Theoretical Informatics and Applications

Theoret. Informatics Appl. **36** (2002) 359–384 DOI: 10.1051/ita:2003002

ON SHUFFLE IDEALS*

PIERRE-CYRILLE HÉAM¹

Abstract. A shuffle ideal is a language which is a finite union of languages of the form $A^*a_1A^* \cdots A^*a_kA^*$ where A is a finite alphabet and the a_i 's are letters. We show how to represent shuffle ideals by special automata and how to compute these representations. We also give a temporal logic characterization of shuffle ideals and we study its expressive power over infinite words. We characterize the complexity of deciding whether a language is a shuffle ideal and we give a new quadratic algorithm for this problem. Finally we also present a characterization by subwords of the minimal automaton of a shuffle ideal and study the complexity of basic operations on shuffle ideals.

Mathematics Subject Classification. 68Q45, 68Q70.

1. Preliminaries

1.1. INTRODUCTION

The shuffle product is an operation on languages which is strongly connected to combinatorics on words and which was widely studied in the literature [4, 18, 21, 24, 26].

The main topic of this paper is the algorithmic study of shuffle ideals which are rational languages of the form $\cup A^*a_1A^* \cdots A^*a_kA^*$ where A is a finite alphabet and the a_i 's are letters. This is an interesting class of languages which is both connected to combinatorics on words [11,17] and to the algebraic classification of rational languages: indeed it represents the first half level of a hierarchy of starfree languages which was introduced by Straubing [29] and Thérien [30] and which is still intensively studied [2,3,9,10,23,25,27,28,31].

© EDP Sciences 2003

^{*} This work was done while the author was at LIAFA, Université Paris 7.

¹ Laboratoire d'Informatique de Franche-Comté, Université de Franche-Comté, 16 route de Gray, 25030 Besancon Cedex, France; e-mail: heampc@lifc.univ-fcomte.fr

In this paper, we first show that a rational language is a shuffle ideal if and only if it can be represented by a finite automaton whose strongly connected components are complete and reduce to one state. Next we prove that computing such an automaton accepting the shuffle ideal K generated by a language given by a finite automaton can be done in polynomial time, while computing a rational expression $\cup A^* a_1 A^* \cdots A^* a_k A^*$ (where the union is finite) representing K is exponential in the worst case. Moreover we give a quadratic algorithm to test whether a language given by a deterministic automaton is a shuffle ideal (the best algorithm known previously works approximately in time $O(n^6)$ [23]). We also consider the complexity of this problem by proving that it is NLOGSPACE-complete and that it becomes PSPACE-complete when the language is given by a nondeterministic automaton. In Section 5, we present a connection between the minimal automaton of a shuffle ideal and combinatorics on words: we show how to construct this automaton using properties on subwords. We also study the complexity of computing the union, the intersection and the product of two shuffle ideals. Finally, it is known [8,14,15] that star-free languages are exactly the languages which can be represented by a linear temporal logic formula (some fragments of this logic are studied in [6, 32, 33]). In the last section, we give a temporal logic characterization of shuffle ideals and we prove that this restricted temporal logic evaluated over infinite words defines exactly the languages of the form LA^{ω} , where L is a shuffle ideal.

1.2. Some background

For more information on combinatorics on words we refer the reader to [17]. For a general reference on automata theory the reader is referred to [5,7,13], and for basic results on complexity see [19].

For a finite set K, we denote its cardinality by |K|.

Recall that an *alphabet* is a finite set whose elements are called *letters*. A *word* is a finite sequence of letters. A *language* is a set of words and the set of all words over the alphabet A is denoted by A^* . If L is a language of A^* , L^c denotes the complement of L relatively to A^* . Any subsequence v of u is called a *subword* of u and we denote this by $v \prec u$. We denote by |u| the length of a word u, by ε the empty word and by A^+ the language $A^* \setminus \{\varepsilon\}$.

Recall that a *finite automaton* is a 5-tuple $\mathcal{A} = (Q, A, E, I, F)$ where Q is a finite set of states, A is the alphabet, $E \subseteq Q \times A \times Q$ is the set of transitions, $I \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states. With the above notation, if $p \in Q$ and $u \in A^*$, $p \cdot u \subseteq Q$ denotes the set of states q of \mathcal{A} such that there exists a path in \mathcal{A} from p to q labeled by u. A finite automaton is said to be *complete* if for every state q and every letter $a, q \cdot a \neq \emptyset$. If \mathcal{A} is a finite automaton, then $L(\mathcal{A})$ denotes the language accepted by \mathcal{A} . In this paper, minimal automata are deterministic but not necessary complete and all considered automata are finite. In general, in this paper, automata are nondeterministic.

If L is a language and u a word, then $u^{-1}L$ denotes the language $\{v \in A^* \mid uv \in L\}$ and is called a quotient of L. It is known that L is rational if and only if L has

a finite number of quotients. Furthermore, in this case, the minimal automaton of L has as many states as quotients of L.

A rational expression is a finite expression using the empty set, the letters and the symbols union, product and star. We recursively define the size of a rational expression E, denoted $\tau(E)$ by:

•
$$\tau(\emptyset) = 1$$

- $\tau(v) = 1$, $\tau(E) = 1$ if E = a with $a \in A$; $\tau(EF) = \tau(E) + \tau(F) + 1$;

•
$$\tau(E^*) = \tau(E) + 1;$$

• $\tau(E \cup F) = \tau(E) + \tau(F) + 1.$

The size of a rational expression represents the number of symbols occurring in it. A language which can be represented by a rational expression is *rational*. Kleene's theorem [16] states that a language is rational if and only if it is recognizable by a finite automaton. In this paper, we shall make no distinction between a rational expression and the language it represents. Moreover we need the following result [12]:

Theorem 1.1. Let E be a rational expression of size n. One can compute in time $O(n \log^2(n))$ a corresponding automaton with O(n) states and $O(n \log^2(n))$ transitions.

Recall that the *shuffle* of two words u and v is the language of all words w such that $w = u_1 v_1 u_2 v_2 \cdots u_n v_n$ with $u_i, v_i \in A^*$, $u_1 u_2 \cdots u_n = u$ and $v_1 v_2 \cdots v_n = v$. The shuffle of u and v is denoted $u \sqcup v$. This notion can be extended to languages as follows:

$$L \sqcup K = \bigcup_{u \in L, v \in K} u \sqcup v.$$

A language L is a *shuffle ideal* if $L \sqcup A^* = L$. For any language L, $L \sqcup A^*$ is the smallest (for the inclusion) shuffle ideal containing L; it is called the shuffle ideal generated by L. We need the following important result due to Higman [11] (see also [17], Chap. 6).

Theorem 1.2. A language L is a shuffle ideal if and only if there exists a finite set K such that $L = K \sqcup A^*$.

In particular, since the shuffle operator preserves rational languages, every shuffle ideal is a rational language. Moreover, it is known [23] that the class of shuffle ideals is a positive variety of languages. In particular, we have the following proposition [23]:

Proposition 1.3. The union and the intersection of two shuffle ideals is a shuffle ideal and left and right quotients of a shuffle ideal are shuffle ideals.

Finally we need the following result which leads to an algorithm to test whether a rational language is a shuffle ideal [23].

Proposition 1.4. Let $\mathcal{A} = (Q, A, E, i, F)$ be a deterministic complete automaton. The language $L(\mathcal{A})$ is a shuffle ideal if and only if there are no states p and q and words u, v, w such that $i \cdot w = p, p \cdot u = q, p \cdot v \in F$ and $q \cdot v \notin F$.

We denote by A^{ω} the set of infinite words over A and a set of infinite words is called ω -language. If $\mathcal{A} = (Q, A, E, I, F)$ is an automaton, we denote by $L_{\omega}(\mathcal{A})$ the set of infinite words which are the label of an infinite path of \mathcal{A} visiting infinitly often a state of F. In this case \mathcal{A} is said to be a *Büchi automaton*.

2. Representing shuffle ideals

An automaton $\mathcal{A} = (Q, A, E, I, F)$ is called *saturated* if for all states $p \in Q$ and all letters $a \in A$, $(p, a, p) \in E$. An automaton is said to be *locally trivial* if each of its strongly connected components is reduced to one state. A rational expression is called *polynomial* if it is a finite union of expressions of the form $A^*A_1A^* \cdots A^*A_kA^*$ where $A_i \subseteq A$.

Theorem 2.1. Let L be a language of A^* . The following assertions are equivalent:

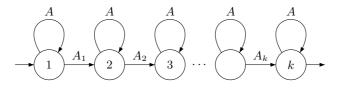
- (1) L is a shuffle ideal;
- (2) L can be described by a polynomial expression;
- (3) L is recognizable by a saturated automaton;
- (4) L is recognizable by an automaton that is saturated and locally trivial.

Proof. Let K be a finite set. Since

$$K$$
ш $A^* = \bigcup_{a_0a_1\cdots a_k \in K} A^*a_0A^*a_1A^*\cdots A^*a_kA^*$

 $(1) \Rightarrow (2)$ is a direct consequence of Theorem 1.2.

Each language of the form $A^*A_1A^*\cdots A^*A_kA^*$ is accepted by the automaton



which is saturated and locally trivial. Since a disjoint union of saturated, locally trivial automata is a saturated, locally trivial automaton, we have $(2) \Rightarrow (4)$.

The condition (4) is stronger than the condition (3), thus we have $(4) \Rightarrow (3)$.

Now, let $\mathcal{A} = (Q, A, E, I, F)$ be a saturated automaton and let $w \in L(\mathcal{A}) \sqcup \mathcal{A}^*$. By definition of the shuffle product, we have $w = u_1 v_1 \cdots u_k v_k$ where $u_1 \cdots u_k = u \in L(\mathcal{A})$ and $v_i \in \mathcal{A}^*$. Thus, considering an accepting path in \mathcal{A} labeled by u, each time we finish to read u_i , we can use transitions of the form (p, a, p) to

read v_i . So we construct an accepting path in \mathcal{A} labeled by w. It follows that $L(\mathcal{A}) \sqcup \mathcal{A}^* \subseteq L(\mathcal{A})$. Consequently $L(\mathcal{A})$ is a shuffle ideal and $(3) \Rightarrow (1)$.

3. Computing generated shuffle ideals

In this section we study the following problem: given a rational language, we want to compute one of the representations of Theorem 2.1 of its generated shuffle ideal. We first study how to compute a saturated automaton accepting the shuffle ideal of a given rational language. Secondly we study the same question for the computation of a locally trivial, saturated automaton. Finally we study the complexity of computing a polynomial expression representing the shuffle ideal of a given rational language.

3.1. Computing saturated automata

We need the following algorithm called <u>Saturate</u> which applies to an automaton $\mathcal{A} = (Q, A, E, I, F)$.

Algorithm 3.1. Saturate

1. E' = E.

- 2. For all $p \in Q$ and for all letter $a \in A$, $E' = E' \cup \{(p, a, p)\}.$
- 3. Return $\mathcal{A}' = (Q, A, E', I, F)$.

Graphically, during the algorithm, we just add on each state of the automaton a loop labeled by A.

Proposition 3.2. If \mathcal{A} is an automaton, then $\mathcal{A}' = \underline{\operatorname{Saturate}}(\mathcal{A})$ is a saturated automaton which recognizes $L(\mathcal{A}) \sqcup \mathcal{A}^*$.

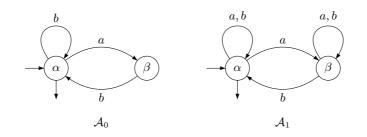
Proof. The automaton \mathcal{A}' is saturated because we precisely add the missing transitions to make it saturated.

By Theorem 2.1, $L(\mathcal{A}')$ is a shuffle ideal. Moreover, by construction, $L(\mathcal{A}) \subseteq L(\mathcal{A}')$. It follows that $L(\mathcal{A}) \sqcup \mathcal{A}^* \subseteq L(\mathcal{A}')$. Conversely, let w be in $L(\mathcal{A}')$: there exists a successful path in \mathcal{A}'

$$(p_0, a_1, p_1), (p_1, a_2, p_2), \cdots, (p_{k-1}, a_k, p_k)$$

labeled by w. If we delete in this path all the transitions of the form (p, a, p) we obtain a new path which is both successful in \mathcal{A}' and in \mathcal{A} . Let u be the label of this new path. Since $u \in L(\mathcal{A})$ and since $w \in \{u\} \sqcup \mathcal{A}^*$, $w \in L(\mathcal{A}) \sqcup \mathcal{A}^*$. It follows that $L(\mathcal{A}') = L(\mathcal{A}) \sqcup \mathcal{A}^*$.

For example the shuffle ideal generated by the language $(b^*(ab)^*)^*$ whose minimal automaton is \mathcal{A}_0 (see below), is accepted by the saturated automaton \mathcal{A}_1 .



Proposition 3.3. Let L be a rational language given by an n-state, m-transition automaton. Then one can compute in time O(m + n|A|) an n-state, saturated automaton with at most m + n|A| transitions, accepting the shuffle ideal generated by L.

Proof. By Proposition 3.2, it is sufficient to apply the algorithm <u>Saturate</u>. \Box

As a direct consequence, using Theorem 1.1, one has the following corollary:

Corollary 3.4. Let L be a rational language given by a rational expression of size k. One can compute in $O(k \log^2(k))$ time an O(k)-state saturated automaton accepting the shuffle ideal generated by L.

3.2. Computing locally trivial, saturated automata

Lemma 3.5. Let L be a shuffle ideal accepted by a saturated n-state automaton and K be the set of words of L of length less than or equal to n. Then

$$L = K \sqcup A^*.$$

Proof. Since $K \subseteq L$, $K \sqcup A^* \subseteq L \sqcup A^* = L$. Now, let w be in L, \mathcal{A} be a saturated n-state automaton accepting L and m be a successful path in \mathcal{A} labeled by w. We consider now a successful path in \mathcal{A} obtained from m by deleting the loops. We denote by u the label of this new path. By construction $u \in K$. Furthermore, $w \in \{u\} \sqcup \mathcal{A}^*$. Consequently, $w \in K \sqcup \mathcal{A}^*$, proving the lemma. \Box

The following algorithm has as input an *n*-state finite automaton $\mathcal{A} = (Q, A, E, I, F)$ and computes an automaton accepting precisely the shuffle ideal generated by $L(\mathcal{A})$.

Algorithm 3.6. LTSaturate

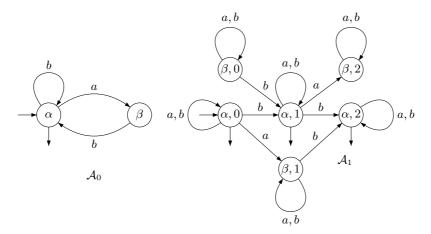
1. $Q' = Q \times \{0, \dots, n\}$ 2. $E' = \{((p, i - 1), a, (q, i)) \mid (p, a, q) \in E, i \in \{1, \dots, n\}\}$ 3. $I' = I \times \{0\}$ 4. $F' = F \times \{0, \dots, n\}$ 5. $\mathcal{A}' = (Q', A, E', I', F')$ 6. Return Saturate(\mathcal{A}').

Proposition 3.7. Let \mathcal{A} be an n-state automaton. Then <u>LTSaturate</u>(\mathcal{A}) is a saturated, locally trivial, $O(n^2)$ -state automaton with $O(mn + |\mathcal{A}|n^2)$ transitions accepting the shuffle ideal generated by $L(\mathcal{A})$.

Proof. The automaton \mathcal{A}' is the product of \mathcal{A} by the minimal automaton accepting the set of words of length less than or equal to n. It follows that \mathcal{A}' accepts the set of words of $L(\mathcal{A})$ of length less or equal to n. Thus, by Lemma 3.5 and Proposition 3.2, the automaton <u>LTSaturate</u>(\mathcal{A}) recognizes the shuffle ideal generated by $L(\mathcal{A})$.

Moreover, by construction, the automaton $\underline{\text{LTSaturate}}(\mathcal{A})$ is saturated and has n(n+1) states. Now, we claim that \mathcal{A}' is locally trivial. Indeed, assume that we have a non trivial loop in \mathcal{A}' , *i.e.* there exist two distinct states (p, i) and (q, j) in Q' and words u and v such that $(p, i) \in (q, j) \cdot v$ and $(q, j) \in (p, i) \cdot u$. By definition of E', it means that i < j and j < i, a contradiction. Since the algorithm <u>Saturate</u> does not change the strongly connected components, it follows that <u>LTSaturate}(\mathcal{A}) is locally trivial.</u>

For example the shuffle ideal generated by the language $(b^*(ab)^*)^*$ whose minimal automaton is \mathcal{A}_0 (see below) and which is accepted by the saturated and locally trivial automaton $\mathcal{A}_1 = \underline{\text{LTSaturate}}(\mathcal{A}_0)$.



Proposition 3.8. Let L be a rational language given by an n-state, m-transition automaton. Then one can compute in time $O(|A|n^2 + mn)$ a saturated, locally trivial automaton accepting the shuffle ideal generated by L.

Proof. This is a direct consequence of Proposition 3.7.

As a consequence of the above proposition and Theorem 1.1, one has the following corollary:

Corollary 3.9. Let L be a rational language given by a rational expression of size k. Then one can compute in time $O(k^2 \log^2 k)$ an $O(k^2)$ -state saturated, locally trivial automaton accepting the shuffle ideal generated by L.

3.3. Computing polynomial expressions

Proposition 3.10. Let \mathcal{A} be an n-state automaton. Then one can compute in time $O((|\mathcal{A}| + 1)^n)$ a polynomial expression of size $O((|\mathcal{A}| + 1)^n)$ accepting the shuffle ideal generated by $L(\mathcal{A})$.

Proof. By Lemma 3.5, one has

$$L(\mathcal{A}) \bigsqcup A^* = \bigcup_{\substack{a_1 \cdots a_k \in L(\mathcal{A}) \\ a_i \in A, \ k \le n}} A^* a_1 A^* a_2 \cdots A^* a_k A^*.$$
(1)

Since $\tau(A) = \tau(\bigcup_{a \in A} a) = 2|A| - 1$, we have $\tau(A^*a_1A^*a_2\cdots A^*a_kA^*) = (k + 1)\tau(A^*) + k + 2k = 2|A|(k+1) + 3k$. It follows that the size of the expression (1) is at most $|A|^{n+1}[2|A|(n+1) + 3n] = O((|A| + 1)^n)$.

Now we shall prove that computing a polynomial rational expression representing a given language is exponential in the worst case. Let L be a rational language. We denote by m(L) the number of states of the minimal automaton of L and by M(L) the minimal number of states of an automaton accepting L. Obviously $m(L) \ge M(L)$. We also denote by e(L) the minimal size of a rational expression representing L.

We introduce the following invariants (with the convention $\inf \emptyset = 0$):

$$R(n) = \max_{L:m(L) \le n} \left[\inf_{E:E=L} \{ \tau(E) \mid E \text{ polynomial} \} \right]$$
$$S(n) = \max_{L:M(L) \le n} \left[\inf_{E:E=L} \{ \tau(E) \mid E \text{ polynomial} \} \right]$$
$$T(n) = \max_{L:e(L) \le n} \left[\inf_{E:E=L} \{ \tau(E) \mid E \text{ polynomial} \} \right].$$

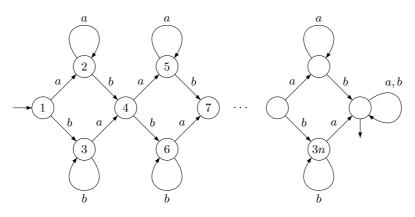
Theorem 3.11.

$$\Omega(2^{n/3}) = R(n) \le S(n) = O((|A| + 1)^n).$$

Proof. Since $m(L) \ge M(L)$, we have $R(n) \le S(n)$. Moreover, by Proposition 3.10, one has $S(n) = O((|A| + 1)^n)$.

To prove the lower bound, we consider the alphabet $A = \{a, b\}$ and the language $L = A^* a A^* b A^* \cup A^* b A^* a A^*$ which is a shuffle ideal. Since the class of shuffle ideals

is closed under product, for all n, L^n is a shuffle ideal. Moreover, the minimal automaton of L^n has the following form:



It follows that $m(L^n) = 3n + 1$. Now let E be a polynomial expression accepting L^n . Assume that $A^*A_1A^* \cdots A_kA^*$ is an expression appearing in E. Then the word $u = a_1a_2\cdots a_k$, with $a_i \in A_i$ belongs to L^n . Since each word of L^n has length 2n, we have that $k \geq 2n$.

Let $K = \{ab, ba\}$. Each word u in K^n belongs to L^n . Consequently there exists an expression

$$E_0 = A^* A_1 A^* \cdots A_k A^*$$

appearing in E such that $u \in E_0$. But $k \ge 2n$, hence k = 2n. It follows that $u \in A_1 A_2 \cdots A_{2n}$.

Now we claim that if $A_1 \cdots A_{2n} \subseteq L^n$ (with $A_i \neq \emptyset$), then for all i, $|A_i| = 1$. Indeed assume there exists an i such that $A_i = \{a, b\}$. In this case, there exist x and y in A^* such that |x| = i - 1, |x| + |y| = 2n - 1 and $xay, xby \in K^n$. Thus, when we read xay and xby in the minimal automaton of L^n , we do not use any loop. Thus |x| must be odd (otherwise after reading xa we must read b, and after reading xb we must read a: the first letter of y should be both a and b). Starting from the final state, the same argument leads to conclude that |y| is odd, which is in contradiction with |x| + |y| = 2n - 1.

It follows that each expression of the form

$$A^*a_1A^*a_2A^*\cdots a_{2n}A^*$$

with $a_1 \cdots a_{2n} \in K^n$ appears in E. But

$$\tau(A^*a_1A^*a_2A^*\cdots a_{2n}A^*) = 2|A|(2n+1) + 6n = 14n + 4$$

and there are $|K^n| = 2^n$ expressions of this kind in E. Thus

$$\tau(E) \ge 14n2^n.$$

Since R(n) is an increasing function and since $m(L^n) = 3n + 1$, one can conclude that

$$R(n) = \Omega(2^{n/3}).$$

Proposition 3.12. There exists a constant $\alpha \in \mathbb{N}$ such that

$$\Omega(2^{n/15}) = T(n) \le S(\alpha n) = O((|A| + 1)^{\alpha n}).$$

Proof. By Theorem 1.1, there exists a constant $\alpha \in \mathbb{N}$ such that for all rational language L, $M(L) \leq \alpha e(L)$. Consequently $T(n) \leq S(\alpha n)$. The equality $S(\alpha n) = O((|A| + 1)^{\alpha n})$ has been already proved.

To prove the lower bound we use the languages L^n defined in the proof of Theorem 3.11. Since

$$L^n = (aa^*b \cup bb^*a)^n \{a, b\}^*$$

and since

$$\tau(aa^*b \cup bb^*a) = 6 + 1 + 6 = 13,$$

we have, for all $n \ge 4$,

$$e(L^n) \le 14n + 4 \le 15n.$$

Furthermore, we have seen that

$$\inf_{E:E=L} \{\tau(E) \mid E \text{ polynomial}\} \ge 14n2^n.$$

It follows that

$$T(n) \ge \frac{14}{15}n2^{n/15}.$$

Consequently, $T(n) = \Omega(2^{n/15})$.

4. Testing whether a language is a shuffle ideal

In this section we consider the complexity of testing whether a rational language is a shuffle ideal.

4.1. Deterministic automata

In [23], an algorithm based on Proposition 1.4 is proposed to test whether a language given by a deterministic *n*-state automaton is a shuffle ideal. This algorithm requires to compute the transitive closure of a graph with $|A|^2 n^2$ vertices

and its time complexity is about $O(|A|^6n^6)$. We propose here an $O(|A|^2n^2)$ algorithm. We call our algorithm <u>TestShuffle</u>. Its input is a deterministic complete automaton $\mathcal{A} = (Q, A, E, i, F)$.

Algorithm 4.1. <u>TestShuffle</u>

1. $\mathcal{A}_0 = (Q_0, A, E_0, i_0, F_0) = \underline{Saturate}(\mathcal{A}).$ 2. $Q_1 = Q_0 \times Q.$ 3. $E_1 = \{((p_0, p), a, (q_0, q)) \mid (p, a, q) \in E, (p_0, a, q_0) \in E_0\}.$ 4. $i_1 = (i_0, i).$ 5. $F_1 = F_0 \times (Q \setminus F).$ 6. $\mathcal{A}_1 = (Q_1, A, E_1, i_1, F_1).$

7. If $L(A_1) = \emptyset$ return true, else return false.

Let us note that the automaton \mathcal{A}_0 has a unique initial state: indeed the algorithm <u>Saturate</u> does not change the initial states and \mathcal{A} is deterministic and has hence a unique initial state.

Proposition 4.2. Let \mathcal{A} be a deterministic complete n-state automaton. The algorithm <u>TestShuffle</u>(\mathcal{A}) returns true if and only if $L(\mathcal{A})$ is a shuffle ideal. Moreover the answer is given in time $O(|\mathcal{A}|^2n^2)$.

Proof. By Proposition 3.2, $L(\mathcal{A}_0) = L(\mathcal{A}) \sqcup \mathcal{A}^*$. Moreover, by construction, $L(\mathcal{A}_1) = L(\mathcal{A})^c \cap L(\mathcal{A}_0)$. Consequently the algorithm returns **true** if and only if $L(\mathcal{A}_0) \subseteq L(\mathcal{A})$, if and only if $L(\mathcal{A}_0) = L(\mathcal{A})$, if and only if $L(\mathcal{A})$ is a shuffle ideal.

Assume that \mathcal{A} has m transitions. The first step of the algorithm can be done in time O(n|A| + m) by Proposition 3.3 and the automaton \mathcal{A}_0 has n states and O(m + n|A|) transitions. It follows that computing \mathcal{A}_1 can be done in time $O(n^2 + m^2 + mn|A|)$ and \mathcal{A}_1 has n^2 states and $O(m^2 + mn|A|)$ transitions. Now one can test whether $L(\mathcal{A}_1)$ is empty by testing whether a final state of \mathcal{A}_1 is accessible from i_1 . It can be done by computing a spanning tree of \mathcal{A}_1 rooted on i_1 (for example by a depth-first search from i_1) and by testing whether this tree contains a final state. Hence, testing whether $L(\mathcal{A}_1)$ is empty can be done in time $O(n^2 + m^2 + mn|A|)$. Since \mathcal{A} is deterministic and complete, m = |A|n. Consequently the algorithm works in time $O(n^2 + 2n^2|A|^2) = O(n^2|A|^2)$.

Lemma 4.3. Let $\mathcal{A} = (Q, A, E, i, Q \setminus \{q\})$ be a complete deterministic automaton such that there is only one state $q \neq i$ which is not final. There is no path in \mathcal{A} from i to q if and only if $L(\mathcal{A})$ is a shuffle ideal.

Proof. According to Proposition 1.4, if $L(\mathcal{A})$ is a shuffle ideal then there is no path from i to q.

Conversely, if there is no path from i to q, $L(\mathcal{A}) = A^*$ because \mathcal{A} is complete and all accessible states are final.

Theorem 4.4. Testing whether a rational language over A given by a deterministic automaton is a shuffle ideal is a NLOGSPACE-complete problem if $|A| \ge 2$.

Proof. Let \mathcal{A} be a deterministic automaton whose initial state is *i*. By Proposition 1.4, $L(\mathcal{A})$ is not a shuffle ideal if and only if there exist three words w, u

and v and two states p and q such that $i \cdot w = p$, $p \cdot u = q$, $p \cdot v$ is final and $q \cdot v$ is not final. This can be tested in NLOGSPACE by guessing p and q and successively the letters of u and v. Since NLOGSPACE=co-NLOGSPACE, testing whether a rational language given by a deterministic automaton is a shuffle ideal is an NLOGSPACE problem.

Now, let G = (V, E) be a finite directed graph such that there are no more than two edges starting from each vertex. Let p and q be two distinct vertices of G. It is known that testing whether there is a path from p to q is an NLOGSPACEcomplete problem. Now consider a deterministic automaton $\mathcal{A}_0 = (V, A, E_0, p, V \setminus \{q\})$ such that $(r, a, s) \in E_0$ if and only if $(r, s) \in E$: we put labels on the edges of G to make it a deterministic automaton (there are several ways to do it). If \mathcal{A}_0 is not complete we make it complete by adding a state $t \notin V$: we consider the automaton

$$\mathcal{A}_1 = (V \cup \{t\}, A, E_0 \cup E_1, p, V \cup \{t\} \setminus \{q\})$$

where $E_1 = \{(s, a, t) \mid s \in V \cup \{t\}, a \in A \text{ and } (\{s\} \times \{a\} \times V) \cap E_0 = \emptyset\}$. If \mathcal{A}_0 is complete we define $\mathcal{A}_1 = \mathcal{A}_0$. Now, by Lemma 4.3, there is no path from p to q in G if and only if $L(\mathcal{A}_1)$ is a shuffle ideal. Thus, testing whether a rational language given by a deterministic automaton is a shuffle ideal is a NLOGSPACE-hard problem.

4.2. Nondeterministic automata

It is known that testing whether a rational language over A given by a finite nondeterministic automaton accepts A^* is a PSPACE-complete problem [1]. Since it is NLOGSPACE decidable whether the empty word belongs to a rational language given by a finite automaton (just guess a state which is both initial and final), testing whether a rational language containing the empty word and given by a finite automaton accepts every word is a PSPACE-complete problem.

Theorem 4.5. If $|A| \ge 2$, then testing whether a rational language given by a finite automaton is a shuffle ideal is a PSPACE-complete problem.

Proof. Testing the equality of two rational languages given by finite automata is a PSPACE problem. Thus, using Proposition 3.3, testing whether a rational language given by a finite automaton is a shuffle ideal is a PSPACE problem.

Let \mathcal{A} be a finite automaton containing the empty word. Then $L(\mathcal{A})$ is a shuffle ideal if and only if $L(\mathcal{A}) = A^*$. Thus, testing whether a rational language given by a finite automaton is a shuffle ideal is PSPACE-hard.

5. On the minimal automaton of a shuffle ideal

In this section we first study properties of the minimal automaton of a shuffle ideal. Next, since the class of shuffle ideals is closed under union, intersection and product we study the complexity of computing the union, the intersection and the product of two shuffle ideals given by their minimal automata (for a survey on these problems for finite languages or for rational languages see [34]).

5.1. Properties of the minimal automaton of a shuffle ideal

The following proposition is proved in [28].

Proposition 5.1. The minimal automaton of a shuffle ideal is complete and locally trivial.

Moreover, the following lemma will be useful later:

Lemma 5.2. The minimal automaton of a shuffle ideal has a unique final state.

Proof. Let $\mathcal{A} = (Q, A, E, i, F)$ be the minimal automaton of a shuffle ideal L and f and g be two final states. There exist u and v such that $i \cdot u = f$ and $i \cdot v = g$. Since L is a shuffle ideal, for all $w \in A^*$, uw and vw are both in L. It follows that f = g. Moreover, we proved that $f \cdot A^* = f$.

The minimal automaton of the shuffle ideal generated by a language can be much smaller than the minimal automaton of that language. For instance, $m(A^n \cup$ $\{\varepsilon\}$ = n + 1 and $m((A^n \cup \{\varepsilon\}) \sqcup A^*) = 1$ because $(A^n \cup \{\varepsilon\}) \sqcup A^* = A^*$.

Proposition 5.3. Let L be a language. There exists a unique finite language, denoted $S_{\min}(L)$, such that

- S_{min}(L) ⊥⊥ A* = L ⊥⊥ A*;
 If u, v ∈ S_{min}(L) and if u is a subword of v, then u = v.

Proof. By Theorem 1.2, there exists a finite language K such that $K \sqcup A^* =$ $L \sqcup A^*$. Assume there exist u and v in K such that u is a subword of v and $u \neq v$. Then $v \sqcup A^* \subseteq u \sqcup A^*$. It follows that $K \setminus \{v\} \sqcup A^* = L \sqcup A^*$. In this way, one can build a finite language S satisfying the hypotheses of the proposition. Assume now that there are two such finite languages S_1 and S_2 . Let $u \in S_1$. Since $S_1 \subseteq L \sqcup A^* = S_2 \sqcup A^*$, there exists $v \in S_2$ such that $v \prec u$. By a symmetric argument, there exists $w \in S_1$ such that $w \prec v$. It follows that $w \prec v \prec u$. Thus, by hypotheses on S_1 , we have u = w. Hence u = v, proving that $S_1 \subseteq S_2$. In the same way, we prove that $S_2 \subseteq S_1$. It follows that $S_1 = S_2$.

Let us note that in the above proposition we do not require L to be a shuffle ideal. Moreover, for each language L, $S_{\min}(L) = S_{\min}(L \sqcup A^*)$.

Proposition 5.4. Let L be a non empty language and $\ell = \min\{|u| \mid u \in L\}$. Set $K_0 = L \cap A^{\ell}$ and, by induction, for $i \geq 1$,

$$K_i = K_{i-1} \cup ((L \cap A^{\ell+i}) \cap (K_{i-1} \sqcup A^*)^c).$$

It holds that

$$S_{\min}(L) = \bigcup_{i \in \mathbb{N}} K_i.$$

Proof. Let $K = \bigcup_{i \in \mathbb{N}} K_i$. By a direct induction, for all $i \in \mathbb{N}$, $K_i \subseteq L$. Thus $K \subseteq L$ and, therefore, $K \sqcup A^* \subseteq L \sqcup A^*$. Conversely, assume that $u \in L \sqcup A^*$. There exists $v \in L$ such that $v \prec u$. If $v \in K_{|v|-\ell}$, then $u \in K_{|v|-\ell} \sqcup A^* \subseteq K \sqcup A^*$. If $v \notin K_{|v|-\ell}$, then there exists $w \in K_{|v|-\ell-1}$ such that $w \prec v$. In this case $w \prec u$ and $u \in K_{|v|-\ell-1} \sqcup A^* \subseteq K \sqcup A^*$. It follows that

$$L \sqcup L A^* = K \sqcup L A^*.$$

Now assume that u and v are two distinct words of K such that $u \prec v$. Thus |u| < |v|. Since all words of K of length $\ell + i$ are in K_i , it follows that $u \in K_{|u|-\ell} \subseteq K_{|v|-\ell-1}$ and that $v \in K_{|v|-\ell}$. Moreover all words of K_i have length less than or equal to $\ell + i$, thus $v \in K_{|v|-\ell} \setminus K_{|v|-\ell-1}$. By definition of $K_{|v|-\ell}$, $v \notin K_{|v|-\ell-1} \sqcup A^*$. It means that u is not a subword of v, a contradiction. Therefore, by Theorem 1.2, K is finite, which concludes the proof.

For each language L we define the mapping ϕ_L from A^* into the set of finite languages by letting $\phi_L(u) = S_{\min}(u^{-1}(L \sqcup A^*))$.

Let $Q_L = \{\phi_L(u) \mid u \in A^*\}$ and $E_L = \{(\phi_L(u), a, \phi_L(ua)) \mid a \in A, u \in A^*\}.$

Proposition 5.5. Let *L* be a non empty language. The set Q_L is finite and the automaton $\mathcal{A}_L = (Q_L, A, E_L, S_{\min}(L), \{\varepsilon\})$ is the minimal automaton of $L \sqcup A^*$.

Proof. Since there are finitely many $u^{-1}(L \sqcup A^*)$, the set Q_L is finite. Moreover $|Q_L| \leq |\{u^{-1}(L \sqcup A^*) \mid u \in A^*\}| = m(L \sqcup A^*)$. Furthermore, by construction, \mathcal{A}_L is deterministic.

Now we claim that \mathcal{A}_L accepts $L \sqcup A^*$. Indeed assume that $u \in L(\mathcal{A}_L)$. Then, by definition of \mathcal{A}_L , $S_{\min}(u^{-1}(L \sqcup A^*)) = \{\varepsilon\}$. It follows that $\varepsilon \in u^{-1}(L \sqcup A^*)$, proving that $u \in L \sqcup A^*$. Conversely, if $u \in L \sqcup A^*$, then $\varepsilon \in u^{-1}(L \sqcup A^*)$. But the only shuffle ideal containing ε is A^* and, by Proposition 1.3, $u^{-1}(L \sqcup A^*)$ is a shuffle ideal. It follows that $u^{-1}(L \sqcup A^*) = A^*$. Therefore $S_{\min}(u^{-1}(L \sqcup A^*)) = \{\varepsilon\}$, proving the claim.

The automaton \mathcal{A}_L is deterministic, accepts $L \sqcup A^*$ and has at most $m(L \sqcup A^*)$ states. Thus it is the minimal automaton of $L \sqcup A^*$.

Now we explain how to recursively compute $\phi_L(u)$. We need three lemma

Lemma 5.6. Let L be a shuffle ideal, $a \in A$ and $u \in A^*$. If

$$u \in a^{-1}S_{\min}(L) \cup \bigcup_{b \neq a} b(b^{-1}S_{\min}(L)),$$

then $u \in a^{-1}L$.

Proof. If $u \in a^{-1}S_{\min}(L)$, then $au \in S_{\min}(L) \subseteq L$. It follows that $u \in a^{-1}L$. If $u \in b(b^{-1}S_{\min}(L))$, then u = bw, with $b \neq a$ and $w \in b^{-1}S_{\min}(L)$. Consequently $u \in S_{\min}(L) \subseteq L$. Since L is a shuffle ideal, $au \in L$. Thus $u \in a^{-1}L$.

Lemma 5.7. Let L be a shuffle ideal, $a \in A$ and $u \in A^*$. If $L \neq A^*$ and if $u \in a^{-1}L$, then there exists

$$z \in a^{-1}S_{\min}(L) \cup \bigcup_{b \neq a} b(b^{-1}S_{\min}(L))$$

such that $z \prec u$.

Proof. Since $au \in L$ and since L is a shuffle ideal, there exists $v \in S_{\min}(L)$ such that $v \prec au$. Two cases arise:

- $v \in aA^*$: there exists $z \in A^*$ such that v = az. Since $az \prec au$, one has $z \prec u$. Moreover, since $v \in S_{\min}(L)$, one has $z \in a^{-1}S_{\min}(L)$;
- $v \notin aA^*$: if $v = \varepsilon$, then $L = A^*$. Thus, by hypothesis $v \neq \varepsilon$. Therefore there exists $w \in A^*$ such that v = bw with $b \neq a$. Since bw is a subword of au, bw also is a subword of u. Consequently $z = bw \prec u$ and $z \in b(b^{-1}S_{\min}(L))$.

Lemma 5.8. Let L be a shuffle ideal distinct of A^* , $a \in A$ and

$$K = \left(a^{-1}S_{\min}(L) \cup \bigcup_{b \neq a} b(b^{-1}S_{\min}(L))\right).$$

We have $a^{-1}L = K \coprod A^*$.

Proof. If $u \in a^{-1}L$, then, by Lemma 5.7, there exists $z \in K$ such that $z \prec u$. Thus $u \in K \sqcup A^*$.

If $u \in K \sqcup A^*$, then there exists $v \in K$ such that $v \prec u$. By Lemma 5.6, $v \in a^{-1}L$. Since $a^{-1}L$ is a shuffle ideal, we have $u \in a^{-1}L$, proving the lemma. \Box

The next proposition provides a recursive and constructive method to compute $\phi_L(u)$ when $S_{\min}(L)$ is known:

Proposition 5.9. Let L be a language. We have $\phi_L(\varepsilon) = S_{\min}(L)$ and, for all $u \in A^*$ and $a \in A$, if $\phi_L(u) \neq \{\varepsilon\}$, then

$$\phi_L(ua) = S_{\min}\left(a^{-1}\phi_L(u) \cup \bigcup_{b \in A \setminus \{a\}} b(b^{-1}\phi_L(u))\right).$$

Proof. According to the definition of ϕ_L , $\phi_L(\varepsilon) = S_{\min}(L \sqcup A^*)$. But, by definition of S_{\min} , $S_{\min}(L \sqcup A^*) = S_{\min}(L)$. Thus $\phi_L(\varepsilon) = S_{\min}(L)$.

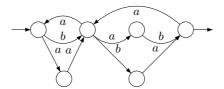
Assume that $\phi_L(u) \neq \{\varepsilon\}$. Then $u^{-1}(L \sqcup A^*) \neq A^*$. Moreover, by Proposition 1.3, $u^{-1}(L \sqcup A^*)$ is a shuffle ideal. Thus, by Lemma 5.8, we have

$$a^{-1}u^{-1}(L \sqcup A^*) = \left(a^{-1}S_{\min}(u^{-1}(L \sqcup A^*)) \cup \bigcup_{b \neq a} b(b^{-1}S_{\min}(u^{-1}(L \sqcup A^*))) \right) \sqcup A^*.$$

By definition of ϕ_L , it follows that

$$\phi_L(ua) = S_{\min}\left(a^{-1}\phi_L(u) \cup \bigcup_{b \in A \setminus \{a\}} b(b^{-1}\phi_L(u))\right).$$

Moreover, if $\phi_L(u) = \{\varepsilon\}$, then $u^{-1}(L \sqcup A^*) = A^*$. Consequently, for every $a \in A$, $\phi_L(ua) = S_{\min}(a^{-1}u^{-1}(L \sqcup A^*)) = S_{\min}(a^{-1}A^*) = S_{\min}(A^*) = \{\varepsilon\}$. For example, consider the language L given by the following automaton:



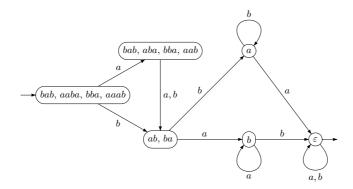
By Lemma 3.5, the shuffle ideal generated by L is also generated by the words of $aaab, babab, babba, bbaaba, bbaaab, bababa, babaab, aaabab, aaabba, baaaab, baaaba\}).$ Now, according to Proposition 5.4, one has $S_{\min}(L) = \{bab, bba, aaba, aaab\}$.

By Proposition 5.9, we have

- $\phi_L(\varepsilon) = \{bab, bba, aaba, aaab\};$
- $\phi_L(a) = S_{\min}(\{aba, aab, bab, bba\}) = \{aba, aab, bab, bba\};$
- $\phi_L(b) = S_{\min}(\{ab, ba, aaba, aaab\}) = \{ab, ba\};$
- $\phi_L(aa) = S_{\min}(\{ab, ba, bab, bba\}) = \{ab, ba\};$
- $\phi_L(ab) = S_{\min}(\{ba, ab, aba, aab\}) = \{ab, ba\};$
- $\phi_L(ba) = \phi_L(aaa) = \phi_L(aba) = S_{\min}(\{b, ba\}) = \{b\};$

- $\begin{aligned} &-\phi_L(ba) = \phi_L(aba) = \phi_L(aba) = S_{\min}(\{0, ba\}) = \{b\}, \\ &-\phi_L(bb) = \phi_L(aab) = \phi_L(abb) = S_{\min}(\{a, ab\}) = \{a\}; \\ &-\text{ For all } k > 0; \ \phi_L(baa^k) = \phi_L(a^{k+3}) = \phi_L(abaa^k) = S_{\min}(\{b\}) = \{b\}; \\ &-\text{ For all } k > 0; \ \phi_L(bbb^k) = \phi_L(aabb^k) = \phi_L(abbb^k) = S_{\min}(\{a\}) = \{a\}; \\ &-\text{ For all } k > 0 \text{ and for all } u \in A^*; \ \phi_L(baa^kbu) = \phi_L(a^{k+3}bu) = \phi_L(abaa^kbu) = \phi_L(abaa^k$ $S_{\min}(\{\varepsilon\}) = \{\varepsilon\};\$
- For all k > 0 and for all $u \in A^*$; $\phi_L(bbb^k au) = \phi_L(aabb^k au) = \phi_L(abbb^k au) =$ $S_{\min}(\{\varepsilon\}) = \{e\};$
- For all $u \in A^*$; $\phi_L(babu) = \phi_L(bbau) = S_{\min}(\{\varepsilon\}) = \{\varepsilon\}.$

It follows that \mathcal{A}_L is the following automaton:



We introduce the following invariant:

$$U(n) = \max_{L:m(L) \le n} [m(L \sqcup A^*)].$$

Proposition 5.10. One has $U(n) = O(2^n)$ and $U(n) = \Omega\left(\left(\frac{n}{|A|}\right)^{|A|}\right)$.

Proof. Let \mathcal{A} be an *n*-state minimal automaton. The automaton <u>Saturate</u>(\mathcal{A}) has also *n* states and accepts $L(\mathcal{A}) \sqcup \mathcal{A}^*$. It follows that $m(L(\mathcal{A}) \sqcup \mathcal{A}^*) \leq 2^n$. Consequently $U(n) = O(2^n)$.

Consider now the language $L = \bigcup_{a \in A} \{a^N\}$. One has m(L) = (N-1)|A| + 2. Let $A = \{a_1, a_2, \dots, a_n\}$. We claim that if (k_1, k_2, \dots, k_n) and $(\ell_1, \ell_2, \dots, \ell_n)$ are two distinct *n*-tuples of positive integers less than or equal to N - 1, then

$$(a_1^{k_1}a_2^{k_2}\cdots a_n^{k_n})^{-1}(L\sqcup A^*) \neq (a_1^{\ell_1}a_2^{\ell_2}\cdots a_n^{\ell_n})^{-1}(L\sqcup A^*).$$

Indeed, assume for example that $k_i > \ell_i$. Then $a_i^{N-\ell_i}$ belongs to the second quotient and not to the first one, proving the claim. It follows that

$$m(L \sqcup A^*) \ge (N-1)^{|A|}$$

Consequently, one has $U(n) = \Omega\left(\left(\frac{n}{|A|}\right)^{|A|} \right)$.

Proposition 5.11. If $|A| \ge 2$, then $U(n) = \Omega(r^{\sqrt{n}})$, where $r = (\frac{1+\sqrt{5}}{2})^{\frac{\sqrt{2}}{2}} \ge 1.39$.

Proof. Let a and b be two distinct letters of A and $n \ge 4$. If u is a word on A, we denote by u^R its reverse (*i.e.* the word obtained by reading u from the right to the left) and by $|u|_a$ (resp. $|u|_b$) the number of a (resp. b) in u. We consider the following languages:

$$L = \{uu^R \mid u \in \{a, b\}^*, \ |u|_a = 1, \ |u| = n\} \text{ and } K = L \sqcup A^*.$$

Since |L| = n and since each word of L has length 2n we have $m(L) \leq 2n^2$.

Now we prove that $m(K) \geq \frac{1}{7}(\frac{1+\sqrt{5}}{2})^n$. Let u and v be two distinct words of $\{a,b\}^* \cap (\{a,b\}^*aa\{a,b\}^*)^c$ such that |u| = |v| = n and $|u|_a \geq 2$ and $|v|_a \geq 2$. We claim that $u^{-1}K \neq v^{-1}K$. We may assume that $|u|_b \geq |v|_b$. One of the following cases arises:

 $\begin{aligned} |u|_b > |v|_b: & \text{let } w = b^{n-1-|u|_b}aab^{n-1} \in (b^{|u|_b})^{-1}L. \text{ Since } b^{|u|_b}w \in L, \, uw \in L \amalg A^*. \\ A^*. & \text{Thus } w \in u^{-1}K. \text{ Assume that } w \in v^{-1}K. \text{ Then, } vw \in L \amalg A^*. \\ & \text{Thus there exists } z \prec vw \text{ such that } z \in L. \text{ Consequently } |z|_b = 2n-2. \\ & \text{It follows that } |vw|_b \ge 2n-2. \text{ Therefore } |v|_b \ge 2n-2 - |w|_b = |u|_b, \\ & \text{a contradiction;} \end{aligned}$

$$|u|_b = |v|_b$$
: since $u \neq v$, there exist w , u_0 and v_0 such that $u = wbu_0$ and $v = wav_0$. The word $b^{|w|_b}ab^{|u|_b - |w|_b}$ is a subword of v . Let $s = b^{2n-2-|u|_b-|w|_b}ab^{|w|_b}$. Since $b^{|w|_b}ab^{|u|_b-|w|_b}s \in L$, $s \in v^{-1}K$. Assume that $s \in u^{-1}K$. There exists a subword z of us such that $z \in L$. Since $z \in L$, z is of the form $b^i ab^{2n-2-2i}ab^i$.

$$us: \qquad w \qquad b \qquad u_0 \qquad b^{2n-2-|u|_b-|w|_b}ab^{|w|_b}$$
$$z: \qquad b^i \qquad a \qquad b^{2n-2i-2} \qquad a \qquad b^i \qquad b^i$$

But $|us|_b = 2n - 2 = |z|_b$, thus, since $z \prec us$, z is obtained from us only by deleting some a's.

Since $|us|_b = |z|_b$, if the second *a* of *z* corresponds to the last *a* of *us*, then $i = |w|_b$; otherwise $i \ge 2n - 2 - |u|_b$, which is not possible since $i \le n - 1$ ($z \in L$) and since $|u|_b \le |u| - 2 \le n - 2$. Thus $i = |w|_b$. Now, two cases arise:

- $w \neq \varepsilon$: since v does not contain the factor aa, the last letter of w is b. If the first a of z corresponds to an a of w, then $i < |w|_b$; otherwise $i > |w|_b$, a contradiction;
- $w = \varepsilon$: in this case, $us = bu_0 b^{2n-2-|u|_b} a$. Since z is obtained from us only by deleting some a's, the first letter of z is a b. But we proved that $i = |w|_b$, thus $z = ab^{2n-2}a$, a contradiction.

Now we evaluate $|(\{a,b\}^*aa\{a,b\}^*)^c \cap \{a,b\}^*a\{a,b\}^*a\{a,b\}^* \cap \{a,b\}^n|$. For $d \in \{a,b\}$, let

$$R_n(d) = (\{a,b\}^* aa\{a,b\}^*)^c \cap \{a,b\}^* a\{a,b\}^* a\{a,b\}^* \cap \{a,b\}^n \cap \{a,b\}^* da\{a,b\}^* a\{a,b\}^n \cap \{a,b\}^* da\{a,b\}^n \cap \{a,b\}^n da\{a,b\}^n \cap \{a,b\}^n da\{a,b\}^n da\{a,b$$

We have $R_n(b)a \subseteq R_{n+1}(a)$ and $R_{n+1}(b) = R_n(a)b \cup R_n(b)b$ (this union is disjoint). Consequently,

$$|R_{n+1}(b)| = |R_n(a)b| + |R_n(b)b|$$

= |R_n(a)| + |R_n(b)|
$$\geq |R_{n-1}(b)a| + |R_n(b)|$$

$$\geq |R_{n-1}(b)| + |R_n(b)|.$$
(*)

Since

$$|R_4(b)| = |\{abab\}| = 1 \ge \frac{1}{7} \left(\frac{1+\sqrt{5}}{2}\right)^4 \approx 0.98$$

and since

$$|R_5(b)| = |\{ababb, babab, abbab\}| = 3 \ge \frac{1}{7} \left(\frac{1+\sqrt{5}}{2}\right)^5 \approx 1.58,$$

we have, for all $n \ge 4$, $R_n(b) \ge \frac{1}{7} \left(\frac{1+\sqrt{5}}{2}\right)^n$. Indeed this result holds for n = 4 and n = 5. Moreover, by induction, for all $n \ge 5$,

$$|R_{n+1}(b)| \ge |R_{n-1}(b)| + |R_n(b)| \qquad (by (*))$$

$$\ge \frac{1}{7} \left(\frac{1+\sqrt{5}}{2}\right)^{n-1} + \frac{1}{7} \left(\frac{1+\sqrt{5}}{2}\right)^n \text{ (by induction)}$$

$$\ge \frac{1}{7} \left(\frac{1+\sqrt{5}}{2}\right)^{n-1} \left(1 + \left(\frac{1+\sqrt{5}}{2}\right)\right)$$

$$= \frac{1}{7} \left(\frac{1+\sqrt{5}}{2}\right)^{n-1} \left(\frac{1+\sqrt{5}}{2}\right)^2$$

$$= \frac{1}{7} \left(\frac{1+\sqrt{5}}{2}\right)^{n+1} \cdot$$

It follows that $m(K) \ge \frac{1}{7}(\frac{1+\sqrt{5}}{2})^n$. Since U(n) is an increasing function and since $m(L) \le 2n^2$, we have

$$U(n) = \Omega(r^{\sqrt{n}}).$$

5.2. Union and intersection of shuffle ideals

Proposition 5.12. For all $n, m \ge 0$ it holds that

$$\max_{L_1,L_2 \text{ shuffle ideals}} \{m(L_1 \cup L_2) \mid m(L_1) \leq n, \ m(L_2) \leq m\} = \Theta(mn).$$

Proof. It is a general result for regular languages that $m(L_1 \cup L_2) \leq m(L_1)m(L_2)$. For the lower bound consider the languages $L_1 = a^n \sqcup A^*$ and $L_2 = b^m \sqcup A^*$.

One has $m(L_1) = n + 1$ and $m(L_2) = m + 1$. Let $k, k' \in \{0, ..., n-1\}$ and $\ell, \ell' \in \{0, ..., m-1\}$. Assume that $(k, \ell) \neq (k', \ell')$.

One has, for example, k > k'. In this case,

$$a^{n-k} \in (a^k b^\ell)^{-1}(L_1 \cup L_2)$$
 and $a^{n-k} \notin (a^{k'} b^{\ell'})^{-1}(L_1 \cup L_2).$

It follows that $L_1 \cup L_2$ has at least mn right quotients. It follows that $m(L_1 \cup L_2) \ge mn$.

Proposition 5.13. For all $n, m \ge 0$ it holds that

$$\max_{L_1,L_2 \text{ shuffle ideals}} \{ m(L_1 \cap L_2) \mid m(L_1) \le n, \ m(L_2) \le m \} = \Theta(mn).$$

Proof. It is a general result for regular languages that $m(L_1 \cap L_2) \leq m(L_1)m(L_2)$.

For the lower bound consider the languages $L_1 = a^n \sqcup A^*$ and $L_2 = b^m \sqcup A^*$. One has $m(L_1) = n + 1$ and $m(L_2) = m + 1$. Let $k, k' \in \{1, \ldots, n\}$ and $\ell, \ell' \in \{1, \ldots, m\}$. Assume that $(k, \ell) \neq (k', \ell')$. One has, for example, k > k'. In this case,

$$a^{n-k}b^{m-\ell} \in (a^k b^\ell)^{-1}(L_1 \cap L_2)$$
 and $a^{n-k}b^{m-\ell} \notin (a^{k'}b^{\ell'})^{-1}(L_1 \cap L_2).$

It follows that $L_1 \cap L_2$ has at least mn right quotients. Consequently $m(L_1 \cap L_2) \ge mn$.

5.3. Product and star of shuffle ideals

Proposition 5.14. If L is a shuffle ideal, then $m(L^*) \leq m(L) + 1$.

Proof. We claim that $L^2 \subseteq L$. Indeed if $u \in L^2$, one has u = vw with $v \in L$ and $w \in L$. It follows that $u \in v \sqcup A^* \subseteq L$, proving the claim.

By a direct induction, one can prove that $L^* = \{\varepsilon\} \cup L$. Let $\mathcal{A} = (Q, A, E, i, F)$ be the minimal automaton of L and $p \notin Q$. The automaton $\mathcal{A}' = (Q', A, E', p, F')$ where

-
$$Q' = Q \cup \{p\};$$

- $E' = E \cup \{(p, a, q) \mid (i, a, q) \in E\};$
- $F' = F \cup \{p\};$

accepts $L \cup \{\varepsilon\}$, is deterministic and has m(L) + 1 states. It follows that $m(L^*) \leq m(L) + 1$.

We may have $m(L^*) < m(L)$. For example, if $L = A^* a A^* \cup A^* b A^*$ and if $A = \{a, b\}$, then m(L) = 2. But $L^* = A^*$ and so $m(L^*) = 1$.

Proposition 5.15. If L_1 and L_2 are non empty shuffle ideals, then

$$m(L_1L_2) = m(L_1) + m(L_2) - 1.$$

Proof. Recall that by Lemma 5.2, the minimal automaton of a shuffle ideal has a unique final state. Now let $\mathcal{A}_1 = (Q_1, A, E_1, i_1, f_1)$ and $\mathcal{A}_2 = (Q_2, A, E_2, i_2, f_2)$ be the minimal automata of L_1 and L_2 . We may assume that $Q_1 \cap Q_2 = \emptyset$. Let

$$\mathcal{A}_3 = (Q_1 \cup Q_2 \setminus \{i_2\}, A, E_3, i_1, f_2)$$

where

$$E_{3} = E_{1} \setminus \{ (f_{1}, a, p) \mid (f_{1}, a, p) \in E_{1} \}$$

$$\cup E_{2} \setminus \{ (i_{2}, a, p) \mid (i_{2}, a, p) \in E_{2} \}$$

$$\cup \{ (f_{1}, a, p) \mid (i_{2}, a, p) \in E_{2} \} \cdot$$

We claim that \mathcal{A}_3 is the minimal automaton of L_1L_2 .

- By construction, \mathcal{A}_3 is deterministic.
- By construction $L(\mathcal{A}_3) \subseteq L_1L_2$. Let $u = vw \in L_1L_2$, with $v \in L_1$ and $w \in L_2$ and let v' be the shortest prefix of v which is in L_1 . We have u = v'zw with v = v'z. Since L_2 is a shuffle ideal, $w' = zw \in L_2$. By construction, one can read v' in \mathcal{A}_1 by using transitions in $E_1 \setminus \{(f_1, a, p) \mid (f_1, a, p) \in E_1\}$. It follows that $v'w' \in L(\mathcal{A}_3)$, proving that $L(\mathcal{A}_3) = L_1L_2$.
- Now we shall prove that \mathcal{A}_3 is minimal. Let p and q be two states of \mathcal{A}_3 such that $p \cdot u = f_2$ if and only if $q \cdot u = f_2$ for all $u \in A^*$. The following cases arise:
- case (1): p and q are in $Q_2 \cup \{f_1\}$. Since \mathcal{A}_2 is minimal, p = q;
- case (2): $p \in Q_1$ and $q = f_1$. Let v be a word of L_2 of minimal length. We have $q \cdot v = f_2$. Thus $p \cdot v = f_2$. Since all paths from p to f_2 go through q and by the minimality of $v, p \cdot v = f_2$ implies p = q;
- case (3): p and q are both in Q_1 . Assume that there exists u such that $p \cdot u = f_1$ and $q \cdot u \neq f_1$. If $q \cdot u \in Q_1$ we are in case (2) and p = q. If $q \cdot u \in Q_2$ we are in case (1) and p = q. Now if $p \cdot u = f_1$ if and only if $q \cdot u = f_1$, we have p = q by minimality of \mathcal{A}_1 ;
- case (4): $p \in Q_1$ and $q \in Q_2$. Let v be a word such that $p \cdot v = f_1$. Using case (1), we have $q \cdot v = f_1$. Hence, by Proposition 5.1, we have $q = f_1$. Now, using case (2), we have p = q.

It follows that p = q, proving that \mathcal{A}_3 is minimal.

6. Shuffle ideals and linear temporal logic

In this section we first present a temporal logic characterization of shuffle ideals. Next we study the expressive power of this fragment of temporal logic over infinite words.

6.1. Temporal logic on finite words

Linear temporal logic (LTL for short) on an alphabet A is defined as follows [22]. The vocabulary consists of

- (1) an atomic proposition p_a for each letter $a \in A$ and the atomic propositions true and false;
- (2) connectives \lor and \neg ;
- (3) temporal operators X ("next"); \mathcal{U} ("until");

and the formulas are constructed according to the rules

- (1) atomic propositions are formulas;
- (2) if φ and ψ are formulas, so are $\varphi \lor \psi$, $\neg \varphi$, $\mathsf{X}\varphi$, $\varphi \mathcal{U}\psi$.

The semantics is defined by induction. Given a word $u \in A^+$, and $n \in \{1, \ldots, |u|\}$, we say that u satisfies φ at position n (denoted $(u, n) \models \varphi$) as follows

- (1) $(u,n) \models p_a$ if the *n*-th letter u(n) of *u* is an *a*;
- (2) $(u,n) \models \varphi \lor \psi$ if $(u,n) \models \varphi$ or $(u,n) \models \psi$;
- (3) $(u,n) \models \neg \varphi$ if (u,n) does not satisfy φ ;
- (4) $(u,n) \models \mathsf{X}\varphi$ if n < |u| and $(u, n+1) \models \varphi$;
- (5) $(u,n) \models \varphi \mathcal{U} \psi$ if there exists m such that $n \leq m \leq |u|$, $(u,m) \models \psi$ and, for every k such that $n \leq k < m$, $(u,k) \models \varphi$.

Moreover, for all $u \in A^+$ and all $n \in \{1, \ldots, |u|\}$ we have $(u, n) \models \texttt{true}$ and (u, n) does not satisfy false.

Finally, we denote by $L(\varphi)$ the language of words u such that (u, 1) satisfies φ . It is known [8, 14, 15, 20] that a language L is LTL-definable (*i.e.* there exists a LTL formula φ such that $L = L(\varphi)$) if and only if L is star-free. Reference [6] also gives a characterization of languages accepted by LTL formulas defined without the symbol \mathcal{U} .

Now we give a temporal logic characterization of shuffle ideals. For each letter a, we define a new temporal operator F_a and we define SLTL (Shuffle Linear Temporal Logic) formulas according to the rules

- (1) true is a SLTL-formula;
- (2) if φ and ψ are formulas, so are $\varphi \lor \psi, \varphi \land \psi, \mathsf{X}\varphi, \mathsf{F}\varphi$ and $\mathsf{F}_a\varphi$.

The semantics is defined as for LTL and by letting:

 $\begin{array}{l} (u,n) \models \varphi \land \psi \text{ if } (u,n) \models \neg(\neg \varphi \lor \neg \psi); \\ (u,n) \models \mathsf{F}\varphi \text{ if } (u,n) \models \mathsf{true } \mathcal{U}\varphi; \\ (u,n) \models \mathsf{F}_a\varphi \quad \text{if } (u,n) \models \mathsf{F}(p_a \land (\mathsf{X}\varphi)). \end{array}$

A SLTL⁻-formula is a SLTL-formula that uses **true** and the operators \vee and F_a . A language *L* is *SLTL-definable* [resp. SLTL⁻-definable] if there exists a SLTL-formula (resp. SLTL⁻-formula) φ such that $L = L(\varphi)$.

Theorem 6.1. Let $L \subseteq A^+$. The following assertions are equivalent.

(1) L is SLTL-definable.

- (2) L is $SLTL^{-}$ -definable.
- (3) L is a shuffle ideal.

To prove this theorem we need the following lemma:

Lemma 6.2. If L is a shuffle ideal, then AL and A^*L are shuffle ideals.

Proof. Since the product of two shuffle ideals is a shuffle ideal, A^*L is a shuffle ideal.

Since $AA^* = A^*AA^*$, by Theorem 1.2, AL is a shuffle ideal. The same argument holds for LA.

We can now prove Theorem 6.1.

Proof. It is clear that (2) implies (1).

Now we shall prove that (1) implies (3). Let ψ be a SLTL-formula. We prove by induction on ψ that $L(\psi)$ is a shuffle ideal.

- $L(true) = A^+ = A^*AA^*$ is a shuffle ideal.
- If $\psi = \varphi_1 \lor \varphi_2$ (resp. $\psi = \varphi_1 \land \varphi_2$) where $L(\varphi_1)$ and $L(\varphi_2)$ are shuffle ideals, then $L(\psi)$ is a shuffle ideal because the union (resp. intersection) of two shuffle ideals is a shuffle ideal.
- If $\psi = \mathsf{F}\varphi$ where $L(\varphi)$ is a shuffle ideal, then $L(\psi) = A^*L(\varphi)$ is a shuffle ideal (by Lem. 6.2).
- If $\psi = X\varphi$ where $L(\varphi)$ is a shuffle ideal, then $L(\psi) = AL(\varphi)$ is a shuffle ideal (by Lem. 6.2).
- If $\psi = \mathsf{F}_a \varphi$ where $L(\varphi)$ is a shuffle ideal, then $L(\psi) = A^*(aA^* \cap AL(\varphi)) = A^*aL(\varphi)$ is a shuffle ideal.

To finish the proof, we prove that (3) implies (2). Let a be in A. We have $L(\mathsf{F}_a \mathsf{true}) = A^* a A^*$. By a direct induction we have

$$L(\mathsf{F}_{a_1}\mathsf{F}_{a_2}\cdots\mathsf{F}_{a_k}\mathsf{true}) = A^*a_1A^*a_2A^*\cdots A^*a_kA^*.$$

It follows that every shuffle ideal of A^+ is SLTL⁻-definable.

The above proof is constructive: given a shuffle ideal by a polynomial rational expression, one can compute a SLTL⁻-formula representing it, and given a SLTL-formula one can construct the shuffle ideal it represents.

6.2. Temporal logic over infinite words

One can also define a semantic for LTL for infinite words. If σ is an infinite word, we denote by $\sigma(i)$ its *i*-th letter. The semantic for LTL is the following:

- (1) $(\sigma, n) \models p_a$ if $\sigma(n) = a$. Moreover $(\sigma, n) \models$ true and (σ, n) does not satisfy false;
- (2) $(\sigma, n) \models \varphi \lor \psi$ if $(\sigma, n) \models \varphi$ or $(\sigma, n) \models \psi$;
- (3) $(\sigma, n) \models \neg \varphi$ if (σ, n) does not satisfy φ ;
- (4) $(\sigma, n) \models \mathsf{X}\varphi$ if $(\sigma, n+1) \models \varphi$;
- (5) $(\sigma, n) \models \varphi \mathcal{U} \psi$ if there exists m such that $n \leq m$, $(\sigma, m) \models \psi$ and, for every k such that $n \leq k < m$, $(\sigma, k) \models \varphi$;

Finally, we denote by $L_{\omega}(\varphi)$ the language of infinite words σ such that $(\sigma, 1)$ satisfies φ . As for languages of finite words, we say that an ω -language L is SLTL (resp. SLTL⁻) recognizable if there exists a SLTL-formula (resp. a SLTL⁻ formula) φ such that $L = L_{\omega}(\varphi)$.

Proposition 6.3. Let ψ be a SLTL-formula. Then

$$L_{\omega}(\psi) = L(\psi)A^{\omega}$$

Proof. We proceed by induction on the formula ψ .

- $L_{\omega}(\texttt{true}) = A^{\omega} = A^* A^{\omega} = L(\texttt{true})A^{\omega}.$
- If $\psi = \varphi_1 \vee \varphi_2$ where $L_{\omega}(\varphi_1) = L(\varphi_1)A^{\omega}$ and $L_{\omega}(\varphi_2) = L(\varphi_2)A^{\omega}$, then $L_{\omega}(\psi) = L_{\omega}(\varphi_1) \cup L_{\omega}(\varphi_2) = L(\varphi_1)A^{\omega} \cup L(\varphi_2)A^{\omega} = (L(\varphi_1) \cup L(\varphi_2))A^{\omega} = L(\psi)A^{\omega}$.
- If $\psi = \mathsf{F}\varphi$ where $L_{\omega}(\varphi) = L(\varphi)A^{\omega}$, then $L_{\omega}(\psi) = A^*L_{\omega}(\varphi) = A^*L(\varphi)A^{\omega} = L(\mathsf{F}\varphi)A^{\omega} = L(\psi)A^{\omega}$.
- If $\psi = \mathsf{X}\varphi$ where $L_{\omega}(\varphi) = L(\varphi)A^{\omega}$, then $L_{\omega}(\psi) = AL_{\omega}(\varphi) = AL(\varphi)A^{\omega} = L(\mathsf{X}\varphi)A^{\omega} = L(\psi)A^{\omega}$.
- If $\psi = \varphi_1 \wedge \varphi_2$ where $L_{\omega}(\varphi_1) = L(\varphi_1)A^{\omega}$ and $L_{\omega}(\varphi_2) = L(\varphi_2)A^{\omega}$, then $L_{\omega}(\psi) = L_{\omega}(\varphi_1) \cap L_{\omega}(\varphi_2) = L(\varphi_1)A^{\omega} \cap L(\varphi_2)A^{\omega} = [L(\varphi_1)A^* \cap L(\varphi_2)]A^{\omega} \cup [L(\varphi_1) \cap L(\varphi_2)A^*]A^{\omega}$. But $L(\varphi_1)A^* = L(\varphi_1)$ and $L(\varphi_2)A^* = L(\varphi_2)$. Consequently $L_{\omega}(\psi) = L(\psi)A^{\omega}$.
- If $\psi = \mathsf{F}_a \varphi$ where $L_\omega(\varphi) = L(\varphi)A^\omega$, then $L_\omega(\psi) = A^*L_\omega(p_a \wedge \mathsf{X}\varphi) = A^*aL_\omega(\varphi) = A^*aL(\varphi)A^\omega = L(\mathsf{F}_a\varphi)A^\omega = L(\psi)A^\omega$.

Lemma 6.4. Let $\mathcal{A} = (Q, A, E, I, F)$ be a saturated automaton. We have

$$L_{\omega}(\mathcal{A}) = L(\mathcal{A})A^{\omega}$$

Proof. For an automaton we always have $L_{\omega}(\mathcal{A}) \subseteq L(\mathcal{A})A^{\omega}$.

Let u be in $L(\mathcal{A})$ and σ in \mathcal{A}^{ω} . There exist $i \in I$ and $f \in F$ such that $f \in i \cdot u$. Since \mathcal{A} is saturated, for each prefix s of σ , $f \in i \cdot us$. Consequently $u\sigma \in L_{\omega}(\mathcal{A})$, proving that $L(\mathcal{A})\mathcal{A}^{\omega} \subseteq L_{\omega}(\mathcal{A})$.

Theorem 6.5. Let K be an ω -language of A^{ω} . The following conditions are equivalent:

- (1) K is SLTL-recognizable;
- (2) K is $SLTL^{-}$ -recognizable;
- (3) There exists a shuffle ideal L such that $K = LA^{\omega}$;
- (4) K is accepted by a locally trivial, saturated Büchi-automaton;
- (5) K is accepted by a saturated Büchi-automaton.

Proof. By Theorem 6.1 and Proposition 6.3, (1) implies (3). It is clear that (2) implies (1).

Now let L be a shuffle ideal such that $K = LA^{\omega}$. By Theorem 6.1, there exists a SLTL⁻-formula φ such that $L = L(\varphi)$. Thus, by Proposition 6.3, $L_{\omega}(\varphi) = LA^{\omega} = K$. Therefore, (3) implies (2).

The equivalence between (3, 4) and (5) is a direct consequence of Lemma 6.4 and Theorem 2.1.

Acknowledgements. The author wishes to thank Ines Klimann and Anca Muscholl for many contributions and comments about this paper. The author also wishes to thank the anonymous referees for helpful and constructive suggestions.

References

- A. Aho, J. Hopcroft and J. Ullman, The design and analysis of computer algorithms. Addison-Wesley (1974) 395-400.
- J. Almeida, Implicit operations on finite *J*-trivial semigroups and a conjecture of I. Simon. J. Pure Appl. Algebra 69 (1990) 205-218.
- [3] M. Arfi, Opération polynomiales et hiérarchies de concaténation. Theoret. Comput. Sci. 91 (1991) 71-84.
- [4] J. Berstel and L. Boasson, Shuffle factorization is unique, Technical Report. LIAFA, Université Paris 7 (1999).
- [5] J. Berstel, Transductions and context-free languages. Teubner (1979) Verlag.
- [6] J. Cohen, D. Perrin and J.-E. Pin, On the expressive power of temporal logic for finite words. J. Comput. System Sci. 46 (1993) 271-294.
- [7] S. Eilenberg, Automata, Languages and Machines, Vol. A. Academic Press (1974).
- [8] D. Gabbay, A. Pnueli, S. Shelah and J. Stavi, On the temporal analysis of fairness, in 12th ACM Symp. on Principles of Programming Languages (1980) 163-180.
- [9] C. Glasser and H. Schmidt, Level 5/2 of the straubing-thérien hierarchy for two-letter alphabets, in *Conference on Developments in Language Theory (DLT)*. Vienna (2001).
- [10] P.-C. Héam, Some complexity results for polynomial rational expressions. *Theoret. Comput. Sci.* (to appear).
- [11] G. Higman, Ordering by divisibility in abstract algebras, in Proc. of the London Mathematical Society, Vol. 2 (1952) 326-336.
- [12] C. Hagenah and A. Muscholl, Computing ϵ -free nfa from regular expressions in $O(n \log^2(n))$ time. *RAIRO: Theoret. Informatics Appl.* **34** (2000) 257-277.
- [13] J. Hopcroft and J. Ullman, Introduction to automata theory, languages, and computation. Addison-Wesley (1980).
- [14] J.A. Kamp, Tense logic and the theory of linear order, Ph.D. Thesis. University of California, Los Angeles (1968).

- [15] O. Katai, Completeness and the expressive power of next time temporal logical system by semantic tableau method, Technical Report RR-0109. Inria, Institut National de Recherche en Informatique et en Automatique (1981).
- [16] S.C. Kleene, Representation of events in nerve nets and finite automata. Princeton University Press, Automata Studies (1956) 3-42.
- [17] M. Lothaire, Combinatorics on words. Cambridge Mathematical Library (1983).
- [18] M. Nivat, G.D.S. Ramkumar, C. Pandu Rangan, A. Saoudi and R. Sundaram, Efficient parallel shuffle recognition. *Parallel Process. Lett.* 4 (1994) 455-463.
- [19] C.H. Papadimitriou, Computational complexity. Addison Wesley (1994).
- [20] M. Parigot, Automates, réseaux, formules, in Actes des journées "Informatiques et Mathématiques". Luminy (1984).
- [21] B. Pradeep, C. Murthy and S. Ram, A constant time string shuffle algorithm on reconfigurable meshes. Int. J. Comput. Math. 68 (1998) 251-259.
- [22] A. Pnueli, The temporal logic of programs, in 18th FOCS (1977) 46-57.
- [23] J.-E. Pin and P. Weil, Polynomial closure and unambiguous product. Theory Comput. Systems 30 (1997) 1-39.
- [24] D.E. Radford, A natural ring basis for shuffle algebra and an application to group schemes. J. Algebra 58 (1979) 432-454.
- [25] I. Simon, Piecewise testable events, in GI Conf. Springer-Verlag, Lecture Notes in Comput. Sci. 33 (1975) 214-222.
- [26] J.-C. Spehner, Le calcul rapide des mélanges de deux mots. Theoret. Comput. Sci. 47 (1986) 181-203.
- [27] H. Straubing and D. Thérien, Partially ordered finite monoids and a theorem of I. Simon. J. Algebra 119 (1985) 161-183.
- [28] J. Stern, Characterization of some classes of regular events. Theoret. Comput. Sci. 35 (1985) 17-42.
- [29] H. Straubing, Finite semigroups varieties of the form V*D. J. Pure Appl. Algebra 36 (1985) 53-94.
- [30] D. Thérien, Classification of finite monoids: The language approach. Theoret. Comput. Sci. 14 (1981) 195-208.
- [31] W. Thomas, Classifying regular events in symbolic logic. J. Comput. System Sci. 25 360-375.
- [32] D. Thérien and Th. Wilke, Temporal logic and semidirect products: An effective characterization of the until hierarchy, in Proc. of the 37th Annual Symposium on Foundations of Computer Science. IEEE (1996) 256-263.
- [33] Th. Wilke, Classifying discrete temporal properties, in STACS'99. Springer-Verlag, Lecture Notes in Comput. Sci. 1563 (1999) 32-46.
- [34] S. Yu, State complexity of regular languages, in Proc. of the International Workshop on Descriptional Complexity of Automata, Grammars and Related Structures (1999) 77-88.

Communicated by J.-E. Pin.

Received November, 2001. Accepted October, 2002.

To access this journal online: www.edpsciences.org