4
Analyse syntaxique.

1 – Généralités.

Soit $G = (\mathcal{V}, \mathcal{A}, R)$ une grammaire et soit $S \in \mathcal{V}$ une variable choisie comme *axiome*.

Une analyse syntaxique par G d'une "phrase" $u \in \mathcal{A}^*$ est un algorithme qui doit décider si $u \in \mathcal{L}(G, S)$ et :

- dans le cas d'une réponse positive, décrire une dérivation $S \overset{k}{\underset{G}{\Longrightarrow}} u$,
- dans le cas contraire, déterminer pourquoi une dérivation ne peut être construite, c'est–à–dire, faire un diagnostic sur l'"erreur" qui est la cause de cet échec.

Les algorithmes d'analyse syntaxique qui nous intéressent ont quelques points communs :

- Ils lisent le mot à analyser de la gauche vers la droite (ceci justifie la première lettre, un L, des sigles angloricains qui servent à les désigner).
- Ils choisissent, parmi les dérivations équivalentes possibles d'un même mot, une dérivation particulière : soit *à droite* soit *à gauche*.
  - dans le cas de dérivations à gauche (Leftmost), la construction s'effectue de l'axiome vers le mot à analyser : on parle d'analyse "descendante" ou "prédictive",
  - dans le cas de dérivations à droite (Rightmost), la construction s'effectue du mot à analyser vers l'axiome : on parle d'analyse "ascendante".
- Ils sont basés sur l'utilisation d'"automates à pile" (dont la définition générale ne sera pas donnée ici) : le résultat d'une analyse réussie est une suite de règles qui définit une dérivation $S \overset{k}{\underset{G}{\Longrightarrow}} u$.
- Ils sont déterministes.

Ils ne peuvent évidemment pas s'appliquer avec le même succès à toute grammaire, en particulier, les grammaires "ambiguës" que nous allons maintenant définir, sont *a priori* hors du domaine d'application de ces algorithmes.

Soient $X \in \mathcal{V}$ et $\alpha \in (\mathcal{A} + \mathcal{V})^*$: lorsqu'il existe une dérivation $X \overset{k}{\underset{G}{\Longrightarrow}} \alpha$ dans G, il en existe généralement beaucoup d'autres ! Cependant, on sait que deux dérivations équivalentes font le même calcul, c'est–à–dire, ont le même arbre.

On est conduit à dire qu'*une grammaire n'est pas ambiguë ssi, pour toute* $X \in \mathcal{V}$ *et tout* $\alpha \in (\mathcal{A} + \mathcal{V})^*$, *deux dérivations* $d : X \overset{k}{\underset{G}{\Longrightarrow}} \alpha$ *et* $d' : X \overset{k'}{\underset{G}{\Longrightarrow}} \alpha$ *sont équivalentes.*

Le déterminisme des algorithmes d'analyse syntaxique que nous allons étudier implique que les grammaires auxquelles ils s'appliquent ne sont pas ambiguës. Dans la pratique, l'ambiguïté d'une grammaire provient de conventions d'écriture bien répertoriées, par exemple

- **Préséance** : dans une grammaire dont certaines constantes représentent des opérateurs, par exemple $\times$ pour un produit et $\oplus$ pour une somme, une expression de la forme $\alpha \oplus \beta \times \gamma$ devra généralement être analysée comme $\alpha \oplus (\beta \times \gamma)$ et non pas comme $(\alpha \oplus \beta) \times \gamma$: on dira alors que $\times$ *a un degré de préséance supérieur à celui de* $\oplus$.

– **Mode d'association** : de même, une expression de la forme $\alpha \oplus \beta \oplus \gamma$ devra généralement être analysée comme $(\alpha \oplus \beta) \oplus \gamma$ et non pas comme $\alpha \oplus (\beta \oplus \gamma)$: on dira alors que $\oplus$ *est associative à (partir de la) gauche*.

Ceci est évidemment l'objet d'une étude théorique (celle des grammaires d'opérateurs) mais nous n'en parlerons pas. La méthode utilisée ici, pour tenir compte de telles conventions, est plus pragmatique : elle consiste à sélectionner la dérivation voulue parmi toutes celles qui aboutissent à un mot donné (cf. exercices 6 et 7).

Exemple 3.1 (suite).

Notre grammaire habituelle (exemple 1 du chapitre 3) est ambiguë : elle ne peut pas avoir toutes les qualités ! En effet, les dérivations $\underline{S} \overset{1}{\Longrightarrow} \underline{S}bS \overset{1}{\Longrightarrow} SbSbS$ et $\underline{S} \overset{1}{\Longrightarrow} Sb\underline{S} \overset{1}{\Longrightarrow} SbSbS$ ne sont pas équivalentes.

Plus généralement, considérons une grammaire G disposant de règles (pas nécessairement distinctes) $X \longrightarrow X\alpha$ et $X \longrightarrow \alpha' X$ (on parle d'"appels récursifs" à gauche et à droite) alors G est ambiguë, à cause des deux dérivations non équivalentes :

$$\underline{X} \overset{1}{\Longrightarrow} \underline{X}\alpha \overset{1}{\Longrightarrow} \alpha' X\alpha \qquad \text{et} \qquad \underline{X} \overset{1}{\Longrightarrow} \alpha'\underline{X} \overset{1}{\Longrightarrow} \alpha' X\alpha.$$

Cette cause d'ambiguïté n'est pas très grave car on peut effectivement éliminer tout appel récursif à gauche sans pour cela changer les langages engendrés. Il y a des cas plus sérieux : il faut savoir qu'il existe des langages algébriques qui ne peuvent pas être engendrés par une grammaire non ambiguë ! (La vérification de cette affirmation dépasse nos compétences.)

Dans toute la suite, nous supposerons que G est *réduite* pour son axiome S ; ceci signifie que G ne comporte pas de variable inutile, plus précisément que toute $X \in \mathcal{V}$ est :

– *productive* : $\mathcal{L}(G, X) \neq \emptyset$.
– *accessible à partir de* S : il existe α et $\beta \in (\mathcal{A} + \mathcal{V})^*$ tels que $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha X \beta$.

Les grammaires avec axiome et réduites pour leur axiome, peuvent se définir par leurs seules règles, en appliquant les conventions suivantes :

– une variable est un symbole qui forme la partie gauche d'au moins une règle, l'axiome se présente en premier,
– une constante est un symbole, distinct de toute variable, apparaissant dans la partie droite d'au moins une règle (ε désigne le mot vide, comme d'habitude).

2 – Dérivations à droite et analyse ascendante.

Lorsque l'on séquentialise une arborescence, par exemple en utilisant une méthode descendante, on peut choisir systématiquement d'élaguer la racine qui est le plus à droite : toute dérivation est donc équivalente à une dérivation obtenue ainsi. Une telle dérivation applique une règle à l'occurrence de la variable qui est située le plus à droite possible (ceci explique l'expression angloricaine *rightmost derivation*).

Dérivations à droite

Une dérivation élémentaire est dite *à droite* lorsqu'elle se présente sous la forme

$$\pi X v \overset{1}{\underset{G}{\Longrightarrow}} \pi\alpha v$$

pour $v \in \mathcal{A}^*$.
Une dérivation est dite *à droite* lorsqu'elle est définie comme un enchaînement de dérivations élémentaires à droite.

Une dérivation à droite sera symbolisée avec $\beta \overset{k}{\underset{d,G}{\Longrightarrow}} \gamma$ et plus simplement par $\beta \overset{k}{\underset{d}{\Longrightarrow}} \gamma$ lorsque G est évidente.

Le cas des dérivations à gauche (*leftmost derivations*), facile à imaginer, sera envisagé dans l'annexe.

Observations.

• Dans une dérivation à droite, il n'est plus nécessaire de préciser l'occurrence de la variable à laquelle on applique une règle, puisque l'on a convenu une fois pour toutes que c'était la plus à droite !

• Soit $u \in \mathcal{L}(G,S)$, alors, si G n'est pas ambiguë, il existe exactement une dérivation à droite $S \underset{d}{\overset{k}{\Longrightarrow}} u$ (et réciproquement, on peut définir l'ambiguïté d'une grammaire sur la base des seules dérivations à droite).

Exemple 3.1 (suite).

Voici une dérivation à droite $S \underset{d}{\overset{11}{\Longrightarrow}} aabbabaabba$ dans notre grammaire habituelle :

$$\begin{aligned}
S &\underset{d}{\overset{1}{\Longrightarrow}} SbS \\
&\underset{d}{\overset{1}{\Longrightarrow}} SbSbS \\
&\underset{d}{\overset{1}{\Longrightarrow}} SbSba \\
&\underset{d}{\overset{1}{\Longrightarrow}} Sbba \\
&\underset{d}{\overset{1}{\Longrightarrow}} SbSbba \\
&\underset{d}{\overset{1}{\Longrightarrow}} SbSbSbba \\
&\underset{d}{\overset{1}{\Longrightarrow}} SbSbaabba \\
&\underset{d}{\overset{1}{\Longrightarrow}} Sbabaabba \\
&\underset{d}{\overset{1}{\Longrightarrow}} SbSbabaabba \\
&\underset{d}{\overset{1}{\Longrightarrow}} Sbbabaabba \\
&\underset{d}{\overset{1}{\Longrightarrow}} aabbabaabba
\end{aligned}$$

dont l'arbre associé est encore la figure 3 du chapitre 3.

En énumérant les règles de la façon suivante :

$$1 : S \longrightarrow SbS \qquad 2 : S \longrightarrow \varepsilon \qquad 3 : S \longrightarrow a \qquad 4 : S \longrightarrow aa$$

on obtient la suite $1, 1, 3, 2, 1, 1, 4, 3, 1, 2, 4$ avec laquelle on peut effectivement construire une dérivation à droite.

2.1 – Analyse ascendante.

Ce type d'analyse est désigné par le sigle LR : on lit le mot à analyser de gauche à droite (Left to right scanning) et on construit une dérivation à droite (Rightmost derivation). Il est important d'observer que cette méthode construit les dérivations "à l'envers" ; c'est pourquoi on parle d'analyse ascendante : on va du mot à analyser vers l'axiome !

Pour rendre ces méthodes efficaces, on se permet d'observer, dans la mesure du possible, les k symboles qui sont au début de la partie du mot restant à analyser ; on obtient alors les algorithmes de type $LR(k)$ qui sont "d'autant plus déterministes" que k est grand (comme il se doit, une grammaire peut fort bien résister à tous ces traitements !). Nous commencerons par $LR(0)$, c'est–à–dire par un algorithme aveugle puis, nous éprouverons le besoin d'étudier des algorithmes de type $LR(1)$, prévoyant 1 symbole ; contrairement aux précédents, ceux–ci sont très suffisants pour les besoins courants actuels en compilation.

2.2 – Configuration d'analyse partielle et préfixes viables.

L'analyse ascendante de $u \in \mathcal{A}^*$ est la construction d'une dérivation $S \Longrightarrow u$, qui détermine ses règles dans l'ordre inverse de leur enchaînement naturel : on remonte du mot u vers l'axiome S. On peut appeler "analyse partielle de u" une dérivation $\pi \underset{d}{\Longrightarrow} w$ telle qu'il existe une dérivation $S \underset{d}{\Longrightarrow} \pi v$ pour laquelle $wv = u$. La première question qui se pose est donc de trouver une condition utilisable, que doit nécessairement satisfaire un "préfixe" $\pi \in (\mathcal{A} + \mathcal{V})^*$ pour qu'il existe une dérivation à droite $S \underset{d}{\Longrightarrow} \pi v$: un tel préfixe sera dit "viable".

Forme générale des dérivations à droite.

Une dérivation à droite $X \underset{d}{\Longrightarrow} \gamma$ de longueur > 0 (la longueur n'est notée explicitement que lorsqu'elle est supposée être égale à 1) se décompose de la façon suivante :

$$X \overset{1}{\Longrightarrow} \alpha'\beta \underset{d}{\Longrightarrow} \alpha' v$$

où $X \longrightarrow \alpha'\beta$ est sa première règle et où $v \in \mathcal{A}^*$ (bien entendu, on a $\alpha' v = \gamma$).
Si le premier caractère de β est une variable, c'est–à–dire si $\beta = Y\alpha''$ pour $Y \in \mathcal{V}$, on peut préciser l'analyse précédente en décomposant de la façon suivante :

$$X \overset{1}{\Longrightarrow} \alpha' Y \alpha'' \underset{d}{\Longrightarrow} \alpha' Y v'' \underset{d}{\Longrightarrow} \alpha' \gamma v''$$

Nous dirons que cette dérivation est *la composée* de

$$X \overset{1}{\Longrightarrow} \alpha' Y \alpha'' \underset{d}{\Longrightarrow} \alpha' Y v'' \quad \text{et de} \quad Y \underset{d}{\Longrightarrow} \gamma$$

Un raisonnement par induction sur la longueur montre alors que toute dérivation à droite, non triviale, est une composée de dérivations des types précédents :

$$\begin{array}{lllll}
X_0 & \overset{1}{\Longrightarrow} & \alpha'_1 X_1 \alpha''_1 & \underset{d}{\Longrightarrow} & \alpha'_1 X_1 v_1 \qquad \text{pour } v_1 \in \mathcal{A}^* \\
X_1 & \overset{1}{\Longrightarrow} & \alpha'_2 X_2 \alpha''_2 & \underset{d}{\Longrightarrow} & \alpha'_2 X_2 v_2 \qquad \text{pour } v_2 \in \mathcal{A}^* \\
\dots & \dots & \dots & \dots & \dots \\
X_{n-1} & \overset{1}{\Longrightarrow} & \alpha'_n X_n \alpha''_n & \underset{d}{\Longrightarrow} & \alpha'_n X_n v_n \qquad \text{pour } v_n \in \mathcal{A}^* \\
X_n & \overset{1}{\Longrightarrow} & \alpha'_{n+1} \alpha''_{n+1} & \underset{d}{\Longrightarrow} & \alpha'_{n+1} v_{n+1} \qquad \text{pour } v_{n+1} \in \mathcal{A}^*
\end{array}$$

ce qui, dans le cas où $n = 0$ se présente sous la forme particulière suivante :

$$X_0 \overset{1}{\Longrightarrow} \alpha'_1 \alpha''_1 \underset{d}{\Longrightarrow} \alpha'_1 v_1 \qquad \text{pour } v_1 \in \mathcal{A}^*$$

Commentaires.

- La composée de ces dérivations a la forme

$$X_0 \underset{d}{\Longrightarrow} \pi X_n v \overset{1}{\Longrightarrow} \pi \alpha'_{n+1} \alpha''_{n+1} v \underset{d}{\Longrightarrow} \pi \alpha'_{n+1} v_{n+1} v$$

où $\pi = \alpha'_1 \dots \alpha'_n$ et $v = v_n \dots v_1$.

- Les mots $\pi\alpha'_{n+1}$ obtenus ainsi sont les "préfixes" que nous cherchons à caractériser : la définition exacte sera donnée plus bas.

- La dernière dérivation de la liste ci–dessus suppose seulement que α''_{n+1} est un suffixe du membre de gauche d'une règle pour X_n : une dérivation à droite admet donc généralement plusieurs décompositions.

• Un cas particulièrement intéressant est celui où $\alpha''_{n+1} = \varepsilon$: on a alors $v_{n+1} = \varepsilon$ et $X_n \overset{1}{\Longrightarrow} \alpha'_{n+1}$ est la dernière dérivation élémentaire, ce qui veut dire que la règle $X_n \longrightarrow \alpha'_{n+1}$ est la dernière de la liste définissant la dérivation qui nous intéresse.

• Ce qui vient d'être dit est valable pour toute dérivation non triviale : la dérivation triviale $S \overset{0}{\Longrightarrow} S$, qui joue un rôle dans notre algorithme, ne rentre donc pas dans ce cadre. Ceci justifie la notion d'"augmentation", purement technique, dont il va être question maintenant.

Augmentation d'une grammaire.

La désignation effective de l'axiome S, se fait habituellement en "augmentant" la grammaire de la façon suivante : on introduit une nouvelle variable $S' \notin \mathcal{V}$ et une seule nouvelle règle $S' \longrightarrow S$. Ainsi, *S' n'est–elle présente dans aucune autre règle que la sienne*, contrairement à S qui peut très bien être présente dans la partie droite de n'importe quelle règle, et S' est l'axiome de la grammaire ainsi construite. Nous préférons utiliser l'augmentation

$$\emptyset \longrightarrow S$$

produisant S à partir de l'ensemble vide lui–même (qui n'est pas un élément de $\mathcal{V}$) : l'axiome se trouve ainsi engendré à partir de "rien", comme il se doit.

Dans la grammaire ainsi augmentée, toute dérivation $S \overset{k}{\underset{d}{\Longrightarrow}} \gamma$ devient une dérivation $\emptyset \overset{k+1}{\underset{d}{\Longrightarrow}} \gamma$ de longueur > 0 qui commence toujours par $\emptyset \overset{1}{\Longrightarrow} S$.

Nous sommes maintenant en mesure de caractériser les "préfixes" qui nous intéressent, en anticipant sur une notation qui sera l'objet de la section suivante.

Préfixes viables

$\pi\alpha' \in (\mathcal{A} + \mathcal{V})^*$ est appelé un *préfixe viable* ssi il existe une dérivation à droite

$$\emptyset \underset{d}{\Longrightarrow} \pi X v \overset{1}{\Longrightarrow} \pi\alpha'\alpha'' v$$

(v est alors nécessairement un élément de $\mathcal{A}^*$).
La règle $X \longrightarrow \alpha'\alpha''$ définit alors un *item valide* pour ce préfixe viable, que l'on notera $\cdot X \longrightarrow \alpha' \cdot \alpha''$, où $\cdot$ est un nouveau symbole (cf. ci–dessous).

Ce que nous avons observé au sujet des dérivations à droite peut s'exprimer en disant que toute dérivation à droite détermine un enchaînement d'item, c'est–à–dire, avec les notations ci–dessus :
ou bien :

$$\begin{array}{rcl} \emptyset & \longrightarrow & \cdot S \\ \cdot S & \longrightarrow & \alpha'_1 \cdot X_1 \alpha''_1 \\ \cdot X_1 & \longrightarrow & \alpha'_2 \cdot X_2 \alpha''_2 \\ \cdots & & \cdots \\ \cdot X_{n-1} & \longrightarrow & \alpha'_n \cdot X_n \alpha''_n \\ \cdot X_n & \longrightarrow & \alpha'_{n+1} \cdot \alpha''_{n+1} \end{array}$$

ce qui, dans le cas où $n = 0$, se présente de la façon suivante :

$$\begin{array}{rcl} \emptyset & \longrightarrow & \cdot S \\ \cdot S & \longrightarrow & \alpha'_1 \cdot \alpha''_1 \end{array}$$

ou bien, le cas spécial

$$\emptyset \longrightarrow S\cdot$$

qui provient de l'augmentation de la grammaire.

Réciproquement, tout enchaînement d'item comme ci–dessus permet de construire des dérivations à droite assurant la viabilité d'un préfixe donné.

Cette remarque est importante pour la compréhension et la justification de la suite. Par exemple, la propriété importante

- *Tout préfixe (facteur gauche) d'un préfixe viable est un préfixe viable.*

s'observe très facilement en "raccourcissant" l'enchaînement qui définit le préfixe viable en question.

2.3 – Item $LR(0)$ d'une grammaire G.

Pour définir les item $LR(0)$, on introduit un pointeur, c'est–à–dire un nouveau symbole $\cdot \notin \mathcal{A}+\mathcal{V}$.

Les item $LR(0)$

Pour toute $X \in \mathcal{V} + \{\emptyset\}$, et tout couple $\alpha \in (\mathcal{A}+\mathcal{V})^*$ et $\beta \in (\mathcal{A}+\mathcal{V})^*$:

$\cdot X \longrightarrow \alpha \cdot \beta$ est un *item* $LR(0)$ de G ssi $X \longrightarrow \alpha\beta$ est une règle de G.

Nous dirons simplement "item" pour "item $LR(0)$".

Pour définir les item de façon plus active, considérons les mots sur l'alphabet $\cdot + \mathcal{A} + \mathcal{V}$ comportant une et une seule occurrence de $\cdot$ (un tel mot peut s'écrire $\alpha \cdot \beta$ pour $\alpha \in (\mathcal{A}+\mathcal{V})^*$ et $\beta \in (\mathcal{A}+\mathcal{V})^*$). On peut appliquer à ces mots l'opération de "décalage" qui est symbolisée par la règle

$$\cdot \xi \longrightarrow \xi \cdot$$

qui est sensible au contexte.

Alors, pour chaque règle $X \longrightarrow \alpha$ de G :

- $\cdot X \longrightarrow \cdot \alpha$ est un item (qui est l'application de $X \longrightarrow \alpha$ au mot $\cdot X$) ;
- si $\cdot X \longrightarrow \beta \cdot \xi\gamma$ est un item où $\xi \in \mathcal{A}+\mathcal{V}$, alors $\cdot X \longrightarrow \beta\xi \cdot \gamma$ est un item qui est obtenu en appliquant un décalage après $\cdot X \longrightarrow \beta \cdot \xi\gamma$

Exemples.

Compte tenu du fait que $\cdot\emptyset = \emptyset$, la règle $\emptyset \longrightarrow S$ définit deux item :

- *l'item initial* : $\emptyset \longrightarrow \cdot S$
- *l'item final* : $\emptyset \longrightarrow S \cdot$

Une règle $X \longrightarrow \varepsilon$ ne définit que le seul item : $\cdot X \longrightarrow \cdot$ dont le second membre se réduit effectivement à une occurrence de $\cdot$!

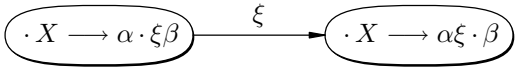
La règle $X \longrightarrow aXaYb$ définit les item suivants :

$$\cdot X \longrightarrow \cdot aXaYb \qquad \cdot X \longrightarrow aX \cdot aYb \qquad \cdot X \longrightarrow aXaY \cdot b$$
$$\cdot X \longrightarrow a \cdot XaYb \qquad \cdot X \longrightarrow aXa \cdot Yb \qquad \cdot X \longrightarrow aXaYb \cdot$$

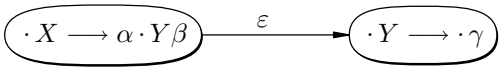
2.3.1 – AFD des item $LR(0)$.

Considérons l'ε–AF **A**, sur l'alphabet $\mathcal{A}+\mathcal{V}$, défini par les données suivantes :

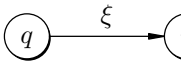
- Les états sont les item ;
- l'entrée est $\emptyset \longrightarrow \cdot S$;
- une transition(*) par $\xi \in \mathcal{A}+\mathcal{V}$ est définie sur les états de la forme $\cdot X \longrightarrow \alpha \cdot \xi\beta$ par un décalage



- une ε–transition est définie sur les états de la forme $\cdot X \longrightarrow \alpha \cdot Y\beta$, par



  pour toute règle $Y \longrightarrow \gamma$

(*) On représente la propriété $r \in \delta(q,\xi)$ par l'arête 

du graphe de transition.

Si l'on fait une sortie de chacun de ses états, **A** reconnaît le langage formé des préfixes viables. Plus précisément

ε–AF des item

Pour tout $\pi \in (\mathcal{A}+\mathcal{V})^*$:
$(\emptyset \longrightarrow \cdot S, \pi) \stackrel{*}{\vdash} (\cdot X \longrightarrow \alpha \cdot \beta, \varepsilon)$ ssi π est un préfixe viable et $\cdot X \longrightarrow \alpha \cdot \beta$ est valide pour π.

Un ε–AF **A** est peu maniable et il est préférable de considérer l'AFD $D(\mathbf{A})$ équivalent obtenu en appliquant l'algorithme de détermination (chapitre 2, section 5), et en négligeant l'état vide.

Ainsi, l'AFD équivalent à **A** est la partie accessible de l'AFD défini de la façon suivante :

- les états sont des ensembles clos non vides d'item,
- l'état initial : $cl(\emptyset \longrightarrow \cdot S)$,
- l'action de ξ sur q, $q \bullet \xi$ est la clôture de la réunion des images des éléments de q,
- les sorties ne jouent aucun rôle ici (on peut, par exemple, prendre tous les états comme sorties).

Lorsque $q \bullet \pi \neq \emptyset$, π définit un chemin partant de $q \in Q$: $ch(q,\pi) \in Chem(q, q \bullet \pi)$.

Plus précisément (cf. la section 7 du chapitre 1) :

- $ch(q,\varepsilon) = q$,
- si $ch(q,\pi) = \chi r$ et si $r \bullet \xi = s$, alors $ch(q,\pi\xi) = ch(q,\pi) \circ rs = \chi rs$.

Propriétés de l'AFD des item.

Toutes ces propriétés viennent directement de la définition de l'AFD et des observations qui ont été faites au sujet des dérivations à droite.

- $q \bullet \xi$ est défini ssi il existe $(\cdot X \longrightarrow \alpha \cdot \xi\beta) \in q$.
- $(\cdot X \longrightarrow \alpha \cdot \xi\beta) \in q$ implique $(\cdot X \longrightarrow \alpha\xi \cdot \beta) \in q \bullet \xi$.
  En conséquence, toutes les transitions aboutissant à un état donné sont étiquetées par le même symbole, c'est–à–dire que, si $q \in \delta(r,\xi)$ et $q \in \delta(s,\eta)$ alors $\xi = \eta$: un chemin détermine donc au plus un mot. Il faudrait respecter cette propriété intéressante si l'on considérait un AFDC : chacun des états improductifs qu'il faudrait introduire pour ce faire correspondrait à une "erreur" particulière.
- on peut reformuler la remarque précédente en disant que, pour un q donné, la connaissance de $ch(q,\pi)$ et celle de π sont équivalentes. Dans la pratique, il est cependant intéressant de les considérer tous les deux simultanément : dans les analyses que nous allons maintenant définir, ils seront traités comme des piles avec sommet à droite ; respectivement "la pile d'états" et "la pile" proprement dite.
- Soient q_0 l'état initial, π un préfixe viable et $q = q_0 \bullet \pi$, alors :
  - q est l'ensemble des item valides pour π,
  - si $(\cdot X \longrightarrow \alpha \cdot \xi\beta) \in q$ alors $\pi\xi$ est viable,
  - si $(\cdot X \longrightarrow \alpha \cdot) \in q$ alors $\pi = \pi'\alpha$ pour un $\pi' \in (\mathcal{A}+\mathcal{V})^*$ et $\pi' X$ est un préfixe viable.

Un item de la forme $\cdot X \longrightarrow \alpha \cdot$ est dit *complet*.

Exemple 1.

Soit G_1 la grammaire :

$$1 : S \longrightarrow (S \wedge S) \qquad 2 : S \longrightarrow \neg S \qquad 3 : S \longrightarrow \mathbf{id}$$

AFD des item $LR(0)$ de G_1.

(Chaque nouvel état I_i est accompagné de l'ensemble Σ_i des symboles ξ tels que $I_i \bullet \xi \neq \emptyset$.)

$$I_0 = \{\emptyset \longrightarrow \cdot S, \quad \cdot S \longrightarrow \cdot(S \wedge S), \cdot S \longrightarrow \cdot \neg S, \cdot S \longrightarrow \cdot \mathbf{id}\} \qquad \Sigma_0 = S + (+\neg + \mathbf{id}$$

$$
\begin{aligned}
I_1 &= I_0 \bullet S && \Sigma_1 = \emptyset \\
&= \{\emptyset \longrightarrow S \cdot\} \\
I_2 &= I_0 \bullet (&& \Sigma_2 = S + (+\neg + \mathbf{id} \\
&= \{\cdot S \longrightarrow (\cdot S \wedge S), \\
&\quad \cdot S \longrightarrow \cdot(S \wedge S), \cdot S \longrightarrow \cdot \neg S, \cdot S \longrightarrow \cdot \mathbf{id}\} \\
I_3 &= I_0 \bullet \neg && \Sigma_3 = S + (+\neg + \mathbf{id} \\
&= \{\cdot S \longrightarrow \neg \cdot S, \\
&\quad \cdot S \longrightarrow \cdot(S \wedge S), \cdot S \longrightarrow \cdot \neg S, \cdot S \longrightarrow \cdot \mathbf{id}\} \\
I_4 &= I_0 \bullet \mathbf{id} && \Sigma_4 = \emptyset \\
&= \{\cdot S \longrightarrow \mathbf{id} \cdot\} \\
I_5 &= I_2 \bullet S && \Sigma_5 = \wedge \\
&= \{\cdot S \longrightarrow (S \cdot \wedge S)\} \\
I_2 &= I_2 \bullet (\\
I_3 &= I_2 \bullet \neg \\
I_4 &= I_2 \bullet \mathbf{id} \\
I_6 &= I_3 \bullet S && \Sigma_6 = \emptyset \\
&= \{\cdot S \longrightarrow \neg S \cdot\} \\
I_2 &= I_3 \bullet (\\
I_3 &= I_3 \bullet \neg \\
I_4 &= I_3 \bullet \mathbf{id} \\
I_7 &= I_5 \bullet \wedge && \Sigma_7 = S + (+\neg + \mathbf{id} \\
&= \{\cdot S \longrightarrow (S \wedge \cdot S), \\
&\quad \cdot S \longrightarrow \cdot(S \wedge S), \cdot S \longrightarrow \cdot \neg S, \cdot S \longrightarrow \cdot \mathbf{id}\} \\
I_8 &= I_7 \bullet S && \Sigma_8 =) \\
&= \{\cdot S \longrightarrow (S \wedge S \cdot)\} \\
I_2 &= I_7 \bullet (\\
I_3 &= I_7 \bullet \neg \\
I_4 &= I_7 \bullet \mathbf{id} \\
I_9 &= I_8 \bullet) && \Sigma_9 = \emptyset \\
&= \{\cdot S \longrightarrow (S \wedge S) \cdot\}
\end{aligned}
$$

Exemple 2.

Soit G_2 la grammaire :

$1 : E \longrightarrow E \oplus T$	$3 : T \longrightarrow T * F$	$5 : F \longrightarrow (E)$
$2 : E \longrightarrow T$	$4 : T \longrightarrow F$	$6 : F \longrightarrow \mathbf{id}$

AFD des item $LR(0)$ de G_2.

$$
\begin{aligned}
I_0 &= \{\emptyset \longrightarrow \cdot E, && \Sigma_0 = E + T + F + (+\mathbf{id} \\
&\quad \cdot E \longrightarrow \cdot E \oplus T, \cdot E \longrightarrow \cdot T, \\
&\quad \cdot T \longrightarrow \cdot T * F, \cdot T \longrightarrow \cdot F, \\
&\quad \cdot F \longrightarrow \cdot(E), \cdot F \longrightarrow \cdot \mathbf{id}\} \\
I_0 \bullet E &= I_1 && \Sigma_1 = \oplus \\
&= \{\emptyset \longrightarrow E \cdot,
\end{aligned}
$$

$$\cdot E \longrightarrow E \cdot \oplus T\}$$

$$I_0 \bullet T = I_2 \qquad \Sigma_2 = *$$
$$= \{\cdot E \longrightarrow T \cdot,$$
$$\cdot T \longrightarrow T \cdot *F\}$$

$$I_0 \bullet F = I_3 \qquad \Sigma_3 = \emptyset$$
$$= \{\cdot T \longrightarrow F \cdot\}$$

$$I_0 \bullet (= I_4 \qquad \Sigma_4 = E + T + F + (+\mathbf{id}$$
$$= \{\cdot F \longrightarrow (\cdot E),$$
$$\cdot E \longrightarrow \cdot E \oplus T, \cdot E \longrightarrow \cdot T,$$
$$\cdot T \longrightarrow \cdot T * F, \cdot T \longrightarrow \cdot F,$$
$$\cdot F \longrightarrow \cdot(E), \cdot F \longrightarrow \cdot \mathbf{id}\}$$

$$I_0 \bullet \mathbf{id} = I_5 \qquad \Sigma_5 = \emptyset$$
$$= \{\cdot F \longrightarrow \mathbf{id} \cdot\}$$

$$I_1 \bullet \oplus = I_6 \qquad \Sigma_6 = T + F + (+\mathbf{id}$$
$$= \{\cdot E \longrightarrow E \oplus \cdot T,$$
$$\cdot T \longrightarrow \cdot T * F, \cdot T \longrightarrow \cdot F,$$
$$\cdot F \longrightarrow \cdot(E), \cdot F \longrightarrow \cdot \mathbf{id}\}$$

$$I_2 \bullet * = I_7 \qquad \Sigma_7 = F + (+\mathbf{id}$$
$$= \{\cdot T \longrightarrow T * \cdot F,$$
$$\cdot F \longrightarrow \cdot(E), \cdot F \longrightarrow \cdot \mathbf{id}\}$$

$$I_4 \bullet E = I_8 \qquad \Sigma_8 =) + \oplus$$
$$= \{\cdot F \longrightarrow (E \cdot),$$
$$\cdot E \longrightarrow E \cdot \oplus T\}$$

$$I_4 \bullet T = I_2$$

$$I_4 \bullet F = I_3$$

$$I_4 \bullet (= I_4$$

$$I_4 \bullet \mathbf{id} = I_5$$

$$I_6 \bullet T = I_9 \qquad \Sigma_9 = *$$
$$= \{\cdot E \longrightarrow E \oplus T \cdot,$$
$$\cdot T \longrightarrow T \cdot *F\}$$

$$I_6 \bullet F = I_3$$

$$I_6 \bullet (= I_4$$

$$I_6 \bullet \mathbf{id} = I_5$$

$$I_7 \bullet F = I_{10} \qquad \Sigma_{10} = \emptyset$$
$$= \{\cdot T \longrightarrow T * F \cdot\}$$

$$I_7 \bullet (= I_4$$

$$I_7 \bullet \mathbf{id} = I_5$$

$$I_8 \bullet) = I_{11} \qquad \Sigma_{11} = \emptyset$$
$$= \{\cdot F \longrightarrow (E) \cdot\}$$

$$I_8 \bullet \oplus = I_6$$

$$I_9 \bullet * = I_7$$

2.4 – L'automate à pile $LR(0)$.

Une configuration d'analyse LR d'un mot $u \in \mathcal{A}^*$ est un couple (π, v) où π est un préfixe viable tel qu'il existe une dérivation $\pi \underset{d}{\Longrightarrow} w$ vérifiant $wv = u$. Pour avoir accès à l'état de l'AFD dans lequel π est validé, il faut modifier un peu ce couple, en considérant la configuration $(ch(q_0, \pi), v)$: comme il a déjà été signalé plus haut, il est intéressant de considérer aussi le mot π, bien que cette information soit redondante.

L'automate à pile pour l'analyse $LR(0)$ de G est défini de la façon suivante :

Une configuration est un couple (χ, v) où

- $\chi = ch(q_0, \pi)$ est traité comme une pile (*la pile des états*) dont le sommet est à droite ; le mot π, lui aussi, est traité comme une pile (*la pile* proprement dite) dont le sommet est encore à droite,
- $v \in \mathcal{A}^*$.

Les transitions sont définies par :

(D) $(\chi q, xv) \vdash\!\!\!- (\chi qr, v)$ si $x \in \mathcal{A}$ et $q \bullet x = r$

(R) $(\chi ch(q, \alpha), v) \vdash\!\!\!- (\chi qr, v)$ si $(\cdot\, X \longrightarrow \alpha\, \cdot) \in q \bullet \alpha$ et $q \bullet X = r$.

La configuration initiale pour l'analyse de u est (q_0, u).

La configuration d'acceptation est $(q_0 q_1, \varepsilon)$ pour $q_1 = q_0 \bullet S$.

Les données nécessaires à la définition des transitions sont consignées dans une table $Action(q, \xi)$ définie pour les états q de l'AFD des item $LR(0)$ et $\xi \in \varepsilon + \mathcal{A} + \mathcal{V}$, de la façon suivante :

Table $LR(0)$

Les $Action(\_, \_)$ sont les plus petits ensembles tels que

- $r \in Action(q, \xi)$ si $q \bullet \xi = r$ pour $\xi \in \mathcal{A} + \mathcal{V}$
- $(X \longrightarrow \alpha) \in Action(q, \varepsilon)$ si $(\cdot\, X \longrightarrow \alpha\, \cdot) \in q$ pour $X \in \mathcal{V}$
- $Acc \in Action(q, \varepsilon)$ si $(\emptyset \longrightarrow S\, \cdot) \in q$

C'est donc la table de transition de l'AFD des item $LR(0)$ de G enrichi d'une colonne pour ε qui comporte, pour chaque état, l'ensemble des "item complets" qu'il contient : on l'appelle *la table $LR(0)$ de G*.

Propriété.

Pour tout $u \in \mathcal{A}^*$:

$$u \in \mathcal{L}(G, S) \text{ ssi } (q_0, u) \vdash\!\!\!-^* (q_0 q_1, \varepsilon)$$

où $\vdash\!\!\!-^*$ signifie l'existence d'un enchaînement de transitions.

La démonstration de cette propriété est basée sur des récurrences très évidentes.

2.5 – Grammaires $LR(0)$.

Une grammaire est dite $LR(0)$ ssi sa table $LR(0)$ vérifie les propriétés suivantes :

pour tout état q de l'AFD

- $Action(q, \varepsilon)$ comporte au plus un élément,
- si $Action(q, \varepsilon) \neq \emptyset$ alors $Action(q, \xi) = \emptyset$ pour tout $\xi \in \mathcal{A}$.

Remarques.

- Lorsque ces conditions sont vérifiées, chaque état q est d'une nature bien déterminée :
  - si $Action(q, \varepsilon) \neq \emptyset$, alors $Action(q, \varepsilon)$ contient un seul élément qui est :
    - ou bien une réduction : q est un *état de réduction*,
    - ou bien Acc : $q = q_1 = q_0 \bullet S$ est l'*état d'acceptation*,
  - sinon, $Action(q, x)$ contient au plus un décalage pour chaque $x \in \mathcal{A}$: q est un *état de décalage*.
- Les conditions signifient que l'automate à pile est déterministe : une analyse, lorsqu'elle est faisable, l'est de façon unique. En particulier, une grammaire $LR(0)$ n'est pas ambiguë.

2.5.1 – Algorithme d'analyse $LR(0)$.

Soit G une grammaire $LR(0)$ dont l'axiome est S.
L'analyse $LR(0)$ de $u \in \mathcal{A}^*$ s'effectue à partir de la configuration initiale (q_0, u) en tentant d'exécuter une suite de transitions (D) ou (R). Chaque application de (R) ajoute une règle au début de la liste d'analyse, initialement vide.

Soit (χ, v) la configuration courante. Lors de l'exécution de l'algorithme, on peut vérifier que χ n'est jamais vide : notons $\chi = \chi' q$ pour mettre en valeur le sommet q de cette pile, qui est l'"état courant" de l'automate. Alors :

- si q est un état de réduction :
  si $Action(q, \varepsilon) = (X \longrightarrow \alpha)$, χ est alors nécessairement de la forme $\chi'' ch(r, \alpha)$ (comme on peut le vérifier par une induction) et si $Action(r, X) = s$: on passe à la configuration $(\chi'' rs, v)$ par application d'une transition (R),
- si q est un état de décalage :
  si $v = xv'$ avec $x \in \mathcal{A}$ et si $Action(q, x) = r$: on passe à la configuration $(\chi' qr, v')$ par application d'une transition (D),
- si $q = q_1 = q_0 \bullet S$ est l'état d'acceptation :
  si $\chi = q_0 q_1$ et si $v = \varepsilon$: on est parvenu à la configuration d'acceptation et l'analyse est terminée de façon satisfaisante.
- Dans les autres cas : l'analyse se termine sur un échec. Une bonne table d'analyse doit comporter un diagnostic d'"erreur syntaxique" pour chacun de ces cas (qui correspondent aux cases vides de la table).

Présentation pratique de la table $LR(0)$.

Les états sont codés par des entiers $(I_0, \ldots, I_n)$ et on note di (décaler en i) au lieu de I_i dans la table de transition. De même, les règles sont codées par des entiers et on note rk (réduire par la règle k), au lieu de la règle elle–même, dans la colonne ε (la réduction par $\emptyset \longrightarrow S$, que l'on n'effectue pas réellement, est codée par Acc). Ceci donne la description suivante de la table $LR(0)$:

- si $I_i \bullet \xi = I_j$ alors $dj \in Action(i, \xi)$
- si $(\cdot\, X \longrightarrow \alpha\, \cdot) \in I_i$ avec $X \in \mathcal{V}$ alors $rk \in Action(i, \varepsilon)$ où k est le numéro de la règle $X \longrightarrow \alpha$
- si $(\emptyset \longrightarrow S\, \cdot) \in I_i$ alors $Acc \in Action(i, \varepsilon)$.

Exemple.

Reprenons la grammaire G_1 de l'exemple 1 ci–dessus, dont on connaît déjà l'AFD des item $LR(0)$.

3 – Analyse ascendante avec symboles de prévision.

Lorsque la première condition $LR(0)$ n'est pas vérifiée, on parle d'un "conflit réduction–réduction", lorsque la deuxième ne l'est pas, d'un "conflit décalage–réduction". Ces conflits sont fréquents dès que G n'est plus extrêmement simpliste : ceci tient à ce que l'on n'est pas prévoyant !

	ε	(	$\wedge$	)	$\neg$	**id**	S
0		$d2$			$d3$	$d4$	$d1$
1	Acc						
2		$d2$			$d3$	$d4$	$d5$
3		$d2$			$d3$	$d4$	$d6$
4	r3						
5			$d7$				
6	r2						
7		$d2$			$d3$	$d4$	$d8$
8				$d9$			
9	r1						

Table $LR(0)$ de G_1.

Pile	Pile d'états	Entrée	Actions
ε	0	$\neg(\mathbf{id} \wedge \neg\mathbf{id})$	($d3$: lecture de $\neg$)
$\neg$	03	$(\mathbf{id} \wedge \neg\mathbf{id})$	($d2$: lecture de ()
$\neg($	032	$\mathbf{id} \wedge \neg\mathbf{id})$	($d4$: lecture de **id**)
$\neg(\mathbf{id}$	0324	$\wedge\neg\mathbf{id})$	$r3 : S \longrightarrow \mathbf{id}$
$\neg(S$	0325	$\wedge\neg\mathbf{id})$	($d7$: lecture de $\wedge$)
$\neg(S\wedge$	03257	$\neg\mathbf{id})$	($d3$: lecture de $\neg$)
$\neg(S \wedge \neg$	032573	$\mathbf{id})$	($d4$: lecture de **id**)
$\neg(S \wedge \neg\mathbf{id}$	0325734	)	$r3 : S \longrightarrow \mathbf{id}$
$\neg(S \wedge \neg S$	0325736	)	$r2 : S \longrightarrow \neg S$
$\neg(S \wedge S$	032578	)	($d9$: lecture de))
$\neg(S \wedge S)$	0325789	ε	$r1 : S \longrightarrow (S \wedge S)$
$\neg S$	036	ε	$r2 : S \longrightarrow \neg S$
S	01	ε	Acc

L'analyse $LR(0)$ de $\neg(\mathbf{id} \wedge \neg\mathbf{id})$ dans G_1.

Exemple de conflit décalage–réduction.

L'état $I_9 = \{\cdot E \longrightarrow E \oplus T \cdot, \cdot T \longrightarrow T \cdot * F\}$ de l'AFD de la grammaire G_2 de l'exemple 2 , contient un item complet, correspondant à une réduction par la règle $E \longrightarrow E \oplus T$ et un item non complet, permettant un décalage sur $*$: G_2 n'est donc pas $LR(0)$.

Dans ce qui suit, nous allons adjoindre des prévisions à 1 caractère pour résoudre ces conflits pour des grammaires assez intéressantes : on est conduit à considérer comme item de type $LR(1)$ les couples $(\cdot X \longrightarrow \alpha \cdot, x)$ où x est un *symbole de prévision*.

La méthode utilisée pour $LR(0)$ s'adapte facilement. Chacun des types d'analyse ainsi obtenu (il y

en a trois) est caractérisé par une gestion particulière des symboles de prévision lors de la définition de l'AF de ses item : nous la préciserons à la fin.

3.1 – Symboles de prévision.

Pour obtenir des algorithmes déterministes, nous allons les rendre "prévoyants". Un choix (ou l'absence de choix) est déterminé par la connaissance approximative de la partie du mot restant à analyser, c'est–à–dire de quelques caractères du début de celle–ci. Voici ce que l'on entend par là :

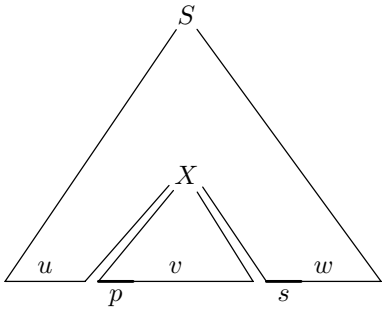
Si k est le nombre de ces caractères et $v \in \mathcal{A}^*$, on définit un mot $Premier_k(v) \in (\varepsilon + \mathcal{A})^k$ (donc, de longueur $\leq k$) de la façon suivante :

- si $|v| \leq k$: $Premier_k(v) = v$,
- sinon $Premier_k(v)$ est le facteur gauche de v dont la longueur est k.

Dans la pratique, il faut étendre cette notion à des mots sur $\mathcal{A}+\mathcal{V}$, en fonction de G et de S, de la façon suivante :

Soit $S \overset{k}{\Longrightarrow} uXw \overset{l}{\Longrightarrow} uvw$ une dérivation du mot $uvw \in \mathcal{A}^*$, dans laquelle on a mis en évidence une sous–dérivation $X \overset{l}{\Longrightarrow} v$ (voir la figure ci–contre). Les approximations qui joueront un rôle sont :

- $p = Premier_k(v)$
- $s = Premier_k(w)$.



$Premier_k(X)$ est l'ensemble de tous les p que l'on peut obtenir ainsi et $Suivant_k(X)$ celui de tous les s que l'on peut obtenir ainsi.

Le cas où $k = 0$ est toujours trivial puisque $Premier_0(v) = \varepsilon$ pour tout v. Le cas où $k = 1$ est assez simple mais déjà très signicatif; de plus, il est suffisant pour analyser les langages que nous avons en vue (des langages de programmation) : nous nous limiterons donc essentiellement à lui et nous ne mentionnerons k que lorsqu'il ne sera pas supposé être égal à 1 : ainsi, *Premier* et *Suivant* signifieront–ils respectivement $Premier_1$ et $Suivant_1$.

3.1.1 – Calcul de *Premier*.

Pour toute $X \in \mathcal{V}$, l'ensemble $Premier(X)$ est formé de facteurs gauches des mots sur $\mathcal{A}$ que l'on peut dériver à partir de X. Plus précisément, $Premier(X) \subseteq \varepsilon + \mathcal{A}$ est le plus petit ensemble tel que :

- si $X \overset{*}{\Longrightarrow} x\alpha$ pour $x \in \mathcal{A}$, alors $x \in Premier(X)$;
- si $X \overset{*}{\Longrightarrow} \varepsilon$, alors $\varepsilon \in Premier(X)$.

La deuxième clause signifie : $\varepsilon \in Premier(X)$ ssi $X \in \mathcal{E}ps(G)$.

$Premier(X)$ n'est pas vide puisque toute variable est productive.

Si, dans la définition de $Premier(X)$, on remplace X par un élément quelconque de $(\mathcal{A}+\mathcal{V})^*$, on obtient facilement les propriétés suivantes.

Premier

1) $Premier(\varepsilon) = \varepsilon$
2) pour tout $x \in \mathcal{A}$ et tout $\alpha \in (\mathcal{A}+\mathcal{V})^*$: $Premier(x\alpha) = x$
3) pour toute $X \in \mathcal{V}$ et tout $\alpha \in (\mathcal{A}+\mathcal{V})^*$:

$$Premier(X\alpha) = \begin{cases} (Premier(X) - \varepsilon) + Premier(\alpha) & \text{si } X \in \mathcal{E}ps(G), \\ Premier(X) & \text{sinon.} \end{cases}$$

4) pour toute règle globale $X \longrightarrow \mathbf{l}$: $Premier(X) = Premier(\mathbf{l})$.

Dans la clause 4), on a étendu l'application *Premier* aux langages de la façon habituelle, elle doit donc se comprendre comme étant :

$$Premier(X) = \sum_{\alpha \in \mathbf{l}} Premier(\alpha).$$

L'ensemble $\mathcal{E}ps(G) \subseteq \mathcal{V}$ des variables de G qui produisent ε, défini par $X \in \mathcal{E}ps(G)$ ssi $\varepsilon \in \mathcal{L}(G, X)$ a été étudié au chapitre 3 (section 4.2).

On a évidemment $Premier_0(X) = \varepsilon$. La définition de $Premier_k$ pour $k > 1$ est techniquement plus compliquée, mais ne présente pas de réelle difficulté.

Exemple.

Appliquons cette construction à la grammaire G :

$1 : E \longrightarrow E \oplus T$ $\quad$ $3 : T \longrightarrow T * F$ $\quad$ $5 : F \longrightarrow (E)$
$2 : E \longrightarrow T$ $\quad$ $4 : T \longrightarrow F$ $\quad$ $6 : F \longrightarrow \mathbf{id}$

- Il est clair que $\mathcal{E}ps(G) = \emptyset$;
- En apliquant 4) et 3) on voit que :

$Premier(E) = Premier(E \oplus T) + Premier(T) = Premier(E) + Premier(T)$
donc $Premier(T) \subseteq Premier(E)$.
$Premier(T) = Premier(T * F) + Premier(F) = Premier(T) + Premier(F)$
donc $Premier(F) \subseteq Premier(T)$.

- En faisant maintenant intervenir 2) :

$Premier(F) = Premier((E)) + Premier(\mathbf{id}) = (+ \mathbf{id}$.

- On peut conclure en prenant les plus petits ensembles vérifiant ces propriétés :

$$Premier(E) = Premier(T) = Premier(F) = (+ \mathbf{id}.$$

3.1.2 – Calcul de *Suivant*.

Pour tout $X \in \mathcal{V}$, l'ensemble $Suivant(X)$ est formé de facteurs de mots sur $\mathcal{A}$ qui peuvent suivre immédiatement une occurrence de X dans un mot que l'on peut dériver à partir de S. Plus précisément, $Suivant(X) \subseteq \varepsilon + \mathcal{A}$ est le plus petit ensemble tel que :

$$\text{si } S \stackrel{*}{\Longrightarrow} \alpha X \beta, \text{ alors } Premier(\beta) \subseteq Suivant(X).$$

$Suivant(X)$ n'est jamais vide puisque toute variable est accessible à partir de S.

Le calcul explicite de *Suivant* est basé sur la propriété suivante :

Suivant

Les $Suivant(\_)$ sont les plus petits ensembles tels que :

1) $\varepsilon \in Suivant(S)$
2) si $Y \longrightarrow \alpha X \beta$ pour $Y \in \mathcal{V}$:
   $Premier(\beta) - \varepsilon \subseteq Suivant(X)$
3) si $Y \longrightarrow \alpha X \beta$ pour $Y \in \mathcal{V}$ et si $\varepsilon \in Premier(\beta)$:
   $Suivant(Y) \subseteq Suivant(X)$

Remarques.

- Il ne faut pas oublier le cas où X se trouve en plusieurs occurrences dans la partie droite d'une règle, lorsque l'on applique les clauses 2) et 3).
- De même, il ne faut pas oublier le cas $\beta = \varepsilon$ dans la clause 3).

La vérification de la propriété est facile.

On a évidemment $Suivant_0(X) = \varepsilon$. La définition de $Suivant_k$ pour $k > 1$ est techniquement plus compliquée, mais ne présente pas de réelle difficulté.

Exemple.

Appliquons cette construction à la grammaire de l'exemple précédent.

- $Suivant(E)$:

Par 1), $\varepsilon \in Suivant(E)$
En appliquant 2) aux règles contenant E à droite, il vient $\oplus \in Suivant(E)$ par 1 : et $) \in Suivant(E)$ par 5 : Comme on a épuisé toutes les possibilités, on peut en conclure que

$$Suivant(E) = \varepsilon + \oplus +)$$

- $Suivant(T)$:

En appliquant 3) à 1 : ou 2 :, il vient $Suivant(E) \subseteq Suivant(T)$
En appliquant 2) à la règle 3 :, on a $* \subseteq Suivant(T)$.
Finalement $Suivant(T) = \varepsilon + \oplus + * +)$.

- $Suivant(F)$:

L'application de 3) à 3 : ou 4 : implique que $Suivant(T) \subseteq Suivant(F)$. Comme il n'y a pas d'autre possibilité, on peut en conclure que

$$Suivant(F) = Suivant(T) = \varepsilon + \oplus + * +)$$

3.2 – Automate à pile de type $LR(1)$.

La définition de l'automate à pile s'effectue comme précédemment, à l'exception des transitions qui tiennent compte des symboles de prévision :

(D) $(\chi q, xv) \vdash\!\!\!- (\chi qr, v)$ si $x \in \mathcal{A}$ et $q.x = r$

(R) $(\chi ch(q,\alpha), v) \vdash\!\!\!- (\chi qr, v)$ si $(\cdot\, X \longrightarrow \alpha\, \cdot, Premier(v)) \in q.\alpha$ et $q.X = r$.

Il est commode de représenter les données nécessaires à la définition des transitions dans une "table" $Action(q, \xi)$ définie pour les états q de l'AFD des item de type $LR(1)$ et $\xi \in \varepsilon + \mathcal{A} + \mathcal{V}$, de la façon suivante :

Table de type $LR(1)$

Les $Action(\_, \_)$ sont les plus petits ensembles tels que

- $r \in Action(q, \xi)$ si $q \bullet \xi = r$ pour $\xi \in \mathcal{A} + \mathcal{V}$
- $(X \longrightarrow \alpha) \in Action(q, x)$ si $(\cdot\, X \longrightarrow \alpha\, \cdot, x) \in q$ pour $X \in \mathcal{V}$
- $Acc \in Action(q, \varepsilon)$ si $(\emptyset \longrightarrow S\, \cdot, \varepsilon) \in q$

C'est donc la réunion de la table de transition de l'AFD en question et d'une table caractérisant la position des "item complets".

Il faut remarquer que l'application d'une transition (R) est maintenant sujette à une condition sur $Premier(v) \in \varepsilon + \mathcal{A}$, contrairement au cas $LR(0)$.

3.2.1 – Les trois types $LR(1)$.

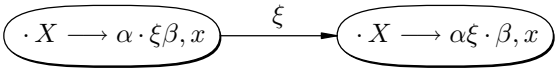
$SLR(1)$ (*Simple* $LR(1)$) :

dans ce type, on forme un item pour $\cdot X \longrightarrow \alpha \cdot$ avec **tout** élément de $Suivant(X)$.

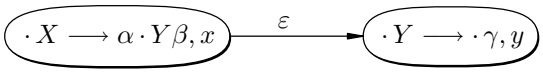
$LR(1)$ proprement dit :

l'action de l'AF est définie de la façon suivante :

- une transition(*) par $\xi \in \mathcal{A} + \mathcal{V}$ est définie sur les états de la forme $(\cdot X \longrightarrow \alpha \cdot \xi\beta, x)$ par



- une ε–transition est définie sur les états de la forme $(\cdot X \longrightarrow \alpha \cdot Y\beta, x)$ par



pour chaque $(Y \longrightarrow \gamma) \in R$ et chaque $y \in Premier(\beta x)$.

$LALR(1)$ (*Look Ahead* $LR(1)$) :

les états de l'AFD correspondant à ce type s'obtiennent en faisant la réunion des états de l'AFD $LR(1)$ qui ne diffèrent que par les symboles de prévision : ses états sont ceux de l'AFD $LR(0)$, aux éléments desquels on a adjoint des symboles de prévision. Il existe des méthodes pour calculer ces symboles qui ne nécessitent pas le calcul de l'AFD $LR(1)$ (Yacc utilise une méthode de ce genre).

3.3 – Grammaires de type $LR(1)$.

Une grammaire est dite $LR(1)$ *(resp.* $SLR(1)$, $LALR(1)$*) ssi sa table* $LR(1)$ *(resp.* $SLR(1)$, $LALR(1)$*) est déterministe,*

c'est–à–dire, ssi chaque $Action(q, \xi)$ comporte toujours au plus un élément.

Remarques.

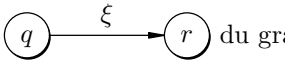
- Cette condition signifie qu'il ne se produit pas de conflit et assure évidemment que l'automate à pile correspondant est déterministe.
- Contrairement au cas $LR(0)$, la nature de la transition à exécuter n'est pas déterminée par le seul état courant (sommet de la pile d'états) mais aussi par le symbole de prévision.
- Comme dans le cas $LR(0)$, on peut voir qu'une grammaire de type $LR(1)$ n'est pas ambiguë.

3.3.1 – Algorithme d'analyse de type $LR(1)$.

Soit G une grammaire de type $LR(1)$ dont l'axiome est S.
L'analyse $LR(1)$ de $u \in \mathcal{A}^*$ s'effectue à partir de la configuration initiale (q_0, u) en tentant d'exécuter une suite de transitions (D) ou (R). Chaque application de (R) ajoute une règle au début de la liste d'analyse, initialement vide.

Soit (χ, v) la configuration courante. Lors de l'exécution de l'algorithme, on peut vérifier que χ n'est jamais vide : notons $\chi = \chi' q$ pour mettre en valeur le sommet q de cette pile, qui est l'"état courant" de l'automate. Alors :

- si $Action(q, Premier(v)) = (X \longrightarrow \alpha)$, χ est alors nécessairement de la forme $\chi'' ch(r, \alpha)$ (comme on peut le vérifier par une induction) et si $Action(r, X) = s$: on passe à la configuration $(\chi'' rs, v)$ par application d'une transition (R),

(*) On représente la propriété $r \in \delta(q, \xi)$ par l'arête 

du graphe de transition.

• si $Action(q, Premier(v)) = r$ où r est un état (alors on a nécessairement $Premier(v) = x \in \mathcal{A}$ et) v s'écrit sous la forme $v = xv'$, et si $Action(q, x) = r$: on passe à la configuration $(\chi' qr, v')$ par application d'une transition (D),

• si $\chi = q_0 q_1$ et si $v = \varepsilon$: on est parvenu à la configuration d'acceptation et l'analyse est terminée de façon satisfaisante.

• Dans les autres cas : l'analyse se termine sur un échec. Une bonne table d'analyse doit comporter un diagnostic d'"erreur syntaxique" pour chacun de ces cas (qui correspondent aux cases vides de la table).

Le codage introduit pour la table $LR(0)$ est encore appliqué ici ; ceci donne la description suivante :

- si $I_i \bullet \xi = I_j$ alors $dj \in Action(i, \xi)$
- si $(\cdot X \longrightarrow \alpha \cdot, x) \in I_i$ avec $X \in \mathcal{V}$ alors $rk \in Action(i, x)$ où k est le numéro de la règle $X \longrightarrow \alpha$
- si $(\emptyset \longrightarrow S \cdot, \varepsilon) \in I_i$ alors $Acc \in Action(i, \varepsilon)$.

Exemple 2 : analyse $SLR(1)$.

Reprenons la grammaire G_2 de l'exemple 2 ci–dessus, dont on connaît déjà l'AFD des item $LR(0)$.

Symboles de prévision.

$\mathcal{E}ps(G) = \emptyset$
$Premier(E) = Premier(T) = Premier(F) = (+\mathbf{id}$
$Suivant(E) = \varepsilon + \oplus +)$
$Suivant(T) = Suivant(F) = \varepsilon + \oplus + * +)$

La table $SLR(1)$ de G_2 et un exemple d'analyse sont présentés dans les tableaux ci–dessous.

	ε	$\oplus$	$*$	(	)	**id**	E	T	F
0				$d4$		$d5$	$d1$	$d2$	$d3$
1	Acc	$d6$							
2	$r2$	$r2$	$d7$		$r2$				
3	$r4$	$r4$	$r4$		$r4$				
4				$d4$		$d5$	$d8$	$d2$	$d3$
5	$r6$	$r6$	$r6$		$r6$				
6				$d4$		$d5$		$d9$	$d3$
7				$d4$		$d5$			$d10$
8		$d6$			$d11$				
9	$r1$	$r1$	$d7$		$r1$				
10	$r3$	$r3$	$r3$		$r3$				
11	$r5$	$r5$	$r5$		$r5$				

Table $SLR(1)$ de G_2.

Pile	Pile d'états	Entrée	Actions
ε	0	$\mathbf{id} \oplus \mathbf{id} * \mathbf{id}$	($d5$: lecture de **id**)
id	05	$\oplus \mathbf{id} * \mathbf{id}$	$r6 : F \longrightarrow \mathbf{id}$
F	03	$\oplus \mathbf{id} * \mathbf{id}$	$r4 : T \longrightarrow F$
T	02	$\oplus \mathbf{id} * \mathbf{id}$	$r2 : E \longrightarrow T$
E	01	$\oplus \mathbf{id} * \mathbf{id}$	($d6$: lecture de $\oplus$)
$E\oplus$	016	$\mathbf{id} * \mathbf{id}$	($d5$: lecture de **id**)
$E \oplus \mathbf{id}$	0165	$* \mathbf{id}$	$r6 : F \longrightarrow \mathbf{id}$
$E \oplus F$	0163	$* \mathbf{id}$	$r4 : T \longrightarrow F$
$E \oplus T$	0169	$* \mathbf{id}$	($d7$: lecture de $*$)
$E \oplus T*$	01697	**id**	($d5$: lecture de **id**)
$E \oplus T * \mathbf{id}$	016975	ε	$r6 : F \longrightarrow \mathbf{id}$
$E \oplus T * F$	0169710	ε	$r3 : T \longrightarrow T * F$
$E \oplus T$	0169	ε	$r1 : E \longrightarrow E \oplus T$
E	01	ε	Acc

L'analyse $SLR(1)$ de $\mathbf{id} \oplus \mathbf{id} * \mathbf{id}$ dans G_2.

Exemple 3 : analyses $LR(1)$ et $LALR(1)$.

Considérons la grammaire G_3 :

$1 : S \longrightarrow G\texttt{=}D \qquad 3 : G \longrightarrow *D \qquad 5 : D \longrightarrow G$
$2 : S \longrightarrow D \qquad 4 : G \longrightarrow \mathbf{id}$

Le début du calcul de l'AFD des item $LR(0)$ montre que G_3 n'est pas $SLR(1)$, en effet, on a

$$\begin{aligned} I_0 = \{&\emptyset \longrightarrow \cdot S, \\ &\cdot S \longrightarrow \cdot G\texttt{=}D, \cdot S \longrightarrow \cdot D, \\ &\cdot G \longrightarrow \cdot *D, \cdot G \longrightarrow \cdot \mathbf{id}, \\ &\cdot D \longrightarrow \cdot G\} \end{aligned}$$

et donc $I_2 = I_0 \bullet G = \{\cdot S \longrightarrow G \cdot \texttt{=}D, \cdot D \longrightarrow G \cdot\}$; or, il est facile de voir que $\texttt{=} \in Suivant(D)$ et donc que la méthode $SLR(1)$ ne résoud pas le conflit décalage–réduction qui se présente dans cet état : nous allons voir que les item $LR(1)$ sont capables de le faire.

AFD des item $LR(1)$ de G_3.

L'état initial est :

$$\begin{aligned} I_0 = \{&(\emptyset \longrightarrow \cdot S, \varepsilon), \\ &(\cdot S \longrightarrow \cdot G\texttt{=}D, \varepsilon), (\cdot S \longrightarrow \cdot D, \varepsilon), \\ &(\cdot G \longrightarrow \cdot *D, \texttt{=}), (\cdot G \longrightarrow \cdot \mathbf{id}, \texttt{=}), \\ &(\cdot D \longrightarrow \cdot G, \varepsilon), \\ &(\cdot G \longrightarrow \cdot *D, \varepsilon), (\cdot G \longrightarrow \cdot \mathbf{id}, \varepsilon)\} \end{aligned}$$

Il est commode de regrouper les item d'un même état qui ne diffèrent que par leur symbole de prévision, en notant (ϱ, P) un ensemble d'item de ce type $\sum_{x \in P} (\varrho, x)$, où $P \subseteq \varepsilon + \mathcal{A}$ (une notation plus standard serait simplement $\varrho \times P$, qui désigne bien l'ensemble des couples que l'on veut représenter).

Reprenons notre calcul en utilisant cette notation.

$$
\begin{aligned}
I_0 = {} & \{(\emptyset \longrightarrow \cdot S, \varepsilon), && \Sigma_0 = S + G + D + * + \mathbf{id} \\
& (\cdot S \longrightarrow \cdot G\texttt{=}D, \varepsilon), (\cdot S \longrightarrow \cdot D, \varepsilon), \\
& (\cdot G \longrightarrow \cdot *D, \varepsilon + \texttt{=}), (\cdot G \longrightarrow \cdot \mathbf{id}, \varepsilon + \texttt{=}), \\
& (\cdot D \longrightarrow \cdot G, \varepsilon)\} \\
I_0 \bullet S = {} & I_1 && \Sigma_1 = \emptyset \\
= {} & \{(\emptyset \longrightarrow S \cdot, \varepsilon)\} \\
I_0 \bullet G = {} & I_2 && \Sigma_2 = \texttt{=} \\
= {} & \{(\cdot S \longrightarrow G \cdot \texttt{=}D, \varepsilon), (\cdot D \longrightarrow G \cdot, \varepsilon)\} \\
I_0 \bullet D = {} & I_3 && \Sigma_3 = \emptyset \\
= {} & \{(\cdot S \longrightarrow D \cdot, \varepsilon)\} \\
I_0 \bullet * = {} & I_4 && \Sigma_4 = D + G + * + \mathbf{id} \\
= {} & \{(\cdot G \longrightarrow * \cdot D, \varepsilon + \texttt{=}), \\
& (\cdot D \longrightarrow \cdot G, \varepsilon + \texttt{=}), \\
& (\cdot G \longrightarrow \cdot *D, \varepsilon + \texttt{=}), (\cdot G \longrightarrow \cdot \mathbf{id}, \varepsilon + \texttt{=})\} \\
I_0 \bullet \mathbf{id} = {} & I_5 && \Sigma_5 = \emptyset \\
= {} & \{(\cdot G \longrightarrow \mathbf{id} \cdot, \varepsilon + \texttt{=})\} \\
I_2 \bullet \texttt{=} = {} & I_6 && \Sigma_6 = D + G + * + \mathbf{id} \\
= {} & \{(\cdot S \longrightarrow G\texttt{=} \cdot D, \varepsilon), \\
& (\cdot D \longrightarrow \cdot G, \varepsilon), \\
& (\cdot G \longrightarrow \cdot *D, \varepsilon), (\cdot G \longrightarrow \cdot \mathbf{id}, \varepsilon)\} \\
I_4 \bullet D = {} & I_7 && \Sigma_7 = \emptyset \\
= {} & \{(\cdot G \longrightarrow *D \cdot, \varepsilon + \texttt{=})\} \\
I_4 \bullet G = {} & I_8 && \Sigma_8 = \emptyset \\
= {} & \{(\cdot D \longrightarrow G \cdot, \varepsilon + \texttt{=})\} \\
I_4 \bullet * = {} & I_4 \\
I_4 \bullet \mathbf{id} = {} & I_5 \\
I_6 \bullet D = {} & I_9 && \Sigma_9 = \emptyset \\
= {} & \{(\cdot S \longrightarrow G\texttt{=}D \cdot, \varepsilon)\} \\
I_6 \bullet G = {} & I_{10} && \Sigma_{10} = \emptyset \\
= {} & \{(\cdot D \longrightarrow G \cdot, \varepsilon)\} \\
I_6 \bullet * = {} & I_{11} && \Sigma_{11} = D + G + * + \mathbf{id} \\
= {} & \{(\cdot G \longrightarrow * \cdot D, \varepsilon), \\
& (\cdot D \longrightarrow \cdot G, \varepsilon), \\
& (\cdot G \longrightarrow \cdot *D, \varepsilon), (\cdot G \longrightarrow \cdot \mathbf{id}, \varepsilon)\} \\
I_6 \bullet \mathbf{id} = {} & I_{12} && \Sigma_{12} = \emptyset \\
= {} & \{(\cdot G \longrightarrow \mathbf{id} \cdot, \varepsilon)\} \\
I_{11} \bullet D = {} & I_{13} && \Sigma_{13} = \emptyset \\
= {} & \{(\cdot G \longrightarrow *D \cdot, \varepsilon)\} \\
I_{11} \bullet G = {} & I_{10} \\
I_{11} \bullet * = {} & I_{11} \\
I_{11} \bullet \mathbf{id} = {} & I_{12}
\end{aligned}
$$

Grâce à la gestion plus parcimonieuse des symboles de prévision, les item $LR(1)$ définissent un AFD dont aucun état ne connaît de conflit ; dans l'état I_2 qui était conflictuel dans la version $SLR(1)$, le symbole = n'est plus utilisé pour déclencher une réduction par la règle $D \longrightarrow G$, ainsi la table $LR(1)$ est–elle déterministe.

	ε	=	$*$	**id**	S	G	D
0			$d4$	$d5$	$d1$	$d2$	$d3$
1	Acc						
2	$r5$	$d6$					
3	$r2$						
4			$d4$	$d5$		$d8$	$d7$
5	$r4$	$r4$					
6			$d11$	$d12$		$d10$	$d9$
7	$r3$	$r3$					
8	$r5$	$r5$					
9	$r1$						
10	$r5$						
11			$d11$	$d12$		$d10$	$d13$
12	$r4$						
13	$r3$						

Table $LR(1)$ de G_3.

Lorsque l'on fait la réunion des états de l'AFD $LR(1)$ qui ne diffèrent que par les symboles de prévision, on ne réintroduit pas de conflit : la table $LALR(1)$ a été construite en numérotant les nouveaux états de la façon suivante :

$$I'_4 = I_4 + I_{11} \qquad I'_5 = I_5 + I_{12} \qquad I'_7 = I_7 + I_{13} \qquad I'_8 = I_8 + I_{10}$$

La grammaire G_3 est donc $LALR(1)$.

4 – Annexe : analyse descendante.

Une analyse de ce type se fait en descendant de l'axiome vers le mot à analyser. L'analyse détermine donc les règles convenables dans le même ordre que celui dans lequel elles seraient appliquées lors d'une dérivation. De même, l'analyse se fait de gauche à droite : puisque l'on "descend", ceci correspond à une dérivation à gauche (d'où le deuxième L, pour "Leftmost derivation", du sigle LL).

	ε	=	$*$	**id**	S	G	D
0			$d4$	$d5$	$d1$	$d2$	$d3$
1	Acc						
2	$r5$	$d6$					
3	$r2$						
4			$d4$	$d5$		$d8$	$d7$
5	$r4$	$r4$					
6			$d4$	$d5$		$d8$	$d9$
7	$r3$	$r3$					
8	$r5$	$r5$					
9	$r1$						

Table $LALR(1)$ de G_3.

4.1 – Configurations d'analyse partielle.

Une dérivation à *gauche* est un enchaînement de dérivations élémentaires à gauche, c'est–à–dire, de la forme

$$wX\pi \overset{1}{\Longrightarrow} w\alpha\pi$$

où $w \in \mathcal{A}^*$: ceci signifie que l'on applique toujours une règle sur la variable qui est le plus à gauche du mot courant.

L'analyse descendante de $u \in \mathcal{A}^*$ est la construction d'une dérivation $S \Longrightarrow u$ dont on connaît l'aboutissement.

On peut appeler "analyse partielle de u" une dérivation $S \underset{g}{\Longrightarrow} w\pi$ telle qu'il existe une dérivation $\pi \underset{g}{\Longrightarrow} v$ pour laquelle $wv = u$. Avec ces notations, une configuration d'analyse partielle de u peut s'écrire $(w; \pi, v)$

la configuration initiale est $(\varepsilon; S, u)$
la configuration d'acceptation est $(u; \varepsilon, \varepsilon)$
les modifications permises sont de deux types :

$$L : (w; x\pi', xv') \overset{1}{\Longrightarrow} (wx; \pi', v') \text{ pour } x \in \mathcal{A}$$
$$D : (w; X\pi', v) \overset{1}{\Longrightarrow} (w; \alpha\pi', v) \text{ pour } X \underset{G}{\longrightarrow} \alpha.$$

- Une "lecture", c'est–à–dire une modification L, ne s'applique que lorsque $\pi = x\pi'$ et $v = xv'$ pour le même $x \in \mathcal{A}$.
- Chaque modification D ajoute une règle à la liste qui constituera l'analyse de u. Or, $S \underset{g}{\Longrightarrow} wX\pi' \underset{g}{\overset{1}{\Longrightarrow}} w\alpha\pi'$ n'est une analyse partiele de u que s'il existe une dérivation $\alpha\pi' \underset{g}{\Longrightarrow} v$. Une condition nécessaire à l'existence d'une telle dérivation est, si l'on prévoit 1 caractère $x = Premier(v)$, que :
  - ou bien $x \in Premier(\alpha)$,
  - ou bien $\varepsilon \in Premier(\alpha)$ et $x \in Suivant(X)$.

4.2 – Automate à pile $LL(1)$.

Ce qui précède est adapté à l'analyse d'un $u \in \mathcal{A}^*$ particulier : si on veut traiter le problème plus globalement, on est conduit à considérer les couples (π, v) au lieu des triplets $(w; \pi, v)$ et à poser la définition suivante.

L'automate à pile pour l'analyse $LL(1)$ de G est défini de la façon suivante :

Une configuration est un couple (π, v) où $v \in \mathcal{A}^*$ et où $\pi \in (\mathcal{A} + \mathcal{V})^*$ est traité comme une pile dont le sommet est à gauche.
Les transitions sont de deux types :

(L) $(x\pi, xv) \vdash\!\!\!- (\pi, v)$ pour tout $x \in \mathcal{A}$,

(D) $(X\pi, v) \vdash\!\!\!- (\alpha\pi, v)$ pour $(X \longrightarrow \alpha) \in M(X, Premier(v))$

où, pour $X \in \mathcal{V}$ et $x \in \varepsilon + \mathcal{A}$.
La configuration initiale pour l'analyse de u est (ε, u).
La configuration finale est $(\varepsilon, \varepsilon)$.

$M(X, x)$ est un ensemble de règles de G défini de la façon suivante :

Table $LL(1)$

Les $M(\_, \_)$ sont les plus petits ensembles de règles tels que :

pour toute $X \in \mathcal{V}$, toute règle $X \longrightarrow \alpha$ et tout $x \in Premier(\alpha)$:

- si $x \in \mathcal{A}$ alors $(X \longrightarrow \alpha) \in M(X, x)$
- si $x = \varepsilon$ alors $(X \longrightarrow \alpha) \in M(X, y)$ pour tout $y \in Suivant(X)$.

L'application à un $u \in \mathcal{A}^*$ particulier permet effectivement de faire le lien avec les observations de la section précédente :

Propriété.

Pour tout $u \in \mathcal{A}^*$:

$$u \in \mathcal{L}(G, S) \text{ ssi } (S, u) \vdash\!\!\!-^* (\varepsilon, \varepsilon)$$

où $\vdash\!\!\!-^*$ signifie l'existence d'un enchaînement de transitions.

La démonstration de cette propriété est basée sur des récurrences très évidentes.

4.3 – Grammaires $LL(1)$.

Une grammaire est $LL(1)$ ssi sa table $LL(1)$ est déterministe, c'est–à–dire ssi chaque $M(X, x)$ contient au plus un élément.

Lorsque c'est le cas, $u \in \mathcal{L}(G, S)$ équivaut donc à l'existence d'exactement une dérivation $(S, u) \vdash\!\!\!-^l (\varepsilon, \varepsilon)$, c'est–à–dire à celle d'exactement une dérivation à gauche $S \underset{g}{\overset{k}{\Longrightarrow}} u$.

Remarques.

- Ce qui vient d'être dit implique en particulier qu'une grammaire $LL(1)$ n'est pas ambiguë.
- En regardant la définition en détails, on peut vérifier qu'une grammaire qui comporte une règle "récursive à gauche" $X \longrightarrow X\alpha$ n'est pas $LL(1)$, mais les méthodes de dérécursion (voir chapitre 3) peuvent alors s'appliquer.
- De même, qu'une grammaire n'est pas $LL(1)$ lorsqu'elle admet une factorisation, c'est–à–dire, deux règles distinctes $X \longrightarrow \xi\alpha$ et $X \longrightarrow \xi\beta$ pour $\xi \in \mathcal{A} + \mathcal{V}$.

En fait, ces deux derniers points peuvent se résoudre en modifiant la grammaire de façon adéquate.

4.3.1 – Algorithme d'analyse $LL(1)$.

Soit G une grammaire avec S comme axiome et dont la table $LL(1)$ est déterministe.

L'analyse $LL(1)$ de $u \in \mathcal{A}^*$ par G s'effectue à partir de la "configuration initiale" (S, u) en tentant d'exécuter une suite d'opérations (L) et (D). Chaque application de (D) ajoute une règle à la "liste d'analyse", initialement vide.

Notons (π, v) la configuration à laquelle on est parvenu (π est la "pile" dont le sommet est à gauche et v l'"entrée" c'est–à–dire le facteur droit du mot restant à analyser) :

- si $\pi = x\pi'$ avec $x \in \mathcal{A}$:
  - si $v = xv'$: on fait la lecture de x, c'est–à–dire que l'on passe à la configuration (π', v') par application de (L),
- si $\pi = X\pi'$ avec $X \in \mathcal{V}$:
  - si $X \longrightarrow \alpha \in M(X, Premier(v))$: on passe à la configuration $(\alpha\pi', v)$, par application de (D),
- si $\pi = \varepsilon$ et $v = \varepsilon$, on est parvenu à la "configuration d'acceptation" : l'analyse est achevée, de façon positive,
- Dans les autres cas : l'analyse se termine sur un échec. Un bonne table d'analyse doit comporter un diagnostic d'"erreur syntaxique" pour chacun de ces cas (qui correspondent aux cases vides de la table).

Exemple.

Considérons la grammaire G_4 :

$1 : E \longrightarrow TE'$ $\quad 4 : T \longrightarrow FT'$ $\quad 7 : F \longrightarrow (E)$
$2 : E' \longrightarrow \oplus TE'$ $\quad 5 : T' \longrightarrow *FT'$ $\quad 8 : F \longrightarrow \mathbf{id}$
$3 : E' \longrightarrow \varepsilon$ $\quad 6 : T' \longrightarrow \varepsilon$

La construction de la table $LL(1)$ de G_4 est basée sur les données ci–dessous, dont le calcul est facile.

$\mathcal{E}ps(G) = E' + T'$
$Premier(E) = Premier(T) = Premier(F) = (+\mathbf{id}$
$Premier(E') = \oplus + \varepsilon$, $Premier(T') = * + \varepsilon$
$Suivant(E) = Suivant(E') =) + \varepsilon$
$Suivant(T) = Suivant(T') = \oplus +) + \varepsilon$
$Suivant(F) = \oplus + *+) + \varepsilon$

	ε	$\oplus$	$*$	(	)	**id**
E				1		1
E'	3	2			3	
T				4		4
T'	6	6	5		6	
F				7		8

La table $LL(1)$ de G_4.

Pile	Entrée	Actions
E	$\mathbf{id} \oplus \mathbf{id} * \mathbf{id}$	$1 : E \longrightarrow TE'$
TE'	$\mathbf{id} \oplus \mathbf{id} * \mathbf{id}$	$4 : T \longrightarrow FT'$
$FT'E'$	$\mathbf{id} \oplus \mathbf{id} * \mathbf{id}$	$8 : F \longrightarrow \mathbf{id}$
$\mathbf{id}T'E'$	$\mathbf{id} \oplus \mathbf{id} * \mathbf{id}$	(Lecture de **id**)
$T'E'$	$\oplus\mathbf{id} * \mathbf{id}$	$6 : T' \longrightarrow \varepsilon$
E'	$\oplus\mathbf{id} * \mathbf{id}$	$2 : E' \longrightarrow \oplus TE'$
$\oplus TE'$	$\oplus\mathbf{id} * \mathbf{id}$	(Lecture de $\oplus$)
TE'	$\mathbf{id} * \mathbf{id}$	$4 : T \longrightarrow FT'$
$FT'E'$	$\mathbf{id} * \mathbf{id}$	$8 : F \longrightarrow \mathbf{id}$
$\mathbf{id}T'E'$	$\mathbf{id} * \mathbf{id}$	(Lecture de **id**)
$T'E'$	$*\mathbf{id}$	$5 : T' \longrightarrow *FT'$
$*FT'E'$	$*\mathbf{id}$	Lecture de $*$
$FT'E'$	$\mathbf{id}$	$8 : F \longrightarrow \mathbf{id}$
$\mathbf{id}T'E'$	$\mathbf{id}$	(Lecture de **id**)
$T'E'$	ε	$6 : T' \longrightarrow \varepsilon$
E'	ε	$3 : E' \longrightarrow \varepsilon$
ε	ε	OK

L'analyse $LL(1)$ de $\mathbf{id} \oplus \mathbf{id} * \mathbf{id}$ dans G_4.

EXERCICES.

Analyse LR.

Exercice 1.

Montrer que la grammaire suivante est $LR(0)$:

$1 : S \longrightarrow CC \qquad 2 : C \longrightarrow \mathbf{c}\,C \qquad 3 : C \longrightarrow \mathbf{d}$

Exercice 2.

Montrer que la grammaire suivante est $LR(0)$:

$1 : S \longrightarrow \mathbf{f}\,SS \qquad 2 : S \longrightarrow \mathbf{g}\,S \qquad 3 : S \longrightarrow \mathbf{a}$

Faire l'analyse de **g f f a g a g a** et construire l'arbre de dérivation correspondant.

Exercice 3.

La grammaire suivante est–elle $LR(0)$? $SLR(1)$?

$1 : S \longrightarrow SS\,\mathbf{f} \qquad 2 : S \longrightarrow S\,\mathbf{g} \qquad 3 : S \longrightarrow \mathbf{a}$

Faire l'analyse de **a g a g a f f g** et construire l'arbre de dérivation correspondant.

Exercice 4.

Construire la table $SLR(1)$ de la grammaire suivante :

$1 : E \longrightarrow TF \qquad 2 : F \longrightarrow \varepsilon \qquad 4 : T \longrightarrow (E)$
$3 : F \longrightarrow \oplus TF \qquad 5 : T \longrightarrow \mathbf{id}$

La grammaire en question est–elle $SLR(1)$? Si oui, utiliser la table pour faire l'analyse de **id** $\oplus$(**id**) et construire l'arbre de dérivation correspondant.

Exercice 5.

Voici quelques grammaires sur la nature desquelles vous pourrez vous interroger. Au passage, vérifiez que, pour toute grammaire G :

- si G est $LR(0)$ alors G est $SLR(1)$,
- si G est $SLR(1)$ alors G est $LALR(1)$,
- si G est $LALR(1)$ alors G est $LR(1)$.

G_5 : $1 : S \longrightarrow \mathbf{a}\,A\,\mathbf{c} \qquad 2 : A \longrightarrow A\,\mathbf{b}\,\mathbf{b} \qquad 3 : A \longrightarrow \mathbf{b}$

G_6 : $1 : A \longrightarrow \mathbf{a}\,S \qquad 2 : S \longrightarrow \mathbf{b}\,S \qquad 3 : S \longrightarrow \mathbf{a}\,\mathbf{a}\,\mathbf{b}$

G_7 : $1 : A \longrightarrow BA \qquad 2 : A \longrightarrow \mathbf{a} \qquad 3 : B \longrightarrow AB \qquad 4 : B \longrightarrow \mathbf{b}$

G_8 : $1 : S \longrightarrow S\,\mathbf{a}\,S\,\mathbf{b} \qquad 2 : S \longrightarrow \varepsilon$

G_9 : $1 : S \longrightarrow AB \qquad 2 : A \longrightarrow \mathbf{a}\,A\,\mathbf{b} \qquad 4 : B \longrightarrow \mathbf{b}\,B$
$3 : A \longrightarrow \varepsilon \qquad 5 : B \longrightarrow \mathbf{b}$

G_{10} : $1 : S \longrightarrow AB \qquad 3 : B \longrightarrow CD \qquad 5 : C \longrightarrow \mathbf{a}\,\mathbf{b} \qquad 7 : E \longrightarrow \mathbf{b}\,\mathbf{b}\,\mathbf{a}$
$2 : A \longrightarrow \mathbf{a} \qquad 4 : B \longrightarrow \mathbf{a}\,E \qquad 6 : D \longrightarrow \mathbf{b}\,\mathbf{b}$

G_{11} : $1 : S \longrightarrow S \oplus A \qquad 3 : A \longrightarrow (S) \qquad 5 : A \longrightarrow \mathbf{a}$
$2 : S \longrightarrow A \qquad 4 : A \longrightarrow \mathbf{a}\,(S)$

G_{12} : $1 : S \longrightarrow \mathbf{a}\,D\,\mathbf{;}\,I\,\mathbf{b} \qquad 2 : D \longrightarrow D\,\mathbf{;}\,\mathbf{d} \qquad 4 : I \longrightarrow \mathbf{i}\,\mathbf{;}\,I$
$3 : D \longrightarrow \mathbf{d} \qquad 5 : I \longrightarrow \mathbf{i}$

Utilisation de grammaires ambiguës.

Les langages de programmation permettent l'usage de quelques ambiguïtés. Les programmes y gagnent en simplicité mais ne seraient pas analysables par une méthode déterministe si certaines conventions n'étaient pas posées, par exemple : il est en général convenu que l'expression **id** + **id** + **id** sera calculée comme (**id** + **id**) + **id** .

Il existe deux méthodes pour faire l'analyse syntaxique de telles expressions :

1) On utilise une grammaire ambiguë, qui suit exactement la syntaxe des espressions en question : les conflits, qui se présentent inévitablement dans la table d'analyse, sont résolus en choisissant (une fois pour toute!) dans chaque état de l'AFD, l'action qui correspond à la façon dont on prétend faire l'analyse. Yacc est capable de faire ce choix, dans des cas simples, lorsqu'on lui donne des informations sur l'associativité (ou la non associativité) des opérations et sur leur préséance relative.
   Cette opération consiste en fait à sélectionner certaines dérivations, parmi toutes celles qui sont possibles : on court ainsi le risque de ne plus pouvoir analyser des phrases qui sont pourtant dans le langage engendré par la grammaire!
2) On utilise une grammaire non ambiguë qui est capable de faire l'analyse correctement. Cette méthode paraît plus saine que la précédente mais, sa mise en œuvre nécessite une grammaire plus complexe, en particulier comportant plus de variables; les analyses dans une telle grammaire sont souvent beaucoup plus longues que dans la méthode 1).

Les deux exercices qui suivent ont pour but de montrer comment on pratique la première méthode. La seconde méthode est seulement illustrée par la donnée d'une grammaire permettant sa mise en œuvre : les vérifications utiles sont laissées à votre initiative personnelle!

Exercice 6. Ambiguïté des expressions arithmétiques.

Calculer la table $SLR(1)$ de la grammaire

$$1 : E \to E \oplus E \qquad 2 : E \to E * E \qquad 3 : E \to (E) \qquad 4 : E \to \mathbf{id}$$

puis résoudre les conflits que l'on peut y observer de telle façon que $\oplus$ et $*$ soient associatives à gauche et que $*$ ait une préséance supérieure à $\oplus$.

(La grammaire

$$\begin{array}{lll} E \to E \oplus T & T \to T * F & F \to (E) \\ E \to T & T \to F & F \to \mathbf{id} \end{array}$$

équivalente à la précédente, est $SLR(1)$ et prend en compte les conventions précédentes.)

Exercice 7. Ambiguïté du *"sinon en suspens"***.**

La grammaire suivante décrit les instruction conditionnelles :

$$I \to \mathbf{si}\ E\ \mathbf{alors}\ I\ \mathbf{sinon}\ I \qquad I \to \mathbf{si}\ E\ \mathbf{alors}\ I \qquad I \to \mathbf{autre}$$

Pour l'étudier, nous en considérons la forme abrégée suivante :

$$1 : I \to \mathbf{i}\ I\ \mathbf{e}\ I \qquad 2 : I \to \mathbf{i}\ I \qquad 3 : I \to \mathbf{a}$$

Calculer la table $SLR(1)$ de cette grammaire puis, résoudre le conflit que l'on peut y observer de telle façon que la règle habituelle soit respectée : "un **sinon** est associé au dernier **alors** en suspens", c'est–à–dire, par exemple, que la phrase **i i a e a** soit analysée comme **i** [**i a e a**].

Serait–il possible de faire un choix autre que celui qui vient d'être fait?

(La grammaire

$$\begin{array}{lll} I \to J & J \to \mathbf{i}\ J\ \mathbf{e}\ J & K \to \mathbf{i}\ I \\ I \to K & J \to \mathbf{a} & K \to \mathbf{i}\ J\ \mathbf{e}\ K \end{array}$$

équivalente à la précédente, est $SLR(1)$ et prend en compte la convention précédente.)

Exercice 8. Récursion et pile d'analyse.

On considère les grammaires sur l'alphabet de terminaux composé des digits binaires **0** et **1** et du point "binaire" **.** pour représenter les rationnels en notation binaire et l'alphabet de variables $\mathcal{V} = S + E + D + B$ où S est choisie comme axiome :

G dont les règles sont

$1 : S \longrightarrow E\,\mathbf{.}\,D$	$3 : E \longrightarrow EB$	$5 : D \longrightarrow BD$	$7 : B \longrightarrow \mathbf{0}$
$2 : S \longrightarrow E$	$4 : E \longrightarrow B$	$6 : D \longrightarrow B$	$8 : B \longrightarrow \mathbf{1}$

et G' dont les règles sont celles de G à l'exception de 5 : qui est remplacée par 5' : $D \longrightarrow DB$.

Les grammaires sont assez simples et admettent une analyse ascendante : il est facile de simuler un analyseur de type LR pour traiter la question qui suit.

Faire une analyse LR de **1 . 0 0 1 1 1** dans les deux grammaires et comparer l'évolution des piles respectives.

(Il manque la définition intrinsèque des grammaires $LR(k)$)

Analyse LL.

(Il faudrait ajouter des exercices sur l'analyse descendante.)