Maîtrise d'Informatique
Année 2000-2001

# Projet TER

« Algorithme d'analyse syntaxique d'Earley »

Robert Cori

## 1. Principe

Cet algorithme a pour données une grammaire $\mathcal{G}$ et un mot $f$ engendré par celle-ci, il donne comme résultat une dérivation droite de $f$.

On peut comparer cet algorithme à l'analyse $LR(0)$ car il calcule des items que nous appelerons des $E$-items, qui s'apparentent aux 0-items de l'analyse $LR$. Toutefois ici ces items sont calculés lors de l'analyse de chaque mot ; rappelez vous dans l'analyse $LR$ les items sont calculés une fois pour toutes pour construire l'automate et l'algorithme d'analyse consiste à entrer le mot à analyser dans l'automate.

Pour l'algorithme d'Earley il n'y a pas d'automate, on construit des ensembles d'items $I_j$ où $j$ prend toutes les valeurs entre 0 et la longueur $n$ du mot à analyser. Une fois tous les ensembles d'items calculés on retrouve la dérivation en remontant les étapes du calcul. Il y a donc à priori une grande quantité d'items à conserver en mémoire et cela pourrait poser des problèmes de place si l'on ne trouve pas un moyen de mettre au rebut les items qui ne seront pas utilisés par la suite.

## 2. Les $E$-items

Soit $\mathcal{G}$ une grammaire, un *$E$-item* est un quadruplet $[X, \alpha_1, \alpha_2, i]$ dans lequel $X$ est une lettre auxiliaire $X \to \alpha_1\alpha_2$ est une règle de $\mathcal{G}$ et $i$ est un entier.

On dit que $[X, \alpha_1, \alpha_2, i]$ est un $E$-item, de niveau $j$, pour le mot $f = a_1a_2 \cdots a_n$ engendré par $\mathcal{G}$, si $0 \le i \le j \le n$ et si on a les dérivations suivantes :

$$S_0 \xrightarrow{*} a_1 \cdots a_i X u \quad \text{et} \quad \alpha_1 \xrightarrow{*} a_{i+1} \cdots a_j$$

L'ensemble des $E$-items de niveau $j$ pour $f$ sera noté $I_j(f)$ et on omettra le plus souvent le $f$ car il est implicite dans les cas considérés par la suite. De même on dira item pour $E$-item car il n'y a pas de confusion possible ici avec les 0-items de l'analyse $LR$ .

**3. Construction des $E$-items de niveau 0.**

Pour construire les $E$-items on commence par ceux de niveau 0 et on poursuit ensuite itérativement pour les niveaux $j$ de 1 à $n$.

Les items de niveau 0 sont donnés par la procédure suivante :

**Initialisation** Pour chaque règle de grammaire $S_0 \rightarrow \alpha$ concernant l'axiome $S_0$, $[S_0, \varepsilon, \alpha, 0]$ est un élément de $I_0$.

**Fermeture** Si $[X, \alpha_1, Y\alpha_2, 0]$ est dans $I_0$ alors ajouter $[Y, \varepsilon, \beta, 0]$ dans $I_0$ pour toutes les règles $Y \rightarrow \beta$.

**Réduction** Si $[X, \alpha_1, Y\alpha_2, 0]$ et $[Y, \beta, \varepsilon, 0]$ sont dans $I_0$ alors ajouter $[x, \alpha_1 Y, \alpha_2, 0]$ à $I_0$.

On notera que la réduction ne peut s'appliquer que pour des $\beta$ égaux à $\varepsilon$ (ou tels que $\beta \stackrel{*}{\rightarrow} \varepsilon$). D'autre part $I_0$ ne dépend pas du mot $f$ à analyser.

Ainsi pour la grammaire $E, T, F$ du cours on trouve :

$$I_0 \quad = \quad \{[E, \varepsilon, T, 0], [E, \varepsilon, E + T, 0], [T, \varepsilon, T * F, 0], [T, \varepsilon, F, 0], [F, \varepsilon, (E), 0], [F, \varepsilon, a, 0]\}$$

**4. Calcul des items de niveau $j$.**

On suppose les items de niveau $i < j$ connus, on a ensuite les mêmes phases que pour $I_0$

**Initialisation** On suppose que la $j$-ième lettre du mot $f$ à analyser est $a_j$. On initialise $I_j$ en utilisant les items $[X, \alpha_1, a_j\alpha_2, i]$ (dont la troisième composante commence par $a_j$) qui sont dans $I_{j-1}$ et on met $[X, \alpha_1 a_j, \alpha_2, i]$ dans $I_j$ (attention on ne change pas la valeur de $i$, quatrième composante de l'item !).

**Fermeture** Si $[X, \alpha_1, Y\alpha_2, i]$ est dans $I_j$ alors ajouter $[Y, \varepsilon, \beta, j]$ dans $I_j$ pour toutes les règles $Y \rightarrow \beta$ concernant $Y$ dans $\mathcal{G}$ (attention on a changé la quatrième composante de l'item ici !)

**Réduction** Si $[Y, \beta, \varepsilon, i]$ est dans $I_j$ et si $[X, \alpha_1, Y\alpha_2, k]$ est dans $I_i$ (attention la valeur de $i$ est donnée par $[Y, \beta, \varepsilon, i]$) alors ajouter $[x, \alpha_1 Y, \alpha_2, k]$ à $I_j$.

**5. Calcul de la dérivation**

Une fois calculés les $I_j$ pour $j = 0, \ldots, n$, le mot $f$ appartient au langage engendré par $\mathcal{G}$ si et seulement si aucun de ceux-ci n'est vide et si $[S_0, \alpha, \varepsilon, 0]$ appartient à $I_n$ pour un certain $\alpha$.

On appelle alors l'algorithme ci-dessous pour $\alpha$ tel que $[S_0, \alpha, \varepsilon, 0] \in I_n$ avec les paramètres suivants : Analyse $(X, \alpha, 0, n)$

**Algorithme**

ANALYSE $(X, \alpha_1\alpha_2\cdots\alpha_m, i, j)$ {
    $k = m; l = j;$
    TANT QUE $k \neq 0$ FAIRE {
        SI $\alpha_k$ TERMINAL ALORS $\{k - -; \quad l - -;\}$
        SI $\alpha_k = Y$ NON TERMINAL ALORS {
        CHERCHER UN COUPLE D'ITEMS
            $[Y, \beta, \varepsilon, p] \in I_l, \quad [X, \alpha_1\alpha_2\cdots\alpha_{k-1}, \alpha_k\cdots\alpha_m, i] \in I_p$ ;
        IMPRIMER(" $Y \to \beta$ ") ;
        ANALYSE $(Y, \beta, p, l)$ ;
        $\{k - -; \quad l = p;\}$
        }
    }
}

**Remarques :**

**1.** Attention, c'est les mêmes $X, \alpha$ et $i$ qui sont paramètres d'appel et qui interviennent dans $[X, \alpha_1\alpha_2\cdots\alpha_{k-1}, \alpha_k\cdots\alpha_m, i] \in I_p$.

**2.** L'algorithme ci-dessus peut sembler assez inéfficace en raison de la recherche d'un couple d'items, il vous est conseillé de réfléchir à l'aide d'exemples afin de trouver une structure de données pour les $I_j$ qui facilite le calcul de la dérivation. On pourra construire un graphe d'items, chaque item pointant sur ceux qui lui ont permis d'entrer dans $I_j$.

**Travail souhaité** Implanter l'algorithme d'Earley et faire des tests, on comparera les performances avec celles de de Yacc.

On utilisera les mêmes conventions que Yacc pour les données. On se limitera à l'analyse, l'ajout d'actions associées aux règles n'est à envisager que lorsque la partie d'analyse pure sera traitée.