PRINCIPES DE L'ANALYSE SYNTAXIQUE

ESI – École nationale Supérieure en Informatique

Analyse syntaxique en bref I

- L'objectif d'une analyse syntaxique est de reconnaître si un programme donné (le programme source dont les entités lexicales ont été codées) appartient au langage engendré par une grammaire dite hors-contexte (de type 2 dans la classification de Chomsky).

LES GRAMMAIRES NON-CONTEXTUELLES

- Une grammaire non-contextuelle G est défini par un quadruplet <**N,T,P,S**,> où
- **N** : ensemble fini non vide de symboles appelés symboles non terminaux,
- **T** : ensemble fini de symboles appelés symboles terminaux,
- **S** : S $\in$ N, symbole de départ ou axiome de la grammaire,
- **P** : règles de réécritures ou ensembles de production,
  - Chaque production est de la forme :
  - $\mathbf{A} \rightarrow \alpha$
  - où $A \in N$ et $\alpha \in (N \cup T)^*$

LES GRAMMAIRES NON-CONTEXTUELLES

- La grammaire G suivante décrit les expressions arithmétiques :
- G = <{E}, {+,-,*,/,(,),id}, P, E>
- P : $\mathbf{E \rightarrow E + E}$
- $\mathbf{E \rightarrow E - E}$
- $\mathbf{E \rightarrow E * E}$
- $\mathbf{E \rightarrow E / E}$
- $\mathbf{E \rightarrow (E)}$
- $\mathbf{E \rightarrow id}$
- L'ensemble des productions peut être réécrit de la manière suivante :
- P : $\mathbf{E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid id}$

LES GRAMMAIRES NON-CONTEXTUELLES

- La grammaire G' suivante décrit également les expressions arithmétiques :
- G' = <{E,T,F}, {+,-,*,/,(,),id}, P, E>
- P : $\mathbf{E \rightarrow E + T \mid T}$
- $\mathbf{T \rightarrow T * F \mid F}$
- $\mathbf{F \rightarrow (E) \mid id}$

LES GRAMMAIRES NON-CONTEXTUELLES

- Quelle Grammaire choisir ?
- Sur quels critères se baser ?

Rappels

- ***Définition d'une dérivation***
- Soit G une grammaire. On dit qu'une chaîne $\omega 2$ se dérive d'une chaîne $\omega 1$ si $\omega 2$ s'obtient de $\omega 1$ par l'application d'une règle de production de G. Cette dérivation est notée :
- $\omega 1 \Rightarrow \omega 2$

Rappels

- **<u>Exemple :</u>**
  - $A \rightarrow \beta$ est une production de la grammaire,
  - $\alpha, \gamma \in (N \cup T)^*$,
  - $\alpha\, A\, \gamma \Rightarrow \alpha\, \beta\, \gamma$

Rappels

- <u>**Série de dérivations :**</u>
- Soient $\alpha 1$, $\alpha 2$, …, α_m des chaînes appartenant à $(N \cup T)^*$, $m \geq 1$ et :
  - $\alpha 1 \Rightarrow \alpha 2$
  - $\alpha 2 \Rightarrow \alpha 3$
  - :
  - $\alpha_{m-1} \Rightarrow \alpha_m$
  - La suite de dérivation notée
  - $\alpha 1 \Rightarrow^* \alpha_m$

Rappels

- <u>**Dérivation la plus à gauche :**</u>
- Dans une suite de dérivation, si à chaque étape de dérivation une production est appliquée au non-terminal le plus à gauche, la suite de dérivation est dite ***la plus à gauche*** ou en terme anglais ***leftmost derivation***.

Rappels

- **Dérivation la plus à droite :**
- Dans une suite de dérivation, si à chaque étape de dérivation une production est appliquée au non-terminal le plus à droite, la suite de dérivation est dite ***la plus à droite*** ou en terme anglais ***rightmost derivation***.

Rappels

- La chaîne aabbaa peut s'obtenir de l'axiome S en utilisant :

  i) La dérivation la plus à gauche :

  - $S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$

  ii) La dérivation la plus à droite :

  - $S \Rightarrow aAS \Rightarrow aAa \Rightarrow aSbAa \Rightarrow aSbbaa \Rightarrow aabbaa$

Rappels

- **Langages**
- Etant donné une grammaire G, on appelle langage engendré par G et on le note L(G) le langage :
- $L(G) = \{ \omega \mid \omega \in T^* \text{ et } S \Rightarrow^+ \omega \}$

Rappels

- **<u>Exemple :</u>**
- Soit la grammaire G = <N,T,P,S,> avec :
- N = {S} ; T = {a, b} ;
- P = { $S \rightarrow aSb$; $S \rightarrow ab$ } ;
- **L(G) = { $a^n b^n \mid n \geq 1$ }**

Arbre syntaxique

- Un arbre syntaxique également appelé arbre de dérivation est une structure pour représenter une suite de dérivations.

Arbre syntaxique

- Soit G=<N,T,P,S> une grammaire context-free, un arbre de dérivation est associé à une chaîne de L(G) et défini comme suit :
  - Chaque sommet de l'arbre a une étiquette qui est un symbole de $N \cup T \cup \{\varepsilon\}$,
  - L'étiquette de la racine est le symbole S,
  - Les sommets non-feuilles de l'arbre ont des étiquettes appartenant à N,
  - Les feuilles de l'arbre ont des étiquettes appartenant à $T \cup \{\varepsilon\}$,
  - Si un sommet d'étiquette $A \in N$ a des fils d'étiquettes X_1, X_2, ..., X_N (de gauche à droite) alors $A \rightarrow X_1 X_2 \ldots X_N$ est une production de P.

Arbre syntaxique

- **<u>Exemple :</u>**
- Considérer la grammaire suivante :

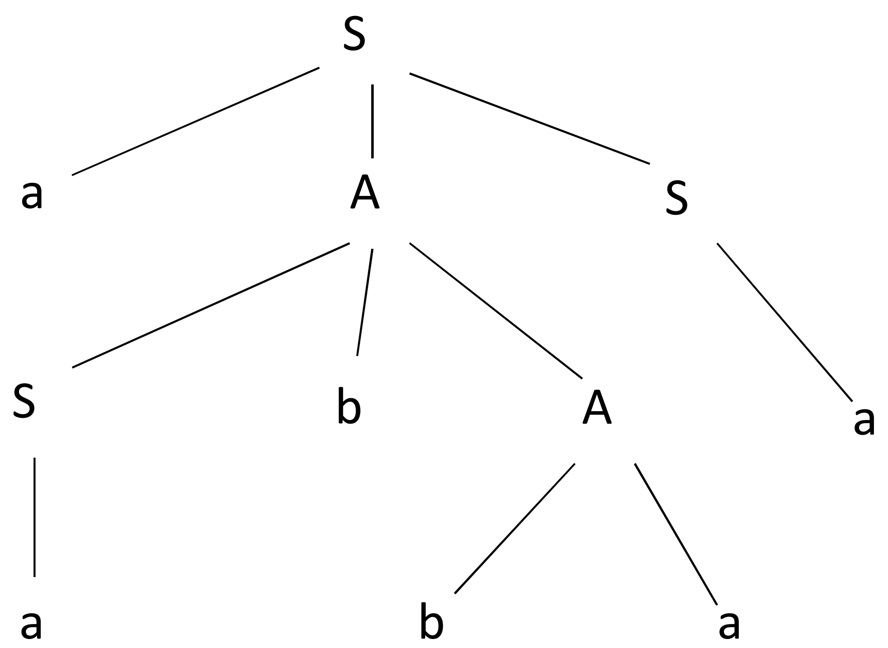
  $G = <\{S,A\}, \{a,b\}, P, S>$

  $P : S \rightarrow aAS \mid a$

  $A \rightarrow SbA \mid SS \mid ba$

- La suite de dérivations suivante pour obtenir la chaîne aabbaa :
  - $S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$

Arbre syntaxique



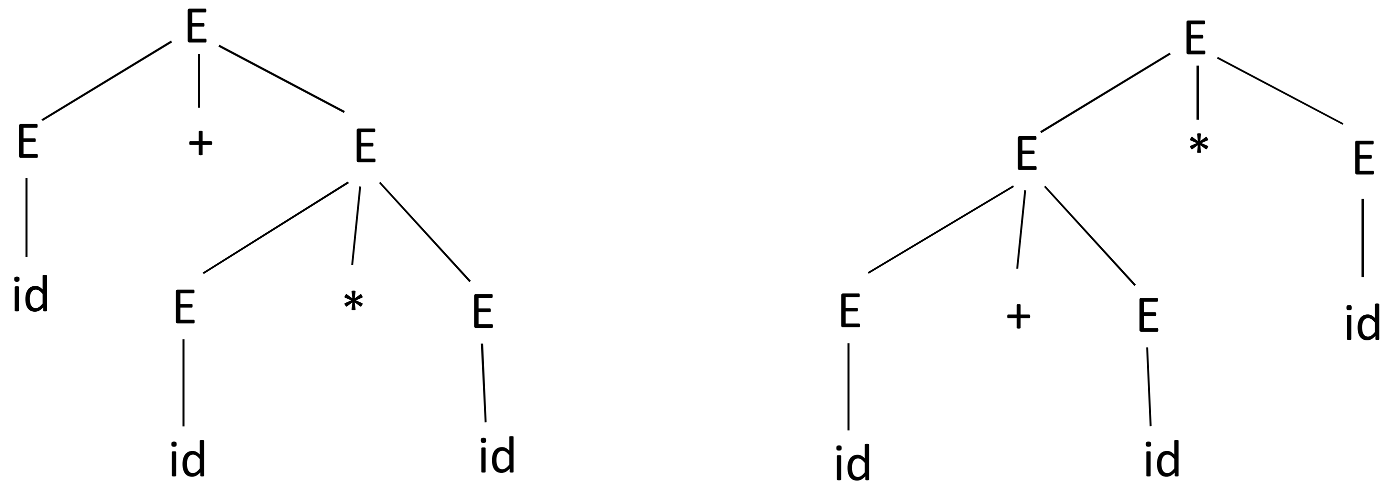
Ambiguïté ou non ambigüité d'une grammaire

- **Définition :**
- Une grammaire non-contextuelle est dite ambiguë si une chaîne ω ($\in T^*$) possède deux (ou plus) arbres syntaxiques.

Ambiguïté ou non ambigüité d'une grammaire

- **<u>Exemple :</u>**
- Soit G = <{E}, {+,-,*,/,(,),id}, P, E>
- P : $\mathbf{E} \rightarrow \mathbf{E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid id}$

Ambiguïté ou non ambigüité d'une grammaire



Ambiguïté ou non ambigüité d'une grammaire

- **<u>Remarques :</u>**
- Obtenir deux suites de dérivations pour une chaîne $\omega \in T^*$ n'implique pas forcément l'existence de deux arbres syntaxiques.
- Dans un cas général, la non-ambigüité d'une grammaire est un problème indécidable. On dispose seulement de conditions suffisantes mais non nécessaires pour prouver qu'une grammaire est non ambigüe.
- Dans le cas des grammaires où nous disposons d'une connaissance sur le langage engendré (e.g. expressions arithmétiques) nous pouvons donner une grammaire non ambigüe qui génère le même langage.

Ambiguïté ou non ambigüité d'une grammaire

- La grammaire G' = <{E,T,F}, {+,-,*,/,(,),id}, P', E> dont les productions sont données ci-après, génère les expressions arithmétiques et elle est non ambigüe :
  - $E \rightarrow \quad E + T \mid E - T \mid T$
  - $T \rightarrow \quad T * F \mid T / F \mid F$
  - $F \rightarrow \quad (E) \mid i$
- Il sera prouvé ultérieurement que cette grammaire est non ambigüe.

Transformations de Grammaires non-contextuelles

- Pourquoi transformer ?
- Comment transformer ?
- Est-il possible de toujours transformer ?

Transformations de Grammaires non-contextuelles

- Si un langage non-contextuel L est non vide, il peut être généré par une grammaire non-contextuelle ayant les propriétés suivantes :
  - Chaque non-terminal et chaque terminal apparaît dans la dérivation d'un mot de L.
  - Il n'y a pas de production unitaire (de la forme $A \to B$ où $(A,B) \in N^2$).
  - Si $\varepsilon \notin L$ alors les ε-productions (de la forme $A \to \varepsilon$ où $A \in N$) peuvent être supprimées.

Principe des analyses descendantes et ascendantes

- L'analyse doit être "déterministe" i.e. lorsque l'analyseur syntaxique s'arrête, soit le programme est syntaxiquement correct ou il existe une erreur dans ce programme.
- Il existe deux grandes classes de méthodes pour effectuer l'analyse syntaxique : les analyses descendantes et les analyses ascendantes.

Principe des analyses descendantes et ascendantes

- **<u>Analyse descendante :</u>**
- On part de l'axiome de la grammaire pour retrouver le programme source en effectuant des dérivations successives.
- Si on se place dans l'arbre syntaxique représentant le programme source, cette stratégie revient à partir de la racine (l'axiome) et à descendre vers les feuilles représentant les symboles terminaux.

Principe des analyses descendantes et ascendantes

- **Analyse ascendante :**
- On part du programme source pour retrouver l'axiome de la grammaire en effectuant des dérivations successives inverses.
- On va cette fois partir des feuilles de l'arbre syntaxique pour remonter vers la racine.

Principe des analyses descendantes et ascendantes

- <u>**Exemple**</u>
- Soit la grammaire G dont les productions sont données ci-après et la chaîne ω=abba à analyser syntaxiquement :
  - $S \rightarrow aA \mid bB$
  - $A \rightarrow aBS \mid bS$
  - $B \rightarrow bB \mid a$

Principe des analyses descendantes et ascendantes

- Le processus d'analyse descendante de la chaîne ω=abba peut illustré par :
  - $S \Rightarrow aA \Rightarrow abS \Rightarrow abbB \Rightarrow abba$
- Le processus d'analyse ascendante de la chaîne ω=abba peut être illustré par :
  - $abba \Leftarrow^R abbB \Leftarrow^R abS \Leftarrow^R aA \Leftarrow^R S$
  - où $\Leftarrow^R$ désigne est une dérivation inverse

Backus–Naur Form (BNF)

- La forme de Backus–Naur est une méta-syntaxe utilisée pour décrire les grammaires non contextuelles (context-free). Cette forme a été développée par John Backus et Peter Naur.
- BNF est très utilisée pour la description des grammaires des langages de programmation. Beaucoup de générateurs de compilateurs ou d'analyseurs syntaxiques s'inspirent de BNF pour l'introduction des grammaires cibles.

Backus–Naur Form (BNF)

- **<u>Définition :</u>**
- Une spécification BNF est un ensemble de règles de dérivation définie par :
  - **<symbole> ::= expression$_1$ | expression$_2$ | … | expression$_N$**

  où

  - <symbole> est un non-terminal, et expression$_i$ est une suite de symboles de la grammaire (terminaux et non-terminaux et éventuellement le mot vide)
  - Les symboles qui n'apparaissent jamais en membre gauche de production sont des terminaux. Les symboles non terminaux sont toujours entre < >

Backus–Naur Form (BNF)

- **<u>Exemple :</u>**
- <Expression> ::= <Expression> "+" <Terme>

  | <Expression> "-" <Terme>

  | <Terme>
- <Terme> ::= <Terme> "*" <Facteur>

  | <Terme> "/" <Facteur>

  | <Facteur>
- <Facteur> ::= "(" <Expression> ")"

  | "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"