Theory of Computation

Thomas Zeugmann

Hokkaido University
Laboratory for Algorithmics

http://www-alg.ist.hokudai.ac.jp/~thomas/ToC/

Lecture 9: Pushdown Automata

Pushdown Automata I

In this lecture we continue with pushdown automata. Recall that we used finite nondeterministic automata as a machine model to characterize the regular languages.

Pushdown Automata I

In this lecture we continue with pushdown automata. Recall that we used finite nondeterministic automata as a machine model to characterize the regular languages.

Now, we shall use pushdown automata to characterize the context-free languages. Generally speaking, a pushdown automaton is an extension of the nondeterministic finite automaton with λ-transitions.

Pushdown Automata I

In this lecture we continue with pushdown automata. Recall that we used finite nondeterministic automata as a machine model to characterize the regular languages.

Now, we shall use pushdown automata to characterize the context-free languages. Generally speaking, a pushdown automaton is an extension of the nondeterministic finite automaton with λ-transitions.

Here, by λ-transition we mean that the automaton is allowed to read the empty word λ on its input tape and to change its state accordingly.

Pushdown Automata II

The *extension* to a nondeterministic finite automaton consists in a *stack*. The stack allows a pushdown automaton to memorize any finite string.

Pushdown Automata II

The *extension* to a nondeterministic finite automaton consists in a *stack*. The stack allows a pushdown automaton to memorize any finite string.

But in contrast to a general model for a computer (e.g., a Turing machine or a random access machine) the access to the stack is in *lifo*-mode only. Here lifo stands for "last in first out."

Pushdown Automata II

The *extension* to a nondeterministic finite automaton consists in a *stack*. The stack allows a pushdown automaton to memorize any finite string.

But in contrast to a general model for a computer (e.g., a Turing machine or a random access machine) the access to the stack is in *lifo*-mode only. Here lifo stands for "last in first out."

That is, the stack can be *read*, *pushed*, and *popped* only at the top, just like the stack data structure you are already familiar with.

Pushdown Automata III

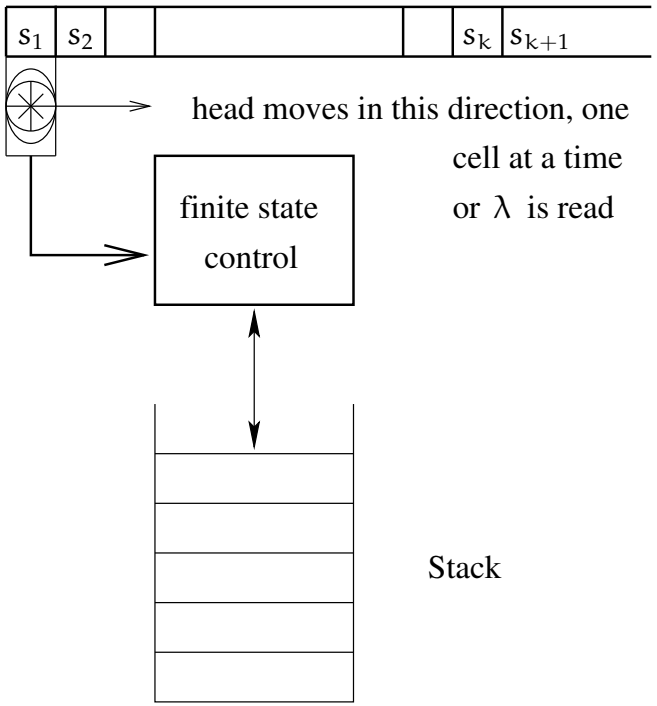

Figure 1: A pushdown automaton

Pushdown Automata IV

More formally, in one transition, the pushdown automaton:

(1) Consumes from the input the symbol it reads. If λ is read then no input symbol is consumed.

Pushdown Automata IV

More formally, in one transition, the pushdown automaton:

(1) Consumes from the input the symbol it reads. If λ is read then no input symbol is consumed.

(2) Goes to a new state which may or may not be the same state as its current state.

Pushdown Automata IV

More formally, in one transition, the pushdown automaton:

(1) Consumes from the input the symbol it reads. If λ is read then no input symbol is consumed.

(2) Goes to a new state which may or may not be the same state as its current state.

(3) Replaces the symbol at the top of the stack by any string. The string could be λ, which corresponds to a pop of the stack. It could be the same symbol that appeared at the top of the stack previously, i.e., no change is made to the stack. It could also replace the symbol on top of the stack by one other symbol. In this case, the pushdown automaton changes the top of the stack but does neither push or pop it. Finally, the top stack symbol could be replaced by two or more symbols which has the effect of (possibly) changing the top stack symbol and then pushing one or more new symbols onto the stack.

Example

We informally show $L = \{ww^T \mid w \in \{0,1\}^*\}$ to be acceptable by a pushdown automaton. Let $x \in \{0,1\}^*$ be given as input.

Example

We informally show $L = \{ww^T \mid w \in \{0,1\}^*\}$ to be acceptable by a pushdown automaton. Let $x \in \{0,1\}^*$ be given as input.

(1) Start in a state q_0 representing a "guess" that we have not yet seen the middle of x. While in state q_0, we read one symbol at a time and store the symbol read in the stack by pushing a copy of each input symbol onto the stack.

Example

We informally show $L = \{ww^T \mid w \in \{0,1\}^*\}$ to be acceptable by a pushdown automaton. Let $x \in \{0,1\}^*$ be given as input.

(1) Start in a state q_0 representing a "guess" that we have not yet seen the middle of x. While in state q_0, we read one symbol at a time and store the symbol read in the stack by pushing a copy of each input symbol onto the stack.

(2) At any time, we may guess that we have seen the middle (i.e., the end of w if $x = ww^T$ is an input string from L). At this time, w will be on the stack with the rightmost symbol of w at the top and the leftmost symbol of w at the bottom. We signify this choice by *spontaneously* changing the state to q_1 (i.e., we read λ instead of the next input symbol).

Example - continued

(3) Once in state q_1, we compare the input symbols with the symbols at the top of the stack. If the symbol read from input is equal to symbol at the top of the stack, we proceed in state q_1 and pop the stack. If they are different, we finish without accepting the input. That is, this branch of computation dies.

Example - continued

(3) Once in state q_1, we compare the input symbols with the symbols at the top of the stack. If the symbol read from input is equal to symbol at the top of the stack, we proceed in state q_1 and pop the stack. If they are different, we finish without accepting the input. That is, this branch of computation dies.

(4) If we reach the end of x and the stack is empty, then we accept x.

Remarks

Clearly, if $x \in L$, then by guessing the middle of x rightly, we arrive at an accepting computation path. If $x \notin L$, then independently of what we are guessing, no computation path will lead to acceptance. Thus, the pushdown automaton described above is a nondeterministic acceptor for L.

Remarks

Clearly, if $x \in L$, then by guessing the middle of x rightly, we arrive at an accepting computation path. If $x \notin L$, then independently of what we are guessing, no computation path will lead to acceptance. Thus, the pushdown automaton described above is a nondeterministic acceptor for L.

Note that a pushdown automaton is allowed to change the stack as described above while performing a spontaneous transition.

Remarks

Clearly, if $x \in L$, then by guessing the middle of x rightly, we arrive at an accepting computation path. If $x \notin L$, then independently of what we are guessing, no computation path will lead to acceptance. Thus, the pushdown automaton described above is a nondeterministic acceptor for L.

Note that a pushdown automaton is allowed to change the stack as described above while performing a spontaneous transition.

Question

How can we formally define the language accepted by a pushdown automaton?

Pushdown Automata V

Looking at the example above, we see that the automaton has finished its computation with empty stack. Thus, it would be natural to define the language accepted by a pushdown automaton to be the set of all strings on which the pushdown automaton has a computation that ends with empty stack.

Pushdown Automata V

Looking at the example above, we see that the automaton has finished its computation with empty stack. Thus, it would be natural to define the language accepted by a pushdown automaton to be the set of all strings on which the pushdown automaton has a computation that ends with empty stack.

Second, we can adopt the method we have used for finite automata. That is, we choose a subset of the set of all states and declare each state in this subset to be an accepting state. If taking this approach, it would be natural to define the language accepted by a pushdown automaton to be the set of all strings for which there is a computation ending in an accepting state.

Pushdown Automata V

Looking at the example above, we see that the automaton has finished its computation with empty stack. Thus, it would be natural to define the language accepted by a pushdown automaton to be the set of all strings on which the pushdown automaton has a computation that ends with empty stack.

Second, we can adopt the method we have used for finite automata. That is, we choose a subset of the set of all states and declare each state in this subset to be an accepting state. If taking this approach, it would be natural to define the language accepted by a pushdown automaton to be the set of all strings for which there is a computation ending in an accepting state.

We give both ideas a try.

Pushdown Automata VI

Definition 1

$\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ is said to be a ***pushdown automaton*** provided

(1) Q is a finite nonempty set (the set of ***states***),

(2) Σ is an alphabet (the so-called input alphabet),

(3) Γ is an alphabet (the so-called stack alphabet),

(4) $\delta\colon Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \longrightarrow \wp_{\mathtt{fin}}(Q \times \Gamma^*)$, the ***transition relation***[a],

(5) $q_0 \in Q$ is the ***initial state***,

(6) k_0 is the so called ***stack symbol***, i.e., $k_0 \in \Gamma$ and initially the stack contains exactly one k_0 and nothing else.

(7) $F \subseteq Q$, the set of ***final states***.

[a] Here we use $\wp_{\mathtt{fin}}(Q \times \Gamma^*)$ to denote the set of all finite subsets of $Q \times \Gamma^*$.

Pushdown Automata VII

In the following, we use small letters from the beginning of the alphabet to denote input symbols, and small letters from the end of the alphabet to denote strings of input symbols. We use capital letters to denote stack symbols from Γ and small Greek letters to denote strings of stack symbols.

Pushdown Automata VII

In the following, we use small letters from the beginning of the alphabet to denote input symbols, and small letters from the end of the alphabet to denote strings of input symbols. We use capital letters to denote stack symbols from Γ and small Greek letters to denote strings of stack symbols.

Next, consider

$$\delta(q, a, Z) = \{(q_1, \gamma_1), (q_2, \gamma_2), \ldots, (q_m, \gamma_m)\},$$

where $q, q_i \in Q$ for $i = 1, \ldots, m$, $a \in \Sigma$, $Z \in \Gamma$ and $\gamma_i \in \Gamma^*$ for $i = 1, \ldots, m$.

Interpretation: $\mathcal{K}$ is in state q, reads a on its input tape and Z on the top of its stack. Then it can nondeterministically choose exactly one (q_i, γ_i), $i \in \{1, \ldots, m\}$ for the transition to be made. So, it changes its state to q_i, moves the head on the input tape one position to the right provided $a \neq \lambda$ and replaces Z by γ_i.

Pushdown Automata VIII

We make the convention that the rightmost symbol of γ_i is pushed first in the stack, then the second symbol (if any) from the right, and so on. Hence, the leftmost symbol of γ_i is the new symbol which is then on the top of the stack. If $\gamma_i = \lambda$, then the interpretation is that Z has been removed from the stack.

Pushdown Automata VIII

We make the convention that the rightmost symbol of γ_i is pushed first in the stack, then the second symbol (if any) from the right, and so on. Hence, the leftmost symbol of γ_i is the new symbol which is then on the top of the stack. If $\gamma_i = \lambda$, then the interpretation is that Z has been removed from the stack.

If $a = \lambda$, the interpretation is the same as above, except that the head on the input tape is *not* moved.

Instantaneous Descriptions

In order to formally deal with computations performed by a pushdown automaton we define *instantaneous descriptions*. An instantaneous description is a triple (q, w, γ), where $q \in Q$, $w \in \Sigma^*$ and $\gamma \in \Gamma^*$.
Let $\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ be a pushdown automaton. Then we write

$$(q, aw, Z\alpha) \xrightarrow[\mathcal{K}]{1} (p, w, \beta\alpha)$$

provided $(p, \beta) \in \delta(q, a, Z)$. Note that $a \in \Sigma \cup \{\lambda\}$.

By $\xrightarrow[\mathcal{K}]{*}$ we denote the reflexive transitive closure of $\xrightarrow[\mathcal{K}]{1}$.

Modes of Acceptance

Definition 2

Let $\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ be a pushdown automaton. We define the language ***accepted by*** $\mathcal{K}$ ***via final state*** to be the set

$$L(\mathcal{K}) = \{w \mid (q_0, w, k_0) \xrightarrow[\mathcal{K}]{*} (p, \lambda, \gamma) \text{ for some } p \in F\,,\ \gamma \in \Gamma^*\}\,.$$

The language ***accepted by*** $\mathcal{K}$ ***via empty stack*** is

$$N(\mathcal{K}) = \{w \mid (q_0, w, k_0) \xrightarrow[\mathcal{K}]{*} (p, \lambda, \lambda) \text{ for some } p \in Q\}\,.$$

Since the sets of final states is irrelevant if acceptance via empty stack is considered, we always set $F = \emptyset$ in this case.

Determinism

Definition 3

A pushdown automaton $\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ is said to be ***deterministic*** if

(1) for every $q \in Q$ and $Z \in \Gamma$ we have $\delta(q, a, Z) = \emptyset$ for all $a \in \Sigma$ if $\delta(q, \lambda, Z) \neq \emptyset$, and

(2) for all $q \in Q$, $Z \in \Gamma$ and $a \in \Sigma \cup \{\lambda\}$ we have $\mathrm{card}(\delta(q, a, Z)) \leqslant 1$.

Remarks

In the latter definition, we had to include Condition (1) to avoid a choice between a normal transition and a spontaneous transition. On the other hand, Condition (2) guarantees that there is no choice in any step. So, Condition (2) resembles the condition we had imposed when defining deterministic finite automata. The language accepted by a deterministic pushdown automaton is defined in the same way as for nondeterministic pushdown automata. That is, we again distinguish between acceptance via final state and empty stack, respectively.

Remarks

In the latter definition, we had to include Condition (1) to avoid a choice between a normal transition and a spontaneous transition. On the other hand, Condition (2) guarantees that there is no choice in any step. So, Condition (2) resembles the condition we had imposed when defining deterministic finite automata. The language accepted by a deterministic pushdown automaton is defined in the same way as for nondeterministic pushdown automata. That is, we again distinguish between acceptance via final state and empty stack, respectively.

As far as finite automata have been concerned, we could prove that the class of languages accepted by deterministic finite automata is the same as the class of languages accepted by nondeterministic finite automata. Note that an analogous result cannot be obtained for pushdown automata.

Comparing the Modes of Acceptance I

Theorem 1

Let $\mathrm{L} = \mathrm{L}(\mathcal{K})$ *for a pushdown automaton* $\mathcal{K}$*. Then there exists a pushdown automaton* $\widetilde{\mathcal{K}}$ *such that* $\mathrm{L} = \mathrm{N}(\widetilde{\mathcal{K}})$.

Comparing the Modes of Acceptance I

Theorem 1

Let $L = L(\mathcal{K})$ *for a pushdown automaton* $\mathcal{K}$*. Then there exists a pushdown automaton* $\widetilde{\mathcal{K}}$ *such that* $L = N(\widetilde{\mathcal{K}})$.

Proof. Clearly, such a theorem is proved by providing a simulation, i.e., we want to modify $\mathcal{K}$ in a way such that the stack is emptied whenever $\mathcal{K}$ reaches a final state. In order to do so, we introduce a new state q_λ and a special stack symbol X_0 to avoid acceptance if $\mathcal{K}$ has emptied its stack without having reached a final state.

Comparing the Modes of Acceptance II

Let $\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ be any given pushdown automaton such that $L = L(\mathcal{K})$. We have to construct a pushdown automaton $\widetilde{\mathcal{K}}$ such that $L = N(\widetilde{\mathcal{K}})$. We set

$$\widetilde{\mathcal{K}} = [Q \cup \{q_\lambda, \tilde{q}_0\}, \Sigma, \Gamma \cup \{X_0\}, \tilde{q}_0, X_0, \tilde{\delta}, \emptyset] \ ,$$

where $\tilde{\delta}$ is defined as follows:

Comparing the Modes of Acceptance II

Let $\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ be any given pushdown automaton such that $L = L(\mathcal{K})$. We have to construct a pushdown automaton $\widetilde{\mathcal{K}}$ such that $L = N(\widetilde{\mathcal{K}})$. We set

$$\widetilde{\mathcal{K}} = [Q \cup \{q_\lambda, \tilde{q}_0\}, \Sigma, \Gamma \cup \{X_0\}, \tilde{q}_0, X_0, \tilde{\delta}, \emptyset] \,,$$

where $\tilde{\delta}$ is defined as follows:

$$\begin{aligned}
\tilde{\delta}(\tilde{q}_0, \lambda, X_0) &= \{(q_0, k_0 X_0)\} \\
\tilde{\delta}(q, a, Z) &= \delta(q, a, Z) \text{ for all } q \in Q \setminus F,\ a \in \Sigma \cup \{\lambda\},\ Z \in \Gamma \\
\tilde{\delta}(q, a, Z) &= \delta(q, a, Z) \text{ for all } q \in F,\ a \in \Sigma \text{ and } Z \in \Gamma \\
\tilde{\delta}(q, \lambda, Z) &= \delta(q, \lambda, Z) \cup \{(q_\lambda, \lambda)\} \text{ for all } q \in F,\ Z \in \Gamma \cup \{X_0\} \\
\tilde{\delta}(q_\lambda, \lambda, Z) &= \{(q_\lambda, \lambda)\} \text{ for all } Z \in \Gamma \cup \{X_0\} \,.
\end{aligned}$$

Comparing the Modes of Acceptance III

By construction, when starting $\widetilde{\mathcal{K}}$, it is entering the initial instantaneous description of $\mathcal{K}$ but pushes additionally its own stack symbol X_0 into the stack. Then $\widetilde{\mathcal{K}}$ simulates $\mathcal{K}$ until it reaches a final state. If $\mathcal{K}$ reaches a final state, then $\widetilde{\mathcal{K}}$ can either continue to simulate $\mathcal{K}$ or it can change its state to q_λ. If $\widetilde{\mathcal{K}}$ is in q_λ, it can empty the stack and thus accept the input.

Comparing the Modes of Acceptance IV

So, let $x \in L(\mathcal{K})$. Then, there is a computation such that

$$(q_0, x, k_0) \xrightarrow[\mathcal{K}]{*} (q, \lambda, \gamma) \text{ for a } q \in F .$$

Consider $\widetilde{\mathcal{K}}$ on input x. By definition, $\widetilde{\mathcal{K}}$ starts in state $\tilde{q}_0$ and with stack symbol X_0. Using the first spontaneous transition,

$$(\tilde{q}_0, x, X_0) \xrightarrow[\widetilde{\mathcal{K}}]{1} (q_0, x, k_0 X_0) .$$

Next, $\widetilde{\mathcal{K}}$ can simulate every step of $\mathcal{K}$'s work; hence

$$(\tilde{q}_0, x, X_0) \xrightarrow[\widetilde{\mathcal{K}}]{1} (q_0, x, k_0 X_0) \xrightarrow[\widetilde{\mathcal{K}}]{*} (q, \lambda, \gamma X_0)$$

Using the last two transitions in the definition of $\tilde{\delta}$ we obtain

$$(q_0, x, k_0 X_0) \xrightarrow[\widetilde{\mathcal{K}}]{*} (q_\lambda, \lambda, \lambda) , \text{ thus } x \in N(\widetilde{\mathcal{K}}) . \quad \blacksquare$$

Comparing the Modes of Acceptance V

Theorem 2

Let $\mathrm{L} = \mathrm{N}(\mathcal{K})$ *for a pushdown automaton* $\mathcal{K}$*. Then there exists a pushdown automaton* $\widetilde{\mathcal{K}}$ *such that* $\mathrm{L} = \mathrm{L}(\widetilde{\mathcal{K}})$.

Comparing the Modes of Acceptance V

Theorem 2

Let $\mathrm{L} = \mathrm{N}(\mathcal{K})$ *for a pushdown automaton* $\mathcal{K}$*. Then there exists a pushdown automaton* $\widetilde{\mathcal{K}}$ *such that* $\mathrm{L} = \mathrm{L}(\widetilde{\mathcal{K}})$*.*

Proof. Again the proof is done by simulation. The pushdown automaton $\widetilde{\mathcal{K}}$ will simulate $\mathcal{K}$ until it detects that $\mathcal{K}$ has emptied its stack. If this happens then $\widetilde{\mathcal{K}}$ will enter a final state and stop. The formal description is given in the book. ∎

PDA and $\mathcal{CF}$ I

So far, we have only dealt with pushdown automata and their acceptance behavior. It remains to clarify what languages are accepted by pushdown automata. This is done by the following theorem. Recall that a derivation is said to be a leftmost derivation if at each step in derivation a production is applied to the leftmost nonterminal.

PDA and $\mathcal{CF}$ II

Theorem 3

Let $\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, \emptyset]$ *be any pushdown automaton and let* $L = N(\mathcal{K})$. *Then* L *is context-free.*

The Proof I

Proof. Let $\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, \emptyset]$ be any pushdown automaton. For proving that L defined as $L = N(\mathcal{K})$ is context-free, we have to construct a context-free grammar $\mathcal{G}$ such that $L = L(\mathcal{G})$. We set $\mathcal{G} = [\Sigma, N, \sigma, P]$, where N is defined as follows. The elements of N are denoted by $[q, A, p]$, where $p, q \in Q$ and $A \in \Gamma$. Additionally, N contains the symbol σ. Next, we have to define the set of productions. P contains the following rules.

(1) $\sigma \rightarrow [q_0, k_0, q]$ for every $q \in Q$,

(2) $[q, A, q_{m+1}] \rightarrow a[q_1, B_1, q_2][q_2, B_2, q_3] \cdots [q_m, B_m, q_{m+1}]$
for $q_1, \ldots, q_{m+1} \in Q$ and $A, B_1, \ldots, B_m \in \Gamma$ such that
$(q_1, B_1 B_2 \cdots B_m) \in \delta(q, a, A)$ provided $m > 0$.
If $m = 0$ then the production is $[q, A, q_1] \rightarrow a$.

The Proof II

To understand the proof it helps to know that the nonterminals and productions of $\mathcal{G}$ have been defined in a way such that a leftmost derivation in $\mathcal{G}$ of a string x is a simulation of the pushdown automaton $\mathcal{K}$ when fed the input x. In particular, the nonterminals that appear in any step of a leftmost derivation in $\mathcal{G}$ correspond to the symbols on the stack of $\mathcal{K}$ at a time when $\mathcal{K}$ has seen as much of the input as the grammar has already generated. In other words, our intention is that $[q, A, p]$ derives x if and only if x causes $\mathcal{K}$ to erase an A from its stack by some sequence of moves beginning in state q and ending in state p.

The Proof III

For showing that $L(\mathcal{G}) = N(\mathcal{K})$ we prove inductively

$$[q, A, p] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x \quad \text{if and only if} \quad (q, x, A) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda)\,. \tag{1}$$

First, we show by induction on i that if

$$(q, x, A) \overset{i}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda) \text{ then } [q, A, p] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x\,.$$

The Proof IV

For the induction basis let $i = 1$. In order to have

$(q, x, A) \xrightarrow[\mathcal{K}]{1} (p, \lambda, \lambda)$ it must hold that $(p, \lambda) \in \delta(q, x, A)$.

Consequently, either we have $x = \lambda$ or $x \in \Sigma$. In both cases, by construction of P we know that $([q, A, p] \rightarrow x) \in P$. Hence,

$[q, A, p] \xRightarrow[\mathcal{G}]{1} x$. This proves the induction basis.

The Proof V

Now suppose $i > 0$. Let $x = ay$ and

$$(q, ay, A) \xrightarrow[\mathcal{K}]{1} (q_1, y, B_1 B_2 \cdots B_n) \xrightarrow[\mathcal{K}]{i-1} (p, \lambda, \lambda) .$$

The string y can be written as $y = y_1 y_2 \cdots y_n$, where y_j has the effect of popping B_j from the stack, possibly after a long sequence of moves. That is, let y_1 be the prefix of y at the end of which the stack first becomes as short as $n - 1$ symbols. Let y_2 be the symbols of y following y_1 such that at the end of y_2 the stack first becomes as short as $n - 2$ symbols, and so on.
This arrangement is displayed below.

The Proof VI

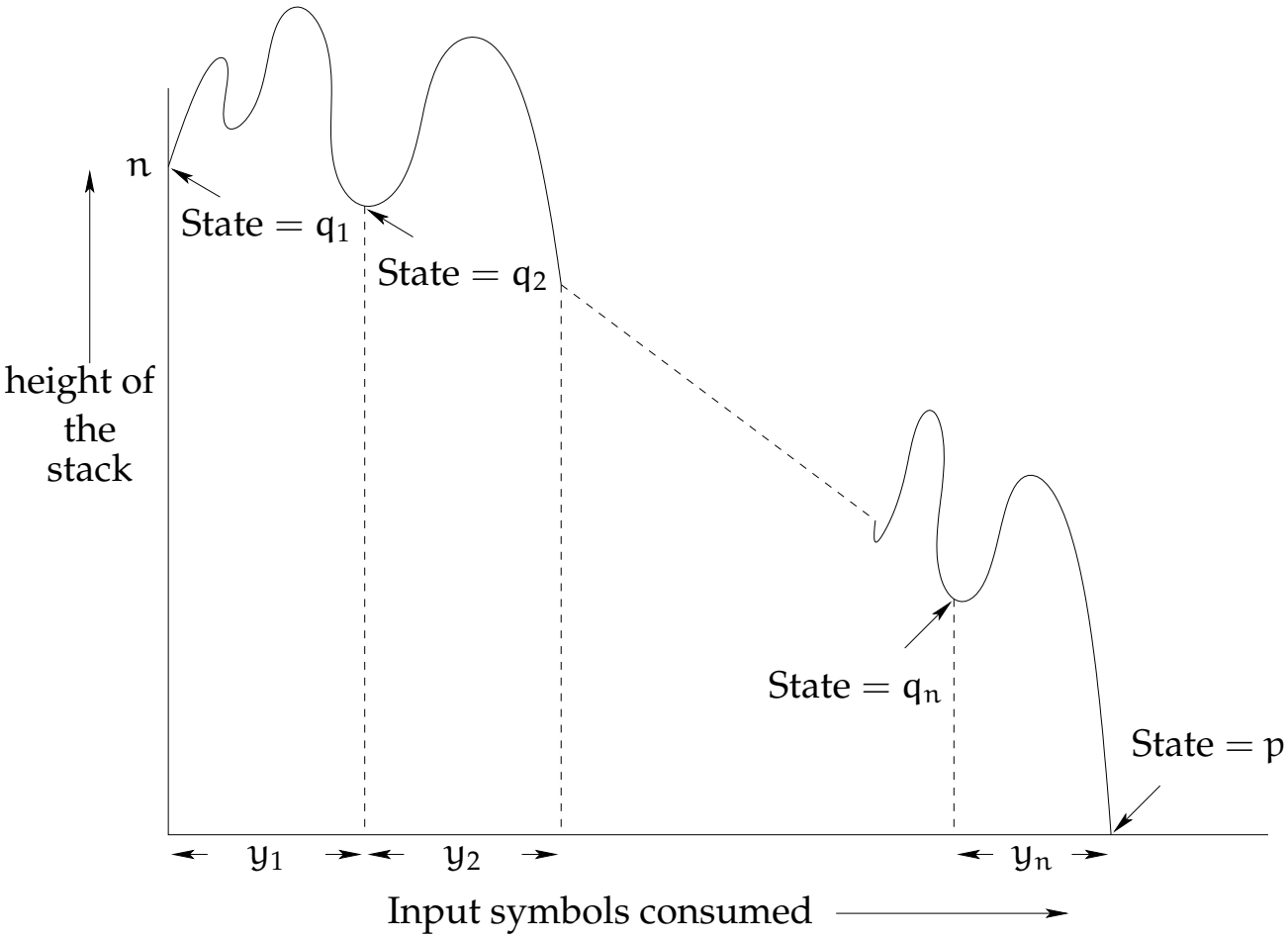

Figure 2: Hight of stack as a function of input symbols consumed

The Proof VII

Note that B_1 does not need to be the nth stack symbol from the bottom during the entire time y_1 is being read by $\mathcal{K}$, since B_1 may be changed if it is at the top of the stack and is replaced by one or more symbols. However, none of $B_2B_3 \cdots B_n$ are ever on top while y_1 is being read. Thus, none of $B_2B_3 \cdots B_n$ can be changed or influence the computation while y_1 is processed. In general, B_j remains on the stack unchanged while $y_1 \cdots y_{j-1}$ is read.

The Proof VIII

There exists states $q_2, q_3, \ldots, q_{n+1}$, where $q_{n+1} = p$ such that

$$(q_j, y_j, B_j) \xrightarrow[\mathcal{K}]{*} (q_{j+1}, \lambda, \lambda)$$

by fewer than i moves. Note that q_j is the state entered when the stack first becomes as short as $n - j + 1$. Thus, we can apply the induction hypothesis and obtain

$$[q_j, B_j, q_{j+1}] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} y_j \text{ for } 1 \leqslant j \leqslant n\,.$$

Recalling the original move

$$(q, ay, A) \xrightarrow[\mathcal{K}]{1} (q_1, y, B_1 B_2 \cdots B_n) \text{ we know that}$$

$$[q, A, p] \;\Rightarrow\; a[q_1, B_1, q_2][q_2, B_2, q_3] \cdots [q_n, B_n, q_{n+1}]\,,$$

and thus $[q, A, p] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} a y_1 y_2 \cdots y_n = x$, and ($\Leftarrow$) of (1) is shown.

The Proof IX

For showing the necessity of (1) suppose $[a, A, p] \overset{i}{\underset{\mathcal{G}}{\Longrightarrow}} x$. We

prove by induction on i that $(q, x, A) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda)$.

For the induction basis we again take $i = 1$. If $[a, A, p] \overset{1}{\underset{\mathcal{G}}{\Longrightarrow}} x$,

then $([a, A, p] \rightarrow x) \in P$ and therefore $(p, \lambda) \in \delta(q, x, A)$.

The Proof X

Next, for the induction step suppose

$$[q, A, p] \Rightarrow a[q_1, B_1, q_2] \cdots [q_n, B_n, q_{n+1}] \overset{i-1}{\underset{\mathcal{G}}{\Longrightarrow}} x\,,$$

where $q_{n+1} = p$. Then we may write x as $x = ax_1 \cdots x_n$, where

$[q_j, B_j, q_{j+1}] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x_j$ for $j = 1, \ldots, n$. Moreover, each derivation

takes fewer than i steps. Thus, we can apply the induction hypothesis and obtain

$$(q_j, x_j, B_j) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (q_{j+1}, \lambda, \lambda) \text{ for } j = 1, \ldots, n\,.$$

The Proof XI

If we insert $B_{j+1} \cdots B_n$ at the bottom of each stack in the above sequence of instantaneous descriptions we see that

$$(q_j, x_j, B_j B_{j+1} \cdots B_n) \xrightarrow[\mathcal{K}]{*} (q_{j+1}, \lambda, B_{j+1} \cdots B_n) \,. \tag{2}$$

Furthermore, from the first step in the derivation of x from $[q, A, p]$ we know that

$$(q, x, A) \xrightarrow[\mathcal{K}]{1} (q_1, x_1 x_2 \cdots x_n, B_1 B_2 \cdots B_n)$$

is a legal move of $\mathcal{K}$. Therefore, from this move and from (2) for $j = 1, 2, \ldots, n$ we directly obtain

$$(q, x, A) \xrightarrow[\mathcal{K}]{*} (p, \lambda, \lambda) \,.$$

This proves the necessity of (1).

The Proof XII

The proof concludes with the observation that (1) with $q = q_0$ and $A = k_0$ says

$$[q_0, k_0, p] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x \quad \text{if and only if} \quad (q_0, x, k_0) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda) .$$

This observation together with rule (1) of the construction of P

says that $\sigma \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x$ if and only if $(q_0, x, k_0) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda)$ for some

state p. Therefore, we finally arrive at $x \in L(\mathcal{G})$ if and only if $x \in N(\mathcal{K})$. ∎

Thank you!