A P-Complete Language Describable with Iterated Shuffle

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

https://hdl.handle.net/2324/3152
A P-Complete Language Describable with Iterated Shuffle

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

September 6, 1991

Abstract
We show that a P-complete language can be described by using 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 $\uparrow$, 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 $\cup, \cdot, *, \cup$, 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* \( \uparrow \) and operators, \(*, +\) as follows:

1. \( L_1 \triangle L_2 = \{x_1 y_1 x_2 y_2 \cdots x_m y_m \mid x = x_1 x_2 \cdots x_m \in L_1, y = y_1 y_2 \cdots y_m \in L_2 \text{ and } x_i, y_i \in \Sigma^* \text{ for } i = 1, \ldots, m\} \) (shuffle operator).
2. \( L_1 \uparrow = \{\varepsilon\} \cup L \cup (L \triangle L) \cup (L \triangle L \triangle L) \cup \cdots \) (iterated shuffle).
3. \( L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\} \) (abbreviated to \( L_1 L_2 \)).
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 the \( w \). Thus, we will denote \( \{w\}^*, \{w\}^+, \{w\}^\uparrow \) by \( w^*, w^+, w^\uparrow \), respectively.

As the basis of our reduction, we use the circuit value problem (CVP) that was shown \( \text{P-complete} \) [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 later 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 a alphabet \( \Sigma - \{v_1, v_2, \ldots, v_m\} \). By using the iterated shuffle operation, a language \( \{v_1^n v_2^n \cdots v_m^n \mid n \geq 1\} \) can be described as \( (v_1 v_2 \cdots v_m)^\uparrow \cap v_1^+ v_2^+ \cdots v_m^+ \). 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

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

We often use this form of languages to define a \( \text{P-complete} \) language. Whenever languages like these are defined in the next section, we will not describe the languages explicitly by using the shuffle operation 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 $\cdot, +, \cup, \cap, \Delta, \dagger$.

3.1 Definition of the language

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

At first, a language $L$ is defined as follows:

\[
L_a = a^+ 0 \cup a^+ 1 = \{a^i\beta \mid i \geq 1 \text{ and } \beta \in \{0, 1\}\},
\]
\[
L_{aba} = (b^+ 1b^+ a^+ 0) \cup (b^+ 0b^+ 1a^+ 1) \cup (b^+ 1b^+ 0a^+ 1) \cup (b^+ 0b^+ 0a^+ 1)
= \{b^i\beta\beta'\beta''\alpha^i\beta \mid i, j, k \geq 1 \text{ and } (\beta', \beta'', \beta) \in \{(1, 1, 0), (0, 1, 1), (1, 0, 1), (0, 0, 1)\}\}
\]
\[
L_b = b^+ 1 = \{b^i 1 \mid i \geq 1\}.
\]

\[
L = cL_a^+ L_{aba}^+ L_b.
\]

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

\[
T_x = \{1dx^i u^i \mid i \geq 1\}, \quad T_y = \{1y^i v^i \mid i \geq 1\},
\]
\[
T_{xy} = \{1dx^i u^i y^j v^j \mid i \geq 1\}, \quad T_{yy} = \{1y^j v^j y^j v^j \mid i \geq 1\}.
\]
\[
T_{odd} = T_{xy} T_{yy} * T_y \cap T_{xy} T_{yy}^* = \{1dx^i u^i (1y^j v^j)^j \mid i \geq 1, j \geq 1 \text{ and } j \text{ is odd}\}.
\]
\[
T_{even} = T_{xy} T_{yy} * T_y \cap T_{xy} T_{yy}^* = \{1dx^i u^i (1y^j v^j)^j \mid i \geq 1, j \geq 1 \text{ and } j \text{ is even}\}.
\]

\[
T = T_x \cup T_{odd} \cup T_{even} = \{1dx^i u^i (1y^j v^j)^j \mid i \geq 1 \text{ and } j \geq 0\}.
\]

$F$ is defined in a similar way. We use a symbol 0 instead of 1 which is used to construct the language $T$.

\[
F = \{0dx^i u^i (0y^j v^j)^j \mid i \geq 1 \text{ and } j \geq 0\}.
\]

Subwords $1y^i v^i$ (resp. $0y^i v^i$) of a word in $T$ (resp. $F$) are combined with $b^i 0$ (resp. $b^i 1$) of words in $L$ and decides the value of the $i$th variable. These three languages $L$, $T$ and $F$ are combined with each other by using the shuffle operation and the iterated shuffle.
\[ \mathcal{J} = L \Delta (T \cup F)^t. \]

A language \( \mathcal{K} \) is used for our language to become polynomial time decidable. We construct the language \( \mathcal{K} \) stepwise as follows:

\[
\begin{align*}
A_{11} & = \{a^i11dx^iu^i \mid i \geq 1\}, \\
A_{00} & = \{a^i00dx^iu^i \mid i \geq 1\}, \\
A_{01} & = \{a^i01dx^iu^i \mid i \geq 1\}.
\end{align*}
\]

In a similar way, following languages are defined.

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

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

The language \( M \) has words whose subwords of the form \( dx^iu^i \) corresponding to the \( i \)th gate occurred more than two times a word. We want these subwords to be occurred exactly one time a word.

\[
N_d = (dxudx^2u^2 \Delta (xuxu)^t) \cap (dx^+u^+dx^+u^+) = \{dx^iu^idx^{i+1}u^{n+1} \mid i \geq 1\}.
\]

\[
N = c((dxuN_d^* \cap N_d^*dx^+u^+u^+) \cup (dxuN_d^*dx^+u^+ \cap N_d^*)) = \{cdxudx^2u^2 \ldots dx^iu^i \mid i \geq 1\}.
\]

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

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

Finally, we defined a language \( \mathcal{L} \) as follows:

\[
\mathcal{L} = \mathcal{J} \cap \mathcal{K}.
\]

### 3.2 Proof of the P-completeness

Theorem 1 follows from a next lemma.
Lemma 1 $L$ is log-space equivalent to CVP, i.e., $L$ is log-space reducible from CVP and CVP is log-space reducible from $L$.

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

$$w_i = \begin{cases} a^i 11 d x^i u^i & (C_i = \text{true}) \\ a^i 00 d x^i u^i & (C_i = \text{false}) \\ b^i 01 y^i b^j 01 y^j v^k a^i 01 d x^i u^i & (C_i = \text{NOR}(C_j, C_k)) \\ b^i 11 y^i v^n & (i = n + 1). \end{cases}$$

It is easy to see that this function is computable in log-space by using a deterministic Turing machine.

We show following two claims.

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

Proof. Let a word $w = c w_1 \cdots w_m w_{m+1} \cdots w_n w_{n+1}$ be a transformed word 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 and gate for $m + 1 \leq i \leq n$ and an output of this circuit is true. This instance has only one tuple of assignments of a boolean value (true or false) to each variables. We describe this assignment as $B = (\beta_1, \ldots, \beta_n)$ such that $\beta_i = 1$ (resp. $\beta_i = 0$) if $C_i = \text{true}$ (resp. $C_i = \text{false}$) for $i = 1, \ldots, n$.

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

1. For $i = 1, \ldots, m$, $w_i' = a^i \beta_i$, $w_i'' = b^i d x^i u^i$.
2. For $i = m + 1, \ldots, n$, $w_i' = b^i \beta_i$, $w_i'' = b^i d x^i u^i$.

We note that since $C_j, C_k$ and $C_i$ are related with each other by an NOR gate, $w_i'$ is in $L_{baa}$.

Figure 1: This circuit is transformed to a word $w$. 

\[ w = a 1 1 d x^a a^2 1 1 d x^a u^a a^3 0 0 d x^a u^a b^1 0 1 y^b b^2 0 1 y^b v^b v^a 0 1 d x^a u^a \] 
\[ b^2 0 1 y^b c^3 0 1 y^c v^c a^5 0 1 d x^a u^a b^3 0 1 y^b v^b v^c 0 1 d x^a u^a c^6 0 1 d x^a u^a b^6 1 1 y^b v^c. \]
(3) \( w_{n+1}' = b^n1, \ w_{n+1}'' = 1y^n v^n \).

It is easy to see that a word \( w' = cw_1' \cdots w_{n+1}' \) is in \( L = L_n^+L_{bbn}^+L_h \).

On the other hand, since \( w'' = w_{1}'' \cdots w_{n+1}'' \) is constructed with subwords of the form \( \beta_i dx^i u^i \) or \( \beta_i y^i v^i \) and for each NOR gate, input gate numbers of this gate are always lower than a number of itself, we can describe the word \( w'' \in t_1 \Delta t_2 \Delta \cdots \Delta t_n \), where \( t_i = \beta_i dx^i u^i \beta_i y^i v^i \cdots \beta_i y^i v^i \). Since \( t_i \in T \) or \( F \), for \( i = 1, \ldots, n \), \( f(C) = cw_1 \cdots w_m w_{m+1} \cdots w_n w_{n+1} \in w' \Delta t_1 \Delta \cdots \Delta t_n \subset L \Delta (T \cup F) \) \( = L \).

Since every words \( w \) of \( L \) is contained in \( M \), \( e \) is of the form \( w = cw_1 \cdots w_m w_{m+1} \cdots w_n w_{n+1} \), where, for \( i = 1, \ldots, n + 1 \),

\[
  w_i = \begin{cases} 
    a^i \beta_i \beta_i dx^i u^i & (1 \leq i \leq m, \beta_i \in \{0, 1\}) \\
    b^i 01 y^i v^i b^i 01 y^i v^i \cdots a^i 01 dx^i u^i & (m + 1 \leq i \leq n) \\
    b^{n+1} 11 y^{n+1} v^{n+1} & (i = n + 1) 
  \end{cases}
\]

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 \( g \) is well-defined function and this function is log-space computable.

Claim 2. \( g(w) \in CVP \), for every \( w \in L \).

Proof. Since \( w \in N \), \( \ell_i = i \) for every \( i = 1, \ldots, n \). Moreover, since some parts of \( w \) are constructed of words which are contained in \( T \) or \( F \), a subword \( y^i v^i \) of \( w \) is never occured before a subword \( dx^i u^i \) of \( w \). Therefore \( j, k \leq i \).

Since \( w \in L \Delta (T \cup F) \) and \( w \) includes \( n \) subwords \( dxu, dx^2 u^2, \ldots, dx^n u^n \), there exist \( n \) words \( t_1, \ldots, t_n \) in \( T \cup F \) which contribute a construction of \( w \) by using the iterated shuffle. Without loss of generality, we assume that \( t_i \) includes \( x^i u^i \) as a subword.

We claim that for \( i = 1, \ldots, n \), \( t_i \in T \) if and only if a value of \( C_i \) is \text{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 \( t_k \in T \). Other case is shown in a similar way. By the assumption, values of \( C_j \) and \( C_k \) is \text{true}. We remove contributions of \( t_j \) and \( t_k \) from \( w_i \). The remaining word is \( b^j 0b^k 0a^j 01 dx^i u^i \). Moreover, \( w_i \) must has a contribution from \( L_{bbn} \). This contribution must be of the form \( b^+ 0b^+ 0a^+ 1 \). Thus, the remaining word after removing this contribution is \( 0dx^i u^i \). Therefore, \( t_i \) must be in \( F \). On the other hand, a value of \( C_i = \text{NOR}(C_j, C_k) \) is \text{false}. Thus, we hold this claim.

Since \( t_n \) must be in \( T \), a value of \( C_n \) is \text{true}. Thus, \( g(w) \in CVP \). □

By the discussion above, we can say that \( L \) have a log-space reduction \( f \) from CVP and CVP have a log-space reduction \( g \) (inverse of \( f \)) from \( L \). □
References


