symbol in the state of the pushdown automaton and do not do anything to the stack until we start reading ζ. At that stage we pop a symbol off the stack every time we read a symbol of ζ, and, when we reach an empty stack (i.e. when the bottom marker resurfaces on the stack) we verify that the symbol we read in ζ is different to the one we stored from the corresponding position in θ.

If $D_1 = R$ then the procedure is very similar except that we read the first u in θ without modifying the stack and then verify that the remainder of θ is not equal to ζ.

As far as condition $kn \neq j$ is concerned, we can verify this by putting a symbol on the stack of our one-counter pushdown automaton every time we read an a in a^n. If $k \in \mathbb{N}$ then we pop a symbol off the stack every time we read k a's in a^j, checking that we do not finish with an empty stack at the end of reading a^j (which would match kn against j). If $\frac{1}{k} \in \mathbb{N}$ then we pop $\frac{1}{k}$ symbols off the stack every time we read an a in a^j, again checking that we do not finish with an empty stack at the end of reading a^j (which, again, would match kn against j).

Let us next consider the possibility that we have a septuple of the form $(u, q, k, r, D_1, p, D_2)$ with $r \neq p$. In this case we have that

$$\begin{array}{lll} K_{u,q} = \{\theta q a^n \zeta p a^j : & (i) & \theta \text{ is of the form } u\eta \text{ (or } \theta = \epsilon \text{ if } u = \blacktriangledown); \\ & (ii) & \text{either } kn \in \mathbb{N}; \\ & & \text{or } D_2 = N \text{ and } \theta \neq \zeta; \\ & & \text{or } D_2 = R \text{ and } \theta \neq u\zeta; \\ & & \text{or } n \neq j\}. \end{array}$$

The argument here is very similar to the previous case. The only real difference is that, at the end, we are verifying that $n \neq j$ (as opposed to $kn \neq j$ as before). This time we put a symbol on the stack of our one-counter pushdown automaton every time we read an a in a^n and then pop a symbol off the stack every time we read an a in a^j, checking that we do not finish with an empty stack (which would match n against j).

Lastly there is the possibility that we have a septuple of the form

$$(u, q, k, p, D_1, p, D_2),$$

i.e. the case where $r = p$. In this case:

$$\begin{array}{rcll} K_{u,q} & = & \{\theta q a^n \zeta p a^j : & (i) \quad \theta \text{ is of the form } u\eta \text{ (or } \theta = \epsilon \text{ if } u = \blacktriangledown); \\ & & & (ii) \quad kn \in \mathbb{N}; \\ & & & (iii) \quad \text{either } D_1 = N \text{ and } \theta \neq \zeta; \\ & & & \qquad \text{or } D_1 = R \text{ and } \theta \neq u\zeta; \\ & & & \qquad \text{or } kn \neq j\} \\ \\ & \bigcup & \{\theta q a^n \zeta p a^j : & (i) \quad \theta \text{ is of the form } u\eta \text{ (or } \theta = \epsilon \text{ if } u = \blacktriangledown); \\ & & & (ii) \quad kn \notin \mathbb{N}; \\ & & & (iii) \quad \text{either } D_2 = N \text{ and } \theta \neq \zeta; \\ & & & \qquad \text{or } D_2 = R \text{ and } \theta \neq u\zeta; \\ & & & \qquad \text{or } n \neq j\}. \end{array}$$

The arguments that the two languages in this union are one-counter are essentially the same as those given above for the previous two cases (remembering that we can check whether or not $kn \in \mathbb{N}$ without using the stack) and we can again use the fact that the family of one-counter languages is closed under union to conclude that the language $K_{u,q}$ is a one-counter language in this last case as well. □

Our aim here is to show that the properties listed in Definition 12 are undecidable for one-counter languages. In order to do this we will need the following technical result:

Proposition 50. *The following problem is undecidable:*

Input: *a one-counter pushdown automaton N with an input alphabet Σ of size at least two such that either $L(N) = \Sigma^* - \Sigma^*\{\alpha\}\Sigma^*$ or else $L(N) = \Sigma^*$ for some word α such that α has length at least two and contains at least two different symbols.*

Output: *"yes" if $L(N) = \Sigma^* - \Sigma^*\{\alpha\}\Sigma^*$;*
"no" if $LN) = \Sigma^*$.

Proof. Our aim is to describe a language L over an alphabet Σ which is closed under taking factors and whose elements do not include a valid computation of a counter machine M (when reading a specified input β) as a factor. This way the language L will be equal to either $\Sigma^* - \Sigma^*\{\alpha\}\Sigma^*$ or Σ^*, where α is the computation path of M accepting β, depending on whether or not M accepts β. Note that the input β will be fixed in what follows.

Since we want L to be closed under taking factors we need to ensure that no factor of a word γ in L is a valid computation of M. We do this by checking that, whenever the initial configuration of M occurs in γ, a valid computation does not follow. Formally, we will consider the following three languages:

(i) $L_1 = \Sigma^* - \Sigma^*\{\beta q_0 a\}\Sigma^*$. This is the set of all words in Σ^* which do not contain the unique initial configuration of M.

(ii) $L_2 = \Sigma^* - \Sigma^*\{q_h a\}\Sigma^*$. This is the set of all words which do not contain the unique halting configuration of M.

(iii) L_3 which is the set of all words which are invalid as computations of M after every instance of the unique initial configuration of M (i.e. all words which do not contain a factor consisting of the unique initial configuration of M followed by a valid computation path ending in the unique halting configuration of M).

L_1 and L_2 are both regular languages and so are one-counter languages; we will now show that L_3 is a one-counter language as well.

The one-counter pushdown automaton accepting L_3 operates as follows. It scans its input doing nothing until it reads the unique initial configuration $\beta q_0 a$ of M (as β is fixed it can check whether or not it reads $\beta q_0 a$ in its states). At this point the machine continues reading the input and attempts to detect an invalid computation step of M (as in Proposition 49). If the machine does not find a factor which is an invalid computation step of M before reading the unique halting configuration $q_h a$ of M then the machine simply scans the rest of its input, doing nothing, and then rejects at the end.

If the one-counter pushdown automaton does find a factor which is an invalid computation step of M then it continues to scan its input until it finds another instance of the unique initial configuration of M (again, as β is fixed, this can be done using the states of the pushdown automaton) and then repeats the above process, accepting the input if and only if, after every instance of the initial configuration $\beta q_0 a$ of M, we do not reach the halting configuration $q_h a$ of M without finding an invalid computation step first. If, at any point, the pushdown automaton finds another instance of the initial configuration of M before an instance of the halting configuration then the pushdown automaton resets its state and attempts again to find an invalid computation step of M starting at the most recent initial configuration read (remember that the initial configuration $\beta q_0 a$ of M only occurs at the beginning of the computation of M with input β).

So $L = L_1 \cup L_2 \cup L_3$ is a one-counter language as the family of one-counter languages is closed under union. Now $L = \Sigma^*$ if and only if M rejects β and $L = \Sigma^* - \Sigma^*\{\alpha\}\Sigma^*$ (for suitable α) if and only if M accepts β. So, if we could distinguish between Σ^* and $\Sigma^* - \Sigma^*\{\alpha\}\Sigma^*$ for one-counter languages, then we could solve the halting problem for counter machines (and hence for Turing machines), a contradiction. □

The technical condition in Proposition 50 that α can be assumed to have length greater than two and to consist of at least two symbols is included only to facilitate the undecidability results that follow. In a similar manner we can establish:

Proposition 51. *The following problem is undecidable:*

Input: *a one-counter pushdown automaton N with an input alphabet Σ of size at least two such that either $L(N) = \Sigma^* - \{\alpha\}$ or else $L(N) = \Sigma^*$ for some α such that α has length at least two and contains at least two different symbols.*

Output: *"yes" if $L(N) = \Sigma^* - \{\alpha\}$;*
"no" if $L(N) = \Sigma^*$.

Having established Propositions 50 and 51 we can now prove our result:

Theorem 52. *All the properties listed in Definition 12 are undecidable for the family of one-counter languages.*

PROOF. Σ^* satisfies all the properties in Definition 12 but $K = \Sigma^* - \Sigma^*\{\alpha\}\Sigma^*$ (where α has length at least two and contains two different symbols) does not satisfy any of the conditions (UPP), (USP), (UFP), (IC), (CCS), (CC). These are reasonably clear. The word α is not a prefix, suffix or factor of any word in K, and so K does not satisfy (UPP), (USP) or (UFP). If $\alpha = \beta\gamma$ with $\beta \neq \epsilon \neq \gamma$ then $\beta, \gamma \in K$ but $\beta\gamma \notin K$; so K does not satisfy (IC) or (CC).

Given that α can be assumed to have two distinct symbols, we can write α in the form $a\delta b\zeta$ for some $a, b \in \Sigma$ with $a \neq b$ and $\delta, \zeta \in \Sigma^*$; if K satisfied (CCS) then, as $b\zeta a\delta \in K$, we would have that $\alpha = a\delta b\zeta \in K$, a contradiction. So all these conditions must be undecidable by Proposition 50.

The remaining properties are (DC), (RDC) and (LDC). If we could decide these then we would be able to distinguish between Σ^*, which satisfies all three properties, and $\Sigma^* - \{\alpha\}$, which doesn't satisfy any of them; for example, for any character x in Σ, $\alpha x \in \Sigma^* - \{\alpha\}$ and $x \in \Sigma^* - \{\alpha\}$ but deleting x from αx yields α which is not a member of $\Sigma^* - \{\alpha\}$), contradicting Proposition 51. □

Remark 53. Given Theorem 52 it is immediate that the problem of deciding any of these properties is undecidable for the family of context-free languages as well. Given our interest in word problems of groups, we have concentrated on the one-counter and context-free languages in this paper, but the argument could be applied to other families of languages as well. □

10. Word problems and decidability

We now turn our attention to word problems, i.e. those languages satisfying both the conditions (UPP) and (DC) in Theorem 13. Given Theorem 13 together with Proposition 46, we immediately have the following result:

Proposition 54. *The following decision problem is decidable:*

Input: *a finite automaton N.*

Output: *"yes" if $L(N)$ is the word problem of a group;*
"no" otherwise.

When we come to the one-counter languages, however, this problem becomes undecidable. This was shown for the family of context-free languages in [22] and, as we will see, the argument used there extends to the family of one-counter languages as well. This result does not follow immediately from Theorem 52; it is possible to have two undecidable problems whose conjunction is decidable. In order to prove the undecidability of this problem we need the concept of a *Hotz group* from [17]:

Definition 55. The *Hotz group* $H(G)$ of a grammar $G = (V, \Sigma, P, S)$ is the group with (group) presentation $\langle V \cup \Sigma : \{\alpha = \beta : (\alpha \to \beta) \in P\}\rangle$.

Hotz showed that the group $H(G)$ for a reduced context-free grammar G depends only on $L(G)$. We will also need the idea of a *collapsing group*:

Definition 56. The *collapsing group* $C(L)$ of a language $L \subseteq \Sigma^*$ is the group with (group) presentation $\langle \Sigma : \{\alpha = \beta : \alpha, \beta \in L\}\rangle$.

The following connection between these two concepts will play a central role in what follows:

Definition 57. A language $L \subseteq \Sigma^*$ is called a *language with Hotz isomorphism* if there exists a reduced grammar $G = (V, \Sigma, P, S)$ with $L = L(G)$ such that the collapsing group of L is isomorphic to $H(G)$.

It is known [8] that all context-free languages are languages with Hotz isomorphism. In fact it is shown in [5] that:

Theorem 58. *A language $L \subseteq \Sigma^*$ is a language with Hotz isomorphism if and only if the collapsing group $C(L)$ is finitely presentable.*

Remark 59. The collapsing group $C(L)$ of a language $L \subseteq \Sigma^*$, where the empty word ϵ lies in L, will have every word in L representing the identity element of $C(L)$ but it may have words outside L representing the identity element as well.

Let $\wp$ denote the (group) presentation

$$\langle\, \Sigma : \{\alpha = 1 : \alpha \in L\}\,\rangle.$$

If L is the word problem of some group K, then $\wp$ is a presentation for K and so K is isomorphic to $C(L)$. If L is not the word problem of a group then the word problem of the group with presentation $\wp$ must contain L as a proper subset.

In particular, if L is a context-free language which is the word problem of a group K, then $C(L)$ is isomorphic to K and we may obtain a finite presentation for K using the facts that K is isomorphic to $H(G)$ (where G is a context-free grammar generating L) and that the definition of $H(G)$ in Definition 55 is via a finite presentation. □

We are now in a position to prove our undecidability result:

Theorem 60. *The following decision problem is undecidable:*
Input: *a one-counter pushdown automaton* $N = (Q, \Sigma, \Gamma, \tau, s, A)$.
Output: *"yes" if* $L(N)$ *is the word problem of a group;*
"no" otherwise.

PROOF. Suppose we had an algorithm $\mathfrak{A}$ which could decide, given a one-counter pushdown automaton N, whether or not $L = L(N)$ was the word problem of some group G. We will show that one could then decide whether or not $L = \Sigma^*$ which is a contradiction by Theorem 2.

Since Σ^* is the word problem of the trivial group $\{1\}$, if $\mathfrak{A}$ outputs "no", then we have that $L \neq \Sigma^*$. On the other hand, since L is context-free, if $\mathfrak{A}$ outputs "yes", then we know that the corresponding group G has a context-free word problem (as one-counter languages are context-free), and so G is virtually free by Theorem 9. We can now obtain a finite presentation $\wp$ for G as in Remark 59 and then use the presentation $\wp$ to test the group G for triviality as in Theorem 6. Since G is trivial if and only if $L = \Sigma^*$ we now have an algorithm for determining whether or not $L = \Sigma^*$, a contradiction. □

Remark 61. Since every one-counter language is context-free, in that a one-counter pushdown automaton is a special case of a pushdown automaton, it immediately follows from Theorem 60 that there is no algorithm to decide whether or not $L(M)$ is the word problem of a group for a pushdown automaton M (as proved in [22]).

The proof given in Theorem 60 will, in fact, work for any family $\mathcal{F}$ of context-free languages where the universe problem is undecidable (provided that $\mathcal{F}$ is specified in such a way that a finite presentation for the Hotz group of any language L in $\mathcal{F}$ can be effectively determined). So, for example, this applies to the family of linear languages as well. In fact, it is a consequence of Theorem 10 that a group whose word problem is linear must be a finite group, and so the word problem would be a regular language; however the problem of deciding whether or not a linear language is regular is also undecidable. □

As we commented in Remark 4, a group is the syntactic monoid of its word problem so that Theorem 60 could be thought of as a decidability result about syntactic monoids in a particular case. It is natural to then ask what happens if we change the focus and ask about syntactic monoids in general. The following result shows that there is no algorithm for constructing a presentation for the syntactic monoid of a one-counter language even if we know that the monoid in question is finitely presented:

Proposition 62. *There is no algorithm to solve the following problem:*
Input: *a one-counter pushdown automaton* $M = (Q, \Sigma, \Gamma, \tau, s, A)$ *such that the syntactic monoid of* $L(M)$ *is finitely presented;*
Output: *a finite presentation for* $Synt(L)$.

PROOF. Suppose that there was such an algorithm $\mathfrak{A}$.

If we had a one-counter pushdown automaton $M = (Q, \Sigma, \Gamma, \tau, s, A)$ such that $L(M) = \Sigma^*$ or $L(M) = \Sigma^* - \{\alpha\}$ for some $\alpha \in \Sigma^*$ then $L = L(M)$ would be regular, and so the syntactic monoid $S = Synt(L)$ would be finite, and hence finitely presented. So we could apply the algorithm $\mathfrak{A}$ to get a finite presentation for S. We could then use a Todd-Coxeter process to enumerate the elements of the finite monoid S (see [25, 29] for example) to see whether or not S is trivial. By Remark 1 this would allow us to determine whether $L(M) = \Sigma^*$ or $L(M) = \Sigma^* - \{\alpha\}$, contradicting Proposition 51. □

Remark 63. We can prove other undecidability results for one-counter languages similar to Proposition 62. For example, suppose we had an algorithm $\mathfrak{A}$ that, given a one-counter pushdown automaton $N = (Q, \Sigma, \Gamma, \tau, s, A)$ accepting a language L and a word $\beta \in \Sigma^*$, could decide whether or not β was trivial in $Synt(L)$. We could apply $\mathfrak{A}$ to each generator of $Synt(L)$ (i.e. to each element of Σ) to see whether or not $Synt(L)$ was trivial, and we would again be able to distinguish between $L(N) = \Sigma^*$ and $L(N) = \Sigma^* - \{\alpha\}$. □

11. Deterministic context-free languages

We saw in Theorem 9 that a group has a context-free word problem if and only if it is virtually free. As we mentioned in Section 4, it is not hard to show that the word problem of a virtually free group is deterministic context-free. So we have the following well-known consequence of Theorem 9 (see Corollary 2.10 of [24] or Theorem 5.25 of [4] for example):

Theorem 64. *If a group G has a context-free word problem, then G has a deterministic context-free word problem.*

However, despite the fact that it is undecidable whether or not a context-free language is the word problem of a group, we will show that this problem becomes decidable if the language is deterministic context-free and is given by a deterministic pushdown automaton.

Theorem 65. *The following decision problem is decidable:*

Input: *a deterministic pushdown automaton $M = (Q, \Sigma, \Gamma, \tau, s, A)$.*

Output: *"yes" if $L(M)$ is the word problem of a group;*
"no" otherwise.

PROOF. If $\epsilon \notin L = L(M)$ then L is not the word problem of a group; so we check first that $\epsilon \in L$ (outputting "no" if that is not the case). We can therefore assume that $\epsilon \in L$ in what follows.

We convert our deterministic pushdown automaton to a reduced context-free grammar Γ such that $L(\Gamma) = L$ and then use the Hotz group construction in Remark 59 to write down a finite (group) presentation $\wp$ of the group $G = H(\Gamma)$ with generating set Σ.

As in Remark 59, if L is the word problem of a group, then it must be the word problem of G. If W is the word problem of G with respect to the generating set Σ, then the question is whether $L = W$ (in which case L is the word problem of a group) or else $L \subset W$ (in which case L is not the word problem of a group).

If L is the word problem of the group G then, as L is context-free, G must be virtually free. With this is mind, we start a process which we will refer to as Process 1.

Process 1 enumerates the finite-index subgroups of G and enumerates all presentations of these finite-index subgroups, checking each such presentation it generates to see if it is a natural presentation of a free group (i.e. a presentation with no relations). This is a semi-decision process; if G is virtually free, then we will eventually find such a presentation (i.e. a presentation for a free subgroup of finite index) and so we will know that G is virtually free, but, if G is not virtually free, then this process will not terminate.

At the same time we start Process 2. Process 2 takes the finite presentation $\wp$ and enumerates the words in Σ^* representing the identity element of G, checking each one it generates for membership of L. If Process 2 ever finds a word which is trivial in the group G but which is not a member of L then we terminate all the running processes and output "no". (If L were the word problem of a group then it has to be the word problem of G, in which case no word trivial in G could lie outside L.) Process 2 is also a semi-decision process; we continue enumerating words whilst we do not have an output of "no".

Eventually one of these two processes must terminate. If it is Process 1 then we know that the group G is virtually free and we now start Process 3. Process 3 uses the presentation $\wp$ of G and the finite-index free subgroup we have found to construct a deterministic pushdown automaton N which accepts the word problem of G. We can then test N for equivalence with M by Theorem 87 in [31] which says that the equivalence problem for deterministic pushdown automata is decidable. We halt all the processes and output the result of this equivalence test as our final output. Note that, if we reach Process 3, then Process 3 will always terminate.

Eventually either Process 2 terminates (and we output "no") or else Process 1 (and therefore Process 3) terminates. Thus we have an algorithm which outputs "yes" if $L(M)$ is the word problem of a group and outputs "no" if it is not, as required. □

Remark 66. As the reader will have seen, the use of the theorem of Sénizergues concerning the decidability of the equivalence problem for deterministic pushdown automata is a crucial component of the proof of Theorem 65. We are also using procedures such as the enumeration of finite-index subgroups of a group searching for one that is a free group. As it is (in general) undecidable as to whether or not a finitely presented group is virtually free, this procedure will not necessarily terminate, and our proof relies on the fact that we can run this in parallel with another semi-decision procedure and that, given our particular situation, one of these two procedures must terminate. Given that the two pro-

cedures we have used will not have computable time complexity in general, we are not claiming any degree of efficiency for this decision procedure, merely that the problem is decidable. □

It is interesting that problems that are undecidable for context-free, or even deterministic context-free, languages become decidable if we restrict ourselves to word problems of groups. An example is the inclusion problem which is undecidable even for (general) deterministic context-free languages:

Proposition 67. *The following decision problem is decidable:*

Input: *two pushdown automata* $M_1 = (Q_1, \Sigma, \Gamma_1, \tau_1, s_1, A_1)$ *and* $M_2 = (Q_2, \Sigma, \Gamma_2, \tau_2, s_2, A_2)$ *such that* $L(M_1)$ *and* $L(M_2)$ *are word problems of groups.*

Output: *"yes" if* $L(M_1) \subseteq L(M_2)$*;*
"no" otherwise.

PROOF. As in the proof of Theorem 65 we can convert our pushdown automata M_1 and M_2 to reduced context-free grammars Γ_1 and Γ_2 and then use the Hotz group construction to write down finite presentations $\wp_1$ and $\wp_2$ of the groups $G_1 = H(\Gamma_1)$ and $G_2 = H(\Gamma_2)$ (over the generating set Σ in each case).

Since $L(M_1)$ and $L(M_2)$ are word problems of groups, they are the word problems of G_1 and G_2 respectively. These groups are both virtually free and so have decidable word problems (and we may use the argument from Theorem 65 to construct presentations for G_1 and G_2 such that we have actual algorithms for solving the word problem of each group).

We now note that $L(M_1) \subseteq L(M_2)$ means that the word problem of G_1 is a subset of the word problem of G_2, which is equivalent to saying that G_2 is a homomorphic image of G_1. We may check whether or not this is true by seeing if every relation in $\wp_1$ also holds in $\wp_2$ (using the decidability of the word problem for G_2). If so, then we output "yes"; otherwise we output "no". □

Acknowledgments

This paper was completed whilst the second author was on study leave from the University of Leicester and he would like to acknowledge the help and support of the university in this respect. The careful reading of the paper by the referees and their constructive comments helped improve the exposition and were very much appreciated. The second author would also like to thank Hilary Craig for all her help and encouragement.

References

[1] Anisimov, V. A., 1971. The group languages. Kibernetika 4, 18–24.

[2] Autebert, J. M., Boasson, L., Sénizergues, G., 1987. Groups and NTS languages. Journal of Computer and System Sciences 35, 243–267.

[3] Berstel, J., 1979. Transductions and Context-free Languages. Teubner.

[4] Chiswell, I., 2009. A course in formal languages, automata and groups. Universitext. Springer-Verlag London, Ltd., London.

[5] Diekert, V., Möbus, A., 1989. Hotz-isomorphism theorems in formal language theory. Informatique Théorique et Applications 23, 29–43.

[6] Diekert, V., Weiß, A., 2013. Context-free groups and Bass-Serre theory, arXiv:1307.8297.

[7] Dunwoody, M. J., 1985. The accessibility of finitely presented groups. Inventiones Mathematicae 81, 449–457.

[8] Frougny, C., Sakarovitch, J., Valkema, E., 1982. On the Hotz-group of a context-free grammar. Acta Mathematica 18, 109–115.

[9] Gilman, R. H., 1996. Formal languages and infinite groups. In: Geometric and computational perspectives on infinite groups (Minneapolis, MN and New Brunswick, NJ, 1994). Vol. 25 of DIMACS Ser. Discrete Math. Theoret. Comput. Sci. Amer. Math. Soc., Providence, RI, pp. 27–51.

[10] Herbst, T., 1991. On a subclass of context-free groups. Informatique Théorique et Applications 25, 255–272.

[11] Herbst, T., Thomas, R. M., 1993. Group presentations, formal languages and characterizations of one-counter groups. Theoretical Computer Science 112, 187–213.

[12] Higman, G., 1952. Ordering by divisibility in abstract algebras. Proceedings of the London Mathematical Society 2, 326–336.

[13] Holt, D. F., Eick, B., O'Brien, E. A., 2005. Handbook of computational group theory. Discrete Mathematics and its Applications, Chapman & Hall.

[14] Holt, D. F., Owens, M. D., Thomas, R. M., 2008. Groups and semigroups with a one-counter word problem. Journal of the Australian Mathematical Society 85, 197–209.

[15] Holt, D. F., Rees, S., Röver, C. E., 2017. Groups, Languages and Automata. Vol. 88 of London Mathematical Society Student Texts. Cambridge University Press, Cambridge.

[16] Hopcroft, J. E., Ullman, J. D., 1979. Introduction to Automata Theory, Languages and Computation. Addison-Wesley.

[17] Hotz, G., 1980. Eine neue Invariante für Kontextfreie Sprachen. Theoretical Computer Science 11, 107–116.

[18] Ibarra, O. H., 1979. Restricted one-counter machines with undecidable universe problems. Mathematical Systems Theory 13, 181–186.

[19] Ito, M., 2004. Algebraic Theory of Automata & Languages. World Scientific Press.

[20] Johnson, D. L., 1997. Presentations of groups, 2nd Edition. Vol. 15 of London Mathematical Society Student Texts. Cambridge University Press, Cambridge.

[21] Jones, S. A. M., Thomas, R. M., 2015. Formal languages and group theory. In: Campbell, C. M., Quick, M. R., Robertson, E. F., Roney-Dougal, C. M. (Eds.), Proceedings of Groups St Andrews 2013. London Mathematical Society Lecture Note Series 422, Cambridge University Press, pp. 306–323.

[22] Lakin, S. R., Thomas, R. M., 2009. Space complexity and word problems of groups. Groups-Complexity-Cryptology 1, 261–273.

[23] Muller, D. E., Schupp, P. E., 1983. Groups, the theory of ends, and context-free languages. Journal of Computer and System Sciences 26, 295–310.

[24] Muller, D. E., Schupp, P. E., 1985. The theory of ends, pushdown automata, and second order logic. Theoretical Computer Science 37, 51–75.

[25] Neumann, B. H., 1967. Some remarks on semigroup presentations. Canadian Journal of Mathematics 19, 1018–1026.

[26] Parkes, D. W., Thomas, R. M., 2000. Syntactic monoids and word problems. Arabian Journal for Science and Engineering 25, 81–94.

[27] Parkes, D. W., Thomas, R. M., 2002. Groups with context-free reduced word problem. Communications in Algebra 30, 3143–3156.

[28] Rampersad, N., Shallit, J., Xu, Z., 2012. The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages. Fundamenta Informaticae 116, 223–236.

[29] Robertson, E. F., Ünlü, Y., 1993. On semigroup presentations. Proceedings of the Edinburgh Mathematical Society 36, 55–68.

[30] Rotman, J. L., 1995. An introduction to the theory of groups. Springer-Verlag.

[31] Sénizergues, G., 2001. $L(A) = L(B)$? Decidability results from complete formal systems. Theoretical Computer Science 251, 1–166.

[32] Sims, C. C., 1994. Computation with finitely presented groups. Encyclopaedia of Mathematics and its Applications 48, Cambridge University Press.

[33] Stewart, I. A., Thomas, R. M., 1999. Formal languages and the word problem for groups. In: Campbell, C. M., Robertson, E. F., Ruškuc, N., Smith, G. C. (Eds.), Groups St Andrews 1997 in Bath, Volume 2. London Mathematical Society Lecture Note Series 261, Cambridge University Press, pp. 689–700.

[34] Valk, R., Vidal-Naquet, G., 1981. Petri nets and regular languages. Journal of Computer and System Sciences 23, 299–325.