A P-Complete Language Describable with Iterated Shuffle

Shoudai, Takayoshi

数理解析研究所講究録 1992(796):1-7

URL: http://hdl.handle.net/2433/82766

Type: Departmental Bulletin Paper

Kyoto University
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 $L_1 \Delta L_2$ 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 \( \Sigma \) be a finite alphabet and \( \Sigma^* \) 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 \( \Sigma^* \) 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, \( *, + \) 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. \( L^\dagger = \{\epsilon\} \cup L \cup (L \triangle L) \cup (L \triangle L \triangle L) \cup \cdots \) (iterated shuffle).
3. \( L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\} \) (abbreviated to \( L_1 L_2 \)).
4. \( L^* = \{\epsilon\} \cup L \cup (L \cdot L) \cup (L \cdot L \cdot L) \cdots \). (4)
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 \( w^*, w^+, w^\dagger, \ldots \) 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) \( C_i = \text{true} \) or \( \text{false} \) \((1 \leq i \leq m)\), (ii) \( C_i = \text{NOR}(C_j, C_k) \) \((m + 1 \leq i \leq n\text{ and } j, k < i)\).

*Problem:* Decide whether the value of \( C_n \) is \text{true}.

In the following section, CVP represents the set of all circuits whose output is \text{true}.

Let \( \Sigma \) 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 \( (v_1 v_2 \cdots v_m)^\dagger \cap v_1^+ v_2^+ \cdots v_m^+ \).

Moreover, we can represent \( \{w_1 w_2 \cdots w_{m+1} \triangle (v_1 v_2 \cdots v_m)^\dagger \cap w_1 v_1^+ w_2 v_2^+ \cdots w_m v_m^+ w_{m+1} \mid n \geq 1\} \) as

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

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 *,+,∪,∩,△,†.

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:

$L_a = a^+0 ∪ a^+1 = \{a^iβ | i ≥ 1 \text{ and } β ∈ \{0, 1\}\}$

$L_{bba} = (b^+b^+1a^+0) ∪ (b^+0b^+1a^+1) ∪ (b^+1b^+0a^+1) = \{b^iβ^\prime b^jβ^\prime' a^kβ | i, j, k ≥ 1 \text{ and } (β^\prime, β^\prime', β) ∈ \{(1, 1, 0), (0, 1, 1), (1, 0, 1), (0, 0, 1)\}\}$

$L_b = b^+1 = \{b^i1 | i ≥ 1\}$.

$L = L_a^+L_{bba}^+L_b$.

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

$T_x = \{1zx^\ast u^i | i ≥ 1\}$, $T_y = \{1y^i v^i | i ≥ 1\}$.

$T_{xy} = \{1zx^i u^1 y^j v^i | i ≥ 1, j ≥ 1\}$, $T_{yy} = \{1y^i v^i 1y^i v^i | i ≥ 1\}$.

$T_{odd} = T_x T_y T_y^* T_y ∩ T_x T_y T_y^* = \{1zx^i u^i (1y^j v^i)^j | i ≥ 1, j ≥ 1 \text{ and } j \text{ is odd.}\}$

$T_{even} = T_x T_y T_y^* T_y ∩ T_x T_y T_y^* = \{1zx^i u^i (1y^j v^i)^j | i ≥ 1, j ≥ 1 \text{ and } j \text{ is even.}\}$

$T = T_x ∪ T_{odd} ∪ T_{even} = \{1zx^i u^i (1y^j v^i)^j | i ≥ 1 \text{ and } j ≥ 0\}$.

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

$F = \{0zx^i u^i (0y^j v^i)^j | i ≥ 1 \text{ and } j ≥ 0\}$.

Subwords $1y^i v^i$ (resp. $0y^i v^i$) 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 $i$th variable. These three languages $L, T$ and $F$ are combined one another by using the shuffle operator and the iterated shuffle.

$\mathcal{J} = L△(T ∪ F)^\dagger$. 
A language $\mathcal{K}$ is used to make our language $\mathcal{L}$ polynomial time decidable. We construct the language $\mathcal{K}$ stepwise as follows:

\[
\begin{align*}
A_{11} &= \{a^i1z^i1u^i \mid i \geq 1\}, \\
A_{00} &= \{a^i00z^i1u^i \mid i \geq 1\}, \\
A_{01} &= \{a^i01z^i1u^i \mid i \geq 1\}.
\end{align*}
\]

In a similar way, the following languages are defined:

\[
\begin{align*}
B_{01} &= \{b^i01y^i \mid i \geq 1\}, \\
B_{11} &= \{b^i11y^i \mid i \geq 1\}.
\end{align*}
\]

\[
M = (A_{11} \cup A_{00})^{+}(B_{01}B_{01}A_{01})^{+}B_{11}.
\]

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

\[
\begin{align*}
N_z &= (zzuxz^2u^2 \Delta(xuxu)^{i}) \cap (zz^i u^i zz^{i+1} u^{i+1} \mid i \geq 1), \\
N_{odd} &= zxu N_z^* \cap N_z^* \Delta u^i = \{zz^2u^2 \cdots zz^i u^i \mid i \geq 1 \text{ and } i \text{ is odd.}\}, \\
N_{even} &= zxu N_z^* zz^i u^i \cap N_z^* = \{zz^2u^2 \cdots zz^i u^i \mid i \geq 1 \text{ and } i \text{ is even.}\}, \\
N &= N_{odd} \cup N_{even} = \{zz^2u^2 \cdots zz^i u^i \mid i \geq 1\}.
\end{align*}
\]

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

\[
\mathcal{K} = M \cap (N \Delta \Sigma' \ast), \text{ where } \Sigma' = \Sigma - \{u, x, z\}.
\]

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 $\mathcal{L}$. 
Figure 1: This above circuit is transformed to the word $w$.

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

$$w = a^{i}11zxua^{i}11zx^{2}u^{2}a^{3}00zx^{3}u^{3}b01yvb^{2}01y^{2}v^{2}a^{4}01zx^{4}u^{4}$$

$$b^{2}01y^{2}v^{2}b^{3}01y^{3}v^{3}a^{5}01zx^{5}u^{5}b^{4}01y^{4}v^{4}b^{5}01y^{5}v^{5}a^{6}01zx^{6}u^{6}b^{6}11y^{6}v^{6}.$$ 

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 $C \in \text{CVP}$.

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 $1 \leq i \leq m$, an NOR gate for $m+1 \leq i \leq n$ 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 $1 \leq i \leq n$.

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

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

(2) For $i = m + 1, \ldots, n$, $w_i' = b^i \beta_j b^j \beta_k a^i \beta_i$, $w_i'' = \beta_j y^i v^j \beta_k y^k v^k \beta_i zx^i u^i$.

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

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


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 $\beta_i z z_i^i u_i$ or $\beta_i y_i^i v_i$ 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 $t_i = \beta_i z z_i^i u_i \beta_i y_i^i v_i \ldots \beta_i y_i^i v_i$.

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 $w' \Delta t_1 \Delta \cdots \Delta t_n \subseteq L \Delta (T \cup F)^\dagger = L$.

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

$$w_i = \begin{cases} a^i \beta_i z z_i^i u_i & (1 \leq i \leq m, \beta_i \in \{0,1\}) \\ b^{i-1} 01 y_i^i v_i b^{i+1} \beta_i y_i^i v_i a^{i-1} 01 z z_i^i u_i & (m + 1 \leq i \leq n) \\ b^{m+1} 11 y_{m+1}^i v_{m+1} & (i = n + 1) \end{cases}$$

We transform a word $w \in L$ 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 $C_i = \text{true}$ else $C_i = \text{false}$.
2. For $i = m + 1, \ldots, n$, $C_i = \text{NOR}(C_j, C_k)$ 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. $g(w) \in \text{CVP}$ for every $w \in L$.

Proof. For $w \in L$, 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 $c_r = \beta_r z z_r^r u_r^r v_r^r$ or $\beta_r y_r^r u_r^r v_r^r$ ($\beta_r \in \{0,1\}, p_r \geq 1$ and $1 \leq r \leq 3n-2m+1$). 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 $c_r (1 \leq r \leq 3n-2m+1)$ is a subword of some $t_i$ ($1 \leq i \leq n$). Thus, without loss of generality, we may assume that for each $i = 1, \ldots, n$, $t_i$ is of the form $\beta_i z z_i^i u_i \beta_i y_i^i v_i \cdots \beta_i y_i^i v_i$ ($\beta_i \in \{0,1\}$). Since $w''$ is also in $N \Delta \Sigma^*$ and for $1 \leq i \leq n$, a subword $\beta_i y_i^i v_i$ of $w''$ does not occur before a subword $\beta_i z z_i^i u_i$ 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 $g$, $C_i = \text{true}$. For $i \geq m + 1$, 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 $b^j 0 b^k 0 a^i 01 z z_i^i u_i$. Moreover, $w_i$ must have 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 $0 z z_i^i u_i$. Therefore, $t_i$ must be in $F$. On the other hand, the value of $C_i = \text{NOR}(C_j, C_k)$ 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 $g(w) \in \text{CVP}$.

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


