Université Libre de Bruxelles

INFO-F302
Informatique Fondamentale

Rapport Informatique Fondamentale

Étudiants :
Hugo Callens
Rayan Contuliano Bravo
Ethan Rogge

Enseignants :
E. Filliot
R. Petit

13 décembre 2023

1 Modélisation d'un automate déterministe en FNC

Voici quelques notations utilisées dans la suite de ce rapport :

- P représente l'ensemble des mots acceptés par l'automate
- N représente l'ensemble des mots non-acceptés par l'automate
- Σ représente l'alphabet de l'automate
- k représente le nombre au plus d'états de l'automate

1.1 Choix des variables

1.1.1 Etats

L'ensemble Q est défini comme suit : $Q = \{q_{0a}, q_{0na}, \ldots, q_{la}, q_{lna} | l \leq k\}$ contenant ainsi $2k$ états dans Q pour un automate à k états. Les variables q_i peuvent être définies comme suit :

- $q_{ia} = 1$ si l'état i est acceptant et donc $q_{ina} = 0$.
- $q_{ina} = 1$ si l'état i est non-acceptant et donc $q_{ia} = 0$.

1.1.2 Transitions

L'ensemble δ est défini comme suit : $\delta = \{d_{i,l,j} | i, j \in Q, l \in \Sigma\}$, représentant l'ensemble des transitions de l'automate. Les variables $d_{i,l,j}$ peuvent être définies comme suit :

- $d_{i,l,j} = 1$ si la transition avec la lettre l existe entre l'état i et l'état j.
- $d_{i,l,j} = 0$ si la transition avec la lettre l n'existe pas entre l'état i et l'état j.

1.1.3 Exécutions

L'ensemble E est défini comme suit : $E = \{e_{m,i,t} | i \in Q, t \in \{-1, \ldots, \text{len}(m)\} m \in P \cup N\}$, représentant l'ensemble des exécutions de l'automate. Les variables $e_{m,i,t}$ peuvent être définies comme suit :

- $e_{m,i,t} = 1$ si l'automate est dans l'état i après avoir lu les t premières lettres du mot m.
- $e_{m,i,t} = 0$ si l'automate n'est pas dans l'état i après avoir lu les t premières lettres du mot m.

1.2 Contraintes

1. Il y a un unique état initial :
$$q_{0a} \vee q_{0na}$$
2. Un état est exclusivement acceptant ou non-acceptant :
$$\bigwedge_{\substack{q \in Q \\ i \in \{1,\ldots,k\}}} q_{ia} \rightarrow \neg q_{ina} \equiv \bigwedge_{\substack{q \in Q \\ i \in \{1,\ldots,k\}}} \neg q_{ia} \vee \neg q_{ina}$$
3. Chaque état a au plus une transition par lettre de l'alphabet :
$$\bigwedge_{\substack{l \in \Sigma \\ i,j,q \in Q}} d_{i,l,j} \rightarrow \neg d_{i,l,q} \equiv \bigwedge_{\substack{l \in \Sigma \\ i,j,q \in Q}} \neg d_{i,l,j} \vee \neg d_{i,l,q}$$
4. Un état i est acceptant s'il existe une exécution d'un mot m de P qui se termine sur l'état

i à l'étape $t = len(m)$:

$$\bigwedge_{\substack{t=len(m)\\q\in Q\\m\in P\\i\in\{1,\dots,k\}}} e_{m,i,t} \to q_{ia} \equiv \bigwedge_{\substack{t=len(m)\\q\in Q\\m\in P\\i\in\{1,\dots,k\}}} \neg e_{m,i,t} \vee q_{ia}$$

5. Si on a une exécution pour le mot m à l'étape t sur l'état i et une transition de i vers j pour la lettre l, alors, on a une exécution pour le mot m à l'étape $t+1$ sur l'état j :

$$\bigwedge_{\substack{i,j\in Q\\l\in\Sigma\\m\in P\\t\in\{0,\dots,len(m)\}\\l=m[t]}} (e_{m,i,t} \wedge d_{i,l,j} \to e_{m,j,t+1}) \equiv \bigwedge_{\substack{i,j\in Q\\l\in\Sigma\\m\in P\\t\in\{0,\dots,len(m)\}\\l=m[t]}} (\neg e_{m,i,t} \vee \neg d_{i,l,j} \vee e_{m,j,t+1})$$

6. Si on a une exécution pour le mot m à l'étape t sur l'état i et une exécution pour le même mot m à l'étape $t+1$ sur l'état j, alors, il existe une transition de i vers j pour la lettre l :

$$\bigwedge_{\substack{i,j\in Q\\l\in\Sigma\\m\in P\\t\in\{0,\dots,len(m)\}\\l=m[t]}} (\neg e_{m,i,t} \vee \neg e_{m,j,t+1} \vee d_{i,l,j}) \equiv \bigwedge_{\substack{i,j\in Q\\l\in\Sigma\\m\in P\\t\in\{0,\dots,len(m)\}\\l=m[t]}} (\neg e_{m,i,t} \vee \neg e_{m,j,t+1} \vee d_{i,l,j})$$

7. Toutes les exécutions sur les mots de N doivent se terminer sur un état non-acceptant :

$$\bigwedge_{\substack{m\in N\\i\in\{1,\dots,k\}q\in Q\\t=len(m)}} e_{m,i,t} \to q_{ina} \equiv \bigwedge_{\substack{m\in N\\i\in\{1,\dots,k\}q\in Q\\t=len(m)}} \neg e_{m,i,t} \vee q_{ina}$$

8. Toutes les exécutions sur les mots de P doivent exister :

$$\bigwedge_{m\in P} \bigvee_{\substack{t\in\{0,\dots,len(word)\}\\i\in\{1,\dots,k\}}} e_{m,i,t+1}$$

9. Toutes les exécutions doivent commencer à l'état initial :

$$\bigwedge_{m\in P\cup N} e_{m,1,0}$$

2 Algorithme de minimisation d'un automate déterministe

Afin de minimiser le nombre d'états d'un automate fini sur Σ, une approche itérative est utilisée en considérant :

$$k \in \{0, \dots, \infty\}$$

Etant donné que l'automate est fini, il existe un nombre fini d'états, éliminant ainsi la nécessité d'une borne supérieure. Pour chaque valeur de k, les contraintes précédemment définies (cf. 1) sont utilisées dans le but de déterminer le k minimal pour lequel le résultat n'est pas `None`.

3 Modélisation d'un automate déterministe complet en FNC

Il suffit de reprendre les contraintes précédentes (cf. 1) et d'ajouter la contrainte suivante :

1. Tous les états doivent avoir au moins une transition par lettre de l'alphabet :

$$\bigwedge_{l \in \Sigma} \bigvee_{i,j \in Q} d_{i,l,j}$$

Du fait que nous avons au moins une transition par lettre de l'alphabet et au plus une transition par lettre de l'alphabet, nous avons donc exactement une transition par lettre de l'alphabet pour chaque état, ce qui implique que l'automate est complet.

4 Modélisation d'un automate réversible en FNC

Il suffit de rajouter aux contraintes précédentes (cf. 1) la contrainte suivante :

1. Toutes les transitions doivent être réversibles, autrement dit, si il existe une transition de i vers j pour la lettre l, alors il existe une transition de j vers i pour la lettre l :

$$\bigwedge_{\substack{i,j \in Q \\ l \in \Sigma}} d_{i,l,j} \leftrightarrow d_{j,l,i} \equiv \bigwedge_{\substack{i,j \in Q \\ l \in \Sigma}} (\neg d_{i,l,j} \vee d_{j,l,i}) \wedge (\neg d_{j,l,i} \vee d_{i,l,j})$$

5 Modélisation d'un automate fini complet avec au plus l états en FNC

Pour que l'automate fini complet ait au plus l états, il suffit de rajouter aux contraintes précédentes (cf. 3) la contrainte suivante (suivant le formalisme de la documentation de PySAT) :

1. Il y a au plus l états acceptants :

$$\sum_{i=0}^{k} q_{ia} \leq l$$

Cette contrainte a été ajoutée au solveur `MiniCard`.