Évaluation d'expressions arithmétiques

IN406 - Theorie des langages

CHIKAR Soufiane LEFEBVRE Theo

Mai 2019

Table des matières

1 Introduction

Le but du projet c'est d'évaluer les expressions booléennes sous forme infixe avec des parenthèses.

Nous allons détailler la démarche de création d'un programme ayant pour but de construire et de parcourir un arbre binaire des expressions arithmétiques fondées sur une grammaire.

Nous procédons par questions séparées afin de bien faire comprendre le processus de conception du programme.

2 Les Questions

2.1 Question 1

Donner une grammaire reconnaissant le langage dont les mots sont les expressions arithmétiques définies ci-dessus.

On peut décrire le langage des expressions arithmétiques par les règles syntaxiques suivantes :

- Une expression est un nombre, ou
- le OU de deux expressions, ou
- l'implication de deux expressions, ou
- le ET de deux expressions, ou
- l'equivlence de deux expressions, ou
- une expression entre parenthèse.

<E> : := <E> + <T>
<E> : := <E> <=> <T> || E> => <T>
<E> : := <T>

<T> : := <T>*<F>
<T> : := <F>

<F> : := *true,false*
<F> : := (<E>) || !(<E>)

<E>,<T> et <F> sont des expressions et des symboles non-terminaux.
+,<=>,=>,*,(,) et *num* sont des symboles terminaux.
<E> est le symbole de départ.

2.2 Question 2

Lire une chaîne de caractère contenant une expression booléennes et la transformer en une liste de tokens.

```
typedef enum token_type_s
{
    OPERATOR,
    CONSTANTE,
    NON,
    PARENTHESE_DROITE,
    PARENTHESE_GAUCHE,
    UNKNOWN
} token_type_t;


typedef struct token_s
{
    char data;
    token_type_t type;
    struct token_s *next;
} token_t;
typedef struct token_s *token_list_t;


token_list_t expression_to_token_list(char *expression)
{
    token_list_t token_list = NULL;
    int len = strlen(expression);
    int i = 1;

    if (get_token_type(expression[i]) != UNKNOWN)
        token_list = init_token_list(expression[0], get_token_type(expression
  [0]));

    while (i < len)
    {
        if (get_token_type(expression[i]) != UNKNOWN)
            token_list = append_token_list(token_list, expression[i],
  get_token_type(expression[i]));
        i++;
    }
    return token_list;
}
```

2.3 Question 3

À partir de la liste de tokens, vérifier si cette liste est correspond à une expression booléenne bien formée.

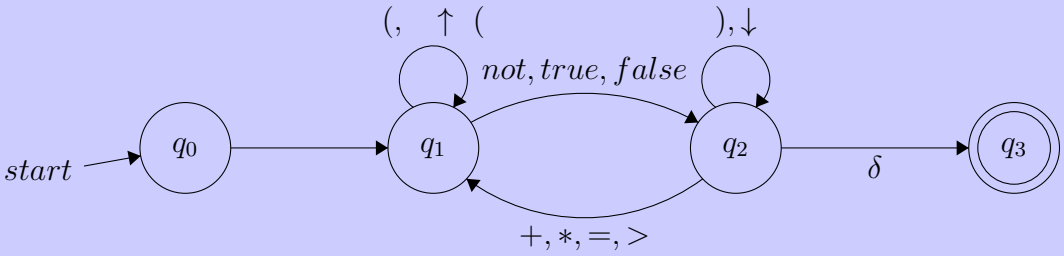
```
int check_token_list(token_list_t head)
{
    int NB_LEFT_PARENTHESE = 0;
    int NB_RIGHT_PARENTHESE = 0;
    int NB_OPE_SUC = 0;
    token_list_t cursor = head;
    while (cursor != NULL)
    {
        /* On verifie le nombre de parenthese */
        if (cursor->type == PARENTHESE_GAUCHE)
            NB_LEFT_PARENTHESE += 1;

        if (cursor->type == PARENTHESE_DROITE)
            NB_RIGHT_PARENTHESE += 1;

        /* On verifie s'il n'y a pas d'erreur de syntaxe, double operateur par
     * exemple */
        if (cursor->next != NULL)
        {
            if (cursor->type == OPERATOR && cursor->next->type == cursor->type
)
                NB_OPE_SUC += 1;
            if (cursor->type == CONSTANTE && cursor->next->type == cursor->
type)
                NB_OPE_SUC += 1;
            if (cursor->type == NON && cursor->next->type == OPERATOR)
                NB_OPE_SUC += 1;
            if (cursor->type == NON && cursor->next->type == PARENTHESE_DROITE
)
                NB_OPE_SUC += 1;
            if (cursor->type == PARENTHESE_GAUCHE && cursor->next->type ==
OPERATOR)
                NB_OPE_SUC += 1;
        }
        else if (cursor->type == OPERATOR && cursor->next == NULL)
        {
            NB_OPE_SUC += 1;
        }
        cursor = cursor->next;
    }
    if (NB_OPE_SUC == 0 && NB_LEFT_PARENTHESE == NB_RIGHT_PARENTHESE)
    {
        return 1;
    }
    else
        return 0;
}
```

Définir le langage qui est l'ensemble des mots qui sont une liste de tokens correspondant à une expression arithmétique correcte. Vous devez donner l'alphabet et donner l'automate (éventuellement à pile) reconnaissant ce langage.

On considère l'alphabet A = {*true*,*false*,+,*,<=>,=>} et le langage L = {w ∈ Σ* | w est une expression arithmétique} :



2.4 Question 4

À partir de la liste de tokens créer l'arbre représentant l'expression arithmétique.

```c
typedef struct token_tree_s token_tree_t;
struct token_tree_s {
  token_t token;
  token_tree_t *parent;
  token_tree_t *left;
  token_tree_t *right;
};
token_tree_t *insert_token_tree(token_tree_t *root, token_t token) {
  if (root == NULL) {
    root = new_tree(token);
  } else {
    token_tree_t *cursor = root;
    while (cursor != NULL) {
      // Seul les operateurs peuvent avoir des fils
      if (cursor->token.type == OPERATOR) {
        if (cursor->left == NULL) {
          cursor->left = new_tree(token);
          return root;
        } else if (cursor->right == NULL) {
          cursor->right = new_tree(token);
          return root;
        } else {
          if (cursor->left->token.type == OPERATOR) {
            cursor = cursor->left;
            continue;
          } else if (cursor->right->token.type == OPERATOR) {
            cursor = cursor->right;
            continue;
          } else {
            cursor = cursor->parent->right;
            continue;
          }
        }
      } else {
        perror("Error, only nodes who hold operators can have children.");
        exit(EXIT_FAILURE);
      }
    }
  }
}
```

2.5 Question 5

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

```
bool evaluate_tree(token_tree_t *treenode)
{
    bool x, y;

    if (treenode->token.type == OPERATOR)
    {
        x = evaluate_tree(treenode->left);
        y = evaluate_tree(treenode->right);

        if (treenode->token.data == '*')
            return et(x, y);
        else if (treenode->token.data == '+')
            return ou(x, y);
        else if (treenode->token.data == '=')
        {
            return eqv(x, y);
        }
        else if (treenode->token.data == '>')
            return imp(x, y);
    }
    else
    {
        return atoi(&treenode->token.data);
    }
}
```

2.6 Question 6

Le programme que vous devez rendre doit prendre en argument la chaîne de caractère et afficher son évaluation ou bien "éxpression incorrecte".

Voir Code.