ELSEVIER

Theoretical Computer Science 250 (2001) 31–53

Theoretical Computer Science

www.elsevier.com/locate/tcs

Shuffle languages are in **P**

Joanna Jędrzejowicz *, Andrzej Szepietowski

Institute of Mathematics, University of Gdańsk, ul Wita Stwosza 57, 80952 Gdańsk, Poland

Received January 1998; revised October 1998
Communicated by A. Salomaa

Abstract

In this paper we show that shuffle languages are contained in **one-way-NSPACE(log n)** thus in **P**. We consider the class of shuffle languages which emerges from the class of finite languages through regular operations (union, concatenation, Kleene star) and shuffle operations (shuffle and shuffle closure). For every shuffle expression E we construct a shuffle automaton which accepts the language generated by E and we show that the automaton can be simulated by a one-way nondeterministic Turing machine in logarithmic space.

Keywords: Complexity; Polynomial-time complexity; Shuffle languages; Shuffle; Automata

1. Introduction

The operations shuffle and shuffle closure have been introduced to describe sequentialized execution histories of concurrent processes [11, 12]. Together with other operations they describe various classes of languages which have been extensively studied (see [1, 2, 4–7, 13]). Here, we consider the class of shuffle languages which emerges from the class of finite languages through regular operations (union, concatenation, Kleene star) and shuffle operations (shuffle and shuffle closure).

It was known that shuffle languages are properly contained in the class of context-sensitive languages and not comparable with the class of context-free languages. The complexity of the shuffle operation was discussed in [5, 9, 10, 13]. In [5] it was shown that the class of extended regular languages, ER, which is an extension of regular languages by the shuffle closure operator only, is contained in **NSPACE**$(\log n)$ and in **P**. On the other hand, in [13] it was shown that the language $\{ab^n cde^n f : n \geqslant 0\}^{\otimes}$ is

* Corresponding author.
E-mail addresses: matjoj@paula.univ.gda.pl (J. Jędrzejowicz), matszp@paula.univ.gda.pl (A. Szepietowski).

0304-3975/01/$ - see front matter
PII: S0304-3975(99)00109-7

NP-complete, and also the problem of deciding for a sequence of words $t, u_1, \ldots, u_m$ whether $t \in u_1 \odot u_2 \cdots \odot u_m$ or the problem of deciding for a pair of words t and v whether $t \in v^{\otimes}$ are **NP**-complete ($\odot$ and $\otimes$ are shuffle and shuffle closure operators).

The aim of this paper is to show that shuffle languages are contained in one-way-NSPACE($\log n$) thus in P (see [3]). Observe that the class of shuffle languages can only form a proper subclass of one-way-NSPACE($\log n$) since $\{a^n b^n : n \geqslant 1\}$ is not a shuffle language [6]. Besides, neither one-way-NSPACE($\log n$) nor NSPACE($\log n$) is closed under shuffle closure, unless $\mathbf{P} = \mathbf{NP}$. This follows from the just mentioned fact that the language $\{ab^n cde^n f : n \geqslant 0\}^{\otimes}$ is **NP**-complete.

To prove our result we introduce a new tool – shuffle automata, which on one hand, accept shuffle languages and on the other hand, they can be simulated by one way nondeterministic Turing machines in logarithmic space. For every shuffle expression E we construct a shuffle automaton A_E which accepts the language generated by E and we show that the automaton can be simulated by a one-way nondeterministic Turing machine in logarithmic space. We use the Turing machine model with one-way read-only input tape and a separate two-way read-write work tape. Only $\log n$ cells of work tape can be used during computations.

The paper is organized as follows. In Section 2 we recall the notion of a shuffle expression and a shuffle language, and we introduce shuffle automata. For every shuffle expression E we construct a shuffle automaton A_E and define the computation with distinguishable markers. In Section 3 we prove that the language accepted by A_E is equal to the language generated by E. In Section 4 for the automaton A_E we define a finite set of types of markers and introduce a computation with nondistinguishable markers. We do not distinguish markers of the same type and only have to remember the number of markers of each type. This kind of computation is more economical in terms of computational complexity. In Section 5 we show that computations with distinguishable and nondistinguishable markers are equivalent. Finally in Section 6 we show that computations with nondistinguishable markers can be simulated by a one-way Turing machine with logarithmic space.

2. Shuffle automata

2.1. Shuffle expressions

Let Σ be any fixed alphabet and λ the empty word. The shuffle operation $\odot$ is defined inductively as follows:

- $u \odot \lambda = \lambda \odot u = \{u\}$, for $u \in \Sigma^*$ and
- $au \odot bv = a(u \odot bv) \cup b(au \odot v)$, for $u, v \in \Sigma^*$ and $a, b \in \Sigma$.

For any languages $L_1, L_2 \subset \Sigma^*$, the shuffle $L_1 \odot L_2$ is defined as

$$L_1 \odot L_2 = \bigcup_{u \in L_1, v \in L_2} u \odot v.$$

For any language L, the shuffle closure operator is defined by

$$L^{\otimes} = \bigcup_{i=0}^{\infty} L^{\odot i} \quad \text{where } L^{\odot 0} = \{\lambda\} \quad \text{and} \quad L^{\odot i} = L^{\odot i-1} \odot L.$$

Definition 2.1. Each $a \in \Sigma$, λ and $\emptyset$ are shuffle expressions. If S_1, S_2 are shuffle expressions, then $(S_1 \cdot S_2)$, ${S_1}^*$, $(S_1 + S_2)$, $(S_1 \odot S_2)$ and ${S_1}^{\otimes}$ are shuffle expressions, and nothing else is a shuffle expression.

The language $L(S)$ generated by a shuffle expression S is defined as follows. $L(a) = \{a\}$, $L(\lambda) = \{\lambda\}$, $L(\emptyset) = \emptyset$. If $L(S_1) = L_1$ and $L(S_2) = L_2$, then $L((S_1 \cdot S_2)) = L_1 \cdot L_2$, $L((S_1 + S_2)) = L_1 \cup L_2$, $L({S_1}^*) = L_1^*$, $L((S_1 \odot S_2)) = L_1 \odot L_2$, and $L({S_1}^{\otimes}) = L_1^{\otimes}$.

2.2. Examples

Before we come to a formal definition of shuffle automata we start with an intuitive description and some examples.

A shuffle automaton A is a five-tuple (Q, Σ, q_0, q_f, T), where Q is a finite set of states, Σ is a finite input alphabet, $q_0, q_f \in Q$ are two distinguished states – the initial one and the final one, $T \subset (\Sigma \cup \{\lambda\}) \times Q^2 \cup \{\lambda\} \times Q^3$ is a finite set of transitions.

We extend the idea of the finite state machine by introducing special states for the shuffle and shuffle closure operator. Similarly, as for Petri nets, markers are introduced and they take part in the process of accepting words by the automaton. Each marker will be placed in a single state or in a set of states, so we can compare the marker to a spider which has a finite number of legs and is moving its legs when performing transitions of the automaton.

Below we consider three examples of shuffle automata constructed, respectively, for a regular expression, for an expression with the shuffle operator and an expression with the shuffle closure operator.

Let $E = (ab)^*$ be the regular expression. The shuffle automaton A_E is a finite state Rabin–Scott automaton shown in Fig. 1, with arrows representing transitions of the automaton. There is only one marker taking part in any computation. It has one leg which is moved around and is placed in this state of the automaton which corresponds to the finite control of the finite automaton. A word w is accepted by A_E if there is a sequence of transitions $t = t_1 t_2 \cdots t_s$ which move the marker from the state q_0 to q_5 and the labels of t form w. For example, when accepting the word $abab$ the marker starts in the initial state q_0 and visits subsequently q_1, q_2, q_3, q_4, q_1, q_2, q_3, q_4, q_5. It is obvious that A_E accepts the language $(ab)^*$.

For the expression $F = ab \odot c$, the shuffle automaton A_F is shown in Fig. 2. We still have one marker. While performing an accepting computation the marker starts in the initial state s with one leg and then proceeds with two legs until it reaches the final state e. For example, for the word acb, the marker moves as follows. It starts in s. After performing the transition (λ, s, q_1, q_5) the marker has two legs, one leg is placed in the state q_1 and the other in q_5. Then the transition (a, q_1, q_2) is applied and the marker is placed in $\{q_2, q_5\}$ (the first leg was moved from q_1 to q_2). The consecutive

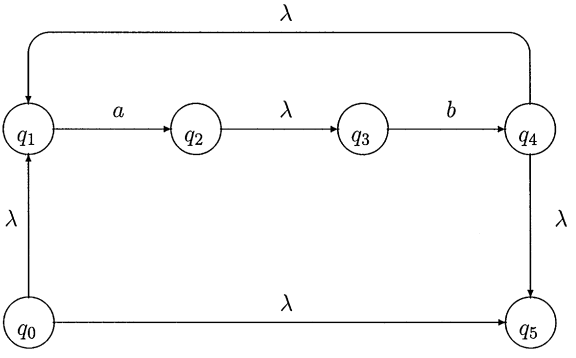

Fig. 1. Shuffle automaton for $(ab)^*$.

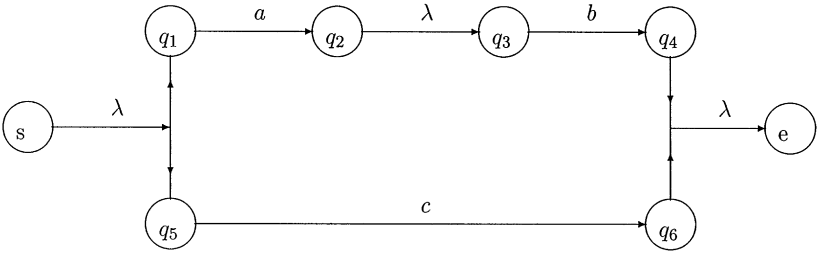

Fig. 2. Shuffle automaton for $ab \odot c$.

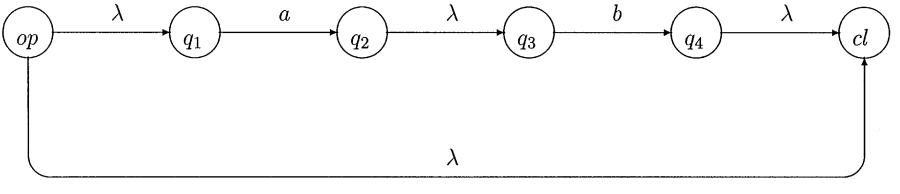

Fig. 3. Shuffle automaton for $(ab)^{\otimes}$.

transitions change the positions of the marker into $\{q_3, q_5\}$, $\{q_3, q_6\}$, $\{q_4, q_6\}$. Finally, the transition (λ, q_4, q_6, e) is applied, the marker has again one leg which is placed in e.

For the shuffle expression $G = (ab)^{\otimes}$, the respective shuffle automaton A_G is shown in Fig. 3. Now the situation is more complicated. There is one main marker M which is initially placed in the initial state op. Each application of the transition (λ, op, q_1) creates a new marker which is a son of the main marker M. Consider a computation of A_G which accepts the word $aababb \in L(A_G)$. We start with the main marker placed in op. After the transition $t_1 = (\lambda, op, q_1)$ is applied a new marker N_1 appears in q_1, the main marker remains in op. The transition $t_2 = (a, q_1, q_2)$ places the marker N_1 in q_2. Then t_1 is applied again and a new marker N_2 is placed in q_1. The other markers remain in their places, that is M in op and N_1 in q_2. Then t_2 moves N_2 to q_2. The

marker N_1 is shifted to q_3 by $t_3=(\lambda,q_2,q_3)$ and then to q_4 by $t_4=(b,q_3,q_4)$. So far the word *aab* has been processed.

Afterwards t_1 is applied, this creates a new marker N_3 in q_1. Consecutive applications of t_2, t_3, t_4 move the marker N_3 to q_4. Similarly, t_3, t_4 move the marker N_2 from q_2 through q_3 to q_4. Then we apply three times the transition $t_5=(\lambda,q_4,cl)$. Each application of t_5 kills one marker placed in q_4 and eventually we only have the main marker M in *op*. Finally, the transition $t_0=(\lambda,op,cl)$ moves M from *op* to *cl*. Thus the transitions

$$t_1t_2t_1t_2t_3t_4t_1t_2t_3t_4t_3t_4t_5t_5t_5t_0$$

were used to accept the word *aababb*. In the computation altogether four markers were used (each of them had one leg during its whole life) – the main marker M which was the father of three markers N_1,N_2,N_3. A word w is accepted by A_G if there is a sequence of transitions $t=t_1t_2\cdots t_s$ which satisfy the conditions:

- each transition t_i moves either the main marker or one of its sons,
- a new marker (a son of M) can be born only if M stands in *op*,
- the transition t_0 is applied at the end of the computation after all the sons have been killed,
- the labels of t form w.

2.3. Construction of the automaton

Now for every shuffle expression E over the alphabet Σ we construct a shuffle automaton A_E which is a five-tuple (Q,Σ,q_0,q_f,T), where Q is a finite set of states, Σ is a finite input alphabet, $q_0,q_f\in Q$ are two distinguished states – the initial one and the final one, $T\subset(\Sigma\cup\{\lambda\})\times Q^2\cup\{\lambda\}\times Q^3$ is a finite set of transitions. We shall assume that the states of the shuffle automaton are of five different kinds, $Q=ORD\cup OP\cup CL\cup ST\cup END$; where *ORD* is the set of ordinary states, *OP* is the set of opening states, *CL* are closing states, *ST* are start states and *END* – end states. The opening and closing states are introduced to simulate the shuffle closure operator $\otimes$. The start and end states are used for the shuffle operator $\odot$.

Definition 2.2. (Construction of the automaton $A_E=(Q_E,\Sigma,q_E,f_E,T_E)$ for a shuffle expression E.) The construction is inductive on the structure of E. For $E=\lambda$, $E=\emptyset$, $E=a$, where $a\in\Sigma$, shuffle automata are defined as follows: $A_\lambda=(\{q_\lambda,f_\lambda\},\Sigma,q_\lambda,f_\lambda,\{(\lambda,q_\lambda,f_\lambda)\})$, $A_\emptyset=(\{q_\emptyset,f_\emptyset\},\Sigma,q_\emptyset,f_\emptyset,\emptyset)$, $A_a=(\{q_a,f_a\},\Sigma,q_a,f_a,\{(a,q_a,f_a)\})$, where all the states are ordinary ones, see Fig. 4. Suppose that for the expressions F and G we have already constructed shuffle automata $A_F=(Q_F,\Sigma,q_F,f_F,T_F)$ and $A_G=(Q_G,\Sigma,q_G,f_G,T_G)$ such that $Q_F\cap Q_G=\emptyset$.

(1) If $E=F+G$, then we take two new ordinary states $q_E,f_E\notin Q_F\cup Q_G$, and set $A_E=(Q_F\cup Q_G\cup\{q_E,f_E\},\Sigma,q_E,f_E,T_E)$ where $T_E=T_F\cup T_G\cup\{(\lambda,q_E,q_F),(\lambda,q_E,q_G),(\lambda,f_F,f_E),(\lambda,f_G,f_E)\}$.

(2) If $E=F\cdot G$, then $A_E=(Q_F\cup Q_G,\Sigma,q_F,f_G,T_F\cup T_G\cup\{(\lambda,f_F,q_G)\})$.

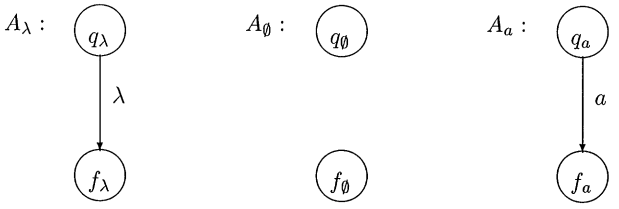

Fig. 4. Automata for λ, $\emptyset$, a.

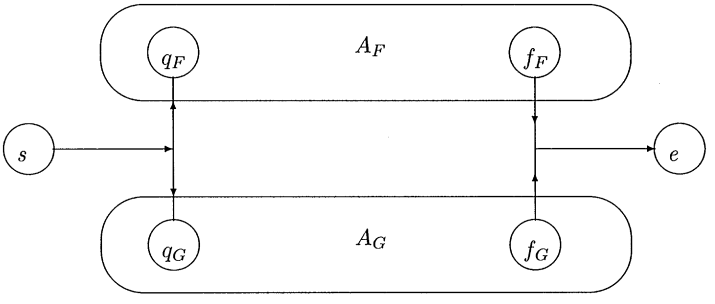

Fig. 5. Construction for shuffle.

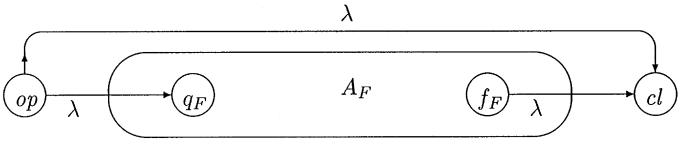

Fig. 6. Construction for shuffle closure.

(3) If $E = F^*$, then we introduce two new, ordinary states $q_E, f_E \notin Q_F$ and define $A_E = (Q_F \cup \{q_E, f_E\}, \Sigma, q_E, f_E, T_F \cup \{(\lambda, q_E, q_F), (\lambda, f_F, f_E), (\lambda, f_F, q_F), (\lambda, q_E, f_E)\})$.
(4) If $E = F \odot G$, then we introduce two new states: a start state $s \in ST$ and an end state $e \in END$, and define $A_E = (Q_F \cup Q_G \cup \{s, e\}, \Sigma, s, e, T_F \cup T_G \cup \{(\lambda, s, q_F, q_G), (\lambda, f_F, f_G, e)\})$, see Fig. 5.
(5) If $E = F^{\otimes}$, then we introduce two new states: an opening state $op \in OP$ and a closing state $cl \in CL$; and set $A_E = (Q_F \cup \{op, cl\}, \Sigma, op, cl, T_F \cup \{(\lambda, op, q_F), (\lambda, op, cl), (\lambda, f_F, cl)\})$, see Fig. 6.

In this paper we consider only automata constructed as shown above. Thus every shuffle automaton corresponds to a shuffle expression.

2.4. Distinguishable markers and computations

As we have already mentioned markers take part in the computation. Distinguishable markers are of the form $\alpha = n_1 \circ n_2 \cdots \circ n_d \in Names = \mathbb{N}^*$, where $\mathbb{N}$ is the set of natural

numbers and $\circ$ denotes concatenation. For every marker α, $Ft(\alpha)=n_1\circ\cdots\circ n_{d-1}$ denotes the father marker of α. We assume that there is a function Bp which for every marker α defines the birth place of the marker, $Bp(\alpha)\in OP\cup\{\perp\}$.

The *configuration* of the automaton $\Gamma: Names\to 2^{Q}$ describes the distribution of markers between the states of the automaton. $\Gamma(\alpha)$ denotes the set of states where the marker α is placed. Only for a finite number of markers α, $\Gamma(\alpha)\neq\emptyset$.

There is one main marker $\alpha=1$. When starting, the main marker is in the initial state q_0. It has no father, $Ft(1)=\lambda$, and no birth place, $Bp(1)=\perp$. Other markers are born while performing transitions of the form $t=(\lambda,op,q)$, with $op\in OP$ and $q\notin CL$. When a new marker α is born, the precondition for the transition t is that the marker $Ft(\alpha)$, the father of α, has one of its legs in op. The state op is the birth place of the new marker, $op=Bp(\alpha)$. In the next configuration α is placed in $\{q\}$. The marker α is called a son of the marker $Ft(\alpha)$.

Transitions of the kind $t=(\lambda,f,cl)$ with $cl\in CL$ and $f\notin OP$ kill markers. If the marker α is killed by t then the precondition for t is that α is placed in $\{f\}$, and in the next configuration α disappears.

Transitions of the form $t=(a,p,q)$ change the position of one leg of some marker. Transition t can be applied to a marker α with one leg in p, this leg is moved from p to q.

In a similar way one performs transitions of the form $t=(\lambda,s,p,q)$ with $s\in ST$, or $t=(\lambda,p,q,e)$ with $e\in END$. In the former case the marker takes its leg from s and puts one leg in p and another in q. In the latter case one leg is taken from p, one from q and one leg is put in e. If a transition is of the form $t=(\lambda,op,cl)$ with $op\in OP$ and $cl\in CL$, then one leg is moved from op to cl, but in this case the additional condition should be satisfied, namely that the marker to be moved has no children born in op.

Now we define more formally one step of the computation with distinguishable markers.

Definition 2.3. For any configurations Γ, Δ, a transition $t\in T_E$ and a marker $\alpha\in Names$, we write $\Gamma\overset{t,\alpha}{\Longrightarrow}\Delta$, if the following conditions are satisfied:

(1) If $t=(\lambda,op,q)$, $op\in OP$, $q\notin CL$, then $\Gamma(\alpha)=\emptyset$, $op\in\Gamma(Ft(\alpha))$, $op=Bp(\alpha)$, and $\Delta(\alpha)=\{q\}$.
(2) If $t=(\lambda,f,cl)$, $cl\in CL$, $f\notin OP$, then $\Gamma(\alpha)=\{f\}$ and $\Delta(\alpha)=\emptyset$.
(3) If $t=(\lambda,op,cl)$, $op\in OP$, $cl\in CL$, then $op\in\Gamma(\alpha)$, $\Delta(\alpha)=(\Gamma(\alpha)-\{op\})\cup\{cl\}$, and for every marker β, if $Bp(\beta)=op$ and $Ft(\beta)=\alpha$, then $\Gamma(\beta)=\emptyset$.
(4) If $t=(\lambda,s,p,q)$, $s\in ST$, then $V\subset\Gamma(\alpha)$ and $\Delta(\alpha)=(\Gamma(\alpha)-V)\cup W$, where $V=\{s\}$ and $W=\{p,q\}$.
(5) If $t=(\lambda,p,q,e)$, with $e\in END$, then exactly like in (4) with $V=\{p,q\}$ and $W=\{e\}$.
(6) If $t=(a,p,q)$, $p\notin OP$, $q\notin CL$, then exactly like in (4) with $V=\{p\}$ and $W=\{q\}$.

In all cases $\Delta(\gamma)=\Gamma(\gamma)$, for $\gamma\neq\alpha$.

In (1) the marker α is born, in (2) α is killed, and in (3)–(6) α is moved by the transition t. In the sequel we shall say that α is moved by the transition t in all cases (1)–(6). Note that if the marker α is born in the transition (λ, op, q) then $op = Bp(\alpha)$. We assume that every marker can be born only once.

Having defined $\stackrel{t,\alpha}{\Longrightarrow}$ for any transition $t \in T$, we extend this notion to $\stackrel{w,\alpha}{\Longrightarrow}$, where w is any sequence of transitions, $w = t_1 \cdots t_n$. We say that

$$\Gamma \stackrel{w,\alpha}{\Longrightarrow} \Delta$$

if there exist configurations $\Gamma = \Gamma_0, \ldots, \Gamma_n = \Delta$ and markers $\alpha_1, \alpha_2, \ldots, \alpha_n$ such that

- $\Gamma_{i-1} \stackrel{t_i,\alpha_i}{\Longrightarrow} \Gamma_i$, for every $1 \leqslant i \leqslant n$, and
- α is a prefix of each α_i.

The initial configuration *in* and the final configuration *fin* are defined as follows $in(1) = \{q_E\}$, $fin(1) = \{f_E\}$, and $in(\alpha) = fin(\alpha) = \emptyset$ for $\alpha \neq 1$.

The language accepted by the automaton A with distinguishable markers is

$$L_d(A) = \{label(w) \in \Sigma^* \mid in \stackrel{w,1}{\Longrightarrow} fin\}$$

where *label* is the natural homomorphism $label : T^* \to \Sigma^*$ defined by $label(t) = a$, for $t = (a, p, q)$ or $t = (a, p, q, r)$, where $p, q, r \in Q$, and $a \in \Sigma \cup \{\lambda\}$.

Example 2.4. The language generated by the expression $E = (a((bc)^{\otimes} \odot d^*))^{\otimes}$ is accepted by the automaton $A_E = (Q, \Sigma, op_1, cl_1, T)$, where $Q = ORD \cup OP \cup CL \cup ST \cup END$, $ORD = \{q_1, \ldots, q_{10}\}$, $OP = \{op_1, op_2\}$, $CL = \{cl_1, cl_2\}$, $ST = \{s\}$, $END = \{e\}$, $\Sigma = \{a, b, c, d\}$ and

$$\begin{aligned} T = \{&(\lambda_1, op_1, cl_1), (\lambda_2, op_2, cl_2), (\lambda_3, op_1, q_1), (a, q_1, q_2), (\lambda_4, q_2, s), (\lambda_5, s, op_2, q_7),\\ &(\lambda_6, op_2, q_3), (b, q_3, q_4), (\lambda_7, q_4, q_5), (c, q_5, q_6), (\lambda_8, q_6, cl_2), (\lambda_9, q_7, q_8),\\ &(d, q_8, q_9), (\lambda_{10}, q_9, q_8), (\lambda_{11}, q_9, q_{10}), (\lambda_{12}, q_{10}, cl_2, e), (\lambda_{13}, e, cl_1), (\lambda_{14}, q_7, q_{10})\} \end{aligned}$$

(see Fig. 7). In transitions labeled by the empty word we use λ_i, $i = 1, \ldots, 14$ instead of λ (assuming $\lambda_i \equiv \lambda$), to show more clearly the process of accepting a word by A_E.

The word $u = abddadc \in L(E)$ is accepted by the computation with the following sequence of labels:

$$\lambda_3 a \lambda_4 \lambda_5 \lambda_6 b \lambda_9 d \lambda_{10} d \lambda_3 a \lambda_4 \lambda_5 \lambda_2 \lambda_9 d \lambda_{11} \lambda_{12} \lambda_{13} \lambda_7 c \lambda_8 \lambda_2 \lambda_{11} \lambda_{12} \lambda_{13} \lambda_1.$$

The computation has length equal 28; there are four markers taking part in the computation: 1, $1 \circ 1$, $1 \circ 1 \circ 1$, and $1 \circ 2$. Let $\Gamma_0, \ldots, \Gamma_{28}$ be the sequence of the consecutive configurations of this computation, Γ_0 is the initial configuration, and Γ_{28} final. Below we show some of the configurations appearing in the computation.

$\Gamma_0(1) = \{op_1\}$;
$\Gamma_4(1) = \{op_1\}$ and $\Gamma_4(1 \circ 1) = \{op_2, q_7\}$;
$\Gamma_{10}(1) = \{op_1\}$; $\Gamma_{10}(1 \circ 1) = \{op_2, q_9\}$; $\Gamma_{10}(1 \circ 1 \circ 1) = \{q_4\}$;
$\Gamma_{16}(1) = \{op_1\}$; $\Gamma_{16}(1 \circ 1) = \{op_2, q_9\}$; $\Gamma_{16}(1 \circ 1 \circ 1) = \{q_4\}$; $\Gamma_{16}(1 \circ 2) = \{cl_2, q_8\}$;

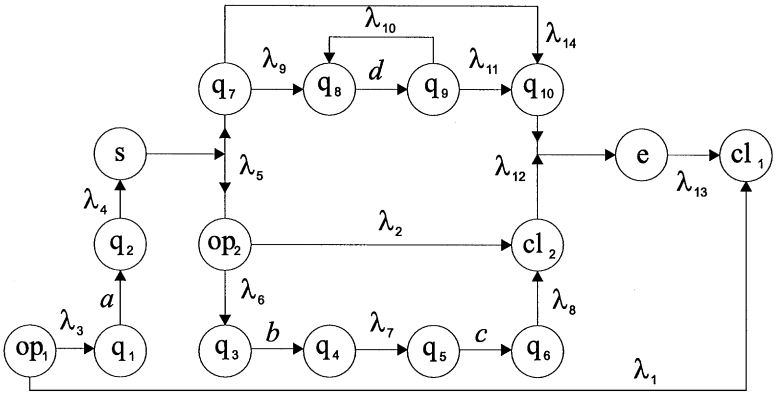

Fig. 7. Shuffle automaton for $(a((bc)^{\otimes} \odot d^*))^{\otimes}$.

$\Gamma_{20} = \Gamma_{10}$;
$\Gamma_{24}(1) = \{op_1\}$; $\Gamma_{24}(1 \circ 1) = \{cl_2, q_9\}$;
$\Gamma_{28}(1) = \{cl_1\}$.

3. Shuffle automata with distinguishable markers accept shuffle languages

Theorem 3.1. *For any shuffle expression E, the shuffle automaton A_E with distinguishable markers accepts exactly $L(E)$.*

Proof. In Section 2.3 we described the construction of the shuffle automaton A_E corresponding to the shuffle expression E. In *Part* 1, we shall show that for each word $u \in L(E)$ generated by E, there exists a computation with distinguishable markers accepting u. In *Part* 2 we show that each word accepted by A_E is generated by E.

Part 1

Inductively on the structure of E, we show that $L(E) \subset L_d(A_E)$.

For $E = \lambda$, $E = \emptyset$, $E = a$, where $a \in \Sigma$, the inclusion is obvious.

Let $A_F = (Q_F, \Sigma, q_F, f_F, T_F)$ and $A_G = (Q_G, \Sigma, q_G, f_G, T_G)$ be the shuffle automata constructed for shuffle expressions F and G. Suppose that $L(F) \subset L_d(A_F)$, $L(G) \subset L_d(A_G)$.

Below we prove that $L(E) \subset L(A_E)$ for each of the cases for E being $F + G, F \cdot G, F^*, F \odot G, F^{\otimes}$.

1. Let $E = F + G$ and suppose that $u \in L(F)$. Then there exists $z_u \in T_F^*$ such that $label(z_u) = u$ and $in \overset{z_u,1}{\Longrightarrow} fin$ in A_F. Let $z = (\lambda, q_E, q_F) \cdot z_u \cdot (\lambda, f_F, f_E)$. We have $in \overset{z,1}{\Longrightarrow} fin$ in A_E, and $label(z) = label(z_u) = u$, so $u \in L_d(A_E)$.
2. If $E = F \cdot G$ and $w \in L(F) \cdot L(G)$, then there exist $u \in L(F)$, $v \in L(G)$, such that $w = u \cdot v$. Thus there exist $z_u \in T_F^*$, $z_v \in T_G^*$, such that $label(z_u) = u$, $label(z_v) = v$, $in \overset{z_u,1}{\Longrightarrow} fin$ in A_F, and $in \overset{z_v,1}{\Longrightarrow} fin$ in A_G. We put $z_w = z_u \cdot t \cdot z_v$, where $t = (\lambda, f_F, q_G)$. We have $in \overset{z_w,1}{\Longrightarrow} fin$ in A_E, and $label(z_w) = w$, so $w \in L_d(A_E)$.

3. In case $E=F^*$, let $u\in(L(F))^*$, then $u=\lambda$, or there exist $u_1,\ldots,u_n\in L(F)$ and $u=u_1\cdots u_n$, for some $n>0$. In the former case $in \overset{t,1}{\Longrightarrow} fin$ in A_E, for $t=(\lambda,q_E,f_E)$. In the latter case, there exist $z_1,\ldots,z_n\in T_F^*$, such that $in \overset{z_i,1}{\Longrightarrow} fin$ in A_F and $label(z_i)=u_i$ for $i=1,\ldots,n$.
   Let $z_u=(\lambda,q_E,q_F)\cdot z_1\cdot(\lambda,f_F,q_F)\cdots z_{n-1}\cdot(\lambda,f_F,q_F)\cdot z_n\cdot(\lambda,f_F,f_E)$. Then $in \overset{z_u,1}{\Longrightarrow} fin$ in A_E, and $label(z_u)=u$.
4. If $E=F\odot G$ and $w\in L(F)\odot L(G)$, then there exist $u\in L(F)$ and $v\in L(G)$, such that $w\in u\odot v$. Thus for some $z_u\in T_F^*$ and $z_v\in T_G^*$, we have $in \overset{z_u,1}{\Longrightarrow} fin$ in A_F and $in \overset{z_v,1}{\Longrightarrow} fin$ in A_G, $label(z_u)=u$, and $label(z_v)=v$. Let $z_w=(\lambda,s,q_F,q_G)\cdot z\cdot(\lambda,f_F,f_G,e)$, where $z\in(T_F\cup T_G)^*$ is a word from $z_u\odot z_v$, for which $label(z)=w$. We have $in \overset{z_w,1}{\Longrightarrow} fin$ in A_E, and $label(z_w)=w$.
5. If $E=F^{\otimes}$ and $u\in(L(F))^{\otimes}$, then $u=\lambda$ or $u\in u_1\odot\cdots\odot u_n$, where $u_i\in L(F)$, for $i=1,\ldots,n$. In the former case $in \overset{t,1}{\Longrightarrow} fin$ in A_E, for $t=(\lambda,op,cl)$.
   In the latter case, for each i, there exist $z_i\in T_F^*$ such that $label(z_i)=u_i$, and $in \overset{z_i,1}{\Longrightarrow} fin$ in A_F. Let $m_1,\ldots,m_n$ be distinct natural numbers which do not appear in any markers in the above computations. It is obvious that also $in_i \overset{z_i,1\circ m_i}{\Longrightarrow} fin_i$ in A_F, where $in_i(1\circ m_i)=\{q_F\}$, $fin_i(1\circ m_i)=\{f_F\}$, and $in_i(\alpha)=\emptyset=fin_i(\alpha)$, for $\alpha\neq 1\circ m_i$.
   Let

   $$z_u=x^n z y^n t,$$

   where $x=(\lambda,op,q_F)$, $y=(\lambda,f_F,cl)$, and z is any word from $z_1\odot z_2\cdots\odot z_n$, for which $label(z)=u$.
   Let IN,FIN be the initial and final configuration of A_E. We have

   $$IN \overset{x,1\circ m_1}{\Longrightarrow} \Gamma_1 \overset{x,1\circ m_2}{\Longrightarrow} \cdots \Gamma_{n-1} \overset{x,1\circ m_n}{\Longrightarrow} \Gamma_n,$$

   where $\Gamma_n(1)=\{op\}$, and $\Gamma_n(1\circ m_i)=\{q_F\}$, for $i=1,\ldots,n$. Furthermore, we have

   $$\Gamma_n \overset{z,1}{\Longrightarrow} \Delta_1,$$

   where Δ_1 is defined in the following way: $\Delta_1(1)=\{op\}$, and $\Delta_1(1\circ m_i)=\{f_F\}$, for $1\leqslant i\leqslant n$. In this computation each marker $1\circ m_i$ is moved by z_i and the markers do not disturb each other in their transitions. It is easy to see that transitions $y^n t$ lead from Δ_1 to the final configuration in A_E.

Part 2

Here we prove that each word accepted by A_E is in the language generated by E. As it happens often with the inductive proofs, since the automata are nested in one another, we will need to prove an invariant, for an expression E. Assume $A_E=(Q,\Sigma,q_E,f_E,T_E)$ is the automaton constructed for E. Possibly A_E is a part of some bigger shuffle automaton. The lemma below describes the following intuition. If a marker is moved from the initial state to the final one, then the transitions which move the marker correspond, through the labelling function, to a word accepted by the automaton.

To formulate the lemma we introduce the notion of a generalized computation. We write

$$\Gamma \overset{w,\alpha}{\rightsquigarrow} \Delta$$

for a marker α and a word $w = t_1 \cdots t_n$ if there exist two sequences of configurations $\Gamma_0, \Gamma_1, \ldots, \Gamma_n$ and $\Delta_0, \Delta_1, \ldots, \Delta_n$ satisfying the following conditions, for $0 \leqslant i \leqslant n$:

1. $\Gamma_i(\beta) = \Delta_i(\beta)$, if $\beta = \alpha$ or $|\beta| \neq |\alpha|$,
2. if $\Delta_i(\delta) \neq \Gamma_i(\delta)$, then $|\delta| = |\alpha|$, and either $\Delta_i(\delta) = \{q_E\}$ and $\Gamma_i(\delta) = \emptyset$, or $\Delta_i(\delta) = \emptyset$ and $\Gamma_i(\delta) = \{f_E\}$,
3. $\Delta_{i-1} \overset{t_i,\beta_i}{\Longrightarrow} \Gamma_i$ for some marker β_i, and $i \geqslant 1$,
4. $\Gamma = \Gamma_0$ and $\Delta_n = \Delta$.

The generalized computation is just like the normal computation a sequence of transitions, but between transitions some markers $\delta \neq \alpha$ can be added to the initial state q_E or taken from the final state f_E. These additional markers are of the same length as α, and can take part in the computation.

Lemma 3.2. *Let $A_E = (Q, \Sigma, q_E, f_E, T_E)$ be the automaton constructed for an expression E, and Γ and Δ two reachable configurations of A_E, $w \in T_E^*$, and α is a marker. Assume that $\Gamma \overset{w,\alpha}{\rightsquigarrow} \Delta$, $\Gamma(\alpha) = \{q_E\}$, $\Delta(\alpha) = \{f_E\}$, and $\Gamma(\beta) = \emptyset$, for all markers β of the form $\beta = \alpha \circ \delta$ with $\delta \neq \lambda$.*

Then

(i) *$\Delta(\beta) = \emptyset$, for each $\beta = \alpha \circ \delta$ with $\delta \neq \lambda$,*

(ii) *there exist $u, v \in T_E^*$ such that $w \in u \odot v$ and u contain all those transitions from w which move any marker of the kind $\alpha \circ \delta$ (i.e. α or any of its successors),*

(iii) *label(u) is generated by E, and in $\overset{u,1}{\Longrightarrow}$ fin in A_E,*

(iv) *$\Gamma' \overset{v,\alpha}{\rightsquigarrow} \Delta'$, where $\Gamma'(\beta) = \Gamma(\beta)$, $\Delta'(\beta) = \Delta(\beta)$ for $\beta \neq \alpha$, and $\Gamma'(\alpha) = \Delta'(\alpha) = \emptyset$.*

To finish the proof of *Part* 2, it is enough to use the above lemma to a normal computation $\Gamma \overset{w,\alpha}{\Longrightarrow} \Delta$ with $\Gamma = in$, $\Delta = fin$, $\alpha = 1$. Then $u = w$, $v = \lambda$, and $label(w) = label(u) \in L(E)$.

Proof of Lemma 3.2. For $E = \emptyset$, the lemma is trivially true since $T_E = \emptyset$.

For $E = \lambda$, $E = a$, we have $T_E = \{t\}$ and $w = t^n$, for some $n \geqslant 1$. Thus $u = t$, $v = t^{n-1}$ and the lemma is true.

Now suppose that for two expressions F, G and automata $A_F = (Q_F, \Sigma, q_F, f_F, T_F)$ and $A_G = (Q_G, \Sigma, q_G, f_G, T_G)$, the lemma is true; we are going to show it for A_E in each of the cases E being $F + G$, $F \cdot G$, F^*, $F \odot G$, $F^{\otimes}$.

1. $E = F + G$.

   Let $t_1 = (\lambda, q_E, q_F)$, $t_2 = (\lambda, q_E, q_G)$, $t_3 = (\lambda, f_F, f_E)$, and $t_4 = (\lambda, f_G, f_E)$. The marker α is either moved by t_1 or t_2. We consider the first case; the proof for the other is similar. The computation can be decomposed into

$$\Gamma \overset{x,\alpha}{\rightsquigarrow} \Gamma_1 \overset{t_1,\alpha}{\Longrightarrow} \Gamma_2 \overset{y,\alpha}{\rightsquigarrow} \Gamma_3 \overset{t_3,\alpha}{\Longrightarrow} \Gamma_4 \overset{z,\alpha}{\rightsquigarrow} \Delta,$$

$w = xt_1yt_3z$, and α is not moved by any transition from x and z, and y is of the form $y \in y_F \odot y_G$, where $y_F \in (T_F \cup \{t_1, t_3\})^*$, $y_G \in (T_G \cup \{t_2, t_4\})^*$ and α is not moved by any transition from y_G. We have

$$\Gamma_2(\alpha) = \{q_F\}, \quad \Gamma_3(\alpha) = \{f_F\}$$

and we use the inductive assumption on the computation $\Gamma_2 \overset{y,\alpha}{\rightsquigarrow} \Gamma_3$ restricted to A_F, thus $\Gamma_2 \overset{y_F,\alpha}{\rightsquigarrow} \Gamma_3$. There exist u_y, v_y such that $y_F \in u_y \odot v_y$, α is moved by u_y and not by v_y, besides $label(u_y) \in L(F)$, and $in \overset{u_y}{\Longrightarrow} fin$ in A_F.
We have

$$w = xt_1yt_3z \in xt_1(y_G \odot u_y \odot v_y)t_3z \subset x(y_G \odot v_y)z \odot t_1u_yt_3$$

and

$$w \in u \odot v$$

for $u = t_1u_yt_3$ and some $v \in x(y_G \odot v_y)z$. α does not appear in transitions from v, but appears in transitions from u. Obviously $label(u) = label(u_y) \in L(E)$, $in \overset{u}{\Longrightarrow} fin$ in A_E, and $\Gamma' \overset{v,\alpha}{\rightsquigarrow} \Delta'$ which proves (ii)–(iv). Besides, (i) follows from the inductive assumption on computations in A_F and the fact that no marker with prefix α appears in states from A_G.

2. $E = F \cdot G$.
The computation can be decomposed into

$$\Gamma \overset{x,\alpha}{\rightsquigarrow} \Gamma_1 \overset{t,\alpha}{\Longrightarrow} \Gamma_2 \overset{y,\alpha}{\rightsquigarrow} \Delta$$

with $\Gamma_1(\alpha) = \{f_F\}$, $t = (\lambda, f_F, q_G)$, and $w = xty$.
Furthermore x can be decomposed into $x \in x_F \odot x_G$, where $x_F \in (T_F)^*$ and $x_G \in (T_G \cup \{t\})^*$, and the marker α is not moved by x_G. From the inductive assumption on the computation $\Gamma \overset{x,\alpha}{\rightsquigarrow} \Gamma_1$ considered in A_F only, we have that there exist $u_F, v_F \in T_F^*$ such that $x_F \in u_F \odot v_F$, the marker α is only moved by u_F, $label(u_F) \in L(F)$, and $in \overset{u_F}{\Longrightarrow} fin$ in A_F.
Furthermore, y can be decomposed into $y \in y_F \odot y_G$, where $y_F \in (T_F \cup \{t\})^*$, $y_G \in T_G^*$, and α is not moved by y_F.
Again, y_G can be decomposed into $y_G \in u_G \odot v_G$, where α is only moved by u_G, $label(u_G) \in L(G)$, and $in \overset{u_G}{\Longrightarrow} fin$ in A_G.
We have

$$\begin{aligned} w = xty \in (x_F \odot x_G)t(y_F \odot y_G) &\subset (u_F \odot v_F \odot x_G)t(u_G \odot v_G \odot y_F) \\ &\subset u_Ftu_G \odot ((v_F \odot x_G)(v_G \odot y_F)). \end{aligned}$$

Hence, there exists $v \in (v_F \odot x_G)(v_G \odot y_F)$ such that $w \in u_Ftu_G \odot v$. Let β be any marker with a proper prefix α. From the inductive assumption it follows that $\Gamma_1(\beta) = \emptyset$, thus $\Delta(\beta)$ does not contain states from Q_F. From the inductive assumption on

the computation $\Gamma_2 \overset{y,\alpha}{\rightsquigarrow} \Delta$ in A_G, it follows that $\Delta(\beta)$ does not contain states from Q_G, which settles (i). Obviously, $label(u_F t u_G) \in L(E)$.

3. $E = F^*$.
Let $x = (\lambda, q_E, q_F)$, $y = (\lambda, f_F, f_E)$, $z = (\lambda, f_F, q_F)$, and $t = (\lambda, q_E, f_E)$. The marker α is either moved by t or by x. The first case $u = t$ is obvious.
In the second case the computation can be decomposed into

$$\Gamma \overset{w_0,\alpha}{\rightsquigarrow} \Gamma_0 \overset{x,\alpha}{\Longrightarrow} \Delta_1 \overset{w_1,\alpha}{\rightsquigarrow} \Gamma_1 \overset{z,\alpha}{\Longrightarrow} \Delta_2 \overset{w_2,\alpha}{\rightsquigarrow} \Gamma_2 \cdots \Delta_n \overset{w_n,\alpha}{\rightsquigarrow} \Gamma_n \overset{t,\alpha}{\Longrightarrow} \Gamma_{n+1} \overset{w_{n+1},\alpha}{\rightsquigarrow} \Delta$$

with $\Delta_i(\alpha) = \{q_F\}$ and $\Gamma_i(\alpha) = \{f_F\}$, for every $1 \leqslant i \leqslant n$. Inductively on i, one can prove that, $\Delta_i(\beta) = \emptyset$, for each $\beta = \alpha \circ \delta$, and that w_i can be decomposed into $w_i \in u_i \odot v_i$, where $u_i \in T_F^*$ and $v_i \in T_E{}^*$, and α is only moved by u_i, $label(u_i) \in L(F)$, and $in \overset{u_i}{\Longrightarrow} fin$ in A_F.
Thus

$$w \in x u_1 z u_2 z \cdots u_n y \odot w_0 v_1 v_2 \cdots v_n w_{n+1}.$$

4. $E = F \odot G$.
Let $t_1 = (\lambda, s, q_F, q_G)$ and $t_2 = (\lambda, f_F, f_G, e)$. The computation is decomposed into

$$\Gamma \overset{x,\alpha}{\rightsquigarrow} \Gamma_1 \overset{t_1,\alpha}{\Longrightarrow} \Gamma_2 \overset{y,\alpha}{\rightsquigarrow} \Gamma_3 \overset{t_2,\alpha}{\Longrightarrow} \Gamma_4 \overset{z,\alpha}{\rightsquigarrow} \Delta,$$

$w = x t_1 y t_2 z$, and α is not moved by x and z.
We have $\Gamma_2(\alpha) = \{q_F, q_G\}$ and $\Gamma_3(\alpha) = \{f_F, f_G\}$. Furthermore y is decomposed into $y \in y_F \odot y_G \odot y_t$, with $y_F \in T_F^*$, $y_G \in T_G^*$, and $y_t \in \{t_1, t_2\}^*$. Now we use the inductive assumptions for the computations:

$$\Gamma_2 \overset{y,\alpha}{\rightsquigarrow} \Gamma_3$$

restricted to A_F and A_G and we obtain decomposition of y_F and y_G into $y_F \in u_F \odot v_F$ and $y_G \in u_G \odot v_G$, where α is only moved by u_F and u_G, $label(u_F) \in L(F)$, $in \overset{u_F}{\Longrightarrow} fin$ in A_F, $label(u_G) \in L(G)$, and $in \overset{u_G}{\Longrightarrow} fin$ in A_G. Thus, w can be decomposed into $w \in u \odot v$ with some $u \in t_1(u_F \odot u_G)t_2$ and $v \in x(v_F \odot v_G \odot y_t)z$. α is only moved by u, and $label(u) \in L(F) \odot L(G)$.

5. $E = F^{\otimes}$.
Let $t_1 = (\lambda, op, q_F)$, $t_2 = (\lambda, f_F, cl)$, and $t_3 = (\lambda, op, cl)$. Let $m \geqslant 0$ stand for the number of those occurrences of the transition t_1 in w, which move the sons of α. We shall prove the lemma by induction on m.
If $m = 0$, then α is only moved by t_3 and the case is obvious. If $m > 0$, then the computation $\Gamma \overset{w,\alpha}{\rightsquigarrow} \Delta$ can be decomposed into

$$\Gamma \overset{x,\alpha}{\rightsquigarrow} \Gamma_1 \overset{t_1,\alpha\circ k}{\Longrightarrow} \Gamma_2 \overset{y,\alpha}{\rightsquigarrow} \Gamma_3 \overset{t_2,\alpha\circ k}{\Longrightarrow} \Gamma_4 \overset{z,\alpha}{\rightsquigarrow} \Delta,$$

where $\alpha \circ k$ is the first son of α appearing in the computation.
We also have

$$\Gamma_2 \overset{y,\alpha\circ k}{\rightsquigarrow} \Gamma_3$$

and y can be decomposed into $y \in u_y \odot v_y$, with $in \overset{u_y,\alpha\circ k}{\Longrightarrow} fin$ in A_F, $u_y \in L(F)$, and $\Gamma_2' \overset{v_y,\alpha\circ k}{\rightsquigarrow} \Gamma_3'$, where Γ_2' is Γ_2 without $\alpha \circ k$, and Γ_3' is Γ_3 without $\alpha \circ k$.
We have obtained a computation

$$\Gamma \overset{x,\alpha}{\rightsquigarrow} \Gamma_2' \overset{v_y,\alpha}{\rightsquigarrow} \Gamma_4 \overset{z,\alpha}{\rightsquigarrow} \Delta$$

with $m-1$ occurrences of the transition t_1.
Let $w' = xv_yz$. By induction, $w' \in u' \odot v'$ and α is only moved by u' and $label(u') \in L(F^{\otimes})$. Hence, there exists $u \in u' \odot u_y$ such that $w \in u \odot v_y$, α is only moved by u, and $label(u) \in L(F^{\otimes})$. □

4. Computation with nondistinguishable markers

4.1. Types of markers

In this section we introduce such computations of shuffle automata which not only correspond to shuffle languages but are also economical in terms of computational complexity. For each shuffle automaton we shall define a finite set of types of markers and a computation with nondistinguishable markers, where markers of the same type are not distinguishable and we only count and remember the number of markers of each type. Thus it will be possible to simulate the computation in logarithmic space. At every moment each marker is of some type and after each step some markers will change their types. In the next section we shall show that the two kinds of computations: with distinguishable and nondistinguishable markers are equivalent in the sense that they accept the same language.

By a type of a marker we mean a sequence $X = T_1 \circ T_2 \circ \cdots \circ T_d$, where for each i, $T_i = (P_i, R_i, q_i)$, $P_i \subset Q$ and $P_i \neq \emptyset$, $R_i \subset P_i \cap OP$ and $q_i \in OP \cup \{\perp\}$. The number d is bounded by the maximal number of nested opening states in the automaton. In the sequel, we shall use the following notations:

- $P(X) = P_d$ is the set of states, where the legs of the marker are placed.
- $Bp(X) = q_d$ is the birth place of the marker. This is to remember where the marker was born. The birth place of the marker will not be changed during the computation.
- $R(X) = R_d$ is the set of "reserved" legs, these are states where the children of the marker will be born,
- $FT(X) = T_1 \circ T_2 \circ \cdots \circ T_{d-1}$ is the father type.

From our further construction it will follow that if there are some markers of the type X, then there is exactly one marker, of the type $Z = FT(X)$, this marker will be the father of all markers of type X.

As we have said we shall not distinguish markers of the same type, and shall only count the number of markers of each type. Thus, we shall not distinguish markers which stand in the same set of states, and have the same father and the same set of reserved legs. Note that from the construction of the shuffle automaton it follows

that the birth place of the type is uniquely determined by its position. It is added for technical reasons.

TYPES will denote the set of all possible types. The type $X_0=(\{q_0\},\emptyset,\perp)$ or $X_0=(\{q_0\},\{q_0\},\perp)$ is a starting type, and the type $X_f=(\{q_f\},\emptyset,\perp)$ is the final type, where q_0 (q_f) is the initial (final) state of the automaton.

A configuration C is a function $C: TYPES \to \mathbb{N}$. For any type X, $C(X)$ denotes how many markers are of the type X at the moment. We assume that there is at most one marker in any type with reserved legs. In other words

(R.1) If $R(X)\neq\emptyset$, then $C(X)\leqslant 1$.

This condition will ensure that if there are some markers of the type X, then there will be exactly one marker, their father, of the type $FT(X)$.

4.2. Computation with nondistinguishable markers

The general idea of the computation with nondistinguishable markers is the following. If we perform one step of the computation $C \Rightarrow^{t,X} D$, for some configurations C, D, a transition t and a type X, then we choose any marker from the type X and change its type according to the rules described by Definition 2.3. But there are some details we should explain first.

- Both configurations C and D satisfy (R.1).
- The precondition for the transition of the form $t=(\lambda,op,q)$, with $q\notin CL$, is that there is exactly one marker of the type $FT(X)$ and $op\in R(FT(X))$. During this transition a new marker is born and it is of the type $X=FT(X)\circ(\{q\},\emptyset,op)$ or $X=FT(X)\circ(\{q\},\{q\},op)$. The state $op=Bp(X)$ is the birth place of the new marker.
- If during the transition t the marker moves one of its legs to a state $q\in OP$, then q can be reserved or not in the new type Z of the marker.
- If the marker changes its type from X to Z, then also all its successors change their types from $X\circ Y$ to $Z\circ Y$, for every $Y\neq\lambda$. This is to ensure that, during the whole computation, if a marker is of a type X, then its father is of the type $FT(X)$.
- If the transition is of the kind $t=(\lambda,op,cl)$, then the precondition for t is that there are no markers of any type Y such that $FT(Y)=X$. Note that if $C(Y)>0$ then, by (R.1), $C(X)=1$ and the only marker from X cannot be moved by t, because it has sons.

Now we shall describe more formally the computations of the automaton with nondistinguishable markers. We start with the definition of one step of the computation.

Definition 4.1. For any configurations C, D, a transition $t\in T$ and a type $X\in TYPES$, $C\Rightarrow^{t,X}D$ is possible if the following conditions are satisfied: C and D satisfy (R.1) and

(1) If $t=(\lambda,op,q)$, $op\in OP$, $q\notin CL$, then $P(X)=\{q\}$, $R(X)\subset\{q\}\cap OP$, $op=Bp(X)$, $op\in R(FT(X))$, $C(FT(X))=1$, $D(X)=C(X)+1$.

(2) If $t=(\lambda, f, cl)$, $cl\in CL$, $f\notin OP$, then $C(X)>0$, $\{f\}=P(X)$, $D(X)=C(X)-1$.
(3) If $t=(\lambda, op, cl)$, $op\in OP$, $cl\in CL$, then
- $C(X)>0$, $op\in P(X)$, $D(X)=C(X)-1$,
- for every type Y, if $FT(Y)=X$, $P(Y)\neq\emptyset$ and $Bp(Y)=op$, then $C(Y)=0$,
- $D(Z)=C(Z)+1$, where Z is defined as $P(Z)=(P(X)-\{op\})\cup\{cl\}$, $R(Z)=R(X)-\{op\}$, $FT(Z)=FT(X)$, and $Bp(Z)=Bp(X)$,
- for every type of the kind $X\circ Y$ with $Y\neq\lambda$, $D(Z\circ Y)=C(X\circ Y)$ and $D(X\circ Y)=0$.

(4) If $t=(\lambda, s, p, q)$ with $s\in START$, then we set $V=\{s\}$ and $W=\{p,q\}$ and
- $C(X)>0$, $V\subset P(X)$, $D(X)=C(X)-1$,
- there is exactly one type Z satisfying $D(Z)=C(Z)+1$, $P(Z)=(P(X)-V)\cup W$, $R(Z)=R(X)\cup U$ where $U\subset W\cap OP$, $FT(Z)=FT(X)$, and $Bp(Z)=Bp(X)$,
- for every type of the kind $X\circ Y$ with $Y\neq\lambda$, $D(Z\circ Y)=C(X\circ Y)$ and $D(X\circ Y)=0$.

(5) If $t=(\lambda, p, q, e)$, $e\in END$, then exactly like in (4) with $V=\{p,q\}$ and $W=\{e\}$.
(6) If $t=(\sigma, p, q)$, $p\notin OP$, $q\notin CL$, then exactly like in (4) with $V=\{p\}$ and $W=\{q\}$.

No other changes. $D(Y)=C(Y)$ if $Y\neq X$ in cases (1) and (2), or if Y has neither a prefix equal to X nor to Z in cases (3)–(6).

A word $u\in\Sigma^*$ is accepted by the automaton A_E with nondistinguishable markers (it belongs to the language $L(A_E)$ then) if there exists an accepting computation for u, or, in other words, if there exists a sequence of configurations: $C_0, C_1, \ldots, C_m$ and a sequence of transitions: $w=t_1t_2\cdots t_m$ such that:
- for every $1\leqslant i\leqslant m$, $C_{i-1}\Rightarrow^{t_i,X_i}C_i$, for some type X_i.
- $C_0=in$ is the initial configuration defined by: $in(X_0)=1$, and $in(X)=0$ for $X\neq X_0$ (where X_0 is a starting type).
- $C_m=fin$ is the final configuration defined by: $fin(X_f)=1$, and $fin(X)=0$ for $X\neq X_f$ (where X_f is the final type).
- $u=label(w)$.

Example 4.2. Consider the shuffle automaton A_E for the expression $E=(a((bc)^{\otimes}\odot d^*))^{\otimes}$. In Example 2.4 we have shown an accepting computation with distinguishable markers for the word $u=abddadc\in L(E)$.

The word u is also accepted by a computation with nondistinguishable markers with the same sequence of labels:

$$\lambda_3 a\lambda_4\lambda_5\lambda_6 b\lambda_9 d\lambda_{10} d\lambda_3 a\lambda_4\lambda_5\lambda_2\lambda_9 d\lambda_{11}\lambda_{12}\lambda_{13}\lambda_7 c\lambda_8\lambda_2\lambda_{11}\lambda_{12}\lambda_{13}\lambda_1.$$

Let $C_0,\ldots,C_{28}$ be the sequence of the consecutive configurations of this computation, C_0 is the initial configuration, and C_{28} final. Below we show some of the types and configurations appearing in the computation.

$$T_1=(\{op_1\},\{op_1\},\perp),\ T_2=T_1\circ(\{op_2,q_7\},\{op_2\},op_1),\ T_3=T_1\circ(\{op_2,q_9\},\{op_2\},$$

$op_1)$, $T_4 = T_3 \circ (\{q_4\}, \emptyset, op_2)$, $T_5 = T_1 \circ (\{cl_2, q_8\}, \emptyset, op_1)$, $T_6 = T_1 \circ (\{cl_2, q_9\}, \emptyset, op_1)$, $T_7 = (\{cl_1\}, \emptyset, \perp)$.

$C_0(T_1) = 1$; $C_4(T_1) = 1$ and $C_4(T_2) = 1$; $C_{10}(T_1) = 1, C_{10}(T_3) = 1$ and $C_{10}(T_4) = 1$;

$C_{16}(T_1) = 1$, $C_{16}(T_3) = 1$, $C_{16}(T_4) = 1$ and $C_{16}(T_5) = 1$;

$C_{20} = C_{10}$; $C_{24}(T_1) = 1$ and $C_{24}(T_6) = 1$; $C_{28}(T_7) = 1$.

5. Equivalence of the two kinds of computation

In this section we present the proof that computations with distinguishable and nondistinguishable markers are equivalent. It is quite easy to change a computation with nondistinguishable marker into a computation with distinguishable markers. At the beginning we name the main marker $\alpha = 1$. Afterwards, if a new marker of a type X is born by a transition (λ, op, q), then we name the new marker $\alpha = \beta \circ i$, where β is the name of the only marker from $FT(X)$ and i is the number of the derivation step. We shall present the details in the proof of Theorem 5.3.

The simulation in the other direction is much more complicated. The main idea is that we shall define types of markers taking part in the computation with distinguishable markers, and we shall count markers of each type Y. But before we can do this we should first rearrange the computation with distinguishable markers in such a way that the condition (R.1) will be satisfied. This is to ensure that for every type X, if there are some markers of the type X, there will be exactly one marker of the father type $FT(X)$.

Consider a computation with distinguishable markers $in = \Gamma_0 \stackrel{t_1,\alpha_1}{\Longrightarrow} \Gamma_1 \cdots \Gamma_{n-1} \stackrel{t_n,\alpha_n}{\Longrightarrow} \Gamma_n$. We define types of markers taking part in this computation. $T_i(\alpha)$ will stand for the type of the marker α in Γ_i. For each $q \in \Gamma_i(\alpha) \cap OP$, let $s(i,q,\alpha)$ and $e(i,q,\alpha)$ be two numbers satisfying:

- $s(i,q,\alpha) \leqslant i < e(i,q,\alpha)$,
- $q \in \Gamma_j(\alpha)$, for all $s(i,q,\alpha) \leqslant j < e(i,q,\alpha)$,
- $q \notin \Gamma_j(\alpha)$, for $j = s(i,q,\alpha) - 1$ or $j = e(i,q,\alpha)$.

Thus $s(i,q,\alpha)$ is the moment when α puts its leg in q before i, and $e(i,q,\alpha)$ is the moment when α takes the leg from q after i.

To define the types $T_i(\alpha)$ we proceed as follows. If $\Gamma_i(\alpha) = \emptyset$, then $T_i(\alpha) = \lambda$. Otherwise, we use induction on the length of α. We start with the main marker $\alpha = 1$, and put $T_i(\alpha) = (\Gamma_i(\alpha), R, \perp)$, where

$$R = R(T_i(\alpha)) = \{op \in \Gamma_i(\alpha) \cap OP \mid \text{there exists a marker } \beta,\ Ft(\beta) = \alpha,\ Bp(\beta) = op, \text{ and } \Gamma_j(\beta) \neq \emptyset, \text{ for some } s(i,op,\alpha) < j < e(i,op,\alpha)\}.$$

Thus, if there is a son β of α born in op during the period from $s(i,op,\alpha)$ till $e(i,op,\alpha)$, then the state op is reserved in $\Gamma_i(\alpha)$.

If $|\alpha|>1$, then $T_i(Ft(\alpha))$ is already defined and we set $T_i(\alpha)=T_i(Ft(\alpha))\circ(\Gamma_i(\alpha),R,Bp(\alpha))$, where R is defined as above.

From the above definition we have that for every marker α, $T_i(\alpha)\in TYPES$ and

(M.1) If $\Gamma_i(\alpha)=\emptyset$, then $T_i(\alpha)=\lambda$.

(M.2) If $\Gamma_i(\alpha)\neq\emptyset$, then $\Gamma_i(\alpha)=P(T_i(\alpha))$.

(M.3) If $\Gamma_i(\alpha)\neq\emptyset$ then $FT(T_i(\alpha))=T_i(Ft(\alpha))$ (we assume $T_i(\lambda)=\lambda$).

(M.4) If $\Gamma_i(\alpha)\neq\emptyset$ and $|\alpha|>1$, then $Bp(\alpha)\in R(T_i(Ft(\alpha)))$.

From (M.3) it follows that, if $\Gamma_i(\alpha)\neq\emptyset$, then

(M.3.1) If $T_i(\alpha)=T_i(\beta)$ then $|\alpha|=|\beta|$

(M.3.2) If $\alpha=\gamma\circ\delta$, then $T_i(\alpha)=T_i(\gamma)\circ Y$, with $|Y|=|\delta|$.

(M.3.3) If $T_i(\alpha)=X\circ Y$, then there exist γ and δ such that $\alpha=\gamma\circ\delta$, $T_i(\gamma)=X$, and $|Y|=|\delta|$.

As we have said, the main idea how to change the computation with distinguishable markers into a computation with nondistinguishable markers will be in counting markers of each type Y and setting $C_i(Y)=|T_i^{-1}(Y)|=|\{\alpha\mid T_i(\alpha)=Y\}|$. But before we can do this we should first rearrange the computation with distinguishable markers in such a way that condition (R.1) will be satisfied.

Now we shall rearrange the computation in such a way that there will be no two markers which meet each other in any type with reserved legs. We shall say that markers α and β meet in the set $R(T_i(\alpha))$ in a configuration Γ_i if:

(R.2) $\quad T_i(\alpha)=T_i(\beta)\quad$ and $\quad R(T_i(\alpha))=R(T_i(\beta))\neq\emptyset$.

Thus we should first get rid of meetings.

Lemma 5.1. *For any computation with distinguishable markers*:

$$in=\Gamma_0\overset{t_1,\alpha_1}{\Longrightarrow}\cdots\overset{t_m,\alpha_m}{\Longrightarrow}\Gamma_m=fin,$$

there exists a computation with the same sequence of transitions satisfying the following condition:

(R.3) *if* $T_i(\alpha)=T_i(\beta)$ *and* $R(T_i(\alpha))\neq\emptyset$, *then* $\alpha=\beta$.

Proof. To simplify the proof we assume that the marker γ born in the step $\overset{t_i,\gamma}{\Longrightarrow}$ has the name $\gamma=\alpha\circ i$, for some α. Thus, if $\gamma_1=\delta_1\circ m_1$, $\gamma_2=\delta_2\circ m_2$, and $m_1=m_2$, then $\gamma_1=\gamma_2$.

We shall clear away the meetings of markers from the computation step by step. In each step we shall deal with one meeting. We shall consider meetings in some specific order: first meetings of shorter markers. Note that, by (M.3.1), if two markers α and β meet each other, then they are of the same length, $|\alpha|=|\beta|$. Hence, if two pairs of markers meet: α with β and γ with δ and $|\alpha|=|\beta|<|\gamma|=|\delta|$, then we shall first deal with the meeting of α and β. Among the markers of the same length we shall first deal with meetings in a larger set of reserved states. More precisely,

if α and β meet in a configuration Γ_i and γ and δ meet in a configuration Γ_j and $R(T_i(\alpha))=R(T_i(\beta))\subsetneq R(T_j(\gamma))=R(T_j(\delta))$, then we first deal with the meeting of γ and δ.

There are no meetings of markers of length 1, because there is only one, main, marker of length 1. Suppose that two markers α and β meet in a configuration Γ_i, satisfying (R.2). We show how to obtain the new computation $\Delta_0 \stackrel{t_1,\beta_1}{\Longrightarrow} \Delta_1 \cdots \stackrel{t_m,\beta_m}{\Longrightarrow} \Delta_m$ with the same sequence of transitions. In the new computation there will be at least one meeting less in the set $R(T_i(\alpha))$. It is possible that there are new meetings in some smaller set of reserved states or of longer markers. It is obvious that after some number of clearings there should be no meetings of markers.

Consider first the situation when there exists $q\in R(T_i(\alpha))$ such that:

- $s(i,q,\alpha)<s(i,q,\beta)\leqslant i<e(i,q,\beta)<e(i,q,\alpha)$ or
- $s(i,q,\beta)<s(i,q,\alpha)\leqslant i<e(i,q,\alpha)<e(i,q,\beta)$.

We shall only deal with the former case (the latter is similar). In this case the marker α stays longer in q than β, i.e α arrives in q before β and leaves q after β does. We make the following changes:

(Ch. 1) For every marker of the form $\gamma=\beta\circ m\circ\delta$ (with $m\in\mathbb{N}$ and $\delta\in Names$), if $Bp(\beta\circ m)=q$ and $\Gamma_j(\gamma)\neq\emptyset$ for some j, $s(i,q,\beta)\leqslant j<e(i,q,\beta)$, then we change the marker γ for the marker $\alpha\circ m\circ\delta$. In order to do this, we:

- set $\Delta_r(\alpha\circ m\circ\delta)=\Gamma_r(\beta\circ m\circ\delta)$ and $\Delta_r(\beta\circ m\circ\delta)=\emptyset$, for every r, and
- replace each transition of the form $\stackrel{t,\beta\circ m\circ\delta}{\Longrightarrow}$ by $\stackrel{t,\alpha\circ m\circ\delta}{\Longrightarrow}$.

This means that all children of the marker β born in q between $s(i,q,\beta)$ and $e(i,q,\beta)$ become children of α (and all successors of these children become successors of α). After these changes $R(T_i'(\beta))=R(T_i(\beta))-\{q\}$, where $T_i'(\beta)$ denotes the type of β in the configuration Δ_i. Hence, in the new computation, markers α and β do not meet in configuration Δ_i. It is possible that the marker β meets some new markers but in a smaller set of reserved legs, and it is possible that children of β get some new meetings, but the children are longer than α and β.

Now we shall consider the case when, for each $q\in R(T_i(\alpha))$, $s(i,q,\alpha)<s(i,q,\beta)\leqslant i<e(i,q,\alpha)<e(i,q,\beta)$ or $s(i,q,\beta)<s(i,q,\alpha)\leqslant i<e(i,q,\beta)<e(i,q,\alpha)$. In this case there is no state in $R(T_i(\alpha))$ where one of the two markers stays longer. Then:

(Ch. 2.1) The markers α and β exchange their names for every $r>i$:

- we set $\Delta_r(\alpha)=\Gamma_r(\beta)$ and $\Delta_r(\beta)=\Gamma_r(\alpha)$,
- replace each transition of the form $\stackrel{t_r,\beta}{\Longrightarrow}$ by $\stackrel{t_r,\alpha}{\Longrightarrow}$, and
- replace each transition of the form $\stackrel{t_r,\alpha}{\Longrightarrow}$ by $\stackrel{t_r,\beta}{\Longrightarrow}$.

After this change, in every state $q\in R(T_i(\alpha))$, either α stays longer than β, or β stays longer than α. Now we shall change the names of some successors of α and β.

(Ch. 2.2) For every $q\in R(T_i(\alpha))$, if $s(i,q,\alpha)<s(i,q,\beta)$, then:

For every marker of the form $\gamma=\beta\circ m\circ\delta$, if $Bp(\beta\circ m)=q$ and $\Gamma_j(\gamma)\neq\emptyset$ for some $s(i,q,\beta)\leqslant j<e(i,q,\beta)$, then we change γ for $\alpha\circ m\circ\delta$ in a similar way as in (Ch. 1). (Note that $\beta\circ m$ is a son of β born in q between $s(i,q,\beta)$ and $e(i,q,\beta)$, and γ is a successor of β). In this case α stays longer than β in q after the change (Ch. 2.1) and

all sons of β born in q between $s(i,q,\beta)$ and $e(i,q,\beta)$ (and their successors) become sons (and successors) of α.

(Ch. 2.3) For every $q \in R(T_i(\alpha))$, if $s(i,q,\beta) < s(i,q,\alpha)$, then:

For every marker of the form $\gamma = \alpha \circ m \circ \delta$, if $Bp(\alpha \circ m) = q$ and $\Gamma_j(\gamma) \neq \emptyset$ for some $s(i,q,\alpha) \leqslant j < e(i,q,\alpha)$, then we change γ for $\beta \circ m \circ \delta$. In this case β stays longer than α in q and all sons of α born in q between $s(i,q,\alpha)$ and $e(i,q,\alpha)$ (and their successors) become sons (and successors) of β.

(Ch. 2.4) If $\gamma = \alpha \circ m \circ \delta$, and

$Bp(\alpha \circ m) \notin R(T_i(\alpha))$ and $\Gamma_j(\gamma) \neq \emptyset$ for some $j > i$, or

$Bp(\alpha \circ m) = q \in R(T_i(\alpha))$ and $\Gamma_j(\gamma) \neq \emptyset$ for some $j > e(i,q,\alpha)$,

then we change γ for $\beta \circ m \circ \delta$. Sons of α born outside $R(T_i(\alpha))$ after the moment i, or born in some state $q \in R(T_i(\alpha))$ but after the moment $e(i,q,\alpha)$, become sons of β. The same applies to the successors of such markers.

(Ch. 2.5) If $\gamma = \beta \circ m \circ \delta$, and

$Bp(\beta \circ m) \notin R(T_i(\alpha))$ and $\Gamma_j(\gamma) \neq \emptyset$ for some $j > i$, or

$Bp(\beta \circ m) = q \in R(T_i(\alpha))$ and $\Gamma_j(\gamma) \neq \emptyset$ for some $j > e(i,q,\beta)$,

then we change γ for $\beta \circ m \circ \delta$.

Here, sons of β born outside $R(T_i(\alpha))$ after the moment i, or born in some state $q \in R(T_i(\alpha))$ but after the moment $e(i,q,\beta)$ become sons of α.

Again markers α and β do not meet in the configuration Δ_i. New meetings are possible in a smaller set of reserved legs or for longer markers.

In the new computation, for every marker γ, its new father $Ft(\gamma)$ has one of its reserved legs in the state $Bp(\gamma)$, and if γ is moved by a transition of the form $t = (\lambda, op, cl)$ then all of its children, old and new, have been already killed. To see these note the following:

- The markers α and β have a common father, $Ft(\alpha) = Ft(\beta)$. This is because $T_i(\alpha) = T_i(\beta)$ yields $T_i(Ft(\alpha)) = T_i(Ft(\beta))$ and $R(T_i(Ft(\alpha)))$ is not empty since $Bp(\alpha) \in R(T_i(Ft(\alpha)))$, and because $T_i(Ft(\alpha))$ satisfies (R.3) since $|Ft(\alpha)| < |\alpha|$.
- If a marker of the form $\gamma = \alpha \circ m$ or $\gamma = \beta \circ m$ is changing its name in (Ch. 1), (Ch. 2.2), or (Ch. 2.3), then its new father stays longer in $Bp(\gamma)$ than its previous father.
- All other markers which change their names, do this together with their fathers. □

Now we are ready to show that every computation with distiguishable markers can be simulated by an computation with nondistinguishable markers.

Theorem 5.2. *For any computation with distinguishable markers*:

$$in = \Gamma_0 \overset{t_1,\alpha_1}{\Longrightarrow} \cdots \overset{t_m,\alpha_m}{\Longrightarrow} \Gamma_m = fin$$

satisfying (R.3), *there exists a computation with nondistinguishable markers and the same transitions*:

$$in = C_0 \Rightarrow^{t_1,X_1} \cdots \Rightarrow^{t_m,X_m} C_m = fin.$$

Proof. We shall define the computation with nondistinguishable markers configuration after configuration so that the following condition holds:

(M.5) $C_i(X) = |T_i^{-1}(X)| = |\{\alpha \mid T_i(\alpha) = X\}|$, where $T_i(\alpha)$ stands for the type of the marker α in the ith step.

Let $X_0 = T_0(1)$ and $C_0(X_0) = 1$, $C_0(X) = 0$, for other types X. Obviously (M.5) holds. Suppose that we have defined the configuration C_{i-1} and that it satisfies (M.5). X_i and C_i are defined in the following way:

(1) If $t_i = (\lambda, op, q)$, then we set $X_i = T_i(\alpha_i)$ and $C_i(X_i) = C_{i-1}(X_i) + 1$.
(2) If $t_i = (\lambda, f, cl)$, then $X_i = T_{i-1}(\alpha_i)$ and $C_i(X_i) = C_{i-1}(X_i) - 1$.
(3) In other cases we set $X_i = T_{i-1}(\alpha_i)$, $Z = T_i(\alpha_i)$, $C_i(X_i) = C_{i-1}(X_i) - 1$, $C_i(Z) = C_{i-1}(Z) + 1$, and for every type of the kind $X_i \circ Y$ with $Y \neq \lambda$, we set $C_i(Z \circ Y) = C_{i-1}(X_i \circ Y)$, and $C_i(X_i \circ Y) = 0$.

We put $C_i(X) = C_{i-1}(X)$, for other types X, not mentioned above.

Again we omit the details of the proof that the transition $C_{i-1} \Rightarrow^{t_i, X_i} C_i$ is possible, and that C_i satisfies (M.5), because it is straightforward. We shall only show that in the case when $t_i = (\lambda, op, cl)$, there is no type Y with $FT(Y) = X_i, Bp(Y) = op$, and $C_{i-1}(Y) > 0$.

Contradicting this, suppose that such a Y exists.

Then there is β such that $T_{i-1}(\beta) = Y$ (because C_{i-1} satisfies (M.5)),

$\Gamma_{i-1}(\beta) = P(T_{i-1}(\beta)) = P(Y) \neq \emptyset$,

$T_{i-1}(Ft(\beta)) = FT(T_{i-1}(\beta)) = FT(Y) = X_i$,

$T_{i-1}(\alpha_i) = X_i$ and $Bp(\beta) = op \in R(X_i)$,

hence $R(X_i) \neq \emptyset$ and by (R.3) $Ft(\beta) = \alpha_i$, a contradiction with the definition of the computation $\Gamma_{i-1} \Rightarrow^{t_i, \alpha_i} \Gamma_i$.

From the assumption that the computation with distinguishable markers satisfies (R.3) it follows that C_i satisfies the condition (R.1). □

Theorem 5.3. *For any computation with nondistinguishable markers*:

$$in = C_0 \Rightarrow^{t_1, X_1} \cdots \Rightarrow^{t_m, X_m} C_m = fin,$$

there exists a computation with distinguishable markers and the same transitions:

$$in = \Gamma_0 \stackrel{t_1, \alpha_1}{\Longrightarrow} \cdots \stackrel{t_m, \alpha_m}{\Longrightarrow} \Gamma_m = fin.$$

Proof. We shall construct the computation with distinguishable markers, configuration by configuration. At the ith step, we define the marker α_i and the function T_i which describes types of all markers in the configuration. Remember that $\Gamma_i(\gamma) = P(T_i(\gamma))$, so the configuration Γ_i is also defined.

We set $\Gamma_0(1) = \{q_0\}$, and $\Gamma_0(\alpha) = \emptyset$ for $\alpha \neq 1$; $T_0(1) = X_0$, where X_0 is the starting type. Suppose that we have already defined the configuration Γ_{i-1} and that it satisfies the conditions (M.1)–(M.5). We define the marker α_i and the function T_i in the following way:

(1) If $t_i = (\lambda, op, q)$, then $C_{i-1}(FT(X_i)) = 1$, thus (by M.5), there exists exactly one marker β such that $T_{i-1}(\beta) = FT(X_i)$; we set: $\alpha_i = \beta \circ i$ and $T_i(\alpha_i) = X_i$.

(2) If $t=(\lambda, f, cl)$, then
$C_{i-1}(X_i)>0$; we choose one marker α_i such that $T_{i-1}(\alpha_i)=X_i$, and kill α_i, by setting $T_i(\alpha_i)=\lambda$.

(3) In other cases, $C_{i-1}(X_i)>0$; we choose α_i such that $T_{i-1}(\alpha_i)=X_i$ and we set: $T_i(\alpha_i)=Z$, where Z is the type chosen in transition $C_{i-1}\Rightarrow^{t_i,X_i}C_i$. We also change the types of ancestor markers. For every marker γ of the form $\gamma=\alpha_i\circ\delta$, if $\Gamma_i(\gamma)\neq\emptyset$, then, by (M.3.2), $T_{i-1}(\gamma)=X_i\circ Y$, and then we set $T_i(\gamma)=Z\circ Y$.

We omit here the details of the proof that the transition $\Gamma_{i-1}\Rightarrow^{t_i,\alpha_i}\Gamma_i$ is possible, and that Γ_i and T_i satisfies (M.1)–(M.5), because it is straightforward. We shall only show that in the case when $t_i=(\lambda, op, cl)$ there is no β such that $Ft(\beta)=\alpha_i, Bp(\beta)=op$, and $\Gamma_{i-1}(\beta)\neq\emptyset$.

Suppose that such a β exists. We set $Y=T_{i-1}(\beta)$,
by (M.3), $FT(Y)=T_{i-1}(Ft(\beta))=T_{i-1}(\alpha_i)=X_i$,
$Bp(Y)=Bp(T_{i-1}(\beta))=Bp(\beta)=op$,
and by (M.5), $C_{i-1}(Y)>0$, a contradiction with the definition of the transition $C_{i-1}\Rightarrow C_i$. □

6. Shuffle languages are in one-way-NSPACE$(\log n)$

In Section 3 we have proven that $L(E)$ is accepted by the automaton with distinguishable markers.

First we shall show that for every $w\in L(E)$ there exists an accepting computation with at most $d\cdot|w|$ markers, where d is the maximal number of nested shuffle closure operators in the expression E. Consider an accepting computation for w. We shall say that a transition t is silent if it is labelled by the empty word. Besides, we call a marker α redundant if all markers of the form $\alpha\circ\delta$ (i.e. α and its successors) are moved only by silent transitions. It is easy to see that if we get rid of all redundant markers and all transitions moving those markers, then we obtain an accepting computation for w with at most $d\cdot|w|$ markers.

From the proof of Theorem 5.2 it follows that there is an accepting computation with at most $d\cdot|w|$ nondistinguishable markers.

Now we define a one-way nondeterministic Turing machine M which accepts $L(E)$ in logarithmic space. M simulates the automaton A_E with nondistinguishable markers. On its work tape M keeps the current configuration of the automaton A_E. Every configuration of A_E can be kept in logarithmic space because there are at most $|Types|$ many numbers of size $d\cdot|w|$ to remember. In each step M guesses the next configuration C_i, transition to be taken t_i and the type X_i, and checks if the step $C_{i-1}\Rightarrow^{t_i,X_i}C_i$ is possible.

References

[1] V.K. Garg, M.T. Ragunath, Concurrent regular expressions and their relationship to Petri nets, Theoret. Comput. Sci. 96 (1992) 285–304.

[2] J.L. Gischer, Shuffle languages, Petri nets and context sensitive grammars, Commun. ACM 24 (1981) 597–605.
[3] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, Reading, MA, 1979.
[4] M. Jantzen, The power of synchronizing operations on strings, Theoret. Comput. Sci. 14 (1981) 127–154.
[5] M. Jantzen, Extending regular operations with iterated shuffle, Theoret. Comput. Sci. 38 (1985) 223–247.
[6] J. Jędrzejowicz, On the enlargement of the class of regular languages by shuffle closure, Inform. Process. Lett. 16 (1983) 51–54.
[7] J. Jędrzejowicz, Nesting of shuffle closure is important, Inform. Process. Lett. 25 (1987) 363–367.
[8] J. Jędrzejowicz, Shuffle Operation in Formal Languages, Wydawnictwo Uniwersytetu Gdanskiego, 1996.
[9] A.J. Mayer, L.J. Stockmeyer, The complexity of word problems – this time with interleaving, Inform. Comput. 115 (1994) 293–311.
[10] W.F. Ogden, W.E. Riddle, C. Rounds, Complexity of expressions allowing concurrency, in: Proc. 5th Ann ACM Symp. on Principles of Programming Languages, 1978, pp. 185–194.
[11] W.E. Riddle, Software system modelling and analysis, Tech. Report, Dept. of Computer and Communication Sciences, Univ. of Michigan, RSSM25, 1976.
[12] A.C. Shaw, Software descriptions with flow expressions, IEEE Trans. Software Eng. SE-4 (1978) 242–254.
[13] M.K. Warmuth, D. Haussler, On the complexity of iterated shuffle, J. Comput. Systems Sci. 28 (1984) 345–358.