Projet – Apprentissage automatique de programmes de validation de chaînes de caractères

INFO-F302 - Informatique fondamentale

Professor :
Emmanuel FILIOT

Author :
Ilias Zahafi
Mohamad CHANAA
Jordan Pacha MATIN

15 December 2023

Contents

1 Introduction

Nous nous intéressons dans le cadre de ce projet à la généralisation de langages acceptés via les automates finis. Ces derniers s'avèrent être d'une efficacité redoutable pour cette tâche (et pas que). Nous allons augmenter au fur et mesure la difficulté de leur implémentation, en spécifiant graduellement davantage de contraintes autant sur les mots du langage accepté (ou rejeté) que sur la structure de l'automate lui même.

2 Question 1

2.1 Choix des variables

Variable 1 : $T_{q,l,q'}$ avec $q, q' \in Q$, $l \in \Sigma$, $T_{q,l,q'}$ vrai $\iff$ $\delta(q,l) = q'$.

Variable 2 : A_q,avec q $\in$ Q, A_q vrai $\iff$ q $\in$ F.

Variable 3 : $E_{m,n,q}$ avec m $\in \Sigma^*$, n $\in \{0,..,len(m)\}$ et q $\in$ Q, $E_{m,n,q}$ vrai $\iff$ après avoir exécuté les n premières lettres de m, on se trouve à l'état q.

2.2 Contraintes

Contrainte 1 (Déterminisme) : Pour chaque état q, pour chaque lettre de l'alphabet, il ne peut y avoir qu'un état q' liant q.

$\forall\ q, q', q'' \in Q$, avec $q' \neq q''$, $\forall$ l $\in \Sigma$:

$$T_{q,l,q'} \rightarrow \neg(T_{q,l,q''})$$

FNC

$$\bigwedge_{\substack{q,q',q'' \in Q \\ q' \neq q''}} \bigwedge_{l \in \Sigma} (\neg \mathrm{T}_{q,l,q'} \vee \neg T_{q,l,q''})$$

Contrainte 2 : Pour chaque exécution d'un mot, on part de l'état initial.

$\forall$m $\in \Sigma^*$:

$$E_{m,0,q_0}$$

FNC :

$$\bigwedge_{m \in \Sigma^*} E_{m,0,q_0}$$

Contrainte 3 : Pour chaque lettre de chaque mot positif, il existe une unique exécution.

$\forall m \in P$, $\forall$ n $\in \{0, ..., len(m)\}$, $\exists!$ q $\in Q$:

$$E_{m,n,q}$$

FNC :

$$\bigwedge_{m \in P} \bigwedge_{n=0}^{len(m)} \wedge ((\bigvee_{q \in Q} E_{m,n,q}) \wedge (\bigwedge_{\substack{q,q' \in Q \\ q \neq q'}} (\neg E_{m,n,q \vee \neg E_{m,n,q'}})))$$

Contrainte 4 : Pour chaque lettre de chaque mot négatif, il existe au plus une exécution.

$\forall m \in$N, $\forall$ n $\in \{0, ..., len(m)\}$, $\exists$ q $\in Q$:

$$E_{m,n,q}$$

FNC :

$$(\bigwedge_{m \in N} \bigwedge_{n=0}^{len(m)})(\bigvee_{q \in Q} E_{m,n,q})$$

Contrainte 5 : Pour chaque exécution, s'il existe une transition, alors la prochaine exécution se trouve à l'états après transition.

$\forall m \in \Sigma^*$, $\forall$ n $\in \{0, ..., len(m) - 1\}$, $\forall q,q' \in$ Q :

$$(E_{m,n,q} \wedge T_{q,ln,q'}) \rightarrow E_{m,q',q}$$

FNC :

$$\bigwedge_{n=0}^{len(m)-1} \bigwedge_{q,q' \in Q} (\neg E_{m,n,q} \vee \neg T_{q,ln,q'} \vee E_{m,n+1,q'})$$

Contrainte 6 : Pour chaque lettre de chaque mot positif, si après avoir exécuté les n premières lettres on se trouve à l'état q alors il existe une transition de l'état q vers un autre état par la n+1 ième lettre.

$\forall m \in$ P, $\forall n \in \{0, ..., len(m) - 1\}$, $\forall q \in$ Q, $\exists p \in$ Q :

$$E_{m,n,q} \rightarrow T_{q,ln+1,p}$$

FNC :

$$\bigwedge_{m \in P} \bigwedge_{n=0}^{len(m)-1} \bigwedge_{q \in Q} (\bigvee_{p \in Q} (\neg E_{m,n,q} \vee T_{q,ln+1,p})$$

Contrainte 7 : L'exécution finale des mots positifs finie dans un état acceptant.

$\forall m \in P, \exists q \in Q$:

$$E_{m,len(m),q} \rightarrow A_q$$

FNC :

$$\bigwedge_{m \in P} \bigvee_{q \in Q} (\neg E_{n,len(m),q} \vee A_p)$$

Contrainte 8 : L'exécution finale des mots négatifs ne finie pas dans un état acceptant.

$\forall m \in N, \forall q \in Q$

$$(E_{n,len(m),q} \rightarrow \neg A_q)$$

FNC :

$$\bigwedge_{n \in N} \bigwedge_{q \in Q} (\neg E_{m,len(m),q} \vee \neg A_q)$$

3 Question 2

Voir project.py.

4 Question 3

Pour cette question, nous allons utiliser la fonction de la question 2 en faisant une boucle commencant par k = 0 et en incrémentant le k de 1 jusqu'à ce que la fonction nous retourne un automate.

5 Question 4

Pour réponde à cette question, il suffit de reprendre la fonction de la question 2 et de vérifier que pour chaque lettre et pour chaque état, il existe une transition en destination d'un état.

$\forall l \in \Sigma$, $\forall q \in Q$, $\exists p \in Q$:

$$T_{q,l,p}$$

FNC :

$$\bigwedge_{l \in \Sigma} \bigwedge_{q \in Q} (\bigvee_{p \in Q} T_{q,l,p})$$

6 Question 5

Pour répondre à cette question, remarquons que la contrainte de déterminisme est assez proche de la contrainte de réversibilité si ce n'est que l'unicite de la transition se fait dans le sens contraire. En effet, il suffit de modifier légèrement la condition de déterminisme pour obtenir celle de réversibilité et enfin de l'ajouter à la fonction de la question 2.
$\forall l \in \Sigma, \forall q, q', q'' \in Q$ avec $q' \neq q''$:

$$T_{q\prime,l,q} \rightarrow T_{q'',l,q}$$

FNC :

$$\bigwedge_{l\in\Sigma} \bigwedge_{\substack{q,q',q''\in Q \\ q'\neq q''}} (\neg T_{q',l,q} \vee \neg T_{q'',l,q})$$

7 Question 6

Pour cette question, nous allons utiliser la fonction de la question 2 et créer une liste qui contient l'ensemble des états acceptants que l'on transmettra en condition au solveur grace a la commande addatmost de ne posséder au maximum l états acceptants.

8 Question 7

Nous obtenons une implémentation (à nouveau) assez similaire, à défaut que les fonctions de transitions ne sont plus déterministes et donc, pour chaque état, nous devons envisager qu'il y ait, pour un symbole, plusieurs états joignables. Pour ce faire, nous avons supprimé la contrainte sur le déterminisme définie par la fonction "determined" et nous avons défini la fonction "nondetermined" qui ne fait que passer cette contrainte. De plus, puisque un mot est positif s'il existe une exécution validante pour ce mot, nous devons considérer toute exécution possible sur ce mot. À l'inverse, puisque un mot est invalidé si et seulement si pour toute exécution de ce mot l'exécution n'est pas acceptée, il suffit de parcourir l'ensemble des exécutions de ce mot et de s'assurer qu'aucune parmi elles n'est satisfaisante.

9 Question 8

Nous utilisons l'astuce mentionnée à la question 6. En effet, nous souhaitons imposer une contrainte sur le nombre de transitions, et pour ce faire, nous allons appeler les méthodes de la classe fournie afin d'établir une contrainte de cardinalité sur les transitions. Nous observons également qu'en cas extrême, c'est-à-dire lorsque chaque transition est définie sur un état différent, le nombre total d'états sera exactement k+1. Par conséquent, nous pouvons ajuster notre contrainte sur le nombre d'états, sachant que ceux-ci ne peuvent pas dépasser k+1.

10 Conclusion

Nous avons exhibé plusieurs types d'automates, non complets complets, déterministes, non déterministes, etc.., Ce sont des outils puissants, dont l'implémentation reste relativement compacte. Un des grands avantages, qu'ils procurent, et exacerbé dans ce rapport, est la facilité avec laquelle nous pouvons switcher d'une condition à une autre. En effet, nous avons parcouru un certain nombres de contraintes différentes sans pour autant en changer grandement la structure. Par ailleurs, les automates sont des outils en alignement total avec la déduction naturelle vue dans un premier temps dans le cours. Sans les clauses formulées en FNC nous aurions beaucoup de mal à implémenter quoique ce soit. Les automates sont donc des outils puissants, et à la fois une application directe de la logique propositionnelle, pilier du cours INFOF-302