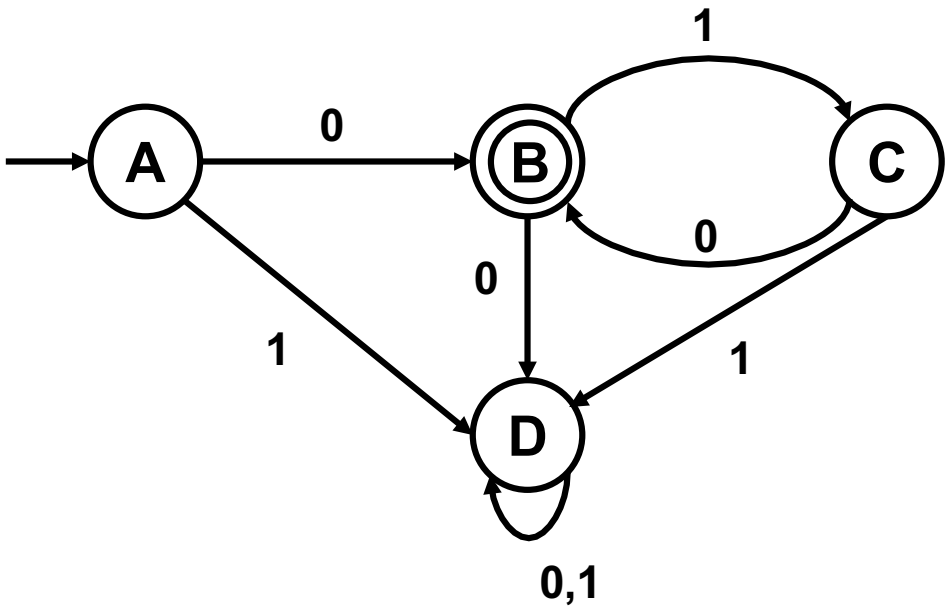
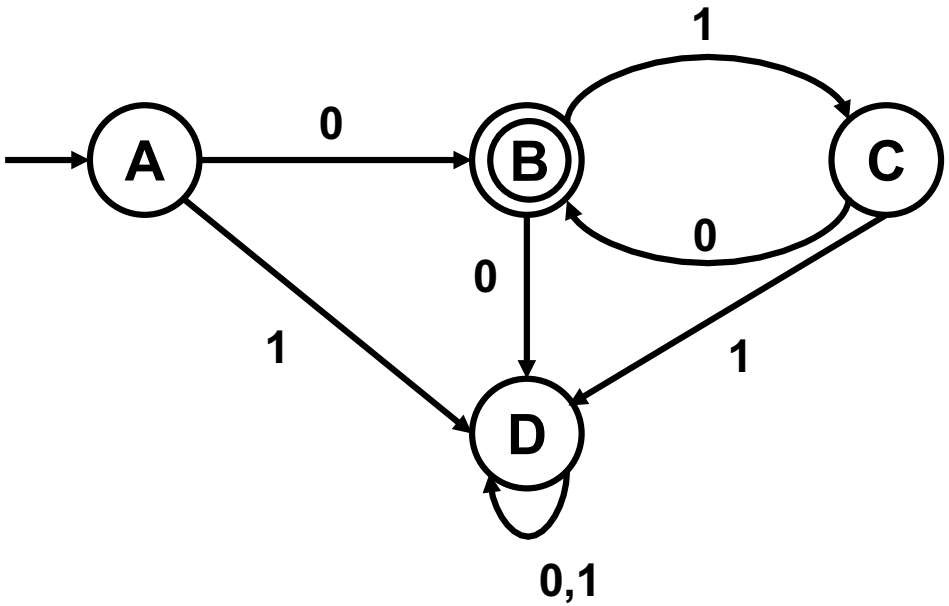
Algorithme DFA → GHC
Exercice

Trouvez la grammaire linéaire droite correspondante



Algorithme DFA → GHC
Solution

A→ 0B |1D |0
B → 0D |1C
C → 0B|1D|0
D → 0D |1D



Après élimination de *D* inutile :

A→ 0B|0
B → 1C
C → 0B|0

Fin Séance 2

Dérivation : Gauche & Droite
Exemple

- Soit la grammaire HC **G3 = <**
  - ✓ ***T={Number,+,-, ×, ÷ },***
  - ✓ ***NT ={Expr, Op},***
  - ✓ ***S=Expr,***
  - ✓ ***P = {r1,..,r6} >***
    - ✓ r1 : Expr → Expr Op Number
    - ✓ r2: | Number
    - ✓ r3 : Op → +
    - ✓ r4 : |-
    - ✓ r5 : |×
    - ✓ r6 : | ÷

Dérivation : Gauche & Droite
Exemple

Exemples de *dérivations gauches* de mots dans ***L(G3)***

1: $\boldsymbol{Expr} \Rightarrow_{r1} \boldsymbol{Expr}\ Op\ Number \Rightarrow_{r2} Number\ \boldsymbol{Op}\ Number \Rightarrow_{r3} Number + Number$

2: $\boldsymbol{Expr} \Rightarrow_{r1} \boldsymbol{Expr}\ Op\ Number \Rightarrow_{r1} \boldsymbol{Expr}\ Op\ Number\ Op\ Number$

$\Rightarrow_{r2} Number\ \boldsymbol{Op}\ Number\ Op\ Number \Rightarrow_{r3} Number + Number\ \boldsymbol{Op}\ Number$

$\Rightarrow_{r5} Number + Number * Number$

Dérivation : Gauche & Droite
Exercice

Dérivation à droite de l'expression Number+Number*Number

Dérivation à Droite

$$\boldsymbol{Expr} \Rightarrow_{r1} Expr\ \boldsymbol{Op}\ Number \Rightarrow_{r5} \boldsymbol{Expr} * Number \Rightarrow_{r1} Expr\ \boldsymbol{Op}\ Number * Number$$

$$\Rightarrow_{r3} \boldsymbol{Expr} + Number * Number \Rightarrow_{r2} Number + Number * Number$$

Règles appliquées : (r1, r5, r1, r3, r2)

Mêmes règles mais pas dans le même ordre

Dérivation à Gauche

$$\boldsymbol{Expr} \Rightarrow_{r1} \boldsymbol{Expr}\ Op\ Number \Rightarrow_{r1} \boldsymbol{Expr}\ Op\ Number\ Op\ Number \Rightarrow_{r2} Number\ \boldsymbol{Op}\ Number\ Op\ Number$$

$$\Rightarrow_{r3} Number + Number\ \boldsymbol{Op}\ Number \Rightarrow_{r5} Number + Number * Number$$

Règles appliquées : (r1, r1, r2, r3, r5)

Analyseur Syntaxique

- L'analyseur syntaxique doit découvrir automatiquement pour une expression donnée en langage ***L***, son arbre syntaxique de dérivation par rapport à la grammaire de ce langage ***L***

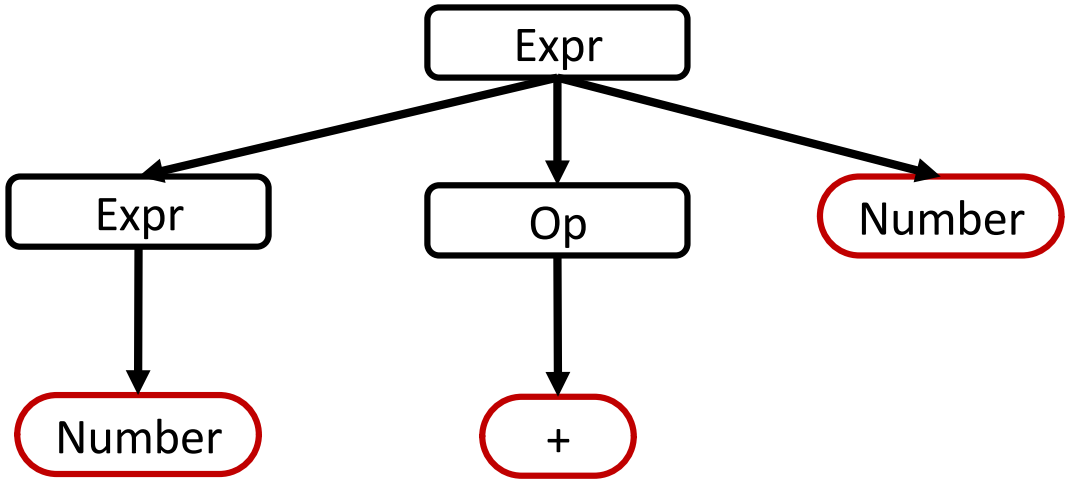
  - La **racine de l'arbre** syntaxique est : le symbole non-terminal initial **S**

  - Les **nœuds de l'arbre** syntaxique sont : le résultat de l'application des règles de production $\in (T \cup NT)*$

  - Les **feuilles de l'arbre** syntaxique sont : les éléments de l'expression donnée en entrée $\in T*$

Arbre Syntaxique (Arbre de dérivation)

$$\boldsymbol{Expr} \Rightarrow_{r1} \boldsymbol{Expr}\ Op\ Number \Rightarrow_{r2} Number\ \boldsymbol{Op}\ Number \Rightarrow_{r3} Number + Number$$

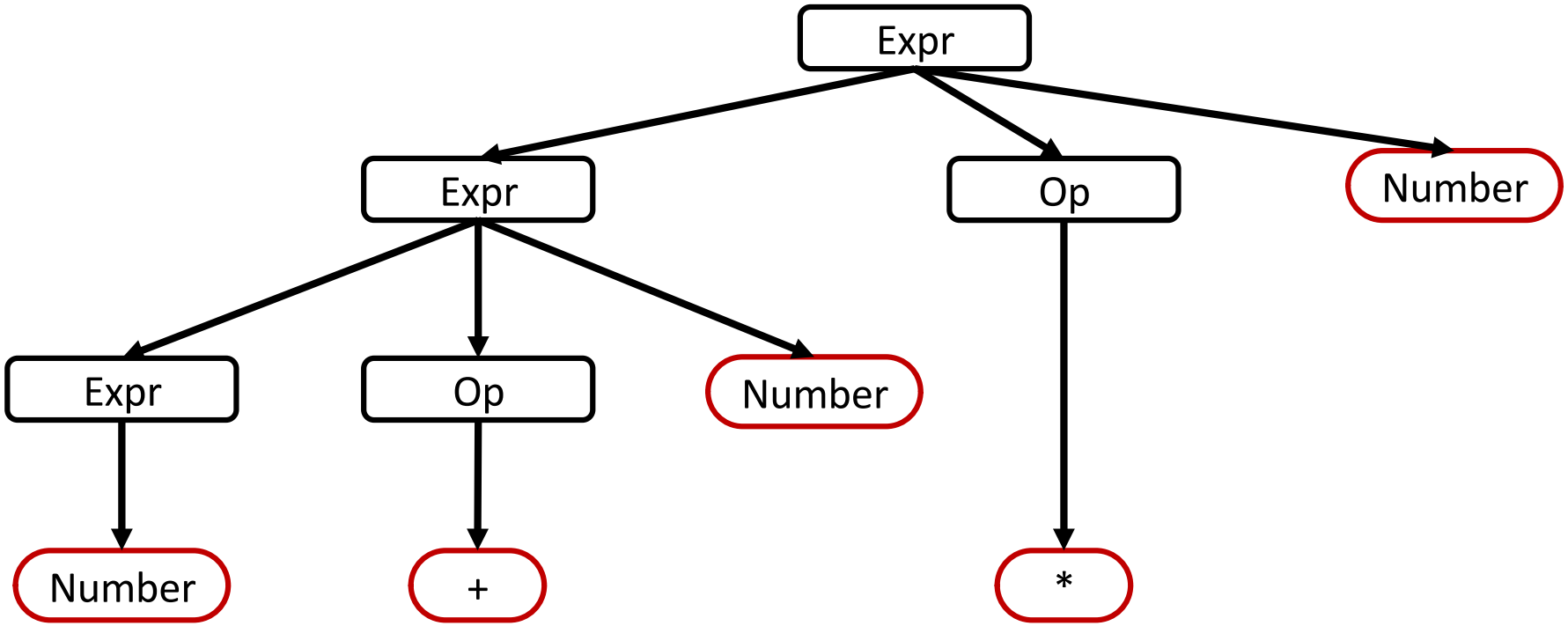


Arbre Syntaxique (Arbre de dérivation)

Dérivation à Gauche

$$\boldsymbol{Expr} \Rightarrow_{r1} \boldsymbol{Expr}\ Op\ Number \Rightarrow_{r1} \boldsymbol{Expr}\ Op\ Number\ Op\ Number \Rightarrow_{r2} Number\ \boldsymbol{Op}\ Number\ Op\ Number$$

$$\Rightarrow_{r3} Number + Number\ \boldsymbol{Op}\ Number \Rightarrow_{r5} Number + Number * Number$$

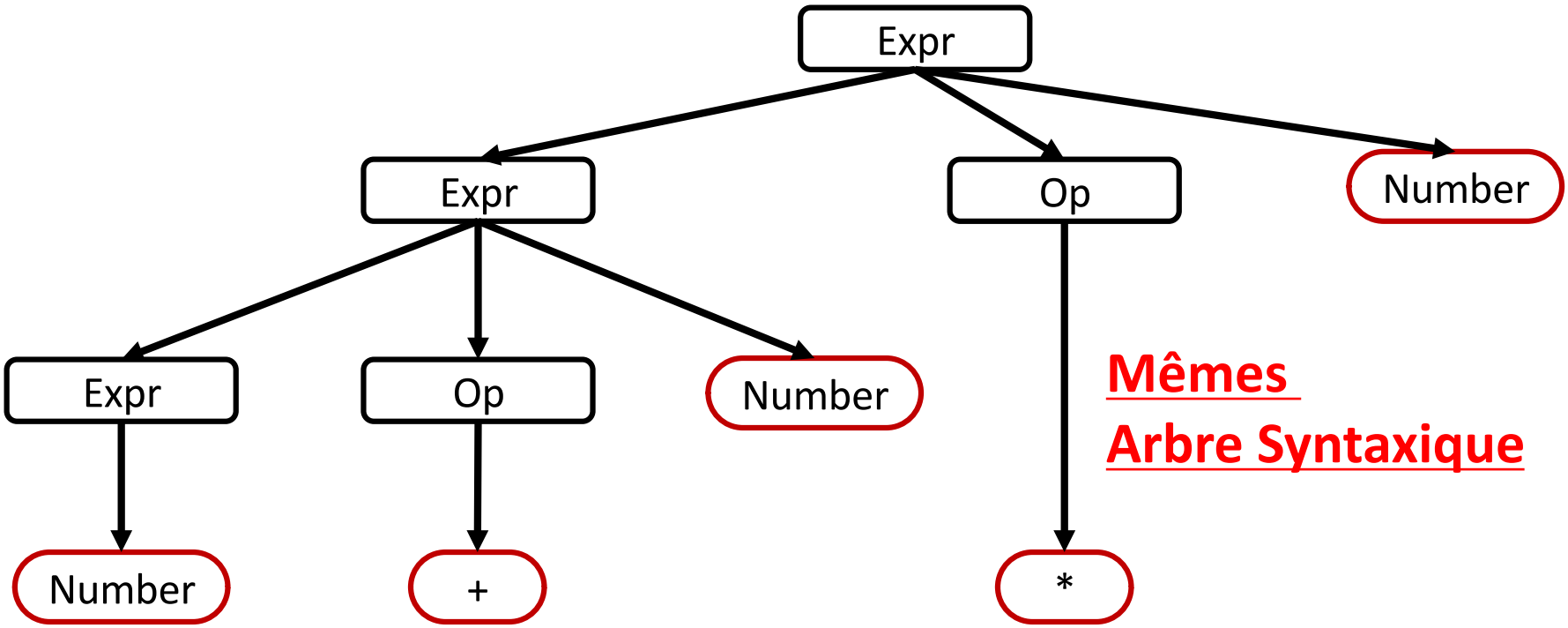


Arbre Syntaxique (Arbre de dérivation)

Dérivation à Droite

$$\boldsymbol{Expr} \Rightarrow_{r1} Expr\ \boldsymbol{Op}\ Number \Rightarrow_{r5} \boldsymbol{Expr}\ *\ Number \Rightarrow_{r1} Expr\ \boldsymbol{Op}\ Number\ *\ Number$$

$$\Rightarrow_{r3} \boldsymbol{Expr} + Number\ *\ Number \Rightarrow_{r2} Number + Number\ *\ Number$$



Grammaire hors-contexte Ambigüe

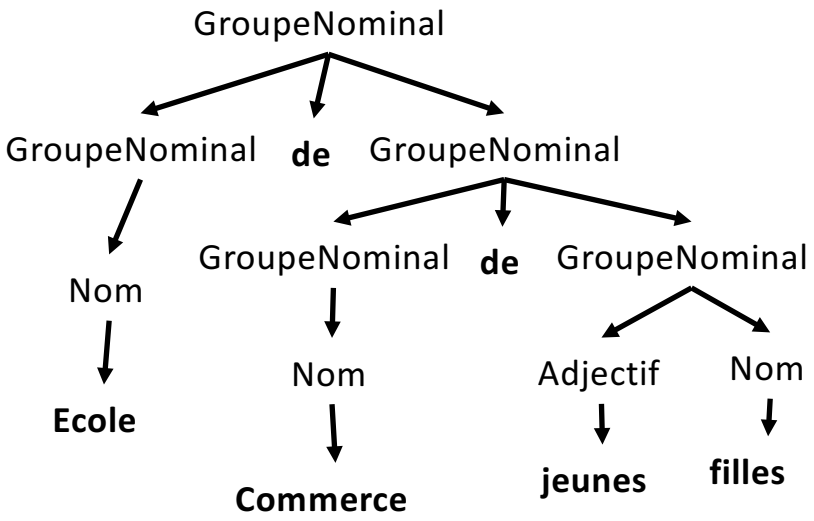
Grammaire ambigüe

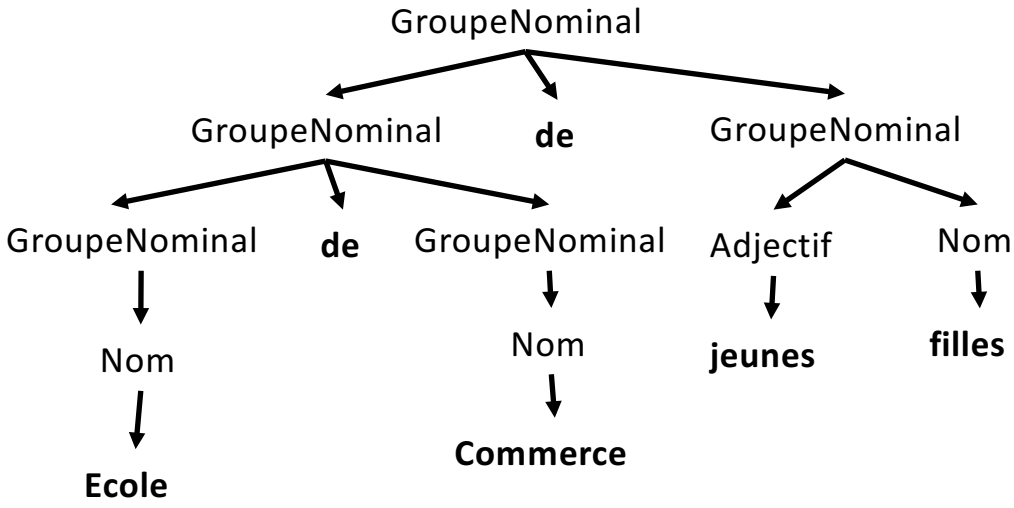
- Pour une CFG *G* tout string *w* de *L*(*G*) a au moins un arbre de dérivation pour *G*.
- $w \in L(G)$ peut avoir plusieurs arbres de dérivation pour *G* : dans ce cas on dira que la grammaire est ambigüe.
- Idéalement, pour permettre le parsing, une grammaire ne doit pas être ambigüe. En effet, l'arbre de dérivation détermine le code généré par le compilateur.

Grammaire ambigüe
Exemple

- Soit les règles de la grammaire suivante :
  - Grammaire Simpliste :
    - r1: **GroupeNominal** → Nom
    - r2 : | **Adjectif** Nom
    - r3 : | **GroupeNominal** de **GroupeNominal**
  - Trouver un arbre de dérivation du mot
    - w = « **Ecole de Commerce de jeunes filles** »

Grammaire ambigüe
Exemple

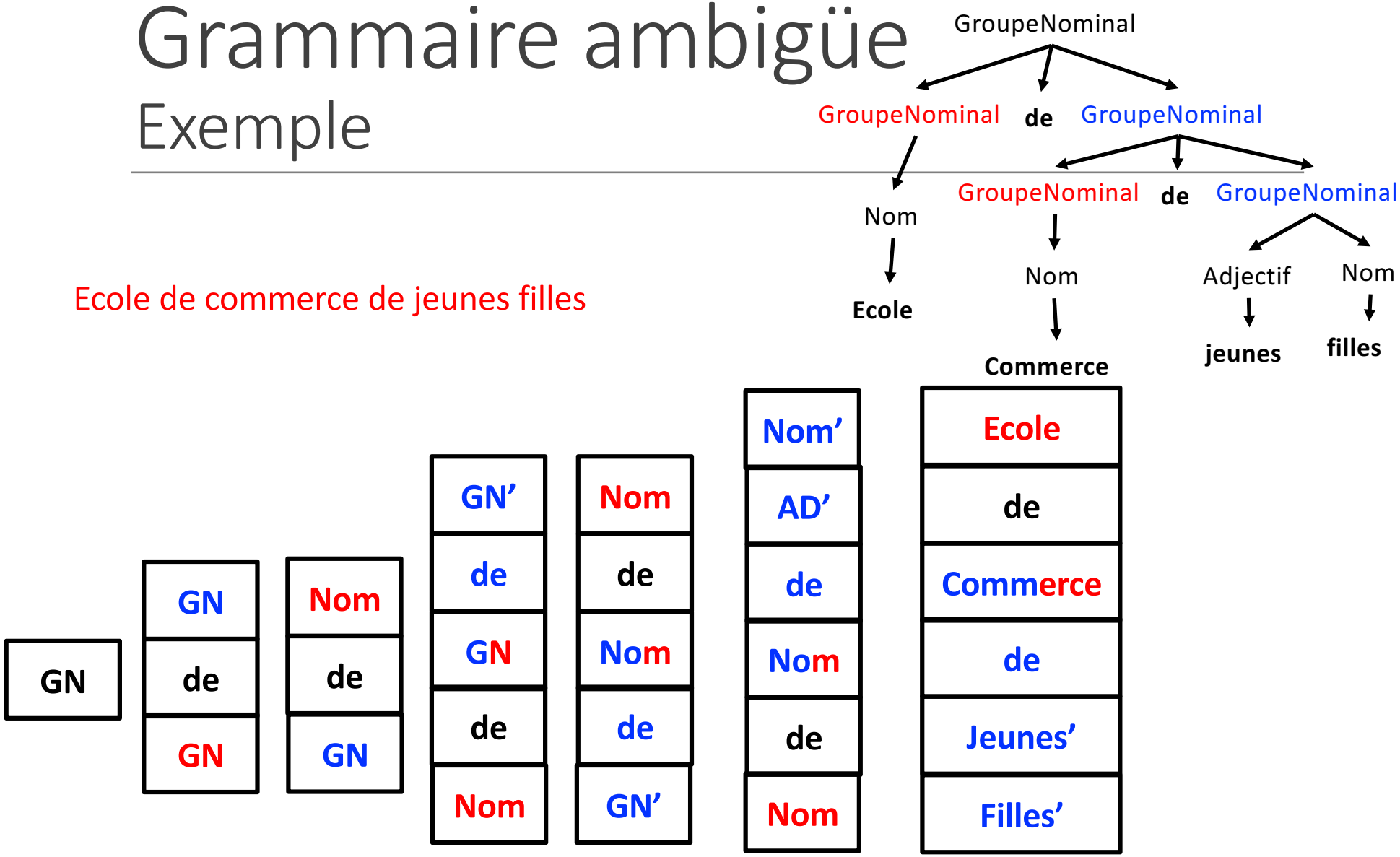




2 arbres syntaxiques différents d'une expression (i.e. **ambigüité de la grammaire**) ⇒ **2** interprétations différentes ⇒ **2** sémantiques différentes ⇒ **2** codes possibles à générer pour cette expression ⇒ **Problème de non-déterminisme pour le compilateur !!**

Grammaire ambigüe

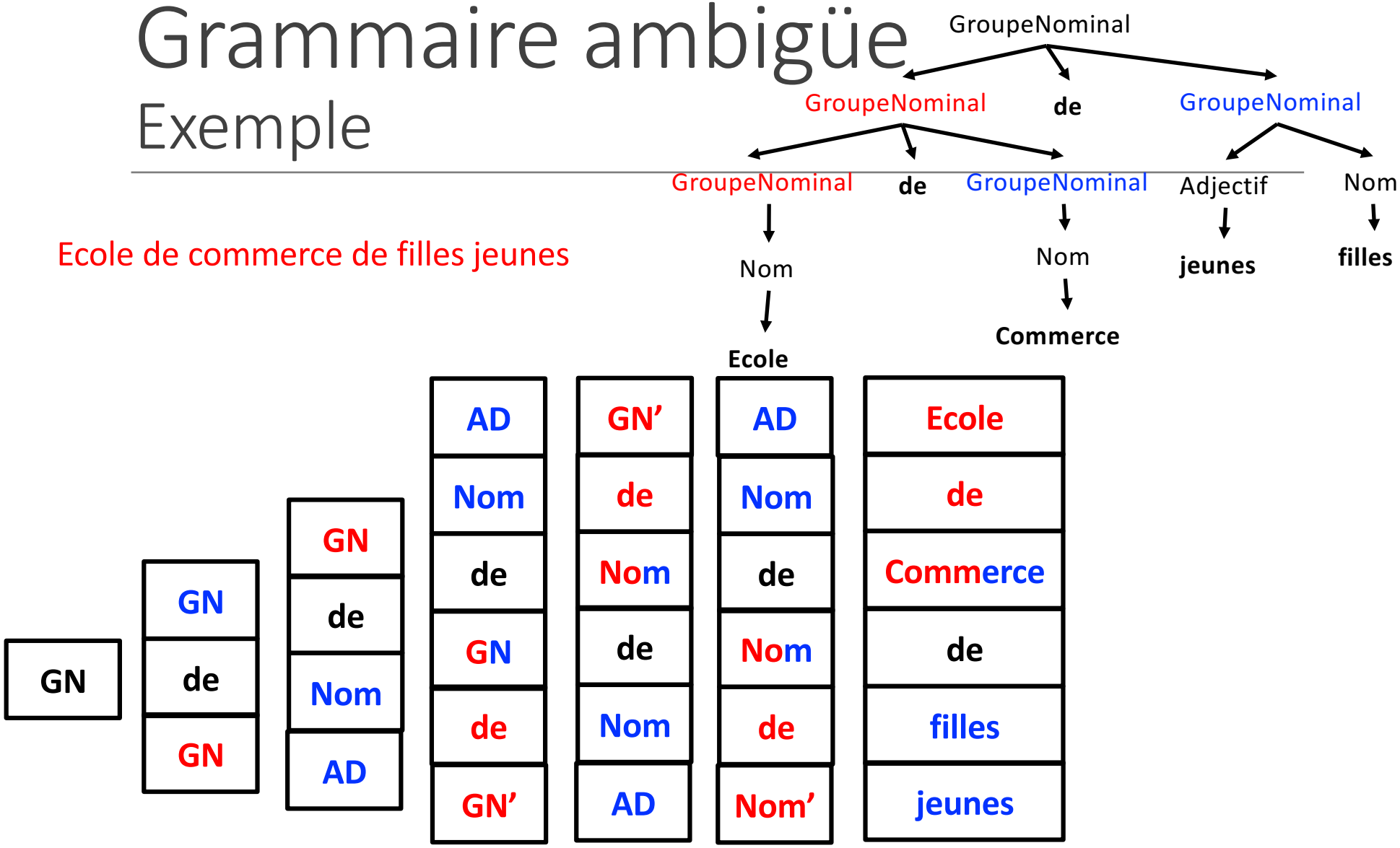
Exemple



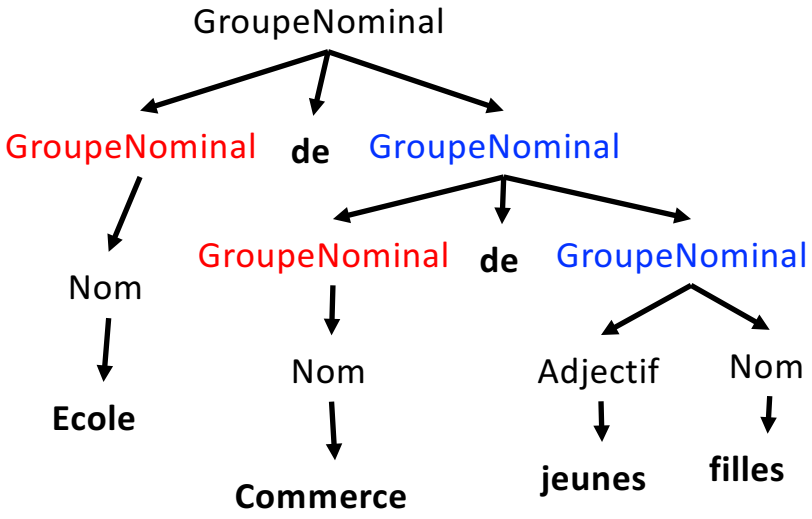
Grammaire ambigüe

Exemple

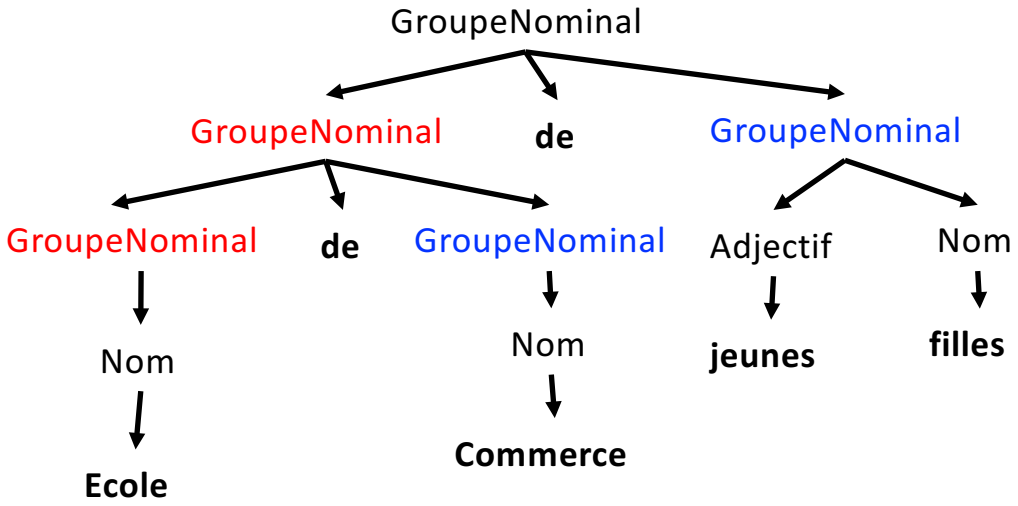
Ecole de commerce de filles jeunes



Grammaire ambigüe
Exemple



Ecole de commerce de jeunes filles



Ecole de commerce de filles jeunes

2 arbres syntaxiques différents d'une expression (i.e. **ambigüité de la grammaire**) ⇒ **2** interprétations différentes ⇒ **2** sémantiques différentes ⇒ **2** codes possibles à générer pour cette expression ⇒ **Problème de non-déterminisme pour le compilateur !!**

Grammaire ambigüe
Exercice 1

La grammaire suivante est-elle ambiguë ?

G_{arith}:

r1 : $E \rightarrow E \times E$

r2 : $E \rightarrow E + E$

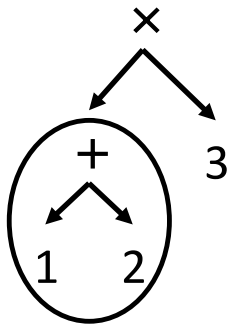
r3 : $E \rightarrow$ Nombre

Si Oui Pourquoi ?

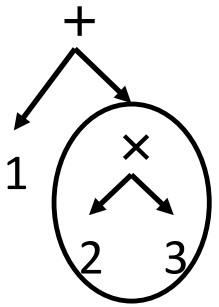
Grammaire ambigüe
Solution

- Le mot "$1 + 2 \times 3$" possède deux dérivations droites
  - $E \Rightarrow_{r1} E \times E \Rightarrow_{r2} E + E \times E \Rightarrow^*_{r3} 1 + 2 \times 3$
  - $E \Rightarrow_{r2} E + E \Rightarrow_{r1} E + E \times E \Rightarrow^*_{r3} 1 + 2 \times 3$
- Donc 2 arbres syntaxiques

Arbre 1



Arbre 2



- Donc 2 interprétations possibles
  - **$(1 + 2) \times 3 \equiv 9$**
  - **$1 + (2 \times 3) \equiv 7$**

Grammaire ambigüe
Exercice

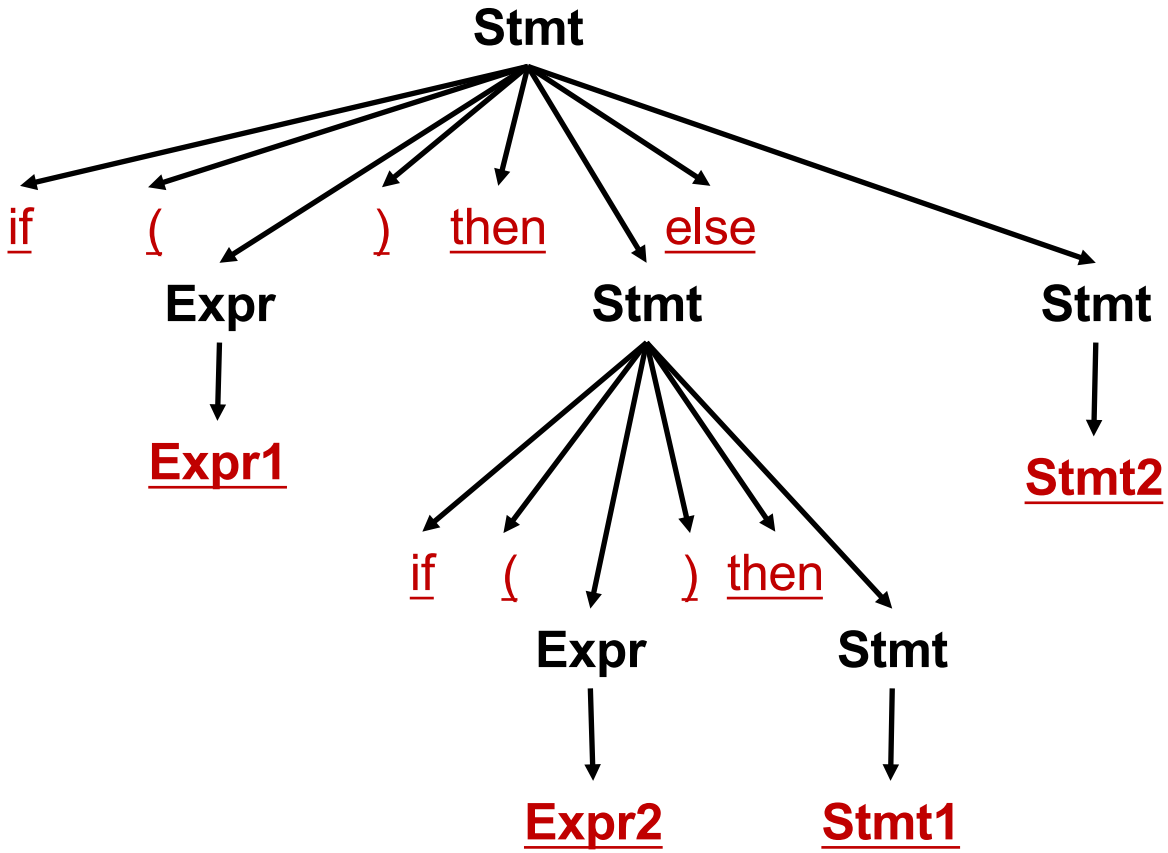
- *G:*
  - r1: **Stmt** → if (**Expr**) then **Stmt** else **Stmt**
  - r2: | if (**Expr**) then **Stmt**
  - …
  - rn | . . .

  - Trouver un arbre de dérivation du mot w = if (*Expr*1) then if (*Expr*2) then *Stmt*1 else *Stmt*2

Grammaire ambigüe

Solution 1

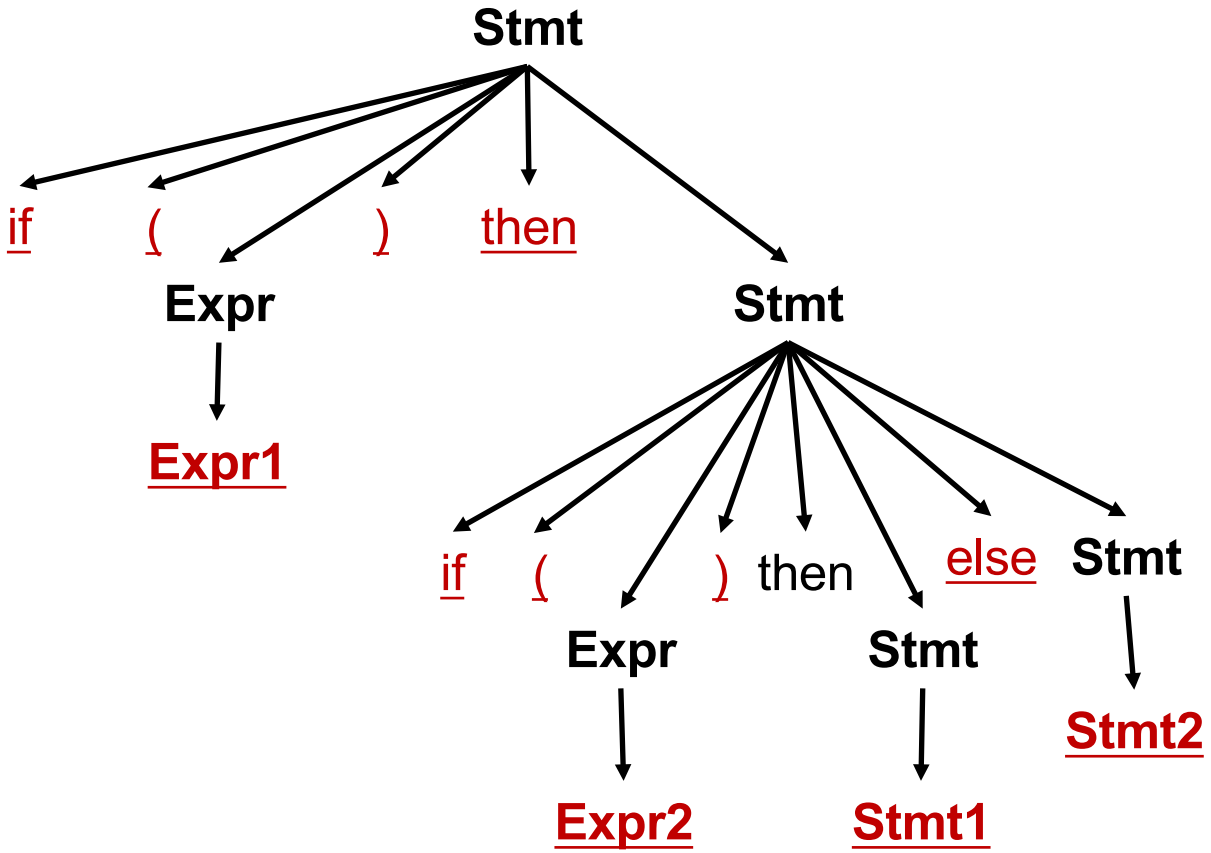
- Stmt $\Rightarrow^{r1}$
- if (**Expr**) then
  - **Stmt**
- else **Stmt**
- $\Rightarrow^{r2}$
- if (**Expr**) then
  - if (**Expr**) then
    - **Stmt**
- else **Stmt**
- …
- $\Rightarrow$
- if (*Expr*1) then
  - if (*Expr2*) then
    - *Stmt*1
- else
  - *Stmt2*



Grammaire ambigüe

Solution 2

- Stmt $\Rightarrow^{r2}$
- if (**Expr**) then
  - **Stmt**
- $\Rightarrow^{r1}$
- if (**Expr**) then
  - if (**Expr**) then
    - **Stmt**
  - else **Stmt**
- …
- $\Rightarrow$
- if (*Expr*1) then
  - if (*Expr*2) then
    - *Stmt*1
  - else
    - *Stmt*2



G est une grammaire ambiguë du fait qu'on a trouvé deux dérivations du mot w = « if (*Expr*1) then if (*Expr*2) then *Stmt*1 else *Stmt*2 »

Grammaire ambigüe

- L'ambiguïté implique que l'analyseur syntaxique ne pourra pas découvrir d'une manière unique et définitive l'arbre syntaxique de cette expression.

- Si l'analyseur syntaxique ne peut pas décider la structure syntaxique d'une expression, décider du sens (i.e. la sémantique) et donc du code exécutable équivalent à cette expression ne sera pas possible !!

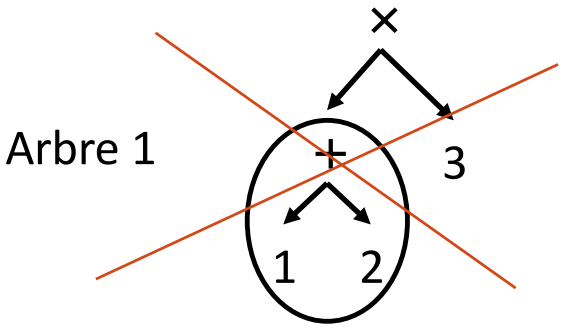
- L'Ambiguïté est donc une propriété indésirable dans une grammaire.

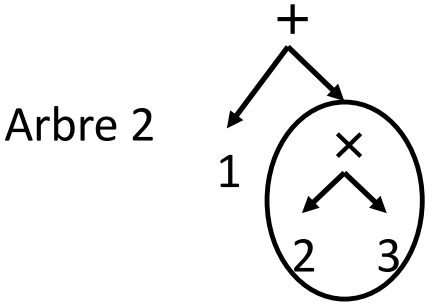
Grammaire ambigüe

- Lorsque le langage définit des strings composés d'instructions et d'opération, l'arbre syntaxique (qui va déterminer le code produit par le compilateur) doit refléter :
  - Les priorités
  - Les associativités

Grammaire ambigüe
Exemplle

- Le mot "$1 + 2 \times 3$" possède deux dérivations droites
  - ~~$E \Rightarrow_{r1} E \times E \Rightarrow_{r2} E + E \times E \Rightarrow^*_{r3} 1 + 2 \times 3$~~
  - $E \Rightarrow_{r2} E + E \Rightarrow_{r1} E + E \times E \Rightarrow^*_{r3} 1 + 2 \times 3$
- Donc 2 arbres syntaxiques





- Donc 2 interprétations possibles
  - ~~**(1 + 2) × 3 ≡ 9**~~
  - **1 + (2 × 3) ≡ 7**

C'est la 2ème interprétation qui correspond aux règles de priorité usuelles des opérateurs arithmétiques !!