INF203 - Travaux pratiques, séance 9
Automates

1 Une machine à café rudimentaire

`[TP9]` Votre répertoire `TP9` contient différents sous-répertoires correspondant aux parties de ce TP. Dans ces différentes parties, nous allons voir plusieurs manières de coder la simulation d'un automate, de la plus spécifique et difficile à faire évoluer à la plus générale.

1.1 Codage de l'automate - initialisation "en dur"

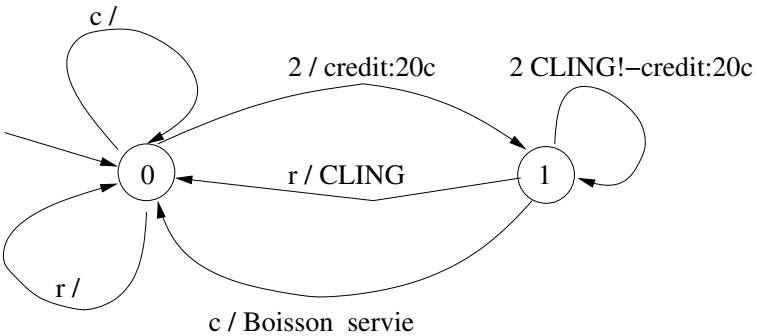
`[Cafe1]` Prenez connaissance des fichiers de ce répertoire. Le type `automate` est défini dans `automate.h`. Pour chaque automate, il comporte :

- son nombre d'états `nb_etats` ;
- le numéro de l'état initial `etat_initial` ;
- un tableau `etats_finals` indicé par les numéros d'états, tel que `etats_finals[i]` vaut `1` si l'état numéro `i` est un état final, `0` sinon, ;
- un tableau `transitions` représentant la fonction de transition de l'automate : `transition[i][j]` est l'état obtenu à partir de l'état `i` lorsque le symbole d'entrée est `j` ;
- un tableau `sortie` représentat la fonction de sortie de l'automate : `sortie[i][j]` est le message de sortie émis lors de la transition `transition[i][j]`.

Dans notre type `automate` :

- les états sont des entiers dans l'intervalle $[0; \texttt{nb\_etats} - 1]$, leur valeur sert de premier indice aux tableaux `transition` et `sortie` ;
- les symboles d'entrée sont des caractères Ascii, leur code Ascii sert de second indice aux tableaux `transition` et `sortie` ;
- les symboles de sorties sont des chaînes de caractères destinées à être affichées à l'écran.

Lisez le contenu du fichier `automate.c`. La fonction `init_par_defaut` initialise tous les champs de l'automate :
[a] Par défaut, vers quel état amènent les transitions partant de l'état `i` quelle que soit l'entrée ? ■
La fonction `init_mon_automate` modifie l'automate préalablement initialisé par défaut, L'automate obtenu est le suivant, vérifiez que vous comprenez le codage.



[b] Quels sont les caractères d'entrée des transitions de cet automate ? A quelles actions d'un utilisateur devant une "vraie" machine à café correspondent-ils ? ■
Complétez la fonction `simule_automate`, comme vu en TD et rappelé en commentaire.
Compilez l'exécutable `test_automate` à partir des fichiers `automate.c`, `automate.h` et `main.c`. Testez-le.
[c] Que se passe-t-il lorsque vous saisissez un caractère d'entrée non prévu ? ■
Que faut-il faire pour terminer le programme ? Pour remédier à ce problème, vous pouvez choisir de terminer

la simulation lorsque l'utilisateur saisit un caractère de votre choix (par exemple `'q'`) : modifiez la fonction `simule_automate` en conséquence.
[**d**] Joignez le texte de la fonction `simule_automate` à votre compte rendu. ■

1.2 Lecture de l'automate dans un fichier

`[Cafe2]` Lisez le fichier `Mon_automate.auto` : il représente l'automate de la question précédente. le format de fichier choisi pour représenter les automates est le suivant :

- le premier entier est le nombre d'états (ici `2`) ;
- l'entier suivant est le nombre d'états finals (ici `0`) ;
- `n` étant le nombre d'états finals, on trouve ensuite `n` entiers qui sont les numéros de ces états (ici, il n'y en a pas) ;
- puis un entier : le nombre de transitions (ici `6`) ;
- puis une ligne par transition, chacune de la forme `état_départ entrée_caractère état_arrivée` ;
- puis un entier : le nombre de sorties (ici `4`) ;
- et enfin, une ligne par sortie, chacune de la fome `état_départ entrée_caractère message_de_sortie`.

Plusieurs choses sont implicites (non précisées dans le format du fichier) :

- le numéro de l'état initial : ce sera toujours `0` ;
- les transitions non précisées sont celles mises en places par `init_par_defaut` ;
- les transitions pour lesquelles aucune sortie n'est spécifiée ont une sortie vide.

Copiez les fichiers `.c` et `.h` de `Cafe1` dans le répertoire courant (`Cafe2`). Vous allez maintenant modifier votre programme afin de lire l'automate dans un fichier.

1.2.1 Fonction de lecture

`[Cafe2]` Dans `automate.c`, écrivez une fonction de prototype `void lecture_automate(automate *A, FILE *f)` qui initialise l'automate avec la fonction `init_par_defaut` puis lit dans le fichier (ouvert en lecture) désigné par `f` un automate suivant le format décrit ci-dessus. À titre indicatif, voici des instructions permettant de lire le nombre de transitions et les transitions que vous pouvez recopier dans votre fonction :

```
int nb_trans, i;
int depart, arrivee;
char symbole_entree;
int entree;

/* ici il faut appeler init_par_defaut, puis lire le nombre d'états,
le nombre d'états finals, et tous les états finals  */

        fscanf(f, "%d", &nb_trans);
        for (i=1 ; i<= nb_trans ; i++) {
                fscanf(f, "%d %c %d", &depart, &symbole_entree, &arrivee);
                entree = symbole_entree;
                A->transitions[depart][entree] = arrivee ;
                A->sortie[depart][entree][0] = '\0' ;
        }

/* terminer par la lecture des sorties */
```

Commentaire :

- `entree = symbole_entree;` permet de stocker la valeur de `symbole_entree` de type `char` (1 octet, type nécessaire pour la lecture au format `%c`) dans une variable de type `int`, type des indices de tableau ;
- `A->sortie[depart][(int)entree][0] = '\0';` initialise par défaut la sortie associée à la transition avec la chaîne vide, puisque les sorties vides ne sont pas indiquées dans le format d'entrée.
  [**e**] Quelle serait la sortie sans cette instruction ? ■

Complétez la fonction `lecture_automate`.
[**f**] ... et joignez-la à votre compte rendu (vous pouvez ne pas recopier les lignes fournies, remplacez les par `[...]`). ■

1.2.2 Modifications à apporter au reste du programme

Ajoutez le prototype de `lecture_automate` à `automate.h`, puis modifiez le programme principal :
- ajoutez les paramètres de la focntion `main` permettant de récupérer les arguments de la ligne de commande ;
- le programme attend un nom de fichier sur la ligne de commande : vérifiez que le nombre d'arguments est correct (sinon message d'erreur et arrêt du programme) ;
- ouvrez (en lecture) le fichier dont le nom est donné en argument et remplacez l'appel à `init_mon_automate` par un appel à `lecture_automate`. Testez : l'automate est le même qu'avant, il doit se comporter de la même façon.

1.2.3 Une récréation qui n'a rien à voir

[g] Comment faire pour connaître la taille (en octets) d'une variable de type automate ? Quelle est cette taille ? ■

2 Votre propre machine à café

Dessinez l'automate d'une machine à café plus élaborée : elle accepte des pièces de 10 et de 20 centimes, elle délivre du café ou du thé, et les boissons coûtent 30 centimes (c'est l'inflation).
[h] Lorsque l'automate est au point, dessinez-le sur votre compte rendu. ■
Ecrivez votre automate dans un fichier suivant le format indiqué. Vérifiez bien que vous ne vous êtes pas trompés ... puis simulez votre machine à café.

3 Un automate mystère

Placez-vous dans le répertoire `[Mystere]`, et copiez-y les fichiers `.c` et `.h` de `Cafe2`.
`[Mystere]`. Compilez et exécutez `./test_automate mystere.auto`. Les symboles d'entrée de cet automates sont des lettres majuscules. Sauriez-vous dessiner l'automate (sans vous aider du fichier `mystere.auto`) ? Si non, lisez `mystere.auto`.
[i] Quels sont les états finals de cet automate ? ■
Dans `automate.c`, modifiez la condition d'arrêt de la boucle de `simule_automate` pour s'arrêter lorsqu'un état final est atteint, et testez cette nouvelle version avec `mystere.auto`.