INF203 : Compte-rendu TP9
Automates

Alexandre Dupré, Maxime Jaunatre, Clément Raspail | INF - 3
Mail | 6 avril 2021

Syntaxe

Pour ce compte rendu la syntaxe des commandes sera la suivante :

```
[~chemin]: commande
retour de la commande
```

Example :

```
[~/INF203]: ls
sauve_TP1 TP1 TP2
```

Si la commande est interactive et demande d'appuyer sur entrée, une caractère '->' est indiqué. Les fichiers sont en *italique* et les commandes (ou détails de retour de commande) en **gras**. Un script sera donc en gras quand il sera appelé comme une commande.
Les fichiers sources en C sont compilés avec clang, et un nom est donné avec l'option **clang -o**. Cela implique que les programmes seront appelés par un autre nom que **a.out**.

1 Une machine à café rudimentaire

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

[a] Par défaut, n'importe quel transitions partant de l'état i reviendra sur i quelque soit l'entrée.
[b] Les 3 entrées de cet automates est : 'c', 'r' et '2'. 'c' correspond au service du café, 'r' pour rendre la monnaie et '2' quand on insert une piece de 20 centimes.
[c] Quand on saisi un caractère non prévu, le terminal renvoie "entree_invalide". Pour terminer le programme il faut l'arrêter avec **Ctrl-C**.

```
[~/INF203/TP9]: clang Cafe1/*.c -o caf1
[~/INF203/TP9]: ./caf1
2
credit:20c
2
CLING!-credit:20c
c
Boisson_servie
2
credit:20c
r
CLING!
a
entree_invalide
q
```

[d]

```
[~/INF203/TP9]: tail -n 13 Cafe1/automate.c
void simule_automate(automate * A) {
  int etat_courant, etat_suivant;
  int entree = ' ';

  etat_courant = A->etat_initial;
  entree = lire_entree();
  while (entree != 'q') {
    etat_suivant = A->transitions[etat_courant][entree];
    printf("%s\n", A->sortie[etat_courant][entree]);
    etat_courant = etat_suivant;
    entree = lire_entree();
  }
}
```

1.2 Lecture de l'automate dans un fichier

1.2.1 Fonction de lecture

[e] Sans cette instruction, la sortie par défaut serait "entree_invalide".
[f]

```
[~/INF203/TP9]: head -n 53 Cafe2/automate.c | tail -n 34
void lecture_automate(automate *A,FILE *f){
  int nb_trans, i, s, nb_sorties;
  int depart, arrivee;
  char symbole_entree, sorties[LG_MAX_SORTIE];
  int entree;

  // init
  init_par_defaut(A);
  // nb etats
  fscanf(f, "%d", &A->nb_etats);
  // etat finaux
  fscanf(f, "%d", &i);
  for (int j = 0; j < i; j++){
    fscanf(f, " %d", &s);
    A->etats_finals[s] = 1;
  }
  // etat de transition
  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' ;
  }
  // message sorties
  fscanf(f, "%d", &nb_sorties);
  for (i=1 ; i<= nb_sorties ; i++) {
    fscanf(f, "%d %c %s", &depart, &symbole_entree, sorties);
    entree = symbole_entree;
    strcpy(A->sortie[depart][entree], sorties);
  }
}
```

1.2.2 Modifications à apporter au reste du programme

```
0  [~/INF203/TP9]: cat Cafe2/main.c
1  #include <stdio.h>
2  #include "automate.h"
3
4  int main(int argc,char* argv[]){
5          automate A ;
6          FILE *f;
7          // init_mon_automate(&A);
8          if (argc != 2){
9                  fprintf(stderr,"Le programme requiert un nom de fichier\n");
10                 return 1;
11         }
12         f = fopen(argv[1], "r");
13         if (f == NULL){
14                 fprintf(stderr,"Le fichier n'existe pas\n");
15                 return 2;
16         }
17         lecture_automate(&A, f);
18         simule_automate(&A);
19         fclose(f);
20         return 0 ;
21 }
```

1.2.3 Une récréation qui n'a rien à voir

[g] On peut avoir la taille de l'automate en rajoutant la ligne suivant dans le fichier *main.c* : **printf("%lu\n", sizeof(&A));**

2 Votre propre machine à café

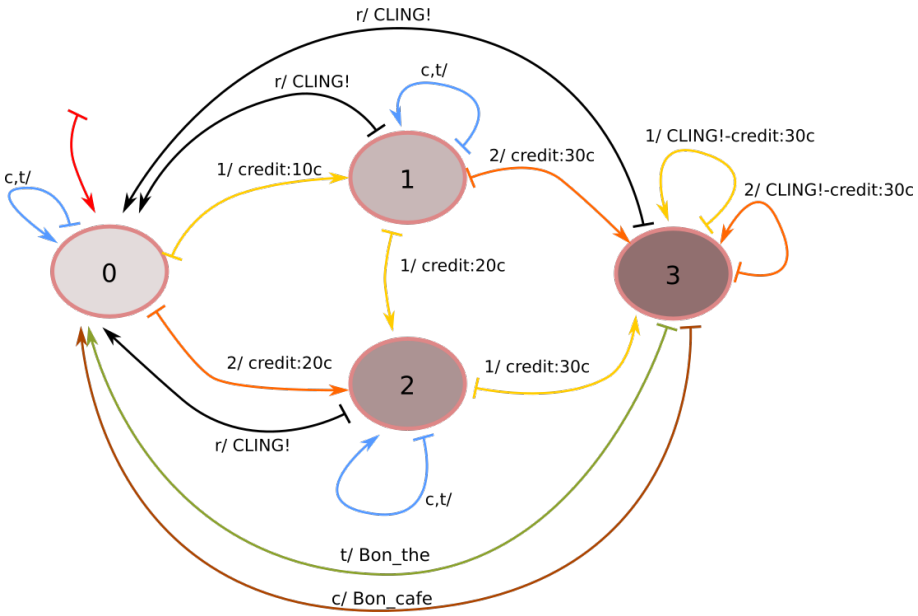

FIGURE 1 – Automate de machine à café. "`x/ sortie`" représente l'entrée `x` et la `sortie` correspondante, pouvant être nulle dans certains cas.

[h]

```
[~/INF203/TP9]: cat Cafe2/Automate_the.auto
4
0
18
0 c 0
0 t 0
0 1 1
0 2 2
1 c 1
1 t 1
1 r 0
1 1 2
1 2 3
2 c 2
2 t 2
2 r 0
2 1 3
3 1 3
3 2 3
3 r 0
3 c 0
3 t 0
12
0 1 credit:10c
0 2 credit:20c
1 1 credit:20c
1 2 credit:30c
1 r CLING!
2 r CLING!
2 1 credit:30c
3 r CLING!
3 1 CLING!−credit:30c
3 2 CLING!−credit:30c
3 c Bon_café
3 t Bon_thé
```

```
[~/INF203/TP9]: ./caf2 Cafe2/Automate_the.auto
2
credit:20c
1
credit:30c
1
CLING!−credit:30c
t
Bon_thé
1
credit:10c
1
credit:20c
1
credit:30c
c
Bon_café
2
```

```
credit:20c
r
CLING!
q
```

3 Un automate mystère

[i] L'état final est quand on arrive à écrire le mot BARBARA.

```
82 [~/INF203/TP9]: tail -n 15 Mystere/automate.c
83 void simule_automate(automate * A) {
84   int etat_courant, etat_suivant;
85   int entree = ' ';
86
87   etat_courant = A->etat_initial;
88   while (entree != 'q' && A->etats_finals[etat_courant] != 1) {
89     entree = lire_entree();
90     if(entree!='q'){
91       etat_suivant = A->transitions[etat_courant][entree];
92       printf("%s\n", A->sortie[etat_courant][entree]);
93       etat_courant = etat_suivant;
94     }
95   }
96 }
```

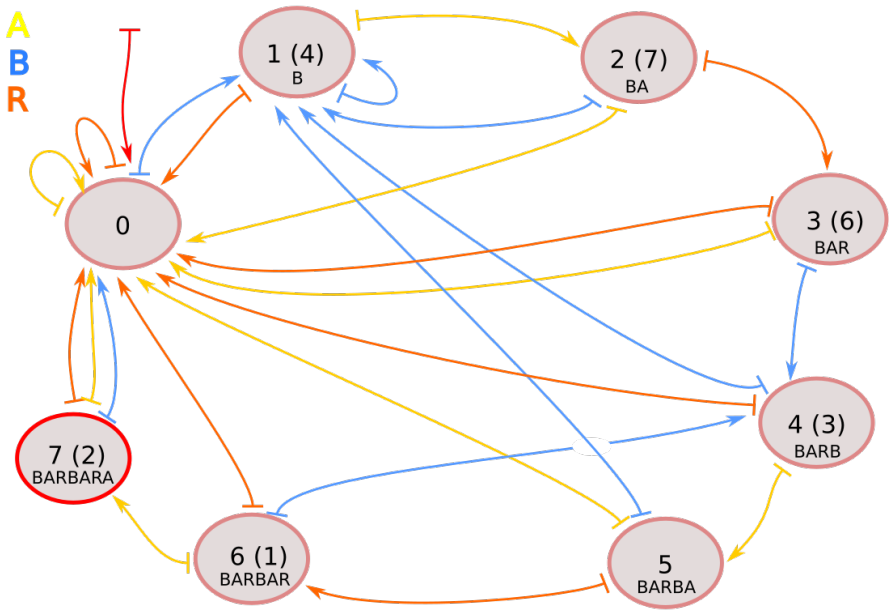

FIGURE 2 – Automate de Barbara. La flèche rouge indique le départ et le rond rouge l'état final. Entre parenthèse figurent les états tels qu'inscrits dans le fichier de départ.

L'automate était initialement dessiné avec les sorties sur BBB, mais la fermeture nous a surpris et nous avons tout perdu. Nous n'avons pas tout recopié pour ne pas surcharger la figure.