

TP2

# *Transformation d'un automate généralisé en un automate simple*

**Présenté par :**

- CHEROUANA Wissem

**Groupe :** 06

**Encadré par :** Mme BENATCHBA

# Sommaire :

Sommaire :	1
Notions de cours :	2
Définitions :	2
▪ L'automate généralisé :	2
▪ L'automate partiellement généralisé :	2
▪ L'automate simple :	2
Proposition :	2
Algorithmes :	3
▪ Classes :	3
▪ Algorithmes :	4
1. Algorithme de transformation d'un automate $A_G$ à $A_{PG}$ :	4
2. Algorithme de vérification si un automate est partiellement généralisé :	5
3. Algorithme de transformation d'un automate $A_{PG}$ à $A_S$ :	5
Jeu d'essai :	8

# Notions de cours :

## Définitions :

### ▪ L'automate généralisé :

Un automate généralisé  $A_G < X^*, S, Si, F, II >$  est caractérisé 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 :  $II : S \times X^* \rightarrow P(S)$

Cet ensemble 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 ( $\epsilon$ )

### ▪ L'automate partiellement généralisé :

Un automate partiellement généralisé  $A_{PG} < X \cup \{\epsilon\}, S', S0', F', II' >$  est caractérisé également 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 ;

Cet ensemble peut contenir 2 types de transitions :

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

### ▪ L'automate simple :

Un automate est dit simple si le passage d'un état à un autre est fait par la lecture d'une lettre (mot de longueur 1,  $|W|=1$ )

## Proposition :

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

# Algorithmes :

## ▪ Classes :

### **Etat : Classe**

- nbEtat : entier
- fin : booléen

### **Transition : Classe**

- Si : Etat
- Xi : chaîne de caractères
- Sj : Etat

### **Automate : Classe**

- ensAlphabet : Liste de caractères
- ensEtats : Liste d'Etat
- ensEtatsInitiaux : Liste d'Etat
- ensEtatsFinaux : Liste d'Etat
- ensTran : Liste de Transition
- nbAlphabet : entier
- nbEtats : entier
- nbEtatsInitiaux : entier
- nbEtatsFinaux : entier
- nbTransitions : entier

## ■ Algorithmes :

Pour passer d'un automate généralisé à un automate simple, il faudra d'abord passer par un automate partiellement généralisé :  $A_G \rightarrow A_{PG} \rightarrow A_S$

### 1. Algorithme de transformation d'un automate $A_G$ à $A_{PG}$ :

Soit  $A_G$  l'automate généralisé établi à l'aide de la méthode CréerAutomateGénéralisé(), et soit  $A_{PG}$  l'automate partiellement généralisé obtenu à partir de  $A_G$  en appliquant l'algorithme qui suit :

```
APG.nbAlphabet= AG.nbAlphabet;  
APG.ensAlphabet= AG.ensAlphabet;  
APG.nbEtats= AG.nbEtats;  
APG.ensEtats=AG.ensEtats;  
APG.ensEtatsInitiaux= AG.ensEtatsInitiaux;  
APG.ensEtatsFinaux= AG.ensEtatsFinaux;  
APG.nbEtatsFinaux= AG.nbEtatsFinaux;  
APG.nbEtatsInitiaux= AG.nbEtatsInitiaux;
```

Pour toute transition ( $S_i, w, S_j$ ) de  $A_G$  faire

DPOUR

Si ( $|w| = 1$  ou  $w = \epsilon$ ) alors

| On rajouter ( $S_i, w, S_j$ ) à l'ensemble d'instructions  $A_{PG}.ensTran$

Sinon // mot de la transition non vide ou de longueur supérieure à 1

| On stocke la transition dans une transition temporaire trTmp ;

| On récupère la longueur du mot  $w$  dans « tailleAlph »

| Soit  $j=0$  ;

| TQ ( $j < \text{tailleAlph}$ ) //Pour parcourir tout le mot

- On ajoute un nouvel état à l'ensemble des états  $A_{PG}.ensEtats$
- La transition résultante est celle ayant comme état initial :  $S_i$ , comme alphabet : la lettre pointée par l'indice  $j$  dans le mot  $w$ , et comme état final : le nouvel état initial
- On rajoute cette transition à l'ensemble des transitions
- $j++$  ;

| FTQ /\* On réitère cette action jusqu'à ce qu'on arrive à former une transition ayant un  $w$  tq  $|w| = 1$  et son état d'arrivée est celui de la transition temporaire enregistrée au début dans trTmp → ajouter (tailleAlph - 1) états intermédiaires à  $A_{PG}.ensTrans$ ; \*/

| FSI

FPOUR

## 2. Algorithme de vérification si un automate est partiellement généralisé :

La méthode qui suit retourne un booléen permettant de savoir si un automate est encore partiellement généralisé ou pas, autrement dit vérifie l'existence de transitions spontanées ou non.

### **AutomatePartilGeneralise (Automate A): booléen**

Début

```
    parGen: booléen  
    parGen : faux;  
    Pour i= 1 à A.nbTransitions  
        Si A.ensTran.(i).Xi == "&" alors  
            parGen=true;  
        FSI  
    FPOUR  
    retourner parGen;
```

Fin

## 3. Algorithme de transformation d'un automate $A_{PG}$ à $A_S$ :

La méthode qui suit permet d'éliminer les transitions spontanées existantes dans l'ensemble des transitions de l'automate partiellement généralisé :

Soit  $A_S$  : Automate, l'automate simple obtenu après application de cet algorithme :

### ■ Initialisation de l'automate $A_S$ :

```
 $A_S$ .nbEtats=  $A_{PG}$ .nbEtats;  
 $A_S$ .ensEtats= $A_{PG}$ .ensEtats;  
 $A_S$ .nbEtatsInitiaux=  $A_{PG}$ .nbEtatsInitiaux;  
 $A_S$ .ensEtatsInitiaux=  $A_{PG}$ .ensEtatsInitiaux;  
 $A_S$ .nbEtatsFinaux=  $A_{PG}$ .nbEtatsFinaux;  
 $A_S$ .nbAlphabet=  $A_{PG}$ .nbAlphabet -1 ; /* exclure le mote vide */  
 $A_S$ .ensAlphabet=  $A_{PG}$ .ensAlphebet; /* à l'aide d'une boucle permettant  
d'ajouter tous le alphabets sauf le mot vide */
```

- Mise à jour de l'ensemble d'états finaux de l'automate simple  $A_S$  :

Pour  $i$  allant de 1 à  $A_{PG}.nbEtatsFinaux$  faire :

DPOUR

- On rajoute à l'ensemble d'états finaux de  $A_S$  l'état final  $A_{PG}.ensEtatsFinaux.(i)$  de l'automate partiellement généralisé
- Pour  $j$  allant de 1 à  $A_{PG}.nbTransitions$  faire :
  - DPOUR
    - Si l'état d'arrivée de la transition est égal à l'état final en question /\*  $A_{PG}.ensTrans.(j).Sj == A_{PG}.ensEtatsFinaux.(i)$  \*/
      - Si le mot de la transition est «  $\epsilon$  » alors
        - On rajoute l'état initial de la transition à l'ensemble d'états finaux de  $A_S$
        - On incrémente le nombre d'états finaux de  $A_S$

FSI

FSI

FPOUR

FPOUR

Après avoir mis à jour l'ensemble des états finaux de l'automate  $A_S$ , vient finalement la dernière étape qui consiste à éliminer toutes les transitions spontanées de l'automate partiellement généralisé  $A_{PG}$ ,

Dans ce qui suit l'algorithme permettant d'effectuer cette opération, qui se base sur l'identification tout d'abord des transitions spontanées existantes, et ensuite sur la création de nouvelles transitions en se référant aux successeurs de l'état d'arrivée des transitions spontanées.

■ Mise à jour de l'ensemble de transitions de l'automate simple  $A_S$  :

```

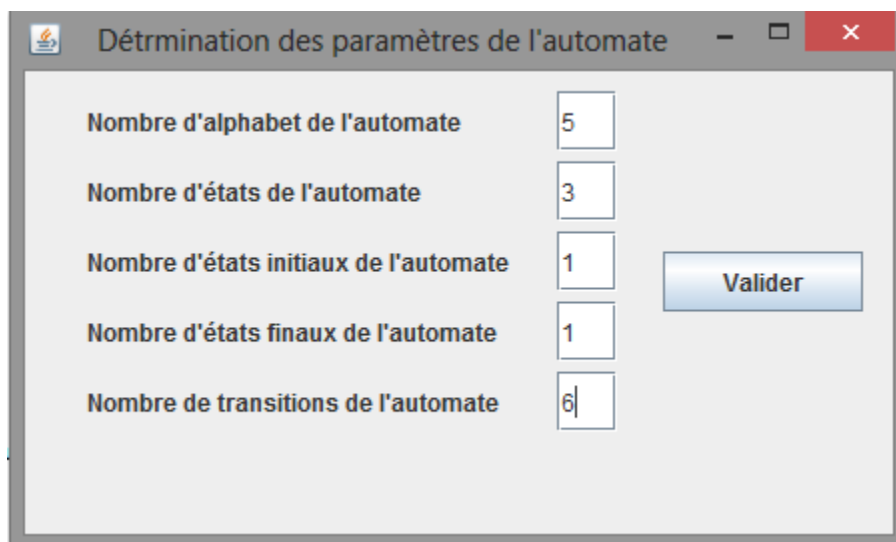
TANTQUE (AutomatePartilGeneralise( $A_{PG}$ )) /* Tant que l'automate
généralisé contient des transitions spontanées */
Pour i allant de 1 à  $A_{PG}.nbTransitions$  faire
DPOUR
    Si ( $A_{PG}.ensTran.(i).Xi \neq " \epsilon "$  /* La transition n'est pas spontanée */
        ○ On rajoute cette transition à l'ensemble de transition de
          l'automate  $A_S$ 
        ○  $A_S.nbTransitions ++$  ;
    Sinon /* La transition est spontanée */
        Pour k allant de 1 à  $A_{PG}.nbTransitions$  faire /*boucle permettant
de déterminer les successeurs de l'état  $S_j$  de la transition */
            Si ( $A_{PG}.ensTran(i).Sj.nbEtat == A_{PG}.ensTran.(k).Si.nbEtat$ )
                ○ On crée une nouvelle transition dont :
                    ■ L'état de départ est  $A_{PG}.ensTran(i).Si$ 
                    ■ L'alphabet est :  $A_{PG}.ensTran.(k).Xi$ 
                    ■ L'état d'arrivée est :  $A_{PG}.ensTran.(k).Sj$ 
                ○ On ajoute cette transition à l'ensemble de
                  transitions de  $A_S$ 
                ○  $A_S.nbTransitions ++$  ;
            FSI
        FPOUR
    FSI
FPOUR
 $A_{PG}.ensTran = A_S.ensTran$  ;
 $A_{PG}.nbTransitions = A_S.nbTransitions$  ;
FTQ

```



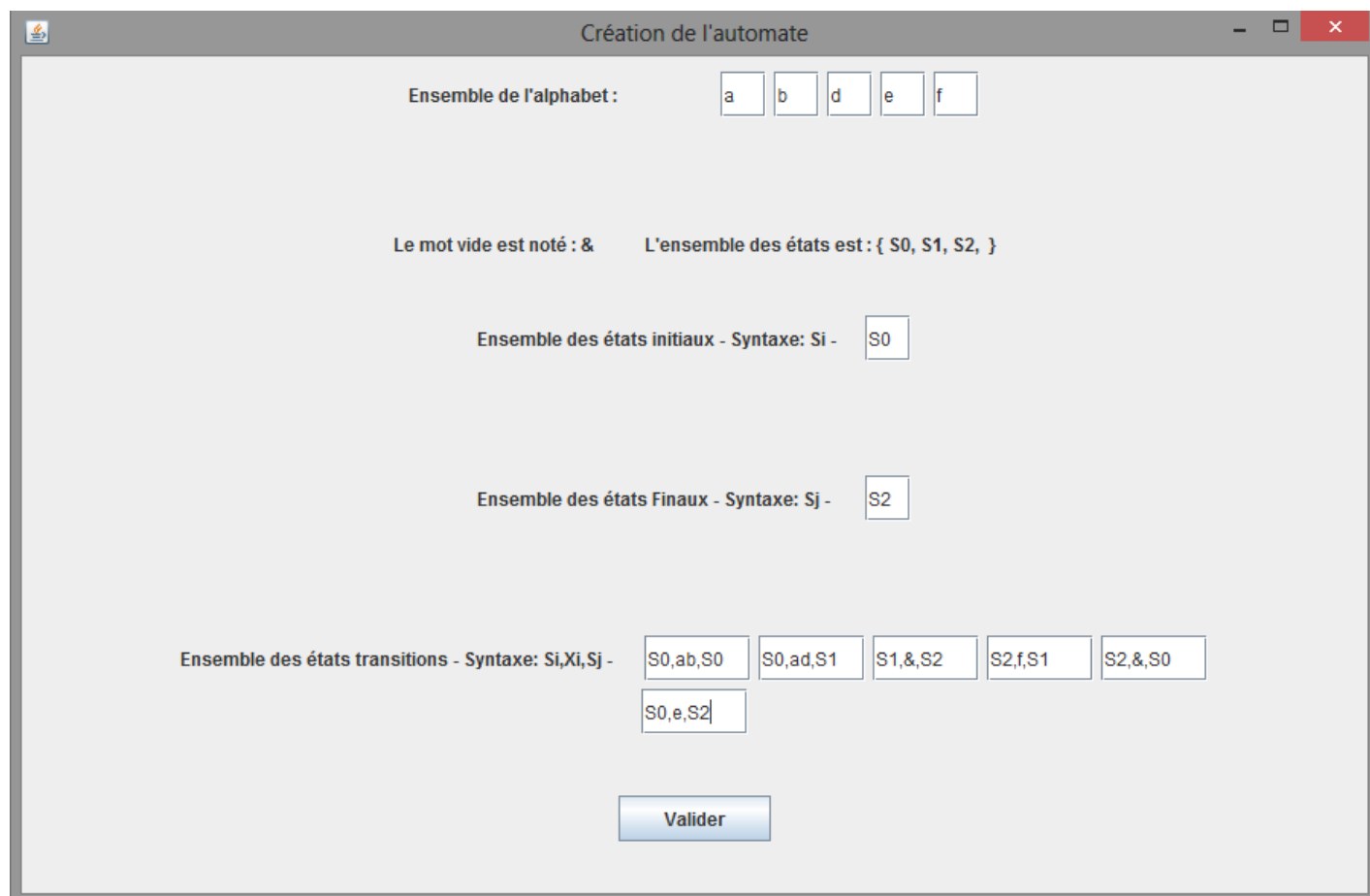
# Jeu d'essai :

## Etape 1 : Détermination des paramètres de l'automate généralisé



A screenshot of a software window titled "Détermination des paramètres de l'automate". It contains five input fields for defining an automaton's parameters: "Nombre d'alphabet de l'automate" (5), "Nombre d'états de l'automate" (3), "Nombre d'états initiaux de l'automate" (1), "Nombre d'états finaux de l'automate" (1), and "Nombre de transitions de l'automate" (6). A "Valider" button is located to the right of the input fields.

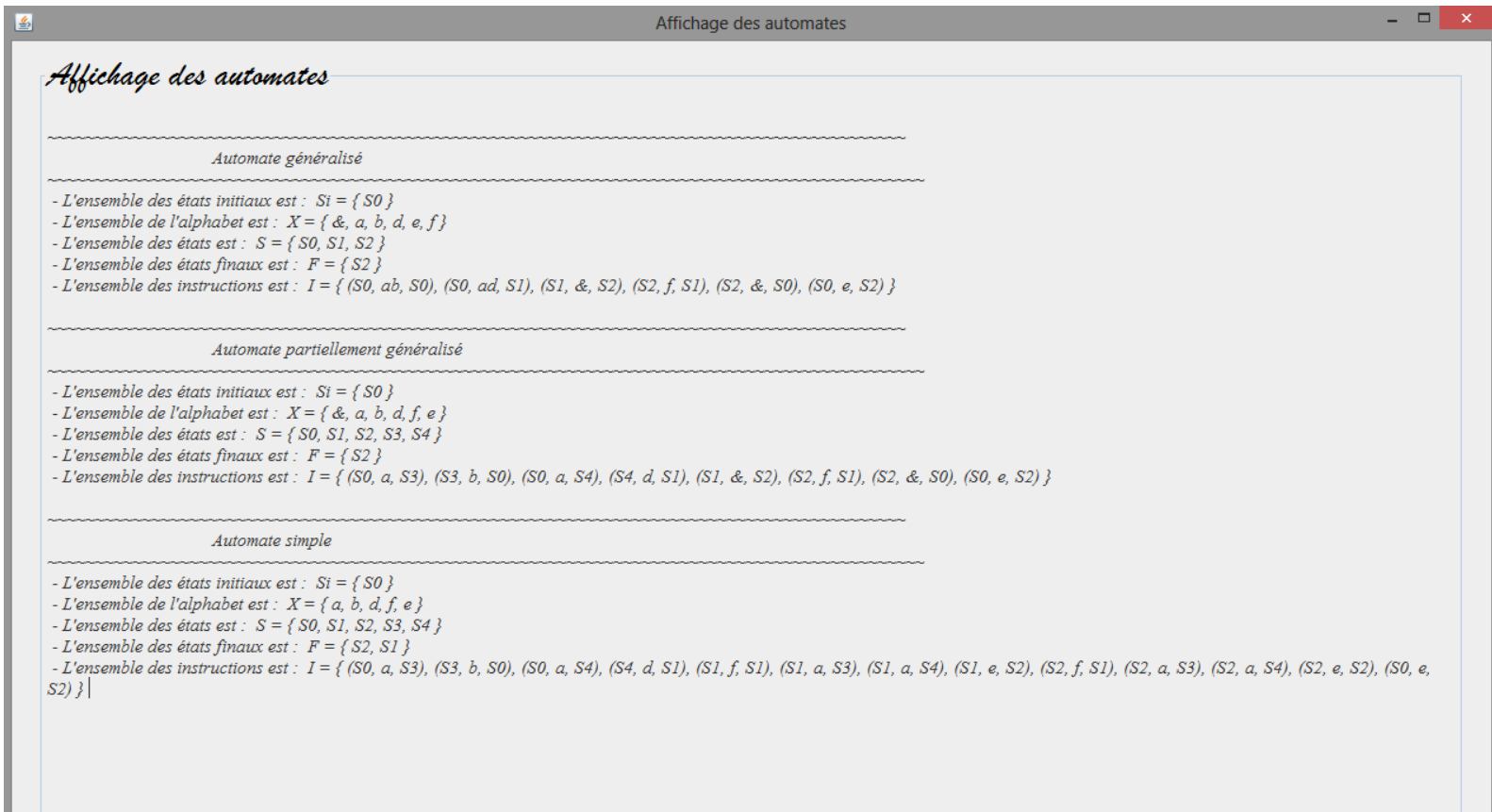
## Etape 2 : Remplissage des champs (Création de l'automate généralisé)



A screenshot of a software window titled "Création de l'automate". It contains several input fields for creating a generalized automaton: "Ensemble de l'alphabet" (a, b, d, e, f), "Le mot vide est noté : &" (empty), "L'ensemble des états est : { S0, S1, S2, }" (empty), "Ensemble des états initiaux - Syntaxe: Si -" (S0), "Ensemble des états Finaux - Syntaxe: Sj -" (S2), and "Ensemble des états transitions - Syntaxe: Si,Xi,Sj -" (S0,ab,S0; S0,ad,S1; S1,&,S2; S2,f,S1; S2,&,S0; S0,e,S2). A "Valider" button is located at the bottom.

**Nota :** Il faudra respecter les syntaxes de saisie pour pouvoir accéder à la fenêtre ci-dessous, si ce n'est pas le cas des messages d'erreurs seront affichés vous indiquant où réside le problème.

### Étape 3 : Affichage des automates



*Affichage des automates*

---

*Automate généralisé*

---

- L'ensemble des états initiaux est :  $S_i = \{ S_0 \}$
- L'ensemble de l'alphabet est :  $X = \{ \&, a, b, d, e, f \}$
- L'ensemble des états est :  $S = \{ S_0, S_1, S_2 \}$
- L'ensemble des états finaux est :  $F = \{ S_2 \}$
- L'ensemble des instructions est :  $I = \{ (S_0, ab, S_0), (S_0, ad, S_1), (S_1, \&, S_2), (S_2, f, S_1), (S_2, \&, S_0), (S_0, e, S_2) \}$

---

*Automate partiellement généralisé*

---

- L'ensemble des états initiaux est :  $S_i = \{ S_0 \}$
- L'ensemble de l'alphabet est :  $X = \{ \&, a, b, d, f, e \}$
- L'ensemble des états est :  $S = \{ S_0, S_1, S_2, S_3, S_4 \}$
- L'ensemble des états finaux est :  $F = \{ S_2 \}$
- L'ensemble des instructions est :  $I = \{ (S_0, a, S_3), (S_3, b, S_0), (S_0, a, S_4), (S_4, d, S_1), (S_1, \&, S_2), (S_2, f, S_1), (S_2, \&, S_0), (S_0, e, S_2) \}$

---

*Automate simple*

---

- L'ensemble des états initiaux est :  $S_i = \{ S_0 \}$
- L'ensemble de l'alphabet est :  $X = \{ a, b, d, f, e \}$
- L'ensemble des états est :  $S = \{ S_0, S_1, S_2, S_3, S_4 \}$
- L'ensemble des états finaux est :  $F = \{ S_2, S_1 \}$
- L'ensemble des instructions est :  $I = \{ (S_0, a, S_3), (S_3, b, S_0), (S_0, a, S_4), (S_4, d, S_1), (S_1, f, S_1), (S_1, a, S_3), (S_1, a, S_4), (S_1, e, S_2), (S_2, f, S_1), (S_2, a, S_3), (S_2, a, S_4), (S_2, e, S_2), (S_0, e, S_2) \}$

Vous pouvez également faire un essai en mode console.