

# Rapport de projet : StartAutomaton

Nicolas ENDREDI, Baptiste ORUEZABAL

27 mars 2013

## 0.1 Présentation du projet

### 0.1.1 Automates et expressions rationnelles

Les expressions rationnelles sont une manière très simple de décrire un langage, notamment des langages définis récursivement. Cependant cette représentation n'est pas pratique pour implémenter des programmes analysant le langage. On utilise donc des automates, créés à partir de ces expressions. On sait, d'après le théorème de Kleene, qu'un langage est reconnu par un automate si et seulement si il peut être décrit par une expression rationnelle. Ainsi toutes les expressions rationnelles traitées durant ce projet, ont un automate correspondant.

### 0.1.2 La bibliothèque Automaton

La bibliothèque Python "Automaton" implémente l'objet "automaton" qui est un automate simple : on peut lui définir son alphabet, ses états, ses états finaux et initiaux et ses transitions.

Pour représenter les "epsilon transitions", on peut définir des caractères qui seront traités comme des "epsilon".

### 0.1.3 Sujet du projet

Le but du projet est d'ajouter des fonctionnalités à la bibliothèque "Automaton" en utilisant des algorithmes vus dans le cours d'Informatique Théorique. Ces fonctionnalités sont des manipulations d'automate ainsi que la conversion d'une expression rationnelle en automate

## 0.2 Fonctionnalités de StartAutomaton

Ici sont listées les différentes fonctionnalités implémentées dans StartAutomaton. Les fonctionnalités demandées dans le projet apparaissent dans la table des matières. Les fonctionnalités supplémentaires ne sont visibles qu'ici.

### 0.2.1 Affichage

#### Afficher l'alphabet de l'automate

Affiche un "pretty set" contenant l'alphabet de l'automate

#### Afficher toutes les transitions de l'automate

Affiche un "pretty set" contenant toutes transitions de l'automate

#### Afficher tous les états de l'automate

Affiche un "pretty set" contenant tous les états de l'automate

#### Afficher les caractères epsilon

Affiche un "pretty set" contenant les caractères qui représentent epsilon dans l'automate

### **Afficher les états finaux**

Affiche un "pretty set" contenant tous les états finaux de l'automate

### **Afficher les état initiaux**

Affiche un "pretty set" contenant tous les états initiaux de l'automate

## **0.2.2 Tests**

### **Tester si l'automate est déterministe**

On renvoie un booléen signifiant si l'automate est déterministe ou non

### **Tester si l'automate est complet**

On renvoie un booléen signifiant si l'automate est complet ou non

## **0.2.3 Sélecteurs et éditeurs**

### **Obtenir l'origine d'une transition**

On renvoie l'état à l'origine d'une transition donnée

### **Obtenir la destination d'une transition**

On renvoie l'état de destination d'une transition donnée

### **Obtenir les epsilon transitions de l'automate**

On renvoie la liste des transitions avec un caractère codant l'épsilon

### **Supprimer un état initial donné de l'automate**

On enlève l'état de la liste des états initiaux de l'automate (ne supprime pas l'état de la liste d'états et ne supprime pas les transitions dans lesquelles il est présent)

### **Supprimer un état final donné de l'automate**

On enlève l'état de la liste des états finaux de l'automate (ne supprime pas l'état de la liste d'états et ne supprime pas les transitions dans lesquelles il est présent)

### **Supprimer un transition donnée de l'automate**

On enlève la transition de la liste des transitions de l'automate (ne supprime pas les états de la liste d'états)

### **Supprimer tous les états initiaux de l'automate**

On vide la liste des états initiaux de l'automate

### **Supprimer tous les états finaux de l'automate**

On vide la liste des états finaux de l'automate

## **Supprimer toutes les transitions de l'automate**

On vide le dictionnaire stockant les transitions

## **Supprimer toutes les epsilon transitions (avec réagencement des autres transitions)**

On supprime les transitions par un caractère encodant epsilon, sans pour autant changer le langage reconnu par l'automate

## **0.2.4 Manipulation d'automates**

### **Compléter**

On ajoute à chaque état les transitions par les lettres de l'alphabet qu'il n'avait pas. Ces transitions sont toutes vers un état puits.

### **Déterminiser**

On fusionne des états de l'automate afin que tous les états n'aient pas deux transitions sortantes avec la même et qu'il n'y ait qu'un seul état initial

### **Miroir**

On inverse les état finaux et initiaux, puis on renverse les transitions

### **Union de deux automates**

On fait l'union de deux automates déterminisés (si non déterministes) et complétés (si non complets)

### **Intersection de deux automates**

On fait l'union de deux automates déterminisés (si non déterministes) et complétés (si non complets)

### **Complément**

On crée l'automate permettant de compléter l'automate original

### **Minimiser**

On minimise l'automate. On crée donc le plus petit automate reconnaissant le langage même langage que l'automate d'origine

## **Créer un sous automate à partir d'une epsilon transition représentant un "+" lu précédemment dans l'expression**

On crée un sous automate à partir de l'expression en paramètre et renvoie tous les états finaux construits par ce sous automate

### **Créer un sous automate à partir d'une "\*" lue précédemment dans l'expression**

On crée un sous automate à partir de l'expression en paramètre et renvoie l'état initial comme état final

### **Créer un sous automate à partir d'une epsilon transition représentant un "." lu précédemment dans l'expression**

On crée un sous automate à partir de l'expression en paramètre et renvoie le dernier état final construit par ce sous automate

### **Reconnaitre l'opérateur et l'action à réaliser pour une expression**

Si l'expression commence par un opérateur, on crée un sous automate avec une des fonctionnalités ci-dessus.

Sinon, on crée juste une transition depuis l'état donné en paramètre vers un nouvel état, par la lettre lue

## **0.2.5 Conversions et utilitaire**

### **Conversion d'une expression rationnelle en automate**

On crée l'automate permettant de reconnaître le langage décrit par l'expression rationnelle donnée

### **Renverser un tuple à trois éléments**

On inverse le premier et le dernier élément sans toucher au deuxième

## **0.3 Problèmes rencontrés**

### **0.3.1 La bibliothèque Automaton**

Le temps nécessaire à la compréhension de toutes les méthodes implémentées par la bibliothèque Automaton a été considérable. Notamment pour comprendre les types d'objets utilisés pour stocker les propriétés d'un automate.

Aussi la fonction "remove\_epsilon\_transitions()" était un peu ambiguë, donc nous avons décidé de redéfinir cette méthode et d'ajouter la méthode "remove\_epsilons()" qui supprime les caractères encodant epsilon (ce que faisait la méthode remove\_epsilon\_transitions() originale)

### **0.3.2 Méthodes de manipulation d'automates**

#### **Union et intersection**

Initialement nous avons voulu commencer par implémenter ces fonctions qui nous semblaient plus simples, mais nous avons très vite vu qu'il serait impossible de les implémenter simplement sans la méthode "déterminiser"

## Déterminiser

Cette méthode est celle qui nous a demandé le plus de temps et de travail. Elle a été faite à deux et a passé beaucoup plus de tests que les autres puisque d'elle dépendent une bonne partie des autres méthodes

## Minimiser

Une première version de minimiser a été tentée sans utiliser le double renversement, mais cette version de la méthode ne fonctionnait pas pour tous les automates. Nous l'avons donc abandonnée au profit du double renversement, plus clair et plus simple à implémenter

## Conversion d'une expression rationnelle vers un automate

Cette méthode a été rapide à coder, mais longue à concevoir. On a étudié l'algorithme de Thomson et essayé de concevoir un algorithme, qui n'aurait pas besoin d'autant d'épsilon transitions. Malgré de nombreux essais ratés, la version définitive semble correctement fonctionner pour tous les cas que nous avons testés.

## Conversion d'une expression rationnelle en préfixée

Par manque de temps, nous avons préféré ne pas implémenter cette fonctionnalité, pour nous concentrer sur la re-factorisation de notre code et sa documentation.

## 0.4 Projection

### 0.4.1 Implémentation de la conversion d'une expression rationnelle en préfixée

L'implémentation de cette fonctionnalité devrait prendre moins d'une semaine selon nos calculs. Ainsi on aimerait pouvoir l'implémenter avant la soutenance de ce projet

### 0.4.2 Création de classes utilisables par StartAutomaton

Lors de la re-factorisation du code, nous avons remarqué que certaines actions méritent d'avoir une classe propre pour augmenter la lisibilité du code

## 0.5 Conclusion

Presque toutes les fonctionnalités demandées ont été implémentées avec un code assez lisible et des fonctionnalités utiles ont été ajoutées. De plus la plus part des méthodes ont été factorisées pour les rendre encore plus lisibles. Ainsi nous pensons avoir respecté les demandes du sujet.

Cependant, on a pu constater qu'il y a de nombreuses optimisations à faire dans les algorithmes que ce soit au niveau du temps ou de la mémoire utilisée, mais, le temps ne nous le permettant pas, nous laissons cela au soins de possibles développeurs qui amélioreront notre classe StartAutomaton

# Table des matières

0.1	Présentation du projet . . . . .	1
0.1.1	Automates et expressions rationnelles . . . . .	1
0.1.2	La bibliothèque Automaton . . . . .	1
0.1.3	Sujet du projet . . . . .	1
0.2	Fonctionnalités de StartAutomaton . . . . .	1
0.2.1	Affichage . . . . .	1
0.2.2	Tests . . . . .	2
0.2.3	Sélecteurs et éditeurs . . . . .	2
0.2.4	Manipulation d'automates . . . . .	3
	Compléter . . . . .	3
	Déterminiser . . . . .	3
	Miroir . . . . .	3
	Union de deux automates . . . . .	3
	Intersection de deux automates . . . . .	3
	Complément . . . . .	3
	Minimiser . . . . .	3
0.2.5	Conversions et utilitaire . . . . .	4
	Conversion d'une expression rationnelle en automate . . . . .	4
0.3	Problèmes rencontrés . . . . .	4
0.3.1	La bibliothèque Automaton . . . . .	4
0.3.2	Méthodes de manipulation d'automates . . . . .	4
	Union et intersection . . . . .	4
	Déterminiser . . . . .	5
	Minimiser . . . . .	5
	Conversion d'une expression rationnelle vers un automate . . . . .	5
	Conversion d'une expression rationnelle en préfixée . . . . .	5
0.4	Projection . . . . .	5
0.4.1	Implémentation de la conversion d'une expression rationnelle en préfixée . . . . .	5
0.4.2	Création de classes utilisables par StartAutomaton . . . . .	5
0.5	Conclusion . . . . .	5