Langages formels et automates – cours 7

Langages sans étoile et minimisation d'automates déterministes

Catalin Dima

Expressions "sans étoile"

- Et si on utilisait seulement les opérations booléennes $+$ et $\bar{\cdot}$, et la concaténation ?
- **Expressions sans étoile** :

$$E ::= a \mid \emptyset \mid \varepsilon \mid E \cdot E \mid E + E \mid \overline{E}$$

- Un langage régulier est sans étoile s'il représente la sémantique d'une expression sans étoile.
  - C'est le cas avec $L_5 =$ tous les mots qui ne contiennent pas aab.

$$L_5 = \left|\overline{\overline{\emptyset} aab \overline{\emptyset}}\right|$$

  - À noter : le complément est pris par rapport à $\Sigma = \{a, b, c\}$!
  - Donc $\overline{\emptyset} = (a + b + c)^*$!
  - Il faut toujours indiquer par rapport à quoi on complémente.
- Peut-on décrire *de cette manière* tous les langages reconnaissables ?
- Le problème se pose pour représenter des langages périodiques comme $(aa)^*$, ou $(ababab)^*$...

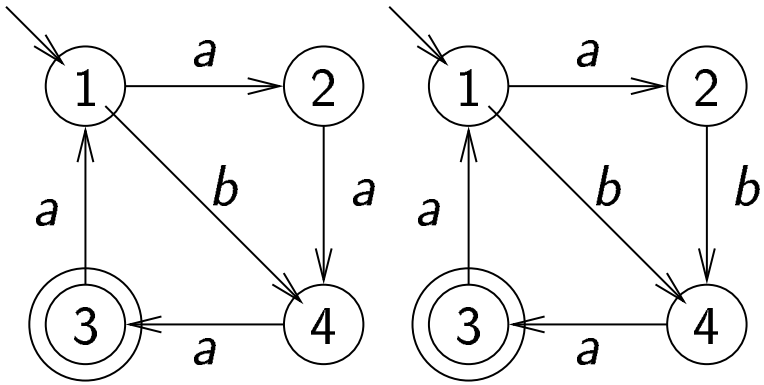
Automates "sans compteurs"

- Considérons un automate $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$.
- Pour tout $q \in Q$, notons L_{qq} le langage de l'automate $\mathcal{A}_{qq} = (Q, \Sigma, \delta, \{q\}, \{q\})$.
  - Donc L_{qq} = tous les mots étiquetant un circuit en q.
- On dit que $\mathcal{A}$ est **sans compteurs** si :

$$\forall w \in \Sigma^*, \forall q \in Q, \text{ si } w^n \in L_{qq} \text{ alors } w \in L_{qq}$$

- En français :
  - Tout circuit n'est pas étiqueté par un mot répété plusieurs fois.

Exemples d'automates sans et avec compteurs



- Comment on teste si un automate est sans compteur ?
  - Lister tous les circuits.
  - Vérifier si un de ces circuits est étiqueté par un mot répété w^n.

Langages apériodique

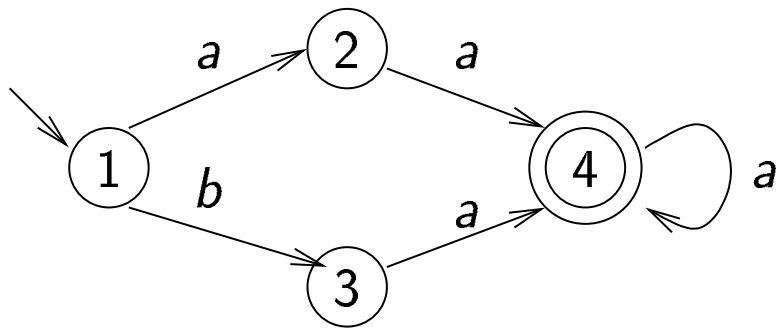
- Un langage régulier est apériodique s'il existe un automate sans compteur qui l'accepte.
- **Théorème de Schutzenberger** (variante) : La classe des langages apériodiques est la même que la classe des langages sans étoile.
- Ok, mais comment on vérifie dans la vraie vie qu'un langage L est sans étoile ?...
  - On essaye tous les automates qui acceptent L ?... il y en a combien ?...
- Un exemple pourquoi on devrait avoir un automate représentatif pour chaque langage régulier !

Automate déterministe minimal

- On cherche un plus petit automate déterministe équivalent à un automate donné.
- En fait, on cherche LE plus petit !
- Est-ce qu'il existe ?...
- C'est quoi "plus petit" ?...
  - La plus simple comparaison : nombre d'états.
- Donc, pour chaque automate déterministe $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_f)$, on cherche un autre automat déterministe $\mathcal{A}^\#$ qui :
  - A moins d'état que $\mathcal{A}$,
  - A le même langage que $\mathcal{A}$, $L(\mathcal{A}) = L(\mathcal{A}^\#)$,
  - Et est le plus petit parmi tous les automates déterministes qui ont le même langage !
- Et on veut même plus : avoir un algorithme qui construit $\mathcal{A}^\#$!

Équivalences d'états

- Pour tout $q \in Q$, notons L_q le langage de l'automate $\mathcal{A}_q = (Q, \Sigma, \delta, \{q\}, Q_f)$.
- Première intuition : dans $\mathcal{A}^\#$, on devrait avoir $L_q \neq L'_q$!
  - Si on avait $L_q = L'_q$, alors un des deux états q et q' est redondant !
- Un exemple...



  - $L_2 = L_3 = ...$
  - Donc on pourrait fusionner 2 et 3 – automate déterministe plus petit.
- Il faut convertir cette intuition en algorithme !

Équivalence d'états – suite

- On définit $q \equiv q'$ ssi $L_q = L'_q$.
- On cherche à construire $\equiv$, ce qui revient à dire qu'on cherche à définir les paires d'états $q \equiv q'$.
- On construit de manière inductive l'équivalence d'états :
  - Pour chaque n, on cherche à construire les paires d'états (q, q') pour lesquels $L_q \cap \Sigma^{\leq n} \neq L_{q'} \cap \Sigma^{\leq n}$.
  - On dénote les équivalences construites de cette manière $\equiv_n$.

1. Pour $n = 0$ c'est simple : $L_q \cap \Sigma^0$ ne peut être que vide ou ε ! !
2. Quand est $L_q \cap \Sigma^0$ vide et quand non ?
3. $q \equiv_0 q'$ si et seulement si $q, q' \in Q_f$ ou $q, q' \notin Q_f$ (en même temps !).

Équivalences d'états – induction

- Supposons qu'on a construit $\equiv_n$, il faut construire donc $\equiv n+1$.
- Toute paire (q, q') pour laquelle $q \not\equiv_n q'$ ne pourra donner lieu que à $q \not\equiv_{n+1} q'$, non ?
- Donc on regarde les paires $q \equiv_n q'$.
- **Règle :**

  Si $q \xrightarrow{a} r$ et $q' \xrightarrow{a} r'$, et on a $r \not\equiv_n r'$, alors on devrait avoir aussi $q \not\equiv_{n+1} q'$!

- Pourquoi ?
  - Parce que $r \not\equiv_n r'$ nous dit qu'il y a un mot $w \in L_r \cap \Sigma^{\leq n}$, qui n'est pas dans $L_{r'} \cap \Sigma^{\leq n}$.
  - Alors $aw \in L_q$ mais $aw \notin L_{q'}$!
- Propriété plus forte :

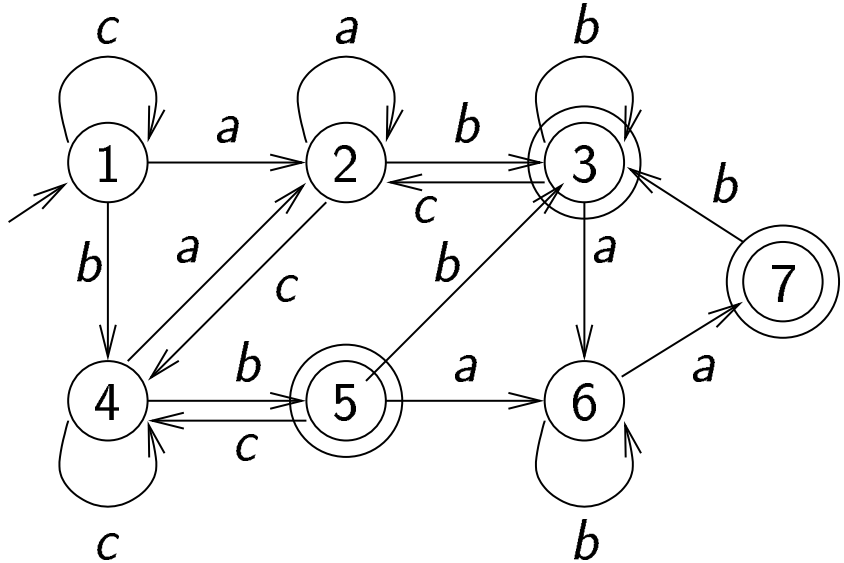
$$q \equiv_n q' \Leftrightarrow \left(L_q \cap \Sigma^{\leq n}\right) \cap \left(L_{q'} \cap \Sigma^{\leq n}\right) = \emptyset$$

Équivalence d'états – résultat

- À chaque pas, on a moins de paires dans $\equiv_{n+1}$ que dans $\equiv_n$.
- Elle ne peut pas décroître sans arrêt !
  - On aura, pour un N le même nombre de paires dans $\equiv_N$ et $\equiv_{N+1}$
  - La séquence se stabilise et $\equiv_N$ est $\equiv$!
- Un état ds. l'automate minimal = une classe d'équivalence ds. $\equiv_N$.
- Formellement $\mathcal{A}^{\#} = (Q/\equiv, \Sigma, \delta^{\#}, [q_0], Q_f/\equiv)$ où :

$$\delta^{\#} = \left\{ [q] \xrightarrow{a} [q'] \mid q \xrightarrow{a} q' \in \delta \right\}$$

- Exemple...



- Mais il faut prouver un certain nombre de choses...

Algorithme de Myhill-Nerode

- En pratique, on nous donne un automate déterministe $\mathcal{A}$.
- ... et on nous demande l'automate minimal $\mathcal{A}^{\#}$.
  1. On met les états en tête de lignes et de colonnes d'une matrice triangulaire.
  2. Au début, on marque (x) chaque case (q, q') avec $q \in Q_f, q' \notin Q_f$ ou vice-versa.
  3. On traverse la matrice, en cherchant si on a une case (q, q'), non marquée, et une lettre $a \in \Sigma$ tel que $q \xrightarrow{a} r$, $q' \xrightarrow{a} r'$ et (r, r') est déjà marqué.
  4. Si cela arrive, on marque la case (q, q').
  5. On s'arrête lorsque, après une traversée de la matrice, celle-ci reste non-modifiée.
- Ensuite on crée des macro-états rassemblant, pour un q donné, tous les q' pour lesquels (q, q') ou (q', q) n'est pas marqué.
- ... et on met les transitions comme décrit sur le transparent précédent.

Ce qui reste à prouver

- Est-ce que $\mathcal{A}^{\#}$ est un automate déterministe ?

$$q \xrightarrow{a} r, q \equiv q', q' \xrightarrow{a} r', \text{ alors est-ce que } r \equiv r' ?$$

- Supposons que non :
  - Alors on devrait avoir $L_r \neq L_{r'}$,
  - Donc un $w \in \Sigma^*$ tel que $\delta(r, w) \in Q_f$ mais $\delta(r', w) \notin Q_f$.
  - Mais alors on aura aussi $\delta(r, aw) \in Q_f$ mais $\delta(r', aw) \notin Q_f$.
  - Contradiction ! car on avait supposé $q \equiv q'$.
- Donc notre automate est bien déterministe !

Ce qui reste à prouver (2)

- Est-ce que $\mathcal{A}^{\#}$ accepte bien le même langage que $\mathcal{A}$?
- Prenons un $w \in L(\mathcal{A})$.
- Donc $\delta^{\#}(q_0, w) = q \in Q_f$.
- Alors $\delta^{\#}([q_0], w) = [q] \in Q_f / \equiv$!
- On a pas fini !
- Prenons aussi un $w \in L(\mathcal{A}^{\#})$.
- Donc $\delta^{\#}([q_0], w) = [q] \in Q_f / \equiv$.
- Alors, par induction, on peut prouver aussi qu'on peut trouver un chemin partant de q_0, s'arrêtant dans un $q' \in [q]$ et étiqueté par w.

Ce qui reste à prouver (3)

- Est-ce $\mathcal{A}^{\#}$ le plus petit automate ?
- Et si on commençait avec un $\mathcal{B}$ différent ?
- Oui, $\mathcal{A}'$ différent mais $L(\mathcal{B}) = L(\mathcal{A})$!
- Comment prouver qu'on obtient $\mathcal{A}^{\#} = \mathcal{B}^{\#}$?
- Enfin, modulo un rénommage des états !
- On doit raffiner l'idée des L_q.

Équivalences de mots définies par un langage

- Soit $L \subseteq \Sigma^*$ un langage.
- Définissons la relation $\sim_L \subseteq \Sigma^* \times \Sigma^*$, par :

$$w_1 \sim_L w_2 \text{ si } \forall w \in \Sigma^*, w_1 w \in L \Leftrightarrow w_2 w \in L$$

  - C'est une relation d'équivalence !
  - Preuve !
- Pour les langages réguliers, cette relation simule notre $\equiv$, mais *sans faire référence à un automate* !
- Si $\sim_L$ nous donne un nombre fini de classes, on peut même définir un automate :

$$\mathcal{A}_L = (\Sigma^*/\sim_L, \Sigma, \delta_L, [\varepsilon]_L, L/\sim_L) \text{ où}$$
$$\delta_L = \left\{[w] \xrightarrow{a} [wa]\right\}$$

- Donc L est régulier si et seulement si $\sim_L$ a un nombre fini de classes d'équivalence.
  - Encore une caractérisation des langages réguliers !
- Il faut prouver aussi que $L(\mathcal{A}_L) = L$!

Équivalence de mots définie par un automate

- Et maintenant prenons un automate $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_f)$ qui accepte L.
- Définissons $\sim_A \subseteq \Sigma^* \times \Sigma^*$ comme suit :

$$w_1 \sim_A w_2 \text{ si } \delta(q_0, w_1) = \delta(q_0, w_2)$$

- On peut prouver que $w_1 \sim_A w_2$ implique $w_1 \sim_L w_2$!
  - Car si $w_1 w \in L$, alors on doit avoir $\delta(q_0, w_1 w) \in Q_f$ et donc aussi $\delta(q_0, w_2 w) \in Q_f$.
- Donc toute classe d'équivalence de type $\sim_A$ est incluse dans une classe d'équivalence $\sim_L$.
  - Ce qui revient à dire que le nombre d'états dans $\mathcal{A}_L$ est minimal parmi tous les automates qui acceptent L !

Finir la preuve

- Il nous reste à prouver que $\mathcal{A}^{\#}$ n'est autre que $\mathcal{A}_L$! (modulo renommage des états).
- On veut d'abord prouver qu'on peut définir une bijection en associant à chaque classe $[q] \in Q/\equiv$ une classe unique $[w] \in \Sigma^*/\sim_L$.
- La bijection : on prend $w \in \Sigma^*$ tel que $\delta^{\#}([q_0], w) = [q]$, et on lui associe $[w]$!.
  - On va la noter ϕ_L.
- Il faut prouver qu'elle est bien définie !
- Il faut prouver que c'est une fonction injective !
- Il faut prouver que c'est une fonction surjective !
- Et, enfin, il faut aussi prouver que cette association fait que $\mathcal{A}^{\#}$ et $\mathcal{A}_L$ soient le même automate !
  - Donc que $\delta^{\#}([q_0], w) = [q]$ si et seulement si $\delta_L(\phi_L([q_0]), w) = \phi_L([q])$!
- On doit juste jouer un peu avec les notations !

Applications

- Test d'égalité de langages : étant donnés $\mathcal{A}_1$ et $\mathcal{A}_2$, est-ce que $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?
  - Il faut construire les deux automates minimaux $\mathcal{A}_1^\#$ et $\mathcal{A}_2^\#$ et prouver que c'est le même...
  - ... modulo un renommage des états !
- Test de langage sans étoile :
  - Propriété : L est sans étoile si et seulement si l'automate minimal pour L, $\mathcal{A}_L$, est sans étoile.
  - Donc on prend un automate quelconque, on le déterminise, on le minimise, et on regarde si on a des circuits étiquetés par une puissance d'un mot w^k.