TEA – Compliation Automates

Sommaire :

Johan GUERRERO & Marie DIEZ & Loïc DRIEU LA ROCHELLE

1 – Fonctionnalités développées

Le but de ce TEA est de mettre en place un moteur d'automate déterministe permattant d'analyser des mots, afin de dire si ils appartiennent au langage décrit par l'automate. Et ce dans l'idée de mettre en place une partie d'un compilateur : l'analyseur lexicale.

Moteur

Nous avons commencé par mettre en place le moteur d'automate, en ne testant que sur des automates déterministes dans un premier temps.

Pour mettre en place ce moteur, nous avons commencé par implémenter la **lecture des fichiers *.descr***, cette étape se réalise à la création de l'automate. Pour cela nous avons travaillé avec des expressions regulières. Si le fichier de description ne corresponds pas à la syntaxe proposé pour ce TEA, la création de l'automate retournera une erreur dans le fichier *automate.log* à la racine et se terminera sans créer d'automate.

```
auto = Automate(filepath_automate)
```

Figure 1: Création de l'automate, parametre : le chemin du fichier .descr

Une fois l'automate créé, nous pouvons lancer l'analyse des mots, afin de déterminer si les mots appartiennent aux langage décris par l'automate. Pour cela nous avons implémenté une fonction **analyse** qui pour chaque mot d'un fichier *input* fournis analyse le mot avec l'automate pour déterminer si il appartient au langage décris. Chaque automate du jeu de test possède son fichier d'entrée (mots à tester) dans le dossier *entrees*. L'analyse nous dira si oui ou non le mot appretient au langage et si oui la sortie associé. Pour ce faire l'analyse comporte 5 étapes :

- On pars du premier état initial
- Est-ce que les mots correspondent au vocabulaire de l'automate ?
- Est-ce que il existe toutes les transitions nécessaires pour que le mot soit accepté ?
- Est-ce que l'état d'arrivé est un état final ?
- Si l'état d'arrivé n'est pas le bon ou si une transition n'existe pas, on va voir l'état initial suivant si il y a en a plusieurs, sinon le résultat est faux.

Il peut alors y avoir plusieurs cas de figure dans lesquels les mots peuvent ne pas être acceptés, le vocabulaire n'est pas en adéquation avec celui de l'automate, la transition n'existe pas ou bien l'état d'arrivé n'est pas un état final. Pour chacun de ses cas, une erreur adéquate va être écrite dans un fichier de log portant le nom de l'automate dans le dossier *logs*. (*Error : 3 not in final states : F -> [4]*).

Que le mot soit accepté ou non nous affichons les transitions réalisé par l'automate, si le mot est accepté alors les transitions sont juste et la sortie aussi. Dans la cas contraire nous pouvons voir jusqu'ou le moteur à pu lire les transistions et voir les détails de l'erreur.

```
1 → a/# → 2
2 → b/# → 2
2 → b/# → 2
2 → c/# → 3

Error : 3 not in final states : F → [4]
"abbc"   Is not a word from the language of the automaton
Sortie : [] ;
```

Figure 2: Mot non accepté à cause de l'état d'arrivé

```
1 → a/# → 2
2 → b/# → 2
2 → c/# → 3
3 → b/# → 4

"abcb"   Is a word from the language of the automaton
Sortie : [] ;
```

Figure 3: Mot accepté : pas de sortie

```
 1 → m/# → 3
 3 → a/# → 5
 5 → m/# → 7
 7 → a/# → 9
 9 → n/F → 8

 "maman"  Is a word from the language of the automaton
 Sortie : ['F'] ; F
```

Figure 4: Mot accepté : Sortie = F

Déterminisation

Nous avons par la suite mis en place la déterminisation de l'automate. Pour cela une fois l'automate créé nous executons la fonction ***determinise*** sur l'automate, qui retourne un automate deterministe, un fichier *generatedAutomate.descr* est créé à la racine par cette fonction. C'est ce fichier qui sera appelé pour la création de l'automate déterministe qui sera retourné par la fonction. Puis nous continons le traitement d'analyse comme expliqué plus haut sur l'automate deterministe.

Pour déterminiser l'automate nous avons suivie l'algorithme présenté dans le cours de M.Viaud.
Nous avons donc réalisé la fonction ***getLambdaClosure,*** de façon à prendre en compte les lambda-transitions.

Nous avons de plus mis en place une fonction ***toDot*** permettant d'écrire dans le langage de Dot l'automate deterministe créé mais aussi l'automate initial. Le fichier dot réalisé se trouve dans le dossier *dotImage* il suffira par la suite de lancer la commande *make graph* pour générer l'image de l'automate initial et de l'automate déterministe.

2 - Utilisation Moteur :

Les résultats de reconnaissance des mots sont dans le dossier *logs*.
Les graphes réalisés avec Dot sont dans le dossier *dotImage*.
Les mots d'entrées sont fournis dans le dossier *entree*.
Les sorties des automates sont dans le dossier *output (*en plus d'être affiché dans les logs).

- Aller à la racine du projet
- *make list* – permet de lister les test d'automate possible.
- *make automate/Nom* – permet de lancer la déterminisation de l'automate et la reconnaissance des mots.
- *make graph* – permet de générer les images avec dot.

Résulat d'un log :

Etat Départ → Caractere Lu / Caractere Ecris → Etat Arrivé

"word" Is a word or not from the language of the automaton
Sortie : ['x, 'x', ...] ; output word

L'automate déterminisé généré *« generatedAutomate.descr »* est à la racine du projet avec le fichier de log de l'automate.

Graph :

Les états finaux sont en noir et les états entourés sont les états initiaux. Le puit est l'état 0.

3 - Résultat Test Moteurs

AS.descr – AS.log

Quelques résultats :

```
1 -> p/p -> 2
2 -> m/m -> 3
3 -> m/# -> 3
3 -> p/p -> 2
2 -> m/m -> 3

"pmmpm"        Is a word from the language of the automaton
Sortie : ['p', 'm', 'p', 'm'] ; pmpm
```

```
1 -> m/m -> 3
3 -> p/p -> 2
2 -> p/# -> 2
2 -> p/# -> 2
2 -> m/m -> 3
3 -> p/p -> 2
2 -> p/# -> 2
2 -> m/m -> 3

"mpppmppm"   Is a word from the language of the automaton
Sortie : ['m', 'p', 'm', 'p', 'm'] ; mpmpm
```

```
1 -> p/p -> 2
2 -> p/# -> 2
2 -> p/# -> 2
2 -> m/m -> 3
3 -> m/# -> 3
3 -> m/# -> 3
3 -> m/# -> 3
3 -> p/p -> 2
2 -> p/# -> 2
2 -> p/# -> 2
2 -> m/m -> 3
3 -> m/# -> 3

"pppmmmmpppmm"   Is a word from the language of the automaton
Sortie : ['p', 'm', 'p', 'm'] ; pmpm
```

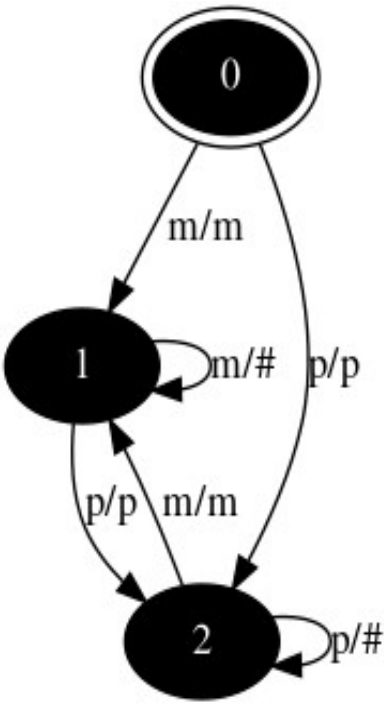

Figure 6: Automate Initial

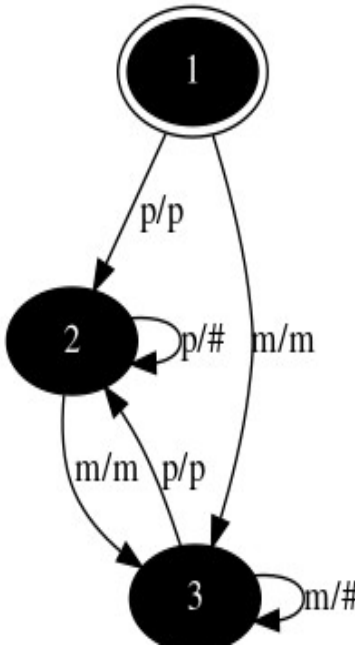

Figure 5: Automate déterministe

C0.descr - C0.log

Quelques résultats :

```
Error : a not in alphabet : V -> ['0', '1']
Error : b not in alphabet : V -> ['0', '1']
Error : c not in alphabet : V -> ['0', '1']

"abc"    Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> 1/# -> 2
2 -> 1/# -> 3

Error : 3 not in final states : F -> [1]
"11"     Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> 1/# -> 2
2 -> 0/b -> 1

"10"     Is a word from the language of the automaton
Sortie : ['b'] ; b
```

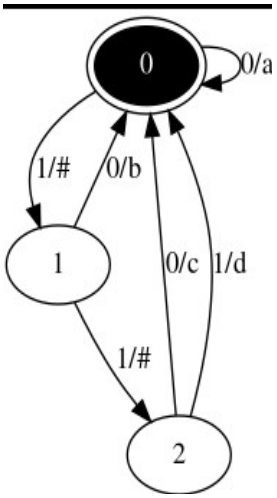

Figure 7: Automate Initial

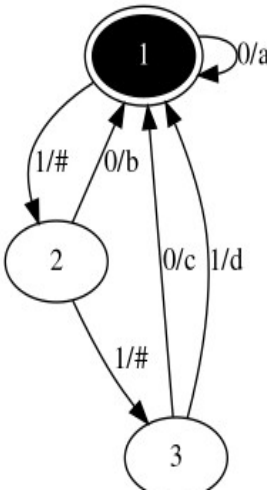

Figure 8: Automate déterministe

S2.descr – S2.log

Quelques résultats :

```
1 -> m/# -> 3
3 -> a/# -> 5
5 -> p/# -> 0
0 -> a/# -> 0

Error : 0 not in final states : F -> [8]
"mapa"          Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> p/# -> 2
2 -> a/# -> 4
4 -> p/# -> 6
6 -> a/H -> 8

"papa"  Is a word from the language of the automaton
Sortie : ['H'] ; H
```

```
1 -> m/# -> 3
3 -> a/# -> 5
5 -> m/# -> 7
7 -> a/# -> 9
9 -> n/F -> 8

"maman"         Is a word from the language of the automaton
Sortie : ['F'] ; F
```

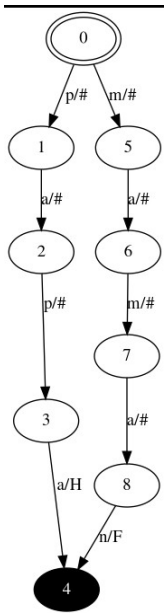

Figure 9: Automate Initial

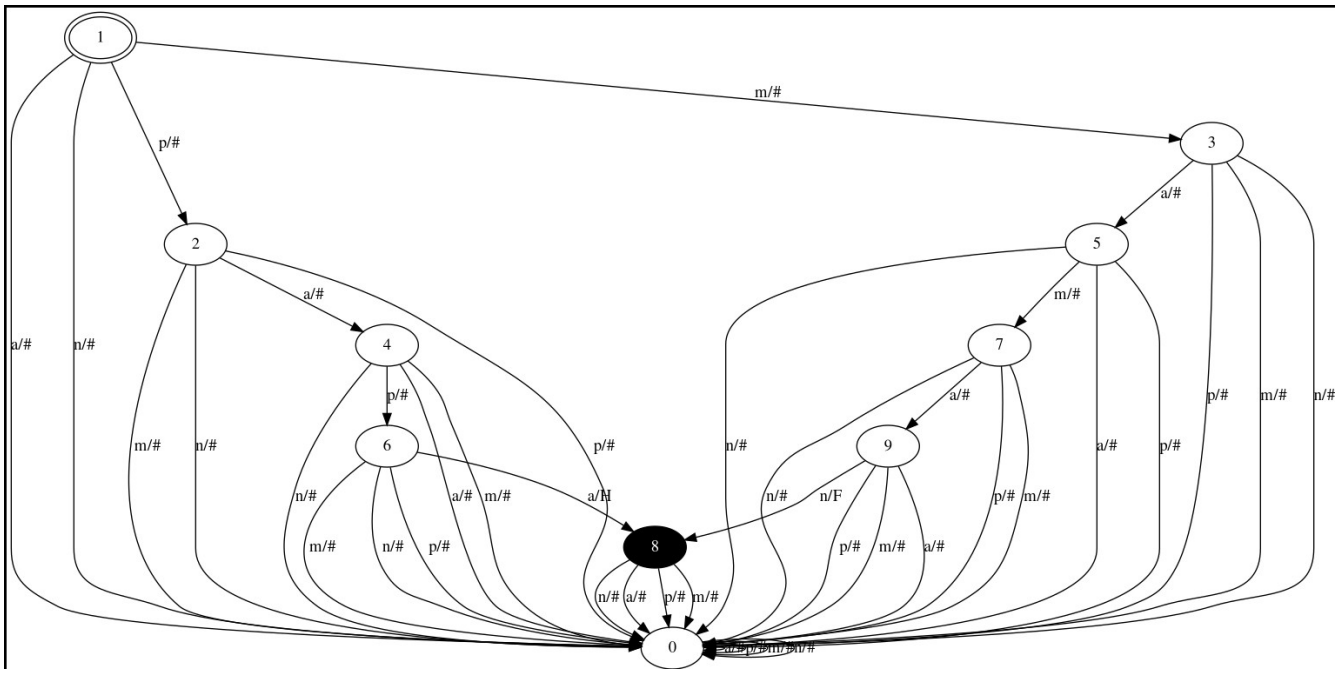

Figure 10: Automate déterministe

Mini.descr – Mini.log

Quelques résultats :

```
Error : 1 not in alphabet : V -> ['a']
Error : 0 not in alphabet : V -> ['a']
"1001"  Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 0

Error : 0 not in final states : F -> [1]
"a"     Is not a word from the language of the automaton
Sortie : [] ;
```

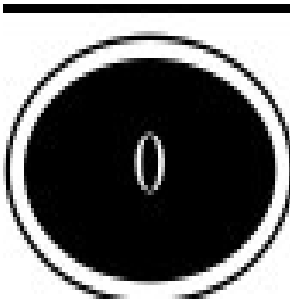

Figure 11: Automate Initial

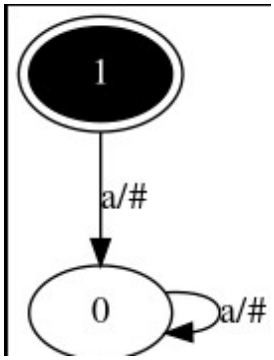

Figure 12: Automate déterministe

S0.descr – S0.log

Quelques résultats :

```
1 -> a/# -> 2
2 -> b/1 -> 1
1 -> b/# -> 0

Error : 0 not in final states : F -> [1]
"abb"    Is not a word from the language of the automaton
Sortie : ['1'] ; 1
```

```
1 -> a/# -> 2
2 -> b/1 -> 1
1 -> a/# -> 2
2 -> c/2 -> 1
1 -> a/# -> 2
2 -> b/1 -> 1
1 -> a/# -> 2
2 -> c/2 -> 1
1 -> a/# -> 2
2 -> b/1 -> 1
1 -> a/# -> 2
2 -> c/2 -> 1

"abacabacabac"          Is a word from the language of the automaton
Sortie : ['1', '2', '1', '2', '1', '2'] ; 121212
```

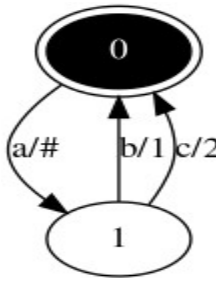

Figure 14: Automate Initial

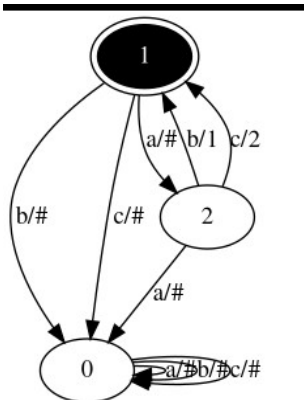

Figure 13: Automate déterministe

S1.descr – S1.log

Quelques résultats :

```
Error : d not in alphabet : V -> ['a', 'b']
Error : s not in alphabet : V -> ['a', 'b']
Error : f not in alphabet : V -> ['a', 'b']
"dsf"     Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> b/i -> 1

"ab"      Is a word from the language of the automaton
Sortie : ['i'] ; i
```

```
1 -> a/# -> 2
2 -> b/i -> 1
1 -> b/# -> 3
3 -> a/o -> 1

"abba"  Is a word from the language of the automaton
Sortie : ['i', 'o'] ; io
```

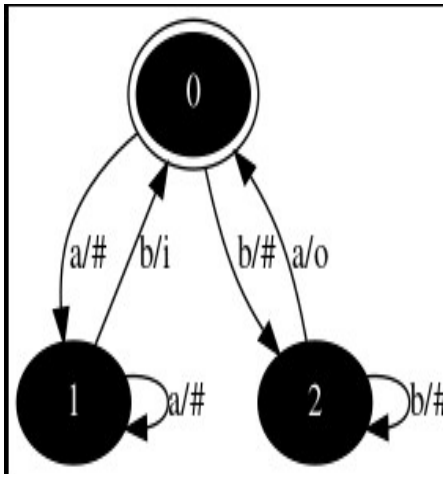

Figure 15: Automate Initial

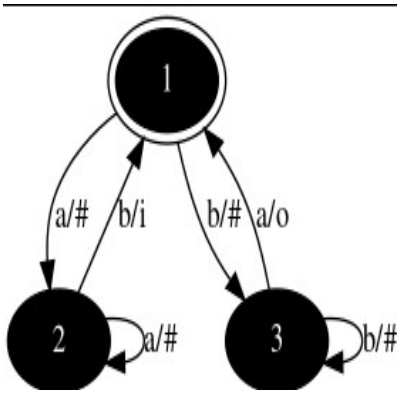

Figure 16: Automate déterministe

T0.descr – T0.log

Quelques résultats :

```
1 -> a/# -> 2
2 -> b/# -> 2
2 -> c/# -> 2
2 -> a/# -> 3
3 -> a/# -> 0
0 -> b/# -> 0
0 -> c/# -> 0

Error : 0 not in final states : F -> [3]
"abcaabc"        Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> a/# -> 3

"aa"      Is a word from the language of the automaton
Sortie : [] ;
```

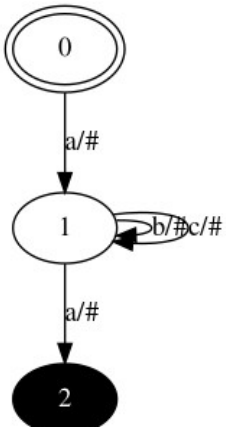

Figure 18: Automate Initial

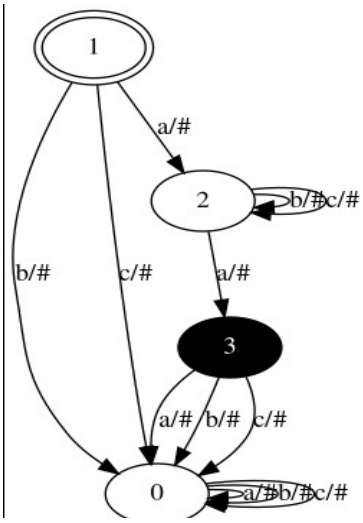

Figure 17: Automate déterministe

T1.descr – T1.log

Quelques résultats :

```
Error : 1 not in alphabet : V -> ['a', 'b']
Error : 2 not in alphabet : V -> ['a', 'b']
Error : 3 not in alphabet : V -> ['a', 'b']

"123"    Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> b/# -> 1

"ab"      Is a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> a/# -> 0
0 -> a/# -> 0
0 -> a/# -> 0

Error : 0 not in final states : F -> [1]
"aaaa"   Is not a word from the language of the automaton
Sortie : [] ;
```

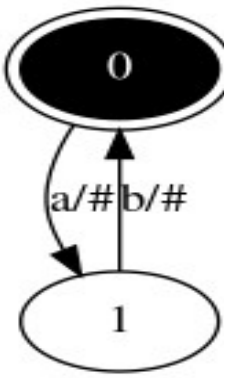

Figure 20: Automate Initial

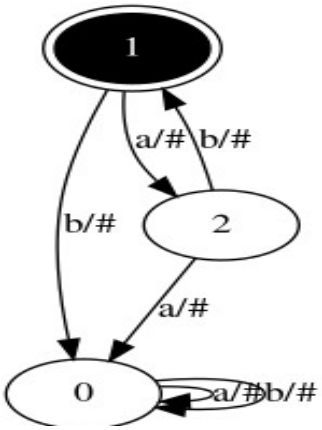

Figure 19: Automate déterministe

T6.descr – T6.log

Quelques résultats :

```
1 -> a/# -> 2
2 -> b/# -> 2
2 -> b/# -> 2
2 -> c/# -> 3

Error : 3 not in final states : F -> [4]
"abbc"  Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> b/# -> 2
2 -> c/# -> 3
3 -> b/# -> 4

"abcb"  Is a word from the language of the automaton
Sortie : [] ;
```

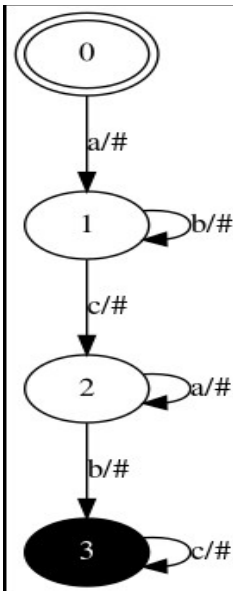

Figure 22: Automate Initial

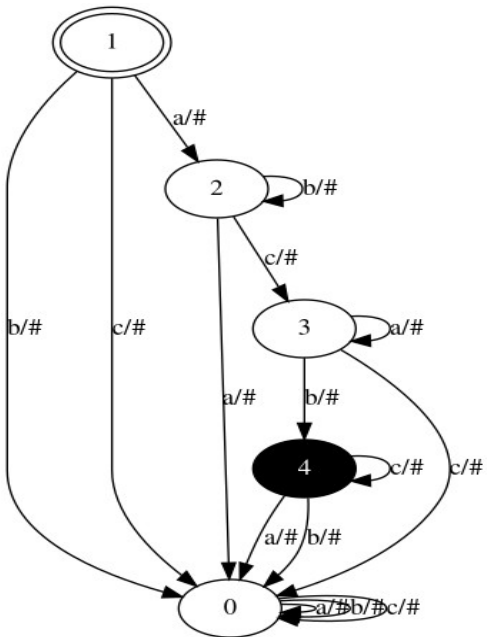

Figure 21: Automate déterministe

3 - Résultat Test Déterminisation

ND01.descr – ND01.log

Quelques résultats :

```
1 -> a/# -> 2
2 -> b/# -> 6
6 -> b/# -> 0
0 -> c/# -> 0

Error : 0 not in final states : F -> [5]
"abbc"  Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> a/# -> 5

"aa"    Is a word from the language of the automaton
Sortie : [] ;
```

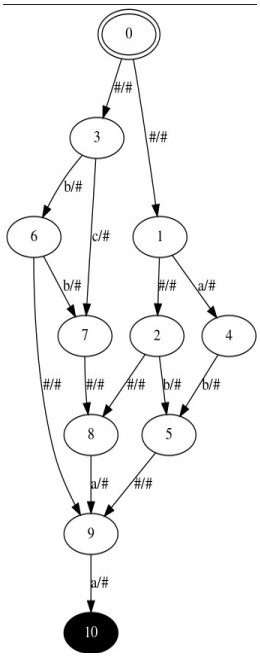

Figure 23: Automate Initial

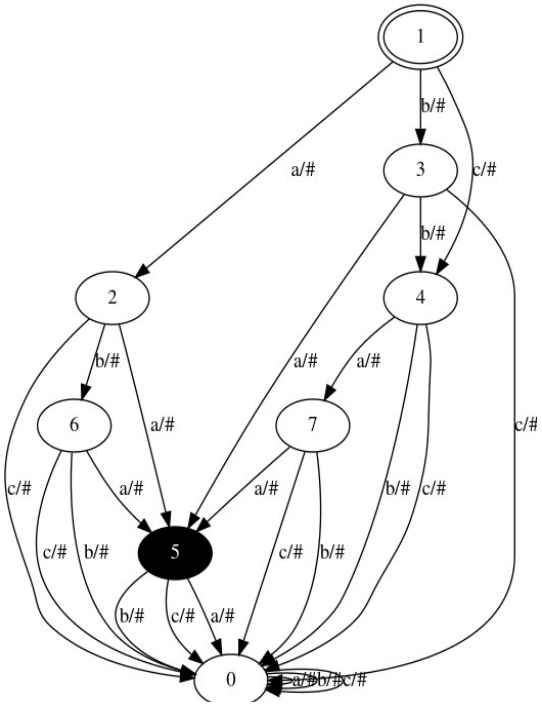

Figure 24: Automate déterministe

ND03.descr – ND03.log

Quelques résultats :

```
Error : c not in alphabet : V -> ['a', 'b']
"abbc"  Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> a/# -> 4

"aa"      Is a word from the language of the automaton
Sortie : [] ;
```

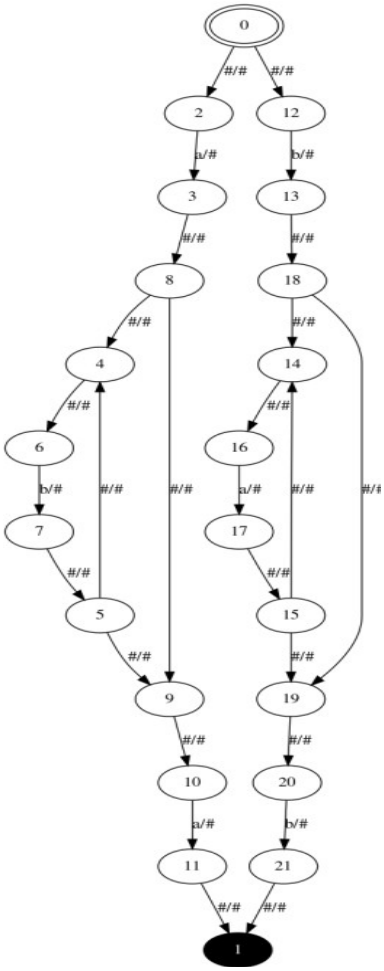

Figure 25: Automate Initial

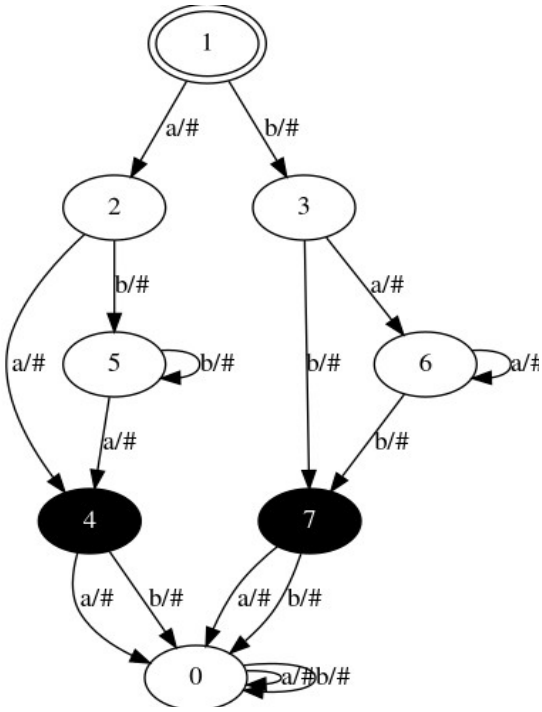

Figure 26: Automate déterministe

NDSL01.descr – NDSL01.log

Quelques résultats :

```
1 -> a/# -> 2
2 -> b/# -> 4
4 -> b/# -> 3
3 -> c/# -> 0

Error : 0 not in final states : F -> [3]
"abbc"  Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> b/# -> 4
4 -> b/# -> 3

"abb"    Is a word from the language of the automaton
Sortie : [] ;
```

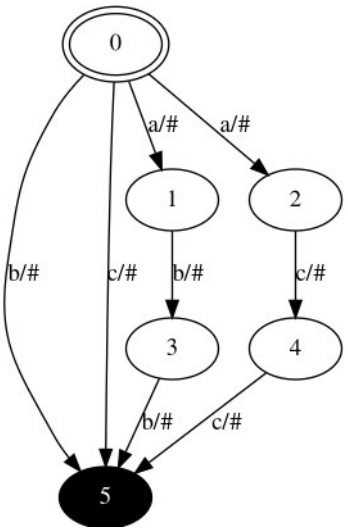

Figure 28: Automate Initial

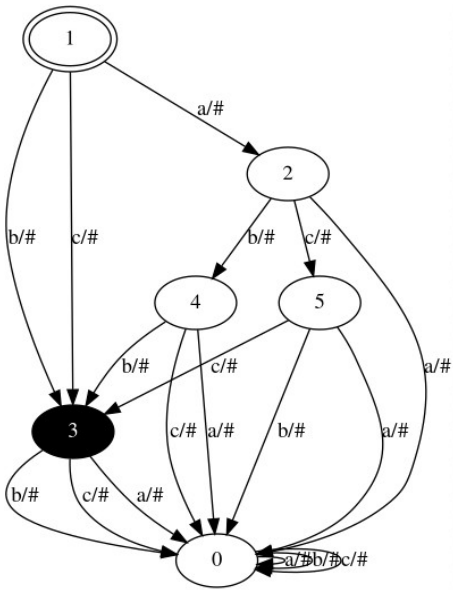

Figure 27: Automate déterministe

NDSL02.descr – NDSL02.log

Quelques résultats :

```
1 -> a/# -> 2
2 -> b/# -> 3
3 -> b/# -> 0
0 -> c/# -> 0

Error : 0 not in final states : F -> [5, 6, 7, 8]
"abbc"  Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> b/# -> 3

Error : 3 not in final states : F -> [5, 6, 7, 8]
"ab"    Is not a word from the language of the automaton
Sortie : [] ;
```

```
1 -> a/# -> 2
2 -> b/# -> 3
3 -> d/# -> 6

"abd"   Is a word from the language of the automaton
Sortie : [] ;
```

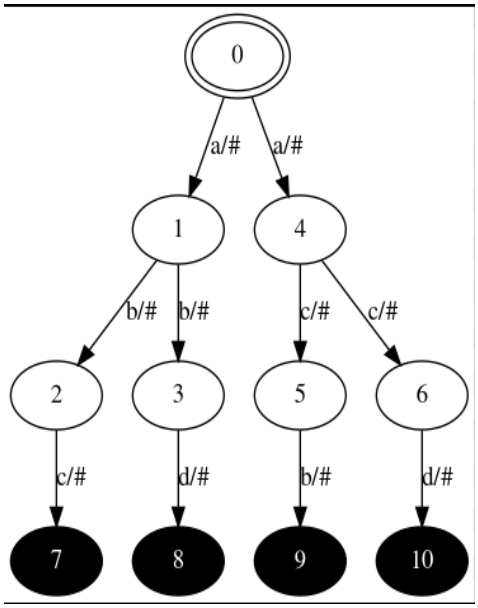

Figure 29: Automate Initial

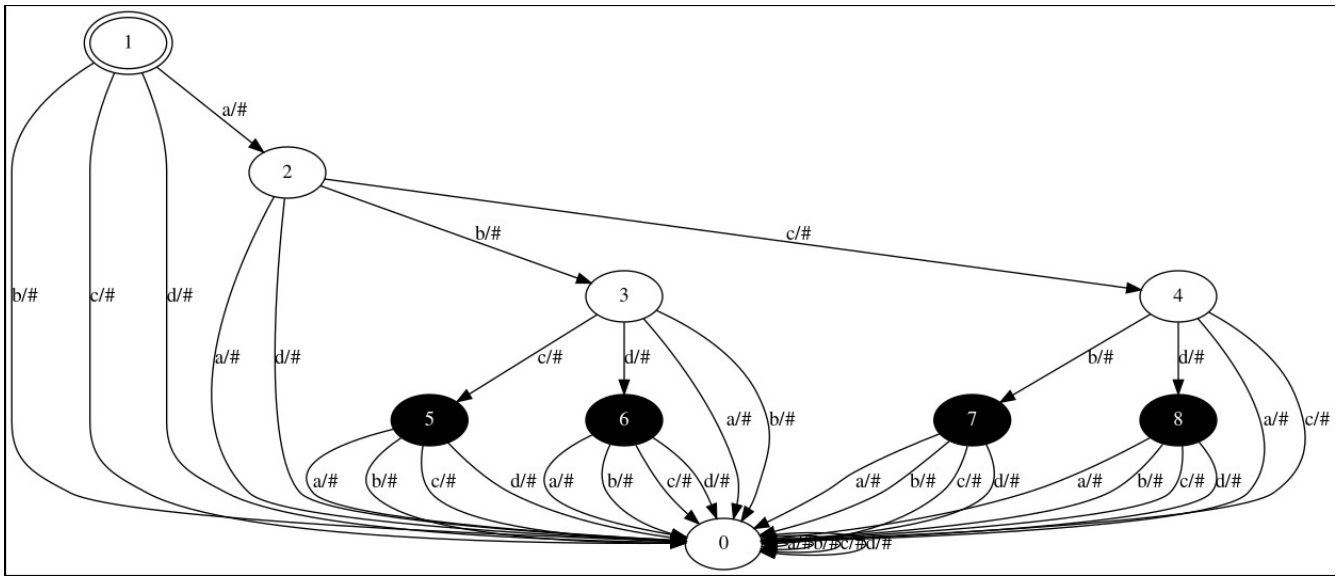

Figure 30: Automate déterministe

NDSL04.descr – NDSL04.log

Quelques résultats :

```
1 -> b/# -> 2
2 -> b/# -> 5
5 -> a/# -> 0
3 -> b/# -> 2
2 -> b/# -> 5
5 -> a/# -> 0
4 -> b/# -> 0
0 -> b/# -> 0
0 -> a/# -> 0
5 -> b/# -> 2
2 -> b/# -> 5
5 -> a/# -> 0

Error : 0 not in final states : F -> [1, 3, 4, 5]
"bba"    Is not a word from the language of the automaton
Sortie : [] ;
```

* test sur chaque état initial de l'automate déterministe

```
"aa"      Is a word from the language of the automaton
Sortie : [] ;

1 -> b/# -> 2
2 -> b/# -> 5
```

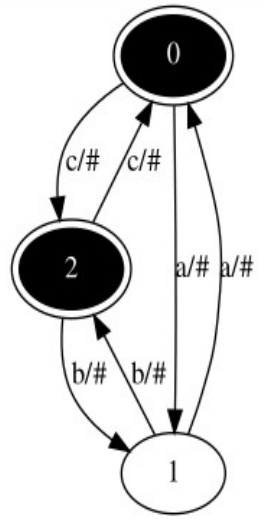

Figure 31: Automate Initial

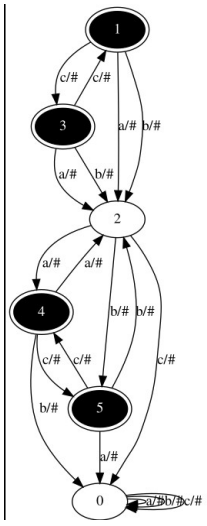

Figure 32: Automate déterministe