数理解析研究所講究録
第 1846 巻 2013 年 27-34

Degree of Nondeterminism for Pushdown Automata

IBARAKI Tatsuya* and KUBOSAWA Shumpei†

1 Introduction

While time complexity and space complexity are commonly studied in computer science, complexity with respect to nondeterminism for pushdown automata (PDA) is not known widely. We thought that we can measure nondeterminism of a PDA by a function which maps an input length n of the PDA to the maximum number of nondeterministic transitions required to accept the input.

In this paper, we first define the measure of nondeterminism for pushdown automata, and then consider the measure of nondeterminism for certain context-free languages (CFL).

While we were preparing materials of this presentation, we found similar definitions and applications of this kind of complexity in the literature. They are Salomaa et al. [1] and Goldstine et al. [2]. But, since the languages they consider are different from ours, we think it worth to present our original ones in this paper.

2 Preliminaries

The definition of PDA which we use here is similar to the definition given by M. Sipser [3]. In this paper we consider only non-deterministic PDA's.

Definition 2.1
For any alphabet A we denote $A\cup\{\varepsilon\}$ by A_{ε}. We define non-deterministic PDA as a 6-tuple

$$M=(Q,\Sigma,\Gamma,\delta,q_{0},F)$$

where Q is a finite set of states, Σ is an input alphabet, Γ is a stack alphabet, $\delta:Q\times\Sigma_{\varepsilon}\times\Gamma_{\varepsilon}\rightarrow\wp(Q\times\Gamma_{\varepsilon})$ is a transition function, $q_{0}\in Q$ is an initial state and $F\subseteq Q$ is a set of final states.

For these pushdown automata, we define acceptance by final state acceptance rather than by empty-stack acceptance.

*Faculty of Commerce and Management, Hitotsubashi University
†Faculty of Social Sciences, Hitotsubashi University

Definition 2.2
Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA. We define a configuration of M by a tuple $c = (r, w, s)$ where $r \in Q$, $w \in \Sigma^\star$ and $s \in \Gamma^\star$. A configuration $c = (r, w, s)$ is an accepting configuration when $r \in F$ and $w = \varepsilon$. For any two configurations $c_1 = (r_1, w_1, s_1)$ and $c_2 = (r_2, w_2, s_2)$, we say that there is a transition from c_1 to c_2, $c_1 \vdash c_2$, if $(r_2, b) \in \delta(r_1, x, a)$ where $w_1 = xw_2$ for $x \in \Sigma_\varepsilon$ and $s_1 = at, s_2 = bt$ for $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^\star$. A sequence of configuration $C = (c_0, c_1, ..., c_m)$ for $m \in \mathbb{N}$ is a computation of M when $c_i \vdash c_{i+1}$ for all $0 \le i \le m-1$. A computation $C = (c_0, c_1, ..., c_m)$ is an accepting computation for $w \in \Sigma^\star$ if $c_0 = (q_0, w, \varepsilon)$ and c_m is an accepting configuration. When there is at least one accepting computation for w, M accepts w. The language accepted by M is $L(M) = \{w \in \Sigma^\star | w \text{ is accepted by } M.\}$.

3 Degree of nondeterminism

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA. For each configuration c on M, we define $\nu_M(c)$ by

$$\nu_M(c) = \begin{cases} 0 & if \quad \#\{c'|c \vdash c'\} < 2 \\ 1 & if \quad \#\{c'|c \vdash c'\} \ge 2 \ . \end{cases}$$

For each computation $C = (c_0, c_1, ..., c_m)$ of M, we define $\nu_M(C)$ by

$$\nu_M(C) = \sum_{i=0}^{m-1} \nu_M(c_i).$$

For all $w \in L(M)$, we define $\nu_M(w)$ by

$$\nu_M(w) = \min\{\nu_M(C) | C \text{ is an accepting computation of } M \text{ for } w\}.$$

For all $n \in \mathbb{N}$, we define $\nu_M(n)$ by

$$\nu_M(n) = \begin{cases} \max\{\nu_M(w) \,|\, w \in L(M)^{(n)}\} & if \quad L(M)^{(n)} \ne \phi \\ 0 & if \quad L(M)^{(n)} = \phi \ . \end{cases}$$

For a function $f : \mathbb{N} \to \mathbb{N}$, we say that M runs at nondeterministic degree of $f(n)$, if $\nu_M(n) \le f(n)$ for all $n \in \mathbb{N}$. Furthermore, for a function $t : \mathbb{N} \to \mathbb{N}$, we define the class $ND(t(n))$ of languages by

$$ND(t(n)) = \{L | L = L(M) \text{ and } M \text{ runs at nondeterministic degree of } \mathcal{O}(t(n)) \text{ for some PDA } M\}.$$

4 A language in $ND(\log n)$

Definition 4.1 (language)
Let a sequence $\{c_t\}_{t=0}^{\infty}$ of strings on $\Sigma = \{0, \#\}$ be defined as follows:

$$\begin{cases} c_t = 0 & t = 0 \\ c_t = c_{t-1}\#0^{2^t} & t \geq 1 \end{cases}$$

Let L_1 and L_2 be defined by

$$L_1 = \{c_t | t \geq 1\}$$

and

$$L_2 = \overline{L_1} (= \Sigma^* \setminus L_1).$$

Here are some examples of strings contained in L_1.

0#00
0#00#0000
0#00#0000#00000000

Theorem 4.1
L_2 has nondeterministic degree of $\mathcal{O}(\log n)$, i.e.,

$$L_2 \in ND(\log n).$$

The idea of following proof is that if M, a recognizer of L_2, read a string w with nondeterministic transitions only when M read a #, the number of nondeterministic transitions never exceed $\log |w|$. This is because if a string w which contains more #'s than the string which is in L_1 with the same length, M accepts w before arriving at the terminal.

Proof of Theorem 4.1
We prove this theorem by showing the existence of PDA M that accepts L_2 and runs by nondeterministic degree of $\mathcal{O}(\log n)$. Here we define that the stack alphabet of M is $\{X, \$\}$. And we denote a sequence of 0's by a "block" and blank symbol on the input tape as B.

We construct M as follows.

M="On input string w:

Step 1 *Push \$ onto the stack.*

Step 2 *If the input head reads 0, push X onto the stack.*
If the input head reads #, ACCEPT.

Step 3 *⟨⟨ Nondeterministic step ⟩⟩*
If the input head reads #, **nondeterminisitically** *either go to Step 4 or Step 5.*
If the input head reads 0, push XX onto the stack and go to Step 3.

Step 4 *Clear the stack and go to step 3.*

Step 5 *(Check the number of 0's - main process)*
If the input head reads 0 and the stack head reads X, pop X up from the stack and go to Step 5.
If the input head reads 0 and the stack head reads \$ up from the stack, go to step 6. If the input head reads B, ACCEPT

Step 6 If the input head reads 0, ACCEPT.

First, it is clear that M accepts L.

Next, we show that M runs at nondeterministic degree of $\mathcal{O}(\log n)$.

By the definition of M, the number of M's guessing steps is less than or equal to the number of #'s in Step 3.

Therefore, it suffices to show that number of #'s in c_t is less than a constant multiple of $\log |c_t|$.

$$|c_t| = \sum_{k=0}^{t}(2^k+1) - 1 = 2^{t+1} + t - 1$$

$$\log(|c_t| - t + 1) - 1 = t$$

Therefore we have

$$t \leq \log |c_t|.$$

□

5 A language in $ND(\sqrt{n})$

In order to describe the next example, we use directed graphs expressed by adjacency matrices. A **source** is a node with at least one outgoing arc and no incoming arc. A **sink** is a node with no outgoing arcs and at least one incoming arc.

Now, we introduce another type of node which we call a **quasi-sink**.

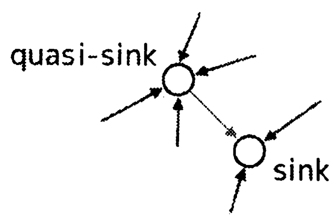

Figure 1: Quasi-sink and sink

Definition 5.1
A quasi-sink is a node of a directed graph with exactly one outgoing arc which goes to some sink and no other outgoing arcs. (It may have any number, including 0, of incoming arcs.)

In this paper, an adjacency matrix is converted into a string in the following way.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \#10101\#00000\#00111\#01000\#10110$$

Each row is serially aligned from left to right and separated by #'s.

Definition 5.2
For an alphabet $\Sigma = \{0, 1, \#\}$ and $l \geq 0$ let Σ^l denote the set of strings over Σ whose length is l. Let $L_{AM} = \{w | ^{\exists} l \in \mathbb{N}, w \in (\sharp\{0,1\}^l)^l\}$ which consists of adjacency matrices expressed by strings as described above. Now, let

$$L_1 = \Sigma^* \setminus L_{AM}$$

and L_2 be the language consisting of strings which correspond to adjacency matrices of directed graphs with at least one quasi-sink. Finally we define that

$$L_{qs} = L_1 \cup L_2.$$

Theorem 5.1
L_{qs} has nondeterministic degree of $\mathcal{O}(\sqrt{n})$, i.e.,

$$L_{qs} \in ND(\sqrt{n}).$$

Before proving the theorem, we explain how our recognizer of L_{qs} works. L_{qs} is the union of two languages L_1 and L_2 as above.

L_1 can further be classified in two cases.

One is the case where there exist at least one pair of adjacent rows with different length. In this case, the recognizer only guesses at each #'s till it reads rows whose length is equal to the next row. Here is an example of those strings.

```
#01000
#00101
#10000
#00110
#0000
```

In the above case, the recognizer nondeterministically transit only four times. At the first three #'s, the machine guesses whether compare the following row and the next row, or not. But at the fourth #, the recognizer has a chance to accept the input because the length of the fourth and the fifth row are different. So $\nu_M(n) \leq \sqrt{n}$ holds. In this case, $n = 29$ and $4 < \sqrt{29}$ where M is the recognizer.

The other case is the case where a string forms a matrix, but it is not a square matrix. The followings are examples of those strings.

```
#01010
#00101
#00000
#00100
```

```
#01010000101010
#10111111101111
#01010111010101
```

To accept these strings, the recognizer is required to guess only at the first #. In other words, the machine needs to compare the length of the first row and the number of columns.

L_2 can be accepted with less than or equal to $\sqrt{n}$ nondeterministic transitions if the stack is efficiently used. The recognizer needs to transit its state nondeterministically at each # before arriving at the quasi-sink or sink row. So the number of guessing does not exceed the square root of the input length.

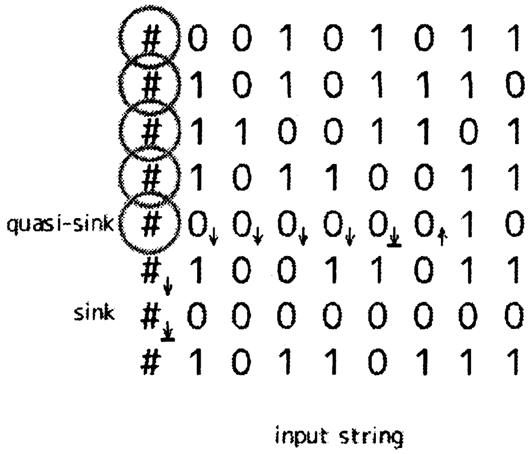

Figure 2: An adjacency matrix with quasi-sink and sink

Encircled letters in Figure 2 indicate the places where nondeterministic transitions occur. Pushing operations occur at these places. So after reading 5th #, there is 5 letters on the stack of the recognizer.

In this figure, tiny arrows located at right-bottom of some letters mean stack operation. The direction of arrows indicates the direction of the result of an operation on the stack. Downward arrows depict popping of one letter, and upward ones depict pushing. And downward arrows with bottom bar express clearing of the stack by the last popping.

Proof of Theorem 5.1

We prove it by construction of the PDA M which runs by nondeterministic degree of $\mathcal{O}(\sqrt{n})$ and recognizes L_{qs} as follows. Here the stack alphabet of M is $\{\$, X\}$.

$M =$ "*On input string w:*

Step 1 *If the input head reads #, push \$ onto the stack. Otherwise, ACCEPT.*

Step 2 $\langle\langle$ *Nondeterministic step* $\rangle\rangle$
Nondeterministically *go to Step 3, go to Step 4, go to Step 5 or go to Step 6.*

Step 3 $\langle\langle$ *Nondeterministic step* $\rangle\rangle$
Nondeterministically *process either following (a) or (b).*

(a) *Check the length of current row and that of the next row. If these rows are different, ACCEPT.*
(b) *Do nothing while the head read current row. And go to Step 3.*

Step 4 *(Check if the number of rows and columns are different.)*
Pop two letters up from the stack.
While the input head reads 0 or 1, push X onto the stack.
If the input head read # and the stack head reads X, pop X up from the stack and do nothing while reading following 0 or 1.
If the input head read # and the stack head reads \$, ACCEPT.
If the input head read B and the stack head reads \$, REJECT.
Otherwise, ACCEPT.

Step 5 *(Check if a quasi-sink exist prior to the sink.)*
In this case, M can decide in the way we described above with Figure 2.

Step 6 ⟨⟨ *Nondeterministic step* ⟩⟩
Nondeterministically *process either following (a) or (b).*
(a) *If the input head reads 0, repeat this step.*
If the input head reads 1, REJECT.
If the input head reads # and the stack head reads X, pop X from the stack and go to Step 7.
(b) *If the input head reads 0 or 1, repeat this step.*
If the input head reads #, push X onto the stack and go to Step 6.

Step 7 ⟨⟨ *Nondeterministic step* ⟩⟩
Nondeterministically *process either following (a-1) or (b).*
(a-1) *If the input head reads 0 and the stack head reads X, pop X from the stack and repeat this step.*
If the input head reads 1 and the stack head reads \$, go to (a-2).
(a-2) *If the input head reads 0, continue (a-2).*
(a-3) *If the input head reads # or B, ACCEPT.*
(b) *If the input head reads 0 or 1, repeat this step.*
If the input head reads #, go to Step 7. "

On this PDA, the number of nondeterministic transitions required to L_1 and a half part of L_2 is stated above. So here we describe why the other part of L_2 can be recognized by nondeterministic degree of $\mathcal{O}(\sqrt{n})$. In step 6, nondeterministic transitions only occur at each #'s. And it is clear that the input forms adjacency matrix in this case. Otherwise such input string is accepted by step 3 or step 4. The worst case of this part is that a quasi-sink is located at the last row. In this case, nondeterministic transitions occur r times if the input matrix consists of r rows. Now this string contains r #'s and r^2 numbers, so the length is $r + r^2$. Hence the number of nondeterministic transitions r is always less than $\sqrt{|w|} = \sqrt{r + r^2}$. □

6 Conclusion

We have defined the degree of nondeterminism for pushdown automata, and obtained an upper bound of the degree of nondeterminism for two languages. The first language can be recognized by a pushdown automaton which runs by nondeterministic degree of $\mathcal{O}(\log(n))$, so it is in $ND(\log(n))$. And the second one is in $ND(\sqrt{n})$.

7 Acknowledgment

We thank all the participants of "RIMS Research Meeting" held in October 2009. And let us express our heartfelt appreciation to Professor Machida for giving us the oppotunity to make this presentation.

References

[1] K. Salomaa, D. Wood, S. Yu, Pumping and pushdown machines, Inform. Théor. Appl. 28 (1994) 221-232.

[2] J. Goldstine, H. Leung, D. Wotschke. Measuring nondeterminism in pushdown automata, J. Comput. System Sci. 71 (2005) 440-466.

[3] M. Sipser, Introduction to the Theory of Computation, Second Edition, Thomson, MA, 2006.