

# Informatique Fondamentale IMA S8

## Cours 2 - Stack automata + Grammars

Laure Gonnord

<http://laure.gonnord.org/pro/teaching/>

[Laure.Gonnord@polytech-lille.fr](mailto:Laure.Gonnord@polytech-lille.fr)

Université Lille 1 - Polytech Lille

2012



Université  
Lille1  
Sciences et Technologies



POLYTECH<sup>®</sup>  
LILLE

## II - Stack automata and Grammars

1 Stack automata

2 Grammars

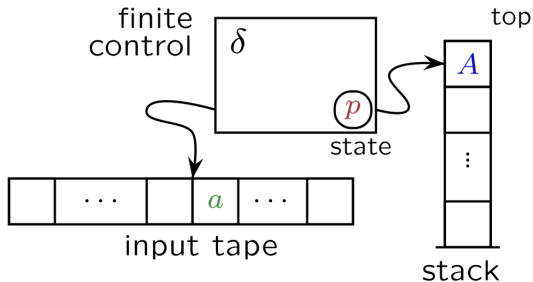
# What for ?

Express more than regular languages !

- ▶ Give a way for automata to “**count**”, to “have a memory”.

# Example

(source : Wikipedia)



# General definition

## Stack/P automata

- States ( $Q$ ), initial state ( $q_0$ ), finite states ( $F$ ).
  - Two alphabets (one for read :  $\Sigma$  one for stack :  $\Gamma$ )
  - $\gamma_0 \in \Gamma$  the end of stack character.
  - Transitions are finitely many.
  - A **stack** to write into.
  - Transitions use the stack.
- The transition function is :

$$Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma^*)$$

# How it works ? 1/3

## Configuration

A **configuration** is a triple  $(q, w, \alpha)$  where :

- $q$  is the current state
- $w \in \Sigma^*$  the part of the input tape which is not yet read
- $\alpha \in \Gamma^*$  the current stack word

Two different conditions for accepting words :

- “**accepting state**” : the read word leads to a configuration of the form  $(q, \varepsilon, \alpha)$  where  $q \in F$  (with any  $\alpha$ )
- “**empty stack**” : ...  $(q, \varepsilon, \gamma_0)$  (with any  $q$ , but  $\gamma_0$  is the “empty stack” character.

## How it works ? 2/3

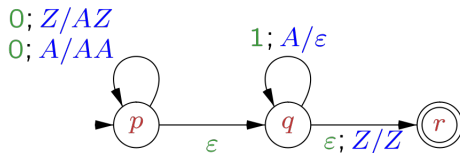
Given a configuration  $(q, w, \alpha)$ , the next one is computed by the help of the transition function  $Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow \mathcal{P}(Q \times \Sigma^*)$  :

- **enabled transitions** : those of the form  $(q, a, A) \rightarrow (q', A')$  where  $a$  is the next letter to read on the input tape (or  $\varepsilon$ ), and  $A$  the letter on top of the stack :  $w = aw'$  and  $\alpha = A\alpha'$ .
- Pick one enabled transition, and compute the next configuration  $(q', w', A'\alpha')$ .



# How it works ? 3/3

(source : wikipedia)



- ▶ Try to derive 0011 and 00111 !
- ▶ The PDA recognises  $\{0^n 1^n | n \in \mathbb{N}\}$  by accepting state.

# Important theoretical results on PDA

- **Important** Stack automata are non deterministic !
- The two acceptance criteria define the same class of languages

1 Stack automata

2 Grammars

# Goal

Problem : Express languages with the same expressivity as stack automata.

- ▶ use **grammars**

# General grammars

## Grammar rule

A **grammar** rule (production rule) is of the form

$$w \longrightarrow w'$$

where  $w$  and  $w'$  are words.

A grammar is a set of rules.

# Grammars

## Grammar

A **grammar** is composed of :

- A finite set  $N$  of non terminal symbols
- A finite set  $\Sigma$  of terminal symbols (disjoint from  $N$ )
- A finite set of production rules, each rule of the form  $w \rightarrow w'$  where  $w$  is a word on  $\Sigma \cup N$  with **at least** one letter of  $N$ .  $w'$  is a word on  $\Sigma \cup N$ .
- A start symbol  $S \in N$ .

# Grammars

**Example :**

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

is a grammar with  $N = \dots$  and  $\dots$

# Associated Language

## Derivation

$G$  a grammar defines the relation :

$$x \Rightarrow_G y \text{ iff } \exists u, v, p, q \ x = upv \text{ and } y = uqv \text{ and } (p \rightarrow q) \in P$$

- ▶ A grammar describes a **language** (the set of words on  $\Sigma$  that can be derived from the start symbol).



# Examples

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

The grammar defines the language  $\{a^n b^n, n \in \mathbb{N}\}$

$$S \rightarrow aBS c$$

$$S \rightarrow abc$$

$$Ba \rightarrow aB$$

$$Bb \rightarrow bb$$

The grammar defines the language  $\{a^n b^n c^n, n \in \mathbb{N}\}$

► Exercises

# Context-free grammars

## Context-free grammar

A **CF-grammar** is a grammar where all production rules are of the form  $N \rightarrow (\Sigma \cup N)^*$ .

## Example of CF-grammar

$$S \rightarrow S + S | S * S | a$$

The grammar defines a language of arithmetical expressions.

► Notion of **derivation tree**.

Draw a derivation tree of  $a^*a+a$ , of  $S+S$ !

# Relationship between stack automata and grammars

## General result

The context-free/algebraic languages are exactly the languages that are described by stack automata.

- ▶ The proof is not difficult.
- ▶ Exercises

## Some other results/definitions

- Regular languages are algebraic languages, but not the converse.
- There exists normal forms for algebraic grammars
- A grammar can be ambiguous.

# Summary

Stack Automata or **PushDown Automata** are :

- Acceptors for context-free languages. But some languages are **not context free**.
- Non deterministic.

▶ [http://en.wikipedia.org/wiki/Pushdown\\_automaton](http://en.wikipedia.org/wiki/Pushdown_automaton) for useful pointers