Language Processors F29LP2, Lecture 6

Jamie Gabbay

February 2, 2014

Pushdown automata (PDA)

A pushdown automaton (PDA) is an NFA (non-deterministic finite automaton) with a stack (i.e. with some memory).

PDAs recognise exactly context-free languages.

A PDA consists of:

- A stack (a string of symbols). The PDA has access only to the leftmost symbol–the top–of the stack.
  During one move of the PDA, the leftmost symbol may be popped (removed from) or pushed (added to) the top of the stack.
- Input tape. Input string to be scanned symbol-by-symbol. ε-moves are also possible.
- Finite state control unit. A non-deterministic finite state machine whose transitions depend on the input symbol and the topmost stack symbol.

PDA

Normal moves. Depending on:

(a) the current state of the control unit,
(b) the next input letter, and
(c) the topmost symbol on the stack

the PDA may

(A) change the state,
(B) pop the topmost element from the stack,
(C) push new symbols to the stack, and
(D) move to the next input symbol.

Spontaneous ε-moves don't have (b) and (D); they don't use the input tape.

Example

Consider a PDA to recognise $L = \{a^n b^n \mid n \geq 1\}$.

It has two states $\{S_a, S_b\}$ and alphabet with two symbols $\{A, Z_0\}$.

It starts at S_a with stack containing one symbol Z_0, called the start symbol of the stack.

Moves are summarized in a table:

State	Top of stack	I n p u t S y m b o l		
		a	b	ε
S_a	Z_0	Push A, stay at S_a	—	—
S_a	A	Push A stay at S_a	Pop A go to S_b	—
S_b	A	—	Pop A stay at S_b	—
S_b	Z_0	—	—	Pop Z_0 stay at S_b

An input word is accepted if the PDA reaches the empty stack after reading all input symbols.

Instantaneous descriptions (ID)

An ID is a ‘snapshot’ (‘core dump’) of the PDA. It is a triple

$$(q, w, \gamma)$$

where

- q is the state of the PDA,
- w is the remaining input, i.e. the suffix of the original input that has not been used yet, and
- γ is the content of the stack.

The ID contains all that is needed in subsequent steps of the computation.

Instantaneous descriptions (ID)

Write

$$(q_1, w_1, \gamma_1) \vdash (q_2, w_2, \gamma_2)$$

when a move exists taking the first ID to the second ID.

Write

$$(q_1, w_1, \gamma_1) \vdash^* (q_2, w_2, \gamma_2)$$

when a (possibly empty) sequence of moves exists from the first ID to the second ID.

Write

$$(q_1, w_1, \gamma_1) \vdash^n (q_2, w_2, \gamma_2)$$

when the first ID becomes the second ID in exactly n moves.

Our example: acceptance by empty stack

For example, when accepting input string *aaabbb* our sample PDA evolves as follows:

$$\begin{array}{rcl}
(S_a, aaabbb, Z_0) & \vdash & (S_a, aabbb, AZ_0) \\
& \vdash & (S_a, abbb, AAZ_0) \\
& \vdash & (S_a, bbb, AAAZ_0) \\
& \vdash & (S_b, bb, AAZ_0) \\
& \vdash & (S_b, b, AZ_0) \\
& \vdash & (S_b, \varepsilon, Z_0) \\
& \vdash & (S_b, \varepsilon, \varepsilon)
\end{array}$$

The word is accepted when all input letters are consumed and the end stack is empty.

Call this acceptance by empty stack.

Acceptance by final state

There is another way of defining which words are accepted.

Some states may be called final states, and a word is accepted when after reading all input letters, the PDA is in an accepting state.

This is called acceptance by final state.

The two modes of acceptance are equivalent: If there is a PDA that recognises language L using empty stack then there (effectively) exists another PDA that recognises L using final state(s), and vice versa.

Formal definition of a PDA

A pushdown automaton M consists of

$$(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

where

- Q is a finite state set,
- Σ is the finite input alphabet,
- Γ is the finite stack alphabet,
- $q_0 \in Q$ is the initial state,
- $Z_0 \in \Gamma$ is the start symbol of the stack,
- $F \subseteq Q$ is the set of final states,
- δ is the transition function from

  $$Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$$

  to finite subsets of

  $$Q \times \Gamma^*.$$

$\delta(q, a, Z)$ is a set of possible outcomes; PDAs are non-deterministic.

Transitions I

The interpretation of executing a transition

$$(p, \gamma) \in \delta(q, a, Z)$$

(where $p, q \in Q$, $a \in \Sigma$, $Z \in \Gamma$, $\gamma \in \Gamma^*$) is that

- in state q, reading input letter a, and Z on the top of the stack, the PDA
- goes to state p, moves to the next input symbol, and replaces Z by γ on top of the stack. The leftmost symbol of γ will be the new top of the stack (if $\gamma \neq \varepsilon$).

In IDs, this means that the move

$$(q, aw, Z\alpha) \vdash (p, w, \gamma\alpha)$$

is allowed for all $w \in \Sigma^*$ and $\alpha \in \Gamma^*$.

Transitions II

The interpretation of executing a transition

$$(p, \gamma) \in \delta(q, \varepsilon, Z)$$

is that

- in state q and Z on the top of the stack, the PDA
- goes to state p and replaces Z by γ on top of the stack. No input symbol is consumed, and the transition can be used regardless of the current input symbol.

In terms of IDs, this transition means that the move

$$(q, w, Z\alpha) \vdash (p, w, \gamma\alpha)$$

is allowed for all $w \in \Sigma^*$ and $\alpha \in \Gamma^*$.

Acceptance by empty stack

A word $w \in \Sigma^*$ is accepted by PDA M by empty stack, when

$$(q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)$$

for some $q \in Q$. Note that q can be any state, final or non-final.

Write $N(M)$ for the language recognized by PDA M using empty stack:

$$N(M) = \{w \in \Sigma^* \mid \exists q \in Q.\ (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}.$$

Acceptance by final state

A word $w \in \Sigma^*$ is accepted by PDA M by final state, when

$$(q_0, w, Z_0) \vdash^* (q, \varepsilon, \gamma)$$

for some $q \in F$, and $\gamma \in \Gamma^*$. Note that q has to be a final state but the stack does not need to be empty.

Write $L(M)$ for the language recognized by PDA M using final state:

$$L(M) = \{w \in \Sigma^* \mid \exists q \in F, \gamma \in \Gamma^*.\ (q_0, w, Z_0) \vdash^* (q, \varepsilon, \gamma)\}.$$

Examples

Our sample PDA is

$$M = (\{S_a, S_b\}, \{a, b\}, \{Z_0, A\}, \delta, S_a, Z_0, \emptyset)$$

where

$$\begin{array}{lcl}
\delta(S_a, a, Z_0) & = & \{(S_a, AZ_0)\} \\
\delta(S_a, a, A) & = & \{(S_a, AA)\} \\
\delta(S_a, b, A) & = & \{(S_b, \varepsilon)\} \\
\delta(S_b, b, A) & = & \{(S_b, \varepsilon)\} \\
\delta(S_b, \varepsilon, Z_0) & = & \{(S_b, \varepsilon)\}
\end{array}$$

and all other sets $\delta(q, a, Z)$ are empty. It does not matter which set we choose as the set of final states, since we use acceptance by empty stack. (Choose, for example $F = \emptyset$.) We have

$$N(M) = \{a^n b^n \mid n \geq 1\}.$$

Examples

We construct a PDA M such that

$$N(M) = \{w \mid w \in \{a,b\}^* \text{ and } w \text{ is a palindrome }\}.$$

The PDA reads symbols from the input and pushes them onto the stack.

At some point it guesses that it is in the middle of the input word, and starts popping letters from the stack and comparing them against the following input letters.

If all letters match, and the stack and the input string become empty at the same time, then the word was a palindrome!

Examples

We define:

$$M = (\{q_1, q_2\}, \{a, b\}, \{Z_0, A, B\}, \delta, q_1, Z_0, \emptyset)$$

where δ is

$$\begin{array}{lcl}
\delta(q_1, \varepsilon, Z_0) & = & \{(q_1, \varepsilon)\} \\
\delta(q_1, a, Z_0) & = & \{(q_1, AZ_0), (q_2, AZ_0), (q_2, Z_0)\} \\
\delta(q_1, b, Z_0) & = & \{(q_1, BZ_0), (q_2, BZ_0), (q_2, Z_0)\} \\
\delta(q_1, a, A) & = & \{(q_1, AA), (q_2, AA), (q_2, A)\} \\
\delta(q_1, b, A) & = & \{(q_1, BA), (q_2, BA), (q_2, A)\} \\
\delta(q_1, a, B) & = & \{(q_1, AB), (q_2, AB), (q_2, B)\} \\
\delta(q_1, b, B) & = & \{(q_1, BB), (q_2, BB), (q_2, B)\} \\
\delta(q_2, b, B) & = & \{(q_2, \varepsilon)\} \\
\delta(q_2, a, A) & = & \{(q_2, \varepsilon)\} \\
\delta(q_2, \varepsilon, Z_0) & = & \{(q_2, \varepsilon)\}
\end{array}$$

State q_1 is used in the first half of the input word, state q_2 in the second half.

Examples

The three possible outcomes of some transitions have the following roles in accepting computations:

- use the first transition, if the input letter is before the end of the first half of the input word.
- use the second transition, if the input word has even length and the current input letter is the last letter of the first half, and
- use the third transition, if the input word has odd length and the current input letter is exactly in the middle of the input word.

Examples

For example, the word *bab* is accepted because

$$(q_1, bab, Z_0) \vdash (q_1, ab, BZ_0) \vdash (q_2, b, BZ_0) \vdash (q_2, \varepsilon, Z_0) \vdash (q_2, \varepsilon, \varepsilon),$$

and the word *abba* is accepted, because

$$\begin{aligned}(q_1, abba, Z_0) \vdash (q_1, bba, AZ_0) \vdash (q_2, ba, BAZ_0)& \\ \vdash (q_2, a, AZ_0) \vdash (q_2, \varepsilon, Z_0) \vdash (q_2, \varepsilon, \varepsilon)&\end{aligned}$$

Deterministic PDAs

Call a PDA deterministic when every ID has at most one possible move.

This means that

- if $\delta(q, \varepsilon, Z)$ is non-empty then $\delta(q, a, Z)$ is empty for every input letter a, and
- All $\delta(q, a, Z)$ and $\delta(q, \varepsilon, Z)$ contain at most one element.

The first condition states that there is no choice between ε-move and non-ε-move.

If one can make a move without reading an input letter, then that is the only possible move.

Our first example was deterministic, while the PDA from slide 15 is non-deterministic.

Deterministic PDAs

Nota bene: There exist languages that are recognised by non-deterministic PDA but not by any deterministic PDA. Language

$$\{a^n b^m c^k \mid n = m \text{ or } n = k\}$$

is an example.

This is different from finite automata, where determinism was equivalent to non-determinism.

Languages that can be recognised by deterministic PDA (using final states) are called deterministic context-free languages.

Converting a context-free grammar to a PDA

PDAs recognise precisely the context-free languages.

In fact, context-free languages are recognised by PDAs having just one state.

We illustrate this by showing how a context-free grammar can be simulated using only the stack.

Consider the grammar $G = (V, T, P, S)$ where $V = \{S\}$, $T = \{a, b\}$ and P contains productions

$$S \longrightarrow aSbS \mid bSaS \mid \varepsilon.$$

We construct a PDA M that recognises $L(G)$ using empty stack.

The stack symbols of M are the terminals and non-terminals of G.

At all times, the content of the stack is a suffix of a sentential form by G.

Converting a context-free grammar to a PDA

Initially the stack contains the start symbol of the grammar.

- If the topmost symbol of the stack is a terminal symbol it is compared against next input letter. If they are identical, the symbol is popped from the stack and the machine moves to the next input letter. If they are different the simulation halts.
- If the topmost symbol of the stack is a variable the machine rewrites it by a righthand side of a production using an ε-move.

Converting a context-free grammar to a PDA

We get the machine

$$M = (\{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \emptyset)$$

with δ is defined as follows:

Transitions for terminals on the top of the stack are

$$\begin{array}{rcl} \delta(q, a, a) & = & \{(q, \varepsilon)\}, \\ \delta(q, a, b) & = & \emptyset, \\ \delta(q, b, b) & = & \{(q, \varepsilon)\}, \\ \delta(q, b, a) & = & \emptyset, \end{array}$$

Transitions for the non-terminal S on top of the stack are

$$\delta(q, \varepsilon, S) = \{(q, aSbS), (q, bSaS), (q, \varepsilon)\}.$$

There are no other transitions.

Converting a context-free grammar to a PDA

This machine simulates leftmost derivations of G on the stack.

Terminals generated on the left end have to match the input.

A non-terminal at the left extreme is rewritten in the stack using a production of the grammar.

Recall the productions

$$S \longrightarrow aSbS \mid bSaS \mid \varepsilon$$

and ...

Converting a context-free grammar to a PDA

…compare the leftmost derivation of *aababb* on the left,

with the accepting computation on the right:

$$\begin{aligned} S &\Longrightarrow aSbS \\ &\Longrightarrow aaSbSbS \\ &\Longrightarrow aabSbS \\ &\Longrightarrow aabaSbSbS \\ &\Longrightarrow aababSbS \\ &\Longrightarrow aababbS \\ &\Longrightarrow aababb \end{aligned}$$

$$\begin{aligned} (q, aababb, S) &\vdash (q, aababb, aSbS) \\ &\vdash (q, ababb, SbS) \\ &\vdash (q, ababb, aSbSbS) \\ &\vdash (q, babb, SbSbS) \\ &\vdash (q, babb, bSbS) \\ &\vdash (q, abb, SbS) \\ &\vdash (q, abb, aSbSbS) \\ &\vdash (q, bb, SbSbS) \\ &\vdash (q, bb, bSbS) \\ &\vdash (q, b, SbS) \\ &\vdash (q, b, bS) \\ &\vdash (q, \varepsilon, S) \\ &\vdash (q, \varepsilon, \varepsilon). \end{aligned}$$