Université d'Aix-Marseille III - Licence Math-Info, 2ème année

I5 : Langages formels, automates et grammaires - TP n^o 4

Ruban infini

Les machines de Turing utilisent un ruban infini constitué d'une suite de cases dont certaines sont vides et d'autres contiennent un symbole. La tête de lecture de la machine pointe sur une case du ruban. Les opérations que peut effectuer la machine sur le ruban sont : lire le symbole de la case pointée par la tête de lecture de la machine, écrire un symbole dans la case pointée par la tête de lecture, se déplacer d'une case vers la gauche et se déplacer d'une case vers la droite. Comme il n'est évidemment pas possible de créer un ruban contenant un nombre infini de cases, on va définir un ruban contenant un nombre fini de cases mais infiniment extensible à chaque bout. Toutes les cases qui ne font pas partie du ruban (fini) sont considérées comme étant vides. Si on veut se déplacer à droite (resp. gauche) de la case la plus à droite (resp. gauche) du ruban, on étend le ruban d'une case à droite (resp. gauche) puis on s'y déplace.
Pour représenter une case en C, on définit le type suivant :

```
typedef struct chainon{
   struct chainon* gauche;
   struct chainon* droite;
   char symbole;
} Case;
```

Si le champ `gauche` (resp. `droite`) contient `NULL`, ça signifie qu'il n'y a pas de case à gauche (resp. droite), c'est-à-dire que toutes les cases situées à gauche (resp. droite) sur le ruban infini sont vides.

Ecrivez les fonctions suivantes :

- `Case* nouvelle_case(Case* gauche, Case* droite)` : crée une nouvelle case vide (une case contenant le caractère ' ') dont `gauche` (resp. `droite`) est l'adresse de la case qui est à sa gauche (resp. droite).
- `char lis(Case* tete)` : retourne le symbole pointé par la tête de lecture.
- `void ecris(Case* tete, char symbole)` : écris un symbole dans la case pointé par la tête de lecture.
- `Case* deplace_droite(Case* tete)` : retourne l'adresse de la case située à droite de la tête de lecture en ayant créé une case vide s'il n'existait aucune case à droite.
- `Case* deplace_gauche(Case* tete)` : idem mais à gauche.
- `Case* cree_ruban(char* s)` : crée un ruban dont chacune des cases contient un caractère de la chaîne `s` et retourne l'adresse de la case la plus à gauche.
- `void affiche_ruban(Case* tete)` : affiche la suite complète des caractères du ruban dont on a fourni l'adresse de la case pointée par la tête de lecture.

Machine de Turing (déterministe)

Ecrivez la fonction `int accepte(Case* ruban, transitions delta, int final[Q], int etat)` *qui détermine si le mot présent sur le ruban est accepté par la machine de Turing dont l'état initial est* `etat`*, la fonction de transition est* `delta` *et l'ensemble des état terminaux est mémorisé par le tableau de booléens* `final` . On pourra définir les types `QXGXS` et `transitions` ainsi :

```
typedef struct{
   int etat;
   char symbole;
   char sens;
} QXGXS;
typedef QXGXS transitions[Q][128];
```

`QXGXS` désigne le type des éléments de l'ensemble $Q \times \Gamma \times S$, avec $S = \{+,-\}$. Si une variable `delta` est de type `transitions` alors `delta[i][c] == qgs` signifie qu'en lisant le caractère dont le code ascii (compris entre 0 et 127) est `c` à partir de l'état q_i on arrive à l'état q_k avec $k =$ `qgs.etat`, on inscrit le caractère `qgs.symbole` à la place de `c` et on déplace la tête de lecture dans la direction indiquée par `qgs.sens`.

Voici comment on peut alors utiliser la fonction dans un programme qui prend en argument le mot initialement inscrit sur le ruban et qui lit la définition d'une machine de Turing sur l'entrée standard :

```
main(int argc, char** argv)
{
   char s, s2, sens;
   int final[Q], e, e2, i, j;
   Case* ruban;
   transitions delta;
   QXGXS rien = {-1, ' ', '+'};

   for(i = 0; i < Q; i++)
      for(j = 0; j < 128; j++)
         delta[i][j] = rien;
   for(i = 0; i < Q; i++)
      final[i] = 0;
   scanf("%d", &e);
   final[e] = 1;
   do
   {
      scanf("%d,%c->%d,%c,%c", &e, &s, &e2, &s2, &sens);
      if(e != -1)
      {
         delta[e][s].etat = e2;
         delta[e][s].symbole = s2;
         delta[e][s].sens = sens;
      }
   }while(e != -1);
   ruban = cree_ruban(argv[1]);
   printf("%d\n", accepte(ruban, delta, final, 0));
   affiche_ruban(ruban);
}
```

Voici un exemple de fichier d'entrée où est défini une machine de Turing incrémentant un entier binaire. La première ligne contient le numéro de l'unique état final de l'automate (on n'autorisera qu'un seul état final) et le reste des lignes contient toutes les transitions. La dernière ligne contenant "-1" signale qu'il n'y a plus de transitions. L'état initial n'est pas indiqué car on considérera qu'il porte obligatoirement le numéro 0.

```
2
0,0->0,0,+
0,1->0,1,+
0, ->1, ,-
1,1->1,0,-
1,0->2,1,+
1, ->2,1,+
-1
```

Si on appelle `inc.mt` le fichier texte précédent et que le nom du fichier exécutable, dont la fonction `main` a été donnée plus haut, est `MT` alors la commande `$ MT 1011 < inc.mt` affiche "1" (le mot 1011 a été accepté) puis "`1100`", qui est ce qui est inscrit sur le ruban lorsque la machine s'arrête.

Soustracteur

Définissez une machine de Turing effectuant la soustraction entre deux entiers binaires. Le format de l'entrée est : `...-...=`*, où ... représente une suite quelconque de 0 et de 1. Lorsque la machine s'arrête, le résultat est écrit après le symbole = (et le reste des cases est inchangé). Testez votre machine avec le programme.*