Higher order stacks can not replace nondeterminism

Paweł Parys
University of Warsaw
ul. Banacha 2, 02-097 Warszawa, Poland
parys@mimuw.edu.pl

February 10, 2010

Abstract

We show that deterministic higher order pushdown automata (even with panic) can not recognize some context-free languages.

1 Definition

A *deterministic h-th order pushdown automaton* (DhPDA for short) is given by a tuple $(A, \Gamma, \gamma_I, Q, q_I, \delta)$ where A is an input alphabet, Γ is a stack alphabet, γ_I an initial stack symbol, Q a set of states, q_I an initial state and $\delta\colon Q \times A \times \Gamma \to Q \times Ops$. The set Ops contains the following operations: pop, $\mathsf{push}(\gamma)$ for any $\gamma \in \Gamma$, $\mathsf{copy}(i)$ for $i = 1, \dots, h-1$, next_0, next_{acc}. When $\delta(q, a, \gamma) = (q', op)$ we write $q, a, \gamma \to q', op$. A *deterministic h-th order pushdown automaton with panic* (DhPDAP for short) has additionally a panic operation.

The automaton has a h-th order stack of $h-1$-th order stacks of ... of first order stacks; the first order stacks contain symbols from Γ. None of the stacks is empty; if the last element of a stack is removed, we remove also the whole stack from a higher order stack. At the beginning there is one stack of each order and the first order stack contains one γ_I symbol. The automaton always sees only the last (topmost) symbol on the last (topmost) stack. When the current state is q, the letter of the input word under the head is a, and the last stack symbol is γ, the automaton looks at the transition $q, a, \gamma \to q', op$, changes its state into q' and:

- if $op = \mathsf{pop}$, it removes one symbol from the last first order stack; when any stack becomes empty, it is removed from the higher order stack; when the h-th order stack becomes empty, the automaton fails;
- if $op = \mathsf{push}(\gamma)$, it places the symbol γ on the top of the first order stack;
- if $op = \mathsf{copy}(i)$, it copies the last i-th order stack;
- if $op = \mathsf{next}_0$, it moves the head to the next letter of the input word; if we are on the end of the word, the automaton fails;
- if $op = \mathsf{next}_{acc}$, it moves the head to the next letter of the input word; if we are on the end of the word, the automaton accepts the word;
- if $op = \mathsf{panic}$, it restores the stack configuration in which this particular symbol γ was put on a first order stack.

2 Theorem

Theorem 1. *There is a context-free language, which can not be recognized by an deterministic h-th order pushdown automaton with panic, for any h.*

As the counterexample we take a language L over an alphabet $A = \{a, b, c, s\}$ defined by the following context-free grammar:

$$S \to aS \mid bS \mid sS \mid sT \mid sX$$
$$T \to bTa \mid sb$$
$$X \to bXbb \mid aY$$
$$Y \to bYbb \mid aYa \mid s.$$

Let us introduce some notation. For any language L over an alphabet A (in particular for the language defined above) and a word $w = w_1 \dots w_n \in A^*$ we define a word $w_L \in (A \times \{0, acc\})^*$: together with each letter w_i we write acc if $w_1 \dots w_i \in L$ and 0 otherwise. On the other hand, for a word $w' \in (A \times \{0, acc\})^*$, let $\pi_A(w')$ be its projection to the first coordinate. The number of a letters in a word w is denoted as $\#_a(w)$.

The key observation is that a deterministic automaton knows for each prefix of the input word if it is accepted or not, which is described by the following lemma.

Lemma 2. *Let L be a language over an alphabet A recognized by a DhPDAP and K a regular language over an alphabet $A \times \{0, acc\}$. Then the language $\{w : w_L \in K\}$ is accepted by a DhPDAP. The number of states is multiplied by the size of a deterministic automaton recognizing K.*

Proof
Let L be recognized by $\mathcal{A} = (A, \Gamma, \gamma_I, Q, q_I, \delta)$ and K by a deterministic finite automaton with states P. The new automaton would have states $Q \times P$. To define δ', for any $a \in A$, $\gamma \in \Gamma$, $q \in Q$ we look at $\delta(a, \gamma, q) = (q', op)$. If $op = \mathsf{next}_k$ for each $p \in P$ we take $\delta'(a, \gamma, (q, p)) = ((q', p'), \mathsf{next}_l)$ where p' is read from the transition $p \xrightarrow{a,k} p'$ and $l = acc$ if p' is accepting, $l = 0$ otherwise. Otherwise for each $p \in P$ we take $\delta'(a, \gamma, (q, p)) = ((q', p), op)$. It is easy to see that $\mathcal{A}'$ recognizes the desired language. □

As K_h^k we take the regular language of all words $w' \in (A \times \{0, acc\})^*$ such that

1. $\pi_A(w')$ starts with sb^ks and ends with sb^*s, and
2. in $\pi_A(w')$ there are exactly $2h + 2$ fragments of the form sb^*s, and
3. a letter $(s, ?)$ appears if and only if the previous letter was $(?, acc)$, or if it is the first or the second $(s, ?)$ letter in the word.

Notice that this language can be implemented by a deterministic automaton with $O(k)$ states (for fixed h), because the only dependence in k is in the length of sb^ks.

For any $k, l \geq 0$ we define ex_2 as the following tower of powers of 2:

$$ex_2(k, 0) = k \qquad ex_2(k, l + 1) = 2^{ex_2(k,l)}.$$

Assuming that L is recognized by a DhPDAP, from the above lemma we get that $M_h^k = \{w : w_L \in K_h^k\}$ is also recognized by a DhPDAP. We will show in M_h^k there is exactly one word, of length greater than $ex_2(k, 2h + 1)$.

For any word w, a fragment between two consecutive s letters is called a *block* (note that each word in M_h begins and ends with s). Condition 3 and the grammar of L say that for each two consecutive blocks u, v, the word usv has to be generated by a T or X nonterminal. The first block is b^k and the last block is b^*. Now see that for each block u, the next block v is uniquely determined:

- When $\#_a(u) > 0$, only X fits to usv. The word v has to be the mirror image of u, in which every b is duplicated and the last a is removed. Hence $\#_a(v) = \#_a(u) - 1$, $\#_b(v) = 2 \cdot \#_b(u)$. Observe that
$$\#_b(v) \cdot 2^{\#_a(v)} = \#_b(u) \cdot 2^{\#_a(u)}.$$

- When $\#_a(u) = 0$, only T fits to usv. Thus $v = ba^{\#_b(u)}$. In this case

$$\#_b(v) \cdot 2^{\#_a(v)} = ex_2(\#_b(u) \cdot 2^{\#_a(u)}, 1).$$

In the second case we can also finish the word without creating the next block. Finishing is possible only when we are in the $2h+2$-th b^* block, as this is checked by K_h^k (in the second condition). Moreover, in that case we have to finish: if we continue, too many b^* blocks will be present in the word, as the last block will be also a b^* block. Note that when processing in that way, we will finish at some moment: the number of consecutive blocks with $\#_a(u) > 0$ is always finite, as $\#_a(u)$ decreases; after a finite number of them next b^* block appear. The conditions described above are not only necessary, but also sufficient: we get a word in M_h^k.

We will calculate the number of b letters in the last block. It is important to look at the number $\#_b(u) \cdot 2^{\#_a(u)}$, which we call the characteristic of a block. The characteristic of the first block is k (recall that the first block is b^k). In the next block it becomes 2^k, and it stays the same until the next b^* block. Then it becomes 2^{2^k}, and so on; at the $2h+2$-th b^* block it is $ex_2(k, 2h+1)$. This means that in the last block we have $ex_2(k, 2h+1)$ letters. Hence the length of the word is greater than this number.

Recall the following theorem.

Theorem 3 ([1], Corollary 9.7)**.** *Let $\mathcal{A}$ be a (nondeterministic) h-th order pushdown automaton. If $\mathcal{A}$ accepts a word of length at least*

$$ex_2(h3^{h-1}h!|Q|^3, 2h)$$

then the language recognized by $\mathcal{A}$ is infinite.

Recall also a folklore result, that an deterministic h-th order pushdown automaton with panic can be converted into a (nondeterministic) h-th order pushdown automaton (without panic), and moreover that the number of states is increased only polynomially. Intuitively, while putting a symbol on the stack, the nondeterministic automaton should guess if this symbol will be used to do the panic operation or not. If yes, this place should be marked appropriately. When later there should be a panic operation, we start removing symbols from the stacks, until we reach this place (it is possible to guarantee that it works exactly like the real panic operation).

We have a DhDPAPT of size $O(k)$ recognizing M_h^k. Thus we also have a (nondeterministic) h-th order pushdown automaton of size $poly(k)$ recognizing M_h^k. For k big enough, the number $ex_2(k, 2h+1)$ becomes greater than $ex_2(h3^{h-1}h!|Q_k|^3, 2h)$ (where Q_k are the states of the automaton recognizing M_h^k). This causes a contradiction with the theorem.

References

[1] A. Blumensath. On the structure of graphs in the Caucal hierarchy. *Theor. Comput. Sci.*, 400(1-3):19–45, 2008.