Évaluation d'expressions booléennes

LAZZARI-ARMOUR Raphael et POULIQUEN Chloé

Mai 2021

1 Question 1

Question: Lire une chaîne de caractère contenant une expression arithmétique et la transformer en une liste de tokens.

Pour cette première question, nous avons eu besoin de définir la structure suivante. Le but du projet étant sur les "expressions booléennes", nous avons décidé de faire une structure composé uniquement de booléen. La taille d'un booléen étant de 1 bit, cela nous fait par la même occasion économiser sur la mémoire allouée pour la liste par la suite.

```
struct token{
  struct token* next;
  type TYPE;
  union{
    bool value;             //if TYPE is a CONSTANT, value is 'true' if '1' else 'false' if '0'
    bool bracket;           //if TYPE is a BRACKET, value is 'true' if '(' else 'false' if ')'
    bool unary;             //if TYPE is a UNARY_OP, value is 'true' if 'NON' else 'false'
    op_binary OpBinary;     //if TYPE is a BINARY_OP, value depends on enum op_binary
  }attribute;
};
typedef struct token* TokenList;
```

L'appel de la fonction $TokenList\ string\_to\_token(char * string)$ dans le main nous permet de convertir la chaîne de caractère entrée par l'utilisateur en une liste de token. Cette fonction prends en paramètre une chaîne de caractères et retourne une liste de tokens.

2 Question 2

Question: Donner un automate à pile reconnaissant le langage dont les mots sont les expressions booléennes.

Pour cette question, nous proposant un automate reconnaissant le langage par état final et par pile vide. Une pile qui empile '(' et dépile ')'.

Règle sur les constantes '0' et '1': Doit contenir un opérateur binaire, unaire ou '(' avant mais pas une autre constante. Peut être suivi par un opérateur binaire ou ')' mais pas par une constante.

Règle sur les opérateurs binaires '+' , '.' , '⇒' et '⇔' : Doit contenir une constante ou ')' avant et doit contenir une constante ou '(' après. Exception pour l'opérateur '¡=¿' qui doit contenir une expression booléenne complète avant. On doit donc vérifier que la pile est vide.

Règle sur l'opérateur unaire 'NON' et '(': Ne peut pas contenir une constante avant. L'opérateur 'NON' doit être suivi par une constante ou par l'opérateur '('. Il ne peut ainsi pas être suivi par un opérateur binaire ou par ')'.

Après étude de ces règles, on peut définir l'automate suivant. La transition δ , $\Leftrightarrow$, δ est seulement possible si la pile est vide.

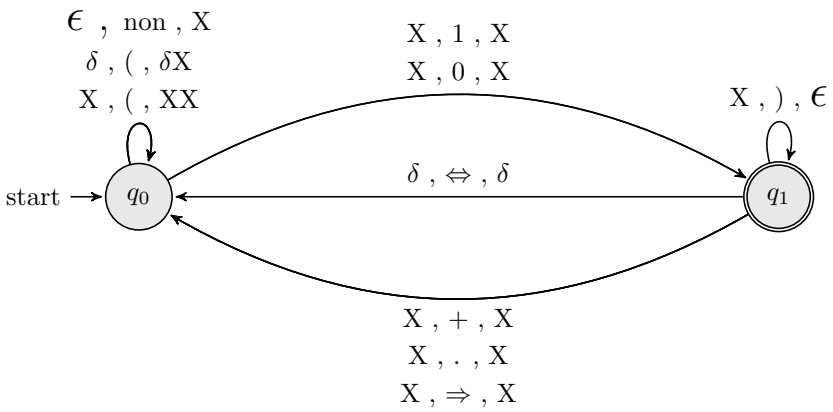

Figure 1: Automate à pile qui reconnaît le language par état final et par pile vide.

3 Question 3

Question: Écrire une fonction en langage C qui teste si une liste de token appartient au langage ou non.

4 Question 4

Question: À partir de la liste de tokens et en utilisant l'automate à pile, créer l'arbre représentant l'expression booléenne. Vous pouvez utiliser la fonction qui teste si la liste des tokens appartient au langage en la modifiant.

5 Question 5

Question: Calculer la valeur de l'expression arithmétique et afficher le résultat.