

# Formal Languages-Course 4.

Géraud Sénizergues

Bordeaux university

14/05/2020

Master computer-science MINF19, IEI, 2019/20

# contents

- 1 Closure properties of regular languages
- 2 Decision problems for regular languages
- 3 Non-regular languages
- 4 Context-free grammars
  - Word rewriting
  - Context-free grammars
  - Syntax-trees
  - Derivations

# Closure properties of regular languages

## regular operations

## Theorem

Let  $\Sigma$  be a finite alphabet. The set of regular languages over  $\Sigma^*$  is closed under the operations : *union, product, star, cross*.

**Proof:** Let  $L, L'$  be regular languages over  $\Sigma^*$ . Let  $e, e' \in \text{RE}(\Sigma)$  such that  $\nu(e) = L, \nu(e') = L'$ . Then  $f := e \cup e'$  is also a regular expression. Since  $\nu(f) = L \cup L'$ ,  $L \cup L'$  is regular too.  $\square$   
For product, star and cross similar arguments apply.

## boolean operations

## Theorem

Let  $\Sigma$  be a finite alphabet. The set of regular languages over  $\Sigma^*$  is closed under the operations : *union, intersection, complement, set-difference*.

By Kleene's theorem we can use, as well, the notion of recognizable language.

**Closure under complement :**

Let  $L$  be a regular language over  $\Sigma^*$ . By Kleene's theorem  $L$  is recognized by some complete dfa  $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ .

Let  $\mathcal{A}' = \langle Q, \Sigma, \delta, q_0, Q \setminus F \rangle$ . For every  $w \in \Sigma^*$  :

$w \in CL \Leftrightarrow \text{not } w \in L \Leftrightarrow \text{not } \delta^*(q_0, w) \in F \Leftrightarrow \delta^*(q_0, w) \in Q \setminus F \Leftrightarrow w \in L(\mathcal{A}')$ . Hence  $CL$  is recognizable.  $\square$

## boolean operations

**proof of the theorem :**

Let  $L_1, L_2$  be regular languages over  $\Sigma^*$ .

We have seen that  $CL_1$  is regular.

$$L_1 \cap L_2 = C(CL_1 \cup CL_2).$$

Hence closure under complement and union shows that  $L_1 \cap L_2$  is regular.

$$L_1 \setminus L_2 = L_1 \cap CL_2.$$

Hence, closure under intersection and complement shows that  $L_1 \setminus L_2$  is regular.  $\square$

# homomorphisms

Let  $\Sigma, \Delta$  be two finite alphabets.

A monoid-homomorphism :  $h : \Sigma^* \rightarrow \Delta^*$  is a map which is **compatible** with the product and the neutral element i.e. :

$$\forall u, v \in \Sigma^*, h(u \cdot v) = h(u) \cdot h(v), \quad h(\varepsilon) = \varepsilon.$$

Since every word is a finite product of letters, a monoid-homomorphism is completely defined by the images of the letters :  $h(x)$  for  $x \in \Sigma$ . Then :

$$h(x_0 \cdots x_i \cdots x_{\ell-1}) = h(x_0) \cdots h(x_i) \cdots h(x_{\ell-1})$$

## homomorphisms : examples

Let  $\Sigma = \{a, b\}$ ,  $\Delta = \{a, b, c, d\}$ .

Let  $h : \Sigma^* \rightarrow \Delta^*$  homomorphism such that

$$h(a) = abc, \quad h(b) = dc$$

Then

$$h(abb) = h(a) \cdot h(b) \cdot h(b) = abc d c d c$$

$$h(bba) = h(b) \cdot h(b) \cdot h(a) = d c d c a b c$$



## substitutions

Let  $\Sigma, \Delta$  be two finite alphabets.

A substitution is a map  $\mathcal{P}(\Sigma^*) \rightarrow \mathcal{P}(\Sigma^*)$  which is **compatible** with the product and the (arbitrary) union, and with the two neutral elements  $\{\varepsilon\}$  (neutral element wrt  $\cdot$ ),  $\emptyset$  (neutral element wrt  $\cup$ ) i.e.  $:\forall L, M \in \mathcal{P}(\Sigma^*), (L_i)_{i \in I}$  family of languages over  $\Sigma$  :

$$\sigma(L \cdot M) = \sigma(L) \cdot \sigma(M), \quad \sigma\left(\bigcup_{i \in I} L_i\right) = \bigcup_{i \in I} \sigma(L_i)$$

$$\sigma(\{\varepsilon\}) = \{\varepsilon\}, \quad \sigma(\emptyset) = \emptyset.$$

The substitution is called **regular** if every  $\sigma(x)$  for  $x \in \Sigma$  is **regular**.

## substitutions : examples

## Example

Let  $\Sigma = \{a, b\}$ ,  $\Delta = \{a, b, c, d\}$ ,  
 $\sigma : \mathcal{P}(\Sigma^*) \rightarrow \mathcal{P}(\Delta^*)$  substitution such that

$$\sigma(a) = \{a, ac\}, \quad \sigma(b) = \{dd, dc\}$$

Then

$$\begin{aligned} \sigma(\{abb\}) &= \sigma(a) \cdot \sigma(b) \cdot \sigma(b) \\ &= \{a, ac\} \cdot \{dd, dc\} \cdot \{dd, dc\} \\ &= \{add, adc, acdd, acdc\} \cdot \{dd, dc\} \\ &= \{adddd, adcdd, acdddc, acdcdd, adddc, adcdc, \\ &\quad acdddc, acdcdc\} \end{aligned}$$

## substitutions : examples

## Example

$$\begin{aligned}\sigma(\{a, abb\}) &= \sigma(\{a\}) \cup \sigma(\{abb\}) \\ &= \{a, ac\} \cup \{adddd, adcdd, acdddd, acdcdd, adddc, \\ &\quad adcdc, acdddc, acdcdc\} \\ &= \{a, ac, adddd, adcdd, acdddd, acdcdd, adddc, \\ &\quad adcdc, acdddc, acdcdc\}.\end{aligned}$$

## substitutions : examples

## Example

Let  $\Sigma = \{a, b\}$ ,  $\Delta = \{a, b, c, d\}$ .

Let  $\sigma : \mathcal{P}(\Sigma^*) \rightarrow \mathcal{P}(\Delta^*)$  substitution such that

$$\sigma(a) = \{a^{n^2} \mid n \geq 1\}, \quad \sigma(b) = \{bbb\}$$

Then

$$\begin{aligned} \sigma(\{ab\}) &= \sigma(a) \cdot \sigma(b) \\ &= \{a^{n^2} bbb \mid n \geq 1\} \end{aligned}$$

$$\begin{aligned} \sigma(\{aba\}) &= \sigma(a) \cdot \sigma(b) \cdot \sigma(a) \\ &= \{a^{n^2} bbb a^{m^2} \mid n \geq 1, m \geq 1\} \end{aligned}$$

## substitutions : examples

## Example

$$\sigma(\{aba\}) = \{a^{n^2} bbba^{m^2} \mid n \geq 1, m \geq 1\}.$$

Hence :

$$aaaabbba \in \sigma(\{aba\}), \quad abbbaaaaaaaaaa \in \sigma(\{aba\}).$$

## regular substitutions

## Theorem

Let  $\Sigma, \Delta$  be finite alphabets.

1- if  $L \in \text{REG}(\Sigma^*)$  and  $h : \Sigma^* \rightarrow \Delta^*$  is a monoid-homomorphism then  $h(L) \in \text{REG}(\Delta^*)$ .

2- if  $L \in \text{REG}(\Sigma^*)$  and  $\sigma : \Sigma^* \rightarrow \Delta^*$  is a regular substitution, then  $\sigma(L) \in \text{REG}(\Delta^*)$ .

In short : the family of regular languages is closed under the operations : **homorphism, regular substitution**.

## regular substitutions

**Proof**

Suppose  $L = \nu(e)$  for some  $e \in \text{RE}(\Sigma)$ .

Let  $\sigma(x) = \nu(e_x)$  for  $e_x \in \text{RE}(\Delta)$ . We recall the auxiliary alphabet

$$\text{AUX} = \{0, \oplus, \otimes, \star, \langle, \rangle\}$$

Let us consider the new regular expression :

$$f := h(e)$$

where  $h : (\Sigma \cup \text{AUX})^* \rightarrow (\Delta \cup \text{AUX})^*$  is the monoid-homomorphism that fixes all the letters in  $\text{AUX}$  and maps every  $x \in \Sigma$  onto

$$h(x) = e_x.$$

## regular substitutions

One can check that, for every regular expression  $e \in \text{RE}(\Sigma)$

$$\nu(h(e)) = \sigma(\nu(e)).$$

(by induction on the size of  $e$ ). Hence

$$\nu(f) = \nu(h(e)) = \sigma(\nu(e)) = \sigma(L).$$

Hence  $\sigma(L)$  is regular.  $\square$



## regular substitutions :example

## Example

Let  $\Sigma = \{a, b\}$ ,  $\Delta = \{a, b, c, d\}$ .

Let  $\sigma : \mathcal{P}(\Sigma^*) \rightarrow \mathcal{P}(\Delta^*)$  substitution such that

$$\sigma(a) = (bc)^*, \quad \sigma(b) = (ac)^* \cup bbb$$

Let  $L = \nu(e)$  where

$$e = ab^*a$$

.

Then

$$f = (bc)^* \cdot ((ac)^* \cup bbb)^* \cdot (bc)^*.$$

$$\sigma(L) = \nu(f).$$

# Reversal

## Proposition

Let  $\Sigma$  be a finite alphabets.

If  $L \in \text{REG}(\Sigma^*)$  then  $L^R \in \text{REG}(\Sigma^*)$ .

### Proof :

We define inductively the map  $e \mapsto e^t : \text{RE}(\Sigma) \rightarrow \text{RE}(\Sigma)$  by :

$\forall x \in \Sigma, \forall e, e' \in \text{RE}(\Sigma),$

$$0^t = 0 \quad x^t = x$$

$$\langle e \oplus e' \rangle^t = \langle e^t \oplus e'^t \rangle$$

$$\langle e \otimes e' \rangle^t = \langle e'^t \otimes e^t \rangle$$

$$\langle e \star \rangle^t = \langle e^t \star \rangle.$$

$$\langle e \rangle^t = \langle e^t \rangle.$$

# Reversal

We can prove by induction that :  $\forall e \in \text{RE}(\Sigma)$

$$\nu(e^t) = (\nu(e))^R.$$

Let  $L \in \text{REG}(\Sigma^*)$ . There exists some  $e \in \text{RE}(\Sigma)$  such that  $L = \nu(e)$ . By the above equality :

$$L^R = \nu(e)^R = \nu(e^t)$$

hence  $L^R$  is regular.

□

# Decision problems for regular languages

## membership and emptiness

The following problems are **decidable** :

**INPUT** : a dfa  $\mathcal{A}$  and a word  $w$ .

**QUESTION** :  $w \in L(\mathcal{A})$ ?

(in time  $O(|w|)$ ).

Just compute  $\delta^*(q_0, w)$  by the algorithm we mentioned earlier.

**INPUT** : a regular expression  $e$  and a word  $w$ .

**QUESTION** :  $w \in L_e$ ?

(in time  $O(|w| \cdot |e|)$ ).

Compute (by Glushkov algorithm) a nfa  $\mathcal{A}$  recognizing  $L_e$ . Then **simulate** the determinized automaton  $\mathcal{D}$  over input  $w$ .

**INPUT** : a nfa  $\mathcal{A}$ .

**QUESTION** :  $L(\mathcal{A}) = \emptyset$ ?

(in time  $O(\|\mathcal{A}\|)$ ).

Compute the set of accessible states  $Q_1$  and test whether  $Q_1 \cap F = \emptyset$ .

## inclusion

**INPUT** : two dfa  $\mathcal{A}, \mathcal{B}$ .

**QUESTION** :  $L(\mathcal{A}) \subseteq L(\mathcal{B})$ ?

(in time  $O(\|\mathcal{A}\| \cdot \|\mathcal{B}\|)$ ).

Compute a dfa  $\mathcal{C}$  recognizing the difference

$L(\mathcal{A}) \setminus L(\mathcal{B}) = L(\mathcal{A}) \cap \text{CL}(\mathcal{B})$ .

Test  $\mathcal{C}$  for emptiness.

**INPUT** : two regular expressions  $e, f$ .

**QUESTION** :  $L_e \subseteq L_f$ ?

(in time  $2^{O(|e| \cdot |f|)}$ ).

Translate, by Glushkov algorithm  $e, f$  into nfa  $\mathcal{A}, \mathcal{B}$ . Determinize the two nfa into dfa  $\mathcal{A}', \mathcal{B}'$ . Apply previous algorithm to these dfa .

# Non-regular languages

## An example

## Proposition

The language  $L = \{a^n b^n \mid n \geq 1\}$  is *not* regular.

*Proof :*

Suppose that some nfa  $\mathcal{A}$  is such that

$$L = L(\mathcal{A}) \tag{1}$$

Let  $N = \text{Card}(Q)$ . Let us note

$$u = (q_0, x_0, q_1)(q_1, x_1, q_2) \cdots (q_{\ell-1}, x_{2N-1}, q_{2N})$$

be a computation reading  $\text{tr}(u) = w = a^N b^N = x_0 x_1 \cdots x_{2N-1}$ .

Since  $\text{Card}(\{q_0, q_1, \dots, q_N\}) = N + 1 > \text{Card}(Q)$  there must exist two different integers  $i < j$  such that

$$0 \leq i < j \leq N \text{ and } q_i = q_j.$$



## An example

Let  $\alpha = a^i$ ,  $v = a^{j-i}$ ,  $\beta = a^{N-j}b^N$ . We have

$$q_0 \xrightarrow{\alpha}_{\mathcal{A}} q_i \xrightarrow{v}_{\mathcal{A}} q_j \xrightarrow{\beta}_{\mathcal{A}} q_{2N}.$$

As well, since  $q_i \xrightarrow{v}_{\mathcal{A}} q_j = q_i$  we get the computation

$$q_0 \xrightarrow{\alpha}_{\mathcal{A}} q_i \xrightarrow{v}_{\mathcal{A}} q_j \xrightarrow{v}_{\mathcal{A}} q_j \xrightarrow{\beta}_{\mathcal{A}} q_{2N}.$$

The trace of this new computation is :

$$w' = \alpha v^2 \beta = a^{N+j-i} b^N,$$

showing that  $L(\mathcal{A}) \not\subseteq L$ , contradicting (1). We have proved, ad absurdum, that for every nfa  $\mathcal{A}$

$$L \neq L(\mathcal{A}).$$

NB : this way of showing that a language  $L$  is not regular can be adapted to numerous languages.

□

# Context-free grammars

# Rewriting

Let  $A$  be an alphabet (i.e. a set) and

$$R \subseteq A^* \times A^*.$$

We define binary relations over  $A^*$

$$\longrightarrow_R, \xrightarrow{n}_R, \xrightarrow{*}_R, \xrightarrow{+}_R$$

where  $n$  is an integer, by  $u \longrightarrow_R v$  iff  $\exists \alpha, \beta \in A^*, \exists (l, r) \in R$

$$u = \alpha l \beta, v = \alpha r \beta.$$

$u \xrightarrow{n}_R v$  iff  $\exists u_0, u_1, \dots, u_n \in A^*$  such that

$$u = u_0 \longrightarrow_R u_1 \longrightarrow_R \dots \longrightarrow_R u_i \longrightarrow_R u_{i+1} \dots \longrightarrow_R u_n = v$$

# Rewriting

$$\longrightarrow_R^* = \bigcup_{n \geq 0} \longrightarrow_R^n, \quad \longrightarrow_R^+ = \bigcup_{n \geq 1} \longrightarrow_R^n .$$

A sequence  $(u_0, u_1, \dots, u_n)$  such that

$u = u_0 \longrightarrow_R u_1 \longrightarrow_R \dots \longrightarrow_R u_i \longrightarrow_R u_{i+1} \dots \longrightarrow_R u_n$  is called a *derivation* (modulo  $R$ ). Its *length* is  $n$ .

## c.f. grammars :definition

## Definition

A context-free grammar is a triple  $\langle A, N, R \rangle$  where :

- $A$  is a finite alphabet, called the **terminal** alphabet
- $N$  is a finite alphabet, called the **non-terminal** alphabet
- $R$  is a finite set of ordered pairs,  $R \subseteq N \times (A \cup N)^*$  ;  $R$  is called the set of **productions**.

$R$  is also called the set of “rules” of the grammar.

For every word  $w \in (N \cup A)^*$  we define

$$L(G, w) = \{u \in A^* \mid S \xrightarrow{*}_R u\}$$

it is the **language** generated by  $G$  from  $w$ .

## c.f. grammars : examples

A rule  $(S, m) \in R$  is often denoted by

$$S \longrightarrow m$$

## Example

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

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

$$R : S \longrightarrow aSS, \quad S \longrightarrow b.$$

A derivation :

$$S \longrightarrow aSS \longrightarrow aSaSS \longrightarrow abaSS \longrightarrow abaSb \longrightarrow ababb$$

Hence  $ababb \in L(G, S)$ .

## c.f. grammars : examples

## Example

$H = \langle A, N, R \rangle$  where

$$N = \{C\}, \quad A = \{s, b, e, w, i\}$$

$R$  consists of the rules :

$$C \longrightarrow CsC, \quad C \longrightarrow bCe, \quad C \longrightarrow wC, \quad C \longrightarrow i$$

abbreviated as :

$$C \longrightarrow CsC \mid bCe \mid wC \mid i$$

*Intuitive meaning :*

$s$  = separator,  $b$  = begin,  $e$  = end,

$w$  = while cond,  $i$  = instruction.

## c.f. grammars : examples

## Example

$H = \langle A, N, R \rangle$  with set of rules :

$$C \longrightarrow CsC | bCe | wC | i$$

*Some derivation and word :*

$$\begin{aligned} C &\longrightarrow wC \longrightarrow wbCe \longrightarrow wbCsCe \longrightarrow wbwCsCe \\ &\longrightarrow wbwisCe \longrightarrow wbwisie \end{aligned}$$



## c.f. grammars : examples

## Example

*wbwise can be seen as :*

```
while cond
--begin
--while cond
----i
--;
--i
--end
```

# context-free languages

## Definition

A language  $L \subseteq A^*$  is called *context-free* iff there exists some context-free grammar  $G = \langle A, N, R \rangle$  and some non-terminal  $S \in N$  such that

$$L = L(G, S)$$

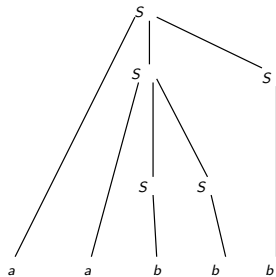
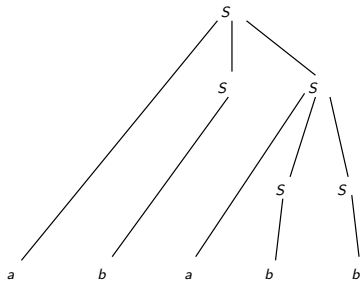
We denote by  $\text{CF}(A^*)$  the set of all context-free languages over the alphabet  $A$ .

We sometimes include  $S$  in the grammar  $\langle A, N, R, S \rangle$  and call  $S$  the “starting-symbol” or the “*axiom*” of the grammar.

## Derivation-trees : examples

Some **derivation-trees** for the grammar :  $G_1 = \langle \{a, b\}, \{S\}, R_1 \rangle$   
with

$$R_1 : S \longrightarrow aSS, S \longrightarrow b.$$



# Derivation-trees : definition

## Definition

We call *derivation-tree* for the grammar  $G = \langle A, N, R \rangle$ , every planar, rooted tree,  $T$ , labelled over  $A \cup N \cup \{\varepsilon\}$  fulfilling both properties : for every node  $x$ , with sequence of sons  $y_1, y_2, \dots, y_k$ ,

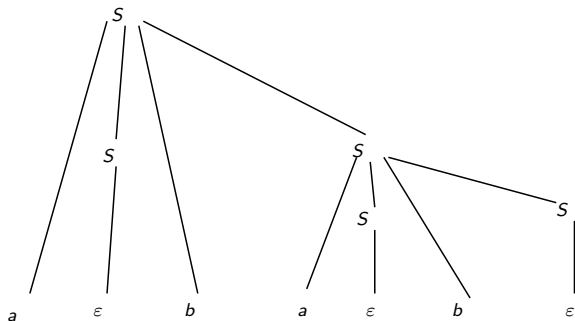
(D1) :  $T(x) \rightarrow T(y_1) \cdot T(y_2) \cdots T(y_k)$  is a rule of  $R$

(D2) : if  $T(y_i) = \varepsilon$  then  $k = i = 1$ .

# Derivation-trees : examples

Some **derivation-tree** for the grammar :  $G_2 = \langle \{a, b\}, \{S\}, R_2 \rangle$  with

$$R_2 : S \longrightarrow aSbS, S \longrightarrow bSaS, S \longrightarrow \varepsilon.$$

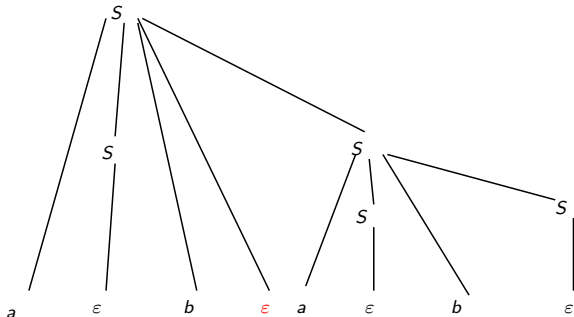


## Derivation-trees : examples

$G_2 = \langle \{a, b\}, \{S\}, R_2 \rangle$  with

$$R : S \longrightarrow aSbS, S \longrightarrow bSaS, S \longrightarrow \epsilon.$$

The following planar tree is **not** a derivation-tree (D2 is violated)

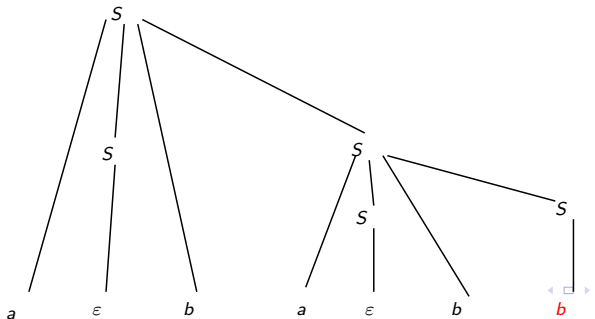


# Derivation-trees : examples

$G_2 = \langle \{a, b\}, \{S\}, R_2 \rangle$  with

$$R : S \longrightarrow aSbS, S \longrightarrow bSaS, S \longrightarrow \varepsilon.$$

The following planar tree is **not** a derivation-tree for  $G_2$  (D1 is violated)



## Derivation-trees : the frontier

Let  $T$  be a rooted, planar, tree labelled over  $A \cup \{\varepsilon\}$  (i.e each node  $x$  has a label  $T(x) \in A \cup \{\varepsilon\}$ ). Let  $y_1, \dots, y_i, \dots, y_n$  be the sequence of leaves of  $T$ , ordered from left to right. We call **frontier** of  $T$  the word

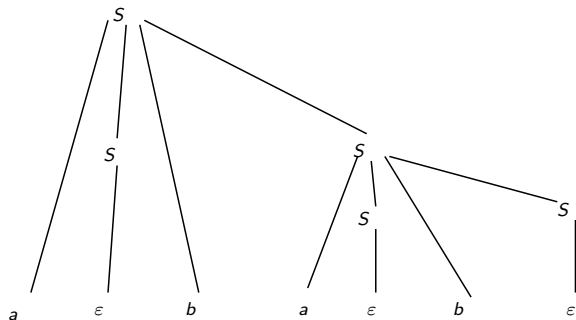
$$\text{fr}(T) = T(y_1) \cdots T(y_i) \cdots T(y_n).$$



# Derivation-trees : the frontier

The frontier of the derivation-tree below is :

$$\text{fr}(T) = a \cdot \varepsilon \cdot b \cdot a \cdot \varepsilon \cdot b \cdot \varepsilon = \mathit{abab}$$



# Construction-trees :definition

## Definition

We call *construction-tree* for the grammar  $G = \langle A, N, R \rangle$ , every planar, rooted tree,  $T$ , labelled over  $R$  fulfilling both properties : for every node  $x$ , with sequence of sons  $y_1, y_2, \dots, y_k$ ,

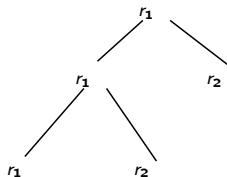
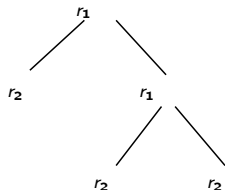
**(C1)** :  $T(x)$  is a rule of the form  $U \longrightarrow w_1 U_1 \cdot w_2 U_2 \cdots w_k U_k w_{k+1}$  where  $w_1, w_2 \dots w_{k+1} \in A^*$ ,  $U_1, U_2, \dots U_k \in N$

**(C2)** :  $\forall j \in [1, k]$ ,  $T(y_j)$  is a rule with left-hand side  $U_j$ .

## Construction-trees : examples

Some **construction-trees** for the grammar :  $G_1 = \langle \{a, b\}, \{S\}, R_1 \rangle$   
with

$$R_1 = \{r_1, r_2\} \quad r_1 : S \longrightarrow aSS, \quad r_2 : S \longrightarrow b.$$



# Construction-trees : the yield

The yield of a construction-tree  $T$  is defined inductively by :

1- if  $r : U \rightarrow w \in A^*$

$$\text{yd}(r) = w$$

2- if  $r : U \rightarrow w_1 U_1 \cdot w_2 U_2 \cdots w_k U_k w_{k+1}$

$$\text{yd}(r(T_1, T_2, \dots, T_k)) = w_1 \cdot \text{yd}(T_1) \cdot w_2 \cdot \text{yd}(T_2) \cdots w_k \cdot \text{yd}(T_k) \cdot w_{k+1}.$$

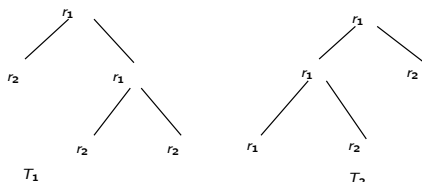
## Construction-trees : examples

Some **construction-trees** for the grammar :  $G_1 = \langle \{a, b\}, \{S\}, R_1 \rangle$   
with

$$R_1 = \{r_1, r_2\} \quad r_1 : S \longrightarrow aSS, \quad r_2 : S \longrightarrow b.$$

$$yd(T_1) = a \cdot yd(T_1/0) \cdot yd(T_1/1) = ab \cdot yd(T_1/1) = ababb,$$

$$yd(T_2) = a \cdot yd(T_2/0) \cdot yd(T_2/1) = a \cdot yd(T_2/0)b = aabbb.$$



# Leftmost derivations

## Definition

Let  $G = \langle A, N, R \rangle$  be a context-free grammar. We define the binary relation  $\ell \rightarrow_R \subseteq (A \cup N)^* \times (A \cup N)^*$  by :  $u \ell \rightarrow_R v$  if and only if,  $\exists \alpha \in A^*, \exists (\ell, r) \in R, \exists \beta \in (A \cup N)^*$

$$u = \alpha \ell \beta, v = \alpha r \beta.$$

# Leftmost derivations

$$\ell \xrightarrow{*}_R = \bigcup_{n \geq 0} \ell \xrightarrow{n}_R, \quad \ell \xrightarrow{+}_R = \bigcup_{n \geq 1} \ell \xrightarrow{n}_R.$$

A sequence  $(u_0, u_1, \dots, u_n)$  such that

$u = u_0 \xrightarrow{R} u_1 \xrightarrow{R} \dots \xrightarrow{R} u_i \xrightarrow{R} u_{i+1} \dots \xrightarrow{R} u_n$  is called a **leftmost derivation** (modulo  $R$ ). Its **length** is  $n$ .

## Leftmost derivations :example

## Example

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

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

$$R : S \longrightarrow aSS, \quad S \longrightarrow b.$$

A leftmost derivation :

$$\begin{aligned} S &\xrightarrow{\ell} aSS &\xrightarrow{\ell} aaSSS &\xrightarrow{\ell} aabSS &\xrightarrow{\ell} aabaSSS &\xrightarrow{\ell} aababSS \\ & & & & & \xrightarrow{\ell} aababbS &\xrightarrow{\ell} aababbb \end{aligned}$$

Hence  $aababbb \in L(G, S)$ .



# Rightmost derivations

## Definition

Let  $G = \langle A, N, R \rangle$  be a context-free grammar. We define the binary relation  $\xrightarrow{R,r} \subseteq (A \cup N)^* \times (A \cup N)^*$  by :  $u \xrightarrow{R,r} v$  if and only if,  $\exists \alpha \in (A \cup N)^*, \exists (l, r) \in R, \exists \beta \in A^*$

$$u = \alpha l \beta, \quad v = \alpha r \beta.$$

# Rightmost derivations

$$\xrightarrow{*}_{\mathcal{R},r} = \bigcup_{n \geq 0} \xrightarrow{n}_{\mathcal{R},r}, \quad \xrightarrow{+}_{\mathcal{R},r} = \bigcup_{n \geq 1} \xrightarrow{n}_{\mathcal{R},r}.$$

A sequence  $(u_0, u_1, \dots, u_n)$  such that

$u = u_0 \xrightarrow{\mathcal{R},r} u_1 \xrightarrow{\mathcal{R},r} \dots \xrightarrow{\mathcal{R},r} u_i \xrightarrow{\mathcal{R},r} u_{i+1} \dots \xrightarrow{\mathcal{R},r} u_n$  is called a **rightmost derivation** (modulo  $R$ ). Its **length** is  $n$ .

# Rightmost derivations

## Example

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

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

$$R : S \rightarrow aSS, \quad S \rightarrow b.$$

A rightmost derivation :

$$\begin{aligned} S &\xrightarrow{\mathcal{R},r} aSS \xrightarrow{\mathcal{R},r} aSb \xrightarrow{\mathcal{R},r} aaSSb \xrightarrow{\mathcal{R},r} aaSaSSb \\ &\xrightarrow{\mathcal{R},r} aaSaSbbb \xrightarrow{\mathcal{R},r} aaSabbbb \xrightarrow{\mathcal{R},r} aababbbb. \end{aligned}$$

Hence  $aababbbb \in L(G, S)$ .

# Trees vs derivations

## Theorem

Given a context-free  $G = \langle A, N, R \rangle$  a non-terminal  $S \in N$  and a word  $u \in A^*$ , the following propositions are equivalent :

- (1)  $\exists T$  *derivation-tree*, with root labelled by  $S$  and frontier equal to  $u$ .
- (2)  $\exists D$  *leftmost-derivation* (modulo  $R$ ) from  $S$  to  $u$ .
- (3)  $\exists D$  *rightmost-derivation* (modulo  $R$ ) from  $S$  to  $u$ .
- (4)  $\exists D$  *derivation* (modulo  $R$ ) from  $S$  to  $u$ .

# Trees vs derivations

Let us sketch a proof of the theorem.

Recall : a “traversal” of a rooted tree, is a total ordering of its nodes  $x_1, x_2, \dots, x_n$  such that, if  $x_i$  is an ancestor of  $x_j$ , then  $i \leq j$ .

(1)  $\Rightarrow$  (2) :

The depth-first, left-to-right, traversal of  $T$  gives a leftmost-derivation of  $\text{fr}(T)$ .

(1)  $\Rightarrow$  (3) :

The depth-first, right-to-left, traversal of  $T$  gives a rightmost-derivation of  $\text{fr}(T)$ .

(2)  $\Rightarrow$  (4) :

Every leftmost-derivation is a derivation.

(3)  $\Rightarrow$  (4) :

Every rightmost-derivation is a derivation.

## Trees vs derivations

(4)  $\Rightarrow$  (1) : by induction on the length of the derivation.

Let us suppose that

$$D : S = v_0 \longrightarrow_R v_1 \longrightarrow_R \dots \longrightarrow_R v_i \longrightarrow_R v_{i+1} \dots \longrightarrow_R v_n = u$$

be a derivation (modulo  $R$ ).

**Basis** :  $n = 1$

The tree with root  $S$  and sons the letters of  $u$  is a derivation-tree  $T$  with  $\text{fr}(T) = u$ .

**Induction-step** :  $n \geq 2$

$$S \rightarrow v_1$$

is a rule of  $R$  :

$$v_1 = w_1 U_1 \cdot w_2 \cdot U_2 \cdots w_k U_k w_{k+1}$$

where  $w_1, w_2 \dots w_{k+1} \in A^*$ ,  $U_1, U_2, \dots U_k \in \mathcal{N}$

## Trees vs derivations

Then  $u$  must have the form

$$u = w_1 u_1 \cdot w_2 u_2 \cdots w_k u_k w_{k+1}$$

where

$$U_k \xrightarrow{n_k}_R u_k, \quad \sum_{j=1}^k n_k = n - 1$$

(this shape for  $u$  will be proved in course 7, fundamental lemma, version 2).

By induction hypothesis : for every  $j \in [1, k]$ , there exists a derivation-tree  $T_k$  with root  $U_k$  and  $\text{fr}(T_k) = u_k$ . The derivation-tree  $T$  on next figure has root  $S$  and  $\text{fr}(T) = u$ .

## Trees vs derivations

