数理解析研究所講究録
第 796 巻 1992 年 1-7

A P-Complete Language Describable with Iterated Shuffle

Takayoshi Shoudai
Department of Control Engineering and Science
Kyushu Institute of Technology
Iizuka 820, Japan

Abstract

We show that a P-complete language can be described as a single expression with the shuffle operator, shuffle closure, union, concatenation, Kleene star and intersection on a finite alphabet.

1 Introduction

In this paper, we construct a P-complete language by using shuffle operator $\triangle$, iterated shuffle $\dagger$, union $\cup$, concatenation $\cdot$, Kleene star $*$ and intersection $\cap$ over a finite alphabet. The shuffle operator was introduced by [10] to describe the class of flow expressions. Formal properties of expressions with these operators have been extensively studied from various points in the literatures [2, 3, 4, 5, 8, 9, 10, 11].

It is known that the complexity of almost classes of languages can be increased by using the iterated shuffle operator. For example, there are two deterministic context-free languages L_1 and L_2 such that $L_1 \triangle L_2$ is NP-complete [9]. Moreover, by allowing the synchronization mechanisms, any recursively enumerable set can be described [1, 3].

In [2, 11], by using the shuffle and iterated shuffle operators together with $\cup, \cdot, *, \cap$, an NP-complete language is described. We employ the same set of operators to describe our P-complete language. In the proof of P-completeness, the intersection operator plays an important role to make the language polynomial-time recognizable. However, we do not know whether the intersection operator is necessary to define a P-complete language as in the case with NP-complete [2, 11].

Recently, P-complete problems have received considerable attentions since they do not seem to allow any efficient parallel algorithms [7]. This paper gives a P-complete problem of a new kind, which is described by a single expression.

2 Preliminaries

Let Σ be a finite alphabet and Σ^* be $\{a_1 \cdots a_n \mid a_i \in \Sigma \text{ for } i = 1, \ldots, n \text{ and } n \geq 0\}$. A subset of Σ^* is called a *language*.

Definition 1 For languages L, L_1 and L_2, we define the *shuffle operator* $\triangle$, the *iterated shuffle* $\dagger$ and operators, $\cdot$,$*$,$+$ as follows:
(1) $L_1 \triangle L_2 = \{x_1y_1x_2y_2 \cdots x_my_m \mid x = x_1x_2 \cdots x_m \in L_1, y = y_1y_2 \cdots y_m \in L_2 \text{ and } x_i, y_i \in \Sigma^* \text{ for } i = 1, \ldots, m\}$ (shuffle operator).
(2) $L^\dagger = \{\varepsilon\} \cup L \cup (L \triangle L) \cup (L \triangle L \triangle L) \cup \cdots$ (iterated shuffle).
(3) $L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$ (abbreviated to L_1L_2).
(4) $L^* = \{\varepsilon\} \cup L \cup (L \cdot L) \cup (L \cdot L \cdot L) \cdots$.
(5) $L^+ = L \cdot L^*$.

We identify a language $\{w\}$ which consists of only one word with w. Thus, we will denote $\{w\}^*, \{w\}^+, \{w\}^\dagger, \ldots$ by $w^*, w^+, w^\dagger$, respectively.

As the basis of our reduction, we use the circuit value problem (CVP) that was shown P-complete [6]. Our definition in this paper slightly different from one in [6].

CIRCUIT VALUE PROBLEM (CVP)
INSTANCE: A circuit $C = (C_1, \ldots, C_m, C_{m+1}, \ldots, C_n)$, where each C_i is either (i) $C_i = \mathit{true}$ or false $(1 \leq i \leq m)$, (ii) $C_i = \mathrm{NOR}(C_j, C_k)$ $(m+1 \leq i \leq n$ and $j, k < i)$.
PROBLEM: Decide whether the value of C_n is *true*.

In the following section, CVP represents the set of all circuits whose output is *true*.

Let Σ be a finite alphabet, $v_1, v_2, \ldots, v_m$ be symbols where $v_i \in \Sigma$ for $i = 1, \ldots, m$ and $w_1, w_2, \ldots, w_{m+1}$ be words on the alphabet $\Sigma - \{v_1, v_2, \ldots, v_m\}$. By using the iterated shuffle operator, the language $\{{v_1}^n {v_2}^n \cdots {v_m}^n \mid n \geq 1\}$ can be described as $(v_1v_2 \cdots v_m)^\dagger \cap {v_1}^+ {v_2}^+ \cdots {v_m}^+$. Moreover, we can represent $\{w_1 {v_1}^n w_2 {v_2}^n \cdots w_m {v_m}^n w_{m+1} \mid n \geq 1\}$ as

$$(w_1w_2 \cdots w_{m+1} \triangle (v_1v_2 \cdots v_m)^\dagger) \cap w_1{v_1}^+ w_2 {v_2}^+ \cdots w_m {v_m}^+ w_{m+1}.$$

We often use this form of languages to define our P-complete language. Whenever such languages are used in the next section, we will not describe them explicitly by using the shuffle operator and the iterated shuffle.

3 A P-complete language

The main result in this paper is the following theorem.

Theorem 1 A P-complete language can be described with operators $\cdot, *, \cup, \cap, \triangle, \dagger$.

3.1 Definition of the language

We will describe a P-complete language $\mathcal{L}$ with the alphabet $\Sigma = \{0, 1, a, b, u, v, x, y, z\}$. This language is defined stepwise.

At first, a language L is defined as follows:

$$
\begin{array}{rcl}
L_a & = & a^+0 \cup a^+1 = \{a^i\beta \mid i \geq 1 \text{ and } \beta \in \{0,1\}\}. \\
L_{bba} & = & (b^+1b^+1a^+0) \cup (b^+0b^+1a^+1) \cup (b^+1b^+0a^+1) \cup (b^+0b^+0a^+1) \\
 & = & \{b^j\beta' b^k \beta'' a^i \beta \mid i, j, k \geq 1 \text{ and } (\beta', \beta'', \beta) \in \{(1,1,0), (0,1,1), (1,0,1), (0,0,1)\}\} \\
L_b & = & b^+1 = \{b^i 1 \mid i \geq 1\}. \\
\hline
L & = & {L_a}^+ {L_{bba}}^+ L_b.
\end{array}
$$

The following language T (resp. F) is used for a distribution of *true* (resp. *false*) value.

$$
\begin{array}{rcl}
T_x & = & \{1zx^iu^i \mid i \geq 1\}, \qquad T_y = \{1y^iv^i \mid i \geq 1\}. \\
T_{xy} & = & \{1zx^iu^i1y^iv^i \mid i \geq 1\}, \quad T_{yy} = \{1y^iv^i1y^iv^i \mid i \geq 1\}. \\
\hline
T_{odd} & = & T_{xy}{T_{yy}}^*T_y \cap T_x{T_{yy}}^* = \{1zx^iu^i(1y^iv^i)^j \mid i \geq 1, j \geq 1 \text{ and } j \text{ is odd.}\}. \\
T_{even} & = & T_x{T_{yy}}^*T_y \cap T_{xy}{T_{yy}}^* = \{1zx^iu^i(1y^iv^i)^j \mid i \geq 1, j \geq 1 \text{ and } j \text{ is even.}\}. \\
\hline
T & = & T_x \cup T_{odd} \cup T_{even} = \{1zx^iu^i(1y^iv^i)^j \mid i \geq 1 \text{ and } j \geq 0\}.
\end{array}
$$

F is defined in a similar way by simply replacing a symbol with 0 in the definition of 1.

$$F \quad = \quad \{0zx^iu^i(0y^iv^i)^j \mid i \geq 1 \text{ and } j \geq 0\}.$$

Subwords $1y^iv^i$ (resp. $0y^iv^i$) of a word in T (resp. F) are combined with b^i0 (resp. b^i1) of words in L and determines the value of the ith variable. These three languages L, T and F are combined one another by using the shuffle operator and the iterated shuffle.

$$\mathcal{J} = L\triangle(T \cup F)^\dagger.$$

A language $\mathcal{K}$ is used to make our language $\mathcal{L}$ polynomial time decidable. We construct the language $\mathcal{K}$ stepwise as follows:

$$
\begin{aligned}
A_{11} &= \{a^i 11 z x^i u^i \mid i \geq 1\}. \\
A_{00} &= \{a^i 00 z x^i u^i \mid i \geq 1\}. \\
A_{01} &= \{a^i 01 z x^i u^i \mid i \geq 1\}.
\end{aligned}
$$

In a similar way, the following languages are defined:

$$
\begin{aligned}
B_{01} &= \{b^i 01 y^i v^i \mid i \geq 1\}. \\
B_{11} &= \{b^i 11 y^i v^i \mid i \geq 1\}. \\
\hline
M &= (A_{11} \cup A_{00})^{+} (B_{01} B_{01} A_{01})^{+} B_{11}.
\end{aligned}
$$

The language M contains a word w in which zx^iu^i occurs more than once in w for some i, where zx^iu^i corresponds to the ith gate. We will remove such words w from M so that each zx^iu^i occurs exactly once for all $1 \leq i \leq n$.

$$
\begin{aligned}
N_z &= (zxuzx^2u^2 \Delta (xuxu)^{\dagger}) \cap (zx^{+}u^{+}zx^{+}u^{+}) = \{zx^iu^izx^{i+1}u^{n+1} \mid i \geq 1\}. \\
\hline
N_{odd} &= zxuN_z{}^{*} \cap N_z{}^{*} zx^{+}u^{+} = \{zxuzx^2u^2 \cdots zx^iu^i \mid i \geq 1 \text{ and } i \text{ is odd.}\}. \\
N_{even} &= zxuN_z{}^{*} zx^{+}u^{+} \cap N_z{}^{*} = \{zxuzx^2u^2 \cdots zx^iu^i \mid i \geq 1 \text{ and } i \text{ is even.}\}. \\
\hline
N &= N_{odd} \cup N_{even} = \{zxuzx^2u^2 \cdots zx^iu^i \mid i \geq 1\}.
\end{aligned}
$$

Then, we define the language $\mathcal{K}$ which will be used for allowing a language $\mathcal{J}$ to be in P.

$$\mathcal{K} = M \cap (N \Delta \Sigma'^{*}), \text{ where } \Sigma' = \Sigma - \{u, x, z\}.$$

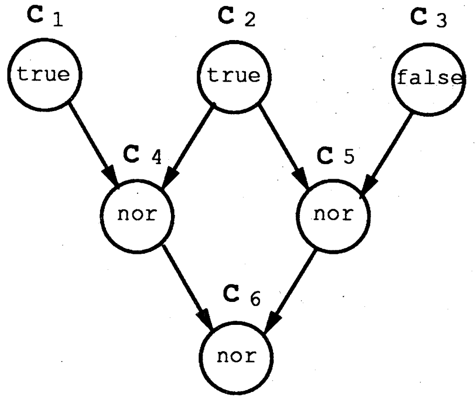
Finally, we defined the language $\mathcal{L}$ as follows:

$$\mathcal{L} = \mathcal{J} \cap \mathcal{K}.$$

3.2 Proof of the P-completeness

Theorem 1 follows from the next lemma.

Lemma 1 $\mathcal{L}$ is log-space equivalent to CVP, i.e., $\mathcal{L}$ is log-space reducible from CVP and CVP is log-space reducible from $\mathcal{L}$.



$$\begin{aligned} w \;=\;& a11zxua^2 11zx^2u^2a^3 00zx^3u^3b01yvb^2 01y^2v^2a^4 01zx^4u^4 \\ & b^2 01y^2v^2b^3 01y^3v^3a^5 01zx^5u^5b^4 01y^4v^4b^5 01y^5v^5a^6 01zx^6u^6b^6 11y^6v^6. \end{aligned}$$

Figure 1: This above circuit is transformed to the word w.

Proof. We will define a function f from CVP to Σ^*. f is a function which transforms $C = (C_1, \ldots, C_n) \in$ CVP to $f(C) = w_1 \cdots w_n w_{n+1} \in \Sigma^*$, where

$$w_i = \begin{cases} a^i 11zx^iu^i & (C_i = true) \\ a^i 00zx^iu^i & (C_i = false) \\ b^j 01y^jv^jb^k 01y^kv^ka^i 01zx^iu^i & (C_i = \mathrm{NOR}(C_j, C_k)) \\ b^n 11y^nv^n & (i = n+1). \end{cases}$$

It is easy to see that this function is computable in log-space.

We show following two claims.

Claim 1. $f(C) \in \mathcal{L}$ for every $C \in$ CVP.

Proof. Let $w = w_1 \cdots w_m w_{m+1} \cdots w_n w_{n+1}$ be a word transformed from some n-gates instance $C = (C_1, \ldots, C_m, C_{m+1}, \ldots, C_n)$ where C_i is an *input* gate for $1 \le i \le m$, an NOR gate for $m+1 \le i \le n$ and an output of this circuit is *true*. Let $\beta_i = 1$ (resp. $\beta_i = 0$) if the value of C_i is *true* (resp. *false*) for $1 \le i \le n$.

According to $B = (\beta_1, \ldots, \beta_n)$, we divide w_i into two words w_i' and w_i'' as follows:

(1) For $i = 1, \ldots, m$, $w_i' = a^i\beta_i$, $w_i'' = \beta_i zx^iu^i$.

(2) For $i = m+1, \ldots, n$, $w_i' = b^j\bar{\beta}_j b^k\bar{\beta}_k a^i\bar{\beta}_i$, $w_i'' = \beta_j y^jv^j\beta_k y^kv^k\beta_i zx^iu^i$.

We note that w_i' is in L_{bba} since $C_i = \mathrm{NOR}(C_j, C_k)$.

(3) $w_{n+1}' = b^n 1$, $w_{n+1}'' = 1y^nv^n$.

It is easy to see that a word $w' = w_1' \cdots w_{n+1}'$ is in $L = L_a{}^+ L_{bba}{}^+ L_b$.

On the other hand, since $w'' = w_1'' \cdots w_{n+1}''$ is constructed with subwords of the form $\beta_i z x^i u^i$ or $\beta_i y^i v^i$ and for each NOR gate, input gate numbers of this gate are always smaller than its number, we can describe the word w'' as word in $t_1 \Delta t_2 \Delta \cdots \Delta t_n$, where $t_i = \beta_i z x^i u^i \beta_i y^i v^i \cdots \beta_i y^i v^i$. Since $t_i \in T$ or F for $i = 1, \ldots, n$, $f(C) = w_1 \cdots w_m w_{m+1} \cdots w_n w_{n+1}$ is in $w' \Delta t_1 \Delta \cdots \Delta t_n \subset L \Delta (T \cup F)^\dagger = \mathcal{L}$. □

Since every word w of $\mathcal{L}$ is contained in M, w is of the form $w = w_1 \cdots w_m w_{m+1} \cdots w_n w_{n+1}$, where, for $i = 1, \ldots, n+1$,

$$w_i = \begin{cases} a^{\ell_i} \beta_i \beta_i z x^{\ell_i} u^{\ell_i} & (1 \le i \le m, \beta_i \in \{0,1\}) \\ b^{\ell_i'} 01 y^{\ell_i'} v^{\ell_i'} b^{\ell_i''} 01 y^{\ell_i''} v^{\ell_i''} a^{\ell_i} 01 z x^{\ell_i} u^{\ell_i} & (m+1 \le i \le n) \\ b^{\ell_{n+1}} 11 y^{\ell_{n+1}} v^{\ell_{n+1}} & (i = n+1) \end{cases}$$

We transform a word $w \in \mathcal{L}$ to a circuit $C = (C_1, \ldots, C_m, C_{m+1}, \ldots, C_n)$ as follows:

(1) For $i = 1, \ldots, m$, if $\beta_i = 1$ then $C_i = true$ else $C_i = false$.

(2) For $i = m+1, \ldots, n$, $C_i = \mathrm{NOR}(C_j, C_k)$ where $j = \ell_i'$ and $k = \ell_i''$.

It is easy to see that this transformation, say g, is a well-defined function computable in log-space.

Claim 2. $g(w) \in$CVP for every $w \in \mathcal{L}$.

Proof. For $w \in \mathcal{L}$, let w'' be the word obtained by dropping off the contribution from L. Then w'' is in $(T \cup F)^\dagger$ and has the form $c_1 c_2 \cdots c_{3n-2m+1}$ where $c_r = \beta_r z x^{p_r} u^{p_r}$ or $\beta_r y^{p_r} v^{p_r}$ ($\beta_r \in \{0,1\}, p_r \ge 1$ and $1 \le r \le 3n-2m+1$). Since w'' contains n z's, there exist n words $t_1, t_2, \ldots, t_n \in L \cup F$ such that w'' is in $t_1 \Delta t_2 \Delta \cdots \Delta t_n$. It is easy to see that each c_r ($1 \le r \le 3n-2m+1$) is a subword of some t_i ($1 \le i \le n$). Thus, without loss of generality, we may assume that for each $i = 1, \ldots, n$, t_i is of the form $\beta_i z x^i u^i \beta_i y^i v^i \cdots \beta_i y^i v^i$ ($\beta_i \in \{0,1\}$). Since w'' is also in $N \Delta \Sigma'^*$ and for $1 \le i \le n$, a subword $\beta_i y^i v^i$ of w'' does not occur before a subword $\beta_i z x^i u^i$ of w'', we have $j, k < i$.

We claim that for $i = 1, \ldots, n$, $t_i \in T$ if and only if the value of C_i is *true*. This is shown by the induction. For $i = 1, \ldots, m$, if $\beta_i = 1$, then t_i must be in T. Thus, by definition of g, $C_i = true$. For $i \ge m+1$, suppose that for $j, k < i$, this claim is true. We only discuss the case of $t_j \in T$ and $t_k \in T$. By the assumption, the values of C_j and C_k are *true*. We remove contributions of t_j and t_k from w_i. The remaining word is $b^j 0 b^k 0 a^i 01 z x^i u^i$. Moreover, w_i must has a contribution from L_{bba}. This contribution must be of the form $b^j 0 b^k 0 a^i 1$. Thus, the remaining word after removing this contribution is $0 z x^i u^i$. Therefore, t_i must be in F. On the other hand, the value of $C_i = \mathrm{NOR}(C_j, C_k)$ is $false$. Other case is shown in a similar way. Thus, this claim holds.

Since t_n must be in T, the value of C_n is *true*. Thus $g(w) \in CVP$. □

By the discussion above, we can say that $\mathcal{L}$ is log-space reducible to CVP via f and CVP has a log-space reduction g (inverse of f) from $\mathcal{L}$. □

References

[1] T. Araki and N. Tokura, Flow languages equal recursively enumerable languages, *Acta Informat.* **15** (1978) 209-217.

[2] T. Hayashi and S. Miyano, Flow expressions and complexity analysis, *Reports of WGSF Meeting of Information Processing Society of Japan* SF2-3 (1982) 1-10.

[3] M. Jantzen, The power of synchronizing operations on strings, *Theoret. Comput. Sci.* **14** (1981) 127-154.

[4] M. Jantzen, Extending regular expressions with iterated shuffle, *Theoret. Comput. Sci.* **38** (1985) 223-247.

[5] J. Jedrzejowicz, On the enlargement of the class of regular languages by the shuffle closure, *Inf. Process. Lett.* **16** (1983) 51-54.

[6] R.E. Ladner, The circuit value problem is log space complete for P, *SIGACT News* **7** (1975) 18-20.

[7] S. Miyano, S. Shiraishi and T. Shoudai, A list of P-complete problems, RIFIS-TR-CS-17, Research Institute of Fundamental Information Science, Kyushu University, 1989 (revised in December, 1990).

[8] M. Nivat, Behaviors of processes and synchronized systems of processes, Lecture note at Marktoberdopf NATO Summer School 1981.

[9] W.F. Ogden, W.E. Riddle and W.C. Rounds, Complexity of expressions allowing concurrency, *Proc. 5th Annual ACM Symposium on Principles of Programming Languages* (1978) 185-194.

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

[11] M.K. Warmuth and D. Haussler, On the complexity of iterated shuffle, *J. Comput. Syst. Sci.* **28** (1984) 345-358.