Formal Languages-Course 6.

Géraud Sénizergues

Bordeaux university

21/05/2020

Master computer-science MINF19, IEI, 2019/20

contents

Simple context-free grammars

Simple context-free grammars

Let $G = \langle A, N, R, E \rangle$ with

$$A = \{\mathrm{o}, \mathrm{e}, \mathrm{b}, \bar{\mathrm{b}}, \mathrm{a}, \mathrm{v}, \mathrm{s}\}, \quad N = \{E, C\},$$

and R consists of the rules :

$$E \longrightarrow \mathrm{o}EE \mid \mathrm{e} \mid \mathrm{v}, \quad C \longrightarrow \mathrm{b}C\mathrm{s}C\bar{\mathrm{b}} \mid \mathrm{v}\,\mathrm{a}\,E$$

Idea :
E : expressions with operator and atom e
C : command (or instruction)
$\mathrm{b}, \bar{\mathrm{b}}$: opening and closing brackets
s :separator
v :variable
a :affectation.

Simple context-free grammars

Let $G = \langle A, N, R, C \rangle$ with

$$E \longrightarrow \mathrm{o}EE \mid \mathrm{e} \mid \mathrm{v}, \quad C \longrightarrow \mathrm{b}C\mathrm{s}C\bar{\mathrm{b}} \mid \mathrm{v}\,\mathrm{a}\,E$$

For example :

$$\mathrm{bvaoeesvaove}\bar{\mathrm{b}}$$

is usally written as

$$(\mathrm{v} := \mathrm{oee}\ ;\ \mathrm{v} := \mathrm{ove})$$

if the atom is 1 and the operator is addition :

$$(\mathrm{v} := +\,1\,1\ ;\ \mathrm{v} := +\,\mathrm{v}\,1)$$

and the intended execution would assign the value 3 to the variable v.

Simple context-free grammars

Let us consider the above word :

$$w = \text{bvaoeesvaoveb}\bar{}$$

How can we find :
- a derivation-tree for w ?
- a derivation for w ? possibly a leftmost-derivation ? or a rightmost-derivation ?

This is called the parsing-problem for G and w.
The techniques developed towards this aim constitute the syntactic analysis.

Simple context-free grammars

Definition

A context-free grammar $G = \langle A, N, R, \sigma \rangle$ *is called simple iff*
1- every rule has the form $S \longrightarrow a \cdot m$ *for some*
$a \in A, m \in (A \cup N)^*$
2- $\forall S \in N, \forall a \in A, \forall m, m' \in (A \cup N)^*$,
$(S \to am \quad and \quad S \to am') \Rightarrow \quad m = m'$.

A language L is called simple deterministic iff there exists a simple context-free grammar $G = \langle A, N, R, \sigma \rangle$ such that

$$L = \mathrm{L}(G, \sigma).$$

Some authors reserve the term "simple" when condition 1 is replaced by the stronger condition :$R \subseteq N \times A \cdot N^*$. We use the above slightly more permissive definition in the sequel. However, these two variants define the same class of languages.

Simple context-free grammars

Example

Let $G_1 = \langle A, N, R, S \rangle$ *with*

$$A = \{a, b\}, \quad N = \{S\}, \quad S \longrightarrow aSS \mid b,$$

G_1 *is simple.*

Simple context-free grammars

Example

Let $G_2 = \langle A, N, R, C \rangle$ *with*

$$E \longrightarrow \mathrm{o}EE \mid \mathrm{e} \mid \mathrm{v}, \quad C \longrightarrow \mathrm{b}C\mathrm{s}C\bar{\mathrm{b}} \mid \mathrm{v}\,\mathrm{a}\,E$$

G_2 *is simple.*

Simple context-free grammars

Theorem

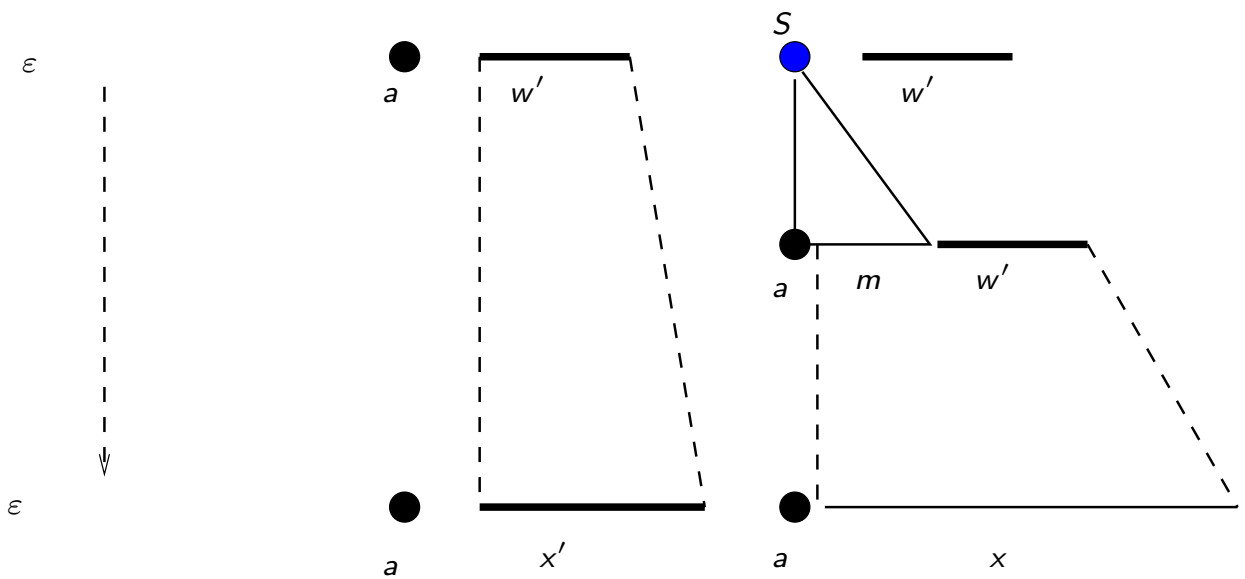
Let $G = \langle A, N, R \rangle$ *be a simple context-free grammar. Let* $w \in (A \cup N)^*, \quad x \in A^*$. *Then*

$$w \overset{*}{\underset{\ell}{\longrightarrow}} x$$

if and only if
(1) either $w = \varepsilon, x = \varepsilon$
(2) or $w = aw', x = ax', a \in A, w' \overset{*}{\underset{\ell}{\longrightarrow}} x'$
(3) or $w = Sw', x = ax', a \in A, S \in N, S \underset{\ell}{\longrightarrow}_R am$ *and* $mw' \overset{*}{\underset{\ell}{\longrightarrow}} x'$.

Simple context-free grammars

$w \overset{*}{\underset{\ell}{\longrightarrow}} x$ iff



Simple context-free grammars

Proof :
case 1 : $w = \varepsilon$.
Then $w = x = \varepsilon$.
case 2 : $w = aw'$ (for some $a \in A, w' \in (A \cup N)^*$)

$$aw' \underset{\ell}{\overset{*}{\longrightarrow}} x$$

By the "fundamental lemma for derivations", $x = ax'$ and $w' \underset{\ell}{\overset{*}{\longrightarrow}} x'$.

Simple context-free grammars

case 3 : $w = Sw'$ (for some $S \in N, w' \in (A \cup N)^*$).
Since the derivation is leftmost, it has the form :

$$Sw' \underset{\ell}{\overset{*}{\longrightarrow}} amw' \underset{\ell}{\overset{*}{\longrightarrow}} x$$

for some $(S, am) \in R$.
By case 1, applied to the (shorter) derivation $amw' \underset{\ell}{\overset{*}{\longrightarrow}} x$, we must have :

$$x = ax' \quad \text{and} \quad mw' \underset{\ell}{\overset{*}{\longrightarrow}} x'.$$

□
This theorem is the basis for constructing, from $S \in N, x \in A^*$
- either a leftmost derivation $S \underset{\ell}{\overset{*}{\longrightarrow}}_R x$
- or the answer NO, $\neg(S \underset{\ell}{\overset{*}{\longrightarrow}}_R x)$

Simple context-free grammars

Corollary

Every simple context-free grammar is non-ambiguous

Let $G = \langle A, N, R \rangle$ be a simple context-free grammar. We prove by induction on the integer n that : $\forall w \in (A \cup N)^*, \forall x \in A^*$ if

$$D_1 : w \overset{n_1}{\underset{\ell}{\longrightarrow}}_R x, \quad D_2 : w \overset{n_2}{\underset{\ell}{\longrightarrow}}_R x \tag{1}$$

and $n = n_1 + n_2 + |x|$, then $D_1 = D_2$.
Assume (1). We apply the theorem.
Case 1 :$w = \varepsilon = x$. Thus $D_1 = D_2$ are the trivial derivation of length 0.
Case 2 :$w = aw', x = ax', a \in A$
$D_1' : w' \overset{n_1}{\underset{\ell}{\longrightarrow}} x', \quad D_2' : w' \overset{n_2}{\underset{\ell}{\longrightarrow}} x'.$
By (IH) $D_1' = D_2'$. But $D_1 = aD_1', \quad D_2 = aD_2'$, hence $D_1 = D_2$.

Simple context-free grammars

Case 3 : $w = Sw', x = ax', a \in A, S \in N,$

$$D_1 : Sw' \underset{\ell}{\longrightarrow}_R am_1w' \overset{n_1-1}{\underset{\ell}{\longrightarrow}} ax',$$

$$D_2 : Sw' \underset{\ell}{\longrightarrow}_R am_2w' \overset{n_2-1}{\underset{\ell}{\longrightarrow}} ax',$$

Since the grammar is simple, $m_1 = m_2$. Since $n_1 + n_2 - 2 + |x| < n$, by (IH) : the derivations $am_1w' \overset{n_1-1}{\underset{\ell}{\longrightarrow}} ax', am_2w' \overset{n_2-1}{\underset{\ell}{\longrightarrow}} ax'$ are equal, showing that $D_1 = D_2$.

Simple context-free grammars

Let us call a language L prefix-free iff, $\forall u, v \in L, \quad u \preceq v \Rightarrow u = v$.

Corollary

Every simple deterministic language is prefix-free.

Proof: Let $G = \langle A, N, R, \sigma \rangle$ be a simple context-free grammar generating a language L.
Let $u, \beta, v \in A^* \mid u \cdot \beta = v, u \in L, v \in L$.
Applying iteratively the theorem, we see that there exists a word $m \in (A \cup N)^*$ such that

$$\sigma \xrightarrow{*}_R u \cdot m, \quad m \xrightarrow{*}_R \varepsilon, \quad m \xrightarrow{*}_R \beta.$$

The first derivation implies $m = \varepsilon$ and the second derivation shows $\beta = \varepsilon$. Hence $u = v$. $\square$

Syntactic analysis

Top-down analysis

Let $G_1 = \langle A, N, R, S \rangle$ be the grammar above. We name the rules :

$$\mathbf{r1} : S \longrightarrow aSS \quad \mathbf{r2} : S \longrightarrow b$$

Let us compute a leftmost-derivation for

$$w = aababbabb$$

Top-down analysis

We apply, iteratively, the theorem :

$$
\begin{array}{rcl}
S & \overset{\ell}{\longrightarrow} & aababbabb \\
 & \Leftrightarrow & (case3) \\
SS & \overset{\ell}{\longrightarrow} & ababbabb \\
 & \Leftrightarrow & (case3) \\
SSS & \overset{\ell}{\longrightarrow} & babbabb \\
 & \Leftrightarrow & (case3) \\
SS & \overset{\ell}{\longrightarrow} & abbabb \\
 & \Leftrightarrow & (case3) \\
SSS & \overset{\ell}{\longrightarrow} & bbabb
\end{array}
$$

Top-down analysis

$$
\begin{array}{rcl}
SSS & \overset{\ell}{\longrightarrow} & bbabb \\
 & \Leftrightarrow & (case3) \\
SS & \overset{\ell}{\longrightarrow} & babb \\
 & \Leftrightarrow & (case3) \\
S & \overset{\ell}{\longrightarrow} & abb \\
 & \Leftrightarrow & (case3) \\
SS & \overset{\ell}{\longrightarrow} & bb \\
 & \Leftrightarrow & (case3) \\
S & \overset{\ell}{\longrightarrow} & b \\
\varepsilon & \overset{\ell}{\longrightarrow} & \varepsilon \\
 & \Leftrightarrow & (case1)
\end{array}
$$

ACCEPT

Top-down analysis

This sequence of equivalences can be seen as a computation :

$Input - letter$	$stack$	$derivation - rule$
$-$	S	$-$
a	SS	$r1$
a	SSS	$r1$
b	SS	$r2$
a	SSS	$r1$
b	SS	$r2$
b	S	$r2$
a	SS	$r1$
b	S	$r2$
b	ε	$r2$

Top-down analysis

This top-down parsing of a word gives :
- the answer YES to the question whether $w \in L(G_1, S)$?
- it gives the leftmost-derivation $S \overset{*}{\underset{\ell}{\longrightarrow}}_R w$:

$$r1, r1, r2, r1, r2, r2, r1, r2, r2.$$

Top-down analysis

Input-word : aaba ;computation :

$Input - letter$	$stack$	$derivation - rule$
$-$	S	$-$
a	SS	$r1$
a	SSS	$r1$
b	SS	$r2$
a	SSS	$r1$
\$	error	

NB : Here symbol \$ denotes the end of the word.
Conclusion : $aaba \notin \mathrm{L}(G_1, S)$.

Top-down analysis

Input-word : aababbbb ;computation :

$Input - letter$	$stack$	$derivation - rule$
$-$	S	$-$
a	SS	$r1$
b	S	$r2$
a	SS	$r2$
b	S	$r1$
b	ε	
b	error	

Conclusion : $ababbb \notin \mathrm{L}(G_1, S)$.

Top-down analysis

Let $G_2 = \langle A, N, R, E \rangle$ be the grammar.
Let us name the rules :

$$\textbf{r1} : C \longrightarrow \mathrm{b}C\mathrm{s}C\bar{\mathrm{b}} \quad \textbf{r2} : C \longrightarrow \mathrm{v}\ \mathrm{a}\ E$$

$$\textbf{r3} : E \longrightarrow \mathrm{o}EE \quad \textbf{r4} : E \longrightarrow \mathrm{e} \quad \textbf{r5} : E \longrightarrow \mathrm{v},$$

Let us compute a leftmost-derivation for

$$w = \mathrm{bvaoeesvaove}\bar{\mathrm{b}}$$

Top-down analysis

We apply, iteratively, the theorem :

$$
\begin{array}{rcl}
C & \overset{\ell}{\longrightarrow} & \mathrm{b}\mathrm{v}\mathrm{a}\mathrm{o}\mathrm{e}\mathrm{e}\mathrm{s}\mathrm{v}\mathrm{a}\mathrm{o}\mathrm{v}\mathrm{e}\bar{\mathrm{b}} \\
 & \Leftrightarrow & (\mathit{case}3) \\
C\mathrm{s}C\bar{\mathrm{b}} & \overset{\ell}{\longrightarrow} & \mathrm{v}\mathrm{a}\mathrm{o}\mathrm{e}\mathrm{e}\mathrm{s}\mathrm{v}\mathrm{a}\mathrm{o}\mathrm{v}\mathrm{e}\bar{\mathrm{b}} \\
 & \Leftrightarrow & (\mathit{case}3) \\
\mathrm{a}E\mathrm{s}C\bar{\mathrm{b}} & \overset{\ell}{\longrightarrow} & \mathrm{a}\mathrm{o}\mathrm{e}\mathrm{e}\mathrm{s}\mathrm{v}\mathrm{a}\mathrm{o}\mathrm{v}\mathrm{e}\bar{\mathrm{b}} \\
 & \Leftrightarrow & (\mathit{case}2) \\
E\mathrm{s}C\bar{\mathrm{b}} & \overset{\ell}{\longrightarrow} & \mathrm{o}\mathrm{e}\mathrm{e}\mathrm{s}\mathrm{v}\mathrm{a}\mathrm{o}\mathrm{v}\mathrm{e}\bar{\mathrm{b}} \\
 & \Leftrightarrow & (\mathit{case}3) \\
EE\mathrm{s}C\bar{\mathrm{b}} & \overset{\ell}{\longrightarrow} & \mathrm{e}\mathrm{e}\mathrm{s}\mathrm{v}\mathrm{a}\mathrm{o}\mathrm{v}\mathrm{e}\bar{\mathrm{b}}
\end{array}
$$

Top-down analysis

$$
\begin{array}{rcl}
EEsC\bar{b} & \overset{\ell}{\longrightarrow} & eesvaove\bar{b} \\
 & \Leftrightarrow & (case3) \\
EsC\bar{b} & \overset{\ell}{\longrightarrow} & esvaove\bar{b} \\
 & \Leftrightarrow & (case3) \\
sC\bar{b} & \overset{\ell}{\longrightarrow} & svaove\bar{b} \\
 & \Leftrightarrow & (case2) \\
C\bar{b} & \overset{\ell}{\longrightarrow} & vaove\bar{b} \\
 & \Leftrightarrow & (case3) \\
aE\bar{b} & \overset{\ell}{\longrightarrow} & aove\bar{b} \\
 & \Leftrightarrow & (case2) \\
E\bar{b} & \overset{\ell}{\longrightarrow} & ove\bar{b}
\end{array}
$$

Top-down analysis

$$
\begin{aligned}
E\bar{\mathrm{b}} &\overset{\ell}{\longrightarrow} \mathrm{ove}\bar{\mathrm{b}} \\
&\Leftrightarrow \quad (\mathit{case}3) \\
EE\bar{\mathrm{b}} &\overset{\ell}{\longrightarrow} \mathrm{ve}\bar{\mathrm{b}} \\
&\Leftrightarrow \quad (\mathit{case}3) \\
E\bar{\mathrm{b}} &\overset{\ell}{\longrightarrow} \mathrm{e}\bar{\mathrm{b}} \\
&\Leftrightarrow \quad (\mathit{case}3) \\
\bar{\mathrm{b}} &\overset{\ell}{\longrightarrow} \bar{\mathrm{b}} \\
&\Leftrightarrow \quad (\mathit{case}2) \\
\varepsilon &\overset{\ell}{\longrightarrow} \varepsilon \\
&\Leftrightarrow \quad (\mathit{case}1)
\end{aligned}
$$

ACCEPT.

Top-down analysis

This sequence of equivalences can be seen as a computation :

Input – letter	*stack*	*derivation – rule*
–	C	–
b	$C\mathrm{s}C\bar{\mathrm{b}}$	$r1$
v	$\mathrm{a}E\mathrm{s}C\bar{\mathrm{b}}$	$r2$
a	$E\mathrm{s}C\bar{\mathrm{b}}$	–
o	$EE\mathrm{s}C\bar{\mathrm{b}}$	$r3$
e	$E\mathrm{s}C\bar{\mathrm{b}}$	$r4$
e	$\mathrm{s}C\bar{\mathrm{b}}$	$r4$
s	$C\bar{\mathrm{b}}$	–
v	$\mathrm{a}E\bar{\mathrm{b}}$	$r2$
a	$E\bar{\mathrm{b}}$	–

Top-down analysis

Input − letter	*stack*	*derivation − rule*
o	$EE\bar{\text{b}}$	*r*3
v	$E\bar{\text{b}}$	*r*5
e	$\bar{\text{b}}$	*r*4
$\bar{\text{b}}$	ε	−

Top-down analysis

This top-down parsing of a word gives :
- the answer YES to the question whether $w \in L(G, C)$?
- it gives the leftmost-derivation $C \overset{*}{\underset{\ell}{\longrightarrow}}_R w$:

$$r1, r2, r3, r4, r4, r2, r3, r5, r4.$$

Top-down analysis

We analyze $w =$ vaesvaove.

Input − letter	*stack*	*derivation − rule*
–	C	–
v	aE	$r2$
a	E	$r2$
e	ε	–
s	error	
v		
a		
o		
v		
e		

Thus vaesvaove $\notin$ L(G_2, C).

Top-down analysis

We analyze $w = \text{bvaesaove}\bar{\text{b}}$.

Input − letter	*stack*	*derivation − rule*
−	C	−
b	$C\text{s}C\bar{\text{b}}$	$r1$
v	$\text{a}E\text{s}C\bar{\text{b}}$	$r2$
a	$E\text{s}C\bar{\text{b}}$	−
e	$\text{s}C\bar{\text{b}}$	$r4$
s	$C\bar{\text{b}}$	
a	error	
o		
v		
e		
$\bar{\text{b}}$		

Thus $\text{bvaesaove}\bar{\text{b}} \notin \text{L}(G_2, C)$.

Top-down analysis

The word in column 2 can be considered as the memory-contents of the automaton.
This memory can be read and modified on the same end only (here the left-end) : it is called a stack. Each new line corresponds to a transition : the left-end of the memory is modified, depending on the top(i.e. leftmost symbol) of the stack and on the input-letter.
Such an automaton is called a pushdown-automaton.
Here a transition is completely determined by the top-symbol and the input-symbol : it is a deterministic pushdown automaton. The leftmost derivation can be considered as an output of the pda .

Top-down analysis :the pushdown-automaton

The pushdown automaton $\mathcal{A}_2$ that achieves the top-down analysis for G_2 has :
- memory $\in (A \cup N)^*$
- transitions : depending on $(top - symbol, input - letter)$

- $\mathrm{push}(\gamma)$: replaces the top-symbol of the stack by the word γ
- pop : removes the top-symbol of the stack ($= \mathrm{push}(\varepsilon)$)
- accept : accepts the input-word
- error : rejects the input-word

Top-down analysis : the pushdown-automaton

The pushdown automaton $\mathcal{A}_2$:

$Top - symbol$	$input - letter$	$transition$
E	o	$\text{push}(EE)$
E	e	$\text{pop} = \text{push}(\varepsilon)$
E	v	pop
C	b	$\text{push}(C\text{s}C\bar{\text{b}})$
C	v	$\text{push}(\text{a}E)$
ε	\$	accept
E	$y \in (A \cup \{\$\}) \setminus \{\text{o}, \text{e}, \text{v}\}$	error
C	$y \in (A \cup \{\$\}) \setminus \{\text{b}, \text{v}\}$	error
$x \in A$	x	pop
$x \in A$	$y \in (A \cup \{\$\}) \setminus \{x\}$	error

Bottom-up analysis

Let $G_1 = \langle A, N, R, S \rangle$ be the grammar above, with rules :

$$\mathbf{r1} : S \longrightarrow aSS \quad \mathbf{r2} : S \longrightarrow b$$

Let us compute a rightmost-derivation for

$$w = aababbabb$$

Bottom-up analysis

$$\mathbf{r1} : S \longrightarrow aSS \quad \mathbf{r2} : S \longrightarrow b$$

$$w = aababbabb$$

Stack	*Input* − *letter*(or ε)	*derivation* − *rule*
−	*a*	
a	*a*	
aa	*b*	
aab	ε	*r*2
aaS	*a*	
aaSa	*b*	
aaSab	ε	*r*2
aaSaS	*b*	
aaSaSb	ε	*r*2
aaSaSS	ε	*r*1

Bottom-up analysis

$$\mathbf{r1} : S \longrightarrow aSS \quad \mathbf{r2} : S \longrightarrow b$$

$Stack$	$Input - letter(\text{ or } \varepsilon)$	$derivation - rule$
$aaSS$	ε	$r1$
aS	a	
aSa	b	
$aSab$	ε	$r2$
$aSaS$	b	
$aSaSb$	ε	$r2$
$aSaSS$	ε	$r1$
aSS	ε	$r1$
S		

Bottom-up analysis

This bottom-up parsing of a word gives :

- the answer YES to the question whether $w \in \mathrm{L}(G_1, S)$?
- it gives the reversal of a rightmost-derivation $S \overset{*}{\underset{R,r}{\longrightarrow}} w$:

$$r2, r2, r2, r1, r1, r2, , r2, r1, r1.$$

Pushdown-automata

Pushdown-automaton

Definition

A pushdown automaton is a 6-tuple

$$\mathcal{A} = \langle A, Z, Q, z_1, q_1, \delta \rangle$$

where
- *A is a finite alphabet, called the input-alphabet*
- *Z is a finite alphabet, called the pushdown-alphabet*
- *Q is a finite set, called the set of states*
- *$z_1 \in Z$ is the starting symbol*
- *$q_1 \in q$ is the starting state*
- *δ is a finite subset of $(A \cup \{\varepsilon\}) \times Q \times Z \times Q \times Z^*$ called the set of transitions.*

Pushdown-automaton

A configuration of the automaton $\mathcal{A}$ is a triple $(w, q, h) \in A* \times Q \times Z^*$. The movement relation on configurations is defined by :

$$(w, q, h) \vdash\!\!- \ _{\mathcal{A}}(w', q', h')$$

iff $\exists x \in A \cup \{\varepsilon\}, g \in Z^*, z \in Z, (x, q, z, q', u') \in \delta$, such that

$$(w, q, h) = (xw', q, gz), \quad h' = gu'.$$

Notation :
we note $qz \xrightarrow{x} q'u'$ a transition (x, q, z, q', u').
we note $qh \xrightarrow{w} q'h$ for $(w, q, h) \vdash\!\!- \ ^*_{\mathcal{A}}(\varepsilon, q', h')$

Pushdown-automaton

A movement is thus of the form :

$$(xw', q, gz) \vdash_{\mathcal{A}} (w', q', gu')$$

where

$$(x, q, z, q', u') \in \delta.$$

NB : Here, each movement modifies the right-end of the stack ; in the top-down analyzers of G_1, G_2 each movement was modifying the left-end of the stack ; every top-down analyzer can be transformed into a pda by reversing the stack-contents.

Pushdown-automaton

$$\mathrm{L}_N(\mathcal{A}) = \{w \in A^* \mid \exists q \in Q, (w, q_1, z_1) \vdash\!\!\!-\!\!\!- \overset{*}{}_{\mathcal{A}}(\varepsilon, q, \varepsilon)\}$$

or, equivalently

$$\mathrm{L}_N(\mathcal{A}) = \{w \in A^* \mid \exists q \in Q, (q_1, z_1) \overset{w}{\longrightarrow}_{\mathcal{A}} (q, \varepsilon)\}$$

Pushdown-automaton

The pda is called deterministic iff for every $q \in Q, z \in Z$:
- either $qz \xrightarrow{\varepsilon} q'u'$ for some $q' \in Q, u' \in Z^*$ and this is the only transition that starts from qz
- or there is no transition of the form $qz \xrightarrow{\varepsilon} q'u'$, and for every $a \in A$, if $qz \xrightarrow{a} q'u'$ and $qz \xrightarrow{a} r'v'$, then $q' = r'$ and $u' = v'$.

In words : locally, the automaton has no choice of transition.

Pushdown-automaton

An example : top-down analyzer for grammar G_1.

$$\mathcal{A}_1 = \langle A, Z, Q, z_1, q_1, \delta \rangle$$

where

$$A = \{a, b\}, \quad Z = \{S\}, \quad z_1 = S, \quad q_1 = q$$

and δ consists of the rules :

$$(q, S) \xrightarrow{a} (q, SS), \quad (q, S) \xrightarrow{b} (q, \varepsilon).$$

Pushdown-automaton

$\mathcal{A}_1$: top-down analyzer for grammar G_1. Computation on $w = aababbabb$.

$$(q, S) \xrightarrow{a} (q, SS) \xrightarrow{a} (q, SSS) \xrightarrow{b} (q, SS) \xrightarrow{a} (q, SSS) \xrightarrow{b} (q, SS)$$

$$\xrightarrow{b} (q, S) \xrightarrow{a} (q, SS) \xrightarrow{b} (q, S) \xrightarrow{b} (q, \varepsilon).$$

The automaton $\mathcal{A}_1$ accepts $aababbabb$.

Pushdown-automaton

An example : top-down analyzer for grammar G_2.

$$\mathcal{A}_2 = \langle A, Z, Q, z_1, q_1, \delta \rangle$$

where

$$A = \{\mathrm{o}, \mathrm{e}, \mathrm{b}, \bar{\mathrm{b}}, \mathrm{a}, \mathrm{v}, \mathrm{s}\}, \quad Z = \{E, C\}, Q = \{q\}, \quad z_1 = C, \quad q_1 = q$$

and δ consists of the rules :

$$(q, C) \xrightarrow{b} (q, \bar{\mathrm{b}} C \mathrm{s} C), \quad (q, C) \xrightarrow{\mathrm{v}} (q, E\mathrm{a}),$$
$$(q, E) \xrightarrow{\mathrm{o}} (q, EE), \quad (q, E) \xrightarrow{\mathrm{e}} (q, \varepsilon), \quad (q, E) \xrightarrow{\mathrm{v}} (q, \varepsilon).$$

Pushdown-automaton

$\mathcal{A}_2$: top-down analyzer for grammar G_2.
Computation on $w = \mathrm{bvaoeesvaove}\bar{\mathrm{b}}$.

$$(q, C) \xrightarrow{\mathrm{b}} (q, \bar{\mathrm{b}}C\mathrm{s}C) \xrightarrow{\mathrm{v}} (q, \bar{\mathrm{b}}C\mathrm{s}E\mathrm{a}) \xrightarrow{\mathrm{a}} (q, \bar{\mathrm{b}}C\mathrm{s}E) \xrightarrow{\mathrm{o}} (q, \bar{\mathrm{b}}C\mathrm{s}EE)$$

$$\xrightarrow{\mathrm{e}} (q, \bar{\mathrm{b}}C\mathrm{s}E) \xrightarrow{\mathrm{e}} (q, \bar{\mathrm{b}}C\mathrm{s}) \xrightarrow{\mathrm{s}} (q, \bar{\mathrm{b}}C) \xrightarrow{\mathrm{v}} (q, \bar{\mathrm{b}}E\mathrm{a})$$

$$\xrightarrow{\mathrm{a}} (q, \bar{\mathrm{b}}E) \xrightarrow{\mathrm{o}} (q, \bar{\mathrm{b}}EE) \xrightarrow{\mathrm{v}} (q, \bar{\mathrm{b}}E) \xrightarrow{\mathrm{e}} (q, \bar{\mathrm{b}}) \xrightarrow{\bar{\mathrm{b}}} (q, \varepsilon).$$

The automaton $\mathcal{A}_1$ accepts $\mathrm{bvaoeesvaove}\bar{\mathrm{b}}$.

Pushdown-automaton

Another example : a bottom-up analyzer for grammar G_1.
Extended pushdown automaton

$$\mathcal{B}_1 = \langle A, Z, Q, z_1, q_1, \delta \rangle$$

where $A = \{a, b\}$, $Q = \{q\}$, $Z = \{S\}$, $z_1 = \varepsilon$, $q_1 = q$ and δ consists of the (extended) rules :

$$(q, aSS) \xrightarrow{\varepsilon} (q, S), \quad (q, b) \xrightarrow{\varepsilon} (q, S),$$

$$(q, \varepsilon) \xrightarrow{a} (q, a), \quad (q, \varepsilon) \xrightarrow{b} (q, b).$$

Can be turned into a deterministic pushdown automaton.

Pushdown-automaton

$\mathcal{B}_1$: bottom-up analyzer for grammar G_1.
Computation on $w = aababbabb$.

$$(q, \varepsilon) \xrightarrow{a} (q, a) \xrightarrow{a} (q, aa) \xrightarrow{b} (q, aab) \xrightarrow{\varepsilon} (q, aaS) \xrightarrow{a} (q, aaSa)$$

$$\xrightarrow{b} (q, aaSab) \xrightarrow{\varepsilon} (q, aaSaS) \xrightarrow{b} (q, aaSaSb) \xrightarrow{\varepsilon} (q, aaSaSS)$$

$$\xrightarrow{\varepsilon} (q, aaSS) \xrightarrow{\varepsilon} (q, aS) \xrightarrow{a} (q, aSa) \xrightarrow{b} (q, aSab)$$

$$\xrightarrow{\varepsilon} (q, aSaS) \xrightarrow{b} (q, aSaSb) \xrightarrow{\varepsilon} (q, aSaSS) \xrightarrow{\varepsilon} (q, aSS) \xrightarrow{\varepsilon} (q, S)$$

The (extended) automaton $\mathcal{A}_1$ accepts the word $aababbabb$.

Pushdown-automaton

Let $L = \{u \in \{a,b\}^* c \mid |u|_a = |u|_b\}$.
Let $\mathcal{A} = \langle A, Z, Q, z_1, q_1, \delta \rangle$ where
$A = \{a,b\}$, $Z = \{\Omega, C\}$, $Q = \{q, \bar{q}\}$, $z_1 = \Omega$, $q_1 = q$
and δ consists of the rules :

$$(q,\Omega) \xrightarrow{a} (q,\Omega C), \quad (q,\Omega) \xrightarrow{b} (\bar{q},\Omega C),$$

$$(\bar{q},\Omega) \xrightarrow{a} (q,\Omega C), \quad (\bar{q},\Omega) \xrightarrow{b} (\bar{q},\Omega C),$$

$$(q,C) \xrightarrow{a} (q,CC), \quad (q,C) \xrightarrow{b} (q,\varepsilon),$$

$$(\bar{q},C) \xrightarrow{a} (q,\varepsilon), \quad (\bar{q},C) \xrightarrow{b} (\bar{q},CC),$$

$$(\bar{q},\Omega) \xrightarrow{c} (\bar{q},\varepsilon), \quad (q,\Omega) \xrightarrow{c} (q,\varepsilon).$$

Pushdown-automaton

A computation :

$$w = aabbbac.$$

$$(q, \Omega) \xrightarrow{a} (q, \Omega C) \xrightarrow{a} (q, \Omega CC) \xrightarrow{b} (q, \Omega C) \xrightarrow{b} (q, \Omega)$$

$$\xrightarrow{b} (\bar{q}, \Omega C) \xrightarrow{a} (\bar{q}, \Omega) \xrightarrow{c} (\bar{q}, \varepsilon).$$

Pushdown-automata

Theorem

A language $L \subseteq A^*$ is context-free if and only of there exists a pushdown automaton $\mathcal{A}$ such that $L = \mathrm{L}_N(\mathcal{A})$.

Theorem

Let G be a simple grammar. Then one can construct a deterministic pushdown automaton $\mathcal{A}$ such that $\mathrm{L}_N(\mathcal{A}) = L$. Moreover the automaton $\mathcal{A}$ can be chosen with one state only and without ϵ-transitions.

NB : unlike for finite automaton, there is no general determinization theorem.
Some context-free languages cannot be recognized by any deterministic pda .

Pushdown-automata

Corollary

If L is a simple context-free language, it is prefix-free i.e. $\forall u, v \in L$ if u is a prefix of v then $u = v$.

second proof

Let $\mathcal{A} = \langle A, Z, Q, z_1, q_1, \delta \rangle$ be a deterministic pda such that $L = \mathrm{L}_N(\mathcal{A})$ and let $u, v \in L$ with $u \preceq v$. Thus $\exists \beta \in A^*, u\beta = v$. Since both words $u, u\beta$ are recognized by $\mathcal{A}$, there are states $r, r' \in Q$ such that

$$(q_1, z_1) \xrightarrow{u}_{\mathcal{A}} (r, \varepsilon),\ (q_1, z_1) \xrightarrow{u\beta}_{\mathcal{A}} (r', \varepsilon).$$

But the automaton is deterministic, hence the second computation has the form

$$(q_1, z_1) \xrightarrow{u}_{\mathcal{A}} (r, \varepsilon) \xrightarrow{\beta}_{\mathcal{A}} (r', \varepsilon).$$

Pushdown-automata

$$(q_1, z_1) \xrightarrow{u}_{\mathcal{A}} (r, \varepsilon) \xrightarrow{\beta}_{\mathcal{A}} (r', \varepsilon).$$

The only possible second part of computation is a trivial one (i.e. of length 0) :

$$\beta = \varepsilon.$$

Hence

$$u = v$$

□