AGP: Algori i rogra a ion

tanguy.risset@insa-lyon.fr
Lab CITI, INSA de Lyon
Version du July 22, 2016

Tanguy Risset

July 22, 2016

Langag s, Gra air s, Co ila rs

- Objectif de cette partie du cours: comprendre le mécanisme de compilation, le lien avec la notion de grammaire et savoir créer un parseur.
- A quoi est du le succès des ordinateurs?
  - d
  - A m d d v d mm
    - Langage de haut niveau
    - Compilateurs rapides, codes portables
- La notion de compilation n'est pas limitée aux langages de programmation.
- L'informaticien produit en permanence des programmes qui font passer les données d'une représentation à une autre.

Tabl of Con n s

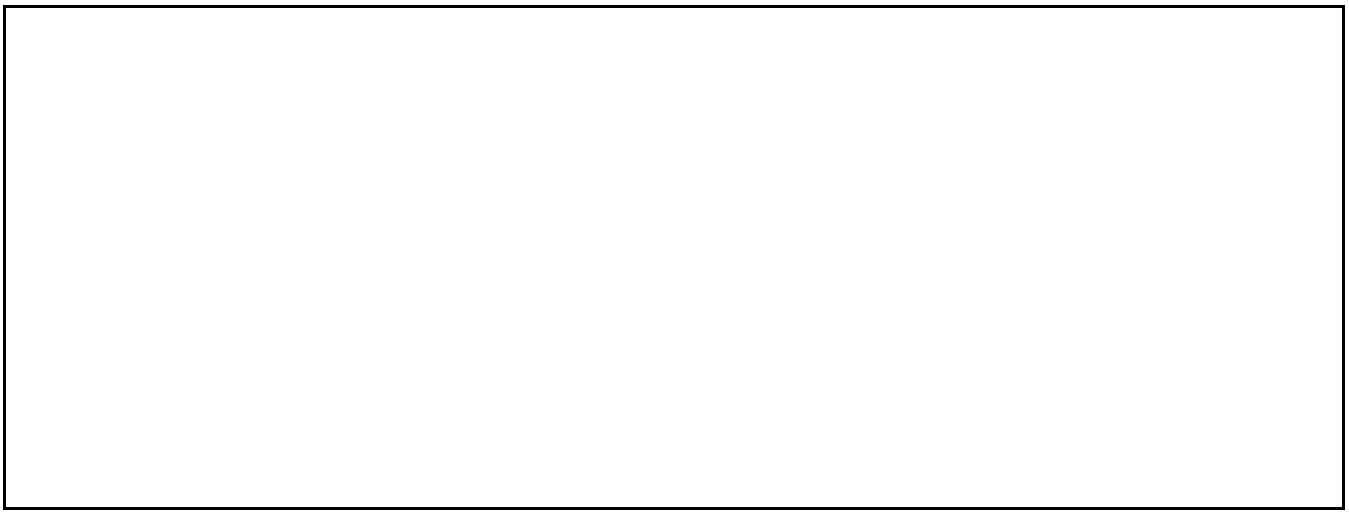
Yacc (bison)

- est l'implémention de par Vern Paxson et est la version de , dans la suite on parlera toujours de et pour désigner et
- signifie
- peut parser des flots de , il doit donc être utilisé avec un front-end qui transforme un flot de caractères en un flot de tokens (par exemple)

Pr i r x l si l : n i addi ionn r

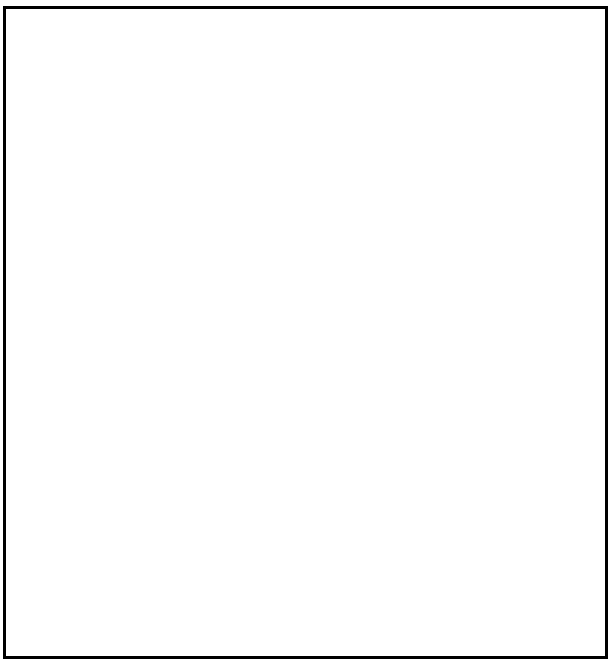
- On doit lire une suite d'addition et afficher le résultat.
  - d :
  - ffi d :
- Proposition de grammaire élémentaire:
  - G mm (S,T,N,),
  - S={ },
  - N={ }
  - T={ }
  - ={ }

Addi ionn r si l : fic r add r.l



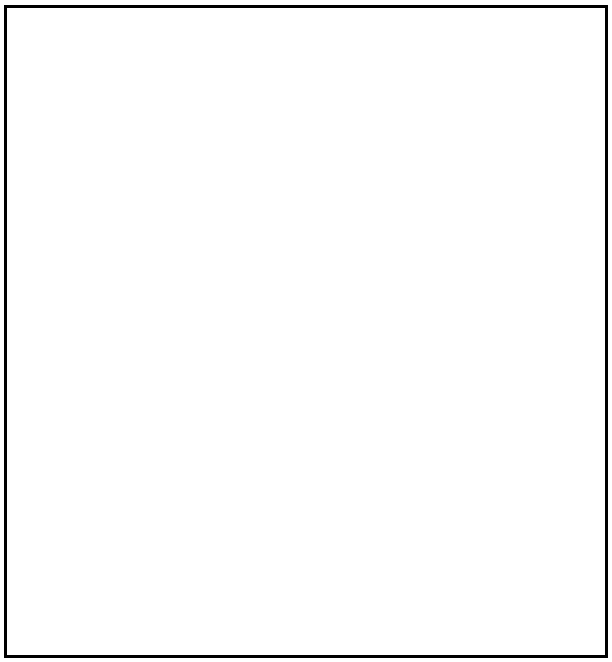
- sera généré par la commande , à inclure dans le fichier
- et sont des variables partagées par et : quand reconnaît un lexème, il le met dans la variable et le transmet à . Quand c'est une valeur, s'attend à le trouver dans

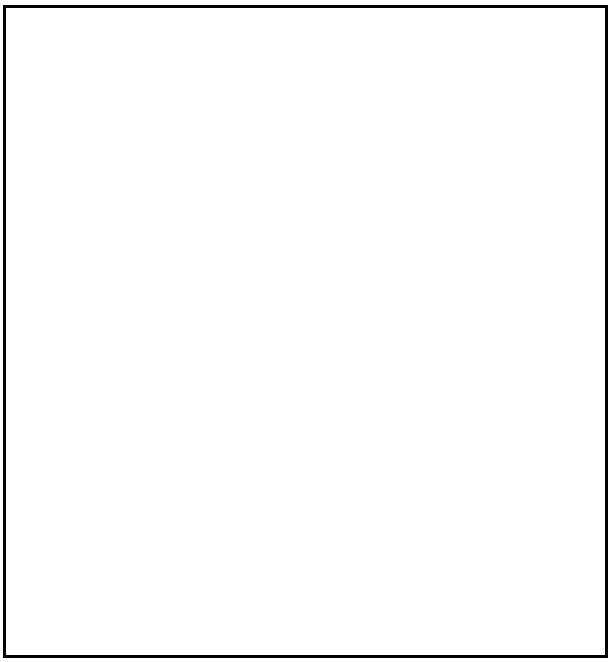
Addi ionn r si l : gra air ⇒ add r.y



- A chaque réduction de règles est associée une action
- chaque symbole renvoie un objet
- $1, $2,... correspondent aux objets renvoyés par la partie droite de la règles. $$ correspond à l'objet renvoyé par la partie gauche

Addi ionn r si l : fic r add r.y co l



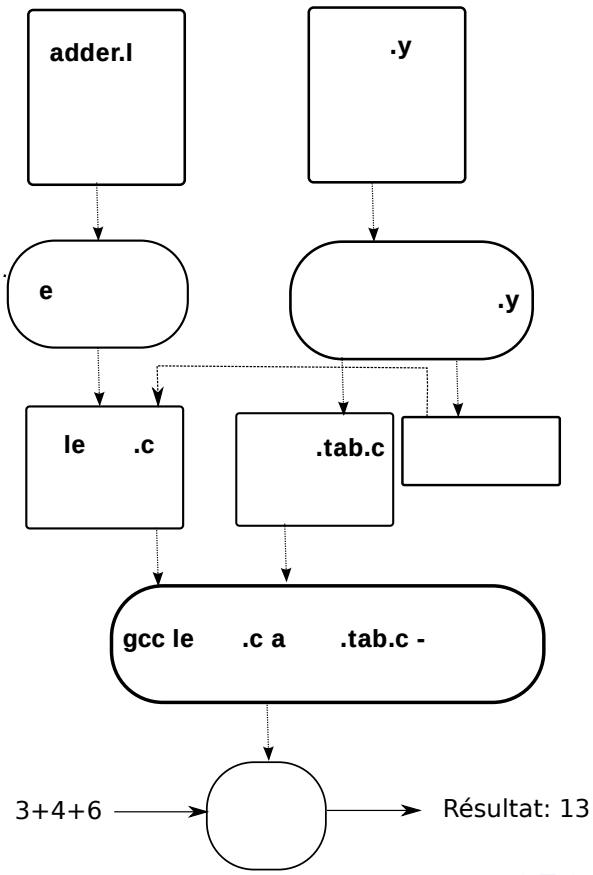


- A chaque réduction de règles est associé une action
- chaque symbole renvoie un objets
- $1, $2,... correspondent aux objets renvoyés par la partie droite de la règles. $$ correspond à l'objet renvoyé par la partie gauche

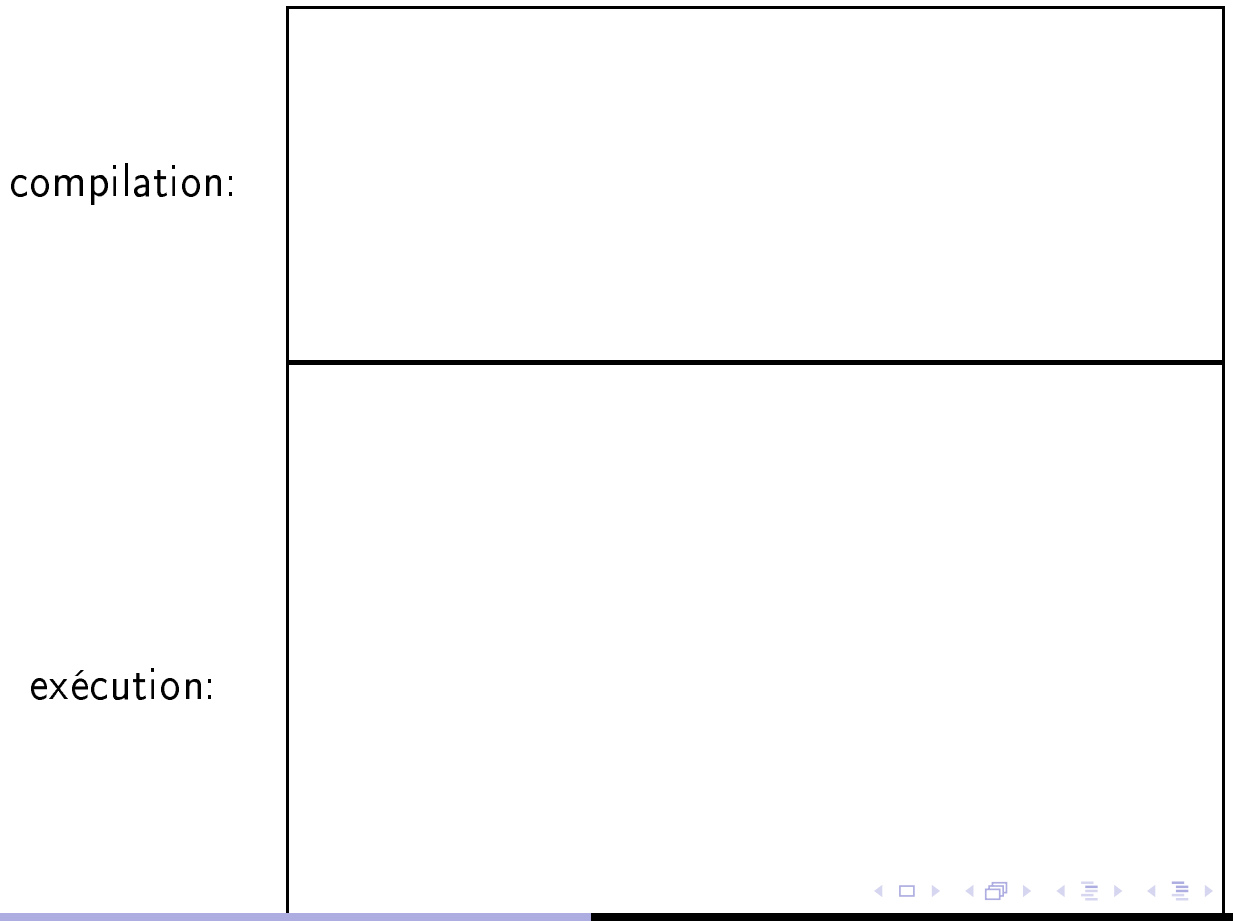
S r c r d' n fic i r yacc/bison

- Définitions:
  - C d C d d fi (%{ %})
  - O d mm d fi
  - d fi d k (y , v v m)
- Règles:
  - d fi d d mm v x d
- Code:
  - L d v j z, m m m mm

Add r: Proc ss s d co ila ion global



Co ila ion ilisa ion d rogra add r



L Langag TC-LOGO

- Le langage TC-LOGO (version 1.0) est hérité du formalisme LOGO ( )
  destiné à dessiner un graphique.
- Il sert à décrire le chemin que va suivre le crayon pour dessiner ce graphique.
- Il comporte 4 instructions:
  -
  -
  -
  - [ ]
- Il n'y a pas de point-virgule en TC-LOGO, les séparateurs valides sont l'espace, le retour chariot ou la tabulation

L Langag TC-LOGO

- Exemple de programme TC-LOGO:

- Grammaire (S,T,N,P),
- S={ }, N={ }
- T={ }
- et sont des lexèmes qui seront identifiés par l'analyse lexicale

Gra air ossibl o r TC-LOGO

P={

L x Yacc n rés é

- Outils pour produire des parseurs
- Utiles pour traiter les fichiers de données ou pour analyser des formats simples.
- Outils open source (gnu) extrêmement solides et portables (produisent du C travaillant sur les E/S standard).

Tabl of Con n s

Ex l d gra air a big ë

- Une grammaire est ambigue lorsque deux arbres de dérivations différents peuvent donner le meme mot du langage.
- Exemple: grammaire ambigue pour

1 → _ ___ ___
2 | _ ___
3 |
4 | ....

- Avec cette grammaire, le code

- peut être compris de deux manières:

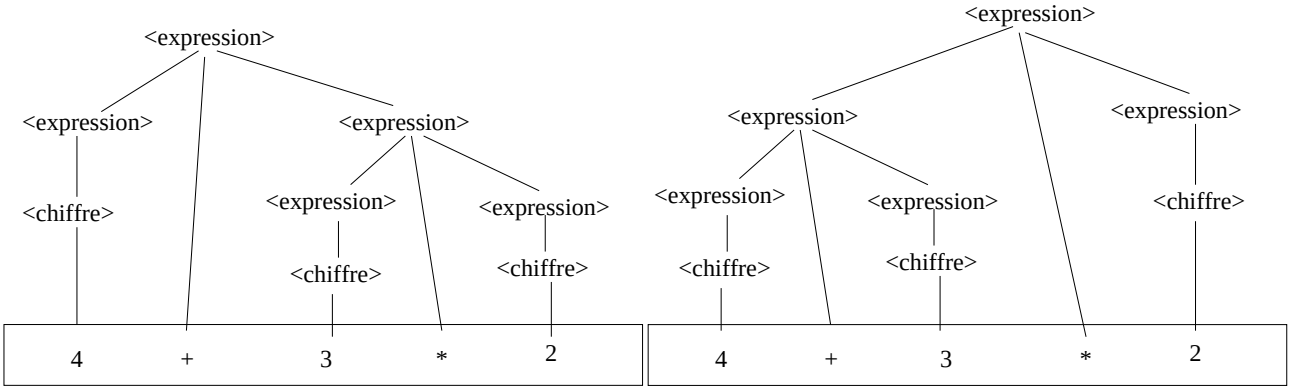
Sol ion: d sa big i isa ion (d sa big a ion ?)

- Il faut changer la grammaire

1 →
2 |
3 → _ ___ ___
4 |
5 | ....
6 → _ ___
7 | _ ___ ___
8 | ....

R o r s r la gra air d s x r ssion a big ë

- Soit la grammaire suivante:



- Deux dérivations sont équivalentes si elles ont le même arbre de dérivation (seul l'ordre dans lequel on a choisit les règles peut changer).

Sol ion: odifi r la gra air

- ... Sans modifier le langage:

- $\Rightarrow$ un seul arbre de dérivation possible.
- On peut faire d'autres améliorations pour l'associativité des opérateurs et les parenthèses. Une grammaire couramment utilisée pour les expressions arithmétiques est la suivante:

S ns d la réc rsion

- On a souvent le choix du sens de la récursion:

  $\left\{ \begin{array}{l} \rightarrow \_\_\_\_ \\ \rightarrow \_\_\_\_ \end{array} \right\}$ $\left\{ \begin{array}{l} \rightarrow \quad \_\_\_\_ \\ \rightarrow \_\_\_\_ \end{array} \right\}$

  à droite ou gauche

- La récursion à gauche est préférable pour des raisons de performances (taille de la pile générée pendant le parsing)
- Mais attention, cela influe sur l'associativité implicite de l'opérateur: si on choisi la récursion à gauche, sera interprété comme (associatif à gauche).
- La plupart des opérateurs sont associtatifs à gauche.
- Contre exemple: l'affectation en C. $\Leftrightarrow$