ECOLE SUPERIEURE D'INGENIEUR DE RENNES

Compilation

syntaxe abstraite

Olivier Ridoux

UNIVERSITÉ DE RENNES 1

Plan

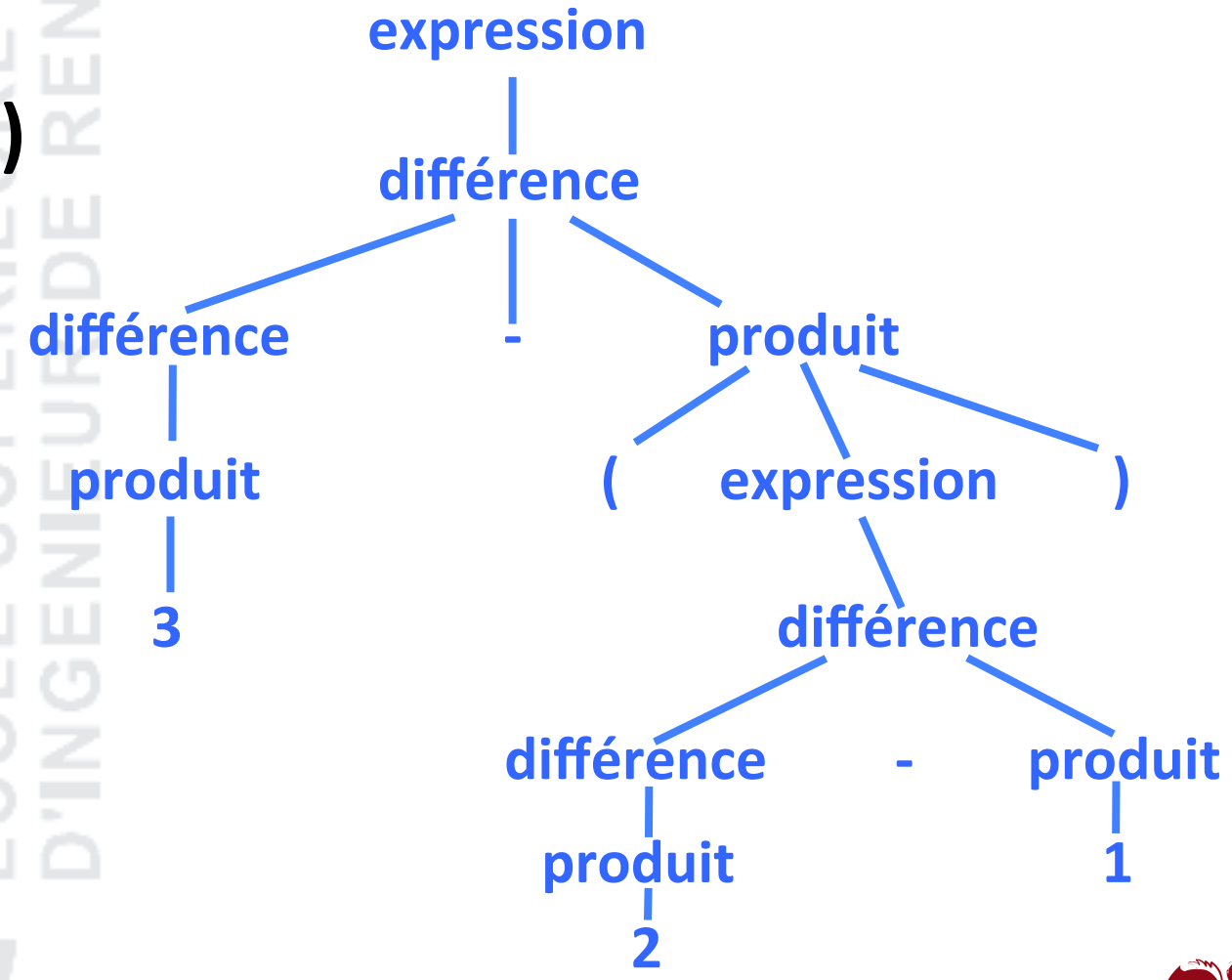
- Syntaxe abstraite
- Production de l'arbre de syntaxe abstraite en ANTLR

Syntaxe abstraite

- Syntaxe concrète
  - (presque) tous les détails du document
    - détail de la dérivation et ponctuation
    - mais mise en page souvent omise
  - AD, **arbre de dérivation** (*PT, parse tree*)
- Syntaxe abstraite
  - la structure du document, sans les détails
  - ASA, **arbre de syntaxe abstraite** (*AST, abstract syntax tree*)

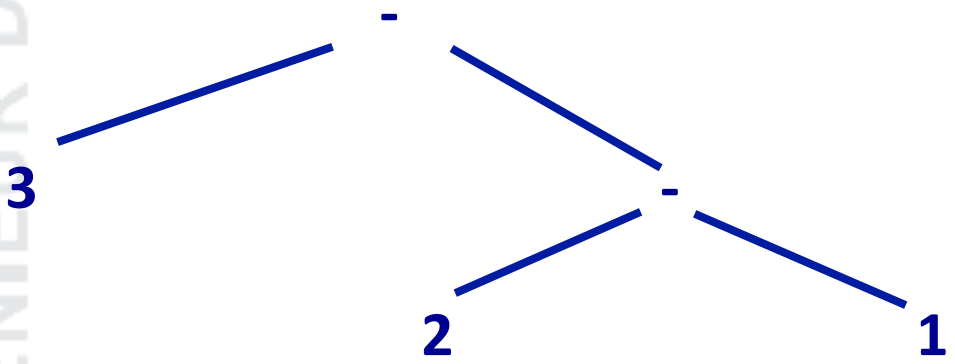
Exemple - syntaxe concrète

- **3-(2-1)**



Exemple - syntaxe abstraite

- **3-(2-1)**



Remarque : syntaxe concrète

- Souvent la syntaxe concrète est déjà une abstraction
  - omission des commentaires
    - gênant pour les commentaires codifiés à la **Doxygen** ou **Javadoc**
  - omission de la mise en page
    - gênant quand la mise en page est signifiante, à la **Scala** ou **Haskell**

Relation mot - syntaxe concrète

- Relation formelle
  - grammaire
    - **G**
  - dérivation
    - **preuve de m ∈ $\mathcal{L}$(G)**
  - arbre dérivation
    - **arbre de preuve de m ∈ $\mathcal{L}$(G)**

Complètement automatisable

Relation mot - syntaxe abstraite

- Relation informelle
  - la syntaxe abstraite préfigure la sémantique, or le lien syntaxe-sémantique est arbitraire...

  ...aucun système automatique ne peut l'inventer en totalité

Procéder à la main !

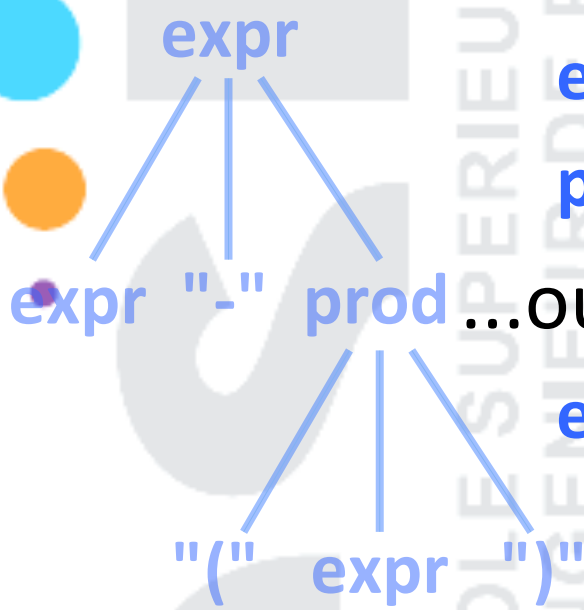
Construction de l'arbre de syntaxe abstraite (1)

- Sans filet : YACC

```
expr : expr "-" prod
        {$0 = malloc(node) ;
         $0->op = SUB ;
         $0->arg1 = $1 ;  $0->arg2 = $3 ;
        } ;
```

- Risques : gestion de la mémoire, manipulation de pointeurs, ...

Construction de l'arbre de syntaxe abstraite (2)

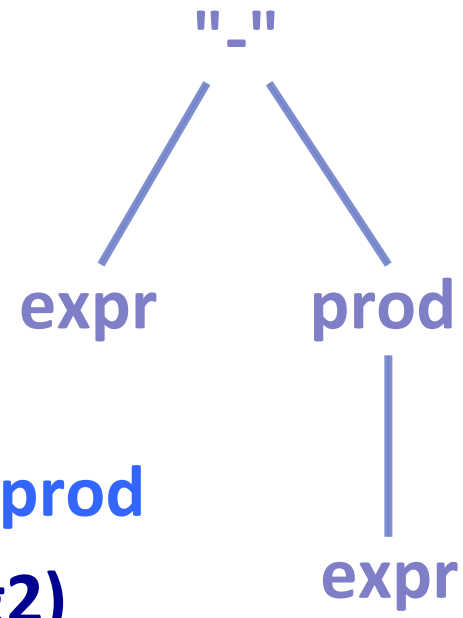


- Avec filet : ANTLR

```
expr : expr "-"^ prod ;
prod : "("! expr ")"!
```

...ou bien

```
expr : arg1=expr "-" arg2=prod
        → ^(SUB $arg1 $arg2)
```



- Pas de programmation
  - risque moindre

Arbre de dérivation vs. arbre de syntaxe abstraite

- ASA pas toujours seulement simplification de AD

- Élimination de ambiguïtés et de récursivité gauche peut priver de la grammaire désirée...

...et produire des AD très différents de ASA désiré

Grammaire des expressions avec récursivité à gauche

expression ⟶ différence

différence ⟶ différence '-' produit | produit

produit ⟶ '(' expression ')'

- Cette forme montre bien l'associativité à gauche de **'-'**

Grammaire des expressions sans récursivité à gauche

expression ⟶ différence

différence ⟶ produit différence'

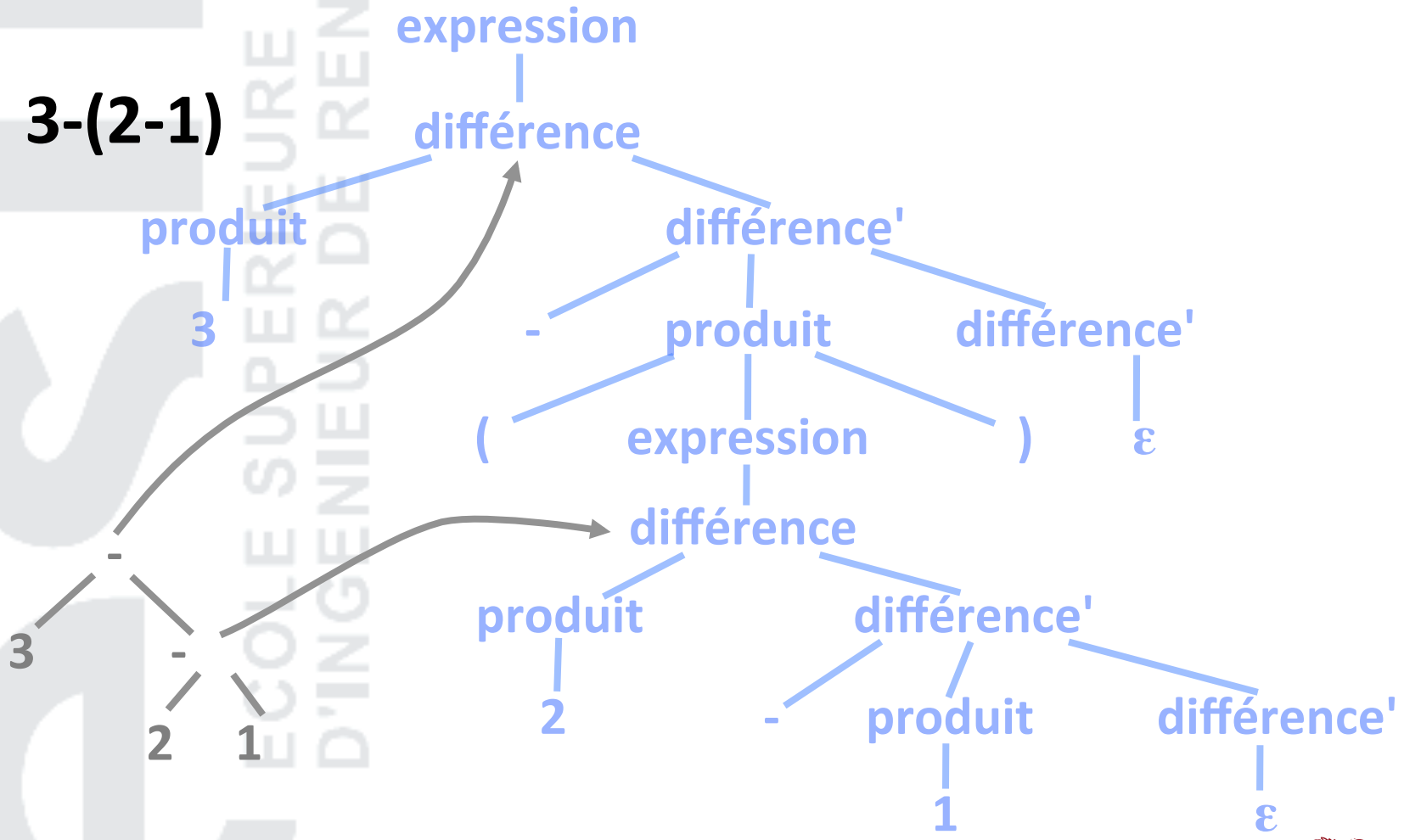
différence' ⟶ '-' produit différence' | ε

produit ⟶ '(' expression ')'

- Cette forme ne montre pas bien l'associativité à gauche de **'-'**

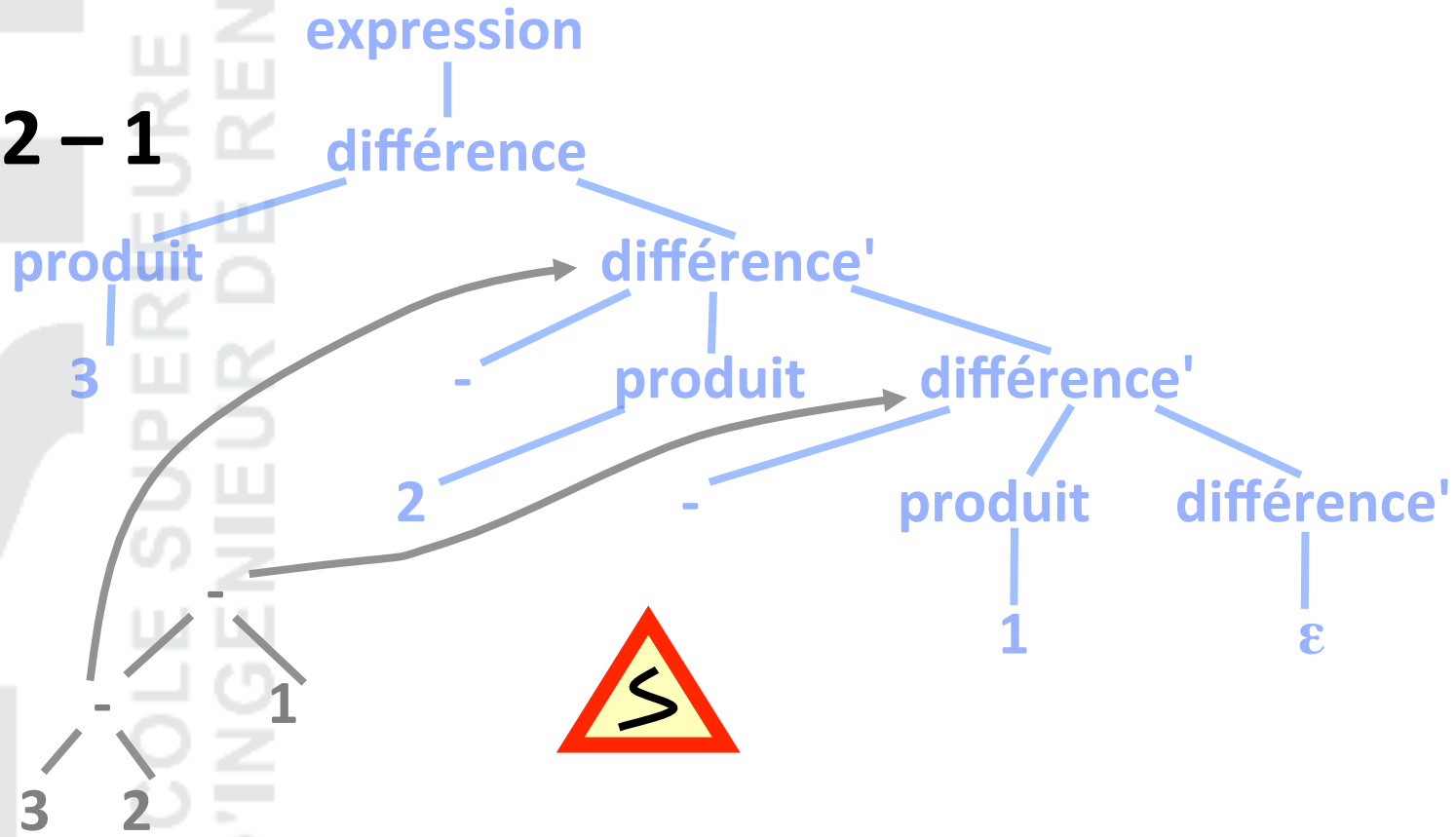
Exemple - syntaxe concrète

- **3-(2-1)**



Exemple - syntaxe concrète

- **3 – 2 – 1**



Construction de l'arbre de syntaxe abstraite (3)

```
expr : diff → diff

diff : p=prod  d=diff'[$p.tree] → $d

diff'[p] : mp=minusprod[p]   d=diff'[$mp.tree]  → $d
         |  ε → {$p}

minusprod[p] : "-"  p'=prod  → ^(DIFF {$p} $p')

prod : "("  expr  ")" → expr
```

légende : **syntaxe concrète**, **syntaxe abstraite**

Conclusion

- Arbre désiré vs. arbre imposé
- Analyse syntaxique largement automatisée...

...mais production d'arbre de syntaxe abstraite non-automatisable

processus créatif

- Savoir-faire avec EBNF