Information Processing Letters 16 (1983) 51–54
North-Holland Publishing Company

26 February 1983

ON THE ENLARGEMENT OF THE CLASS OF REGULAR LANGUAGES BY THE SHUFFLE CLOSURE

Joanna JĘDRZEJOWICZ

Institute of Mathematics, University of Gdańsk, ul. Wita Stwosza 57, 80-952 Gdańsk, Poland

Communicated by W.M. Turski
Received 28 May 1982
Revised 22 November 1982

Keywords: Control flow, regular expressions, shuffle operation, shuffle closure, flow expressions, context-sensitive languages

1. Introduction

In [6] Shaw introduced two operations on languages: shuffle $\odot$ and shuffle closure $\circledcirc$, and defined the class of flow expressions by introducing these two additional operations into the class of regular expressions. The class of flow languages was defined as the class generated by flow expressions similarly as $\mathfrak{RL}$-regular languages with some restrictions imposed by the so-called lock and signal symbols which are capable of describing concurrency and synchronization. Flow languages were intensively investigated, and recently, in [2], it was shown that their class is exactly equal to the class of all recursively enumerable languages. If one ignores the restrictions imposed by the special meaning of signal and lock symbols, then the obtained class of languages (we call them $\mathfrak{PFL}$-pseudo-flow languages in the sequel) is properly contained in the class of flow languages. The aim of this paper is to study the properties of $\mathfrak{PFL}$.

It is well known [6] that the class of regular languages is not enlarged if one introduces the shuffle operation into it. In Section 4 we show that in case of shuffle closure this is not true and that the class $\mathfrak{PFL}$ forms a proper subclass of $\mathfrak{CSL}$ (the class of context-sensitive languages).

In Section 3 we prove a version of a pumping lemma for $\mathfrak{PFL}$ which provides a necessary condition for a nonregular language to be in $\mathfrak{PFL}$.

2. Basic definitions

The class of flow expressions $\mathfrak{FE}$ and languages generated by $\mathfrak{FE}$ was introduced in [6] as follows.

Definition 2.1. Let Σ stand for a finite alfabet.

(1) Each $a \in \Sigma$, ϕ and ϵ are *flow expressions.*

(2) If S_1, S_2 are flow expressions, then (S_1), $S_1 \cdot S_2$, $S_1 + S_2$, S_1^*, $S_1 \odot S_2$ and $S_1^{\circledcirc}$ are flow expressions.

For a flow expression S we define

$$S^0 = \epsilon, \qquad S^i = S^{i-1} \cdot S \quad \text{for } i > 0,$$
$$S^{\odot 0} = \epsilon, \qquad S^{\odot i} = S^{\odot i-1} \odot S \quad \text{for } i > 0.$$

For a flow expression S we define the *language* L(S) generated by S.

Definition 2.2. Let $a \in \Sigma$, $S_1, S_2 \in \mathfrak{FE}$.

The languages $L(a)$, $L(\phi)$, $L(\epsilon)$, $L((S_1))$, $L(S_1 \cdot S_2)$, $L(S_1 + S_2)$ and $L(S_1^*)$ are defined in a usual way (these are regular expressions).

If $L(S_1) = L_1$, $L(S_2) = L_2$, then

(i) $S_1 \odot S_2$ generates the *shuffle of languages* L_1 and L_2,

$$L(S_1 \odot S_2) = L_1 \odot L_2 = \{x_1 y_1 \cdots x_k y_k : x_i, y_i \in \Sigma^*, \; i = 1, \ldots, k, \; x_1 x_2 \cdots x_k \in L_1 \cdot y_1 y_2 \cdots y_k \in L_2\},$$

0020-0190/83/0000–0000/$03.00

(ii) $S_1^{\circledast}$ generates the *shuffle closure* of L_1,

$$L(S_1^{\circledast}) = L_1^{\circledast} = \bigcup_{i=0}^{\infty} L(S_1^{\circledast i}).$$

Definition 2.3. $\mathfrak{PFL} = \{L(S): S \in \mathfrak{FE}\}$ is the class of pseudo-flow languages.

3. Pumping lemma for pseudo-flow languages

Lemma 3.1. *If* $L \subset \Sigma^*$ *is a pseudo-flow language which is nonregular, then there exists a constant* z *such that for every* $n \geqslant z$ *there is a word* $s \in L$, $|s| \geqslant n$ *which can be represented as* $s = utw$, u, t, $w \in \Sigma^*$ *with* t *containing at least two different letters and* $L(ut^{\circledcirc} w) \subset L$.

Proof. Let S stand for the flow expression which generates L, i.e., $L = L(S)$. L is nonregular and therefore S contains at least one $\circledast$ operator.

Thus, there exists a flow expression T such that

$$S = \alpha T^{\circledcirc} \beta$$

and

$$\alpha, \beta \in (\Sigma + \{\cdot, +, *, \odot, \circledast, (,)\})^*$$

and α, β may not be in $\mathfrak{FE}$.

The language L(T) generated by T contains a word X having at least two different letters of Σ as otherwise $T^{\circledcirc}$ might have been replaced by T*, we have $L((a^k)^{\circledcirc}) = L((a^k)^*)$.

We are going to prove Lemma 3.1 using the induction on the number ℓ of symbols of type $+, \cdot, *, \odot, (,), \circledast$ appearing in α and β. Suppose $\ell = 0$, thus $S = T^{\circledcirc}$.

In this case we define z to be equal to the minimal length of a word from L(T) containing at least two different letters. It should be obvious that $L(T^{\circledcirc})$ contains appropriately long words and if $t \in L(T^{\circledcirc})$, then $L(t^{\circledcirc}) \subset L(T^{\circledcirc})$.

Induction step. Suppose that the lemma is true for the number of operators in α and β not exceeding ℓ.

We are going to prove it for $\ell + 1$.

Thus, there exists a flow expression S' such that $T^{\circledcirc}$ is a subword of S', and S' contains no more than ℓ of the considered operators.

Besides, one of the following cases takes place:

Case 1. $S = (S')$.

Case 2. $S = S'^*$.

Case 3. $S = S'^{\circledcirc}$.

Case 4. There exists a flow expression S'' such that

Case 4a. $S = S' + S''$,

Case 4b1. $S = S' \cdot S''$,

Case 4b2. $S = S'' \cdot S'$,

Case 4c. $S = S' \odot S''$.

Let z' stand for the constant of Lemma 3.1 defined so far for the flow expression S'.

For each of Cases 1–3 and 4a we define $z = z'$ and use the following obvious containments,

$$L(S) \subset L(S^*),$$

$$L(S) \subset L(S^{\circledcirc}),$$

$$L(S) \subset L(S + R) \quad \text{for any } R \in \mathfrak{FE}.$$

For Case 4b, let $z = z' + |r|$ where r is the shortest word from L(S''). Let $m \geqslant z$. Using the induction step we claim that there exists an $s = utw \in L(S')$ with $|s| \geqslant m - |r| \geqslant z'$ and $L(ut^{\circledcirc} w) \subset L(S')$; thus, for Case b1,

$$sr = utwr \in L(S' \cdot S''),$$

$$|sr| \geqslant m \quad \text{and} \quad L(ut^{\circledcirc} wr) \subset L(S' \cdot r) \subset L(S).$$

Case b2 follows in a similar way.

For Case 4c, let $z = z'$ and use $L(S' \cdot S'') \subset L(S' \odot S'')$.

Corollary 3.2. *The word* s *of Lemma* 3.1 *may be written as*

$$s = utw = uxt_1t_2yw, \quad x,y \in \Sigma^*, \ t_1,t_2 \in \Sigma, \ t_1 \neq t_2$$

and

$$s_k = ux^k t_1^k t_2^k y^k w \in L,$$

$$r_k = ux^k (t_1t_2)^k y^k w \in L \quad \textit{for each } k = 0, 1, 2, \ldots .$$

Proof. Both $s_k, r_k \in L(ut^{\odot k} w) \subset L$.

Example 3.3. In [6] it was mentioned that the nonregular language $L = \{a^n b^n: n \geqslant 0\}$ cannot be expressed without the waits and signals, i.e., $L \notin \mathfrak{PFL}$.

This can be shown straighforward using the above criterion, as in this case

$t_1t_2 = ab$

and

$r_k = ux^k(ab)^k y^k w \notin L \quad \text{for } k \geqslant 2.$

Example 3.4. Let $V = (s, T)$ stand for the Vectro Addition System (see [4] for definition) defined as follows,

$s = (2, 1, 0),$

$T = \{t_1, t_2, t_3\} \quad \text{with } t_1 = (-1, -1, 1),$

$t_2 = (2, 0, 0), t_3 = (0, 1, -1).$

We shall investigate the language $L^V \subset T^*$ of V defined as a set of all legal strings $w \in T^*$ (a string $w \in T^*$ is legal in V if, for every prefix v of w, $s + v \in N^3$ where concatenation of letters of T stands for the componentwise sum).

For example t_1t_1 is not a legal string as

$$s + t_1t_1 = (2, 1, 0) + (-1, -1, 1) + (-1, -1, 1) = (0, -1, 2)$$

but t_1 is legal. It can easily be shown (using the criterion of [3]) that in this case L^V is not regular.

Let R(V) stand for the reachability set of V, i.e.,

$R(V) = \{s + w: w \text{ legal in } V\} \subset N^3.$

Using the well-known method of Karp–Miller [4] one can check that the set R(V) is infinite, and besides

$$\text{if } m = (m_1, m_2, m_3) \in R(V), \quad \text{then } m_2, m_3 \in \{0, 1\}. \tag{1}$$

It follows then, that

$$xt_1^2 \notin L^V, \tag{2}$$

$$xt_3^2 \notin L^V \quad \text{for } x \in T. \tag{3}$$

Conversely, suppose that

$xt_1^2 \in L^V,$

then obviously

$xt_1 \in L^V.$

Let

$s + x = m = (m_1, m_2, m_3) \in R(V),$

$m + t_1 \in R(V) \quad \text{as } xt_1 \in L^V,$

and thus $m_2 = 1$ (if one uses (1) and $m_2 + (-1) \geqslant 0$),

$$m + t_1 = (m_1, 1, m_3) + (-1, -1, 1) = (m_1 - 1, 0, m_3 + 1) = m'.$$

t_1 may not be applied to m′ and therefore $xt_1^2 \notin L^V$.

A similar argument may be used to establish (3).

Suppose that L^V is a pseudo-flow language. Then there exist a constant z and a word $s \in L^V$, $|s| \geqslant z$, which can be represented as

$s = uxt_it_jyw, \quad i \neq j;$

then

$s_2 = ux^2t_i^2t_j^2yw \in L^V \quad \text{from Corollary 3.2}$

and from $i \neq j$ it follows that

$s_2 = x't_i^2y' \in L^V$

or

$s_2 = x't_3^2y' \in L^V,$

which is in contradiction to (2) or (3) respectively.

It is worth noticing that L^V is certainly the flow language (as L^V is recursive) and is generated by the following flow expressions S with [,] playing the special role of lock symbols:

$$S = \left((t_2 \cdot [t_1t_3t_1t_3])^{\circledast} + t_1t_3t_1t_3 \odot t_2\right)^* \cdot (\epsilon + t_1 + t_1t_3 + t_1t_3t_1 + t_1t_3t_1t_3) \odot t_2^*.$$

4. Other properties of the class $\mathfrak{PFL}$

In [5] it was shown that the class of context-free languages is not closed under the shuffle operation. We shall prove that in case of context-sensitive languages this no longer holds.

Theorem 4.1. *The class of context-sensitive languages is closed under the shuffle operation* $\odot$ *and shuffle closure* $\circledast$.

Proof. At first, suppose that L is any context-sensitive language, i.e., there exists a nondeterministic Turing machine $\overline{M}$ which accepts L in space S(n) = n (constant speed-up may be used).

We are going to show the construction of the machine M accepting $L^{\odot}$.

Now, let us make the following obvious observation:

if $|y| = k$, then $y \in L^{\odot}$

iff $y \in L^{\odot 0} \cup L^{\odot 1} \cup \cdots \cup L^{\odot k}$.

M operates as follows:

Let $y = y_1 \cdots y_k$ and $q_0 * y_1 \cdots y_k *$ be the initial configuration, with $*$ the marker and q_0 the initial state.

Step 1. If $k = 0$, then y is accepted; if not, then go to Step 2.

Step 2. M marks out two sections A and B each of length k on the tape, as shown in Fig. 1.

Step 3. In part A, a number i ($1 \leqslant i \leqslant k$) is nondeterministically generated and in B a word $\bar{y}$, which is some permutation of y, is written.

Step 4. The word $\bar{y}$ in B is then nondeterministically divided into i parts $\bar{y}^1, \ldots, \bar{y}^i$, $y = \bar{y}^1 \bar{y}^2 \cdots \bar{y}^i$ (maybe some $\bar{y}^j$ are empty; $j \leqslant i$).

Step 5. Now the machine $\overline{M}$ checks whether each of $\bar{y}^1, \ldots, \bar{y}^i$ is accepted (no more space than that of B is needed). If so, then M still has to check whether $y \in L(\bar{y}^1 \odot \bar{y}^2 \odot \cdots \odot \bar{y}^i)$, i.e., whether the order of letters in parts $\bar{y}^1, \bar{y}^2, \ldots, \bar{y}^i$ is the same as in y. If this is true (which M can establish nondeterministically without using any extra space), then y is accepted by M. It should be obvious that M accepts exactly $L^{\odot}$ and uses space $S(n) = 3 \cdot n$, i.e., linear.

The construction of the machine accepting the shuffle $L_1 \odot L_2$ of languages should be obvious from the above discussion. In this case i = 2, and in Step 5 the machine M_1 accepting L_1 checks $\bar{y}^1$, and M_2 accepting L_2 checks $\bar{y}^2$.

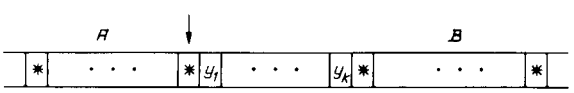

Fig. 1.

Theorem 4.2. *The class $\mathcal{PFL}$ of pseudo-flow languages is a proper subclass of the class $\mathcal{CSL}$ of context-sensitive languages.*

Proof. The proof follows from Theorem 4.1.

Besides, the context-sensitive language $\{a^n b^n c^n : n \geqslant 0\}$ is not pseudo-flow (it follows directly from the pumping lemma similarly as in Example 3.3).

To conclude, we would like to make some observations with respect to where the class $\mathcal{PFL}$ fits in the Chomsky hierarchy of languages.

(a) $\mathcal{PFL} \subsetneq \mathcal{CSL}$ (Theorem 4.2).

(b) $\mathcal{REG} \subsetneq \mathcal{PFL}$ (definition of $\mathcal{PFL}$ and trivial example of $L((ab)^{\odot})$ not regular).

(c) $\mathcal{PFL} \not\subset \mathcal{CFL}$ as the language $L((abc)^{\odot})$ is not context-free. If it were, then its intersection with a regular language

$$L\big((abc)^{\odot}\big) \cap L(a^*b^*c^*) = \{a^n b^n c^n : n \geqslant 0\}$$

would have to be context-free as well (see [1]) and it is not.

(d) $\mathcal{CFL} \not\subset \mathcal{PFL}$ as the language from Example 3.3 is obviously context-free.

(e) $\mathcal{REG} \subsetneq \mathcal{CFL} \cap \mathcal{PFL}$ as $L((ab)^{\odot})$ is not regular and is context-free and the containment follows from (b) and obvious $\mathcal{REG} \subset \mathcal{CFL}$.

References

[1] A. Aho and J. Ullman, The Theory of Parsing, Translation and Compiling (Prentice-Hall, Englewood Cliffs, NJ, 1972).

[2] T. Araki and N. Tokura, Flow languages equal recursively enumerable languages, Acta Inform. 15 (1981) 209–217.

[3] A. Ginzburg and M. Yoeli, Vector addition systems and regular languages, J. Comput. Syst. Sci. 20 (1980) 277–284.

[4] R.M. Karp and R.E. Miller, Parallel program schemata, J. Comput. Syst. Sci. 3 (1969) 147–195.

[5] D. Leonarski, Shuffle of formal languages, its properties and applications to some problems in the theory of concurrency, Unpublished manuscript, Institute of Informatics, University of Warsaw, 1982.

[6] A.C. Shaw, Software descriptions with flow expressions, IEEE Trans. Software Engrg. SE-4 (3) (1978) 242–254.