TDDD14/TDDD85 Lecture 9: Pushdown Automata

Jonas Wallgren

Abstract

This lecture inroduces the automata handling context-free languages (CFLs).

1 Introductory example

Recall figure 4 in lecture 1. To be able to handle e.g. nested brackets we introduce a stack to the mechanism of a DFA/NFA. Every time we read a symbol (or ε) we may check the stack before and we may change the stack afterwards. (E.g. in recognizing $\{0^n1^n|n \geqslant 0\}$ you can push an x everytime you read a 0 and pop everytime you read a 1.) The automaton is called a push-down automaton, PDA. It is important to know that in its standard form a PDA is non-deterministic, it's an NPDA. It is the type of automaton that accepts CFLs. The deterministic form, DPDA, accepts a restricted form of CFLs. We will return to such automata later.

The transition function/relation in this case takes an argument triple: current state, read symbol (or ε), and stack top. It gives a result pair: new state and result on stack.

Example 1. *A push-down automaton for* $L = \{wcw^R|w \in \{0,1\}^*\}$, *where* x^R *is* w *reversed, i.e. palindromes over* $\{0,1\}$ *with a* c *in the center.*

- *Read 0, 1, or c.*
- *Put B, G, or R on the stack.*[1]
- *Start in state* q_1.
- *Start with R on the stack.*

The following table shows the behaviour of the transition relation (In this case it is a (partial) function.):

[1] The stack symbols are independent of the symbols in the analyzed string.

Top of stack	State	Symbol read		
		0	1	c
B	q_1	$PushB$ q_1	$PushG$ q_1	q_2
	q_2	Pop q_2		
G	q_1	$PushB$ q_1	$PushG$ q_1	q_2
	q_2		Pop q_2	
R	q_1	$PushB$ q_1	$PushG$ q_1	q_1
	q_2	Without reading : Pop q_2		

To show how such an automaton tries to accept a string the steps are illustrated in the following way: On the first line you see the start state, the string, and the start stack. Then each successive line shows the current state, the unread part of the string, and the current stack contents after performing the action specified by the transition relation on the preceding line.

Example 2. *Does 01c10 belong to the language?*

State	Remaining string	Stack[2]	Comment
q_1	$01c10$	R	The start situation
q_1	$1c10$	BR	According to column 0, row Rq_1 in the table
q_1	$c10$	GBR	According to column 1, row Bq_1 in the table
q_2	10	GBR	etc.
q_2	0	BR	
q_2		R	
q_2			String read, stack empty : Accept!

So, the whole string could be read, and when it was completely read the stack was empty, which is one of two ways that a string coud be accepted. A string is also accepted if you end up in a final state.

Now, you can see how the different symbols and states work in this example, the "meaning" of them: All symbols before c are read in q_1, all symbols after c are read in q_2. If you read a 0 before the c you push a B on the stack. You can read a 0 after the c only if there is a B on the stack top. Correspondingly for 1 and G.

2 PDA

After the introductory example we now can define a PDA.

Definition 1. *A PDA is a septuple* $\langle Q,\Sigma,\Gamma,\delta,s,\bot,F\rangle$*, where*

[2]The stack top is to the left.

- Q=*set of states*
- Σ=*(input) alphabet*
- Γ=*stack alphabet*
- s=*start state*$\in Q$
- F=*final states*$\subseteq Q$
- $\perp$=*start stack symbol*
- δ=*transition relation*$\subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*))$

The first part of a δ element is a triple of a state, a read symbol or ε, and a stack symbol. The second part is a pair of a state and a *string* of stack symbols. The result of a use of δ thus is not just one new symbol on the stack, but a sequence of stack symbols (or nothing). That will be used in the transformation from a CFG to a PDA.

$\langle\langle p, a, A\rangle, \langle q, B_1, B_2 \dots B_n\rangle\rangle \in \delta$ means:
In state p with A on top of the stack:

- Read a,
- go to state q,
- change the stack top A to $B_1 B_2 \dots B_n$ (the left end being the new stack top).

$\langle\langle p, \varepsilon, A\rangle, \langle q, B_1, B_2 \dots B_n\rangle\rangle \in \delta$ means:
In state p with A on top of the stack, without reading anything:

- Go to state q,
- change the stack top A to $B_1 B_2 \dots B_n$ (the left end being the new stack top).

Example 3. *The PDA in* **Example 1** *can formally be specified as* $\{\{q_1, q_2\}, \{0, 1, c\}, \{B, G, R\}, \delta, q_1, R, \emptyset\}$*, where* δ *is defined according to the table.*

2.1 Configurations

What we handled in **Example 2** are called configurations.

Definition 2. *A configuration is a triple* $\langle q, x, \gamma\rangle$*, where:*

- q=*state*$\in Q$
- x=*string*$\in \Sigma^*$

- γ=*stack*$\in \Gamma^*$

Describing the action of a PDA could be done by showing the steps of the following relation:

Definition 3. *The next-configuration relation* $\rightarrow$
If $\langle\langle p, a, A\rangle, \langle q, \gamma\rangle\rangle \in \delta$ *then* $\langle p, ay, A\beta\rangle \rightarrow \langle q, y, \gamma\beta\rangle\rangle$
If $\langle\langle p, \varepsilon, A\rangle, \langle q, \gamma\rangle\rangle \in \delta$ *then* $\langle p, y, A\beta\rangle \rightarrow \langle q, y, \gamma\beta\rangle\rangle$

From **Example 2**:
$\langle q_1, 01c01, R\rangle \rightarrow \langle q_1, 1c01, BR\rangle$

Many steps between configurations is written $\xrightarrow{*}$. Using the transition relation of PDA M in n steps is written $\underset{M}{\xrightarrow{n}}$.

2.2 Acceptance

A PDA can accept a string if the stack is empty or if it reaches a final state.

Definition 4. *A PDA accepts the string x if* $\langle s, x, \perp\rangle \xrightarrow{*} \langle q, \varepsilon, \gamma\rangle$ *when* $q \in F$.

Definition 5. *A PDA accepts the string x if* $\langle s, x, \perp\rangle \xrightarrow{*} \langle q, \varepsilon, \varepsilon\rangle$ *when* $q \in Q$.

So, there are two ways to define acceptance for PDAs. I have found no motivation in the literature but a plausible explanation is that if you think mainly in a DFA way you want the concept of final states and if you think that the stack is the most important part of a PDA you want the empty stack to signal acceptance.

Definition 6. *The language of a PDA M.*
L(M)= the set of all strings that are accepted by M in either way.

3 Equivalence between ways of acceptance

We will prove that if there is a PDA accepting the string x with empty stack then there is a PDA accepting it in final state and v.v., i.e. the two ways of accepting a string are equally powerful, they define the same class of languages.

We will perform the two proofs (the two directions of the implications making up the equivalence) somewhat in parallel.

Our starting point is the PDA M=$\langle Q, \Sigma, \Gamma, \delta, s, \perp, F\rangle$. It accepts strings in either way.

Two new sets are defined:

If M accepts	If M accepts
with empty stack :	in final state :
$G = Q$	$G = F$
$\Delta = \{\perp\!\!\!\perp\}$	$\Delta = \Gamma \cup \{\perp\!\!\!\perp\}$

We now perform an operation a litle bit like the one done when constructing the regular expression from a DFA — we add some extra handling in the start and in the end.

We define a new PDA $M' = \langle Q \cup \{u, t\}, \Sigma, \Gamma \cup \{\perp\!\!\!\perp\}, \delta', u, \perp\!\!\!\perp, \{t\}\rangle$, where

- u=new start state
- t=new final state
- $\perp\!\!\!\perp$=new stack bottom symbol
- δ'=new transition relation, defined as: $\delta' = \delta \cup \{ \quad \langle\langle u, \varepsilon, \perp\!\!\!\perp\rangle, \langle s, \perp\perp\!\!\!\perp\rangle\rangle,$
  $\langle\langle q, \varepsilon, A\rangle, \langle t, A\rangle\rangle$ for $q \in G, A \in \Delta,$
  $\langle\langle t, \varepsilon, A\rangle, \langle t, \varepsilon\rangle\rangle$ for $A \in \Gamma \cup \{\perp\!\!\!\perp\}\}$

So, the the first new element of δ' says that in M' you first just go from its start state to the start state of M and you put the stack bottom of M on the stack — to prepare for the simulation of M. The next new part of δ' says that when you are in an accepting situation in M you go to the accepting state of M'. The third part says that if there is anything left on the stack you could remove it.

So, M' accepts with empty stack and in final state.

Lemma 1. *If M accepts x with empty stack then M' accepts it.*

Proof. M accepts with empty stack: $\langle s, x, \perp\rangle \overset{n}{\underset{M}{\to}} \langle q, \varepsilon, \varepsilon\rangle$.

Then $\langle u, x, \perp\!\!\!\perp\rangle \overset{1}{\underset{M'}{\to}} \langle s, x, \perp\perp\!\!\!\perp\rangle \overset{n}{\underset{M'}{\to}} \langle q, \varepsilon, \perp\!\!\!\perp\rangle \overset{1}{\underset{M'}{\to}} \langle t, \varepsilon, \perp\!\!\!\perp\rangle \overset{1}{\underset{M'}{\to}} \langle t, \varepsilon, \varepsilon\rangle$. The first step as the new elements of δ' states, the second step since δ' contains δ, the last two steps as the new elements of δ' states.
So, $\langle u, x, \perp\!\!\!\perp\rangle \overset{*}{\underset{M'}{\to}} \langle t, \varepsilon, \varepsilon\rangle$, i.e. M' accepts x if M accepts it with empty stack. □

Lemma 2. *If M accepts x in final state then M' accepts it.*

Proof. M accepts in final state: $\langle s, x, \perp\rangle \overset{n}{\underset{M}{\to}} \langle q, \varepsilon, \gamma\rangle$, where $q \in F$.

Then $\langle u, x, \perp\!\!\!\perp\rangle \overset{1}{\underset{M'}{\to}} \langle s, x, \perp\perp\!\!\!\perp\rangle \overset{n}{\underset{M'}{\to}} \langle q, \varepsilon, \gamma\perp\!\!\!\perp\rangle \overset{1}{\underset{M'}{\to}} \langle t, \varepsilon, \gamma\perp\!\!\!\perp\rangle \overset{*}{\underset{M'}{\to}} \langle t, \varepsilon, \varepsilon\rangle$. The first step as the new elements of δ' states, the second step since δ' contains δ, the last two steps as the new elements of δ' states. □

Lemma 3. *If M accepts x then M' accepts it.*

Proof. Follows from **Lemma 1** and **Lemma 2**. □

Lemma 4. *If M' accepts x then M accepts it.*

Proof. Consider this sequence of steps:
$\langle u, x, \bot\!\!\bot \rangle \xrightarrow[M']{1} \langle s, x, \bot \bot\!\!\bot \rangle \xrightarrow[M']{n} \langle q, y, \gamma \bot\!\!\bot \rangle \xrightarrow[M']{1} \langle t, y, \gamma \bot\!\!\bot \rangle \xrightarrow[M']{*} \langle t, \varepsilon, \varepsilon \rangle$
The first step is the same initial one again. For M' to accept x it must in the process have reached a state $q \in G$ since that is the only way δ' can get the automaton into the state t. That is shown in step 2, with a y that maybe not is ε. The third step is just a move from q to t. Since we know that M' accepts x there must be some way to perform the last step. But in state t M cannot read anything, so y must be ε. So if the second step is read from the M perspective we get $\langle s, x, \bot \rangle \xrightarrow[M]{n} \langle q, \varepsilon, \gamma \rangle$, i.e. M accepts x. □

Theorem 1. *Accepting with empty stack and accepting in final state are equivalent.*

Proof. It follows from **Lemma 3** and **Lemma 4**. □

4 More to think about

1. Can you think of a context-free language where we (at least intuitively) need the power of non-determinism in a PDA to regognize the languge?

2. How could you simulate a PDA on an input string in your favourite programming language? How would such a simulation differ from simulating an NFA? How much memory does your implementation need?