Langages formels et automates – cours 5

Expressions régulières

Catalin Dima

Algorithmes de décision

- **Appartenence, langage vide, langage infini :** on sait les résoudre.
- **Inclusion de langages :** Étant donnés deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$, décider si $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$.
  - On se sert des opérations sur les langages : intersection et complémentation.

$$X \subseteq Y \Leftrightarrow X \cap \overline{Y} = \emptyset$$

  - Donc on construit l'automate $\mathcal{B}$ pour $\overline{L(\mathcal{A}_2)}$ et on teste si $L(\mathcal{A}_1) \cap L(\mathcal{B}) = \emptyset$.
  - Ce qu'on sait faire : construction de l'intersection et application de l'algo de décision pour langage vide.
- **Égalité de langages :** Étant donnés deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$, décider si $L(\mathcal{A}_1) = L(\mathcal{A}_2)$.
  - On utilise l'algorithme pour l'inclusion, car :

$$X = Y \Leftrightarrow X \subseteq Y \wedge X \supseteq Y$$

Expressions régulières – introduction

- On a défini toute une série d'opérations sur les langages : $\cup, \cdot, \cap, \bar{\cdot}, {}^*$....
- Est-ce qu'on peut définir tous les langages reconnaissables à l'aide de ces opérations ?
- Ça nous aiderait bcp. dans la construction d'automates – donc pour prouver qu'un langage est reconnaissable.
- Mais il faut définir ce qu'on entend par "définir tous les langages" !
  - Comment on construit les "définitions" ?
  - Quelles sont les briques de base ?
- Et il faut prouver qu'on n'obtient ni plus ni moins que $Rec(\Sigma)$!

Expressions régulières

- Expressions construites itérativement :
  1. À partir de lettres,
  2. En utilisant la concatenation $\cdot$, l'union $+$ et l'étoile *.
  3. Éventuellement utilisant aussi les paranthèses.
- On peut les décrire avec une "grammaire" :

$$E ::= a \mid \varepsilon \mid E + E \mid E \cdot E \mid E^* \mid (E)$$

où a est n'importe quelle lettre dans Σ.

  - On dit que a est une expression atomique.
- Exemples :

$$(a + b)^* \quad (a + bc^*)^* acc^* (bca)^*$$

- Notation pour l'ensemble de toutes les expressions régulières sur Σ :

$$Regexp(\Sigma)$$

Classe des langages **réguliers**

- Sémantique (ou langage) d'une expression régulière

$$|a| = \{a\} \qquad |\varepsilon| = \{\varepsilon\}$$
$$|E_1 + E_2| = |E_1| \cup |E_2| \qquad |E_1 \cdot E_2| = |E_1| \cdot |E_2|$$
$$|E^*| = |E|^*$$

- Donc $|(a + b)^*| = \{a, b\}^*$ = tous les mots sur a, b
- Et $|(a + bc^*)^* acc^* (bca)^*|$?...
- Classe des langages **réguliers**, notée

$$Reg(\Sigma)$$

  - Tous les langages qui représentent la sémantique d'une expression régulière.

Théorème de Kleene

- $Rec(\Sigma) = Reg(\Sigma)$
  - Il faut prouver les deux inclusions !
- Inclusion $\supseteq$: Partir d'une expression régulière, construire un automate.
- Mais ça, on sait faire !
- **Induction structurelle** :
  1. Automate pour a.
  2. Étant donnés automates $\mathcal{A}_1$ pour E_1 et $\mathcal{A}_2$ pour E_2, construire automate pour $E_1 + E_2$.
  3. Étant donnés automates $\mathcal{A}_1$ pour E_1 et $\mathcal{A}_2$ pour E_2, construire automate pour $E_1 \cdot E_2$.
  4. Étant donnés automates $\mathcal{A}$ pour E, construire automate pour E^*.
- Exemple...

Théorème de Kleene, inclusion $\subseteq$

- Malheureusement l'induction structurelle ne peut marcher !
  - Les automates ne sont pas définis à partir des petites briques en appliquant itérativement certaines opérations...
- Tout ce qu'on peut faire, c'est une induction **sur le nombre d'états** !
- Mais comment réduire le problème de construction d'une expression régulière pour un automate à n états, à une construction d'expression régulière pour un automate à $n-1$ états ?...
- Idée : fusion de transitions !

$$\text{si } q \xrightarrow{a} r \text{ et } r \xrightarrow{b} s, \text{ alors on met } q \xrightarrow{ab} s$$

- Sauf que, comme ça, on n'a plus d'automate fini comme il faut...
- Et qu'est-ce qu'on fait avec les boucles ?...

Automates étendus

- Automate étendu : $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$ où :
  - Q, Σ, Q_0, Q_f portent les mêmes noms et propriétés que pour le cas "classique".
  - $\delta \subseteq Q \times Regexp(\Sigma) \times Q$ est la relation de transition.
  - Mais on suppose que le nombre de transitions dans δ est fini !
- Notions de trajectoire, trajectoire initialisée, trajectoire acceptante – les mêmes.
- Expression régulière associée à une trajectoire acceptante = concaténation des expressions régulières sur les transitions de la trajectoire.
- Langage accepté = union des sémantiques de toutes les expressions régulières associées à toutes les trajectoire acceptantes.
- Exemple ...

Elimination d'un état dans un automate étendu

- On nous donne $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$ et un état $q \in Q \setminus (Q_0 \cup Q_f)$.
- Et on nous demande $\mathcal{A}' = (Q', \Sigma, \delta', Q_0, Q_f)$
  - Avec $Q' = Q \setminus \{q\}$.
  - Et ayant le même langage : $L(\mathcal{A}) = L(\mathcal{A}')$.
- Il faut fusionner les transitions entrantes en q avec les transitions sortantes de q.
- Mais si q possède des boucles, il faut les insérer dans les fusions !
- Exemple...

Elimination d'un état dans un automate étendu

Formalisation :

- Supposons que

$$A = \left\{ E \in Regexp(\Sigma) \mid q \xrightarrow{E} q \in \delta \right\}$$

- Alors pour toute paire de transitions $r \xrightarrow{E_1} q, q \xrightarrow{E_2} s \in \delta$, avec $r \neq q \neq s$, on met dans δ'

$$r \xrightarrow{E_1 \cdot \left(\sum_{E \in A} E \right)^* \cdot E_2} s$$

- Et on copie en δ' toute transition qui n'implique pas q.

Théorème de Kleene, inclusion $\subseteq$

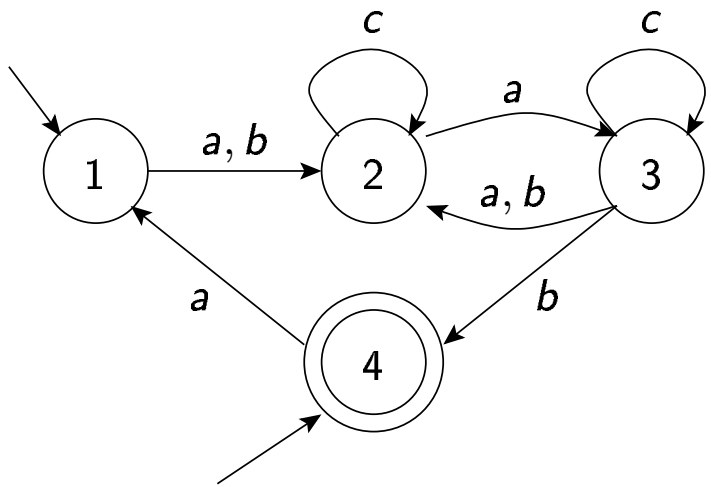
- Et le thèoreme de Kleene est presque prouvé !
- On peut éliminer tous les états, ou presque...
- Car on ne sait éliminer que les états non-initiaux, non-finaux...
  - Alors on rajoute une copie de chaque état initial, et une copie de chaque état final.
  - Les copies des états initiaux copient aussi toutes les transitions sortantes des anciens états initiaux.
  - Et les copies des états finaux copient aussi toutes les transitions entrantes des anciens états finaux.
  - Donc on pourra éliminer tranquillement aussi les anciens états initiaux, non ?
- À la fin, on restera avec un automate qui aura que des transitions

$$q \xrightarrow{E} r, q \in Q_0, r \in Q_f, Q_0 \cap Q_f = \emptyset$$

- Et ça, c'est facile :

$$L(\mathcal{A}) = \left| \sum_{q \xrightarrow{E} r \in \delta} E \right|$$

Exemple de construction itérative de l'expression régulière



Une technique plus "algébrique" : équations de langages

- Considérons l'équation "linéaire" :

$$X = a \cdot X + b$$

  - Est-ce que cette équation a une solution ? laquelle ?
  - Et celles-ci :

$$X = (a + b)X + ac$$
$$X = X + b$$
$$X = aX$$

- Cas général : si $\varepsilon \notin A$, alors l'équation

$$X = A \cdot X + B$$

  a une seule solution,

$$X^{sol} = A^* B$$

- Preuve ! (double inclusion !)
- Et si $\varepsilon \in A$?...

Systèmes d'équations sur les langages

- On sait résoudre des équations.
- Et si on se compliquait la vie :

$$\begin{cases} X_1 &= abX_1 + (b+c)X_2 \\ X_2 &= (a+b^*c)X_1 + caX_2 + aa^* \end{cases}$$

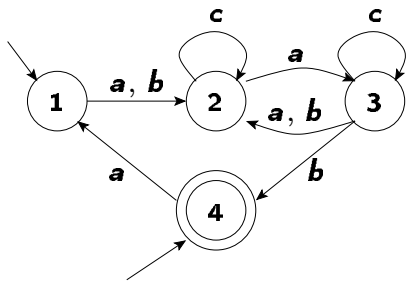
- Résoudre ce système !
  - Méthode itérative de substitution de Gauss :
    - On résout la dernière équation en X_2, comme si X_1 était un ensemble :

$$X_2 = ca^* \cdot \left((a+b^*c)X_1 + aa^*\right) \qquad (1)$$

    - Heureusement, si on applique la distributivité (ouf ! on l'a prouvée !) on se retrouve avec une équation toujours linéaire en X_1.
    - Et ça, on sait faire !
    - Et enfin, on remplace le X_1 trouvé dans (1).
  - Généralisation facile (non ?).

Systèmes d'équations et automates

- On va construire alors un petit système d'équations pour chaque automate donné :
  - Une variable X_q par état.
  - Une équation par variable X_q, qui code les transitions sortantes de q.
  - Exemple :



$$\begin{cases} X_1 &= (a+b)X_2 \\ X_2 &= cX_2 + aX_3 \\ X_3 &= (a+b)X_2 + cX_3 + bX_4 \\ X_4 &= aX_1 + \varepsilon \end{cases}$$

car l'état 4 est final !

- Résoudre ce système, et comparer le résultat !

Systèmes d'équations et automates

- Formalisation : on construit un système d'équations dans lequel l'équation concernant X_q est comme suit :

$$X_q = \sum_{r \in Q} \left(\sum_{q \xrightarrow{a} r} aX_r \right) + B_q$$

où B est soit ε soit $\emptyset$, en fonction du fait que q soit final ou non :

$$B_q = \begin{cases} \varepsilon & \text{si } q \in Q_f \\ \emptyset & \text{sinon} \end{cases}$$

- Une fois résolu le système, l'expression équivalente avec l'automate est

$$\sum_{q \in Q_0} X_q^{sol}$$