Projet Maison

- De Tabbakh Mohamed Amine && Khaldi Jalal -

Notation : Pour tout le projet , soit $\sum$ un alphabet finis, nous désignerons A pour un Automate finis sur cet alphabet. $A = (S, T, I, F)$ avec S un ensemble fini (non-vide) d'états, $T \subset S \times (A \cup \{\epsilon\}) \times S$ un ensemble de transitions, $I \subset S$ les états initiaux et enfin $F \subset S$ l'ensemble des états finaux. De même pour le projet, nous étudierons que les Grammaires hors-contexte. Avec $G = (\sum, V, R, S)$ tel que $\sum$ désigne l'alphabet, V l'ensemble des symboles de variables, S l'axiome de départ $\in V$ et $R \subset V \times (\sum \cup V)^*$ l'ensemble des dérivations possibles. Dans la partie I et III: la constante $None$ désigne ϵ.

I. Construction d'une Grammaire régulière à partir d'un Automate fini :

Préambule : Nous travaillerons sur un Automate fini déterministe, pour celà avant toute chose, nous utiliserons si nécessaire la fonction fournis lors du premier TME afin de déterminiser ce dernier.

S l'ensemble des états est représenté par une liste d'état sous la forme $[q_1, .., q_n]$ avec $\forall i$ q_i une liste d'entier du type $[1, 2]$ ou $[3]$ représentant les états de l'automate après déterminisation. Ainsi en réalité S n'est en outre qu'une liste de liste d'entier. Quant à T, elle est représenté simplement par une liste de transitions. I est l'unique état initiale (sachant que l'automate est déterministe) ainsi I est une liste d'entier du type $[1, 3, 4]$. Enfin F sont les états finaux avec $F \subset S$. Quant à l'alphabet $\sum$ ce dernier est représenté par une simple liste.

Nous allons maintenant voir comment est créée la Grammaire. Avant tout de chose définissons notre fonction Id. L'on sait que pour tout état q de notre automate , ce dernier est codé sous forme d'une liste simple d'entier du type $["a_0", ..., "a_{n-1}"]$ avec $a_i \in \mathbb{N}$. Notre fonction id retourne pour $\forall q$ la valeur suivante :

$$id(q) = \sum_{i=0}^{n-1} a_i 10^i \tag{1}$$

Celà nous permet de nous assurer que deux états ayant le même valeur par la fonction id sont identiques et donc identifiable.
Nous allons maintenant voir les structures utilisées. $\sum$ n'est en outre que l'alphabet utilisé par l'Automate fini soit une liste et V également une liste que l'on notera L_v. L'on a que :

$$L_v = [\ id(q)\ \forall q \in S\ ]$$

Quant à l'axiome S ce dernier n'est que : $id(I)$ avec I l'unique état initial. Enfin les règles de dérivations R sont représenté dans un dictionnaire que l'on nommera L_r. Pour toutes transitions du type $(s_1, x, s_2) \in T$ avec $s_1, s_2 \in S$ et $x \in \sum$ nous allons créer la règle de dérivation suivante :

$$id(s_1) \to xid(s_2) \tag{2}$$

Cette règle sera codé dans une liste de la manière suivante $["x", id(S_2)]$ et sera rajouté à $L_r[id(s_1)]$.

ATTENTION : Si $s_1 \in F$ soit un état final de l'Automate, c'est la dérivation suivante : $id(s_1) \to \epsilon$ qui sera rajouté à $L_r[id(s_1)]$. La dérivation $id(s_1) \to \epsilon$ est codé par la liste suivante : $[chr(0), None]$. Quant à la constante $chr(0)$ cette dernière représente un symbole fictif, utilisé pour les besoin du codage, qui permet seulement de mettre à terme à la dérivation.

ex : Si l'on a l'Automate ayant trois états codant le mot "ab" avec les transitions suivantes (q_i, a, q_c) et (q_c, b, q_f). L'on aura donc la grammaire suivantes :

$$id(q_i) \to aid(q_c)$$
$$id(q_c) \to bid(q_f)$$
$$id(q_f) \to \epsilon$$

Et la dernière transition sera codé de la manière suivante : $[chr(0), None]$ stocké dans $L_r[id(q_f)]$.

Pour information : Si un état s de l'Automate n'est pas accessible, lors de la finalisation de la grammaire, l'on supprimera de L_r l'élément $L_r[id(s)]$ afin de néttoyer et alléger.

II. Construction d'un automate fini à partir d'une grammaire régulière:

Étudions tout d'abord les structures utilisées : les symboles de variables sont stockés dans une liste L_s. L'alphabet $\sum$ qui sera le même pour l'Automate et la Gramaire dans une simple liste. Quant aux règles de dérivation, elles sont stockées dans une liste L_r de la forme suivante : $\forall x \in V$ ou $x \in L_s$ (soit un symbole de variable), l'on a qu' $\exists i \in \mathbb{N}$ tel que $L_r[i] = (x, l_x)$ avec l_x l'ensemble des dérivations issues de x codées sous forme d'une liste de liste. Par exemple si l'on a $a, b \in \sum$ respectivement $X, Y, Z \in V$ tel que :

$$X \to aY \mid b$$

Et i_x désigne l'indice propre à X comme expliqué plus haut, l'on a la liste suivante : $[[\text{"}a\text{"}, \text{"}Y\text{"}], [\text{"}b\text{"}], None]$ dans $L_r[i_x]$. Quant à S s'est juste une chaine de caractère représentant l'Axiome.

Nous allons voir maintenant comment est créé l'Automate :
Pour les états, ces derniers sont représentés dans une liste que l'on nommera L_t qui est composé des symbole de variables de la Grammaire. Si par exemple $X \in V$ (soit un symbole de variable dans G) l'on aura l'état X dans l'Automate.
Pour l'état initial, ce dernier n'est en outre que l'Axiome S de la Grammaire.
Enfin pour l'unique état final que l'on nomera q_f ce dernier ne représente aucun symbole de variable.

Les règles de transitions sont codées à l'aide d'un dictionnaire Python (ou table de hachage) que l'on nommera T de la manière suivante : tel que $T[q]$ représente l'ensemble des transitions issues de q .
Soit q un état de l'Automate nous avons par disjonctions des cas, les possibilités suivantes :

Si on est dans le cas $q \to aq_1$ dans la Grammaire où a est une lettre de l'alphabet et q_1 un symbole de variable identiquement un état de l'Automate, l'on aura que la transition suivante (q, a, q_1). Donc (a, q_1) sera stocké dans $T[q]$.

Dans le cas où l'on a la transition suivante $q \to a$ alors c'est la transition (a, q_f) qui sera stocké dans $T[q]$ avec q_f désignons l'unique état final comme décrit plus haut.

Puis par la suite, dans le cas cas où q se dérive en $a_1...a_nq_z$ dans G. Dans ce cas l'on créera **si nécessaire** des états intermédiaires $q_1, ..., q_{n-1}$ et les transitions $(q, a_1, q_1)...(q_{n-1}, a_n, q_z)$ comme expliqué plus haut avec par exemple (a_1, q_1) dans $T[q]$...

Et enfin dans le cas où q se dérive en $a_1...a_n$, l'on a presque la même chose que précédamment sauf que l'on utilisera q_f l'unique état finale pour la dernière transition (q_{n-1}, a_n, q_f) stocké sous la forme (a_n, q_f) dans $T[q_{n-1}]$.

III. Savoir est-ce qu'une Grammaire est régulière :

Avant toute chose, l'on a que toute Grammaire régulière est hors-contexte.

Une grammaire est régulière droite si et seulement si ces dérivations sont de la forme :

$$V \times ((\sum)^* V \cup (\sum)^*)$$

Et gauche ssi de la forme :

$$V \times (V(\sum)^* \cup (\sum)^*)$$

Ainsi pour qu'une dérivation ne pose pas de problème, il faut qu'elle contienne **au plus** un symbole de variable et que selon les cas ce dernier se situe soit au début (gauche) soit à la fin (droite).

ATTENTION : Pour cette partie du code, nous avons exactement repris les notations du premier TME pour les Grammaires. En effet les dérivations sont codés de la manière suivante : soit R la liste des dérivations possible et $X \in V$ on a le tuple $(X, lr[x]) \in R$ avec $lr[X]$ les dérivations possibles depuis X et $lr[X]$. $lr[X]$ est donc une liste de liste.

Nous l'allons tout d'abord voir la vérification pour une Grammaire régulière droite puis gauche. L'on passe dérivation par dérivation.

Dans un premier temps pour la régularité droite :

L'on vérifie d'abord que l'on est pas dans le cas $X \to a$ avec a soit une lettre de l'alphabet ou ϵ. Dans ce cas l'on continue car cette dérivation ne pose pas de problème.
Puis l'on vérifie tous les cas sur cette dérivation qui rendrait la Grammaire **non-régulière droite**.
L'on vérifie d'abord qu'il y'a exactement un symbole de variable car par exemple si l'on $X \to aXaV$ avec X, V des symboles de variables alors cette Grammaire est déjà ni régulière droite ni gauche d'ailleurs.
Puis l'on vérifie que le dernier élément de la liste est une variable et que cette dérivation n'est pas consituté uniquement que de lettre de l'alphabet car si l'on a par exemple $X \to abc$ une telle transition ne pose pas de problème.
Si telle est le cas, la dérivation contient au plus un symbole de variable et si elle en contient un, ce dernier se situe à droite. L'on peut donc passer à la dérivation suivante. Une Grammaire validant tous les tests sur toutes ses dérivations est donc régulière droite.

De même pour la régularité gauche, à exception que l'on vérifie que l'unique symbole de variable "s'il existe", c'est à dire que l'on est pas dans le cas constitué uniquement de lettre de l'alphabet ou une epsilone transition du type : $X \to abc$ ou $X \to \epsilon$, se trouve en premier soit à gauche de la dérivation (au rang $lr[0]$ si lr est la liste représentant une dérivation depuis une certaine variable.)