Theoretical Informatics and Applications
Theoret. Informatics Appl. **35** (2001) 379–388

ON THE EXPRESSIVE POWER OF THE SHUFFLE OPERATOR MATCHED WITH INTERSECTION BY REGULAR SETS

Joanna Jędrzejowicz[1] and Andrzej Szepietowski[1]

Abstract. We investigate the complexity of languages described by some expressions containing shuffle operator and intersection. We show that deciding whether the shuffle of two words has a nonempty intersection with a regular set (or fulfills some regular pattern) is NL-complete. Furthermore we show that the class of languages of the form $L \cap R$, with a shuffle language L and a regular language R, contains non-semilinear languages and does not form a family of mildly context-sensitive languages.

Mathematics Subject Classification. 68Q15, 68Q45.

Introduction

Parallel composition of words appears to be an important issue both in the theory of concurrency and formal languages. Usually it is modeled by the shuffle operation. If the class of regular languages is extended by the shuffle operation $\odot$ and the iteration of the shuffle operation, $\otimes$, then we obtain the class of shuffle languages SL, which is useful in describing concurrent non-communicating processes [6]. We know [3] that for each shuffle language L there exists a one-way nondeterministic Turing machine which decides the membership problem for L in logarithmic space. This implies that shuffle languages are context sensitive and that they are recognizable in polynomial time. But the shuffle operation can be helpful to describe more complex languages. Warmuth and Haussler [8] show that if we add the intersection operation then we can obtain an NP-complete language, namely the language $\{\$a^n b^n c^n d^n \mid n \geq 0\}^{\otimes} = \big((\$ + abcd)^{\otimes} \cap (\$a^* b^* c^* d^*)\big)^{\otimes}$. They

Keywords and phrases: Formal languages, shuffle, space complexity.

[1] Institute of Mathematics, University of Gdańsk, ul Wita Stwosza 57, 80952 Gdańsk, Poland; e-mail: jj@math.univ.gda.pl & matszp@math.univ.gda.pl

also show that the problem of deciding for any words $w, w_1, \ldots, w_n \in \{a,b,c\}^*$ whether $w \in w_1 \odot \cdots \odot w_n$ is NP-complete.

In this paper we consider some languages described with the help of both the shuffle operations and the intersection with regular sets. Firstly, we consider languages of the form $\{u\$v \mid u \odot v \cap R \neq \emptyset\}$ or $\{u\$v \mid u \odot v \subset R\}$, where R is a regular language. The latter is the problem of deciding whether the shuffle of two words fulfills the regular pattern R. We show that every such language belongs to NL (is acceptable in nondeterministic logarithmic space) and that there exists an NL-complete language of each of these forms. We also show that there are NL-complete languages of the form $\{u \mid u \odot u \cap R \neq \emptyset\}$ or $\{u \mid u \odot u \subset R\}$.

Furthermore we consider the class $SL \wedge Reg = \{L \cap R \mid \ L \in SL,\ R \in Reg\}$ of languages represented as the intersection of a shuffle language and a regular one. The reason for starting this investigation was, as it seemed, the connection of $SL \wedge Reg$ with mildly context-sensitive languages considered in [4] and defined as follows: a family $\mathcal{L}$ is a mildly context-sensitive family of languages if each language in $\mathcal{L}$ is semilinear, and recognizable in deterministic polynomial time, and if $\mathcal{L}$ contains the following three languages: $L_1 = \{a^i b^i c^i \mid i \geq 0\}$ (multiple agreements), $L_2 = \{a^i b^j c^i d^j \mid i,j \geq 0\}$ (crossed agreements), and $L_3 = \{ww \mid w \in \{a,b\}^*\}$ (duplications). The class $SL \wedge Reg$ seemed a good candidate for a family of mildly context-sensitive languages in view of fulfilling the condition of polynomial complexity of membership problem [3] and neat description of multiple agreements and crossed agreements by suitable expressions. In this paper we show that $SL \wedge Reg$ contains some non-semilinear languages and does not contain L_3. Hence $SL \wedge Reg$ does not form a family of mildly context-sensitive languages.

1. Preliminaries

Let Σ be any fixed alphabet and λ the empty word. By $u \cdot v$ we denote the concatenation of two words u and v. We shall also use the notation $\prod_{i \in I} \sigma_i$, to denote the concatenation $\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_s}$, where $I = \{i_1, i_2 \ldots i_s\}$ and $i_k < i_{k+1}$ for every $1 \leq k \leq s-1$.

The shuffle operation $\odot$ is defined inductively as follows:

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

Note that $u \odot v$ consists of all words $z \in \Sigma^*$ which can be decomposed into $z = w_1 \cdot w_2 \cdots w_r$ with $w_i \in \Sigma^*$, $u = \prod_{i \in I} w_i$ and $v = \prod_{i \notin I} w_i$, for some subset $I \subset \{1, 2, \ldots, r\}$. The shuffle operation is extended in a natural way to languages: 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:

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

Definition 1.1. Each $a \in \Sigma$, as well as 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 shuffle 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 $L({S_1}^{\otimes}) = L_1^{\otimes}$.

In what follows we shall not distinguish between the shuffle expression and the language generated by this expression. Shuffle languages are denoted by SL and regular languages by Reg.

Assume that $\Sigma = \{a_1, \ldots, a_n\}$. The Parikh mapping, denoted by Ψ, is Ψ: $\Sigma^* \to \mathbb{N}^n$:

$$\Psi(w) = (\#_{a_1} w, \ldots, \#_{a_n} w),$$

where $\#_x w$ denotes the number of occurrences of the letter x in the word w.

For a language $L \subset \Sigma^*$ its Parikh image is defined by

$$\Psi(L) = \bigcup_{w \in L} \Psi(w).$$

A linear set is a set $A \subseteq \mathbb{N}^n$ such that $A = \{v_0 + \sum_{i=1}^m x_i v_i \mid x_i \in \mathbb{N}\}$ for some $v_0, v_1, \ldots, v_m \in \mathbb{N}^n$. A semilinear set is a finite union of linear sets and a language L is semilinear if $\Psi(L)$ is a semilinear set. Observe that for any sets $A, B \subseteq \Sigma^*$, we have

$$\Psi(A \odot B) = \Psi(AB),$$

$$\Psi(A^{\otimes}) = \Psi(A^*),$$

and since regular languages are semilinear, shuffle languages are semilinear, as well.

Example 1.2. Consider the language generated by the intersection of two shuffle expressions S_1, S_2 and a regular expression R:

$$S_1 = (((ab)^{\otimes} \odot X)Y)^*,$$

$$R = (a^* X b^* Y)^*,$$

$$S_2 = aX(((ba)^{\otimes} \odot Y)aX)^* b^* Y$$

over the alphabet $\{a, b, X, Y\}$. It is easy to see that $S_1 \cap R$ contains words of the form

$$w = a^{n_1} X b^{n_1} Y \cdots a^{n_k} X b^{n_k} Y.$$

If $w \in S_2 \cap R$, then

$$w = aXb^{m_1} Y a^{m_1+1} X b^{m_2} Y a^{m_2+1} \cdots b^{m_l} Y a^{m_l+1} X b^{m_{l+1}} Y.$$

And if $w \in S_1 \cap R \cap S_2$ we have $k = l + 1$ (due to the number $\#_Y w$), besides $n_1 = 1$, $m_1 = n_1 = 1$, $n_2 = m_1 + 1 = 2$, $m_2 = n_2 = 2$, ..., $n_k = m_{k-1} + 1 = k$. Thus

$$S_1 \cap R \cap S_2 = \{aXbYa^2Xb^2Y \cdots a^k X b^k Y \mid \ k \geq 0\}$$

is a non-semilinear language.

By $NSPACE(\log n)$ or NL we shall denote the class of languages accepted by nondeterministic Turing machines within logarithmic space and by one-way-$NSPACE(\log n)$ the class of languages accepted by one-way-nondeterministic Turing machines within logarithmic space.

2. Finding the regular pattern in the shuffle of words

We consider the problem of deciding whether the shuffle of two words fulfills some regular pattern or has a nonempty intersection with a regular set. In other words the languages of the form $\{u\$v \mid u \odot v \subset R\}$, or $\{u\$v \mid u \odot v \cap R \neq \emptyset\}$ where $R \in Reg$ and \$ is a special separating symbol.

Theorem 2.1. *For any regular language R the sets $\{u\$v \mid u \odot v \subset R\}$ and $\{u\$v \mid u \odot v \cap R \neq \emptyset\}$ are accepted in nondeterministic logarithmic space and there exists a NL-complete language of each of the above forms.*

Proof. First we show that the language $\{u\$v \mid u \odot v \cap R \neq \emptyset\}$ is accepted by a nondeterministic Turing machine M in logarithmic space. The machine M guesses one by one letters of a word r of length $n = |u| + |v|$ and checks if r belongs to $u \odot v \cap R$. To check if $r \in u \odot v$, M uses two pointers; the first points to the successive letters of u, the second to the letters of v, at the beginning they point to the first letters of u and v. M guesses decomposition of $r = r_1 \cdot r_2 \cdot \cdots \cdot r_n$ into $\prod_{i \in I} r_i = u$ and $\prod_{i \notin I} r_i = v$, for some subset $I \subset \{1, 2, \ldots, n\}$ and $r_i \in \Sigma$. In the i-th step M guesses r_i and whether $i \in I$ or not. If $i \in I$ then checks if the first pointer points to r_i and moves this pointer right to the next position. If $i \notin I$, checks and moves the second pointer. In order to check if $r \in R$, M simultaneously runs the finite automaton accepting R on the successive letters of r.

Note that

$$\{u\$v \mid u \odot v \subset R\} = \{u\$v \mid u \odot v \cap R^c \neq \emptyset\}^c \cap (\Sigma^*\$\Sigma^*).$$

Where A^c is the complement of the language A. Using the fact that the class NL is closed under complement and intersection with regular languages (see [5, 7]) we have that the language

$$\{u\$v \mid u \odot v \subset R\}$$

belongs to NL, as well.

Now we show that there is an NL-complete language of the form

$$\{u\$v \mid u \odot v \cap R \neq \emptyset\}\cdot$$

To do this we reduce the language GAP to the language

$$L = \{u\$v \mid u \odot v \cap R \neq \emptyset\}$$

with the regular language

$$R = d1d\big((a+b+e+D+E+0+1)^*eE(0a+1b)^*eE(d+e+0+1)^*dD(0a+1b)^*dD\big)^*$$

over the alphabet $\{0, 1, a, b, d, e, D, E\}$.

The language GAP consists of directed graphs $G = (V, E)$, which have a path from the first vertex to the last one. We shall assume that $V = \{1, \ldots, n\}$ for some n and that if $(i, j) \in E$ then $i < j$ (even in this case GAP remains NL-complete, see [5]). The graph G is coded in the following way

$$G = dc(1)dec\left(i_1^1\right)e\cdots ec\left(i_{j_1}^1\right)e\cdots dc(k)dec\left(i_1^k\right)e\cdots ec\left(i_{j_k}^k\right)e\cdots dc(n)d,$$

where $c(i)$ is the number i coded in binary (note that $c(1) = 1$), the symbols d and e play the role of the separators, and $i_1^k, \ldots, i_{j_k}^k$ are the vertices which are joined by an edge from k.

Reduction from GAP to $L = \{u\$v \mid u \odot v \cap R \neq \emptyset\}$ is performed by the function $h(G) = u\$v$ with $u = G$ and

$$v = EC(1)EDC(1)DEC(2)EDC(2)D\cdots EC(n)EDC(n)D,$$

where $C(i)$ is the number i coded in binary with a standing for 0 and b standing for 1 and the symbols D, E play the role of the separators,

First we show that if $G \in$GAP then there exists $r \in u \odot v \cap R$. Suppose that there is a path $1 = s_1, s_2, \ldots, s_t = n$ joining 1 with n. Then there is a word

$$r = U_1U_2\cdots U_t \in u \odot v \cap R$$

where $U_1 = dc(1)d$ and for each i, $2 \leq i \leq t$

$$U_i \in (a+b+e+D+E+0+1)^*eE(0a+1b)^*eE(d+e+0+1)^*dD(0a+1b)^*dD$$

is of the form:

$$U_i = v_i eEw_i eEx_i dDy_i dD,$$

where:

- the part $eEw_i eE \in eE(0a+1b)^* eE$ is composed from $ec(s_i)e$ and $EC(s_i)E$, the part $ec(s_i)e$ is taken from the list

$$ec\left(i_1^{s_{i-1}}\right) e \ldots ec\left(i_{j_{s_{i-1}}}^{s_{i-1}}\right) e$$

  and $EC(s_i)E$ is taken from v;
- the part $dDy_i dD \in dD(0a+1b)^* dD$ comes from a shuffle of $dc(s_i)d$ (taken from u) and $DC(s_i)D$ (taken from v);
- $x_i \in (d+e+0+1)^*$ is the fragment of u standing between $ec(s_i)e$ and $dc(s_i)d$;
- $v_i \in (a+b+e+D+E+0+1)^*$ is the concatenation (or any other shuffle) of the fragment of u standing between $dc(s_{i-1})d$ and $ec(s_i)e$ and the fragment of v standing between $DC(s_{i-1})D$ and $EC(s_i)E$.

For every i, $2 \leq i \leq t$ the pair $EC(s_i)EDC(s_i)D$ joins the element $ec(s_i)e$ from the list

$$ec\left(i_1^{s_{i-1}}\right) e \cdots ec\left(i_{j_{s_{i-1}}}^{s_{i-1}}\right) e$$

(remember that there is an edge from s_{i-1} to s_i and $ec(s_i)e$ is on the list) with the element $dc(s_i)d$ which stands in front of the list of vertices joined by an edge from s_i.

Now we show that if there exists $r \in u \odot v \cap R$ then there is a path from 1 to n in G.

Since $r \in R$ then r is of the form

$$r = U_1 U_2 \cdots U_t$$

with $U_1 = d1d = dc(1)d$ and for each i, $2 \leq i \leq t$

$$U_i \in (a+b+e+D+E+0+1)^* eE(0a+1b)^* eE(d+e+0+1)^* dD(0a+1b)^* dD$$

is of the form

$$U_i = v_i eEw_i eEx_i dDy_i dD$$

with $v_i \in (a+b+e+D+E+0+1)^*$, $w_i \in (0a+1b)^*$, $x_i \in (d+e+0+1)^*$, and $y_i \in (0a+1b)^*$.

Note that $eEw_i eE \in eE(0a+1b)^* eE$ can be only composed from $ec(\alpha_i)e$ and $EC(\gamma_i)E$ with $ec(\alpha_i)e$ taken from some list

$$ec\left(i_1^{\beta_i}\right) e \ldots ec\left(i_{j_{\beta_i}}^{\beta_i}\right) e$$

and $EC(\gamma_i)E$ taken from v. This composition is possible only if $\alpha_i = \gamma_i$ (ensured by $w_i \in (0a+1b)^*$).

Similarly $dDy_idD \in dD(0a+1b)^*dD$ can be only composed from $dc(\delta_i)d$ taken from u and $DC(\epsilon_i)D$ taken from v. Note that this composition is possible only if $\delta_i = \epsilon_i$.

Observe that:

1. in U_2 there is no symbol d in v_2, hence $ec(\alpha_2)e$ is taken from the first list in u, $\beta_2 = 1$, and there is an edge from 1 to α_2;
2. in U_i there is no symbol E or D in x_i, hence $\gamma_i = \epsilon_i$, and $\alpha_i = \delta_i$;
3. there is no symbol d standing between last D in U_i and the first e in U_{i+1}, hence $\beta_{i+1} = \delta_i = \alpha_i$, and α_{i+1} is taken from the list

$$ec(i_1^{\alpha_i})e \dots ec(i_{j_{\alpha_i}}^{\alpha_i})e$$

   so there is an edge between α_i and α_{i+1};
4. there is no symbol d behind dDy_tdD in U_t, hence $\delta_t = \alpha_t = n$.

Hence, we have shown that the sequence $1 = \alpha_1, \alpha_2, \dots, \alpha_t = n$ forms a path from 1 to n and that the language L is NL-complete.

Because NL is closed under complement the language non-GAP is also NL-complete. It is easy to see that the function h reduces non-GAP into

$$\{u\$v \mid u \odot v \subset R^c\}$$

and thus this language is also NL-complete. □

Consider now the languages of the form

$$\{u \mid u \odot u \cap R \neq \emptyset\}$$

with some regular language R. Of course every such language belongs to NL and also among such languages there are NL-complete ones. Let

$$T = X(0+1+d+e)^*YXRZY(a+b+D+E)^*Z$$

where X, Y, Z are new separating symbols. It is easy to see that if $w = XuYvZ$ then $w \odot w \cap T \neq \emptyset$ if and only if $u \odot v \cap R \neq \emptyset$. Hence we have:

Corollary 2.2. *For any regular language R the sets $\{u \mid u \odot u \subset R\}$ and $\{u \mid u \odot u \cap R \neq \emptyset\}$ are accepted in nondeterministic logarithmic space and there exists a NL-complete language of each of the above forms.*

3. Intersections with regular languages

Now we consider the class $SL \wedge Reg = \{L \cap R \mid L \in SL,\ R \in Reg\}$ of languages represented as the intersection of a shuffle language and a regular one. The reason

for starting this investigation was, as it seemed, the connection of $SL \wedge Reg$ with mildly context-sensitive languages considered in [4] and defined as follows:

Definition 3.1. A family $\mathcal{L}$ is a mildly context-sensitive family of languages if the following conditions are fulfilled:

- each language in $\mathcal{L}$ is semilinear;
- for each language in $\mathcal{L}$ the membership problem is solvable in deterministic polynomial time, and
- $\mathcal{L}$ contains the following three non-context-free languages
  - $L_1 = \{a^i b^i c^i \mid i \geq 0\}$ (multiple agreements);
  - $L_2 = \{a^i b^j c^i d^j \mid i, j \geq 0\}$ (crossed agreements);
  - $L_3 = \{ww \mid w \in \{a, b\}^*\}$ (duplications).

The class $SL \wedge Reg$ seemed a good candidate for a family of mildly context-sensitive languages in view of fulfilling the condition of polynomial complexity of membership problem and neat description of multiple agreements and crossed agreements by suitable expressions. By [3], $SL \subset$ one-way-$NSPACE(\log n)$, and the class one-way-$NSPACE(\log n)$ is closed under intersections with regular languages [7], so $SL \wedge Reg \subset$ one-way-$NSPACE(\log n) \subset P$.

The languages: multiple agreements L_1 and crossed agreements L_2 are in $SL \wedge Reg$ since

$$L_1 = (abc)^{\otimes} \cap a^* b^* c^*,$$

$$L_2 = (ac + bd)^{\otimes} \cap a^* b^* c^* d^*.$$

But the language L_3 does not belong to one-way-$NSPACE(\log n)$ [7], hence L_3 is not in $SL \wedge Reg$. It is interesting that $SL \wedge Reg$ contains two languages which are similar to duplicates.

Suppose that for any letter $a \in \Sigma$ we introduce a new symbol a' and denote $\Sigma' = \{a' \mid a \in \Sigma\}$. By a primed word we understand a word where all the symbols of the original word are replaced by primed symbols, that is if $w = a_1 \cdots a_n \in \Sigma^*$, then $w' = a'_1 \cdots a'_n \in \Sigma'^*$.

Example 3.2. The following language $L \subseteq (\Sigma \cup \Sigma')^*$, of non-duplicates, is in the class $SL \wedge Reg$:

$$\{uv' \mid v^R \neq u\} = \left(\bigcup_{a,b \in \Sigma, a \neq b} \Sigma^* a X^{\otimes} b' \Sigma'^* \right) \cap \Sigma^* \Sigma'^*,$$

where $X = \{xx' \mid x \in \Sigma\}$. Also the language of 'almost' duplicates is in $SL \wedge Reg$:

$$\{wv' \mid v \in com(w)\} = EQ(\Sigma) \cap \Sigma^* \Sigma'^*,$$

where $com(w) = \{u \in \Sigma^* \mid \text{ for each } a \in \Sigma, \#_a w = \#_a u\}$ is the commutative closure of w and $EQ(\Sigma) = (a_1 a'_1 + \cdots + a_n a'_n)^{\otimes}$, for $\Sigma = \{a_1, \ldots, a_n\}$.

Now we show that $SL \wedge Reg$ contains non-semilinear languages. Araki and Tokura proved the following:

Theorem 3.3. [1] *For any alphabet Σ and each recursively enumerable language K over Σ there exist two alphabets $\Gamma = \{[_1, [_2, \dots, [_c,]_1,]_2, \dots,]_c\}$, $\Delta = \{\sigma_1, \omega_1, \dots, \sigma_d, \omega_d\}$ and a shuffle expression S over $\Sigma \cup \Gamma \cup \Delta$ such that*

$$K = \{x_1 \cdots x_k \in \Sigma^* \mid \text{ there exist } y_1, \dots, y_k \in (\Gamma \cup \Delta)^*, \\ \text{such that } x_1 y_1 \cdots x_k y_k \in S, \text{ and } y_1 \cdots y_k \in S_{\text{lock}} \odot S_{\text{signal}}\},$$

where $S_{\text{lock}} = ([_1\ ]_1)^* \odot \cdots \odot ([_c\ ]_c)^*$, $S_{\text{signal}} = (\sigma_1 + \sigma_1\omega_1)^* \odot \cdots \odot (\sigma_d + \sigma_d\omega_d)^*$.

Note that K can be expressed in the form

$$K = er_\Sigma(S \cap R),$$

where er_Σ is a weak identity or erasing homomorphism defined by

$$er_\Sigma(x) = \begin{cases} x & \text{if } x \in \Sigma \\ \lambda & \text{otherwise} \end{cases}$$

and $R = S_{\text{lock}} \odot S_{\text{signal}} \odot \Sigma^*$ is regular, because the shuffle of regular languages is regular.

Take now any non-semilinear recursively enumerable language K, by the above theorem it can be represented by

$$K = er_\Sigma(L)$$

for some $L \in SL \wedge Reg$. Since the erasing homomorphism preserves semilinearity the language L is also non-semilinear. Hence we have:

Corollary 3.4. *There exists a non-semilinear language in the class $SL \wedge Reg$.*

Corollary 3.5. *The class $SL \wedge Reg$ does not form a family of mildly context-sensitive languages.*

Remark. We think that Theorem 3.3 can be strengthened in the following way. For any alphabet Σ there exists an alphabet $\Gamma \supset \Sigma$ and a shuffle language S over Γ which is a "very special language" ([2]), that is for each recursively enumerable language K over Σ there exists a regular language over Γ such that $K = er_\Sigma(S \cap R)$.

4. Open problems

1. Is there an NL-complete language in the class SL?

By [3] $SL \subset NL$. This problem is equivalent to the problem whether there is an NL-complete language in $SL \wedge Reg$. Indeed suppose that there exists a reduction h from GAP to $L \cap R \subset \Sigma^*$ with some $L \in SL$ and $R \in Reg$, and let \$ be an arbitrary symbol not in Σ. Now, there exists the reduction g from GAP to the shuffle

language $L \subset (\Sigma \cup \{\$\})^*$: on the input x, g computes $h(x)$, checks simultaneously if it belongs to R, and sets $g(x) = h(x)$ if $h(x) \in R$, and $g(x) = h(x)\$$ otherwise.

2. Is there an NL-complete language of the form

$$L = \{u\$v_1\$\cdots\$v_k \mid u \in v_1 \odot \cdots \odot v_k, \text{ and } u, v_1, \ldots v_k \in \Sigma^*\}$$

for some constant k and some alphabet Σ?

Similarly as in the proof of Theorem 2.1 one can show that $L \in NL$. By [8], L is NP-complete, if k is not a constant and $\Sigma = \{a, b, c\}$.

References

[1] T. Araki and N. Tokura, Flow languages equal recursively enumerable languages. *Acta Inform.* **15** (1981) 209-217.
[2] D. Haussler and P. Zeiger, Very special languages and representations of recursively enumerable languages *via* computation stories. *Inform. and Control* **47** (1980) 201-212.
[3] J. Jędrzejowicz and A. Szepietowski, Shuffle languages are in **P**. *Theoret. Comput. Sci.* **250** (2001) 31-53.
[4] C. Martin-Vide and A. Mateescu, Special families of sewing languages, in *Workshop – Descriptional complexity of automata, grammars and related structures*. Magdeburg (1999) 137-143.
[5] C.H. Papadimitriou, *Computational Complexity*. Addison-Wesley Publ. Co (1994).
[6] A.C. Shaw, Software descriptions with flow expressions. *IEEE Trans. Software Engrg.* **3** (1978) 242–254.
[7] K. Wagner and G. Wechsung, *Computational Complexity*. Reidel, Dordrecht, The Netherlands (1986).
[8] M.K. Warmuth and D. Haussler, On the complexity of iterated shuffle. *J. Comput. Syst. Sci.* **28** (1984) 345-358.

Communicated by Ch. Choffrut.
Received December, 2000. Accepted October, 2001.