République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Ecole Nationale Supérieure d'Informatique

Rapport TP n°2 THL

Construire un automate simple à partir d'un automate généralisé

Réalisé par :

Bouhenni Sarra

Groupe 8, section B

2014 / 2015

Table des matières :

1. Problématique :

Un automate généralisé est défini par le quintriplet A_G <X^*, S, S_0, IF, II>

$II : S \times X^* \rightsquigarrow P(S)$

On trouve dans un automate généralisé 3 types de transitions :

1- Transitions causées par les lettres de X.
2- Transitions causées par les mots de longueur supérieure à 1.
3- Les transitions causées par le mot vide (transitions spontanées).

Donc, comment passer d'un automate généralisé à un automate simple contenant que des transitions causées par les lettres de l'alphabet ?

2. Analyse générale :

Pour passer d'un automate généralisé à un automate simple, on utilise un automate intermédiaire dit : partiellement généralisé.

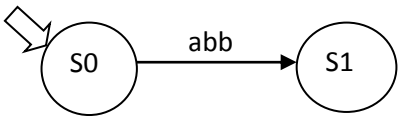
2.1. Passage d'un automate généralisé AG à un automate partiellement généralisé APG.

APG est défini par : $AG < X^*, S, S0, F, II > \rightsquigarrow APG < X \cup \{\varepsilon\}, S, S_0, F, I >$

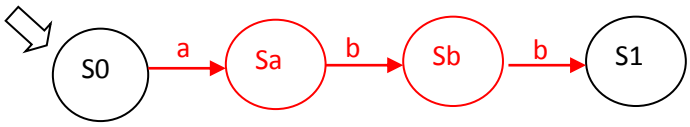
Avec L(APG) = L(AG)

On construit APG à partir de AG en éliminant les transitions causées par des mots de longueur supérieure à 1.

Exemple :



Automate généralisé



Automate partiellement généralisé équivalent

Initialisation de l'automate APG :

$S'_0 = S_0$; S'=S ; X'= X ; F'=F ; II'=ø

- Pour toutes les transitions de II (S_i, w, S_j)
- Si (|w|=0 ou |w|=1) alors : II'=II' ∪ {(Si, w, Sj)}
- Sinon (|w|=n et $w=w_1.w_2 \ldots w_n$)
  - II'=II' ∪ {(Si, w_1, S_{i1}), (S_{i1}, w2, S_{i2}), …, ($S_{i\,(n-1)}$, w_n, Sj)}
  - S'=S' ∪ {S_{i1}, S_{i2}, … $S_{i\,(n-1)}$}

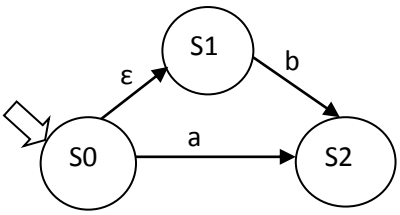
2.2. Passage d'un automate partiellement généralisé APG à un automate simple AS:

AS est défini par : $APG < X \cup \{\varepsilon\}, S, S_0, F, I > \rightsquigarrow AS < X, S, S0, F, II >$

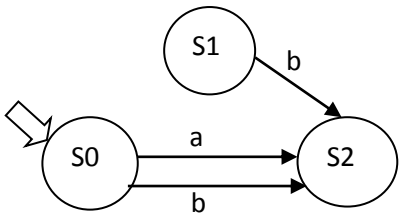
Avec L(AS) = L(APG)

On construit AS à partir de APG en supprimant les transitions spontanées (Causées par le mot vide)

Exemple :



Automate partiellement généralisé



Automate simple équivalent

Initialisation de l'automate AS :

On prend X : l'ensemble de l'alphabet

S'=S ; $S'_0=S_0$; F'=F ; II'= ø

1- On parcourt l'ensemble des transitions de APG (Si, w, Sj)
2- Si c'est une transition spontanée : w= ε alors
   - Si (Sj ϵ F) alors F'=F ∪ Si
   - On supprime la transition spontanée
   - Si (Si # Sj) Tous les successeurs de Sj deviennent des successeurs de Si
3- Sinon (transition non spontanée)
   - II'= II' ∪ {(Si, w, Sj)}
4- Parcourir II', si on trouve une transition spontanée : aller à 1
   Sinon fin.

3. Structures de données utilisées :

Structure = Etat //Cette structure contient le nom de l'état

Nom : chaine de caractères

Structure = instruction // Une structure contenant l'instruction : <Si, wi, Sj>

Si : Etat
Lettre : chaine de caractères
Sj : Etat

Structure = ListEtat // Structure de la liste des états

Etat : Etat
suivant : (pointeur vers) ListEtat

Structure = ListInstruction //Structure de la liste des états

```
Ins :  instruction
Suivant : (pointeur vers) ListInstruction
```

Structure = Automate //Structure de l'automate

```
X : tableau [50] de caractères
S : (pointeur vers) ListEtat
S0 : Etat
F : (pointeur vers) ListEtat
I : (pointeur vers) ListInstruction
```

4. Liste des fonctions et procédures utilisées :

construireAPG (A : Automate) : Automate

Cette procédure retourne l'automate partiellement généralisée équivalent à l'automate généralisé A.

construireAS (A : Automate) : Automate

La procédure construireAS retourne l'automate simple équivalent à l'automate partiellement généralisé A qui est donné à l'entrée.

InsertionEtat (liste : (ptr) listetat, nometat : chaine de caractères) : Listetat (ptr)

La fonction insertion état insère un état donné dans la liste des états dans l'entrée et retourne la tête de la liste mise à jour.

Insertioninst (liste_I : listinstruction, ins : instruction) : Listinstruction (ptr)

Insérer une instruction dans la liste des instructions donnée à l'entrée et retourner la tête de la liste des instructions mise à jour.

Existe (nometat : chaine de caractères, etats : listetats(ptr)) : booléen

Cette fonction vérifie si un état "Sj" existe déjà dans la liste des états "etats"

Existe_Inst(instruction inst, I : listinstruction (ptr)) : booléen

Vérifier si une instruction existe déjà dans la liste des instructions donnée à l'entrée ou non.

existe_alphabet(alphabet : tableau[] de caractères, c : caractères) :booléen

Tester si une lettre 'c' existe déjà dans le tableau de l'alphabet donné à l'entrée.

lire_automate() :

Lecture de l'automate déterministe simple non réduit inséré par l'utilisateur

L'alphabet : X
Les états de l'automate : S
L'état initial de l'automate : S0
Les états finaux de l'automate : F
La liste des instructions de l'automate : I

affiche_automate (A : Automate)

Afficher un automate A, donné à l'entrée .

affiche_listinstruction (p : (ptr) listinstruction)

Affichage de la liste des transitions dans une liste d'instructions donnée.

affiche_listetat (p : (ptr) listetat)

Afficher la liste des états contenus dans une liste donnée à l'entrée de la procédure.

5. Algorithmes utilisés :

5.1. Construire un automate partiellement généralisé à partir d'un automate généralisé :

```
    int i=0,n=0,j=0;
    for(i=0;i<strlen(AG.X);i++) APG.X[i]=AG.X[i];   /* construire X de APG à partir de X de AG*/
    APG.S=AG.S ;      APG.S0=AG.S0;     APG.F=AG.F;
    ListInstruction *liste=NULL, *p=AG.I;
    instruction inst;      ListEtat *si=NULL, *sj=NULL;
    char e[]="S";           char c='a';
while(p!=NULL)    /* parcourir la liste des instructions de AG*/
{
   si=NULL;     sj=NULL;
   if(strlen(p->ins.lettre)>1)
     { /* si le mot est de longueur > 1 */
            si=insertionEtat(si,p->ins.Si.nom);  /* construire un nouveau etat portant le nom de Si */
             n=strlen(p->ins.lettre);              /* n est le nombre de transtions qu'on va ajouter */
             for(i=0;i<n;i++){
                  if  (i<n-1)
                        {       /* si c'est la dernière transition, on prend Sj comme état d'arrivée */
                          sprintf(etatName[j],"%s%c\0",e,c) ;  /* créer le nom du nouvel état */
                          sj=insertionEtat(sj,etatName[j]);
                          j++;  c++;
                         }
                   Else       sj=insertionEtat(sj,p->ins.Sj.nom);
              /* on insere le nouvel état dans la liste des états de APG */
              APG.S=insertionEtat(APG.S,sj->etat.nom);
              /* Créer une nouvelle transition avec un mot de taille=1 */
              inst.lettre[0]=p->ins.lettre[i];
              inst.lettre[1]='\0';
              inst.Si=si->etat;
              inst.Sj=sj->etat;
              liste=insertionInst(liste,inst);  /* insertion de la nouvelle transition
                                                   dans la liste des instructions de APG */
              si=sj;  /* l'état d'arrivée devient l'état de départ de la prochaine instruction à créer */
              sj=NULL;
      }
    }
    Else
       {
              /* le mot est de longueur inférieure ou égale à 1 */
             strcpy(inst.lettre,p->ins.lettre);
             inst.Si=p->ins.Si;
             inst.Sj=p->ins.Sj;
             /* I'=I' union { (si, w, sj)}  */
             liste= insertionInst(liste,inst);
        }
    p=p->suivant;  /* passer à la prochaine instruction */
}
     APG.I=liste;
```

5.2. Construire un automate simple à partir d'un automate partiellement

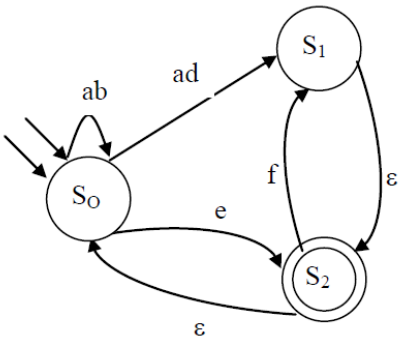
```
ListInstruction *I=NULL, *p=APG.I, *q=APG.I;
ListEtat *F=APG.F;
instruction ins;   int trouv=1,i=0;

while(trouv)
 {  /*Tant qu'il y a des transitions spontanées, on reboucle pour les éliminer*/
    trouv=0;
  while(p!=NULL)
   {  /*parcourir la liste des instructions pour chercher les transitions spontanées*/
      if(p->ins.lettre[0]!='$')
    {
       ins.Si=p->ins.Si;
       strcpy(ins.lettre, p->ins.lettre);
       ins.Sj=p->ins.Sj;
       I=insertionInst(I,ins);
    }
    else{
          trouv=1;
          if ( existe(p->ins.Sj.nom,APG.F))  /*Si sj est un etat final alors si devient final*/
             F=insertionEtat(F,p->ins.Si.nom);
         while(q!=NULL) /*parcourir la liste des instructions*/
          {     if   (strcmp(p->ins.Sj.nom,q->ins.Si.nom)==0)
                /* si Sj' est un succésseur de Sj, Sj' devient successeur de Sj */
                 {
                        ins.Si=p->ins.Si;
                        ins.Sj=q->ins.Sj;
                        strcpy(ins.lettre,q->ins.lettre);
                        I=insertionInst(I,ins);  /* insérer ins dans la nouvelle liste des transitions*/
                 }
                q=q->suivant ;
          }
       }
     p=p->suivant;        q=APG.I;
 }
If (trouv==1)   /* si on a trouvé une transition spontanée , on refait la
                   même chose avec la nouvelle liste d'instructions */
{
   p=I;    I=NULL;    q=p;
}
  }
AS.F=F;  AS.I=I;  AS.S=APG.S;  AS.S0=APG.S0;

/*l'ensemble de l'alphabet de AS = L'ensemble de l'alphabet de APG sans le mot vide*/
if(strlen(APG.X)>0)      for(i=0;i<strlen(APG.X)-1;i++) AS.X[i]=APG.X[i];
```

6. Déroulement d'un exemple d'automate généralisé :

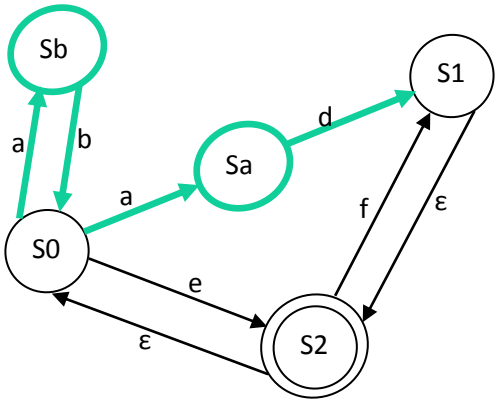
Soit l'automate A<X^*, S, S_0, F, I> avec : X= {a, b, c, d, e, f}



Automate généralisé AG

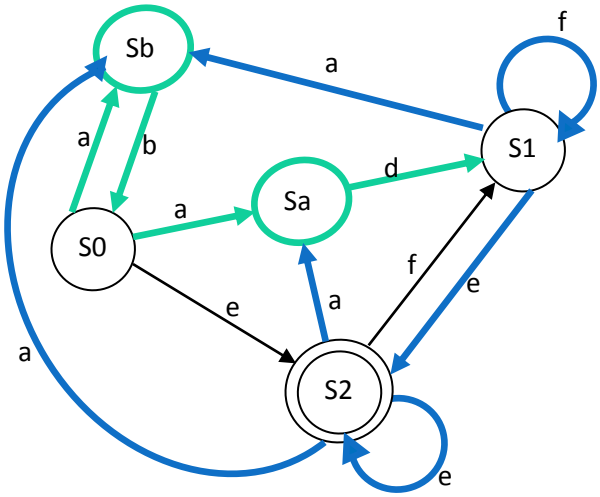
Construction de l'automate partiellement généralisé équivalent à cet automate :

Les états ajoutés sont : Sa, Sb donc S=S ∪ {Sa, Sb}



Automate partiellement généralisé APG

Construction de l'automate simple à partir de l'automate généralisé :

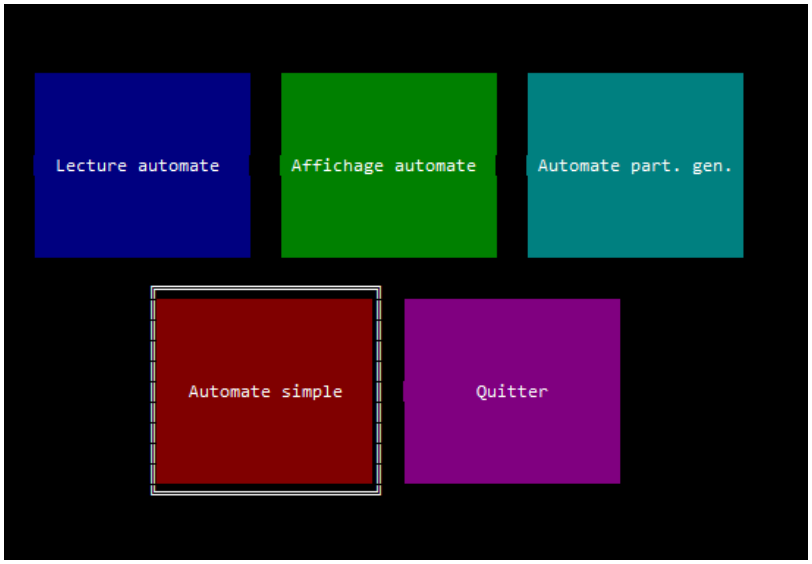


Automate simple AS

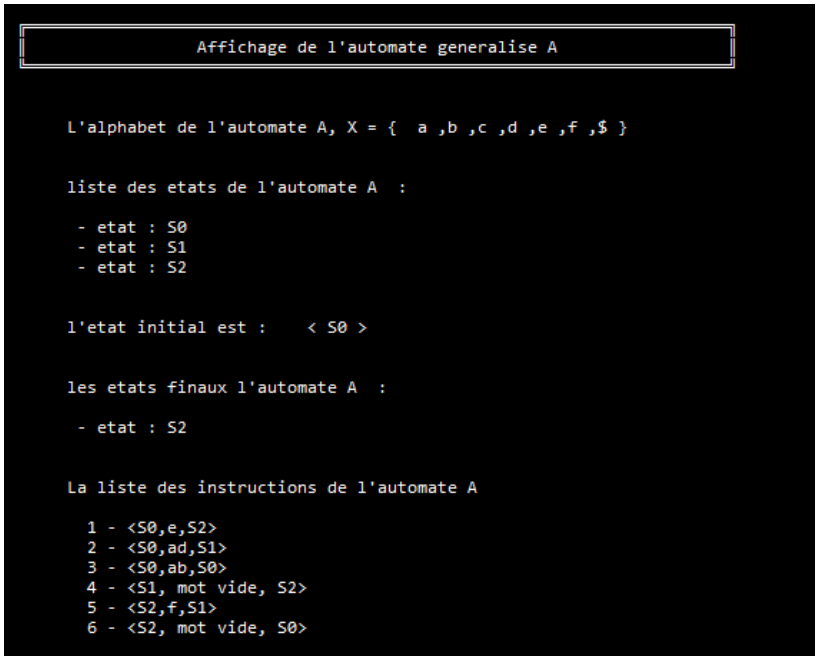
7. Résultats de l'exécution des algorithmes de construction de l'automate simple :

- Lancement du programme :

- Affichage du menu :



- Affichage de l'automate généralisé inséré par l'utilisateur :



- Affichage de l'automate partiellement généralisé équivalent :

```
L'alphabet de l'automate partiellemnt generalise, X = {  a ,b ,c ,d ,e ,f ,mot vide }

liste des etats de l'automate partiellement generalise  :

 - etat : S0
 - etat : S1
 - etat : S2
 - etat : Sa
 - etat : Sb

l'etat initial est :    < S0 >

les etats finaux l'automate partiellement generalise  :

 - etat : S2

La liste des instructions de l'automate partiellemnt generalise

  1 - <S2, mot vide, S0>
  2 - <S2,f,S1>
  3 - <S1, mot vide, S2>
  4 - <Sb,b,S0>
  5 - <S0,a,Sb>
  6 - <Sa,d,S1>
  7 - <S0,a,Sa>
  8 - <S0,e,S2>
```

- Affichage de l'automate simple équivalent :

```
L'alphabet de l'automate simple A, X = {  a ,b ,c ,d ,e ,f }

liste des etats de l'automate simple A  :

 - etat : S0
 - etat : S1
 - etat : S2
 - etat : Sa
 - etat : Sb

l'etat initial est :    < S0 >

les etats finaux l'automate simple A  :

 - etat : S2
 - etat : S1

La liste des instructions de l'automate simple A

  1 - <S0,e,S2>
  2 - <S0,a,Sa>
  3 - <Sa,d,S1>
  4 - <S0,a,Sb>
  5 - <Sb,b,S0>
  6 - <S1,f,S1>
  7 - <S1,a,Sb>
  8 - <S1,a,Sa>
  9 - <S1,e,S2>
  10 - <S2,f,S1>
  11 - <S2,e,S2>
  12 - <S2,a,Sa>
  13 - <S2,a,Sb>
```