Acta Informatica 19, 85-96 (1983)

Informatica

The Universe Problem for Unrestricted Flow Languages

Kazuo Iwama

Department of Computer Sciences, Kyoto Sangyo University, Kyoto 603, Japan

Summary. Unrestricted flow languages (UFL's) are denoted by unrestricted flow expressions, namely regular expressions extended with the shuffle operator and its closure. It is shown that the universe problem for UFL's over a binary alphabet is unsolvable. "Over a binary alphabet" should be emphasized since the shuffle operator prevents us from careless application of the usual coding technique. To develop a way of mapping arbitrary unrestricted flow expressions into those over a binary alphabet while preserving as many characteristics as possible is another main purpose of this paper.

1. Introduction

Finite expressions including the shuffle operation (interleaving two strings) have been introduced by several authors aiming at the description of concurrent software, which are also of great interest from the language theoretic viewpoints [1, 2, 6, 8-11]. In spite of their superficial variety, most of them are essentially regular expressions extended with the shuffle operator, the shuffle closure operator and some synchronization facilities. In [10] such expressions are called flow expressions (FE's), and unrestricted flow expressions (UFE's) if the synchronization facilities are excluded. Languages denoted by FE's and UFE's are called flow languages (FL's) and unrestricted flow languages (UFL's), respectively.

It is known [1, 6] that the powerfulness of FE's is the highest possible, namely, FL's are recursively enumerable. The class of UFL's, however, seems somewhat peculiar: it includes some context-sensitive languages although it does not include such a simple context-free language as $\{a^n b^n | n \geqq 0\}$, it is not closed under intersection with a regular set, and so on [2, 6, 10, 11]. Thus there is a big difference between UFE's and FE's, which is intuitively due to whether the shuffle operation is carried out at random (UFE's) or is carried

Table 1. Well-known decidability of language problems

	Regular	Simple	Deterministic Two-tape	Context-free	$\mathscr{L}_{UFL}$	Context-sensitive
Emptiness	S	S	S	S	S	U
Universe	S	S	S	U	U	U
Equivalence	S	S	S	U	U	U
Inclusion	S	U	U	U	U	U

out under a certain control denoted by the synchronization facilities (FE's). Such randomness of the former clearly increases the difficulty of several problems: there is still much to be clarified [6, 11].

In this paper it is shown that the universe problem for UFL's is unsolvable (which was an open problem in [11]) and the result still holds for UFL's over a binary alphabet. The second result should be emphasized since, unlike most other formal systems, the number of available symbols plays a very important role in UFL's even if the number is two or more. (In fact, special symbols were necessary to realize the synchronization facilities [2, 6, 8–10].) The shuffle operator prevents us from careless use of the homomorphic coding technique: The shuffle of two code strings easily creates non-code strings and furthermore these non-code strings may produce irrelevant code strings by another shuffle operation. Therefore this paper has another objective, namely that of finding a way of mapping arbitrary UFE's into those over a binary alphabet while preserving as many properties as possible. The method presented in this paper might have many applications in future studies on shuffle-based systems.

The emptiness problem for UFL's is obviously solvable and the equivalence problem for UFL's is unsolvable, which was implied from the emptiness problem for FL's [11]. Note that UFL's become regular if their alphabet includes only one symbol. Table 1 summarizes well-known decidability of language problems including the present one [3–5, 7].

2. Preliminalies

Let L_1 and L_2 be languages over an alphabet Σ. Throughout this paper, the empty string is denoted by ε, the reversal of a string x in Σ^* by x^R, the concatenation of L_1 and L_2 by $L_1 \cdot L_2$ or simply $L_1 L_2$. For a string x and a symbol a, $\#_a(x)$ denotes the number of occurrences of the symbol a within x. The empty set is denoted by $\emptyset$.

The *shuffle* of L_1 and L_2, denoted by $L_1 \odot L_2$, is defined by:

$$L_1 \odot L_2 = \{x_1 y_1 x_2 y_2 \ldots x_n y_n \mid n \geq 1, x_i \text{ and } y_i \text{ in } \Sigma^*, x_1 \ldots x_n \text{ in } L_1 \text{ and } y_1 \ldots y_n \text{ in } L_2\}.$$

The *closure* of $\odot$ denoted by $^{\circledast}$ is defined by:

$$L_1^{\circledast} = \bigcup_{i=0}^{\infty} L_1^{\odot i}$$

where $L_1^{\odot 0}=\{\varepsilon\}$ and $L_1^{\odot i}=L_1^{\odot i-1}\odot L_1$ for $i\geqq 1$. When $L_1=\{x\}$ and/or $L_2=\{y\}$ for strings x and y, we often simplify notation by writing $L_1\odot y$, $x\odot L_2$, $x\odot y$ and $x^{\circledast}$ instead of $L_1\odot\{y\}$, $\{x\}\odot L_2$, $\{x\}\odot\{y\}$ and $\{x\}^{\circledast}$, respectively, and similarly for other operations.

Unrestricted flow expression (UFE) S over alphabet Σ and the *unrestricted flow language (UFL)* $L(S)$ denoted by S are defined recursively as follows:

(1) $\emptyset$ is an UFE and $L(\emptyset)=\emptyset$.

(2) ε is an UFE and $L(\varepsilon)=\{\varepsilon\}$.

(3) For each a in Σ, a is an UFE and $L(a)=\{a\}$.

(4) Let S_1 and S_2 be UFE's such that $L(S_1)=L_1$ and $L(S_2)=L_2$. Then (S_1+S_2), $(S_1\cdot S_2)$ or simply (S_1S_2), (S_1^*), $(S_1\odot S_2)$ and $(S_1^{\circledast})$ are UFE's that denote UFL's $L_1\cup L_2$, $L_1\cdot L_2$, L_1^*, $L_1\odot L_2$ and $L_1^{\circledast}$, respectively.

By the usual convention that operators $\cdot$, *, $\odot$ and $^{\circledast}$ have higher precedence than $+$, we can omit many parentheses. Also many parentheses can be omitted since operators $+$, $\cdot$ and $\odot$ are clearly associative. $\mathscr{L}_{\mathrm{UFL}}$ denotes the class of all UFL's. $\mathscr{L}_{\mathrm{CSL}}$, $\mathscr{L}_{\mathrm{CFL}}$ and $\mathscr{L}_{\mathrm{RL}}$ denotes the classes of context-sensitive, context-free and regular languages, respectively.

Basic properties of UFE's and UFL's are summarized as follows [2, 6, 10, 11]:

(1) Let S be an UFE in which no $^{\circledast}$ appears. Then $L(S)$ is a regular language. (If neither $\odot$ nor $^{\circledast}$ appears then S can be regarded as a regular expression and $L(S)$ is of course regular.)

(2) $L((ab)^{\circledast})=\{x\mid x \text{ in } \{a,b\}^*,\ \#_a(x)=\#_b(x)$ and for each prefix x_1 of x, $\#_a(x_1)\geqq\#_b(x_1)\}$, which is not regular. $L((abc)^{\circledast})$ is, as easily seen, not context-free.

(3) $\{a^nb^n\mid n\geqq 0\}$ in $\mathscr{L}_{\mathrm{CFL}}$ is not in $\mathscr{L}_{\mathrm{UFL}}$.

(4) $\mathscr{L}_{\mathrm{UFL}}$ is obviously closed under $\cup$, $\cdot$, *, $\odot$ and $^{\circledast}$. However, it is not closed under most other operations such as intersection with a regular set, homomorphsims and complement.

(5) $\mathscr{L}_{\mathrm{RL}}\subsetneqq\mathscr{L}_{\mathrm{UFL}}\subsetneqq\mathscr{L}_{\mathrm{CSL}}$. $\mathscr{L}_{\mathrm{UFL}}\mathrel{\substack{\subseteq\\ \supseteq}}\mathscr{L}_{\mathrm{CFL}}$.

An instance of Post correspondence problem (PCP) consists of two lists $A=w_1,w_2,\ldots,w_n$ and $B=x_1,x_2,\ldots,x_n$ of strings over alphabet Γ, which is said to have a solution if there is a sequence of integers $i_1,i_2,\ldots,i_m$ ($m\geqq 1$) such that

$$w_{i_1}w_{i_2}\ldots w_{i_m}=x_{i_1}x_{i_2}\ldots x_{i_m}.$$

It is well-known that PCP is unsolvable [5].

3. Unsolvability of the Universe Problem

In this section we prove the unsolvability of the universe problem for UFL's with no restriction on their alphabet size.

Theorem 1. *The universe problem for* $\mathscr{L}_{\mathrm{UFL}}(L(S)=\Sigma^*?)$ *is unsolvable.*

Proof. We use a conventional approach: Given a PCP P; $A=w_1,\ldots,w_n$ and $B=x_1,\ldots,x_n$ over Γ, we construct an UFE S over Σ such that P has a solution if and only if $L(S)\neq\Sigma^*$, where Σ is the union of Γ, $\Gamma'=\{a'|a \text{ in } \Gamma\}$, $K=\{f_1,\ldots,f_n\}$ ($K\cap\Gamma$ must be empty) and $K'=\{f_1',\ldots,f_n'\}$. When $y=a_1a_2\ldots a_m$ is a string in $(\Gamma\cup K)^*$, y' denotes the string $a_1'a_2'\ldots a_m'$ in $(\Gamma'\cup K')^*$. This notation will be frequently used without notice in this section.

Before constructing the UFE S, we divide $\Sigma^*=(\Gamma\cup\Gamma'\cup K\cup K')^*$ into several subsets. First Σ^* is divided into two disjoint regular sets R_1 and R_2 such that $R_1=L((\Gamma^*K)^*(K'\Gamma'^*)^*)$ and $R_2=\Sigma^*-R_1$. (When $C=\{c_1,\ldots,c_p\}$ is a finite set of symbols or strings, we often write C instead of $(c_1+\ldots+c_p)$ in UFE's.) Next, R_1 is divided into five (not necessarily disjoint) subsets R_{11}, R_{12}, R_{13}, R_{14} and R_{15}. Note that each string r in R_1 can be written as follows:

$$r=u_pf_{i_p}u_{p-1}f_{i_{p-1}}\ldots u_1f_{i_1}f'_{j_1}v'_1\ldots f'_{j_{q-1}}v'_{q-1}f'_{j_q}v'_q$$

where p and q are integers, u_k and v_h are in Γ^*, and f_{i_k} and f_{j_h} are in K for $1\leqq k\leqq p$ and $1\leqq h\leqq q$. Then it is clear that such a string r satisfies at least one of the following propositions $Q_1(r)$, $Q_2(r)$, $Q_3(r)$, $Q_4(r)$ and $Q_5(r)$:

$Q_1(r)$: (i) $f_{i_1}\ldots f_{i_p}=f_{j_1}\ldots f_{j_q}$ (hence it must be $p=q$),
(ii) $(u_p\ldots u_1)^R=v_1\ldots v_q$,
(iii) for all $k\leqq p$, $u_k^R=w_{i_k}$, and
(iv) for all $h\leqq q$, $v_h=x_{j_h}$. (Recall that w_{i_k} and x_{j_h} are strings of PCP P.)

$Q_2(r)$: $f_{i_1}\ldots f_{i_p}\neq f_{j_1}\ldots f_{j_q}$.

$Q_3(r)$: $(u_p\ldots u_1)^R\neq v_1\ldots v_q$.

$Q_4(r)$: For some $k\leqq p$, $u_k^R\neq w_{i_k}$.

$Q_5(r)$: For some $h\leqq q$, $v_h\neq x_{j_h}$.

Let

$$R_{1j}=\{r|r \text{ in } R_1 \text{ and } Q_j(r) \text{ is true}\}$$

for $1\leqq j\leqq 5$. Then we can obviously make the following claims:

Claim 1. $\Sigma^*=R_{11}\cup R_{12}\cup R_{13}\cup R_{14}\cup R_{15}\cup R_2$.

Claim 2. PCP P has a solution if and only if R_{11} is not empty.

Now we construct the UFE S, which consists of five subexpressions $S_{12}, S_{13}, S_{14}, S_{15}, S_2$ associated with $R_{12}, R_{13}, R_{14}, R_{15}, R_2$ above, respectively. The following lemma plays a key role.

Lemma 1. *Suppose that* $\mathrm{Pal}_{\Gamma\Gamma'}$ *denotes the set of palindromelike strings* $\{y^Ry'|y$ *in* $\Gamma^*\}$. *Then we can construct an* UFE $S_{\Gamma\Gamma'}$ *such that* $L(S_{\Gamma\Gamma'})\supseteq\Gamma^*\Gamma'^*-\mathrm{Pal}_{\Gamma\Gamma'}$ *and* $L(S_{\Gamma\Gamma'})\cap\mathrm{Pal}_{\Gamma\Gamma'}=\emptyset$. (*A similar result holds for* $\mathrm{Pal}_{KK'}$ *and* $S_{KK'}$.)

Proof. Let

$$L_1=\{a_p\ldots a_1b'_1\ldots b'_q|a_k \text{ and } b_h \text{ in } \Gamma \text{ and } p>q\geqq 0\},$$
$$L_2=\{a_p\ldots a_1b'_1\ldots b'_q|a_k \text{ and } b_h \text{ in } \Gamma \text{ and } q>p\geqq 0\},$$
$$L_3=\{a_p\ldots a_1b'_1\ldots b'_q|a_k \text{ and } b_h \text{ in } \Gamma \text{ and } a_i\neq b_i \text{ for some } i\geqq 1\}.$$

Then clearly $\Gamma^*\Gamma'^* - \mathrm{Pal}_{\Gamma\Gamma'} = L_1 \cup L_2 \cup L_3$. Let $A = \{ab' \mid a, b \text{ in } \Gamma\}$. Then we determine UFE's T_1, T_2 and T_3 as follows:

$$T_1 = \Gamma^*\Gamma A^{\circledast}.$$

$$T_2 = A^{\circledast}\Gamma'\Gamma'^*.$$

$$T_3 = \underset{a \text{ in } \Gamma,\, b' \text{ in } \Gamma',\, a \neq b}{+} (\Gamma^* a A^{\circledast} b' \Gamma'^*)^{1})$$

We can easily see that $L(T_1) \supseteq L_1$, $L(T_2) \supseteq L_2$, $L(T_3) \supseteq L_3$, and for all $i \leqq 3$, $L(T_i) \cap \mathrm{Pal}_{\Gamma\Gamma'} = \emptyset$. Hence $T_1 + T_2 + T_3$ is the desired UFE as $S_{\Gamma\Gamma'}$. □

Now we determine $S_{12}, \ldots, S_{15}$ as follows where $S_{\Gamma\Gamma'}$ and $S_{K\bar{K}'}$ are UFE's described in the lemma above, and $S_{\bar{w}_i}$ and $S_{\bar{x}_i'}$, both of which clearly exists, are UFE's for $1 \leqq i \leqq n$ such that $L(S_{\bar{w}_i}) = \Gamma^* - \{w_i^R\}$ and $L(S_{\bar{x}_i'}) = \Gamma'^* - \{x_i'\}$:

$$S_{12} = S_{K\bar{K}'} \odot (\Gamma^*\Gamma'^*),$$

$$S_{13} = S_{\Gamma\bar{\Gamma}'} \odot (K^*K'^*),$$

$$S_{14} = (\Gamma^*K)^*\Big(\underset{1 \leqq i \leqq n}{+} S_{\bar{w}_i} f_i\Big)(\Gamma^*K)^*(K'\Gamma'^*)^*,$$

$$S_{15} = (\Gamma^*K)^*(K'\Gamma'^*)^*\Big(\underset{1 \leqq i \leqq n}{+} f_i' S_{\bar{x}_i'}\Big)(K'\Gamma'^*)^*.$$

Carefully observing the expressions above we can see the following facts:

(1) Lemma 1 implies that $L(S_{12}) \supseteq R_{12}$, $L(S_{13}) \supseteq R_{13}$, and both $L(S_{12})$ and $L(S_{13})$ are disjoint with R_{11}.

(2) $L(S_{14}) \supseteq R_{14}$, $L(S_{15}) \supseteq R_{15}$, and both $L(S_{14})$ and $L(S_{15})$ are disjoint with R_{11}.

Because R_2 is regular, we can construct an UFE S_2 such that $L(S_2) = R_2$. Then, as a result, we get:

Claim 3. Let UFE $S = S_{12} + S_{13} + S_{14} + S_{15} + S_2$. Then

$$L(S) \supseteq R_{12} \cup R_{13} \cup R_{14} \cup R_{15} \cup R_2,$$

$$L(S) \cap R_{11} = \emptyset.$$

It follows from Claims 1 and 3 that $L(S) = \Sigma^* - R_{11}$, and therefore Claim 2 implies that $L(S) \neq \Sigma^*$ if and only if the given PCP P has a solution, which completes the proof. □

4. Binary Coding Suitable for UFE's/UFL's

In this section we shall strengthen Theorem 1; it is proved that the theorem remains true for UFL's over a binary alphabet, say $\{0, 1\}$. Our approach is essentially based on the binary coding method but we must be careful since UFE's include interleaving operations $\odot$ and $^{\circledast}$. We first observe why the

---

[1]) The notation means $S_1 + S_2 + \ldots + S_r$ where $S_1, S_2, \ldots, S_r$ are all UFE's of the form $(\Gamma^* a A^{\circledast} b' \Gamma'^*)$ such that a is in Γ, b' in Γ' and $a \neq b$

shuffle operation makes the discussion difficult and then present a way of overcoming such difficulties.

Let $\Delta = \{\emptyset, \varepsilon, (\ ,\ ), +, \cdot, ^*, \odot, ^{\circledast}\}$ and $\Sigma_m = \{c_1, c_2, \ldots, c_m\}$ for $m \geqq 3$. Hereafter *homomorphism* g means inplicitly that

(1) g maps $(\Delta \cup \Sigma_m)^*$ into $(\Delta \cup \{0,1\})^*$,

(2) g is one-to-one, and

(3) g satisfies that $g(a)=a$ for each a in Δ and $g(c_i)$ is in $\{0,1\}^*$ for each c_i in Σ_m.

g is naturally extended to languages such that $g(L)=\{g(x) | x \text{ in } L\}$. Let $\mathrm{CODE}(g)$ and $\overline{\mathrm{CODE}}(g)$ denote:

$$\mathrm{CODE}(g) = g(\Sigma_m^*),$$

$$\overline{\mathrm{CODE}}(g) = \{0,1\}^*\text{-}\mathrm{CODE}(g).$$

If a homomorphism g satisfying the condition

$$\text{(C1)}\quad L(g(S)) = g(L(S))$$

for every UFE S over Σ_m exists, then it would be the best for our purpose. Unfortunately, however, there are several problems:

Problem 1. There exists no homomorphism g satisfying the condition (C1) for all UFE's S over Σ_m. The reason is as follows: Let g be any homomorphism with the properties (1)–(3) above. It will be shown that we can find an UFE S over Σ_m such that $L(g(S)) \neq g(L(S))$. Since g is one-to-one and $m \geqq 3$, there is at least one symbol a in Σ_m such that $g(a)$ can be written as $g(a)=ww'$ for some strings w and w' in $\{0,1\}^*$ which satisfy $ww' \neq w'w$. (Otherwise, $g(a)$ must be of the form 0^i or 1^i for any symbol a in Σ_m, which clearly means that g is not one-to-one.) Consider UFE $a \odot a$ for such symbol a. Obviously $g(L(a \odot a)) = \{ww'ww'\}$. However, $L(g(a \odot a))(=L((ww') \odot (ww')))$ includes string $www'w'$. Since $ww' \neq w'w$, it follows that $www'w' \neq ww'ww'$, and hence $g(L(a \odot a)) \neq L(g(a \odot a))$.

Now we are forced to relax the condition (C1). Our goal is to find a homomorphism g which satisfies the condition

$$\text{(C2)}\quad L(g(S)) = g(L(S)) \cup W$$

for every UFE S over Σ_m and some $W \subseteq \overline{\mathrm{CODE}}(g)$. By the fact mentioned above, it is likely that this condition may be the best possible for our purpose.

Problem 2. Condition (C2) requires us to construct the homomorphism g such that undesirable (excess) strings created by the shuffle operation in $g(S)$ entirely drop into $\overline{\mathrm{CODE}}(g)$. Simple binary codes often used, however, do not have this property. For example, let g_1 be a homomorphism determined by $g_1(c_i)=10^i$. g_1 satisfies the condition (C2) for any regular expression S. One can easily see, however, that $L(g_1(c_2 \odot c_2)) = L((100) \odot (100))$ includes string $101000 (= g_1(c_1 c_3))$ which is not in $g_1(L(c_2 \odot c_2))$. Thus g_1 does not satisfy the condition (C2) even if S includes only one $\odot$. Let us then consider another homomorphism g_2 such

that $g_2(c_i)=10^i1$. It is somewhat better than g_1 since g_2 satisfies the condition (C2) for any UFE S including at most one $\odot$ and no $^{\circledast}$. Nevertheless g_2 is unsatisfactory for S including two or more $\odot$: Let T be UFE $(c_2 \odot c_2)\odot(c_2 c_2)$. Then $L(g_2(T))=L(((1001)\odot(1001))\odot(1001\cdot 1001))\supseteq L((10100101)\odot(10011001))\ni 1011000110001101=g_2(c_1c_3c_3c_1)\notin g_2(L(T))$.

Now we prove the following key lemma:

Lemma 2. *Let homomorphism g_0 be defined by:*

$$g_0(c_i)=11(01)^{m+i}00$$

where m is the number of symbols in Σ_m. Then g_0 satisfies the condition (C2) *for every UFE S and some $W\subseteq\overline{\mathrm{CODE}}(g_0)$.*

Proof. Let

$$\#(x)=\#_1(x)-\#_0(x)$$

for a string x in $\{0,1\}^*$ and let

$\mathrm{SCODE}=L((11(01)^{m+1}(01)^*00)^*)$,

$\mathrm{NCODE}_1=\{x \mid x$ in $\{0,1\}^*$, $\#(x_1)\geqq 0$ for all prefixes x_1 of x and $\#(x_2)\geqq 3$ for some prefix x_2 of $x\}$,

$\mathrm{NCODE}_2=\{x \mid x$ in SCODE and $(01)^{2m+1}$ appears as a substring within $x\}$,

$\mathrm{NCODE}=\mathrm{NCODE}_1\cup\mathrm{NCODE}_2$.

It will be helpful later to summarize the following properties which immediately follow from the definitions above:

(1) If x is in $L(y\odot z)$ for y and z in $\{0,1\}^*$ then $\#(x)=\#(y)+\#(z)$.

(2) $\mathrm{NCODE}\cap\mathrm{CODE}(g_0)=\emptyset$, i.e., $\mathrm{NCODE}\subseteq\overline{\mathrm{CODE}}(g_0)$, and $\mathrm{SCODE}=\mathrm{CODE}(g_0)\cup\mathrm{NCODE}_2$.

(3) If x is in SCODE then $0\leqq\#(x_1)\leqq 2$ for every prefix x_1 of x and $\#(x_1)=0$ only when x_1 ends with two consecutive 0's. Hereafter we call such a prefix, namely a prefix x_1 of x in SCODE such that x_1 is also in SCODE, a *code-prefix*.

We shall show that the following two statements are true by mathematical induction on the number of operators in UFE S.

(A) If x is in $L(g_0(S))$ then x is in $\mathrm{CODE}(g_0)\cup\mathrm{NCODE}$.

(B) If x is in $L(g_0(S))$ and x is in $\mathrm{CODE}(g_0)$ then x is in $g_0(L(S))$.

Since $g_0(L(S))\subseteq L(g_0(S))$ for every UFE S, it is sufficient to show that $g_0(L(S))\cup W\supseteq L(g_0(S))$ for some $W\subseteq\overline{\mathrm{CODE}}(g_0)$. This means that verification of statement (B) only is sufficient for the lemma. (To simplify the proof, however, we prove statement (A) at the same time.) Before the main part of the proof we need to prove the following preliminary results:

Claim 4. Suppose that y and z are in $\mathrm{CODE}(g_0)\cup\mathrm{NCODE}$ and x is any string in $L(y\odot z)$. Then

(1) if y or z is in NCODE then x is in NCODE,

(2) if both y and z are in CODE(g_0) then x is either in NCODE or in $g_0(L(y' \odot z'))$ where $y' = g_0^{-1}(y)$ and $z' = g_0^{-1}(z)$.

It should be noted that (1) and (2) are clearly equivalent to the following statements which will also be referred to:

(3) x is in CODE$(g_0) \cup$ NCODE.

(4) if x is in CODE(g_0) then y and z must be in CODE(g_0) and x is in $g_0(L(y' \odot z'))$.

Proof. To prove (1), the following two cases should be considered. Note that $\#(y_1) \geqq 0$ and $\#(z_1) \geqq 0$ for all prefixes y_1 of y and z_1 of z since y and z are in CODE$(g_0) \cup$ NCODE.

Case 1. Either y or z, say y, is in NCODE$_1$. Since x is in $L(y \odot z)$, for each prefix x_1 of x we can find prefixes y_1 of y and z_1 of z such that x_1 is in $L(y_1 \odot z_1)$. This implies that

$$\#(x_1) = \#(y_1) + \#(z_1) \geqq 0.$$

Also, for each prefix y_2 of y we can find prefixes x_2 of x and z_2 of z such that x_2 is in $L(y_2 \odot z_2)$. Since y is in NCODE$_1$, we can choose y_2 such that $\#(y_2) \geqq 3$. Then it follows that

$$\#(x_2) = \#(y_2) + \#(z_2) \geqq 3.$$

Thus x is in NCODE$_1$ and therefore in NCODE.

Case 2. Either y or z, say y, is in NCODE$_2$ and z is in CODE(g_0) or in NCODE$_2$. We can write $y = \alpha_1 \alpha_2 \ldots \alpha_p$ and $z = \beta_1 \beta_2 \ldots \beta_q$ where α_i and β_j are in $L(11(01)^{m+1}(01)^*00)$. To get a string x in $L(y \odot z)$, we can either,

(i) regard each word $\alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q$ as if it were a single symbol, which means that each prefix x_1 of x is constructed by interleaving prefixes y_1 and z_1 of y and z, respectively such that at least one of y_1 and z_1 is a code-prefix, or

(ii) construct some prefix x_2 of x by interleaving prefixes y_2 of y and z_2 of z neither of which is a code-prefix.

If way (i) is adopted then clearly x is in SCODE and substring $(01)^{2m+1}$ in y remains in x as it is. Thus x is in NCODE$_2$ and therefore in NCODE.

Suppose that way (ii) is adopted to create x, i.e., there exist prefixes x_2, y_2 and z_2 of x, y and z, respectively such that x_2 is in $L(y_2 \odot z_2)$ and neither y_2 nor z_2 is a code-prefix. Since neither y_2 nor z_2 is a code-prefix, $1 \leqq \#(y_2) \leqq 2$ and $1 \leqq \#(z_2) \leqq 2$. So it is enough to consider two cases; (a) $\#(y_2) + \#(z_2) \geqq 3$ and (b) $\#(y_2) = \#(z_2) = 1$. In case (a), it must be that $\#(x_2) \geqq 3$ and therefore x is in NCODE$_1$. In case (b), it is obvious that we can write

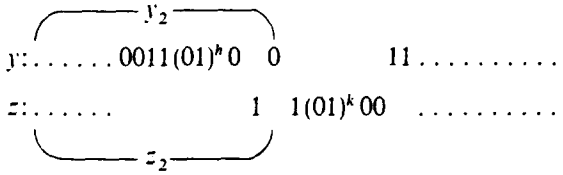
$$y_2 = y_2' 1(10)^i \quad \text{and} \quad z_2 = z_2' 1(10)^j$$

for code-prefixes y_2' and z_2', and some integers $i, j \geqq 0$.

(b.1) $i \neq 0$ and $j \neq 0$. In this case the last symbol of x_2 must be 0 since both y_2 and z_2 end with 0. Then we can write $x_2 = x_3 0$ for a prefix x_3 of x. Since $\#(x_2) = 2$, it follows that $\#(x_3) = 3$, which means that x is in NCODE$_1$.

(b.2) $i=j=0$. In this case the symbol in x immediately following x_2 must be 1 since symbol 1 immediately follows both y_2 and z_2. We can thus write $x_4 = x_2 1$ for a prefix x_4 of x and it follows that $\#(x_4)=3$. Thus x is in NCODE_1.

(b.3) One of i and j is 0 and the other is not 0, say $i\neq 0$ and $j=0$. In the previous cases, x had to include a prefix x' such that $\#(x')\geqq 3$, i.e., x could not remain within SCODE. In the present case, however, x can remain within SCODE in the following way of interleaving y and z:



One can see, in fact, that this is only one way for x to be in SCODE. Then x must include substring $(01)^{h+k+2}$. Since $h\geqq m+1$ and $k\geqq m+1$, $h+k+2\geqq 2m+4$, which means that x is in NCODE_2.

Thus we have shown statement (1) of the claim. Also in the case of statement (2) (i.e., both y and z being in $\mathrm{CODE}(g_0)$), the situation is almost the same as that in Case 2 above. If way (i) is adopted then clearly x is in $g_0(L(y'\odot z'))$ where $y'=g_0^{-1}(y)$ and $z'=g_0^{-1}(z)$. Otherwise x must be in NCODE. □

Corollary of Claim 4. Suppose that $x_1, x_2, \ldots, x_n$ $(n\geqq 3)$ are in $\mathrm{CODE}(g_0)\cup\mathrm{NCODE}$ and x is any string in $L(x_1\odot x_2\odot\ldots\odot x_n)$. Then,

(1) if at least one of the x_i's is in NCODE then x is in NCODE,

(2) if all of the x_i's are in $\mathrm{CODE}(g_0)$ then x is either in NCODE or in $g_0(L(x_1'\odot x_2'\ldots\odot x_n'))$ where $x_i'=g_0^{-1}(x_i)$ for $1\leqq i\leqq n$.

Proof. Clearly there exists $y_2, \ldots, y_{n-1}$ such that

$$
\begin{aligned}
&y_2\in L(x_1\odot x_2),\\
&y_3\in L(y_2\odot x_3),\\
&\quad\vdots\\
&y_{n-1}\in L(y_{n-2}\odot x_{n-1}),\\
&x\in L(y_{n-1}\odot x_n).
\end{aligned}
$$

Since $x_1, \ldots, x_n$ are in $\mathrm{CODE}(g_0)\cup\mathrm{NCODE}$, y_2 must be in $\mathrm{CODE}(g_0)\cup\mathrm{NCODE}$ by Claim 4 and therefore $y_3, \ldots, y_{n-1}$ and x are all in $\mathrm{CODE}(g_0)\cup\mathrm{NCODE}$ by applying Claim 4 iteratively. If x_i is in NCODE then y_i (in $L(y_{i-1}\odot x_i)$) must be in NCODE by Claim 4 and hence $y_{i+1}, \ldots, y_{n-1}$ and x must be in NCODE by applying Claim 4 iteratively.

If x is in $\mathrm{CODE}(g_0)$ then by Claim 4, y_{n-1} and x_n must both be in $\mathrm{CODE}(g_0)$, which implies that y_{n-2} and x_{n-1}, y_{n-3} and $x_{n-2}, \ldots$, y_2 and x_3, and x_1 and x_2 are all in $\mathrm{CODE}(g_0)$ by applying Claim 4 iteratively. Then by

applying Claim 4 iteratively again, we obtain the following:

$$y_2 \in g_0(L(x_1' \odot x_2')),$$
$$y_3 \in g_0(L(g_0^{-1}(y_2) \odot x_3')) \subseteq g_0(L(x_1' \odot x_2' \odot x_3')),$$
$$\vdots$$
$$x \in g_0(L(g_0^{-1}(y_{n-1}) \odot x_n')) \subseteq g_0(L(x_1' \odot x_2' \odot \ldots \odot x_n')). \quad \square$$

Now we start the mathematical induction. If UFE S includes 0 operators then statements (A) and (B) are clearly true. Then assume hypothetically that (A) and (B) are true for any UFE which includes k or less operators and consider an UFE S including $k+1$ operators. Five cases exist: $S=S_1+S_2$, $S=S_1 \cdot S_2$, $S=S_1^*$, $S=S_1 \odot S_2$ and $S=S_1^{\circledast}$, where UFE's S_1 and S_2 must have k or less operators. In the first three cases, it is straightforward; (A) and (B) are true for S from the induction hypothesis, which may be omitted.

Then suppose that $S=S_1 \odot S_2$ and x is in $L(g_0(S))$. Since $L(g_0(S)) = L(g_0(S_1)) \odot L(g_0(S_2))$, x can be written as

$$x \in L(x_1 \odot x_2) \tag{1}$$

for some

$$x_1 \in L(g_0(S_1)) \quad \text{and} \quad x_2 \in L(g_0(S_2)). \tag{2}$$

By the induction hypothesis, (2) implies that

$$x_1, x_2 \in \mathrm{CODE}(g_0) \cup \mathrm{NCODE} \tag{3}$$

so that by Claim 4, (1) and (3) imply that

$$x \in \mathrm{CODE}(g_0) \cup \mathrm{NCODE}.$$

Thus statement (A) has been verified. Suppose in addition that

$$x \in \mathrm{CODE}(g_0). \tag{4}$$

By Claim 4, above (1), (3) and (4) imply that

$$x_1, x_2 \in \mathrm{CODE}(g_0) \tag{5}$$

and

$$x \in g_0(L(x_i' \odot x_2')) \tag{6}$$

where $x_1' = g_0^{-1}(x_1)$ and $x_2' = g_0^{-1}(x_2)$. By the induction hypothesis, (2) and (5) imply that x_1 is in $g_0(L(S_1))$ and x_2 is in $g_0(L(S_2))$, which means that

$$x_1' = g_0^{-1}(x_1) \in L(S_1) \quad \text{and} \quad x_2' = g_0^{-1}(x_2) \in L(S_2). \tag{7}$$

Hence (6) and (7) imply that x is in $g_0(L(x_1' \odot x_2')) \subseteq g_0(L(S_1) \odot L(S_2)) = g_0(L(S_1 \odot S_2)) = g_0(L(S))$. Thus statement (B) is also true.

Now consider the last case, $S=S_1^{\circledast}$. Suppose that x is in $L(g_0(S)) = L(g_0(S_1^{\circledast}))$. Then by the definition x is in $L(g_0(S_{11} \odot S_{12} \odot \ldots \odot S_{1n}))$ for S_{11}

$=S_{12}=\ldots=S_{1n}=S_1$ and some integer $n\geqq 0$. This means that x is in $L(x_1 \odot x_2 \odot \ldots \odot x_n)$ for some $x_1, x_2, \ldots, x_n$ in $L(g_0(S_1))$. If $n\leqq 2$ then there is nothing new to prove. so that we may assume that $n\geqq 3$. Then we can use Corollary of Claim 4. We may omit the description of the induction step since it is almost the same as in the preceding case. □

Corollary of Lemma 2. *UFE's S_1 and S_2 over Σ_m are equivalent (i.e., $L(S_1)=L(S_2)$) if and only if*

$$L(g_0(S_1))\cup\overline{\mathrm{CODE}}\,(g_0)=L(g_0(S_2))\cup\overline{\mathrm{CODE}}\,(g_0).$$

Let $S_1=S$ and $S_2=(c_1+c_2+\ldots+c_m)^*$. Then the corollary above says that $L(S)=\Sigma_m^*$ if and only if

$$L(g_0(S))\cup\overline{\mathrm{CODE}}\,(g_0)=L(g_0((c_1+\ldots+c_m)^*))\cup\overline{\mathrm{CODE}}\,(g_0).$$

Since $\overline{\mathrm{CODE}}\,(g_0)$ is regular, there exists an UFE $S_{\overline{\mathrm{CODE}}\,(g_0)}$ over $\{0,1\}$ such that $L(S_{\overline{\mathrm{CODE}}\,(g_0)})=\overline{\mathrm{CODE}}\,(g_0)$ and note that $L(g_0((c_1+\ldots+c_m)^*))$ clearly includes $g_0(\Sigma_m^*)=\mathrm{CODE}(g_0)$. Then it follows that $L(S)=\Sigma_m^*$ if and only if

$$L(g_0(S)+S_{\overline{\mathrm{CODE}}\,(g_0)})=\{0,1\}^*$$

which means that the universe problem for UFL's is solvable if the same problem is solvable for UFL's over a binary alphabet. Thus we can obtain our main theorem from Theorem 1.

Theorem 2. *The universe problem for UFL's over a binary alphabet is unsolvable.*

Lemma 2 might have many applications, one of which concerns the synchronization facilities of flow expressions (FE's) [10]. FE's are summarized as follows: A FE S is an UFE over $\Sigma\cup\Omega_k$ where Σ is a finite set of *entity symbols* and $\Omega_k=\{\omega_1,\sigma_1,\ldots,\omega_k,\sigma_k\}$ is a set of $2k$ *wait/signal symbols.* Let language

$$U_k=L((\sigma_1\omega_1+\sigma_1)^*\odot(\sigma_2\omega_2+\sigma_2)^*\odot\ldots\odot(\sigma_k\omega_k+\sigma_k)^*).$$

Then the flow language $\hat{L}_k(S)$ ($\subseteq\Sigma^*$) denoted by the FE S is defined by:

$$\hat{L}_k(S)=\{x_1\ldots x_n \mid y_0 x_1 y_1 x_2 \ldots x_n y_n \text{ in } L(S),\ x_1,\ldots,x_n \text{ in } \Sigma^*,$$
$$y_0,\ldots,y_n \text{ in } \Omega_k^* \text{ and } y_0 y_1 \ldots y_n \text{ in } U_k\}.$$

It is known [1, 6] that for any recursively enumerable (r.e.) language W over Σ, there exists an integer k and a FE S such that $W=\hat{L}_k(S)$. Furthermore, as for the value of the integer k. 10 at most is sufficient for any r.e. language W [2].

Since U_k is a regular set, $\hat{L}_k(S)$ can be defined alternatively using a mapping defined by a deterministic sequential machine with accepting states (DSMA). Let M_k be the following DSMA: (i) M_k accepts all strings in $L(U_k\odot\Sigma^*)$. (ii) For each input symbol a in $\Sigma\cup\Omega_k$ which M_k reads in its one-step move, M_k emits a itself if a is in Σ and emits ε otherwise. Then it is clear that

$$\hat{L}_k(S)=M_k(L(S)).$$

Let $\mathscr{L}_{RE}^{m}$ ($\mathscr{L}_{UFL}^{m}$) be the class of all r.e. languages (UFL's) over Σ_m. Then by [2] above and the usual binary coding method, we can get:

Claim 5. For any integer m, there exists a DSMA M such that $\mathscr{L}_{RE}^{m} = M(\mathscr{L}_{UFL}^{22})$.

Lemma 2 can be applied to this result. Let f be the homomorphism from $(\Sigma_2 \cup \Omega_{10})^* \to \{0,1\}^*$ described in the lemma and consider a DSMA M such that (i) M accepts $f(L(U_{10} \odot \Sigma_2^*))$ which is a regular set, (ii) for each sequence $f(c)$ (c in $\Sigma_2 \cup \Omega_{10}$) of input symbols most recently read, M emits c if c is in Σ_2, and emits ε otherwise. Thus we can see that there exists a DSMA such that $\mathscr{L}_{RE}^{2} = M(\mathscr{L}_{UFL}^{2})$. The following theorem immediately follows:

Theorem 3. *For any integer m, there exists a DSMA M such that* $\mathscr{L}_{RE}^{m} = M(\mathscr{L}_{UFL}^{2})$.

Acknowledgement. The author wishes to thank Professor Shuzo Yajima and Professor Yahiko Kambayashi of Kyoto University for their valuable discussion.

References

1. Araki, T., Tokura, N.: Flow languages equal recursively enumerable languages. Acta Informat. **15**, 209-217 (1981)
2. Araki, T., Tsujino, Y., Tokura, N.: Descriptive power of flow expressions and event expressions. Trans. Inst. Electronics Commun. Engrg. Japan **J63-D**, 658-665 (1980)
3. Bird, M.: The equivalence problem for deterministic two-tape automata. J. Comput. Syst. Sci. **7**, 218-236 (1973)
4. Friedman, E.P.: The inclusion problem for simple languages. Theor. Comput. Sci. **1**, 297-316 (1976)
5. Hopcroft, J.E., Ullman, J.D.: Formal languages and their relation to automata. Addison-Wesley, 1969
6. Jantzen, M.: The power of synchronizing operations on strings. Theor. Comput. Sci. **14**, 127-154 (1981)
7. Korenjak, A.J., Hopcroft, J.E.: Simple deterministic languages. Proc. IEEE Symp. on Switching and Automata Theory, pp. 36-46, 1866
8. Ogden, W.E., Riddle, W.E., Rounds, W.C.: Complexity of expressions allowing concurrency. Proc. ACM Symp. on Principles of Programming Languages, pp. 185-194, 1978
9. Riddle, W.E.: An approach to software system behaviour description. Comput. Lang. **4**, 29-47 (1979)
10. Shaw, A.C.: Software description with flow expressions. IEEE Trans. Software Engrg. **SE-4**, 242-254 (1978)
11. Araki, T., Tokura, N.: Decision problems for regular expressions with shuffle and shuffle closure operators. Trans. Inst. Electronics Commun. Engrg. Japan **J64-D**, 1069-1073 (1981)

Received August 12, 1981/October 12, 1982