Automata Theory (2B)

- PushDown Automata (PDA)

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice and Octave.

Deterministic Pushdown Automaton (PDA)

Deterministic PDA (1) – transition relation

An element (p, a, A, q, α) $\in \delta$ is a **transition** of **M**.

in **state** $p \in Q$,
on the **input** $a \in \Sigma \cup \{ \varepsilon \}$ and
with $A \in \Gamma$ as **topmost stack symbol**,

M may

- read a,
- change the **state** to q,
- pop A,
- replacing it by pushing $\alpha \in \Gamma^*$.

https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA (1) – input operations

on the **input** a ∈ Σ ∪ { ε }

the (Σ ∪ { ε }) component of the **transition relation**
is used to formalize that the PDA can
either read a letter from the input, Σ
or proceed leaving the input untouched. ε

https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA (2) – transition function

δ is the **transition function**,

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

into finite subsets of $Q \times \Gamma^*$

$\delta(p, a, A) \rightarrow (q, \alpha)$

Here δ (p, a, A) contains all possible actions in **state** p with A on the **stack**, while reading a on the **input**.

An element $(p, a, A, q, \alpha) \in \delta$ is a **transition** of **M**.

https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA (2) – transition function

δ is the **transition function**,

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

into finite subsets of $Q \times \Gamma^*$

$\delta(p, a, A) \rightarrow (q,\alpha)$

Here δ (p, a, A) contains all possible actions in **state** p with A on the **stack**, while reading a on the **input**.

One writes for example $\delta(p, a, A) = \{ (q, BA) \}$

precisely when $(q, BA) \in \{ (q, BA) \}$, $(q, BA) \in \delta(p, a, A)$

Because $((p, a, A), \{(q, BA)\}) \in \delta$.

Note that finite in this definition is essential.

$\delta(p, a, A) \rightarrow (q,\alpha)$

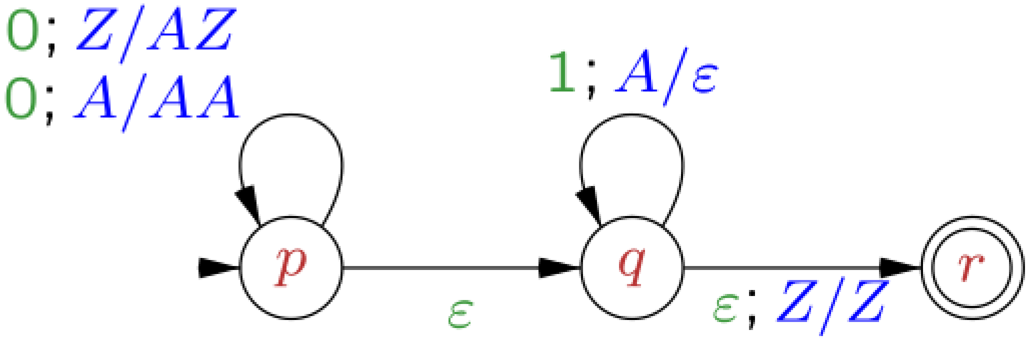
https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA Example (1) – description

The following is the formal description of the PDA
which recognizes the language $\{ 0^n 1^n \mid n \geq 0 \}$ by final state:

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, where

states:	$Q = \{p, q, r\}$
input alphabet:	$\Sigma = \{0, 1\}$
stack alphabet:	$\Gamma = \{A, Z\}$
start state:	$q_0 = p$
start stack symbol:	Z
accepting states:	$F = \{r\}$

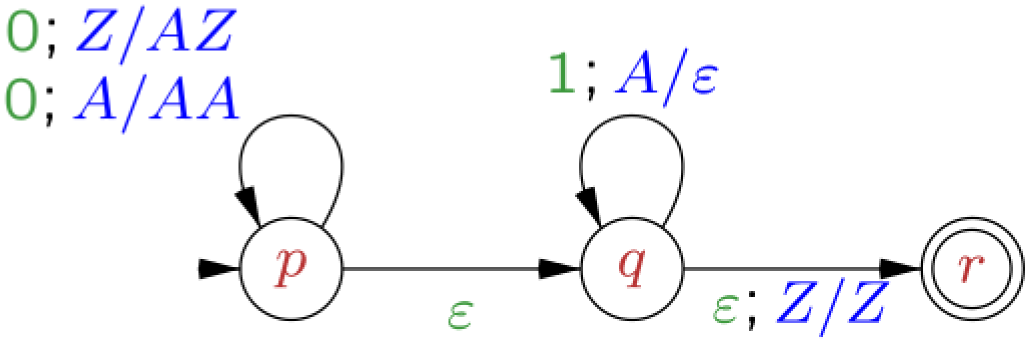


https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA Example (2) – instructions

The **transition relation** δ consists of the following six instructions:

(p, 0, Z, p, AZ)	0; Z/AZ, p→p
(p, 0, A, p, AA)	0; A/AA, p→p
(p, ϵ, Z, q, Z)	ϵ, Z/Z, p→q
(p, ϵ, A, q, A)	ϵ, A/A, p→q
(q, 1, A, q, ϵ)	1, A/ϵ, q→q
(q, ϵ, Z, r, Z)	ϵ, Z/Z, p→r



the instruction (p, a, A, q, α) by an edge from state p to state q labelled by a ; A / α (read a; replace A by α).

https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA Example (3) – instruction description

(p, 0, Z, p, AZ) , (p, 0, A, p, AA),	in state p any time the symbol 0 is read, one A is pushed onto the stack. Pushing symbol A on top of another A is formalized as replacing top A by AA (and similarly for pushing symbol A on top of a Z)
(p, ϵ, Z, q, Z), (p, ϵ, A, q, A),	at any moment the automaton may move from state p to state q.
(q, 1, A, q, ϵ),	in state q, for each symbol 1 read, one A is popped.
(q, ϵ, Z, r, Z).	the machine may move from state q to accepting state r only when the stack consists of a single Z.

https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA Computation (1) – ID

to formalize the **semantics** of the pushdown automaton

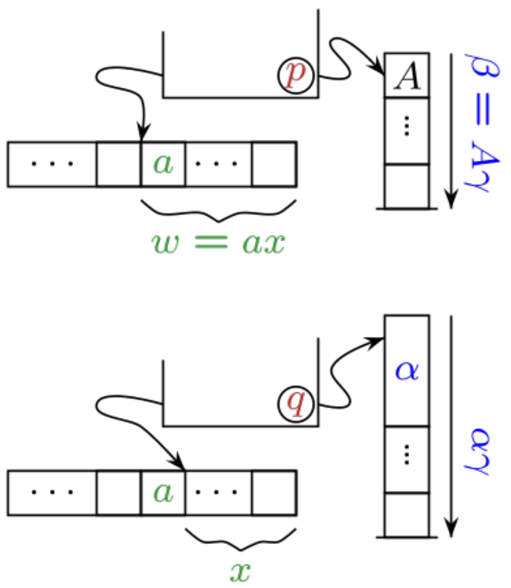
a description of the current situation is introduced.

Any 3-tuple (p , w , β) ∈ Q × Σ^* × Γ^* is called

an **instantaneous description** (ID) of

M = (Q, Σ, Γ, δ, q_0, Z, F) which includes

- the current **state**,
- the part of the **input** tape that has not been read, and
- the contents of the **stack** (topmost symbol written first).



https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA Computation (2) – step-relation

The **transition relation** δ defines

the **step-relation** $\vdash_M$ on **instantaneous descriptions**.

For instruction (p, a, A, q, α) ∈ δ

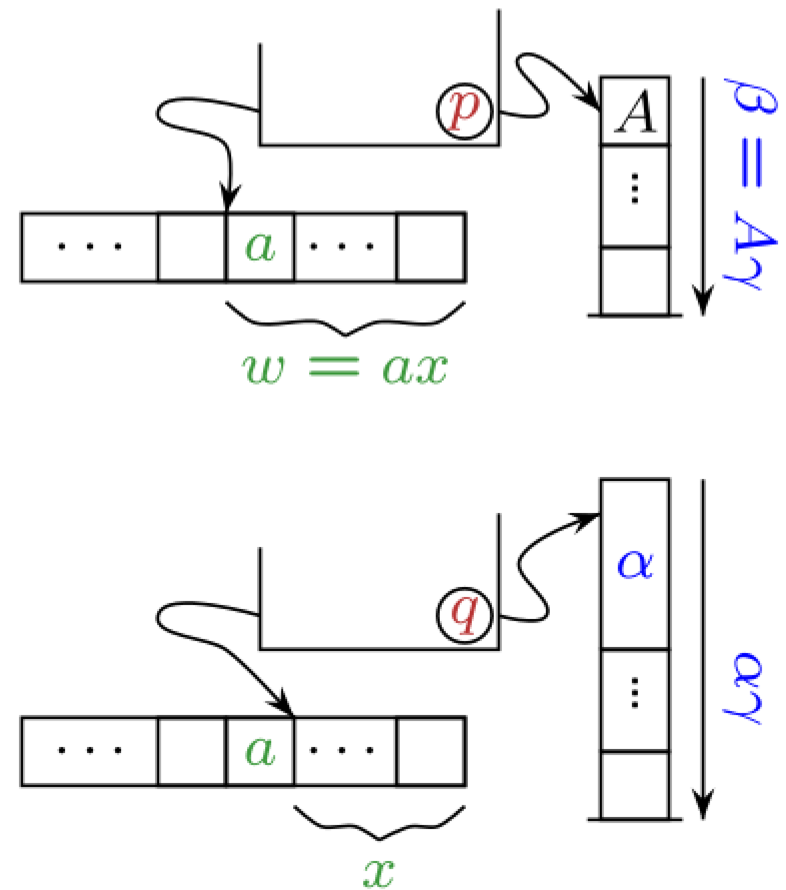
there exists a step (p , ax, Aγ) ⊢ M (q, x , αγ),

for every x ∈ Σ^* and every γ ∈ Γ^* .

p, q : states

ax, x : inputs

Aγ, αγ : stack elementes



https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA Computation (3) – non-deterministic

Nondeterministic :

in a given **instantaneous description** (p, w, β)

there may be <u>several</u> possible **steps**.

Any of these steps can be chosen in a computation.

https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA Computation (4) – pop operation

With the above definition in each step always
a single **symbol** (**top** of the **stack**) is popped,
replacing it with as many symbols as necessary.

As a result no step is defined when the stack is empty.

https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA Computation (5) – initial description

Computations of the pushdown automaton are
sequences of **steps**.

The computation starts in the **initial state** q_0
with the **initial stack symbol** Z on the stack,
and a string w on the **input tape**,
thus with **initial description** (q_0, w, Z).

https://en.wikipedia.org/wiki/Pushdown_automaton

Deterministic PDA Computation (6) – acceptance modes

There are two modes of **accepting**.

either accepts by **final state**,

which means after reading its input the automaton reaches an **accepting state** (in F)

uses the **internal memory** (**state**)

or it accepts by **empty stack** (ε),

which means after reading its input the automaton empties its stack.

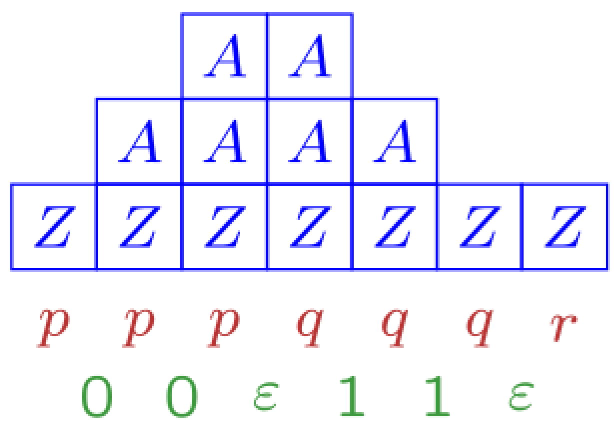
uses the **external memory** (**stack**).

https://en.wikipedia.org/wiki/Pushdown_automaton

Computation Example (1)

The following illustrates
how the above PDA computes
on different input strings.

The subscript M from the step symbol
$\vdash$ is here omitted.



$$
\begin{array}{llrrl}
(p, & 0011, & Z) & \vdash \\
(p, & 011, & AZ) & \vdash \\
(p, & 11, & AAZ) & \vdash \\
(q, & 11, & AAZ) & \vdash \\
(q, & 1, & AZ) & \vdash \\
(q, & \varepsilon, & Z) & \vdash \\
(r, & \varepsilon, & Z) &
\end{array}
$$

https://en.wikipedia.org/wiki/Pushdown_automaton

Computation Example (2)

input string = 0011.

There are various computations, depending on the moment the move from state p to state q is made.

Only one of these is accepting.

$(p , 0011 , Z) \vdash$ $\quad$ (p, ϵ, Z, q, Z),

$(q , 0011 , Z) \vdash$ $\quad$ (q, ϵ, Z, r, Z).

$(r , 0011 , Z)$

1. $(p, 0, Z, p, AZ)$
2. $(p, 0, A, p, AA)$
3. (p, ϵ, Z, q, Z)
4. (p, ϵ, A, q, A)
5. $(q, 1, A, q, \epsilon)$
6. (q, ϵ, Z, r, Z)

https://en.wikipedia.org/wiki/Pushdown_automaton

Computation Example (3)

The final state is accepting, but the input is not accepted this way as it has not been read.

(p , 0011 , Z) ⊢ (p, 0, Z, p, AZ)
(p , 011 , A Z) ⊢ (q, 1, A, q, ϵ)
(q , 011 , A Z)

No further steps possible.

1. (p, 0, Z, p, AZ)
2. (p, 0, A, p, AA)
3. (p, ϵ, Z, q, Z)
4. (p, ϵ, A, q, A)
5. (q, 1, A, q, ϵ)
6. (q, ϵ, Z, r, Z)

https://en.wikipedia.org/wiki/Pushdown_automaton

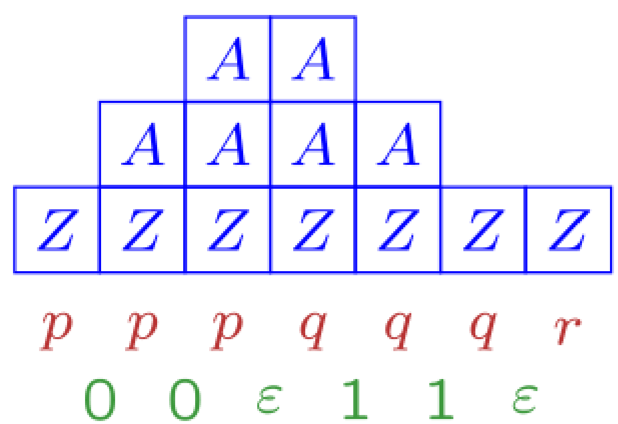
Computation Example (4)

(p , 0011 , Z) ⊢
(p , 011 , AZ) ⊢
(p , 11 , AAZ) ⊢
(q , 11 , AAZ) ⊢
(q , 1 , AZ) ⊢
(q , ϵ , Z) ⊢
(r , ϵ , Z)

(p, 0, A, p, AA)
(p, 0, A, p, AA)
(p, ϵ, A, q, A)
(q, 1, A, q, ϵ)
(q, 1, A, q, ϵ)
(q, ϵ, Z, r, Z)

1. (p, 0, Z, p, AZ)
2. (p, 0, A, p, AA)
3. (p, ϵ, Z, q, Z)
4. (p, ϵ, A, q, A)
5. (q, 1, A, q, ϵ)
6. (q, ϵ, Z, r, Z)

Accepting computation: ends in accepting state, while complete input has been read.



$$\begin{array}{llrrl}
(p, & 0011, & Z) & \vdash \\
(p, & 011, & AZ) & \vdash \\
(p, & 11, & AAZ) & \vdash \\
(q, & 11, & AAZ) & \vdash \\
(q, & 1, & AZ) & \vdash \\
(q, & \varepsilon, & Z) & \vdash \\
(r, & \varepsilon, & Z) &
\end{array}$$

https://en.wikipedia.org/wiki/Pushdown_automaton

Computation Example (5)

Input string = 00111. Again there are various computations. None of these is accepting.

(p , 00111 , Z) ⊢	(p, ϵ, Z, q, Z)
(q , 00111 , Z) ⊢	(q, ϵ, Z, r, Z)
(r , 00111 , Z)	

The final state is accepting,
but the input is not accepted
this way as it has not been read.

1. (p, 0, Z, p, AZ)
2. (p, 0, A, p, AA)
3. (p, ϵ, Z, q, Z)
4. (p, ϵ, A, q, A)
5. (q, 1, A, q, ϵ)
6. (q, ϵ, Z, r, Z)

https://en.wikipedia.org/wiki/Pushdown_automaton

Computation Example (6)

(p , 00111 , Z) ⊢
(p , 0111 , A Z) ⊢
(q , 0111 , A Z)

No further steps possible.

(p, 0, Z, p, AZ)
(p, ϵ, A, q, A)

1. (p, 0, Z, p, AZ)
2. (p, 0, A, p, AA)
3. (p, ϵ, Z, q, Z)
4. (p, ϵ, A, q, A)
5. (q, 1, A, q, ϵ)
6. (q, ϵ, Z, r, Z)

https://en.wikipedia.org/wiki/Pushdown_automaton

Computation Example (7)

(p , 00111 , Z) ⊢
(p , 0111 , A Z) ⊢
(p , 111 , A A Z) ⊢
(q , 111 , A A Z) ⊢
(q , 11 , A Z) ⊢
(q , 1 , Z) ⊢
(r , 1 , Z)

(p, 0, Z, p, AZ)
(p, 0, Z, p, AZ)
(p, ϵ, A, q, A)
(q, 1, A, q, ϵ)
(q, 1, A, q, ϵ)
(q, ϵ, Z, r, Z)

1. (p, 0, Z, p, AZ)
2. (p, 0, A, p, AA)
3. (p, ϵ, Z, q, Z)
4. (p, ϵ, A, q, A)
5. (q, 1, A, q, ϵ)
6. (q, ϵ, Z, r, Z)

The final state is accepting, but the input is not accepted this way as it has not been (completely) read.

https://en.wikipedia.org/wiki/Pushdown_automaton

PDA and Context Free Language (1)

Every **context-free grammar** can be transformed
into an equivalent **nondeterministic pushdown automaton**.

The derivation process of the grammar
is simulated in a **leftmost way**

Where the grammar rewrites a **nonterminal**,
the **PDA** takes the **topmost nonterminal** from its **stack**
and replaces it by the **right-hand part**
of a grammatical rule (expand).

Where the grammar generates a **terminal** symbol,
the **PDA** reads a symbol from **input**
when it is the **topmost symbol** on the **stack** (match).

In a sense the **stack** of the **PDA** contains
the unprocessed data of the grammar,
corresponding to a pre-order traversal of a derivation tree.

https://en.wikipedia.org/wiki/Pushdown_automaton

PDA and Context Free Language (1)

Every **context-free grammar** can be transformed
into an equivalent **nondeterministic pushdown automaton**.

The derivation process of the grammar
is simulated in a **leftmost way**

In a sense the **stack** of the **PDA** contains
the unprocessed data of the grammar,
corresponding to a pre-order traversal of a derivation tree.

https://en.wikipedia.org/wiki/Pushdown_automaton

PDA and Context Free Language (2)

The derivation process of the grammar
is simulated in a **leftmost way**

Where the grammar rewrites a **nonterminal**,
the **PDA** takes the **topmost nonterminal** from its **stack**
and replaces it by the **right-hand part**
of a grammatical rule (**expand**).

Where the grammar generates a **terminal** symbol,
the **PDA** reads a symbol from input
when it is the **topmost symbol** on the **stack** (**match**).

I

https://en.wikipedia.org/wiki/Pushdown_automaton

Computation Example (3)

Technically, given a context-free grammar, the PDA has a single state, 1, and its transition relation is constructed as follows.

(1 , ε , A , 1 , α) for each rule A → α (expand)

(1 , a , a , 1 , ε) for each terminal symbol a (match)

https://en.wikipedia.org/wiki/Pushdown_automaton

PDA and Context Free Language (2)

Technically, given a context-free grammar,
the PDA has a <u>single</u> **state**, 1,
and its **transition relation** is constructed as follows.

(1 , ε , A , 1 , α) for each rule $A \to \alpha$ (expand)

(1 , a , a , 1 , ε) for each terminal symbol a (match)

The PDA **accepts** by **empty stack**.

Its **initial stack symbol** is the grammar's **start symbol**.

https://en.wikipedia.org/wiki/Pushdown_automaton

References

[1] http://en.wikipedia.org/
[2]