Langages formels et automates – cours 4

Algorithmes d'automates, déterminisation et complémentation

Catalin Dima

Algorithmes de décision

- **Appartenence :** Étant donné un automate $\mathcal{A}$ et un mot $w \in \Sigma^*$, décider si $w \in L(\mathcal{A})$:
  - Parcourir en largeur l'arbre de derouelements de l'automate sur le mot w.
- **Langage vide :** Étant donné un automate $\mathcal{A}$, décider si $L(\mathcal{A}) = \emptyset$.
  - Tester s'il existe une paire (q, r) avec $q \in Q_0, r \in Q_f$ qui est reliée par un chemin.
  - Autrement : parcourir en largeur le graphe de l'automate, en insérant dans un ensemble *Reach* tous les états qui se trouvent sur une trajectoire initialisée (c.à.d. partant d'un état initial).
  - *Reach* est l'ensemble des **états atteignables**.
  - Laquelle des deux techniques est la meilleure/plus rapide ?
  - On a aussi des états co-atteignables : des états à partir desquels on peut atteindre par une trajectoire un état final.
  - Exemples ....
- **Langage infini :** Étant donné un automate $\mathcal{A}$, décider si $card(L(\mathcal{A})) < \infty$.
  - Tester s'il existe un état q atteignable, co-atteignable et qui
    - Soit contient une boucle (q, a, q),
    - Soit appartient à une composante fortément connexe ayant au moins deux éléments.

L'intersection dans la modélisation des systèmes

- Modélisation de la concurrence :
- Modèle d'automate fini pour chaque processus :

```
while (true) {                        while (true) {
   flag1 := true;                        flag2 := true;
   while flag2 do no-op;                 while flag1 do no-op;
      section critique 1                    section critique 2
   flag1 := false ;                      flag2 := false ;
}                                     }
```

- Construire un automate pour chacun des processus, puis intersecter.
  - Automate de Mealy, transitions étiquetés par des valeurs de variables.
  - Aussi, boucles dans chaque état, provoquées par les modifications faites par l'autre processus sur ses variables.
  - Ajouter une étiquette indiquant quel processus a exécuté l'opération.
- Observer s'il y a exclusion mutuelle :
  - État correspondant aux deux processus en même temps en section critique !
- Observer s'il y a interblocage :
  - État à partir duquel aucun processus ne peut avancer.

L'intersection dans la modélisation des systèmes (2)

- Et maintenant la bonne solution de Dijsktra :

```
while (true) {
   flag1 := true;
   turn := 2;
   while (flag2 & turn = 2)
              do no-op;
     section critique 1
   flag1 := false ;
}
```

```
while (true) {
   flag2 := true;
   turn := 1;
   while (flag1 & turn = 1)
              do no-op;
     section critique 2
   flag2 := false ;
}
```

- Exclusion mutuelle assurée ?
- Et l'absence d'interblocage ?

Algorithmes, suite

- États inaccessibles :
  - Ne peuvent pas se trouver sur une trajectoire initialisée.
  - Donc on ne peut jamais les franchir à partir d'un état initial.
- Notion duale : états non-coaccessibles :
  - Ne peuvent pas se trouver sur une trajectoire qui s'arrête dans un état final.
- Élimination des états inaccessibles par parcours en largeur :
  - On part avec un ensemble S formé d'états initiaux, qui sont toujours accessibles (non ?).
  - À chaque itération, on rajoute à S tous les états qui sont franchissables de S en une transition.
  - On s'arrête lorsque le nouvel S est le même que celui de l'itération précédente.
- Pareil pour les co-accessibles.

Automates déterministes et complémentation

- Il nous reste une opération importante sur les langages : la complémentation.
- Problème : on nous donne un automate $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$ et on nous demande si $\Sigma^* \setminus L(\mathcal{A})$ est reconnaissable.
- Supposons que $\mathcal{A}$ est déterministe.
  - Un seul état initial, et une seule issue de chaque état avec une lettre donnée !
  - Tout mot est donc associé à *au maximum une trajectoire initialisée*.
  - On peut même faire en sorte que chaque mot soit associé à exactement une trajectoire initialisée!
  - On rajoute un état puits :

Automates déterministes et complémentation

- Bien-sûr, cet état puits n'est pas co-accessible, il serait normalement inutile ...
- ... mais il va bien nous servir pour la complémentation !
- δ dans un automate déterministe complet devient *fonction* :

$$\delta : Q \times \Sigma \longrightarrow Q$$

  (elle était fonction partielle dans un automate déterministe quelconque !)
- On peut même définir $\delta^* : Q \times \Sigma^* \longrightarrow Q$:
  - $\delta(q_0, w)$ donne l'état final de l'unique trajectoire associée à w dans $\mathcal{A}$.
- Complémentation : $\mathcal{A}^c = (Q, \Sigma, \delta, \{q_0\}, Q \setminus Q_f)$:
  - Puisque dans un automate déterministe *complet*, chaque mot est associé à une unique trajectoire initialisée,
  - ... un mot est accepté par $\mathcal{A}$ ssi $\delta(q_0, w) \in Q_f$,
  - ... donc un mot n'est pas accepté par $\mathcal{A}$ ssi $\delta(q_0, w) \notin Q_f$,
  - ... ce qui revient à dire $\delta(q_0, w) \in Q \setminus Q_f$!

Déterminisation des automates

- On prend un automate $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$.
- n voudrait construire un automate déterministe $\mathcal{B}$ ayant le même langage que (c.à.d. équivalent à) $\mathcal{A}$)
- On suit une idée similaire à l'algorithme de test du langage vide :
  - On construit les ensembles d'états (macro-états) franchissables avec le même mot.
  - Construction inductive, par induction sur la longueur des mots.
  - La construction s'arrête lorsqu'on trouve, pour tous les mots de longueur $n + 1$, des ensembles déjà construits pour les mots de longueur inférieure.
- Il faut aussi choisir les états finaux :
  - Ce sont les macro-états qui contiennent un état final !

Déterminisation des automates : formalisation

"Subset construction"

- Pour construire $\mathcal{B} = (R, \Sigma, \delta', r_0, R_f)$ déterministe :

$$R = 2^Q$$

Ce qu'on construit c'est des sous-ensembles d'états.

$$r_0 = Q_0$$
$$R_f = \{S \subseteq Q \mid S \cap Q_f \neq \emptyset\}$$

Car il suffit qu'une seule trajectoire franchisse Q_f

$$\delta' = \{S_1 \xrightarrow{a} S_2 \mid S_2 = \delta(S_1, a)\}$$

Ici, on a considéré que δ est une fonction :

$$\delta(S, a) = \{q \in Q \mid \exists s \in S, q \xrightarrow{a} s\}$$

- Construire $\delta'(S, a)$ revient à prendre tous les états franchissables par une a-transition qui part d'un état de S.
- Exemples :

Déterminisation : preuve

- On peut prouver que

$$\delta'(r_0, w) = \big\{q \in S \mid \exists \rho \text{ trajectoire initialisée associée avec } w \\ \text{et qui s'arrête en } q\big\}$$

- Preuve par induction sur la longueur de w.
- *Remarque :* $\mathcal{B}$ est complète :
  - Pour chaque $S \subseteq Q$ et $a \in \Sigma$, $\delta'(S, a)$ est bien défini,
  - ... même si parfois $\delta'(S, a) = \emptyset$.
  - $\emptyset$ est un état *puits* : $\delta(\emptyset, a) = \emptyset$ pour tout a.
  - $\emptyset \notin R_f$.