

Cairn: Trouver son chemin dans Menhir

Vincent Penelle

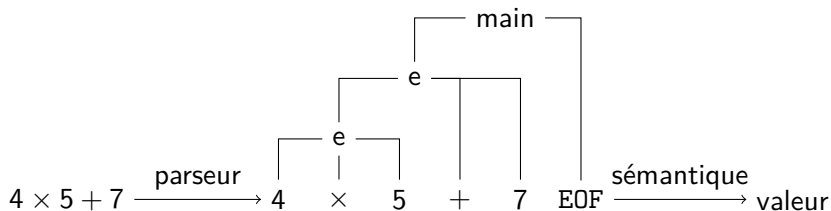
LaBRI, Université de Bordeaux

27 janvier 2026

Rappels analyse syntaxique

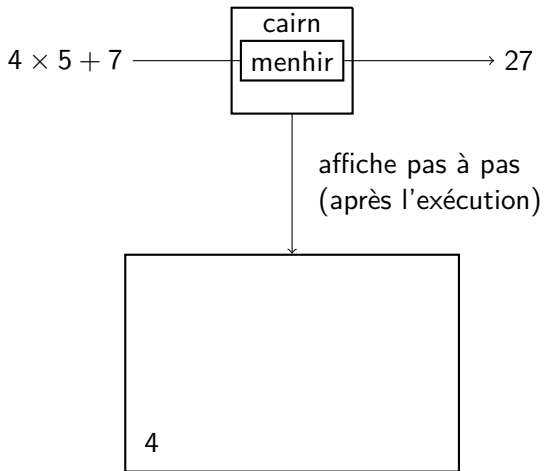
Grammaire algébrique :

- $\text{main} \rightarrow e \text{ EOF}$
- $e \rightarrow n \in \mathbb{N} \mid e + e \mid e \times e$

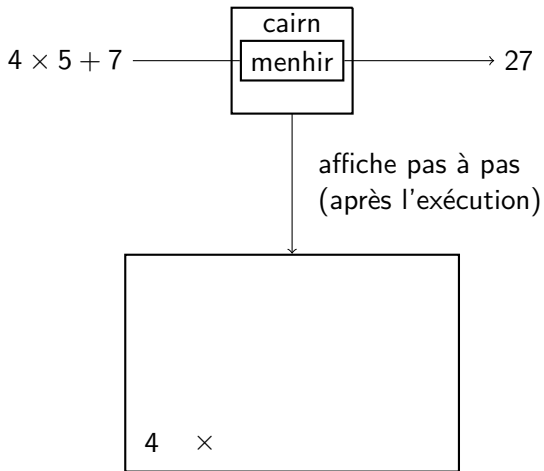


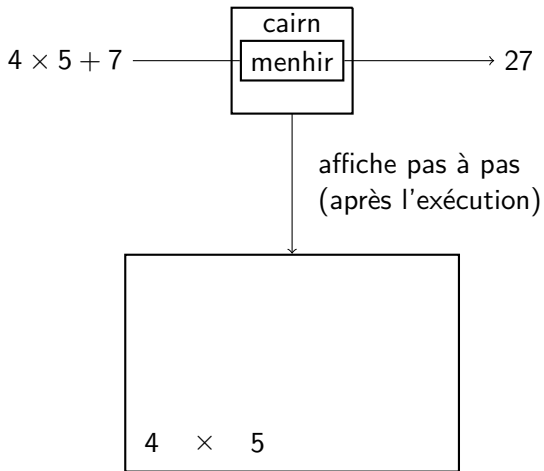
Plusieurs algos (CYK, LL, LR, GLR, etc), différentes complexités, grammaires acceptées et rapport à ambiguïté.

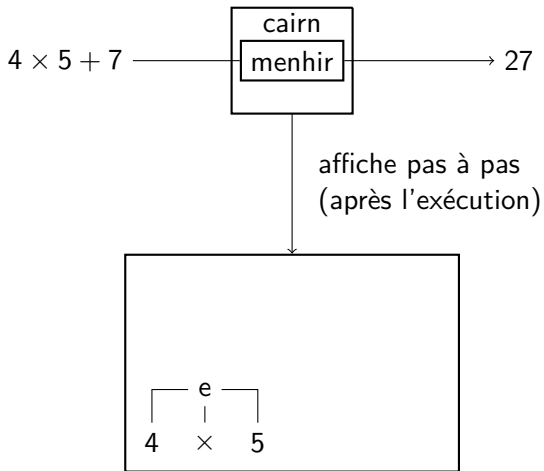
$$4 \times 5 + 7 \longrightarrow \boxed{\text{menhir}} \longrightarrow 27$$



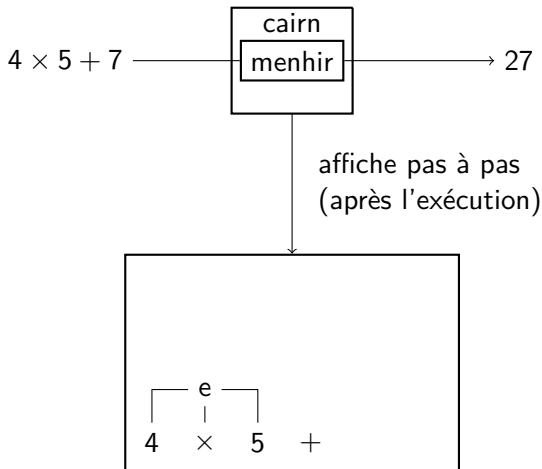
Menhir et Cairn



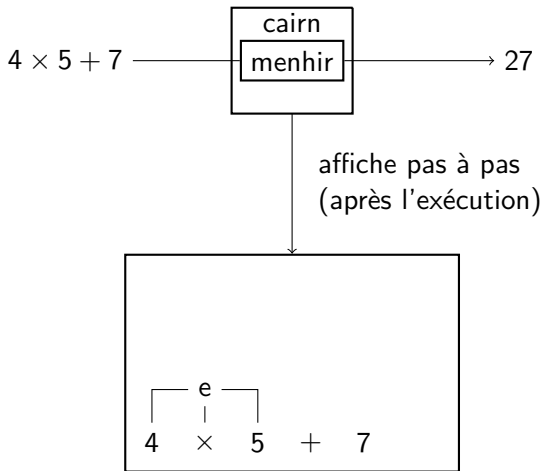




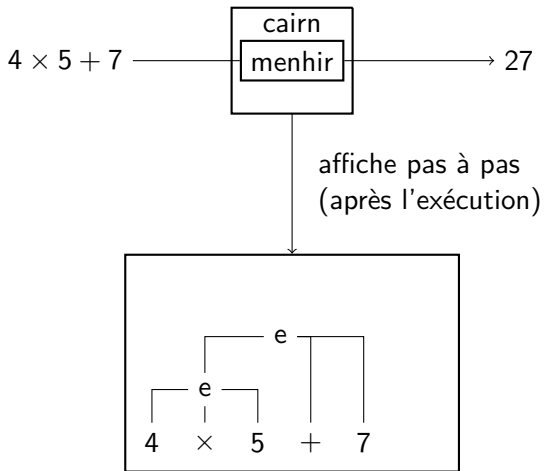
Menhir et Cairn



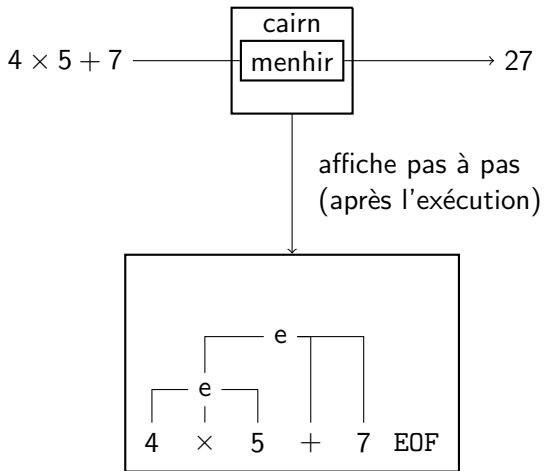
Menhir et Cairn



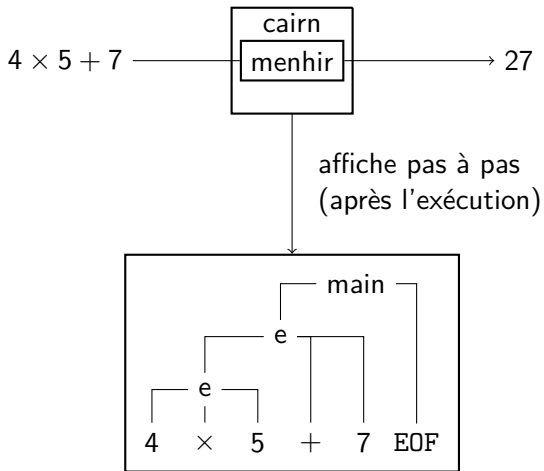
Menhir et Cairn



Menhir et Cairn



Menhir et Cairn



Deux modules (1000 lignes au total) :

- ParserLog :
 - Définit le type pour stocker le log de l'exécution du parseur
 - Gère l'affichage (avec LambdaTerm)
 - Indépendant de Menhir
- Parsing définit un foncteur dont le résultat :
 - Exécute le parseur pas à pas (API Incrémentale)
 - Produit le log (API Cmly + messages d'erreurs)
 - Fournit les fonctions qui remplacent le point d'entrée du parseur

- Absence de fonction de type `token` -> `int` visible dans l'API incrémentale :
 - token du lookahead non récupérable
 - limite l'utilisation d'attributs pour les tokens
- Affichage améliorable
- Plante si erreur du lexeur
- Pas d'affichage ou d'interactivité en cours d'exécution du parseur
- Pas de notification des conflits

Comparaison avec d'autres outils de visualisation

Il existe d'autres outils d'enseignement similaires (Yacv, PAVT, etc), mais qui réimplémentent les algos de parsing.

Avantages de Cairn :

- Comportement visualisé identique au parseur généré, par construction
- Utilisation du même fichier de grammaire que Menhir
- Intégration possible dans un programme qui utilise le résultat de l'analyse
- Surcoût en code faible (quelques lignes) qui peut être caché à l'étudiant · e.

Inconvénient : un seul algorithme visualisé (celui de Menhir).

Utilisé lors des TDs sur l'analyse syntaxiques avec deux buts :

- Comprendre des parseurs existants ($a^n b^n$ et langage de Dyck) : visualisation d'exécutions et comparaison avec l'exécution théorique d'un automate LR
- Aider à produire un fichier .mly (grammaires d'expressions, puis le langage fil rouge)

Intégré dans le projet en travail autonome

Effets observés sur les étudiant · e · s :

- Plus de tests que dans les autres parties du cours/autres UEs où j'ai enseigné
- Une bonne partie savent donner des arbres de dérivation à l'examen (avec une représentation proche de celle de Cairn)
- Produire une grammaire reste difficile pour la plupart

```
opam install cairn
```