A Decidable Instance of the Inclusion Problem for Rational Relations

Wojciech Fraczak, Stéphane Hassen

***Abstract*—We study an instance of the inclusion problem for rational relations over words. In this paper we show how to check if a one generator submonoid $\{w\}^* \subseteq (\Sigma^*)^d$ is included in the prefix closure of a rational relation $R \subseteq (\Sigma^*)^d$ given by a multi-tape finite automaton.**

***Index Terms*—formal language, inclusion problem, multitape automata**

I. Introduction

Multi-tape finite automata play an important role in many areas of the theoretical computer science, computational linguistics, and even software engineering, [2], [5]. However, unlike the usual (one-tape) automata, the multi-tape automata are "difficult". For the rational word relations, the semantic domain of finite multi-tape automata, the *membership* problem is decidable wheras *equality*, and thus *inclusion*, are not [6], [7].

In this paper we study a particular case of the inclusion problem of rational word relations. Let $M = \{w\}^*$ be a one generator submonoid of $(\Sigma^*)^d$ and $R \subseteq (\Sigma^*)^d$ a rational word relation given by a finite multi-tape automaton. We want to check if $M \subseteq \downarrow R$, i.e., if the set of all prefixes of R, denoted by $\downarrow R$, includes M (or, equivalently, if for every $k \geq 0$, w^k is a prefix of an element of R).

Firstly, we show that the problem is indeed decidable by providing an exponential time, but simple, algorithm solving it. Secondly, we show that, assuming the number of dimensions d being a constant, the problem can be solved in a polynomial time. It is worth noting that for $d = 1$ and when R is not necessarily prefix-closed the problem of checking if $w^* \subseteq R$ is co-NP-hard [8], [4].

II. Definitions

Let Σ be a finite set of *letters*, usually called *alphabet*. By Σ^* we denote the set of all *words* which are finite sequences over Σ. The empty word is denoted ε.

A *multiword* of dimension $d \geq 1$, also called d-word, is a d-dimensional vector of words. The i^{th} dimension of a multiword w will be refered to by $w[i]$. We introduce the notation (i, u), where $i \in \{1, \ldots, d\}$ and $u \in \Sigma^*$, to describe the multiword having the empty word ε on each dimension but the i-th one which is u, i.e.:

$$(i,u)[k] \stackrel{\texttt{def}}{=} \begin{cases} u & \text{if } k = i \\ \varepsilon & \text{otherwise.} \end{cases}$$

The set $(\Sigma^*)^d$ of all d-words over an alphabet Σ constitute a monoid where the *concatenation* is defined componentwise

$$(u[1], \ldots, u[d]) \cdot (v[1], \ldots, v[d]) \stackrel{\texttt{def}}{=} (u[1]v[1], \ldots, u[d]v[d])$$

Département d'informatique et d'ingénierie, Université du Québec en Outaouais, C.P. 1250, succursale Hull, Gatineau (Qué) Canada, J8X 3X7.

and the neutral element $\mathbb{1}$ is the d-word composed by ε on each of its dimensions.

The *length* $|w|$ of a multiword w is its total number of letters. Also, if S is a set, by $|S|$ we denote its cardinality. *Active dimention* of a multiword w, denoted by $d(w)$, is the set of all non-empty dimensions of w, i.e., $i \in d(w)$ if and only if $w[i] \neq \varepsilon$, for $i \in \{1, \ldots, d\}$.

Given a set D of dimensions, $D \subseteq \{1, \ldots, d\}$, and a d-word w, the multiword $w \setminus D$ is defined as:

$$(w \setminus D)[i] \stackrel{\texttt{def}}{=} \begin{cases} \varepsilon & \text{if } i \in D \\ w[i] & \text{otherwise} \end{cases}$$

Intuitively, $w \setminus D$ represents the *deactivation* of dimensions D in w. For example, we have: $w \setminus \{1, \ldots, d\} = w \setminus d(w) = \mathbb{1}$.

Let $u, w \in (\Sigma^*)^d$ be two multiwords. We say that u is a *prefix* of w if there exists a multiword v such that $uv = w$. In such a case v is unique and it will be denoted by $u^{-1}w$.

The set of all prefixes of a multiword w will be denoted by $\downarrow w$.

Proposition 1: Let $w \in (\Sigma^*)^d$ be a multiword. We have:

$$|\downarrow w| = \prod_{i=1}^{d} (|w[i]| + 1) \leq \left(\left\lceil \frac{|w|}{d} \right\rceil + 1 \right)^d.$$

Given two multiwords u, w we define the *rotation* of w by u which consists in concatenating w with u and removing the leading u from it, i.e.:

$$u@w \stackrel{\texttt{def}}{=} u^{-1}wu.$$

For example, $(a, aba, a)@(aba, ab, \varepsilon) = (baa, ba, \varepsilon)$.

By $@w$ we denote the set of all rotations of w, i.e. $u \in @w$ if and only if there is a v such that $u = v@w$.

Proposition 2: Let $w \in (\Sigma^*)^d$ be a multiword. We have:

$$|@w| \leq |\downarrow w|.$$

Proof: The key point is to notice that the rotation operation is length preserving.

When $d = 1$, consider w to be written on a ring torus. Then the number $|@w|$ of rotations of w is the number of distinct words of length $|w|$ that can be read on this torus. When $|w| \leq 1$, there is only one such word, w itself. Hence $|@w| = 1$. Otherwise, the word read depends on which letter is considered to be the first. Hence $|@w| \leq |w|$. When $d > 1$, consider d tori, one torus for each dimension. For each dimension the previous reasoning holds and the number of different words read on the d tori is the number of rotations of w:

$$|@w| = \prod_{i=1}^{d} |@w[i]| \leq \prod_{i=1}^{d} |w[i]| \leq |\downarrow w|.$$

■

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Subsets of $(\Sigma^*)^d$ are called *relations* on words. The rational operations (concatenation, union and Kleene-star operation) are defined as usual. Let $R, R' \subseteq (\Sigma^*)^d$:

$$\begin{aligned} RR' &\stackrel{\text{def}}{=} \{uv \mid u \in R \text{ and } v \in R'\} \\ R + R' &\stackrel{\text{def}}{=} R \cup R' \\ R^* &\stackrel{\text{def}}{=} \bigcup_{k\in\mathbb{N}} R^k, \text{ with } R^k \stackrel{\text{def}}{=} \begin{cases} \{\mathbb{1}\} & \text{if } k = 0 \\ RR^{k-1} & \text{if } k > 0 \end{cases} \end{aligned}$$

We define $\downarrow R$ as the set of all prefixes of R, i.e.:

$$\downarrow R \stackrel{\text{def}}{=} \bigcup_{w\in R} \downarrow w.$$

A common way to define relations on words is by automata. We present here a straight generalization of one-tape finite automata.

A *multi-tape finite automaton* of dimension d, or d-tape automaton for short, is a tuple $A = (Q, \Sigma, d, \delta, I, F)$ where Q is a finite set of states, Σ an alphabet, $d > 0$ a number of tapes (dimensions), $\delta \subseteq Q \times (\Sigma^*)^d \times Q$ a finite transition relation, and $I, F \subseteq Q$ are sets of initial and final states, respectively.

An *execution* $\pi \in \delta^*$ *starting in* state q and *ending in* state p is a sequence of transitions $\pi = (q_0, u_0, p_0) \dots (q_k, u_k, p_k)$ such that $q = q_0$, $p = p_k$, and for all $i \in \{1, \dots, k\}$, $q_i = p_{i-1}$. We then say that the state p is *accessible* from q and that q is co-accessible from p. The *label* $\lambda(\pi)$ of execution π is the d-word $u_0 u_1 \cdots u_k$. If $p = q$ then π can be the empty sequence of transitions, describing the empty execution with label $\mathbb{1}$.

An execution starting in an initial state is called an *initial* execution. An initial execution ending in a final state is said to be *accepting*. The set of labels of all accepting executions defines a relation $R(A) \subseteq (\Sigma^*)^d$.

It is well known that the family of all relations which can be defined by a finite multitape automaton coincides with the closure of the finite relations by rational operations. The elements of this family are called *rational relations*.

III. Preliminary results

In the multitape automaton model presented here we haven't considered an order on tapes since rational relations are closed by "tape swapping". As pointed in [1], if the alphabet Σ is a singleton, rational relations correspond to the rational commutative languages. Thus, in this case the general inclusion problem is decidable (e.g., using the Presburger arithmetic decidability results, [3]).

Let's now give properties in the general case, $|\Sigma| \geq 1$.

Proposition 3: Let $R_1, R_2 \subseteq (\Sigma^*)^d$. If $R_1 R_2 = R_2 R_1$, then $(R_1 + R_2)^* = R_1^* R_2^*$.

Proof: Firstly, we show that $(R_1 + R_2)^k \subseteq R_1^* R_2^*$, for every $k \geq 0$. Indeed, since we have $(R_1 + R_2)^k = \bigcup_{i=0}^{k} R_1^i R_2^{k-i}$, and $R_1^i \subseteq R_1^*$, $R_2^{k-i} \subseteq R_2^*$, for every $k \geq i \geq 0$.

The inclusion $R_1^* R_2^* \subseteq (R_1 + R_2)^*$ holds for any relations (i.e., even if $R_1 R_2 \neq R_2 R_1$) because $R_1^* \subseteq (R_1 + R_2)^*$, $R_2^* \subseteq (R_1 + R_2)^*$, $(R_1 + R_2)^*(R_1 + R_2)^* = (R_1 + R_2)^*$, and the concatenation is monotonous with respect to the inclusion order. ∎

Proposition 4: Let $R_1, R_2 \subseteq (\Sigma^*)^d$ be two non-empty relations. If $R_1 R_2 = R_2 R_1$, then $\downarrow(R_1 R_2)^* = \downarrow(R_1^* R_2^*)$.

Proof: Since $R_1 R_2 = R_2 R_1$, we have $(R_1 R_2)^* \subseteq R_1^* R_2^*$. Hence $\downarrow(R_1 R_2)^* \subseteq \downarrow(R_1^* R_2^*)$.

If $w \in R_1^* R_2^*$ then $w \in R_1^k R_2^l$, for some integers k, l. Thus, $w \in \downarrow(R_1^{\max(k,l)} R_2^{\max(k,l)}) = \downarrow(R_1 R_2)^{\max(k,l)} \subseteq \downarrow(R_1 R_2)^*$. ∎

Proposition 5: If R is a rational relation then so is $\downarrow R$.

Proof: Given an automaton $A = (Q, \Sigma, d, \delta, I, F)$ defining a rational relation $R = R(A)$, we build an automaton $\downarrow A$ which defines the relation $\downarrow R = R(\downarrow A)$. Without loss of generality we assume that in A the transitions are labeled by words of length at most 1.

We set $\downarrow A = (Q \times 2^d, \Sigma, d, \delta', I', F')$, where:

- 2^d denotes the set of all subsets of $\{1, \dots, d\}$;
- transitions are defined by:

$$\begin{aligned} \delta' &= \{(p, D) \xrightarrow{w\backslash D} (q, D) \mid (p, w, q) \in \delta\} \\ &\cup \{(p, D) \xrightarrow{\mathbb{1}} (p, D \cup \{i\}) \mid i \in \{1, \dots, d\}\}. \end{aligned}$$

- initial states are: $I' = I \times \{\emptyset\}$;
- final states are: $F' = F \times \{\{1, \dots, d\}\}$.

Intuitively, $\downarrow A$ mimics the behavior of A, however, in $\downarrow A$, at any moment we may decide to stop reading any dimension i by taking a transition of the form $(p, D) \xrightarrow{\mathbb{1}} (p, D \cup \{i\})$. ∎

The following lemma states that deciding the problem of the inclusion of a one-generator submonoid w^* in a prefix-closed relation $\downarrow R$ is the same as checking the inclusion of the monoid $((1, w[1]) + \ldots + (d, w[d]))^*$ induced by d generators, one per tape (dimension).

Lemma 6: Let $w \in (\Sigma^*)^d$ and $R \subseteq (\Sigma^*)^d$. We have:

$$w^* \subseteq \downarrow R \iff ((1, w[1]) + \ldots + (d, w[d]))^* \subseteq \downarrow R.$$

Proof: Since $\downarrow R$ is prefix-closed, $w^* \subseteq \downarrow R$ if and only if $\downarrow w^* \subseteq \downarrow R$. By Propositions 4 and 3 we have: $\downarrow w^* = \downarrow((1, w[1])^* \cdots (d, w[d])^*) = \downarrow((1, w[1]) + \ldots + (d, w[d]))^*$. ∎

IV. Problem statement

Let $R(A) \subseteq (\Sigma^*)^d$ be a rational relation given by a finite multi-tape automaton $A = (Q, \Sigma, d, \delta, I, F)$, and $w \in (\Sigma^*)^d$ a multi-world. We want to check whether w^* is included in the prefixes of $R(A)$:

$$w^* \stackrel{?}{\subseteq} \downarrow R(A).$$

In the next section we give a simple procedure which will always produce a correct answer. A straightforward implementation of the algorithm is highly inefficient since it may have to explore all paths in A of length up to $|Q| \times |\downarrow w| \times d$. On the other hand, our presentation makes the main ideas of the procedure clear and it facilitates the correctness proof. Later, in Section VI, we discuss how the performance of the algorithm can be improved.

V. Decision algorithm

Without loss of generality, we assume that the automaton $A = (Q, \Sigma, d, \delta, I, F)$ is co-accessible, i.e., there is a path from every state in Q to a state in F.

For every state $q \in I$, we build a rooted tree T_q, which represents executions of A starting in q. The edges of T_q

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

are labeled by d-words, according to transitions of A. The vertexes of T_q are labeled by pairs (p, u), where the first component, p, is a state from Q, and the second component, u, is a rotation of w, where some dimensions may have been deactivated, i.e., $u \in @(w \setminus D)$ for some $D \subseteq \{1, \ldots, d\}$.

The label of the root vertex of T_q is (q, w). Each outgoing edge from a vertex labeled by (p, u) is in the correspondence with a transition of A from state p. We follow the transitions labeled by multiwords v such that $v@u$ is defined. For such a transition $(p, v, q) \in \delta$, the corresponding edge in T_q is labeled by v and the newly accessed vertex x is labeled by $(q, v@u \setminus D)$, where D is the set of dimensions of the path in T_q from an ancestor of x labeled by $(q, v@u)$ to vertex x. More precisely:

- If there is an ancestor y of vertex x in T_q labeled by $(q, v@u)$ and α is the multiword corresponding to the label of the path from y to x, then $D = d(\alpha)$;
  Intuitively, α represents a loop in A starting in q such that $\alpha@(v@u) = v@u$ and $v@u \setminus D$ represents $v@u$ with all dimensions of α deactivated.
- Otherwise, i.e., when none of the ancestors of x is labeled by $(q, v@u)$, $D = \emptyset$, and thus $v@u \setminus D = v@u$.

If the newly created vertex x has a label of one of its ancestors then the vertex x is removed from the tree.

The detailed pseudo-code of the above procedure is presented in Algorithm 1.

A. An example

Consider a ternary rational relation R defined by the automaton of Fig. 1 and $w = (ab, xy, ij)$. An initial part of the search tree which proofs that $w^* \subseteq \downarrow R$ is depicted in Fig. 3.

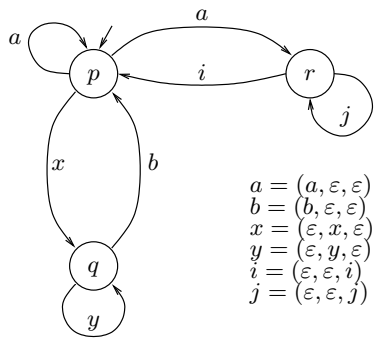

Fig. 1. A multi-tape relation R (all states are final).

B. Proof

Theorem 7: For any $w \in (\Sigma^*)^d$ and a multi-tape automaton A such that all its states are co-accessible from a final state, Algorithm 1 always terminates. The algorithm returns "true" if and only if $w^* \subseteq \downarrow R(A)$.

Proof: The program terminates since the depth of every search tree T_q is bounded by $|Q| \times |\downarrow w| \times d$.

The algorithm reports "true" if there is a search tree T_q with a witness leaf labeled by $(p, \mathbb{1})$, see Fig. 3. In order to show that indeed $w^* \subseteq \downarrow R(A)$, we prove that the (label of the) witness path, $\alpha_1\beta_1\alpha_2\beta_2\cdots\beta_n$, has the following properties:

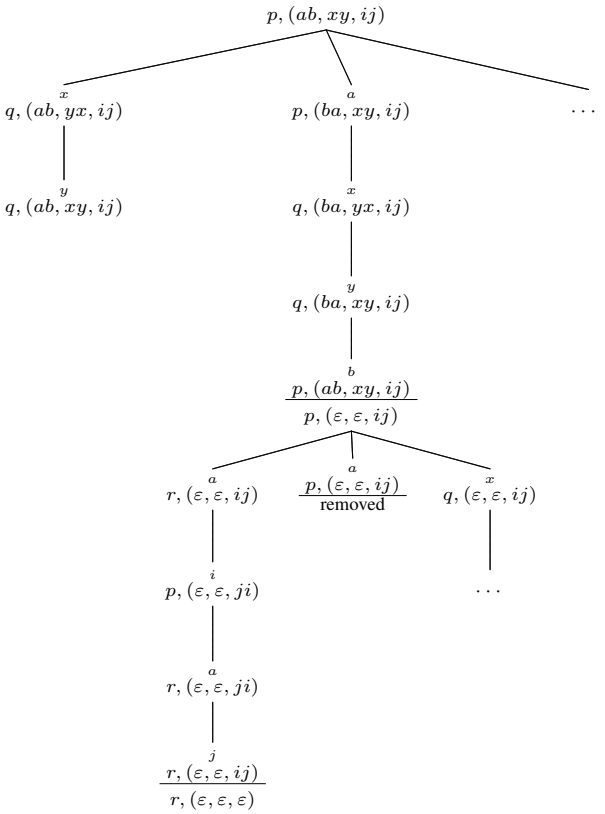

Fig. 2. A fragment of search tree T_p for $(ab, xy, ij)^* \subseteq \downarrow R$. "Fraction nodes" represent label adjustments. Leaf "$r, (\varepsilon, \varepsilon, \varepsilon)$" = "$r, \mathbb{1}$" is a witness of the inclusion.

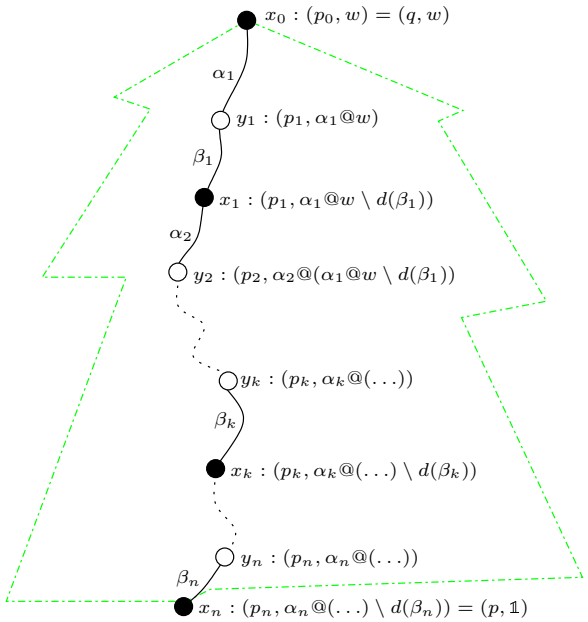

Fig. 3. A wintess path in a search tree T_q.

1) for every $k_1, k_2, \ldots, k_n \geq 0$, multiword $\alpha_1\beta_1^{k_1}\alpha_2\beta_2^{k_2}\cdots\beta_n^{k_n}$ belongs to $\downarrow R(A)$; and
2) for every $l \geq 0$, there exist $k_1, k_2, \ldots, k_n \geq 0$ such that w^l is a prefix of $\alpha_1\beta_1^{k_1}\alpha_2\beta_2^{k_2}\cdots\beta_n^{k_n}$.

The first property follows from the fact that, for $i \in \{1, \ldots, n\}$, β_i is a path in A from state p_i to itself (a loop) and that state $p_n = p$ is co-accessible from a final state.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Algorithm 1 Check whether $w^* \subseteq \downarrow R(a)$

Input: w: WORD, a: AUTOMATON

- **type** AUTOMATON = **struct** { Q: SetOf(STATE), δ: SetOf(TRANSITION), I: SetOf(STATE), ... };
- **type** TRANSITION = (STATE,WORD,STATE);
  -- Transition is a triple: (source state, transition label, destination state)
- **type** VERTEX = **struct** { parent: VERTEX, edge: WORD, label: (STATE,WORD) };
  -- Search tree is represented as a set of vertexes, each carrying the pointer to its `parent`*, the*
  -- label on the incoming `edge` *(from the parent), and its actual* `label`*, i.e., a pair (state,word).*

Output: **true** if $w^* \subseteq \downarrow R(a)$, **false** otherwise

```
var result: BOOLEAN ;
result := false;
for all p ∈ a.I do
    var root: VERTEX ;
    root := {parent ← nil, edge ← 𝟙, label ← (p,w)} ;
    result := result or Explore(a, root) ;
end for
return result ;
```

function **Explore**(a,x): BOOLEAN

Input: a: AUTOMATON, x: VERTEX

Output: **true** if a vertex with label $(*, \mathbb{1})$ is reachable from x

```
var s: STATE, w: WORD, result: BOOLEAN ;
(s,w) := x.label ; -- read the label of vertex x into local variables s and w
if (w == 𝟙) then
    return true ;  -- We found a witness, so the inclusion holds.
end if
result := false;
for all (p, u, q) ∈ a.δ do
    if (p == s) and u@w is defined then
        var y: VERTEX, v: WORD ;  -- New vertex y labeled (q, v) is being created...
        if LabelAlreadyExists(x,(q,u@w)) then
            v := u@w\ (Dims(x,(q,u@w)) ∪ d(u)) ;
            if (u@w == v) then
                break ; -- An ancestor with label (q, v) already exists: abandon y and continue with the for all loop.
            end if
        else
            v := u@w ;
        end if
        y := {parent ← x, edge ← u, label ← (q, v) } ;
        result := result or Explore(a,y) ;
    end if
end for
return result ;
```

function **LabelAlreadyExists**(x, l): BOOLEAN

Input: x: VERTEX, l: (STATE,WORD)

Output: **true** if x or one of its ancestors is labeled by l, **false** otherwise

```
if (x == nil) then
    return false
else if (x.label == l) then
    return true
else
    return LabelAlreadyExists(x.parent, l)
end if
```

function **Dims**(x, l): SetOf(INTEGER)

Input: x: VERTEX, l: (STATE,WORD)

Output: set of active dimensions between x and its ancestor (or itself) labeled by l. Vertex x or one of its ancestors MUST be labeled by l.

```
if (x.label == l) then
    return ∅ ;
else
    return d(x.edge) ∪ Dims(x.parent, l) ;
end if
```

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Consider vertexes y_k, x_k (see Fig. 3) labeled respectively by (p_k, u) and $(p_k, u \setminus d(\beta_k))$, with $u = \alpha_k @ (\ldots)$. Let i be a deactivated dimension in x_k, i.e., $i \in d(u) \cap d(\beta_k)$.

By construction, we have $u[i] = (\alpha @ w)[i]$, with $\alpha = \alpha_1 \beta_1^{k_1} \alpha_2 \beta_2^{k_2} \cdots \beta_{k-1}^{k_{k-1}} \alpha_k$ for any $k_1, \ldots, k_{k-1} \geq 0$, and $\beta_k[i] \neq \varepsilon$. There exist $j, l \geq 1$ such that $u[i]^j = \beta_k[i]^l$. Since $u[i] \in @w[i]$, there exist $x, y \in \Sigma^*$ such that $w[i] = xy$ and $u[i] = yx$. Because $(\alpha @ w)[i] = (\alpha \beta_k @ w)[i] = u[i]$, for all $l \geq 0$ there exists $j \geq 0$ such that $\alpha[i](\beta_k[i])^l = x(yx)^j$. Notice that j can be made arbitrary big by increasing l. Hence, for every $j \geq 0$ there will exist $l \geq 0$ such that $w[i]^j$ is a prefix of $\alpha(\beta_k)^l$.

By Lemma 6 and the fact that for every $j \geq 0$ and any dimension i we can always find sufficiently large $k_1, k_2, \ldots, k_n$ such that $w^j[i] \in \downarrow \alpha_1 \beta_1^{k_1} \alpha_2 \beta_2^{k_2} \cdots \beta_n^{k_n}$, we conclude that w^j is a prefix of $\alpha_1 \beta_1^{k_1} \alpha_2 \beta_2^{k_2} \cdots \beta_n^{k_n}$. ■

VI. Complexity evaluation

The complexity of the decision procedure for $w^* \subseteq \downarrow R$, as presented above, is exponential in the size of the automaton A for R, even for a fixed number of tapes d. However, assuming d being a constant, one can elaborate a polynomial time algorithm. For example, the following approach is possible:

1) We start by constructing a digraph (instead of the tree) whose vertexes are elements of $Q \times @w$, where Q is the set of states of automaton A and $@w$ denotes the set of all rotations of w; there is an edge labeled by $v \in (\Sigma^*)^d$ from vertex (p, u) to vertex (p', u'), i.e., $(p, u) \xrightarrow{v} (p', u')$, if and only if there exists a transition $p \xrightarrow{v} p'$ in A and $u' = v @ u$. Since d is a constant, the graph can be constructed in $O(|@w||A|)$, where $|@w|$ is the number of rotations of w and $|A|$ is the size of the automaton (by size of an automaton we mean the number of states plus the number of transition plus the total length of transition labels).
2) We find all strongly connected components of the constructed graphs accessible from an initial vertex, i.e., (p, w) with p being an initial state in A. With every such strongly connected component C we associate all *its dimensions*: dimension i is attached to C if and only if there is an edge with source and destination being in C and labeled by u such that $u_i \neq \varepsilon$, i.e., u reads at least a letter on the dimension i. Let $d(C)$ denotes the set of those dimensions.
   If there is C such that $d(w) \subseteq d(C)$ then the algorithm stops reporting success, i.e., $w^* \subseteq \downarrow R$.
   Otherwise, let $\mathcal{C} = \{C \mid d(C) \cap d(w) \neq \emptyset\}$, i.e., $\mathcal{C}$ is the set of those strongly connected components which touch active dimensions of w.
   If $\mathcal{C}$ is empty, then the algorithm stops reporting failure, i.e., $w^* \not\subseteq \downarrow R$.
3) For every C from $\mathcal{C}$ we generate a new sub-problem

$$(v \setminus d(C))^* \subseteq \downarrow R_p$$

   by choosing any vertex (p, v) from C; R_p is the rational relation defined by A but with a single initial state set to p.
   Finally, $w^* \subseteq \downarrow R$ if and only if there exists at least one of the sub-problems which is true. Since the depth of the sub-problem generations is bounded by d (d being constant), the number of all sub-problems to consider is polynomial in $|A|$ and $|w|$.

VII. Conclusions

In this paper we present an algorithm which decides whether a one generator submonoid $\{w\}^* \subseteq (\Sigma^*)^d$ is included in the prefix closure of a rational relation $R \subseteq (\Sigma^*)^d$.

So far, we were unable to solve a more general problem when a submonoid is generated by more than one element. It is still not clear for us if the later problem is decidable or not.

References

[1] Olivier Carton, Christian Choffrut, and Serge Grigorieff. Decision problems among the main subfamilies of rational relations. *Theoretical Informatics and Applications*, 40(2):255–275, 2006.
[2] Patrick C. Fischer. Multi-tape and infinite-state automata — a survey. *Commun. ACM*, 8:799–805, December 1965.
[3] Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific J. Math.*, 16(2):285–296, 1966.
[4] Peter Habermehl and Richard Mayr. A note on SLRE. Technical report, LIAFA - Universit Denis Diderot, 2, place Jussieu, Paris Cedex 05, France, 2000. http://homepages.inf.ed.ac.uk/rmayr/slre.ps.gz.
[5] Nils Klarlund. Mona & fido: The logic-automaton connection in practice. In *Selected Papers from the11th International Workshop on Computer Science Logic*, CSL '97, pages 311–326, London, UK, 1998. Springer-Verlag.
[6] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114 –125, april 1959.
[7] Jaques Sakarovitch. *Eléments de théorie des automates*. Vuibert Informatique, 2003.
[8] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time(preliminary report). In *Proceedings of the fifth annual ACM symposium on Theory of computing*, STOC '73, pages 1–9, New York, NY, USA, 1973. ACM.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)