ON THE WORD PROBLEM FOR TENSOR PRODUCTS AND AMALGAMS OF MONOIDS

Dedicated to the Memory of Marcel-Paul Schützenberger

JEAN-CAMILLE BIRGET*

Department of Computer Science
Dalhousie University
Halifax, Canada

STUART W. MARGOLIS*

Department of Mathematics and Computer Science
Bar-Ilan University
52900 Ramat Gan
Israel
E-mail: stwm@math.huji.ac.il

JOHN MEAKIN*

Department of Mathematics and Statistics
University of Nebraska
Lincoln, Nebraska 68588-0323
USA
E-mail: jmeakin@mathstat.unl.edu

Communicated by J.-E. Pin
Received 30 June 1996

AMS Mathematics Subject Classification: 20M05, 20E06, 20F10

We prove that the word problem for the free product with amalgamation $S *_U T$ of monoids can be undecidable, even when S and T are finitely presented monoids with word problems that are decidable in linear time, the factorization problems for U in each of S and T, as well as other problems, are decidable in polynomial time, and U is a free finitely generated unitary submonoid of both S and T. This is proved by showing that the equality problem for the tensor product $S \otimes_U T$ is undecidable and using known connections between tensor products and amalgams. We obtain similar results for semigroups, and by passing to semigroup rings, we obtain similar results for rings as well. The proof shows how to simulate an arbitrary Turing machine as a communicating pair of two deterministic pushdown automata and is of independent interest. A similar idea is used in a paper by E. Bach to show undecidability of the tensor equality problem for modules over commutative rings.

1. Introduction

Amalgamation theory goes back to the work of Schreier [23]. He showed that the variety of groups has what is now known as the strong amalgamation property:

*All authors' research supported in part by NSF grant DMS-9203981 and the Center for Communication and Information Science, University of Nebraska-Lincoln.

International Journal of Algebra and Computation Vol. 9, Nos. 3 & 4 (1999) 271–294

given groups G and H with common subgroup U, we can find a group A containing both G and H and such that the intersection of G and H within A is precisely U. An example of Kimura [14] showed that semigroups do not even satisfy the weak amalgamation property: there are semigroups S and T with common subsemigroup $U = S \cap T$, such that there is no embedding of $S \cup T$ into any semigroup. On the other hand, T. E. Hall [8] proved that the variety of inverse semigroups does have the strong amalgamation property. We recall some basic definitions. An *amalgam* $\mathcal{A}$ of semigroups is given by a semigroup U together with embeddings $\phi : U \to S$ and $\psi : U \to T$ into disjoint semigroups S and T. We write this as $\mathcal{A} = [S, T; U; \phi, \psi]$. It is customary and convenient to think of the underlying set of U as included in the underlying sets of both S and T, and that $S \cap T = U$. In this situation we write $\mathcal{A} = [S, T; U]$. The amalgam $\mathcal{A}$ embeds into a semigroup A if there are embeddings $\lambda : S \to A$ and $\mu : T \to A$ such that $\phi\lambda = \psi\mu : U \to A$. We say that $\mathcal{A}$ strongly embeds into A if we further have that $S\lambda \cap T\mu = U\phi\lambda$. There are two general methods for studying the embeddability problem. In the first, one studies the free product with amalgamation $S *_U T$ of the amalgam $\mathcal{A}$. This is the quotient of the free product $S * T$ by the smallest congruence $\rho_{\mathcal{A}}$ that identifies $u\phi$ with $u\psi$ for all $u \in U$. A standard universal argument shows that $\mathcal{A}$ (strongly) embeds into some semigroup if and only if $\mathcal{A}$ (strongly) embeds into its free product with amalgamation. We define the *word problem* for $\mathcal{A}$ to be the word problem for the congruence $\rho_{\mathcal{A}}$ (i.e. the word problem of the semigroup determined by $\mathcal{A}$).

There are similar definitions in the category of monoids, where of course the embeddings must preserve identity elements. If S is a semigroup, let S^I be the monoid defined by adjoining a new identity element to S, even if S already has an identity. If $\phi : S \to T$ is a semigroup morphism, let $\phi^I : S^I \to T^I$ be the monoid morphism that extends ϕ by sending the identity element of S^I to the identity element of T^I. In this way, every amalgam $\mathcal{A} = [S, T; U]$ of semigroups gives rise to an amalgam $\mathcal{A}^I = [S^I, T^I; U^I]$ of monoids. The following lemma is easy to verify either directly or by noticing that the assignment of S to S^I and ϕ to ϕ^I is the left adjoint to the forgetful functor from monoids to semigroups. The lemma will allow us to work throughout the paper in the category of monoids, and this will help simplify some statements.

Lemma 1.1. *Let $\mathcal{A} = [S, T; U]$ be an amalgam of semigroups. Then $\mathcal{A}$ embeds into a semigroup A if and only if $\mathcal{A}^I$ embeds into A^I. Furthermore, $(S *_U T)^I = S^I *_{U^I} T^I$.*

The following corollary of lemma 1.1 is immediate.

Corollary 1.1. *Let $\mathcal{A}$ be an amalgam of semigroups. Then the word problem for $\mathcal{A}$ is decidable if and only if the word problem for $\mathcal{A}^I$ is decidable.*

Schreier showed that in the case of a free product with amalgamation of groups, every element has a normal form from which one can readily deduce the strong amalgamation property and many other properties of the free product with amalgamation. Howie [12] began the study of semigroup amalgams by proving that if

U is a unitary subsemigroup of both S and T, then $\mathcal{A}$ strongly embeds into its free product with amalgamation; here we use the following definition.

Definition. A subsemigroup U of a semigroup S is *unitary* if $u, us \in U$ implies $s \in U$ and $u, su \in U$ implies $s \in U$ for all $s \in S$.

See [13, 10] for generalizations of this last result. Howie's Theorem is proved by using combinatorial arguments to show that the congruence $\rho_{\mathcal{A}}$ restricted to $(S \cup T) \times (S \cup T)$ is the identity relation.

It can be seen that if $\mathcal{A} = [G, H; U]$ is an amalgam of groups, then if the word problems for G and H are decidable and the generalized word problems for U in both G and H are decidable, then the word problem for $\mathcal{A}$ is decidable. See [17] for details. The main result of this paper, Theorem 1.2 (see also Theorem 2.5), states that for an amalgam $\mathcal{A}$ of semigroups, the word problem for $\mathcal{A}$ may be undecidable, even if S, T, and U satisfy very restrictive decidability conditions and U is a unitary subsemigroup of both S and T.

The second main method in studying an amalgam $\mathcal{A}$ is to try to embed $\mathcal{A}$ into a concrete semigroup such as a semigroup of functions. This method was pioneered by T. E. Hall [9] in the 1970's. He showed that various extension properties concerned with the representation theory of semigroups by functions ensure that an amalgam embeds into some semigroup.

In the 1980's, it was discovered that there was a deep connection between the homological algebra of monoid acts and amalgamation theory. We refer the reader to the survey [13] for more details and references. We summarize here the definitions and results that we will need in this paper. Let S and T be monoids and let U be a common submonoid of S and T. The *tensor product* $S \otimes_U T$ is the set defined to be the quotient of $S \times T$ by the equivalence relation τ generated by the relation:

$$\{(su, t), (s, ut) | s \in S, t \in T, u \in U\}.$$

We write $s \otimes t$ for the equivalence class $(s, t)\tau$.

The *tensor equality problem* is the equivalence problem for τ. That is, given $(s, t), (s', t') \in S \times T$, we want to know if $s \otimes t = s' \otimes t'$. One of our results shows that the tensor equality problem can be undecidable even when the factors have easily decidable word problems. A similar result for tensor products of modules over commutative rings has been proved by E. Bach.

The following lemma gives an explicit, though complicated condition for testing equality of tensors in terms of a chain of equations. It is a special case of the general result of [5] that gives a similar result for arbitrary semigroup acts.

Lemma 1.2. *Let S and T be monoids with U a common submonoid. Then $s \otimes t = s' \otimes t'$ if and only if there exists $n > 0$, $s_1, \ldots, s_n \in S$, $t_2, \ldots, t_n \in T$, $u_1, \ldots, u_n \in U$, $v_1, \ldots, v_n \in U$ such that:*

$s = s_1u_1, : s_1v_1 = s_2u_2, : s_2v_2 = s_3u_3, \ \ldots \ , s_nv_n = s'$ *and*

$u_1t = v_1t_2, : u_2t_2 = v_2t_3, \ \ldots \ , u_nt_n = v_nt'$.

Lemma 1.2 points out why there may be difficulty in deciding equality in tensor products since there is no *a priori* bound on the number n in the lemma. Our main

theorem shows that this is indeed a sufficient cause for undecidability. However for tensor products of groups, we can always take $n = 1$.

Corollary 1.2. *Let G and H be groups with a common subgroup U. Then $g \otimes h = g' \otimes h'$ in $G \otimes_U H$ if and only if there is $u \in U$, such that $gu = g', h = uh'$.*

Proof. The condition is clearly sufficient. Conversely, assume that $g \otimes h = g' \otimes h'$. By Lemma 1.2, there is an $n > 0$ and elements that satisfy the equations listed there. Using the notation of Lemma 1.2, let $u = u_1^{-1} v_1 u_2^{-1} v_2 \dots u_n^{-1} v_n$. A simple group-theoretic manipulation of the equations in Lemma 1.2 shows that $gu = g', h = uh'$ as desired. □

Corollary 1.3. *Let G and H be groups with a common subgroup U. If the word problems for G and H are decidable and the generalized word problem of U in each of G and H is decidable, then the equality problem for the tensor product $G \otimes_U H$ is decidable.*

Proof. It follows from the previous corollary that $g \otimes h = g' \otimes h'$ if and only if $g^{-1}g' = hh'^{-1}$ and this common element is in U. □

It is also known that for groups, every element of the free product with amalgamation has a unique representation as an iterated tensor. See [24], Theorem 2 of Chap. 1 or [26] which uses the language of pregroups. Dekov [6, 7] proves a similar result for an amalgam of semigroups $[S, T; U]$, where U is a unitary subsemigroup of both S and T.

More generally, Renshaw [19, 20] shows how to construct the free product with amalgamation of a monoid amalgam as a direct limit of certain quotients of iterated tensor products. The following theorem follows from the work of Renshaw [19]; it enables us to apply results about tensor products to amalgams.

Theorem 1.1. *Let $[S, T; U]$ be an amalgam of monoids where U is a unitary submonoid of S and T. Then the map $s \otimes t \longmapsto st$ is an embedding of $S \otimes_U T$ into $S *_U T$.*

To prove our main result, we will reduce the membership problem for the language of a Turing machine to the tensor equality problem for an associated tensor product of semigroups. This will prove that the equality problem for tensor products of semigroups with decidable word problems can be undecidable. Theorem 1.1 then extends our result to the word problem for amalgams of semigroups.

Besides the word problem, there are other decision problems that will appear in this paper, which we now define.

Definitions. For a submonoid U of a finitely generated monoid $S = \langle X \rangle$, the *generalized word problem* is the following: on input $s \in X^*$, decide whether the element of S represented by s belongs to U.

For a finitely generated submonoid $U = \langle V \rangle$ of a finitely generated monoid $S = \langle X \rangle$, the *right factorization problem* is the following: on input words $x \in X^*$ and $u \in V^*$, decide whether $x_S = s \cdot u_S$, for some $s \in S$; here x_S and u_S are the

elements of S represented by the words x and u respectively. In this case, we say that u is a *U-suffix* of x.

The *left factorization problem* for U in S, and the notion of *U-prefix*, are defined similarly.

For two finitely generated monoids $S = \langle X \rangle$ and $T = \langle Y \rangle$, with a common submonoid U, the *one-step tensor equality problem* is as follows: on input (x, y), $(x', y') \in X^* \times Y^*$, decide whether there exists $u \in U$ such that $x_S = x'_S \cdot u$, $u \cdot y_T = y'_T$ or $x'_S = x_S \cdot u$, $u \cdot y'_T = y_T$. Again, x_S is the element of S represented by the word x, y_T is the element of T represented by the word y, etc.

Finally, our main result is the following:

Theorem 1.2 (Main Theorem). *There exist finitely presented monoids S and T with a common finitely generated submonoid U, with the following properties:*

- *U is a free monoid, and is a unitary submonoid of S and T: thus S and T embed into $S *_U T$, and their intersection in $S *_U T$ is U.*
- *The following problems are decidable:*
  - *The word problems of S and of T.*
  - *The generalized word problems of U relative to S or to T.*
  - *The right (left) factorization problem of U in S (respectively T).*
  - *The one-step tensor equality problem in $S \otimes_U T$.*

  Moreover, these problems can be decided by linear-time deterministic algorithms, except for the one-step tensor equality problem, for which the time is quadratic.
- *The tensor equality problem for $S \otimes_U T$ and the word problem for the free product with amalgamation $S *_U T$ are undecidable.*
- *All these undecidability, decidability and complexity properties are invariant under change of finite generating sets in S, T and U.*

2. Simulating Turing Machines by Tensor Products of Monoids

In this section, we will show how any deterministic one-tape Turing machine $\mathcal{M}$ can be simulated by a tensor product $R_\mathcal{M} \otimes_U L_\mathcal{M}$, where $R_\mathcal{M}$ and $L_\mathcal{M}$ are finitely presented monoids whose word problems are decidable, and where U is a finitely generated free monoid that is embedded into $R_\mathcal{M}$ and $L_\mathcal{M}$ as a unitary submonoid. The monoid $R_\mathcal{M}$ (respectively $L_\mathcal{M}$) simulates the right (respectively left) moving transitions of $\mathcal{M}$; essentially, it represents a deterministic pushdown automaton. The common submonoid U can be thought of as a "communication wire" that allows for configurations to be shipped back and forth between $R_\mathcal{M}$ and $L_\mathcal{M}$. The unitary property will ensure that information in the communication wire cannot be corrupted by the actions of $R_\mathcal{M}$ and $L_\mathcal{M}$. The reader may be able to use this intuition when studying the technical descriptions below.

The construction presented here is in the spirit of the construction of E. Bach [1] in his work on the tensor equality problem for tensor products of modules over commutative rings. It is easy to modify Bach's construction to show undecidability of the tensor equality problem for monoid acts. However, in the setting described

above, significant additional technicalities are needed to ensure that the common submonoid U is unitary in both $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ and hence to show undecidability of the word problem for the corresponding amalgamated free product.

2.1. *Turing machines*

We assume that the reader is familiar with the basics of Turing machines (see [11, 18] for example). For our purposes here, it will be convenient to use a slight variation on the usual definition of a Turing machine [2]. It is routine to check that a "standard" Turing machine (as in [11]) can be simulated by the model described here, and vice versa, without increase in computational complexity (see the remark below for an outline of the proof).

A one-tape deterministic Turing machine $\mathcal{M}$ is given by the following data:

- A finite set $Q = \overrightarrow{Q} \cup \overleftarrow{Q}$ of states. $\overrightarrow{Q}$ ($\overleftarrow{Q}$) is the set of right (left) moving states. We assume that $\overrightarrow{Q} \cap \overleftarrow{Q} = \emptyset$.
- A finite input alphabet Σ.
- A finite total alphabet Γ, with $\Sigma \subseteq \Gamma$.
- The left endmarker $\rhd$ and the right endmarker $\lhd$.
- The start state $\overrightarrow{q_0} \in \overrightarrow{Q}$.
- The final state $q_f \in Q$.
- The transition function δ, to be described in detail below.

We assume that $\rhd$, $\lhd \notin Q \cup \Gamma$ and that Q and Γ are disjoint. We will denote an arbitrary member of $\overrightarrow{Q}$ (respectively $\overleftarrow{Q}$) by $\overrightarrow{q}$ (respectively $\overleftarrow{q}$).

We now describe the transition function; δ is a finite set of transitions of the following six types.

(1) A *right shift* transition is of the form $\overrightarrow{q}a \to bp$, where $a, b \in \Gamma$ and $p \in Q$.

(2) A *left shift* transition is of the form $a\overleftarrow{q} \to pb$, where $a, b \in \Gamma$ and $p \in Q$. At the right endmarker, the machine has the ability to turn around or extend or shrink the tape.

(3) An *insertion* transition (always at the right endmarker) is of the form $\overrightarrow{q} \lhd \to a\overleftarrow{p} \lhd$, where $a \in \Gamma$.

(4) A *deletion transition* (always at the right endmarker) is of the form $a\overrightarrow{q} \lhd \to \overleftarrow{p} \lhd$. Finally, the turn transitions at the endmarkers are of the forms:

(5) $\overrightarrow{q} \lhd \to \overleftarrow{p} \lhd$, or

(6) $\rhd \overleftarrow{q} \to \rhd \overrightarrow{p}$.

It is important to notice that $\mathcal{M}$ goes to a left moving state after extending the tape on the right, so there will be no infinite sequences of successive right moving transitions in the machine. We do not allow moves over the endmarkers.

Transitions of the form (1), (3), (4), (5) have a right-moving state on their left side, and are called *right-moving transitions*; transitions of the form (2) and (6) have a left-moving state on their left side, and are called *left-moving transitions.*

We assume that the start state does not appear on the right side of any transition and that the final state does not appear on the left side of any transition. We assume also that the Turing machine is deterministic, i.e. no two transitions have the same left side, and we do not allow transitions with left sides $\vec{q} \triangleleft$ and $a\vec{q} \triangleleft$ for the same state $\vec{q}$ and any $a \in \Gamma$.

We now describe how our Turing machine works. A *configuration* of $\mathcal{M}$ is a word of the form $\triangleright uqv \triangleleft$, where $u, v \in \Gamma^*$ and $q \in Q$. A *start configuration* is a configuration of the form $\triangleright q_0 w \triangleleft$, where $w \in \Sigma^*$. The unique *accept configuration* is $\triangleright q_f \triangleleft$. We are thus making the assumption that the machine cleans up its tape before accepting.

We think of the configuration $\triangleright uqv \triangleleft$ as standing for: uv is on the tape, $\mathcal{M}$ is in state q and the head of $\mathcal{M}$ is *between* the last letter of u and the first letter of v. If $q \in \vec{Q}$, then $\mathcal{M}$ is reading the first letter of v, and if $q \in \overleftarrow{Q}$, then $\mathcal{M}$ is reading the last letter of u. This is a difference from the standard picture (as in [11]), where one thinks of the read head as always reading the first letter of v.

Let $u = a_1 \dots a_i$ and $v = a_{i+1} \dots a_n$. If $\vec{q}\, a_{i+1} \to bp$ is a transition of $\mathcal{M}$, then we write $\triangleright u\vec{q}v \triangleleft \ \to \ \triangleright ubpa_{i+2} \dots a_n \triangleleft$. If $a_i \overleftarrow{q} \to p'b'$ is a transition of $\mathcal{M}$, then we write $\triangleright u\overleftarrow{q}v \triangleleft \ \to \ \triangleright a_1 \dots a_{i-1} p'b'v \triangleleft$. We make similar definitions in the case of endmarker transitions. This defines the one-step derivation relation of $\mathcal{M}$. The derivation relation is the reflexive transitive closure $\overset{*}{\to}$ of $\to$. The language of $\mathcal{M}$ is $L(\mathcal{M}) = \{w \in \Sigma^* | \triangleright q_0 w \triangleleft \overset{*}{\to} \triangleright q_f \triangleleft\}$.

Remark. The Turing machines that we use here were chosen for their convenience for producing word problem. They differ from the more standard Turing machine (as in [11]) in two ways:

(1) They use a finite tape of varying length (as opposed to a tape containing an infinite sequence of blank symbols on one side).

(2) In a configuration, the head is between two adjacent cells (instead of being on a cell).

Finite tapes that can expand and shrink is not a new idea. Usually, the proof of the Markov–Post Theorem (that there exists finitely presented semigroups with undecidable word problem) uses such Turing machine (see e.g. [18, 20]). It is easy to see that such Turing machines accept the same languages as the standard ones: when a standard machine prints on blank cells, the new machine simulates this by insertion transitions, and vice versa.

Placing the head between cells rather than on a cell is also a convenience for manufacturing word problems from Turing machines. The two types of machines can simulate each other, by doing a zig-zag movement (so as to read the letters in the two neighboring cells, and remember them in the state); this increases the number of states and the time complexity at most linearly.

2.2. *Monoids associated with a Turing machine*

In this section, we will associate a number of finitely presented monoids with a given Turing machine $\mathcal{M}$. The most straightforward of such monoids is the Markov–

Post monoid, whose relations are just the rules of $\mathcal{M}$. It is well known that, for a deterministic Turing machine $\mathcal{M}$, the membership problem for $L(\mathcal{M})$ is reducible to the word problem for the Markov–Post monoid of $\mathcal{M}$.

We define the *Markov–Post* monoid of $\mathcal{M}$ to be the (finitely presented) monoid $S_{\mathcal{M}}$ with generators $Q \cup \Gamma \cup \{\rhd\ ,\ \lhd\}$ and relations $\{\alpha = \beta \mid \alpha \to \beta$ is a transition of $\mathcal{M}\}$. The following well known result states that the membership problem for $L(\mathcal{M})$ is reducible to the word problem for $S_{\mathcal{M}}$. The proof of this theorem can be found in [18]. It should be noted that the proof carries through for our model of Turing machine.

The special word problem of $S_{\mathcal{M}}$ is to decide, on input $x \in A^*$, whether $\rhd uqv \lhd\ =\ \rhd q_f \lhd$ in $S_{\mathcal{M}}$.

Theorem 2.1. (A. Markov, E. Post, 1947). *Let $\mathcal{M}$ be a one tape deterministic Turing machine. Assume that the start state of $\mathcal{M}$ does not appear on the right hand side of any transition, the accept state does not appear on the left side of any transition, and the only accepting configuration is $\rhd q_f \lhd$. Then for all $q \in Q$ and for all $u, v \in \Gamma^*$: $\rhd uqv \lhd \overset{*}{\to} \rhd q_f \lhd$ in $\mathcal{M}$ if and only if $\rhd uqv \lhd\ =\ \rhd q_f \lhd$ in $S_{\mathcal{M}}$. Thus, the membership problem of $L(\mathcal{M})$ reduces to the special word problem of $S_{\mathcal{M}}$, and the reduction is computable in linear time and is one-to-one.*

We next introduce a monoid $L_{\mathcal{M}}$ that models the left moves of $\mathcal{M}$, and a monoid $R_{\mathcal{M}}$ that models the right moves of $\mathcal{M}$, and a common submonoid U that models a communication channel between them. We will think of a Turing machine as performing alternating sequences of right moves and sequences of left moves. Of course, in general, the number of alternations may be infinite (this leads to the undecidability of the Halting Problem) although by the way we have defined Turing machines, there are no infinite sequences of successive right moves nor of successive left moves. We will decompose an arbitrary Turing machine $\mathcal{M}$ into a machine that performs only the right moves of $\mathcal{M}$ and one that only performs the left moves of $\mathcal{M}$. We will design a communication channel to pass information between these two machines when $\mathcal{M}$ is about to switch from right moves to left moves or vice versa. Fortunately (if you like undecidable problems!), we can implement this idea by the tensor product of two finitely presented monoids with decidable word problems over a common finitely generated free unitary submonoid U that acts as the communication channel.

We prove that the word problems for $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ are both decidable with low complexity and that the factorization problems for U in $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ are also decidable. We will then reduce the membership problem for $L(\mathcal{M})$ to the word problem of the tensor product $R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$.

As is usual in this type of reduction argument, we design the monoids to mimic certain moves of $\mathcal{M}$. The hard part in the reduction occurs for two reasons. First of all, we must deal with the fact that moves in a Turing machine are directed, while monoid congruences are symmetric. Secondly, we must deal with all possible words in the generators of the monoid, not just those that represent configurations of the machine.

Let $\mathcal{M}$ be a deterministic one-tape Turing machine. We define the monoids $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ by the following finite presentations. Two new symbols, θ and ζ are needed to implement our "communication wire". For convenience, we group the relations into three classes.

Presentation of $R_{\mathcal{M}}$

$R_{\mathcal{M}}$ has generators $Q \cup \Gamma \cup \{\rhd , \lhd\} \cup \{\theta, \zeta\}$, and relations:

(1) Right moving relations: $\{\alpha = \beta \mid \alpha \to \beta$ is a right move or right endmarker transition$\}$.
(2) θ-insertion and θ-deletion: $\{\zeta\theta = \zeta,\ \zeta = \theta\zeta\}$.
(3) Commutation relations: $\{\zeta a = a\zeta \mid a \in Q \cup \Gamma \cup \{\rhd , \lhd\}\}$.

Presentation of $L_{\mathcal{M}}$

$L_{\mathcal{M}}$ has generators $Q \cup \Gamma \cup \{\rhd , \lhd\} \cup \{\theta, \zeta\}$, and relations:

(1) Left moving relations: $\{\alpha = \beta \mid \alpha \to \beta$ is a left move or a left endmarker transition$\}$.
(2) θ-insertion and θ-deletion: $\{\zeta\theta = \zeta,\ \zeta = \theta\zeta\}$.
(3) Commutation relations: $\{\zeta a = a\zeta \mid a \in Q \cup \Gamma \cup \{\rhd , \lhd\}\}$.

The submonoid U

Let U_L (respectively U_R) be the submonoid of $L_{\mathcal{M}}$ (respectively $R_{\mathcal{M}}$) generated by the finite set

$$\{\theta a\theta \mid a \in Q \cup \Gamma \cup \{\rhd , \lhd\}\}.$$

We will see soon that U_L and U_R are free monoids over these generators, hence they are isomorphic; usually, both monoids will be called U.

2.3. *Properties of the monoids associated with a Turing machine*

Lemma 2.1. *The monoids U_L and U_R are both free monoids on the given set of generators; so U_L and U_R are isomorphic (and usually, we will call both of them U). Moreover, U_L (and U_R) is a unitary submonoid of $L_{\mathcal{M}}$ (respectively $R_{\mathcal{M}}$).*

If $w \in \{\theta a\theta \mid a \in Q \cup \Gamma \cup \{\rhd , \lhd\}\}^+$, then the congruence class of w in both $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ is the singleton set $\{w\}$.

Proof. We will work in $L_{\mathcal{M}}$, the proof for $R_{\mathcal{M}}$ being similar. First note that no relation in the presentation defining $L_{\mathcal{M}}$ applies to any word in $\{\theta a\theta | a \in Q \cup \Gamma \cup \{\rhd , \lhd\}\}^+$. This gives the last claim of the lemma and implies that the natural map maps this submonoid injectively into $L_{\mathcal{M}}$. It is clear that $\{\theta a\theta | a \in Q \cup \Gamma \cup \{\rhd , \lhd\}\}^+$ is a free submonoid of the free monoid on the generating set of $L_{\mathcal{M}}$, since $\{\theta a\theta | a \in Q \cup \Gamma \cup \{\rhd , \lhd\}\}$ is a biprefix code [15]. This proves that U_L is a free submonoid $L_{\mathcal{M}}$, freely generated by $\{\theta a\theta | a \in Q \cup \Gamma \cup \{\rhd , \lhd\}\}$ as desired.

Finally, note that any word that is a prefix (suffix) of a word in $\{\theta a\theta | a \in Q \cup \Gamma \cup \{\rhd , \lhd\}\}^+$ is equal to its own congruence class. It follows easily that U_L is a unitary submonoid of $L_{\mathcal{M}}$.

Therefore, the monoids U_L and U_R are isomorphic and are both free monoids of the same rank. □

Corollary 2.1. *The generalized word problem for U in $R_{\mathcal{M}}$ (and in $L_{\mathcal{M}}$) is decidable by a finite automaton, hence with linear time complexity.*

We will write U for the isomorphic monoids U_L and U_R, and we form the tensor product $R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$ and the amalgam $[R_{\mathcal{M}}, L_{\mathcal{M}}; U]$. The next theorem will allow us to reduce the membership problem for $L(\mathcal{M})$ to the word problem for $R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$.

Theorem 2.2. *The assignment $x \mapsto (\zeta x, \zeta)$ induces an injective function $\phi : S_{\mathcal{M}} \longrightarrow R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$. This yields a reduction of the membership problem for $L(\mathcal{M})$ to the tensor equality problem for $R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$, and this reduction is one-to-one and computable in linear time.*

Proof. To prove that ϕ is well defined, suppose $x = y$ in $S_{\mathcal{M}}$. We will prove by induction on the number of applications of relations of $S_{\mathcal{M}}$ that $\zeta x \otimes \zeta = \zeta y \otimes \zeta$. Clearly, it suffices to prove this for one such application.

Suppose then that $x = v\alpha w$, that $y = v\beta w$ and that $\alpha = \beta$ is a relation of $S_{\mathcal{M}}$. Then either $\alpha \to \beta$ or $\beta \to \alpha$ is a transition of $\mathcal{M}$. If this transition is a right moving or a right endmarker transition, then $\alpha = \beta$ is a relation of $R_{\mathcal{M}}$, so clearly, $\zeta x \otimes \zeta = \zeta y \otimes \zeta$. If this transition is a left moving transition, then we proceed as follows.

If $z = a_1 \dots a_r$ where each $a_i \in Q \cup \Gamma \cup \{\rhd , \lhd\}$, let $\Theta(z) = \theta a_1 \theta \theta a_2 \theta \dots \theta a_r \theta$. By using the commutation and θ-insertion and θ-deletion relations of $R_{\mathcal{M}}$ and of $L_{\mathcal{M}}$, we see that $\zeta z = \zeta\Theta(z) = \Theta(z)\zeta = z\zeta$ for any $z \in (Q \cup \Gamma \cup \{\rhd , \lhd\})^+$ in *both* $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$. Furthermore, $\Theta(z) \in U$ for any $z \in (Q \cup \Gamma \cup \{\rhd , \lhd\})^+$.

It follows that, $\zeta v\alpha w \otimes \zeta = \zeta\Theta(v\alpha w) \otimes \zeta = \zeta \otimes \Theta(v\alpha w)\zeta = \zeta \otimes v\alpha w\zeta = \zeta \otimes v\beta w\zeta = \zeta \otimes \Theta(v\beta w)\zeta = \zeta\Theta(v\beta w) \otimes \zeta = \zeta v\beta w \otimes \zeta$ as desired.

Let us next prove the injectiveness of ϕ. If $u \in (Q \cup \Gamma \cup \{\rhd , \lhd\} \cup \{\zeta, \theta\})^*$, let $\overline{u}$ be the word in $(Q \cup \Gamma \cup \{\rhd , \lhd\})^*$ obtained by erasing all occurrences of $\{\zeta, \theta\}$ from u. Note that the map $(Q \cup \Gamma \cup \{\rhd , \lhd\} \cup \{\zeta, \theta\})^* \to (Q \cup \Gamma \cup \{\rhd , \lhd\})^*$ that sends u to $\overline{u}$ is a morphism; moreover, it induces morphisms from $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ to $S_{\mathcal{M}}$ since the relations of $S_{\mathcal{M}}$ are preserved.

We claim that if $\zeta x \otimes y\zeta = \zeta t \otimes v\zeta$ in $R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$, then $\overline{xy} = \overline{tv}$ in $S_{\mathcal{M}}$. Letting y and v be the empty word and assuming that x and t are in $(Q \cup \Gamma \cup \{\rhd , \lhd\})^*$ gives injectiveness. We prove this claim by induction of the number of steps needed to derive the relation $\zeta x \otimes y\zeta = \zeta t \otimes v\zeta$. Again, it suffices to prove this for one step, by induction. There are four cases.

(1) Apply a relation of $R_{\mathcal{M}}$: If the relation is a θ-insertion or θ-deletion, or a commutation relation, changing x to z then $\overline{xy} = \overline{zy}$.
Consider next the case where we are applying a right moving relation $\overrightarrow{qa} = bp$ to x. In this case, $x = w\overrightarrow{q}az$ for some w and z and we must prove that $\overline{xy} = \overline{wbpzy}$ in $S_{\mathcal{M}}$. But $\overline{x} = \overline{wbpz}$ in $S_{\mathcal{M}}$, since $\overrightarrow{qa} = bp$ is a relation of $S_{\mathcal{M}}$ and $\overline{x} = \overline{w}\overrightarrow{q}a\overline{z}$. The result follows in this case. The case of right endmarker transitions is similar.

(2) Apply a relation of $L_{\mathcal{M}}$: This case is dual to the previous case.

(3) Shift an element of U from left to right (over the $\otimes$ symbol): Then $x = x'u$ in $R_{\mathcal{M}}$ for some $u \in U$ and we must prove that $\overline{xy} = \overline{x'(uy)}$ in $S_{\mathcal{M}}$. But, $x = x'u$ in $R_{\mathcal{M}}$ implies that $\overline{x} = \overline{x'u}$ in $S_{\mathcal{M}}$ and the result follows.

(4) Shift an element of U from right to left: Dual to the previous case.

Finally, let us prove the claim about the reduction. By Theorem 2.1, we see that $x \in L(\mathcal{M})$ if and only if $\rhd q_0 x \lhd = \rhd q_f \lhd$ in $S_{\mathcal{M}}$. By the proof of well-definedness and injectiveness of ϕ, this is true if and only if $\zeta \rhd q_0 x \lhd \otimes \zeta = \zeta \rhd q_f \lhd \otimes \zeta$. Thus the membership problem for $L(\mathcal{M})$ reduces to the tensor equality problem for $R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$.

From a computational point of view it is easy to produce $(\zeta x, \zeta)$ on input x in linear time (one just has to print ζ, then copy x, then print ζ). □

We see from Theorem 2.2 that the word problem for tensor products is in general undecidable. By contrast, we show next that the word problems for the factors, $L_{\mathcal{M}}$ and $R_{\mathcal{M}}$ are decidable for *every* Turing machine M; furthermore, the generalized word problem, the right and left factorization problems for U in $R_{\mathcal{M}}$ (respectively $L_{\mathcal{M}}$), and the one-step tensor equality problem are decidable.

The idea for solving the word problem for $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ is that these monoids admit a decidable (albeit infinite) *complete* rewrite system. A rewrite system is complete if and only if it is confluent and terminating. Thus, every word can be reduced to a unique normal form relative to an effectively given set of rewrite rules (see for example [25, 4] for the basic facts about rewrite systems).

We work with $R_{\mathcal{M}}$. The proof for $L_{\mathcal{M}}$ is dual. The idea is to use the rules of $R_{\mathcal{M}}$ itself as part of a rewrite system, with slight modifications. The normal forms are basically those words in which no right moving or right endmarker transition applies. In our model of Turing machine, it is clear that every word can be rewritten into one in which no such right transition occurs (recall that there is no infinite sequence of successive right move or right endmarker transitions).

The **right moving rewrite system** μ_R, for the monoid $R_{\mathcal{M}}$, has alphabet $Q \cup \Gamma \cup \{\rhd, \lhd, \theta, \zeta\}$, and rules:

Right moving rules: $\alpha \to \beta$, where $\alpha \to \beta$ is a transition of $\mathcal{M}$ such that the state in α is right moving.

Commutation rules: $a\zeta \to \zeta a$ for all $a \in Q \cup \Gamma \cup \{\rhd, \lhd\}$.

Left θ-erasing rule: $\theta\zeta \to \zeta$.

Right θ-erasing rules: $\zeta z \theta \to \zeta z$ for all $z \in (Q \cup \Gamma \cup \{\rhd, \lhd\})^*$.

The rules in μ_R can be generated by first suitably orienting the relations of $R_{\mathcal{M}}$ and then applying the Knuth–Bendix procedure for strings to generate a complete system (see [25, 4]). This leads to the infinite set of rules indicated above. We could appeal to the Knuth–Bendix Theorem here for a completeness proof, but the direct proof is more straightforward.

First note that the congruence generated by this rewrite system is exactly the congruence that defines $R_{\mathcal{M}}$. Clearly, every relation defining $R_{\mathcal{M}}$ is derivable from the symmetric closure of the rewrite system. Conversely, every rule of the rewrite

system is a consequence of the defining relations of $R_{\mathcal{M}}$. The only non-obvious case is the derivation of $\zeta z\theta \to \zeta z$ which follows from $|z|$ applications of relations of the form $\zeta a = a\zeta$ followed by $\zeta\theta = \zeta$, followed by $|z|$ applications of $a\zeta = \zeta a$.

Second, note that although the rewrite system has an infinite number of rules, it is easily decidable given a word w over the input alphabet, to list all the (finitely many) rules that apply to w. Algorithmically, this can be done in time $O(|w|)$.

Next, let us determine the reduced words or normal forms for μ_R. These are the words that contain no left sides of rules as contiguous subsegments. It is routine to show that the normal forms fall into the following classes.

(1) θ^i, with $i > 0$.
(2) $\theta^{i_0} y_1 \theta^{i_1} \dots \theta^{i_{n-1}} y_n \theta^{i_n}$, where
   - $n > 0$, $i_0, i_n \geq 0$ and $i_1, \dots, i_{n-1} > 0$,
   - $y_i \in (\Gamma \cup Q \cup \{\rhd, \lhd\})^+$ and no right moving rules apply to any $y_i, 1 \leq i \leq n$.
(3) ζ^i, with $i > 0$.
(4) $\zeta^i y$, with $i \geq 0$, $y \in (\Gamma \cup Q \cup \{\rhd, \lhd\})^+$ and no right moving rules apply to y.

Now note that every word z over the input alphabet can be rewritten to a word in normal form in a finite number of steps and that there are no words that lead to an infinite number of reductions. This is an immediate consequence of the fact that our Turing machines can only make a finite number of successive right moves on any configuration. Thus the rewrite system μ_R is terminating.

To prove that μ_R is complete, that is, that every word reduces to a unique normal form, we must show that all ambiguities can be resolved. Since we have proved that the system μ_R terminates, we need only show that it is locally confluent (see e.g. [25, 4] for details).

We need then only verify that rules that have overlapping occurrences of left hand sides can be resolved. We will list all overlaps and show how ambiguities can be resolved. First note that by the determinism of $\mathcal{M}$, no two rules from δ can overlap.

(1) Suppose that $z = x\alpha y$, where $\alpha \to \beta$ is a right moving or right endmarker transition of $\mathcal{M}$, and thus a rule of μ_R. Then there is an ambiguity arising from the rule $\zeta z\theta \to \zeta z$. The word $\zeta z\theta$ can derive both $\zeta x\beta y\theta$ and ζz. However, both these words derive $\zeta x\beta y$ and thus this ambiguity is resolved.
(2) Suppose that $\alpha \to \beta$ is a right moving rule of μ_R where $\alpha = qa$. Then this rule overlaps with the rule $a\zeta \to \zeta a$. The word $qa\zeta$ can derive both $\beta\zeta$ and $q\zeta a$, in one step. But $\beta\zeta$ derives $\zeta\beta$ in $|\beta|$ steps, and $q\zeta a$ derives first ζqa and then $\zeta\beta$. This resolves this type of ambiguity.
(3) The rule $\theta\zeta \to \zeta$ overlaps with the rule $\zeta z\theta \to \zeta z$ in two ways. For the first kind of overlap, $\theta\zeta z\theta$ can derive both $\theta\zeta z$ and $\zeta z\theta$. Clearly, both these words derive ζz. In the second kind of overlap, the word $\zeta z\theta\zeta$ derives $\zeta z\zeta$, by each of the above two rules.

(4) Finally, the rules $a\zeta \to \zeta a$ and $\zeta z\theta \to \zeta z$ overlap. The word $a\zeta z\theta$ derives both $\zeta az\theta$ and $a\zeta z$. Both these words derive ζaz in one step and this resolves this ambiguity.

It is straightforward to see that these are all the ambiguities and thus the system μ_R is complete. Dually, we define a rewrite system μ_L by replacing the word "right" by "left" in the rules defining μ_R. The monoid defined by μ_L is then $L_{\mathcal{M}}$ and the proof of completeness of μ_L is analogous to that for μ_R. We summarize these results in the following theorem.

Theorem 2.3. *Let $\mathcal{M}$ be a Turing machine. Then μ_R and μ_L are complete and decidable rewrite systems for $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ respectively; the set of left side of all the rules in μ_R (respectively μ_L) is a regular language. Thus, the word problems for both $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ are decidable.*

We will prove in the next section that the word problems for both $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ can be decided in deterministic linear time.

Before turning to the factorization problems, we need to look at congruence classes in $L_{\mathcal{M}}$ and $R_{\mathcal{M}}$ relative to the given presentations.

Lemma 2.2. *Let w be a reduced word in the rewriting system μ_R.*

(1) *If w does not contain the letter ζ, then the congruence class of w is finite.*

(2) *If w contains ζ, then the congruence class of w contains only a finite number of words that do not contain θ; all other words in the congruence class of w are obtained by inserting arbitrary occurrences of θ anywhere into the θ-free words equivalent to w.*

Proof. We work with $R_{\mathcal{M}}$, the proof for $L_{\mathcal{M}}$ being similar. If w does not contain ζ, then no word containing ζ can be equivalent to w, since no rule of μ_R removes occurrences of ζ. Let v be a word that is in the congruence class of w. Since μ_R is complete and w is reduced, v reduces to w. The only rules that apply to v are the right moving rules of μ_R. Only the right endmarker shrinking or extending transitions can change the length of v. In the latter case, since the new state after such a transition is a left moving state, no rule of μ_R can apply again at that location. Thus, the number of length changing rules that apply to v in reducing it to w is at most equal to the number of occurrences of $\lhd$ in v. It follows that the length of v is at most twice the length of w. Therefore, the congruence class of w is finite.

If w contains an occurrence of ζ, then $w = \zeta^i y$ where $i > 0$ and y has no occurrences of either ζ or θ and no right moving rules apply to y. Every word in the congruence class of w must have exactly i occurrences of ζ since no rule of μ_R changes this number. Let v be any word that reduces to w and contains no θ. We can first apply the commutation rules to reduce v to a word of the form $\zeta^i z$ of the same length as v and such that z contains no occurrences of either θ or ζ. Furthermore, z reduces to y and this bounds the number of such z by the same argument as in the preceding paragraph. Therefore the number of words in

the congruence class of w with no occurrences of θ is finite. Finally, it is clear that every word that contains ζ reduces to a word with no θ, and that inserting θ anywhere in a word containing ζ does not change the congruence class. The result follows. □

We now come to the factorization problems of U in $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$.

Lemma 2.3. *Every word z in the generators of $R_{\mathcal{M}}$(respectively $L_{\mathcal{M}}$) has a finite number of U-suffixes (respectively U-prefixes). Furthermore, there is an algorithm that on input z outputs the set of U-suffixes (and U-prefixes) of z.*

Proof. Again we work on the case of $R_{\mathcal{M}}$ as the case for $L_{\mathcal{M}}$ is dual. Let z be a reduced word for μ_R. If z does not contain ζ, then every U-suffix of z must be a suffix of z regarded as a word in the free monoid. This is because only right moving rules apply to any word that reduces to z and no word of the form $\theta a \theta$ can be produced by such a rule, where a is a single letter. Thus, the number of U-suffixes is bounded by $|z|$ in this case. Clearly all such U-suffixes can be found effectively.

If z contains occurrences of ζ, then $z = \zeta^i y$ where $i > 0$ and y contains no occurrences of θ or ζ and no right moving rule applies to y. By Lemma 2.2, we know that z is congruent to at most a finite number of words of the form $\zeta^i y'$ where y' contains no occurrences of θ or ζ. Every suffix of such a y' can be turned into a U-suffix of z by using the commutation rules to surround each letter with θ. Conversely, if $z = su$ in $R_{\mathcal{M}}$ where $u \in U$, then the word su is reducible to z by using the rules of μ_R. Since the word su must contain i occurrences of ζ, we can first use the commutation rules and the θ-erasing rules to transform su into the form $\zeta^i y'$ as above. In the process, the word u is stripped of all occurrences of θ and the resulting word is a suffix of y'. Thus all U-suffixes of z are obtained from suffixes of such y' by insertions of θ. Therefore, there are only finitely many U-suffixes of z and they can all be found effectively. □

Corollary 2.2. *The right (respectively left) factorization problem for U in $R_{\mathcal{M}}$ (respectively $L_{\mathcal{M}}$) is decidable.*

We now come to the one-step tensor equality problem.

We say that a monoid S has the *finite factorization property* if and only if for all $s \in S$, the set $\{(x, y) \mid x, y \in S, s = xy\}$ is finite. The following lemma is straightforward to prove.

Lemma 2.4. *Let S and T be monoids with the finite factorization property. Then the free product $S * T$ and the direct product $S \times T$ also have the finite factorization property.*

Lemma 2.5. *The submonoid of $R_{\mathcal{M}}$ (or of $L_{\mathcal{M}}$) consisting of all elements with no occurrences of ζ has the finite factorization property.*

Proof. Again the arguments for $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$ are similar, so we work with $R_{\mathcal{M}}$. Let $S = \{w \in R_{\mathcal{M}} \mid \zeta \text{ does not occur in } w\}$, and let T be the submonoid of $R_{\mathcal{M}}$

generated by $Q \cup \Gamma \cup \{\triangleright, \triangleleft\}$. That is, T consists of those elements in $R_{\mathcal{M}}$ that have no occurrences of either θ or ζ.

By the properties of the rewrite system μ_R, we see that S is isomorphic to the free product of T and the submonoid generated by θ. Note that the submonoid generated by θ is isomorphic to the positive natural numbers under addition. It suffices then by Lemma 2.4 to show that T has the finite factorization property.

First note that the set of elements of $R_{\mathcal{M}}$ whose normal form contains either an occurrence of θ or an occurrence of ζ is an ideal of $R_{\mathcal{M}}$. That is, T is the complement of an ideal in $R_{\mathcal{M}}$ and thus any factorization of an element of T must have both factors in T.

Suppose then that $t = xy$, where $t, x, y \in T$ and t is a normal form for μ_R. Then xy reduces to t using only the right moving rules of μ_R. The only right moves that change length are those that have an occurrence of $\triangleleft$ on the left side of the rule. Notice that the right side of such a rule has a left moving state and thus no rule of μ_R can be applied again using that occurrence of $\triangleleft$. It follows that the number of length changing rules in this reduction is bounded by the number of occurrences of $\triangleleft$ in t and thus the length of xy is bounded by $2|t|$. Thus, T has the finite factorization property. □

Lemma 2.6. *Let s be a word over the generators of $R_{\mathcal{M}}$, and let u be a word over the generators of U. Then the set of words $\{s' \mid s'u = s \text{ in } R_{\mathcal{M}}\}$ is finite. Furthermore, there is an algorithm that on input (s, u) outputs this set. A similar result holds for $L_{\mathcal{M}}$.*

Proof. Assume first that s has no occurrences of ζ. Then by Lemma 2.5, s has a finite number of factorizations, $s = xy$ for any words x, y over the generators of $R_{\mathcal{M}}$. The proof of Lemma 2.5 shows that all these factorizations can be found algorithmically, and since the generalized word problem for U is decidable, we can recognize the desired set in this case.

Assume then that $s = \zeta^i y$ where $i > 0$ and y has no occurrences of either θ or ζ and no right moving rule applies to y. Assume that $s = s'u$ for some u over the generators of U, and some s'. Since u contains no occurrences of ζ, it follows that s' must contain exactly i occurrences of ζ. Thus, s' reduces to $\zeta^i y'$ where y' has no occurrences of either θ or ζ and no right moving rule applies to y'. Let $\overline{u}$ be the word obtained by deleting all occurrences of θ in u. It follows from the properties of μ_R that $y'\overline{u} = y$. By Lemma 2.5, y has only a finite number of factorizations. Thus there are only a finite number of such s'. Since the factorizations of y can be algorithmically found, so can the desired set in this case as well. □

Theorem 2.4. *The one-step tensor equality problem for $R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$ is decidable. The tensor equality problem for $R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$ is recursively enumerable.*

Proof. Given (s, t), we can use Lemma 2.3 to algorithmically find the finite set X of U-suffixes of s. Since multiplication is decidable, we can decide for each $u \in X$ whether $s = s'u, ut = t'$. If we ever get an equality, we say "yes", otherwise

we exhaust the elements of X and say "no". The same argument works if we shift u in the other direction.

Now $s \otimes t = s' \otimes t'$ if and only if we can obtain $s' \otimes t'$ from $s \otimes t$ by some finite number of one-step tensor equalities. Now we can use lemma 2.6 and the solution to the basic tensor equality problem to find all possible ways to find all the finitely many pairs (s_1, t_1), such that $s \otimes t = s_1 \otimes t_1$ in one step. Finally, we use a breadth first search to list all pairs (s', t') such that $s \otimes t = s' \otimes t'$. Therefore, the tensor equality problem is recursively enumerable. □

We can now state the more detailed technical form of our main theorem. We will say that a triple $(S, T; U)$ consisting of two finitely presented monoids S, T and a common submonoid U *has a decidable description* if and only if:

(1) The word problems for both S and T are decidable.
(2) The generalized word problems for U in both S and T are decidable.
(3) The right (respectively left) factorization problem for U in S (respectively T) is decidable.
(4) The one-step tensor equality problem for $S \otimes_U T$ is decidable.

This is the least amount of information one needs to reasonably ask algorithmic questions about the tensor equality problem for $S \otimes_U T$ or the word problem for the free product with amalgamation $S *_U T$. Putting together what we have proved in this section, we have the following theorem, which is the more technical form of main result of this paper.

Theorem 2.5. *Let $\mathcal{M}$ be a Turing machine and let $R_{\mathcal{M}}$, $L_{\mathcal{M}}$ and U be the monoids associated with $\mathcal{M}$, as defined earlier. Then the following holds.*

(1) *$(R_{\mathcal{M}}, L_{\mathcal{M}}; U)$ has a decidable description.*
(2) *U is a finitely generated free monoid and is a unitary submonoid of both $R_{\mathcal{M}}$ and $L_{\mathcal{M}}$.*
(3) *The membership problem for $\mathcal{M}$ is reducible (via a one-to-one linear-time reduction) to the word problem for $R_{\mathcal{M}} \otimes_U L_{\mathcal{M}}$, and to the word problem for $R_{\mathcal{M}} *_U L_{\mathcal{M}}$.*

Proof. The first two items and the undecidability for the tensor product follow from the results of this section. For the word problem of an amalgam, we use Theorem 1.1. □

2.4. *The case of rings and algebras over a field*

Amalgams and the corresponding embedding problems for free product with amalgamation have been studied extensively in the category of rings as well. See [22] and the references therein. However, as far as we know, there has been no extensive study of the word problem for amalgams in the category of rings. We can obtain a result similar to the one above by passing to monoid rings. Recall that the monoid ring $R[M]$ of a monoid M is the ring whose additive group is the free Abelian group with basis M and with multiplication induced by that of M. The

assignment of M to $R[M]$ is part of a functor from monoids to rings that is left adjoint to the forgetful functor from rings to monoids that assigns the underlying multiplicative monoid $U(R)$ to a ring R.

Lemma 2.7. *Let $[S, T; U]$ be an amalgam of monoids. Then $[R[S], R[T]; R[U]]$ is an amalgam of rings. Furthermore, $R[S *_U T] = R[S] *_{R[U]} R[T]$.*

Proof. Clearly the embeddings from U into S and T extend to embeddings from $R[U]$ into $R[S]$ and $R[T]$. Furthermore, the functor R is a left adjoint and thus preserves the limit $S *_U T$. □

Corollary 2.3. *Let $\mathcal{M}$ be a Turing machine and let $R_\mathcal{M}$, $L_\mathcal{M}$ and U be the associated monoids of $\mathcal{M}$. Then the rings $R[R_\mathcal{M}], R[L_\mathcal{M}]$ and $R[U]$ are finitely presented and have decidable word problems. Furthermore, $R[U]$ is a free subring of both $R[R_\mathcal{M}]$ and $R[L_\mathcal{M}]$. If the language accepted by $\mathcal{M}$ is undecidable, then the word problem for the amalgam $[R[R_\mathcal{M}], R[L_\mathcal{M}]; R[U]]$ and the tensor equality problem for $R[S] \otimes_{R[U]} R[T]$ is undecidable.*

As we mentioned, the result about tensor products of rings (with a somewhat different construction and different properties in the details) was first proved by E. Bach [1].

Let K be a field. By using the monoid algebra $K[M]$, we can obtain similar undecidability results for amalgams in the category of K-algebras. Of course, suitable computability assumption should be made on the structure of K to make sure that the input data have "a decidable description"; for our results, we can, for example use any finite field or the rational numbers for K.

3. The Complexity of the Word Problem of $R_\mathcal{M}$ and $L_\mathcal{M}$, and of Other Decidable Problems

In this section, we study the computational complexity of the problems that were shown to be decidable in the previous section. We show that the word problem for $R_\mathcal{M}$ and $L_\mathcal{M}$ as well as the generalized word problem the factorization problems for the submonoid U are all decidable in linear time, and that the one-step tensor equality problem is decidable in quadratic time.

First, note that we already have a quadratic time algorithm for the word problem for $R_\mathcal{M}$ by using the complete rewrite system μ_R to compute normal forms. Of course two words are equal in $R_\mathcal{M}$ if and only if they have the same normal form, and the literal equality of words in normal form can be decided in linear time.

To compute a normal form, we must deal with words of the form $w = x_0 q_1 x_1 \ldots q_k x_k$ where $\{q_i \mid 1 \leq i \leq k\}$ is the set of state symbols that appear in w. When $k > 1$, such words are not Turing machine configurations, of course, but rules of μ_R can possibly be applied at some or all of these states. These applications can interact in complicated ways. In the above word w, it could happen that the k state symbols are near the left end of w, and they they are able to move all the way right; if $k > cn$ (where $n = |w|$ and c is a constant with $0 < c < 0.1$, for example), then this process takes time $> c' \cdot n^2$, for some constant $c' > 0$).

In order to improve the performance of this algorithm from quadratic to linear time, we modify the Turing machines (without sacrificing any of their power) by using the *oriented letters trick* discussed by Birget in [2]. The oriented letters trick appears in various guises in many papers dealing with the word problem for groups and semigroups. We proceed as follows. Let $\mathcal{M}$ be a one-tape Turing machine with tape alphabet Γ. We first replace Γ by two disjoint bijective copies $\overleftarrow{\Gamma}$ and $\overrightarrow{\Gamma}$. Moreover, we modify the transitions of $\mathcal{M}$ so that at any time during a computation, there are only left pointing letters (i.e. members of $\overleftarrow{\Gamma}$) to the left of the head on the tape, and only right pointing letters (i.e. members of $\overrightarrow{\Gamma}$) to the right of the head. This is done by replacing every right moving transition $\overrightarrow{q}a \to bp$ of $\mathcal{M}$ by the corresponding oriented version, $\overrightarrow{q}\overrightarrow{a} \to \overleftarrow{b}p$. Left moving transitions $a\overleftarrow{q} \to pb$ are replaced by $\overleftarrow{a}\overleftarrow{q} \to p\overrightarrow{b}$. We identify the input alphabet Σ with its right pointing copy $\overrightarrow{\Sigma}$. With this identification, the modified machine accepts the same language as the original machine and has exactly the same time complexity. Thus, every deterministic one-tape Turing machine can be simulated by an oriented-letters Turing machine, with the same time complexity.

Furthermore, the results from the previous section apply to oriented-letters machines (since they are just a special case of deterministic one-tape Turing machines).

In the oriented-letters machine, the problem of a quadratic time to reduce words with more than one state symbol does not occur. As a state moves over a letter from left to right, it reverses the orientation of the letter from right-pointing to left-pointing. Therefore, once a state symbol can no longer move to the right, no right moving rule of μ_R can ever apply to that state again. Another way to put it: in a word with oriented letters, every occurrence of a state symbol has a range in which it can move; different occurrences of state symbols have disjoint ranges (this is made more precise by the "oriented factorization", defined below). We can now show that the monoids of the modified machine have linear time algorithms for their word problems.

Theorem 3.1. *Let $\mathcal{M}$ be a Turing machine with oriented letters, and let $R_{\mathcal{M}}$ (respectively $L_{\mathcal{M}}$) be the right (respectively left) moving monoid of $\mathcal{M}$. Then the word problem for $R_{\mathcal{M}}$ (and for $L_{\mathcal{M}}$) is decidable in linear time.*

Proof. Let μ be the complete rewrite system for $R_{\mathcal{M}}$. We prove that the unique reduced word corresponding to input word w can be computed in time $O(|w|)$. Moreover, the literal equality of (reduced) words can be decided in linear time.

Let $w = x_0 q_1 x_1 \dots q_k x_k$ where $\{q_i : 1 \le i \le k\}$ is the set of those state symbols that appear in w. Our algorithm is implemented by a multitape Turing machine, which first scans w from left to right and copies all occurrences of ζ to a second tape; at the same time, all letters other than ζ or θ are copied on a third tape (this produces a word w', obtained from w by erasing ζ and θ everywhere).

If the number of ζ's is 0, then right moving rules (of the rewrite system μ_R) are applied to w until no more right moving rules are applicable. By the use of

oriented letters, this application of right moving rules only takes linear time (since all the state symbols occurring in w have disjoint ranges). At this point, the word obtained from w is reduced.

On the other hand, if the number of ζ's is $i > 0$, then we scan w' (on the third tape) and apply right moving rules to w' until no more right moving rules are applicable. Again, this takes linear time, because of the oriented letters. The word obtained from w' in this way, with ζ^i concatenated in front, is the normal form of w.

The proof for $L_{\mathcal{M}}$ is similar. □

Theorem 3.2. *The generalized word problem of U in $R_{\mathcal{M}}$ (respectively $L_{\mathcal{M}}$) is decidable in linear time.*

Proof. We saw this already in the proof of Lemma 2.1. □

Before dealing with the complexity of the remaining problems, we need a more detailed analysis of Turing machines with oriented letters (this is adapted from [2]).

Definition. Let $(Q, \overleftarrow{\Gamma} \cup \overrightarrow{\Gamma}, \Sigma, \delta, q_0, q_f)$ be a deterministic one-tape Turing machine with oriented letters $\Gamma = \overleftarrow{\Gamma} \cup \overrightarrow{\Gamma}$, and let w be a word over $Q \cup \Gamma \cup \{\triangleright, \triangleleft\}$. The *oriented factorization* of w is obtained by cutting w wherever one of the following two-letter segments occurs in w (cut between the two letters):

$\overrightarrow{a}\overleftarrow{c}$, with $\overrightarrow{a} \in \overrightarrow{\Gamma} \cup \{\triangleleft\}$ and $\overleftarrow{c} \in \overleftarrow{\Gamma} \cup \{\triangleright\}$;

$q\overleftarrow{c}$, with $q \in Q$ and $\overleftarrow{c} \in \overleftarrow{\Gamma} \cup \{\triangleright\}$;

$\overrightarrow{a}q$, with $\overrightarrow{a} \in \overrightarrow{\Gamma} \cup \{\triangleleft\}$ and $q \in Q$;

pq, with $p, q \in Q$.

Lemma 3.1. *If $w \in (Q \cup \Gamma \cup \{\triangleright, \triangleleft\})^*$ has an oriented factorization $w = w_1 \cdot w_2 \cdot \ldots \cdot w_k$ then:*

(1) *w_i belongs to $\overleftarrow{\Gamma}^* Q \overrightarrow{\Gamma}^* \cup \overleftarrow{\Gamma}^* \overrightarrow{\Gamma}^*$, for all i $(1 \leq i \leq k)$;*

(2) *If $v \in (Q \cup \Gamma \cup \{\triangleright, \triangleleft\})^*$ is congruent to w in $R_{\mathcal{M}}$ (or in $L_{\mathcal{M}}$), then the oriented factorization of $v = v_1 \cdot v_2 \cdot \ldots \cdot v_h$ is such that $h = k$ and v_i is congruent to w_i in $R_{\mathcal{M}}$ (respectively $L_{\mathcal{M}}$), for all i $(1 \leq i \leq k)$;*

(3) *The oriented factorization of w can be computed in linear time (by a finite-state machine).*

Proof. Property (1) is obvious from the definition of the oriented factorization. Proof of property (2): The relations in the presentations of $R_{\mathcal{M}}$, when applied to a word in $(Q \cup \Gamma \cup \{\triangleright, \triangleleft\})^*$, do not change the number of factors in the oriented factorization of the word. Hence $h = k$. Moreover, no relation uses letters of different factors (as is clear from form of the segments where a word is cut, in the definition of the oriented factorization). Therefore, relations can only be applied within the factors. Thus v_i is congruent to w_i, for all i.

Regarding property (3), it is clear from the definition of the oriented factorization (via special subsegments) that a finite-state machine can find the places where w is cut. □

For the next theorems, we want to put an additional constraint on the Turing machine M, namely that it is **injective**. By definition, a deterministic Turing machine is injective if and only if for any configuration uqv, there is at most one transition $\alpha \to \beta$ such that the "reverse transition", $\beta \to \alpha$, is applicable to the configuration uqv. In other word, the "reverse machine" of M is deterministic.

It is a remarkable fact (due to Lecerf [16] and later, independently to Bennett [3]) that every decidable language is also accepted by some deterministic injective Turing machine which always eventually halts (and which can be assumed to have just one tape, see [2]).

Theorem 3.3. *The right (respectively left) factorization problem of U in $R_{\mathcal{M}}$ (respectively $L_{\mathcal{M}}$), is decidable deterministically in linear time.*

Proof. We will consider $R_{\mathcal{M}}$ (the case of $L_{\mathcal{M}}$ is similar). Let us denote the set of generators of $R_{\mathcal{M}}$ by X (so $X = Q \cup \Gamma \cup \{\rhd\ ,\ \lhd\} \cup \{\theta, \zeta\}$). Let $V = \{\theta a \theta \mid a \in Q \cup \Gamma \cup \{\rhd\ ,\ \lhd\}\}$ be the free generating set of U. Recall that the right factorization problem of U in $R_{\mathcal{M}}$ takes as input a word $s \in X^*$ and a word $u \in V^*$, and asks whether u is a U-suffix of s (i.e. whether there exists $s' \in X^*$ such that $s = s'u$ in $R_{\mathcal{M}}$).

Case 1. s does not contain ζ. Then u is a U-suffix of s if and only if u is a literal suffix of s (as words). Indeed, in the absence of ζ, no relation is applicable to the occurrences of θ in s, so s has u as a U-suffix in $R_{\mathcal{M}}$ if and only if s has u as a literal suffix. Moreover, one can clearly decide in linear time (on a two-tape Turing machine, for example) whether one word is a literal suffix of another word.

Case 2. s contains ζ. Then the normal form of s is of the form $\zeta^i x$, where $x \in X^*$ contains no ζ and no θ. We saw already (in Theorem 3.1 about the word problem of $R_{\mathcal{M}}$) that normal forms are computable in linear time. Also, let $\bar{u}$ be the word obtained from u by first removing all occurrences of θ, and then putting the remaining word into normal form; then $\bar{u}$ contains no ζ and no θ. Since s and $\bar{u}$ contain no ζ and no θ, let us consider their oriented factorizations $x = x_p \cdot \ldots \cdot x_1$ and $\bar{u} = u_k \cdot \ldots \cdot u_1$ (which are computable in linear time, see Lemma 3.1). We have:

Claim. *u is a U-suffix of s if and only if the following two conditions hold:*
(1) *$k \leq p$ and $u_i = x_i$; (literal equality) for $i = 1, \ldots, k-1$;*
(2) *u_k is a U-suffix of x_k.*

Proof of the Claim. Since $\zeta^i x = s$ in $R_{\mathcal{M}}$, we have: u is a U-suffix of s if and only if u is a U-suffix of $\zeta^i x$, if and only if for some word s', $\zeta^i x = s'u$. Since $\zeta^i x$ contains ζ, we can apply rewrite rules that remove every θ and then move every ζ to the left end. Then $s'u = \zeta^i x' \bar{u}$ in $R_{\mathcal{M}}$(for some reduced word x'). Continuing the reduction process, $\zeta^i x' \bar{u} = \zeta^i x' \bar{u}$ in $R_{\mathcal{M}}$, where $x = x'\bar{u}$ in $R_{\mathcal{M}}$. Then by Lemma 3.1 and the uniqueness of normal forms we have, $u_i = x_i$ for $i = 1, \ldots, k-1$, and $x_k = x'_k u_k$ in $R_{\mathcal{M}}$, for some x'_k.

This prove the claim.

Let us now see how one can check deterministically in linear time whether u is a U-suffix of s. We saw in the Claim that $u_i = x_i$ for $i = 1, \ldots, k-1$, and that $x_k = x'_k u_k$ in $R_{\mathcal{M}}$(for some reduced word x'_k). Since the word $x'_k u_k$ is one factor in the oriented factorization of x, it contains at most one state letter.

Case 2.1. If $x'_k u_k$ contains no state letter, then it is in normal form (no rewrite rule is applicable), hence $x_k = x'_k u_k$. Similarly, if $x'_k u_k$ contains a state but the state is left-moving, no rule is applicable. Also, if the state letter is in u_k, or in the inside (not at the left side) of x'_k, no rule is applicable (since x'_k and u_k are already reduced). Thus in all these cases, $x_k = x'_k u_k$ is in normal form (hence equal to x_k), and therefore we have:

$\bar{u}$ is a literal suffix of x.

Case 2.2. The only other case is when $x_k = x'_k u_k$ contains a state letter at the left end of x'_k, i.e. x'_k is of the form $x'_k = t_k q_k$, where t_k is reduced. In that case, u_k contains no state letter.

Then, to reduce $t_k q_k u_k$, we can apply right-moving transitions of the Turing machine to $q_k u_k$ and eventually obtain a reduced word of the form $z'_k q'_k u'_k$, where u'_k is a literal suffix of u_k.

Finally, in Case 2.2, we process as follows to check whether u_k is a U-suffix of x_k; we assume that we have already checked whether $\bar{U}$ is a literal suffix of x (which can be done deterministically in linear time) and found out that it is not.

(1) Check whether x_k has the form $z''_k q'_k u'_k$ where u'_k is a literal suffix of u_k and q'_k is a state letter. (If not, then u is not a U-suffix of s.)

(2) Then, by injectiveness of the Turing machine, we run the Turing machine deterministically in reverse (using only inverse of right-moving transitions), starting from the configuration $z''_k q'_k$. When we reach a configuration of the form vqu_k (for some v and q), then u is a U-suffix of s (otherwise it is not). This takes time $\leq |z''_k| < |x_k| \leq |x|$.

By the definition of U-suffix, u is a U-suffix of s if and only if $\bar{u}$ is a U-suffix of $\zeta^i x$.

By the uniqueness of normal forms, and by invariance of the oriented factorization (Lemma 3.1(2)), we have: $\bar{u}$ is a U-suffix of $\zeta^i x$ if and only if the oriented factorization of x is such that $x = x_p \cdot \ldots \cdot x_1 = x_p \cdot \ldots \cdot x_{k+1} \cdot (x''_k x'_k) \cdot x_{k-1} \cdot \ldots \cdot x_1$, where $x_i = u_i$ in $R_{\mathcal{M}}$ for $1 \leq i \leq k-1$, and $x'_k = u_k$ in $R_{\mathcal{M}}$. Moreover, since x and $\bar{u}$ (and hence their factors too) are in normal form, these last conditions are equivalent to $x_i = u_i$ for $1 \leq i \leq k-1$, and $x'_k = u_k$ (literal equalities). This means that u is a U-suffix of s if and only if $\bar{u}$ is a literal suffix of x, and this can be checked in linear time.

Theorem 3.4. *The one-step tensor equality problems is decidable deterministically in quadratic time.*

Proof. Again, we denote the set of generators of $R_{\mathcal{M}}$ (and of $L_{\mathcal{M}}$) by X (respectively Y); so $X = Q \cup \Gamma \cup \{\triangleright, \triangleleft\} \cup \{\theta, \zeta\}$ and Y is a bijective copy of X. Let $V = \{\theta a \theta \mid a \in Q \cup \Gamma \cup \{\triangleright, \triangleleft\}\}$ be the free generating set of U. Recall that the

one-step tensor problem takes as input two pairs of words $(s_1, s_2), (s_1', s_2') \in X^* \times Y^*$, and asks whether there exists $u \in V^*$ such that $s_1 = s_1' u$ in $R_{\mathcal{M}}$ and $us_2 = s_2'$ in $L_{\mathcal{M}}$ (or $s_1' = s_1 u$ in $R_{\mathcal{M}}$ and $us_2' = s_2$ in $L_{\mathcal{M}}$, but we need not consider this since it is similar to the other problem). Note that u here is a U-suffix of s_1 and a U-prefix of s_2. Observe also that for a given word u, we can check in linear time whether $s_1 = s_1' u$ in $R_{\mathcal{M}}$and $us_2 = s_2'$ in $L_{\mathcal{M}}$, because the word problems of $R_{\mathcal{M}}$and $L_{\mathcal{M}}$are decidable in linear time.

If (s_1, s_2) and (s_1', s_2') are equivalent in one step, then s_1 and s_1' must have the same number of occurrences of ζ, and also s_2 and s_2' must have the same number of occurrences of ζ. The numbers of ζ's are easy to compare in linear time. From now on we assume that s_1 and s_1' have the same number of ζ's, and s_2 and s_2' have the same number of ζ's.

Case 1. s_1 and s_1' contain no ζ (the case where s_2 and s_2' contain no ζ is similar).

Then every U-suffix of s_1 is a literal suffix of the normal form $\zeta^i x$ of s_1 (as we saw in Case 1 of the proof of the previous theorem). Each literal suffix of x can be computed in linear time and compared in linear time with all the prefixes of the normal form of s_2, to check whether it is a U-prefix of s_2 in $L_{\mathcal{M}}$ (which can also be done in linear time, by the previous Theorem); there are linearly many literal suffixes of x, so the whole test will take quadratic time.

Case 2. Each of s_1, s_1', s_2 and s_2' contains occurrences of ζ.

The normal forms of these words (which can be computed in linear time, by the proof of Theorem 3.1): (s_1, s_2) becomes $(\zeta^i x, y\zeta^j)$, and (s_1', s_2') becomes $(\zeta^i x', y'\zeta^j)$. Here x, x', y and y' contain no ζ and no θ. Let us consider the oriented factorizations $x = x_p \cdot \ldots \cdot x_1$ and $y' = y'_1 \cdot \ldots \cdot y'_r$, and $\bar{u} = u_k \cdot \ldots \cdot u_1$.

As we saw in the Claim in the proof of Theorem 3.3, u is a U-suffix of s_1 and a U-prefix of $s\prime_2$ if and only if $\bar{u}$ satisfies:

$$x = x_p \cdot \ldots \cdot x_1 = x_p \cdot \ldots \cdot x_{k+1} \cdot (x_k'' u_k) \cdot u_{k-1} \cdot \ldots \cdot u_1, \text{ and}$$

$$y' = u_k \cdot \ldots \cdot u_2 \cdot (u_1 y_k'') \cdot y'_{k+1} \cdot \ldots \cdot y'_r.$$

Therefore, if u exists and if $k \geq 2$, then $\bar{u} = u_k \cdot \ldots \cdot u_1$ must satisfy:

$$u_1 = x_1,$$

$$u_i = x_i = y'_{k-i+1} \quad \text{for all } i = 2, \ldots, k-1, \text{ and}$$

$$u_k = y'_1 .$$

Therefore, x and y' determine at most one possible $\bar{u}$ once k (the number of factors in the oriented factorization of $\bar{u}$) is known. Moreover, $k \leq p \leq |x|$. Also, u is uniquely determined by $\bar{u}$. Thus, the number of possible words u to consider is linearly bounded, and as we saw, it takes a linear time to check whether u is such that $s_1 = s_1' u$ in $R_{\mathcal{M}}$and $us_2 = s_2'$ in $L_{\mathcal{M}}$. Thus the overall time is quadratic.

On the other hand, if $k = 1$ (i.e. $\bar{u}$ consists of one factor in the oriented factorization), then $\bar{u}$ contains at most one state symbol. Also, $\bar{u}$ is U-suffix of x_1 and a U-prefix of y_1'. If $\bar{u}$ contains no state or a left-moving state, then (by Case 2.1 of Theorem 3.3) $\bar{u}$ is a literal suffix of x_1. Similarly, if $\bar{u}$ contains a right-moving state, then $\bar{u}$ is a literal prefix of y_1'. Just as in Case 1 above, such word $\bar{u}$ can be formed in quadratic time (if they exist). □

Theorem 3.5. *All the undecidability, decidability, and complexity results in this paper are invariant under change of the finite generating sets of $R_{\mathcal{M}}$, $L_{\mathcal{M}}$, and U (provided the new generating sets are also finite).*

Proof. The given set of generators $Q \cup \Gamma \cup \{\triangleright\ ,\ \triangleleft\} \cup \{\theta, \zeta\}$ of $R_{\mathcal{M}}$ (and of $L_{\mathcal{M}}$) is minimal: any set of generators must include the given ones. For generators in $Q \cup \Gamma \cup \{\triangleright\ ,\ \triangleleft\} \cup \{\theta\}$, this is obvious; since the congrence class of such a generator is a singleton, none of these generators can be written as the product of other elements of $R_{\mathcal{M}}$ (or $L_{\mathcal{M}}$). For ζ, the congruence class is $\theta^* \zeta \theta^*$, so every element that is congrent to ζ also contains ζ. So ζ cannot be written as the product of other elements of $R_{\mathcal{M}}$ (or $L_{\mathcal{M}}$).

The submonoid U is free over $\{\theta a \theta \mid a \in Q \cup \Gamma \cup \{\triangleright\ ,\ \triangleleft\}\}$, so these generators are necessary.

The invariance of the theorem now follows trivially. The only possible changes in the generating sets are additions of redundant generators. This does not change the decidability results and can only decrease the complexity. For word problems of finitely generated monoids, it is well known that undecidability does not depend on the choice of a finite set of generators (and this applies in particular to the monoid $R_{\mathcal{M}} *_U L_{\mathcal{M}}$). For the word problem of the tensor product, a similar proof shows that its undecidability does not depend on the choice of a finite set of generators. □

References

1. E. Bach, *Tensor products and computability*, J. Symbolic Comput. **18** (1994), 585–593.
2. J. C. Birget, *Time-complexity of the word problem for semigroups and the Higman Embedding theorem*, Report UNL-CSE-95-009, Department of Computer Science and Engineering, University of Nebraska, May 1995; Int. J. Algebra and Comput. (to appear).
3. C. Bennett, *Time/Space tradeoffs for reversible computation*, SIAM J. Computing **18** (1989), 766–776.
4. R. Book and M. Otto, *String Rewriting Systems*, Springer Verlag, 1993.
5. S. Bulman-Fleming and K. McDowell, *Absolutely flat semigroups*, Pacific J. Math. **107** (1983), 319–333.
6. D. V. Dekov, *The embedding of semigroup amalgams*, J. Algebra **141** (1991), 158–161.
7. D. V. Dekov, *Free products with amalgamation of semigroups*, Semigroup Forum **46** (1993), 54–61.
8. T. E. Hall, *Free products with amalgamation of inverse semigroups*, J. Algebra **34** (1975), 375–385.
9. T. E. Hall, *Representation extension and amalgamation in semigroups*, Quart. J. Math. Oxford **2** (1978), 309–334.

10. P. M. Higgins, *Techniques of Semigroup Theory*, Oxford University Press, New York, 1992.
11. J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, Massachusetts, 1979.
12. J. Howie, *Embedding theorems with amalgamation for semigroups*, Proc. London Math. Soc. **12**(3) (1962), 511–534.
13. J. Howie, *Amalgamations: A survey*, in *Semigroups: Algebraic Theory and Applications to Formal Languages and Codes*, eds. C. Bonzini, A. Cherubini and C. Tibiletti, World Scientific, Singapore, 1993.
14. Kimura, N., *On semigroups*, Ph.D. Thesis, Tulane University, 1957.
15. G. Lallement, *Semigroups and Combinatorial Applications*, Wiley, New York, 1979.
16. Y. Lecerf, *Machines de Turin réversible...*, Comptes Rendus de l'Académie des Sciences, Paris **257** (18) (October 1963), 2597–2600.
17. R. Lyndon and P. Schupp, *Combinatorial Group Theory*, Springer-Verlag, Berlin, 1978.
18. R. McNaughton, *Elementary Computability, Formal Languages and Automata*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.
19. J. Renshaw, *Extension and amalgamation in monoids and semigroups*, Proc. London Math. Soc. **52** (3) (1986), 119–141.
20. J. Renshaw, Flatness and amalgamation in monoids, J. London Math. Soc. **36** (2) (1986).
21. J. Rotman, *An Introduction to the Theory of Groups*, 4th edn., Spring Verlag, 1994.
22. L. Rowen, *Ring Theory*, Vol. 1, Academic Press, New York, 1983.
23. O. Schreier, *Die Untergruppen der freien Gruppen*, Abh. Math. Sem., University of Hamburg **5** (1927), 161–183.
24. J. P. Serre, *Trees*, Springer-Verlag, New York, 1980.
25. C. Sims, *Computation with finitely presented groups*, Wiley, New York, 1994.
26. J. Stallings, *Group theory and three-dimensional manifolds*, Yale Monographs **4** (1971).