Programme de passage d'un automate généralisé à un automate simple

TP MODULE THP

TP réalisées par :

BELBAKI Yasmine

LAZAZNA Khalida

-ESI- Ecole Nationale Supérieure d'Informatique

2015/2016

Sommaire

I. Quelques définitions :

1. Automate généralisé :

Un automate généralisé AG < X*, S, Si, F, II> est caractérise par 5 paramètres :

- **X :** l'alphabet de l'automate ;
- **S :** l'ensemble fini d'états de l'automate ;
- **Si :** l'ensemble des états initiaux de l'automate ;
- **F :** l'ensemble des états finaux de l'automate ;
- **II :** l'ensemble des instructions de l'automate, avec : II : S x X* ⍰⍰P(S)

Il peut contenir 3 types de transitions :

- Les transitions causées par des lettres de X.
- Les transitions causées par des mots de X* telle que |w| >1
- Les transitions spontanées causées par le mot vide (ε)

2. Automate partiellement généralisé :

Un automate partiellement généralisé est également caractérisé par 5 paramètre : APG< X ⋃ {ε}, S', S0', F', II'>

Il peut contenir 2 types de transitions :

- Les transitions causées par des lettres de X (|w| =1).
- Les transitions spontanées causées par le mot vide (ε)

3. Automate simple :

Il ne peut contenir qu'un seul type de transitions :

- Les transitions causées par des lettres de X (mot de longueur 1, |W|=1).

II. Proposition :

A tout automate généralisé AG< X*, S, Si, F, II >, il existe un automate simple AS < X, S', S0', F', II'> équivalent telle que : L(AG)=L(AS)

III. Analyse :

Etape 1 : Passage d'un automate généralisé à un automate partiellement généralisé :

APG est défini par : AG <X∗,S,S0,F,II> ⇝ APG< X ∪{ε},S',S'0,F',I'>
Avec L(APG) = L(AG)
On construit partiellement généralisé à partir de l'automate généralisé en éliminant les transitions causées par des mots de longueur supérieure à 1.

Initialisation :
S'0= S0 ; S'=S ; X'= X ; F'=F ; II'=ø

Pour toutes les transitions de II (Si, w, Sj)

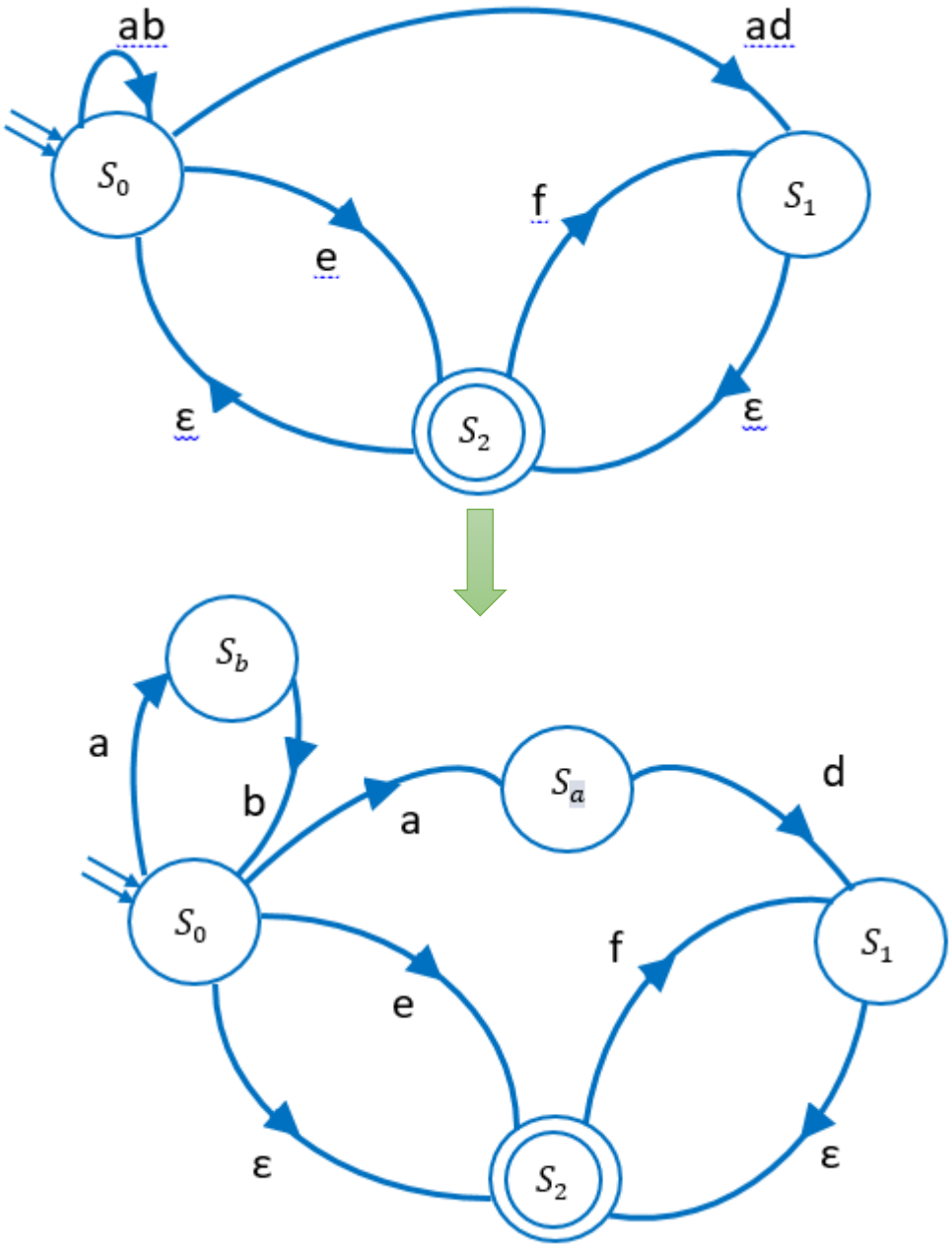
Si (|w|=0 ou |w|=1) alors : II'=II' ∪{(Si,w,Sj)}

Sinon (|w|=n et w=w1.w2… wn)

II'=II' ∪ {(Si, w1, Si1), (Si1, w2, Si2), …, (Si (n-1), wn, Sj)}

S'=S' ∪ {Si1, Si2, … Si (n-1)}

Exemple :



Etape 2 : Passage d'un automate partiellement généralisé à un automate simple :

L'automate simple AS est défini par : APG< X ∪{ε},S',$S'0$,F',I'> ⇝AS <X,S'',S''0,F',II''>
Avec L(AS) = L(APG)

On construit l'automate simple à partir de l'automate partiellement généralisé en supprimant les transitions spontanées (Causées par le mot vide).

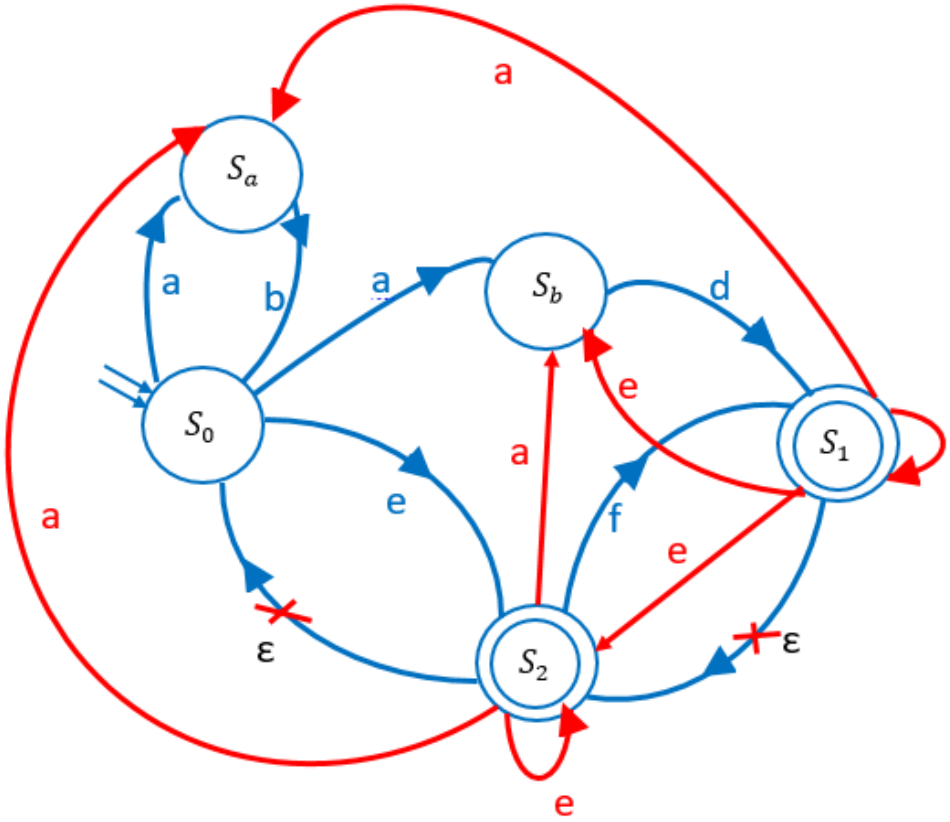
Initialisation :

X : l'ensemble de l'alphabet
S''=S' ; S''0=S'0 ; F''=F' ; II''= ø

- On parcourt l'ensemble des transitions de APG (Si, w, Sj)
- 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
- Sinon (transition non spontanée)
  - II''= II'' ∪{(Si,w,Sj)}
- Parcourir II', si on trouve une transition spontanée : aller à 1

  Fin

Suite de l'exemple précédent :



IV. Déclaration des structures utilisées :

```
Type Etat : structure
  |       nom : chaine de caractères
Fin
```

```
Type Listes_des_Etats : structure
    |       etat : Etat
    |       *suivant : Listes_des_Etats
Fin
```

```
Type instruction : structure
    |       Si : Etat
    |       lettre : Tableau[1..10] de caractèress
    |       Sj : Etat
Fin
```

```
Type Liste_des_Instructions  : structure
    |       instruct : Instruction
    |       *suivant : Liste_des_Instructions
Fin
```

A<X,S,S0,F,I>

```
Type Automate  : structure
    |       X : Tableau[1..50] de caractères
    |       *S : Liste_des_Etats
    |       S0 : Etat
    |       *F : Liste_des_Etats
    |       *I : Liste_des_Instrucutions
Fin
```

V. Algorithmes :

Algorithme de passage d'un automate généralisé à un automate partiellement généralisé :

```
construireA_partiellement_generalise()
entier i=0,n=0,j=0;
Début
    Pour i allant de 0 à longueur (A_g.X) faire
    A_p_g.X[i]=A_g.X[i];      /* L'automate partiellement généralisé et généralisé ont
                                 le même alphabet*/
    A_p_g.S=A_g.S;   /* Initialiser l'ensemble des états de l'automate part_généralsé avec
                        les  états de l'automate généralisé*/
    A_p_g.S0=A_g.S0; /* Initialisation de l'états initial avec l'état initiale de l'automate
                        généralisé*/
    A_p_g.F=A_g.F;   /* Initialiser l'ensemble des états finaux de l'automate part_généralisé
          avec l'ensemble des états finaux de l'automate généralisé*/
    Liste_des_Instructions *liste=NULL; /* Initialiser l'ensemble des instruction à NULL */
```

```
Liste_des_Instructions *p=A_g.I;
Instruction inst;
Liste_des_Etats *si=NULL, *sj=NULL;
char e[]="S";
char c='a';
Tant que (p!=NULL)
Début TQ
    si=NULL; sj=NULL;
    Si (longueur(lettre de l'instruction>1)   /* Ce n'est pas une lettre, c'est un mot de
                                                 longueur > 1*/
     Début
        si=InsererEtat(Liste des états , Si);
        n=longueur(lettre de l'instruction); /* (n-1) représente le nombre d'états
                                                intermédiaires à rajouter*/
        Pour ( i allant de 0 à n) faire

        Début
                Si (i<n-1) /* Quand on arrive à la dernière transition, l'état final
                              correspond à  l'état sj de l'instruction*/
                Début
                        etatName[j]=e+c;
                        sj=InsererEtat(sj,etatName[j]);
                         j++;
                         c++; /*  Les états créés sont : Sa, Sb, Sc, ...etc. */
                Fin
                Sinon  sj=InsererEtat(liste des états, Sj);

                A_p_g.S=InsererEtat(liste des états de l'automate part_gen,Sj);
                inst.lettre[0]=lettre de l'instruction;
                inst.Si=si->etat;
                inst.Sj=sj->etat;
                liste=InsererInstruction(liste,inst);
                si=sj;
                sj=NULL;
        Fin
    Fin Si

    Sinon /* Il s'agit d'une lettre ou du mot vide (longueur inférieure ou égale à 1) */
            inst.lettre =p->instruct.lettre);
            inst.Si=p->instruct.Si;
            inst.Sj=p->instruct.Sj;
            liste=InsererInstruction(liste,inst);
    Fin Si
```

```
    p=p->suivant;
  Fin TQ
  A_p_g.I=liste;
Fin
```

Algorithme de passage d'un automate partiellement généralisé à un automate simple :

```
construireA_simple()

Début
    Liste_des_Instructions *I=NULL, *p=A_p_g.I, *q=A_p_g.I;
    Liste_des_Etats *F=A_p_g.F;
    Instruction ins;
    int trouv=1,i=0;
    Tant que (trouv) /* Tant qu'on trouve des transitions spontanées on fait ce qui
                        suit*/
        Début
            trouv=0;
            Tant que(p!=NULL)
             Début TQ
                SI(lettre de l'instruction !='ε')
                Début
                    ins.Si=p->instruct.Si;
                    ins.lettre= p->instruct.lettre;
                    ins.Sj=p->instruct.Sj;
                    I=insererInstruction(I,ins);
                 Fin
                Sinon
                 Si (nom instruction Si == nom instruction Sj)
                 Début
                    trouv=1;
                   Si (existe(p->instruct.Sj.nom,A_p_g.F)) /* 1ère règle
                        : Si sj est un état final alors si devient final */
                    F=insererEtat(F,p->instruct.Si.nom) ;
                   Tant que(q!=NULL) /* recherche des successeurs de
                                        Sj*/
                     Début
                        Si (strcmp(p->instruct.Sj.nom,q
                                                >instruct.Si.nom)==0)
                /* 2ème règle : si Sk est un successeur de Sj, // Sk devient successeur de Si*/
```

```
                                        Début
                                            ins.Si=p->instruct.Si;
                                            ins.Sj=q->instruct.Sj;
                                            ins.lettre=q->instruct.lettre;
                                            I=insererInstruction(I,ins);
                                        Fin
                                    q=q->suivant;
                                Fin TQ
                            Fin Si
                        Fin Sinon
                    Fin TQ
                    p=p->suivant;
                    q=A_p_g.I;
            Fin TQ

            if(trouv==1)
             Début
                p=I;
                I=NULL;
                q=p;
             Fin
        Fin TQ
        A_simple.F=F;
        A_simple.I=I;
        A_simple.S=A_p_g.S;
        A_simple.S0=A_p_g.S0;
        Si(nombre de lettre de l'alphabet X de A_p_g >0)
            Pour (i allant de  à nombre de lettres de l'alphabet X)  faire
                A_simple.X[i]=A_p_g.X[i];
Fin
```

VI. Jeu d'essai :

Ci-dessous le résultat de l'exécution des algorithmes de construction d'un automate simple à partir d'un automate généralisé. (Application sur l'exemple précédent)

Affichage de l'automate généralisé introduit :

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


       Ensemble des états de l'automate A  :

        - etat : S0
        - etat : S1
        - etat : S2


       L'état initial est :    < S0 >


       Les états finaux l'automate A  :

        - etat : S2


       L'ensemble des instructions de l'automate A

         1 - <S0,e,S2>
         2 - <S0,ad,S1>
         3 - <S0,ab,S0>
         4 - <S1, mot vide, S2>
         5 - <S2,f,S1>
         6 - <S2, mot vide, S0>

Press any key to continue . . .
```

Affichage de l'automate partiellement généralisé :

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


      liste des états de l'automate partiellement généralisé  :

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


      l'etat initial est :    < S0 >


      les etats finaux l'automate partiellement généralisé :

       - etat : S2


      La liste des instructions de l'automate partiellemnt généralisé

        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>

Press any key to continue . . .
```

Affichage de l'automate simple :

```
Affichage de l'automate simple équivalent a l'automate généralisé A
        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>

Press any key to continue . . .
```