Reachability in Unions of Commutative Rewriting Systems is Decidable

Mikołaj Bojańczyk and Piotr Hoffman

Institute of Informatics, Warsaw University, Poland
{bojan,piotrek}@mimuw.edu.pl

Abstract. We consider commutative string rewriting systems (Vector Addition Systems, Petri nets), i.e., string rewriting systems in which all pairs of letters commute. We are interested in reachability: given a rewriting system R and words v and w, can v be rewritten to w by applying rules from R? A famous result states that reachability is decidable for commutative string rewriting systems. We show that reachability is decidable for a union of two such systems as well. We obtain, as a special case, that if $h : U \to S$ and $g : U \to T$ are homomorphisms of commutative monoids, then their pushout has a decidable word problem. Finally, we show that, given commutative monoids U, S and T satisfying $S \cap T = U$, it is decidable whether there exists a monoid M such that $S \cup T \subseteq M$; we also show that the problem remains decidable if we require M to be commutative, too.

Topic classification: Logic in computer science – rewriting

1 Summary of results

A *string rewriting system* R over a finite alphabet Σ is simply a finite set of rules of the form $v \mapsto w$, where v and w are words over Σ (string rewriting systems are also called *semi-Thue systems*). Such a system defines a one-step rewriting relation $\to_R$ and a multistep rewriting relation $\to_R^*$ on words over Σ: a word v rewrites in one step to a word w if there exist words $t, v_0, u, w_0 \in \Sigma^*$ such that $v = tv_0u$, $w = tw_0u$ and $v_0 \mapsto w_0$ is a rule of R; the multistep rewriting relation is the reflexive-transitive closure of the one-step relation. In the sequel, the statement "v rewrites to w in R" shall mean that $v \to_R^* w$.

The *(uniform) reachability problem* is defined as follows: Given a string rewriting system R and words v and w in the alphabet of that system, answer whether v rewrites to w in R? This problem is one of the most basic undecidable problems. However, for appropriate restrictions on the form of the rewriting system R, the problem may become decidable.

A string rewriting system is said to be *commutative* if for any two letters a and b of the alphabet it contains the rule $ab \mapsto ba$. Commutative string rewriting

[1] First author supported by the EC Research Training Network Games, second author by EC project Sensoria (No. 016004).

systems are also called *Vector Addition Systems* or *Multiset Rewriting Systems*, since they treat words as multisets of letters — or elements of $\mathbb{N}^\Sigma$, where Σ is the alphabet. These systems are equivalent to *Petri nets*.

The following is a famous result [1, 2]:

Theorem 1. *Reachability in commutative string rewriting systems is decidable.*

If R_Σ and R_Γ are rewriting systems over alphabets Σ and Γ, which may be distinct and may have a non-empty intersection, then one may consider the *union system* $R_\Sigma \cup R_\Gamma$ over the alphabet $\Sigma \cup \Gamma$. This system is constructed by simply taking both the rules from R_Σ, as well as the rules from R_Γ.

Note that the union of string rewriting systems may be much more complex than its parts. This is shown by the following example.

A string rewriting system is said to be *symmetric* if for any rule $v \mapsto w$ in the system, the system also contains the rule $w \mapsto v$. Such systems are called *Thue systems*. Sapir [3] (for an outline of the proof see [4]) constructed two symmetric string rewriting systems R_1 and R_2 such that the sets

$$\{v\#w : \ v \text{ rewrites to } w \text{ in } R_1\} \qquad \{v\#w : \ v \text{ rewrites to } w \text{ in } R_2\}$$

are both regular languages, but reachability in the (symmetric) union $R_1 \cup R_2$ is undecidable!

In this paper, we consider unions of commutative string rewriting systems. We do not make any assumptions on how the alphabets Σ and Γ of these systems relate to each other, whether they are disjoint or not, etc. Notice that the union $R_\Sigma \cup R_\Gamma$ of commutative systems R_Σ and R_Γ will not be commutative itself: if a is a letter from $\Sigma \setminus \Gamma$ and b a letter from $\Gamma \setminus \Sigma$, then $ab \mapsto ba$ will not be in the union system; moreover, ab will in general not rewrite to ba in the union system.

The main contribution of the paper is the following theorem:

Theorem 2. *Let R_Σ and R_Γ be commutative string rewriting systems. Then the reachability problem in the union $R_\Sigma \cup R_\Gamma$ is decidable.*

This theorem properly extends Th. 1. Its proof is quite complex, and in the next two sections we only outline it. The full proof will be found in the full version of this paper. The same applies to other omitted proofs.

In Sec. 4, we present results related to *amalgamations* [5] of commutative monoids. Some of these results are straightforward consequences of Th. 2, while others do not depend on it. The following is obtained easily from Th. 2:

Corollary 1. *Let $h : U \to S$ and $g : U \to T$ be homomorphisms of commutative monoids, and let $h' : S \to P$ and $g' : T \to P$ form a pushout of h and g (i.e. $h' \circ h = g' \circ g$ and any homomorphisms $h'' : S \to P'$ and $g'' : T \to P'$ with $h'' \circ h = g'' \circ g$ can be factored as $u \circ h' = h''$ and $u \circ g' = g''$ for a unique homomorphism $u : P \to P'$). Then P has a decidable word problem.*

If the homomorphisms h and g above are injective, then without loss of generality one may assume that $S \cap T = U$ and that h and g are inclusions. In this case the

pushout P is called the *amalgamated product*. If S and T are given by symmetric rewrite systems, then the amalgamated product is given by the union of these systems.

In algebra, the following natural problem is considered [5–9, 3]: Given monoids U, S and T satisfying $S \cap T = U$, answer whether some monoid M jointly extends S and T, that is, such that $M \supseteq S \cup T$. If so, $S \cup T$ is said to *embed* into M, or to be *embeddable*. To see why a union of monoids may not be embeddable, consider two distinct copies $\mathbb{Z}_1$ and $\mathbb{Z}_2$ of the integers with zero and addition, intersecting on the natural numbers with zero and addition: $\mathbb{Z}_1 \cap \mathbb{Z}_2 = \mathbb{N}$. The union $\mathbb{Z}_1 \cup \mathbb{Z}_2$ is not embeddable, because the two copies $-1_1, -1_2$ of -1 would have to be identified:

$$-1_1 = -1_1 + 1 + -1_2 = -1_2 \ .$$

The embeddability problem is in general undecidable. In fact, Sapir [3] proved the following result:

Theorem 3. *It is undecidable, given finite monoids U, S and T that satisfy $S \cap T = U$, whether there exists a monoid jointly extending S and T.*

We prove that in the case of commutative monoids the situation is rather different. More precisely, the following theorems hold:

Theorem 4. *It is decidable, given commutative monoids U, S and T that satisfy $S \cap T = U$, whether there exists a commutative monoid jointly extending S and T.*

Theorem 5. *It is decidable, given commutative monoids U, S and T that satisfy $S \cap T = U$, whether there exists a monoid jointly extending S and T.*

2 Theorem 2: Proof strategy

This section and the next section are devoted to a proof of Th. 2. We begin by outlining the proof strategy.

When rewriting a word v into a word w in the system $R_\Sigma \cup R_\Gamma$, each intermediate step can be decomposed into a number of blocks, which are words from either Σ^* or Γ^*. Since the system is a union of two systems, one over Σ and the other over Γ, a single rewriting rule can only be applied within such a block. At first glance, in order to rewrite the word v into w, we may need to introduce new blocks along the way; moreover, there is no apparent bound on the number of these introduced blocks. The essence of our proof is that such a bound in fact exists. We show that we need only consider derivations where almost all blocks can be found already in v or w. Then we show that, when the number of introduced blocks is bounded, the problem is decidable by a reduction to reachability in commutative rewriting systems and an application of Th. 1.

We now give a formal definition of derivation, and then state the two key results: Prop. 1, which says that only derivations with a bounded number of new blocks are needed, and Prop. 2, which says that reachability is decidable when restricted to such derivations. The proof of the former is outlined in Sec. 3.

A derivation is a proof that one word can be rewritten into another, annotated with some geometrical structure. We will use this structure when manipulating derivations.

A *derivation* is a sequence of rows. A *row* contains a *global state*, which is simply a word u over $\Sigma \cap \Gamma$, and a sequence of nodes labeled by words over Σ or words over Γ (in particular, labels with words over $\Sigma \cap \Gamma$ are allowed as well). Each row corresponds to a word in the derivation: the concatenation of all the node labels. The first row in a derivation is called the *source* row, while the last row is called the *target* row. The most important information in the derivation is an edge relation. The idea is that an edge connects a node in one row with the corresponding node in the next row. There are several ways – called *rules* – in which nodes in one row can be connected to nodes in the next row. Because the global state and labels are elements of Σ^* or of Γ^*, and because we are interested in commutative string rewriting systems, the order of letters in the global state and in the labels is irrelevant. Therefore, we treat the global state and the labels as multisets, and if v and w are two such multisets, then we consider them equal if each letter appears in v and w the same number of times. Concatenation on such words is really just multiset union, and we denote such a union by $v + w$. The empty multiset is denoted by ϵ.

The rules on how one row may be connected to the next row are the following:

- *Transition* rule. This is the only rule where the underlying rewriting system is invoked. In this case, the label of one of the nodes is rewritten according to the system R_Σ or R_Γ. The global state can also be used in this rule. More formally, if $v + u$ rewrites in one step to $w + u'$ in the system R_Σ or R_Γ, where $u, u' \in (\Sigma \cap \Gamma)^*$, then two rows can be connected as follows (in the first column, we have the global state, in the subsequent columns we have the contents of the rows):

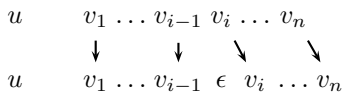
$$\begin{array}{cccccccc} u & v_1 & \dots & v_{i-1} & v & v_{i+1} & \dots & v_n \\ & \downarrow & & \downarrow & \downarrow & \downarrow & & \downarrow \\ u' & v_1 & \dots & v_{i-1} & w & v_{i+1} & \dots & v_n \end{array}$$

  The remaining rules are structural rules, which account for creating, deleting, separating and merging nodes.
- *Load* rule. In this rule a value u_2 from the global state $u + u'$ is loaded into a node with label v:

$$\begin{array}{cccccccc} u + u' & v_1 & \dots & v_{i-1} & v & v_{i+1} & \dots & v_n \\ & \downarrow & & \downarrow & \downarrow & \downarrow & & \downarrow \\ u & v_1 & \dots & v_{i-1} & v + u' & v_{i+1} & \dots & v_n \end{array}$$

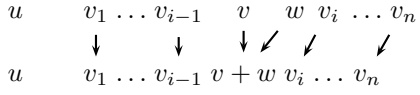
  This rule has a dual *Store* rule, which works exactly in the opposite direction, taking a value $u' \in (\Sigma \cap \Gamma)^*$ from a node labeled $v + u'$ and storing it into the global state. (These rules could be simulated by transition rules, if both rewriting systems contained the rule $\epsilon \mapsto \epsilon$.)

– *Creation* rule. In this rule, a new node with empty label is created. The new node has indegree 0. The graph looks as follows:



This rule has a dual *Delete* rule, again working in the exact opposite direction: it removes a node labeled by ϵ.

– *Merge* rule. In this rule, two neighboring nodes with labels v and w are merged into one node with label $v + w$ without affecting the global state. In this case v and w must either both belong to Σ^*, or both belong to Γ^*.



Again, the merge rule has a dual *Split* rule, where a node with label $v + w$ is split into two nodes with labels v and w.

A derivation is just an annotated way of showing that a word v can be rewritten into another word w in $R_\Sigma \cup R_\Gamma$. This is formally stated in the following lemma:

Lemma 1. *For any words $v = a_1 \dots a_n$ and $w = b_1 \dots b_k$ over $\Sigma \cup \Gamma$, v rewrites to w in $R_\Sigma \cup R_\Gamma$ iff there is a derivation with a source row containing a global state ϵ and nodes labeled $a_1, \dots, a_n$, and a target row containing a global state ϵ and nodes labeled $b_1, \dots, b_k$.*

Note that the large majority of edges connect nodes with the same labels. A rule is said to *act* on a node if this is not the case. Formally, the transition rule acts on two nodes (with labels v and w), the store and load rules act on two nodes (with labels $v + u'$ and v), the creation and deletion rules act on one node each (with label ϵ), while the merge and split rules act on three nodes each (with labels v, w and $v + w$).

Two nodes in a derivation are considered *connected* if they are connected by an edge (orientation of the edges is irrelevant). A *component* is a maximal set of nodes that can be pairwise connected by a sequence of edges.

In general, a component can contain labels from both Σ^* and Γ^*, since edges can go from Γ^* to $(\Gamma \cap \Sigma)^*$, and then to Σ^*. However, we can absorb the word from $(\Gamma \cap \Sigma)^*$ into the global state using a load and a deletion rule, and then recreate and restore this node (in a new component) using a creation and store rule, which shows:

Lemma 2. *For any derivation, there exists a derivation with the same source and target rows and such that each of its components contains nodes that are either all labeled by words from Σ^*, or all labeled by words from Γ^*.*

From now on we shall assume that all derivations are of the above type.

Th. 2 is an immediate corollary of Lemmas 1 and 2, and the following two propositions:

Proposition 1. *For any derivation, there is a derivation that has the same source and target rows and uses at most $k + 2$ creation rules, where k is the number of nodes in the target row.*

Proposition 2. *Given $k \in \mathbb{N}$ and source and target rows, it is decidable whether there is a derivation containing at most k creation rules with the given source and target rows.*

3 Theorem 2: eliminating creation rules

In this section we outline our proof of Prop. 1.

Our strategy is as follows. First we show in Sec. 3.2 that without loss of generality one may consider only derivations without *islands* (islands are components that intersect neither the source, nor the target row). Then we show in Sec. 3.3 and 3.4 that every derivation is equivalent to one where each component contains at most one creation rule. This is done in two steps. First in Sec. 3.3 we show that a weaker condition – called compactness – can be obtained. Then in Sec. 3.4 we show how compactness implies Prop. 1.

First however, we need to provide some auxiliary definitions.

3.1 Partial derivations

A partial derivation is a generalized type of derivation that can also use a *silent* rule:

$$\begin{array}{ll} u & v_1 \ldots v_n \\ & \downarrow \quad \downarrow \\ u' & v_1 \ldots v_n \end{array}$$

Partial derivations are used to decompose non-partial ones into pieces: if we remove part of a derivation, we must still keep track of the changes to the global state that are caused by the removed part. These changes are witnessed by the silent rule.

Two partial derivations D, D' are considered equivalent, if they have the same number of rows, same source and target rows, same global states, and use silent rules in the same rows.

At the risk of confusion, we will omit the word partial from the term partial derivation. The previously defined derivations – ones without silent rules – will be referred to as non-partial.

We now introduce two operations on derivations: one extracts a smaller derivation from a larger one, the other combines several derivations into a single one.

Subderivations. Let X be a union of components. We denote by $D[X]$ the derivation where only nodes coming from X are left, and the other nodes are removed. Moreover, rules that acted on nodes outside X are replaced by silent rules (which may modify the global state). The following lemma proves the correctness of this construction:

Lemma 3. *If X is a union of components in D, then $D[X]$ is a derivation.*

We say that derivations $D_1, \ldots, D_n$ are *compatible* if they have the same number of rows, they have the same global states in each row, and for each row, at most one of the derivations uses a non-silent rule. Compatible derivations $D_1, \ldots, D_n$ can be joined into a bigger derivation $D_1 \cdots D_n$ by concatenation. In each row of the concatenated derivation, we have a concatenation of the appropriate rows in $D_1, \ldots, D_n$.

Lemma 4. *If derivations $D_1, \ldots, D_n$ are compatible, then the concatenation $D_1 \cdots D_n$ is a derivation.*

3.2 Island removal

Recall that an island in a derivation is a component that has nodes neither in the source, nor in the target row. As a first step, we eliminate islands. This process is rather straightforward: we move all islands to the right side of the derivation. Since all islands are either of type Σ^* or of type Γ^*, they can be squeezed into two components. We will from now on assume that the derivation does not contain any islands.

3.3 Compact components

Recall that we want to convert a derivation into one where each component has at most one creation rule. In this section we prove a weaker version of this statement, where only the number of "unguarded" creation rules is bounded.

A *neighbor* of x is a node in the same row, which is directly to the left or right of x. A node is *guarded* if it has a neighbor in the same component. A component is *compact* if each created node is either guarded or is the only node of the component in its row. In particular, a compact component has at most one unguarded created node.

The following proposition is the main result of this section:

Proposition 3. *Every derivation is equivalent to one with all components compact.*

The rest of this section is devoted to a proof of Prop. 3. The proof is by induction on the number of components in the derivation. Before we proceed with the proof, we introduce several auxiliary concepts.

Paths and enclosures A path is a connected set of nodes X such that each node $x \in X$ is connected with at most two nodes from X. Nodes $x \in X$ that are connected to exactly one node in X are called *ends* of the path. A path with at least two nodes has exactly two ends. Note that the path need not be a directed path: for instance, if two nodes x, y in one row are merged into a node $x \cdot y$ in the next row, then the three nodes $\{x, y, x \cdot y\}$ form a path, whose ends are the nodes x and y.

An *enclosure* is a path X, whose ends $x, y \in X$ are either both in the source row, or both in the target row. The nodes z that satisfy $x < z < y$ are called

the *base* of the enclosure (the order $<$ and its non-strict version $\leq$ refer to which node comes first in a row). An enclosure is called *tight* if its base does not contain nodes in the same component as X. The *cobase* is the set of nodes in the source and in the target rows that are not in the base and are not x, y. An enclosure X partitions the nodes of a derivation into three parts:

- The enclosure X itself.
- The *interior* $in(X)$ of the enclosure. These are the nodes that can be connected to the base of the enclosure without passing through X.
- The *exterior* $ex(X)$ of the enclosure. These are the remaining nodes. Stated differently, these are the nodes that can be connected to the cobase of the enclosure without passing through X.

Note that both the exterior and interior can contain nodes from the component of X. We extend the notions of interior and exterior to arbitrary sets of nodes X: the interior $in(X)$ is the union of all interiors $in(Y)$ of all enclosures $Y \subseteq X$. The exterior is the intersection of all the exteriors $ex(Y)$.

We say that a set of nodes Y is *enclosed* by a set of nodes X if $Y \subseteq in(X)$.

The height of a set of nodes is the number of rows used by this set. The *inner depth* of an enclosure X is the maximal height of an enclosed component. The *outer depth* is the height of the enclosure.

One outermost component When there is only one component, the statement is trivial. A component is *outermost* in a derivation if it is not enclosed by any other component. It can be shown that one only needs to consider derivations with one outermost component:

Lemma 5. *Without loss of generality, one can consider only derivations with one outermost component.*

Compacting By the above lemma, we may assume that the derivation in Prop. 3 has only one outermost component, which we call X. We now proceed to the heart of the proof of Prop. 3: making components compact.

The procedure we are going to use does not create any new nodes, nor does it modify the type of rules used or the global state. It only rearranges the order of nodes in each row. In particular, all the transformed derivations will have the same nodes.

The following straightforward lemma is given without proof:

Lemma 6. *One can find tight enclosures $X_1, \ldots, X_n$ with pairwise disjoint interiors such that all components enclosed by X are enclosed by one of the X_i.*

For each $i = 0, \ldots, n$, we will inductively correct our derivation, so that the following invariant is satisfied:

- All components enclosed by $X_1, \ldots, X_i$ are compact.
- For $j = 1, \ldots, i$, one cannot find nodes $y_1 \leq x \leq y_2$ lying in one row and such that $x \in X$ and the nodes $y_1, y_2 \notin X$ are enclosed by X_j.

It can be shown that at the end of the process all components become compact, and hence Prop. 3 is obtained:

Lemma 7. *If the invariant is satisfied for $i = n$, then all components in the derivation are compact.*

So, we now proceed to prove the invariant. The base case $i = 0$ is immediate. So let us assume that the invariant holds for $i - 1$; our aim is to make it hold for i.

We assume without loss of generality that the base of X_i is in the source row. Let $\mathcal{Y}$ be the components that are enclosed by X_i. Each of these components has nodes in the source row, but no nodes in the target row (since they are enclosed by X_i). Let y be a node of maximal depth k (row number) among all nodes in components from $\mathcal{Y}$. One can see that y is unique, since it must be acted upon in a deletion rule, and in each row only one rule is applied. Let Y be any path with one end in y and the other end in the source row (such a path must exist, since there are no islands). By maximality of k, both the path Y and all components in $\mathcal{Y}$ are contained in the rows up to k.

We define three groups of nodes of the derivation:

- The nodes A (for "above") whose depth is at most k;
- The nodes B (for "below") whose depth is strictly greater than k.
- The nodes C that belong to one of the components $\mathcal{Y}$.

Note that $A \cap B = \emptyset$ and $C \subseteq A$. Our strategy is to partition the set $A \setminus C$ into two parts: A_0 and A_1. The idea is that nodes in A_0 are to the left of the path Y, while nodes in A_1 are to the right. Now as it is, this is not a clear concept, since the path Y may bend and turn many times; hence the need for a more formal definition. The definition we provide is based on the parity of the number of turns in the path Y. Using this definition, we will reorder the nodes in A so that in each row, all nodes in A_0 are to the left of all nodes in A_1. Finally, we will apply the induction assumption to C, and then place it between A_0 and A_1 as in Fig. 1.

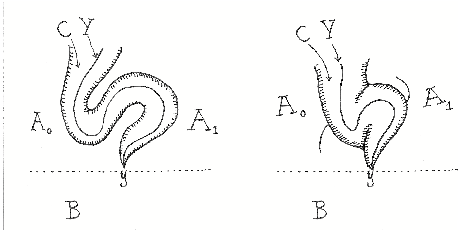


Fig. 1. Reordering the derivation to make it compact.

Let x be a node in Y. We say x node is *bending* if it is the target of merge of two nodes from Y or the source of a split into two nodes from Y. Otherwise

the node is called *nonbending*. Note that both ends of Y are nonbending. Given a node $x \notin Y$, we write $\#(x)$ for the number of nonbending nodes in Y that are in the same row as x and to the left of x. (This value is 0 when x is outside A.) We define A_0 (resp. A_1) to be the set of nodes $x \in A \setminus C$ where $\#(x)$ is even (resp. odd).

Let $D[C]$ be a derivation obtained from D by only leaving the nodes from C. By the induction assumption, $D[C]$ is equivalent to a compact derivation E. This derivation E only uses rows $1, \ldots, k$. We now construct a derivation D' where the components contained by enclosures $X_1, \ldots, X_i$ are compact. This derivation is obtained from D as follows.

- A row $j = 1, \ldots, k$ of the derivation D' is obtained as follows. We first place all the nodes from row j in D that belong to A_0 (preserving the order from D). Then we insert the j-th row of E. Finally, we place the nodes from row j in D that belong to A_1 (again, preserving the order).
- The rows $> k$ are left unaltered.

The following lemmas show the correctness of this construction; this concludes the proof of Prop. 3.

Lemma 8. *D' is a derivation.*

Lemma 9. *In D' the invariant holds for i.*

3.4 Few creation rules

In this section we conclude the proof of Prop. 1. We show that a derivation without islands and with all components compact can be transformed into one with few creation rules. Together with the island removal from Sec. 3.2 and the compacting procedure from Prop. 3, this concludes our proof of Prop. 1.

Let D be a derivation, and x a created node. The node x lies in some component X, which is compact. Since X is compact, x must either be guarded, or be the only node of the X in its row. We show that if x is guarded, then the derivation can be modified so that all nodes remain as in D, and all rules but the one creating x remain as in D as well; the rule creating x will be replaced by a split rule.

The modification of D is straightforward. If x is guarded, then it must have a neighbor x' from X on its left or right side. Suppose x' lies to the right of x and has label v:

$$\begin{array}{lc} u & v_1 \ldots v_{i-1}\ v\ v_i \ldots v_n \\ & \downarrow \quad \downarrow \quad \searrow\ \searrow \quad \searrow \\ u & v_1 \ldots v_{i-1}\ \epsilon\ v\ \ v_i\ \ldots v_n \end{array}$$

All we have to do is replace this creation rule by a split rule. This is done by adding an arrow from the node labeled v in the upper row to the node x in the lower row. Since $v = \epsilon + v$, we obtain a legitimate split rule. The same procedure may be used if x' lies to the left of x. Applying this construction

iteratively gives us a derivation where all created nodes are the only nodes of their component in their row. Thus, in components that intersect with the source row, no created nodes may appear. In other components, at most one created node may appear per component. Since there are no islands, all the other components must intersect with the target row. Therefore there are at most as many creation rules as there are nodes in the target row.

4 Embeddability of unions of commutative monoids

Any symmetric string rewriting system R over an alphabet Σ naturally defines a monoid M_R: it is the quotient of the free monoid Σ^* by the least congruence containing the rules of R. Elements of the monoid M_R are equivalence classes of the form $[v]$, where v is a word over Σ; classes $[v]$ and $[w]$ are equal iff v rewrites to w in R. The *word problem* for M_R is defined as follows: given words v and w over Σ, answer whether $[v] = [w]$ holds, that is, whether v and w represent the same element of the monoid M_R. Th. 2 implies that the word problem is decidable for monoids M_R, where R is the union of two commutative symmetric string rewriting systems; since pushouts of commutative monoids are of this form, Cor. 1 follows.

We now turn to the embeddability of unions of commutative monoids. Let U, S and T be commutative monoids given by symmetric commutative string rewriting systems R_U, R_S and R_T over the alphabets $\Sigma \cap \Gamma$, Σ and Γ respectively, satisfying $S \cap T = U$. Note that words u, u' over $\Sigma \cap \Gamma$ rewrite one to another in R_U if and only if they rewrite to each other in R_S (resp., R_T). This follows from the fact that U is a submonoid of S (resp., T).

We ask whether a monoid M exists such that $S \cup T \subseteq M$. A second question is whether the monoid M can be commutative. It is well-known [5] that if such an M exists, then as M one may take the pushout of the inclusions of U into S and of U into T (if M is to be commutative, then one takes the pushout in the category of commutative monoids). This is expressed by the slogan: if $S \cup T$ is embeddable, then it is embeddable in its pushout. Let P together with homomorphisms $\sigma : S \to P$, $\tau : T \to P$ be the pushout (or commutative pushout). All that has to be checked is whether σ and τ define an embedding of $S \cup T$ in the pushout, that is, whether they are "jointly injective". Formally,

Lemma 10. *$S \cup T$ is embeddable if and only if:*

1. *σ and τ are injective,*
2. *for all $s \in S$ and $t \in T$, if $\sigma(s) = \tau(t)$, then $s = t \in U$.*

The pushout and commutative pushout of $S \cup T$ may be defined easily. The pushout is given by the union $R_{nc} = R_S \cup R_T$ over $\Sigma \cup \Gamma$, while the commutative pushout is given by the union $R_c = R_S \cup R_T \cup \{ab \mapsto ba | a \in \Sigma, b \in \Gamma\}$ over $\Sigma \cup \Gamma$; here σ and τ are the canonical homomorphisms. This together with Lemma 10 leads to the following result:

Lemma 11. *$S \cup T$ is embeddable if and only if:*

1. *for all words v, v' over Σ, if v rewrites to v' in R_{nc}, then the same is true in R_S, and an analogous implication holds for Γ and R_T,*
2. *for all words v over Σ and w over Γ, if v rewrites to w in R_{nc}, then there is a word u over $\Sigma \cap \Gamma$ such that w rewrites to u in R_S and u rewrites to w in R_T.*

An analogous equivalence, with R_{nc} is replaced by R_c, holds for embeddability in a commutative monoid.

It can be shown that in the commutative case the above conditions can be effectively checked, and thus Th. 4 follows. The proof is based on the following powerful result of Taiclin [10]:

Theorem 6. *Given a symmetric commutative string rewriting system over a finite alphabet $\{a_1, \ldots, a_k\}$, one can compute a Presburger formula θ of $2k$ variables, such that:*

$$(n_1, \ldots, n_k)\, \theta\, (m_1, \ldots, m_k)$$

holds in $(\mathbb{N}^k, 0^k, +^k)$ if and only if $a_1^{n_1} \ldots a_k^{n_k}$ rewrites to $a_1^{m_1} \ldots a_k^{m_k}$ in the system.

As for embeddability in a monoid which is not necessarily commutative (Th. 5), this follows from Th. 4 and from the following proposition:

Proposition 4. *If a union of commutative monoids is embeddable, then it is embeddable in a commutative monoid.*

References

1. Mayr, E.W.: An algorithm for the general Petri net reachability problem. SIAM J. Comp. **13**(3) (1984) 441–459
2. Kosaraju, S.R.: Decidability of reachability in vector addition systems (preliminary version). In: STOC'82, ACM (1982) 267–281
3. Sapir, M.V.: Algorithmic problems for amalgams of finite semigroups. J. Algebra **229**(2) (2000) 514–531
4. Hoffman, P.: Unions of equational monadic theories. In: RTA'06. Volume 4098 of LNCS. (2006) 81–95
5. Howie, J.M.: Fundamentals of Semigroup Theory. London Math. Soc. Monogr. (N.S.), No. 12. Oxford Univ. Press (1996)
6. Howie, J.M.: Embedding theorems with amalgamation for semigroups. Proc. London Math. Soc. (3) **12** (1962) 511–534
7. Howie, J.M.: Epimorphisms and amalgamations: A survey of recent progress. Coll. Math. Soc. J. Bolyai **39** (1981) 63–82
8. Hall, T.E.: Representation extension and amalgamation for semigroups. Quart. J. Math. Oxford (2) **29**(2) (1978) 309–334
9. Birget, J.C., Margolis, S., Meakin, J.: On the word problem for tensor products and amalgams of monoids. Intnl. J. Alg. Comp. **9** (1999) 271–294
10. Taiclin, M.A.: Algorithmic problems for commutative semigroups. Soviet Math. Dokl. **9**(1) (1968) 201–204