Déterminiser l'automate caractéristique

L'automate caractéristique :

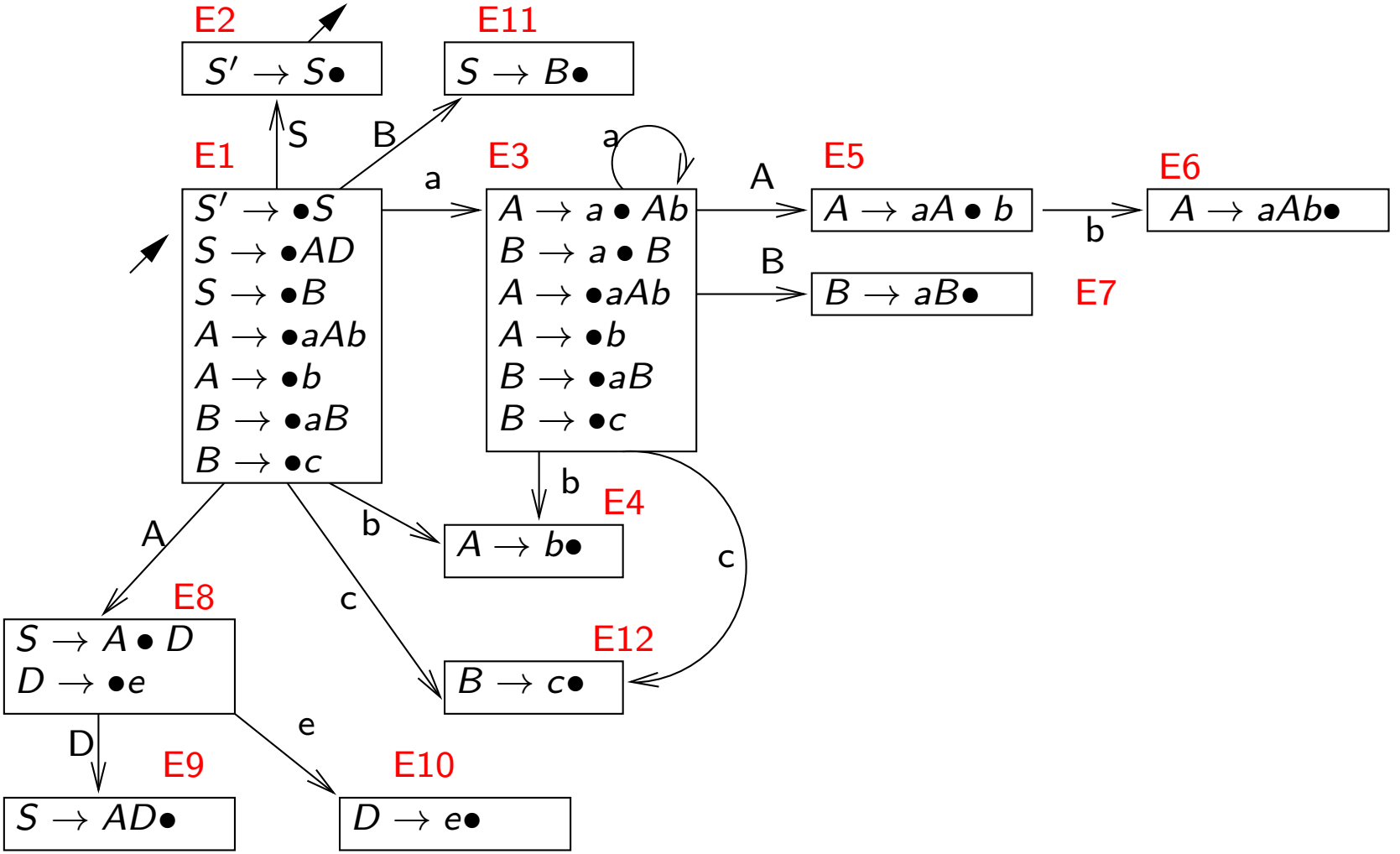
- est non déterministe (des ϵ-transitions) ;
- contient des expansions (justement les ϵ-transitions).

On veut un analyseur ascendant :

- déterministe ;
- sans expansions explicites (lectures et réductions).

$\Rightarrow$ on déterminise l'automate caractéristique.
$\Rightarrow$ on obtient un automate dit LR-AFD.

Automate LR-AFD, exemple



Automate LR(0)

L'automate LR-AFD décrit un automate à pile déterministe appelé automate LR(0) effectuant 2 types d'actions :

- lecture
- réduction

Dans un état contenant $X \to \cdots \bullet a \ldots$: Lecture de a

Dans un état contenant $X \to \alpha\bullet$: Réduction par $X \to \alpha\bullet$

La pile permet de mémoriser les états parcourus lors des lectures et des réductions.

Définition de l'automate LR(0)

Un état est un ensemble d'item : si Q est l'ensemble des états

$$Q \subseteq \mathcal{P}(It_G)$$

L'alphabet de pile est Q.

L'état initial q_0 :

- contient l'item initial de la forme $[S' \to \bullet S]$;
- sert à initialiser la pile.

L'état final q_f contient l'item final, de la forme $[S' \to S\bullet]$.

Définition de l'automate LR(0) : relation de transition

On note δ la relation de transition de l'AF LR-AFD.

$\delta(q, X) = q'$ signifie :

- si l'état courant est q ;
- et que $X \in V_T \cup V_N$ est le symbole courant à traiter ;
- alors l'état courant devient q'.

Exemple :

- $\delta(E_1, S) = E_2$ ou $E_1 \xrightarrow{S} E_2$
- $\delta(E_1, b) = E_4$ ou $E_1 \xrightarrow{b} E_2$

Définition de l'automate LR(0) : relation de transition

Lecture

Relation de transition de l'automate LR(0) pour une lecture :

$$(q, a) \rightarrow q\delta(q, a)$$

- si q est en sommet de pile
- si a est sous la tête de lecture
- et l'un des items de q est de la forme $[X \rightarrow \cdots \bullet a \ldots]$;
- alors on empile l'état successeur de q pour a dans δ.

Définition de l'automate LR(0) : relation de transition

Réduction

Relation de transition de l'automate LR(0) pour une réduction :

$$(qq_1 \dots q_n, \epsilon) \rightarrow q\delta(q, X)$$

- si q_n est en sommet de pile ;
- si l'un des items de q_n est de la forme $[X \rightarrow \alpha\bullet]$, $\mid \alpha \mid = n$;
- alors on dépile n états ;
- puis on empile $\delta(q, X)$ le successeur par X de l'état q en sommet de pile.

Et les expansions ?

Les ϵ-transition d'expansion ont disparu avec la déterminisation.

Elles se font implicitement à l'intérieur des états.

E1
$[S' \to \bullet S]$
$[S \to \bullet AD]$
$[S \to \bullet B]$
$[A \to \bullet aAb]$
$[A \to \bullet b]$
$[B \to \bullet aB]$
$[B \to \bullet c]$

$\xrightarrow{c}$ E12

lecture de c possible après expansions successives par :

- $S' \to S \rightsquigarrow [S' \to \bullet S] \in E1$
- $S \to B \rightsquigarrow [S \to \bullet B] \in E1$
- $B \to c \rightsquigarrow [B \to \bullet c] \in E1$

Construction de LR-AFD - en première approche

On construit Q (les états) et δ (les transitions) de l'automate caractéristique à partir de la grammaire.

On le déterminise, on obtient LR-AFD.

En fait, on peut construire directement LR-AFD (ouf !).

Construction algorithmique directe de LR-AFD

Principe :

- on sature les états par expansion ;
- on transite sur chaque symbole Y tel que $[\cdots \rightarrow \cdots \bullet Y \ldots]$.

Saturation des états par expansion

Un ensemble d'items E est saturé si :

- pour tout item $[X \to \alpha \bullet Y\beta]$ de E, $Y \in V_N$;
- pour toute production $Y \to \gamma$ de G de membre gauche Y ;
- l'item $[Y \to \bullet\gamma]$ appartient aussi à E.

On en déduit la fonction `Saturation` pour une grammaire G :

```
fonction Saturation (s : ensemble d'items) :
                                    ensemble d'items
// calcule tous les items d'un état de LR-AFD(G)
// contenant au moins les items de s
// retourne cet état.
```

Algorithme de construction de Q et δ

L'état initial est $\texttt{Saturation}([S' \to \bullet S])$.

Ensuite, pour chaque état saturé E et chaque symbole $Y \in V_T \cup V_N$ (lecture pour V_T, réduction pour V_N) :

- si E contient un ensemble de n items de la forme $\ll \bullet Y \gg$:

$$\{\ [X \to \alpha_i \bullet Y \beta_i] \mid 1 \leq i \leq n\}$$

- alors on calcule

$$E' = \texttt{Saturation}(\{[X \to \alpha_i Y \bullet \beta_i] \mid 1 \leq i \leq n\}\ )$$

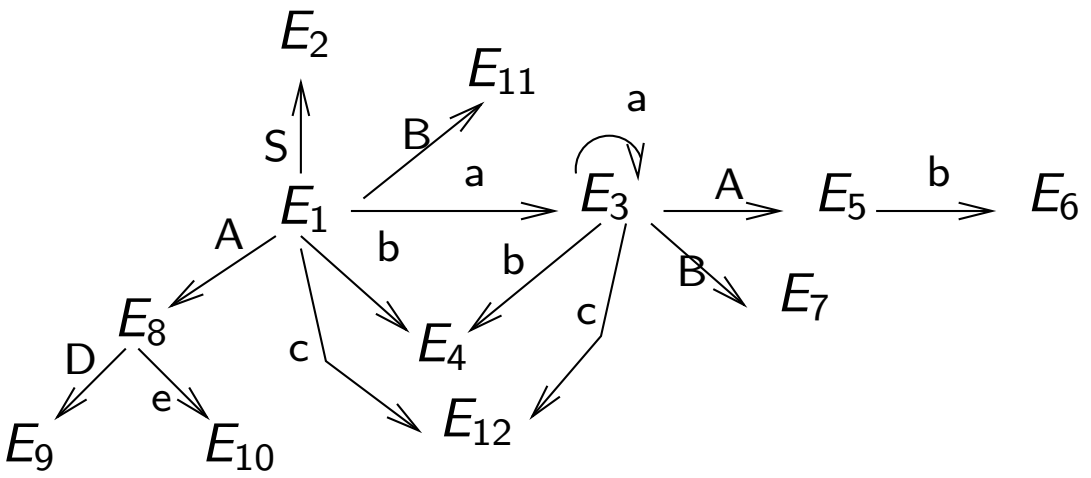
- si cet état E' n'existe pas, on l'ajoute à Q ;
- et on définit $\delta(E, Y) = E'$.

Exemple et remarque

Pour ne pas manquer de place sur sa feuille : séparer le contenu des états et la relation de transition.

E1
$[S' \to \bullet S]$
$[S \to \bullet AD]$
$[S \to \bullet B]$
$[A \to \bullet aAb]$
$[A \to \bullet b]$
$[B \to \bullet aB]$
$[B \to \bullet c]$

E3
$[A \to a \bullet Ab]$
$[B \to a \bullet B]$
$[A \to \bullet aAb]$
$[A \to \bullet b]$
$[B \to \bullet aB]$
$[B \to \bullet c]$



Conflits LR(0), grammaire LR(0)

L'automate LR(0) construit peut ne pas être déterministe (2 cas).

État autorisant 2 réductions (ou plus) :

conflit LR(0) reduce/reduce Ex : $\boxed{\begin{array}{l}[A \to b\bullet] \\ [B \to b\bullet]\end{array}}$

État autorisant 1 réduction et 1 lecture (ou plus) :

conflit LR(0) shift/reduce Ex : $\boxed{\begin{array}{l}[A \to \bullet b] \\ [B \to c\bullet]\end{array}}$

Conflits LR(0), grammaire LR(0)

Une grammaire est dite LR(0) si aucun de ses états ne contient de conflit LR(0) :

- ni shift-reduce
- ni reduce-reduce

Les conflits shift/shift n'existent pas (aucun sens).