Cours A&G – Automates non-déterministes & déterminisation

1 Motivation :

- ⊕ Les automates non déterministes sont en général de taille plus petite, plus simple à décrire
- ⊖ mais ils nécessitent d'examiner plusieurs exécutions simultanément pour décider si un mot est accepté ou rejeté.

La reconnaissance d'un mot de longueur ℓ par un automate non déterministe (ANDEF) peut nécessiter ℓ exécutions en parallèle (donc ℓ machines).

- ⊕ Pour un mot donné, un automate déterministe n'a qu'une exécution possible et donc la reconnaissance peut se faire avec une seule machine
- ⊖ mais un automate déterministe (ADEF) peut avoir $2^{|\mathcal{Q}|}$ où $|\mathcal{Q}|$ est le nombre d'état de l'ANDEF équivalent

2 Déterminisation

Principe de déterminisation Simulation de l'exécution simultanée de plusieurs automates en parallèle. En appliquant ce principe à un ANDEF A , on obtient un ADEF A^D dont

- les états sont des sous-ensembles de $\mathcal{Q}_A$ (les états de l'automate A)
- l'état initial de A^D est $Init(A)$ (l'ensemble des états initiaux de A)
- les états accepteurs de A^D sont tous les sous-ensembles de $\mathcal{Q}_A$ qui contiennent au moins un état accepteur de A

Exemple voir ex. ??

3 Résultats importants

Pour chaque automate non-déterministe il existe un automate déterministe équivalent.

Étant donné un langage L (reconnaisable par un automate) il existe un unique automate déterministe minimal tel que $\mathcal{L}(A) = L$.

Autrement dit,

Si A et A' sont deux automates déterministes minimaux qui reconnaissent le même langage alors

- *ils sont identiques après renommage des états.*
- *ils ont la même relation de transition*

Application Pour prouver (par un algorithme) que deux langages $\mathcal{L}(A_1)$ et $\mathcal{L}(A_2)$ sont égaux, on construit les ADEF minimaux A_1^M et A_2^M et on vérifie qu'ils sont identiques (après un éventuel renommage des états).

Exemple voir ex. ??

Implantation de la déterminisation en OCAML Voir site web A&G → outils → `determinization.ml`