Compte - rendu : Evaluation d'expressions arithmétiques

JACQUET Julien 21400579

27 mai 2018

Note : *cr.tex* est situé dans le sous-dossier *cr*. *eval.c* et $Makefile$ sont situés dans le sous-dossier *src*.

Question 1

Soit une grammaire G définie par $G = (\Sigma, V, S, P)$ telle que :
$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /, (,)\}$
$V = \{S\}$
$S = \{S\}$

$P = \{S \to (S),$
$S \to \varepsilon,$
$S \to S + S,$
$S \to S - S,$
$S \to S * S,$
$S \to S/S,$
$S \to n\}$

$n =$ est un nombre quelconque.

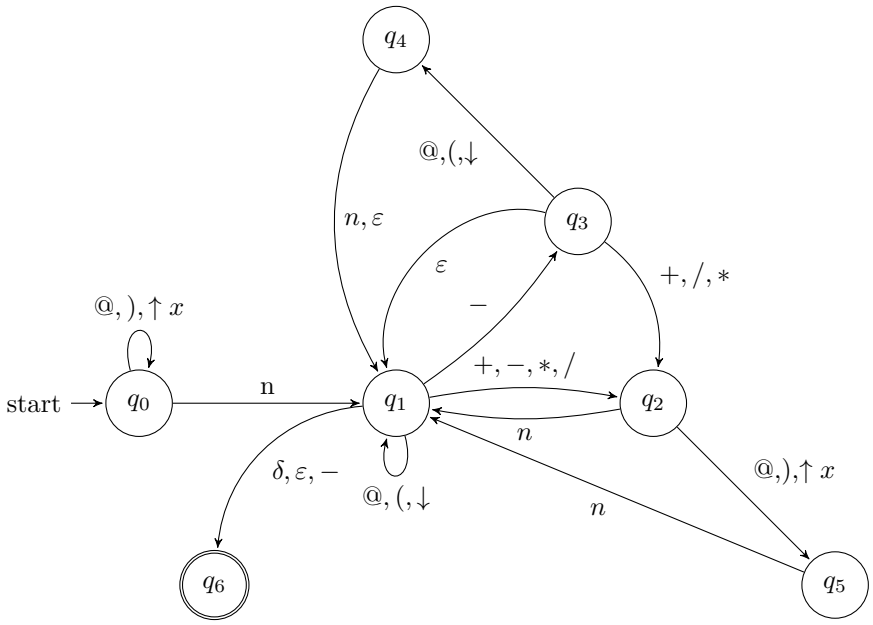
Question 3 :

Etant donné la nature des symboles donnés pour notre langage, il faut renommer certains éléments de notre langage :

$s = \{+, *, /\}$
$m = \{-\}$
$p_o = \{(\}$
$p_f = \{)\}$
$n = nombre$
$L = \{(p_o\ (m + \varepsilon)(s + \varepsilon)\ n\ (m + \varepsilon)\ s\ n\ p_f) + ((m + \varepsilon)(s + \varepsilon)\ n\ (m + \varepsilon)\ s\ n) + n\ \}$

note : Une fois une partie de l'equation evaluée elle devient un n rendant le parenthèsage "récursif". Le n à la fin du langage est présent pour éviter que notre équation soit vide.

On obtient l'automate suivant, avec une pile.
$\Gamma = \{x\}$
δ symbole pile vide.
@ peut importe ce qu'il y a dans la pile (x>=0).
La reconnaissance se fait par pile vide et état final.
On suppose que la pile est vide en q_0.
Il est impossible de pop si la pile est vide $\to$ Rejet.
Le $-$ dans la transition de q_1 à q_6 signifie que la pile est simplement lue mais pas changée.
Enfin la pile n'est pas utilisée pour toutes les transitions.



Question 5 :

Je n'ai pas résusi à transformer mon arbre en int.
Du coup pour evaluer mon expression je vais utiliser une pile d'évaluation.
On prends chaques éléments de la liste de token et on les ajoute à la pile , lorsque l'on rencontre une parenthèse fermante on evalue le contenu de la pile jusqu'à la première parenthèse ouvrante. On vide les tokens du bloc de parenthèses, et on push la valeur numérique resultant de l'évaluation du bloc. Une fois que la liste de token à été complètement chargée dans la pile, on fini d'évaluer la pile jusqu'à ce qu'elle soit vide. Le dernier token présent dans la pile sera un entier égal au resultat de l'évaluation de l'expression entière.

Question 6 :

En théorie pour prendre en compte les espaces présents dans l'expression il faudrait simplement fusionner tous les *argv* en une seule expression continue. Cependant les parenthèses ont une signification spéciale pour le *shell* et mettre des espaces entre des blocs de parenthèse entrainera des erreurs de *shell*. Il y a deux solutions pour contourner ce probleme :
Utiliser des "escapes" : ‘*expression*‘ (accents graves(alt-gr + 7)) ou des guillemets ”*expression*” ou meme ′*expression*′. Il est toujours possible d'utiliser des espaces tant que chaque fragment est encadré.

Tests :

Le premier test est celui donné dans l'énoncé.
Le second test est pour montrer un exemple inhabituel mais syntaxiquement

correct.
Le troisième test est pour montrer un résultat négatif.
Le quatrième test est pour montrer des nombres plus grands.
Le cinquième test est le même que le premier mais avec une parenthèse inversée, la syntaxe est donc NON conforme.