

# Dahu

Jonathan Aceituno    Johan Jegard    Julien Virey

Université de Bourgogne, Algorithmes et complexité

January 21, 2010



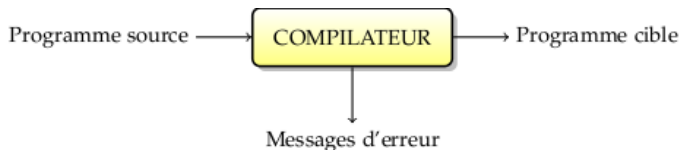
# Sommaire

- 1 Introduction
  - Contexte
  - But du projet
- 2 Analyse syntaxique
  - Algorithme
  - Tables d'analyse
  - Arbre syntaxique
- 3 Le programme DAHU
  - Gestion des données
  - Compilation & installation
  - Utilisation

# Contexte

## Les compilateurs

Le premier fut écrit par Grace Hopper. Un compilateur traduit un langage source en langage cible.



## La compilation

Deux étapes essentielles à la compilation :

- L'analyse lexicale, découpe le texte entré en lexèmes.
- L'analyse syntaxique, qui est l'objet de notre étude.

# But du projet

- Réécriture d'un programme d'analyse syntaxique en ocaml.
- Les grammaires sont décrites dans des fichiers.
- Le programme doit prendre en entrée une expression.
- Le résultat doit être la génération de l'arbre syntaxique correspondant.

# algorithme

## Analyse ascendante

- Prend en entrée :
  - le texte à analyser.
  - la table d'analyse de la grammaire.
- Analyse de bas en haut.
- Association des lexèmes aux règles jusqu'à l'axiome.
- Construction d'un arbre syntaxique interprétable par DOT.

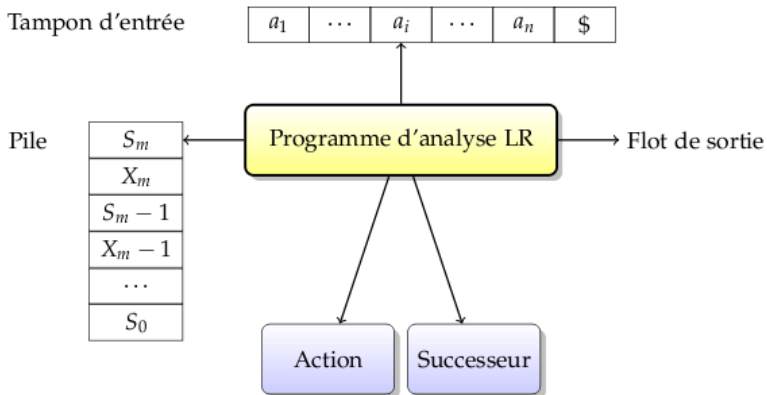
## Procédure

- Pile initialisée à 0.
- États ajoutés en fonction de la table d'analyse.
  - Soit on décale et empile l'état.
  - Soit on réduit et empile la valeur du successeur.



```
initialiser le pointeur p sur le premier symbole
begin boucler indéfiniment
    soit s le sommet de pile et a le symbole pointé par p;
    si Action[s,a]=décaler s' alors begin
        empiler s';
        avancer p sur le prochain symbole en entrée;
    end
    sinon si Action[s,a]=réduire par  $V \rightarrow w$  alors begin
        dépiler le dernier état;
        soit s' le nouvel état en sommet de pile;
        empiler Successeur[s',V];
    end
    sinon si Action[s,a]=accepter alors
        retourner>(*la grammaire est reconnue *)
    sinon Erreur();
end
end
```

# Fonctionnement



# Construction des tables d'analyse

**Soit la grammaire :**



# Construction des tables d'analyse

**Soit la grammaire :**

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

# Construction des tables d'analyse

**Soit la grammaire :**

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

**Elle est augmentée :**

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

# Construction des tables d'analyse

**Soit la grammaire :**

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

**Elle est augmentée :**

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

**On cherche ensuite l'ensemble  
des items.**

# Construction des tables d'analyse

**Soit la grammaire :**

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

**Elle est augmentée :**

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

**Elle engendre donc :**

# Construction des tables d'analyse

**Soit la grammaire :**

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

**Elle est augmentée :**

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

**Elle engendre donc :**

$$I_0 : E' \rightarrow \bullet E$$

$$E \rightarrow \bullet E a$$

$$E \rightarrow \bullet a$$

# Construction des tables d'analyse

**Soit la grammaire :**

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

**Elle est augmentée :**

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

**Elle engendre donc :**

$$I_0 : E' \rightarrow \bullet E$$

$$E \rightarrow \bullet E a$$

$$E \rightarrow \bullet a$$

$$I_1 : E' \rightarrow E \bullet$$

$$E \rightarrow E \bullet a$$

# Construction des tables d'analyse

**Soit la grammaire :**

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

**Elle est augmentée :**

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

**Elle engendre donc :**

$$I_0 : E' \rightarrow \bullet E$$

$$E \rightarrow \bullet E a$$

$$E \rightarrow \bullet a$$

$$I_1 : E' \rightarrow E \bullet$$

$$E \rightarrow E \bullet a$$

$$I_2 : E \rightarrow a \bullet$$

# Construction des tables d'analyse

**Soit la grammaire :**

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

**Elle est augmentée :**

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

**Elle engendre donc :**

$$I_0 : E' \rightarrow \bullet E$$

$$E \rightarrow \bullet E a$$

$$E \rightarrow \bullet a$$

$$I_1 : E' \rightarrow E \bullet$$

$$E \rightarrow E \bullet a$$

$$I_2 : E \rightarrow a \bullet$$

$$I_3 : E \rightarrow E a \bullet$$



# Construction des tables d'analyse

**Soit la grammaire :**

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

**Elle est augmentée :**

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

**Elle engendre donc :**

$$I_0 : E' \rightarrow \bullet E$$

$$E \rightarrow \bullet E a$$

$$E \rightarrow \bullet a$$

$$I_1 : E' \rightarrow E \bullet$$

$$E \rightarrow E \bullet a$$

$$I_2 : E \rightarrow a \bullet$$

$$I_3 : E \rightarrow E a \bullet$$

Voici la table d'analyse.

État	Actions		Successeurs
	$a$	$\$$	
0	$d2$		1
1	$d3$	ok	
2	$r1$	$r1$	
3	$r0$	$r0$	

# Construction des tables d'analyse

Soit la grammaire :

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

Elle est augmentée :

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

Elle engendre donc :

$$I_0 : E' \rightarrow \bullet E$$

$$E \rightarrow \bullet E a$$

$$E \rightarrow \bullet a$$

$$I_1 : E' \rightarrow E \bullet$$

$$E \rightarrow E \bullet a$$

$$I_2 : E \rightarrow a \bullet$$

$$I_3 : E \rightarrow E a \bullet$$

État	Actions		Successeurs
	$a$	$\$$	
0	$d2$		1
1	$d3$	ok	
2	$r1$	$r1$	
3	$r0$	$r0$	

- Les non-terminaux sont copiés dans successeurs.
- Les terminaux sont copiés dans actions en tant que décalage.
- Réduction :  $A \rightarrow w \bullet$ .

# Construction des tables d'analyse

Soit la grammaire :

$$(0) E \rightarrow E a$$

$$(1) E \rightarrow a$$

Elle est augmentée :

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E a$$

$$(2) E \rightarrow a$$

Elle engendre donc :

$$I_0 : E' \rightarrow \bullet E$$

$$E \rightarrow \bullet E a$$

$$E \rightarrow \bullet a$$

$$I_1 : E' \rightarrow E \bullet$$

$$E \rightarrow E \bullet a$$

$$I_2 : E \rightarrow a \bullet$$

$$I_3 : E \rightarrow E a \bullet$$

État	Actions		Successeurs
	<i>a</i>	<i>\$</i>	
0	<i>d2</i>		1
1	<i>d3</i>	<i>ok</i>	
2	<i>r1</i>	<i>r1</i>	
3	<i>r0</i>	<i>r0</i>	

- *dn* indique que l'état suivant est *n*.
- *rm* indique qu'il faut faire une réduction par la règle *m*.
- *ok* indique que l'analyse est ok.

# Construction de l'arbre syntaxique

- On commence avec une liste d'arbre vide.
- L'action *décaler* rajoute un arbre composé du symbole lu à la liste.
- L'action *réduire* réunit  $n$  arbres en un seul arbre.
- L'arbre résultant est imprimé sur la sortie standard en syntaxe DOT.

# Philosophie

## Développement \*nix

Nous suivons la philosophie de fonctionnement des programmes en ligne de commande sous Unix.

- Saisie de l'expression en argument du programme
- Saisie sur l'entrée standard en mode interactif
- Sortie pouvant être rejetée sur l'entrée de dot.

## En entrée

### Fichier grammaire

- Le fichier de grammaire est lu.
- Chaque jeton devient un **symbole**.
- Chaque ligne devient une **production**.
- Une grammaire est donc une liste de **production**.

```
type symbole = Epsilon | Dollar | S of string;;  
type production = symbole * (symbole list);;  
type grammaire = production list;;
```

# Compilation

Le fichier que nous vous proposons est une archive compressée avec bzip2. Il faut donc l'extraire, puis compiler le programme avant de l'installer :

- \$ tar xavf dahu.tar.bz2
- \$ make
- \$ su -
- \$ make install

Le programme est alors installé dans le système, ainsi que sa page de manuel.

# Utilisation

Le programme dahu permet la création d'un arbre syntaxique à partir d'une grammaire et d'un langage associé.

## En mode saisie

```
dahu [options] fichier_grammaire [token [token1] [...]] | dot -Tgtk
```

## En mode interactif

```
dahu [options] fichier_grammaire | dot -Tgtk
```

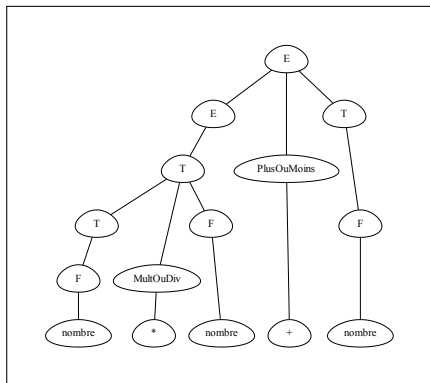
Le langage à analyser doit ensuite être saisi sur l'entrée standard. Le mot clef QUIT ou la combinaison de touche *CTRL + D* permettent de sortir du mode interactif.



## Exemples d'utilisation

En mode saisie

dahu arithmetique 'nombre \*  
nombre + nombre' — dot -Tgtk



# Exemples d'utilisation

En mode interactif

dahu arithmetique — dot -Tgtk

```
barbuk ~$  
--> dahu arithmetique | dot -Tgtk  
nombre  
*  
nombre  
+  
nombre
```