Langages formels et automates – cours 8

Langages réguliers et leurs limites

Catalin Dima

Rappels

- Langages réguliers = langages représentant la sémantique d'une expression régulière.
- Langages reconnaissables = lanages des automates finis.
- Théorème de Kleene = les deux classes sont identiques.
- Langages sans étoile = sous-classe stricte des langages réguliers.
  - Exemple de langage régulier qui n'est pas sans étoile :

$$L = \{a^{2n} \mid n \in \mathbb{N}\}$$

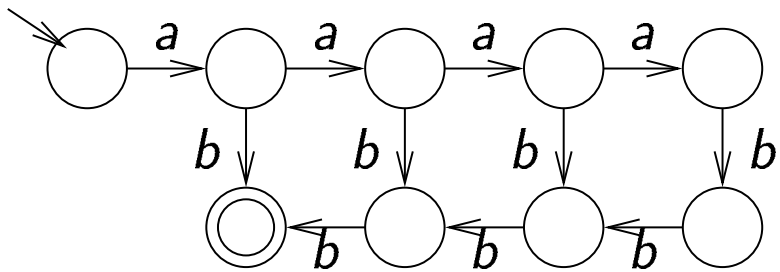
- Est-ce que tout langage est régulier ?...

Un possible langage non-régulier

- Un possible non-exemple :

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

- Automate déterministe pour $L_4 = \{a^n b^n \mid 0 \leq n \leq 4\}$.



- Faire de même pour L_5, L_6, etc.
- Intuitivement, pour L on aurait besoin d'un escalier "infini".
- Mais comment prouver formellement que L n'est pas régulier ?

Une propriété intéressante des langages réguliers

- Supposons L régulier et infini.
- Prenons un automate sans ε-transitions qui accepte L (par exemple, l'automate minimal).
- On doit avoir donc une composante fortement connexe (CFC) non-triviale dans Q,
- ... dont tous les états sont accéssibles et co-accéssibles.
- Prenons ensuite une trajectoire acceptante qui passe par un état q_1 de cette CFC :

$$q_0 \rightsquigarrow q_1 \rightsquigarrow q_2, q_2 \in Q_f$$

- Puisque q_1 fait partie d'une CFC non-triviale, il existe un circuit $q_1 \rightsquigarrow q_1$ de longueur ≥ 1.
- Alors on aura aussi les trajectoires acceptantes suivantes :

$$q_0 \rightsquigarrow q_1 \rightsquigarrow q_1 \rightsquigarrow q_2$$

$$q_0 \rightsquigarrow q_1 \rightsquigarrow q_1 \rightsquigarrow q_1 \rightsquigarrow q_2$$

$$\ldots\ldots$$

$$q_0 \rightsquigarrow \underbrace{q_1 \rightsquigarrow q_1 \ldots \rightsquigarrow q_1}_{n \text{ fois}} \rightsquigarrow q_2$$

Une propriété intéressante des langages réguliers

- Reprenons les trajectoires, cette fois-ci avec leurs étiquettes :

$$q_0 \overset{x}{\rightsquigarrow} q_1 \overset{w}{\rightsquigarrow} q_1 \overset{y}{\rightsquigarrow} q_2$$

$$q_0 \overset{x}{\rightsquigarrow} q_1 \overset{w}{\rightsquigarrow} q_1 \overset{w}{\rightsquigarrow} q_1 \overset{y}{\rightsquigarrow} q_2$$

$$\cdots\cdots$$

$$q_0 \overset{x}{\rightsquigarrow} \underbrace{q_1 \overset{w}{\rightsquigarrow} q_1 \ldots \overset{w}{\rightsquigarrow} q_1}_{n \text{ fois}} \overset{y}{\rightsquigarrow} q_2$$

- Donc on a $xwy \in L$,
- ... et aussi $xw^2y \in L$, et $xw^3y \in L$, et $xw^ny \in L$ pour tout $n \in \mathbb{N}$.
- On dit que L contient aussi le rayon xw^ny.
- Observer que, dans un rayon, un seul infix peut être gonflé autant qu'on veuille !
- Cette propriété pourrait nous servir pour a^nb^n, car ce langage ne respecte pas, intuitivement, cette propriété de rayon :
  - Dans a^nb^n, il y a deux infix qui sont gonflés simultanément !

Lemme de l'étoile

Énoncé :

- Si L est régulier, alors il existe un entier $N \in \mathbb{N}$ tel que tout mot de L ($z \in L$) contenant plus de N lettres peut s'écrire $z = xwy$, avec les propriétés suivantes :
  1. $w \neq \varepsilon$.
  2. Le rayon $xw^n y$ est contenu dans L : pour tout $n \in \mathbb{N}$, $xw^n y \in L$.

Preuve :

- Prenons l'automate déterministe minimal pour L.
- Et fixons $N = card(Q)$.
- Prenons ensuite un mot ayant plus de N lettres, $z \in L$ – soit $n = |z|$.
- Il doit être accepté sur une trajectoire ayant $n + 1$ états.
- Mais un des états doit se répeter dans la trajectoire !

Preuve du lemme de l'étoile – suite

- Écrivons la trajectoire acceptant z, en prenant soin de mettre en évidence l'état qui se repète :

$$q_0 \overset{x}{\rightsquigarrow} q_1 \overset{w}{\rightsquigarrow} q_1 \overset{y}{\rightsquigarrow} q_2$$

- On a trouvé notre décomposition de z !
  - Il faut observer que $w \neq \varepsilon$, car on a choisi deux apparitions distinctes de q_1 dans la trajectoire !
  - Selon le raisonnement connu, on devrait avoir aussi $xw^n y \in L$.
  - On peut aussi choisir q_1 comme le premier état qui se repète.
  - Alors on devrait avoir aussi la propriété suivante :

$$|xw| \leq N$$

Comment on se sert du lemme de l'étoile

- On s'en sert pour prouver qu'un langage L n'est pas régulier !
- Preuve par réduction à l'absurde :
  1. On suppose que L est régulier.
  2. On prouve alors que le lemme de l'étoile nous amène à une contradiction.
- Comment trouver une contradiction :
  - On prend un mot $z \in L$ et on le décompose de toutes les manières possibles en $z = xwy$.
  - Pour chaque décomposition, on devrait prouver que le rayon $xw^n y$ ne peut pas être inclus dans L.

Premier exemple : $a^n b^n$

- Prenons toute décomposition d'un mot $a^n b^n$ en xwy.
- Trois cas possibles :
  1. w ne contient que des a.
  2. w ne contient que des b.
  3. w contient des a et des b.
- Dans chaque cas, il faut prouver qu'il existe des membres du rayon $xw^n y$ qui ne sont pas dans L.
- Assez souvent, on prouve que $xwwy = xw^2 y$ n'est pas dans le langage !
- Et parfois on prouve aussi que $xy = xw^0 y$ n'est pas dans le langage !
  - Se rappeler que le rayon est défini comme $xw^n y$ pour tout $n \in \mathbb{N}$!

Deux décompositions de $a^n b^n$

- Dans ce cas on a $k + l \leq n$ tel que

$$x = a^k \qquad w = a^l \qquad y = a^{n-k-l} b^n$$

- Et bien-sûr, $l \geq 1$!
- Mais donc qu'en est-il de xw^2y ?

$$xw^2y = a^k a^{2l} a^{n-k-l} b^n = a^{n+l} b^n \notin L$$

car $n + l \neq n$!

- La même preuve pour la décomposition dans laquelle w n'a que des b !

La 3e décomposition de $a^n b^n$

- Et si w contient des a et des b :

$$x = a^{n-k} \qquad w = a^k b^l \qquad z = b^{n-l}$$

- Et $k + l \neq 0$!
- Alors le 2e élément du rayon est :

$$xw^2y = a^{n-k} a^k b^l a^k b^l b^{n-l}$$

- Deux situations :
  1. $l = 0$, alors on doit avoir $k \geq 1$ et donc
  $$xw^2y = a^{n+k} b^n \notin L$$
  2. $k, l \neq 0$, alors on a un mot qui mélange les a et les b,
  $$xw^2y = a^n b^l a^k b^n \notin L$$

Conclusion pour $a^n b^n$

- On a essayé toutes les décompositions pour des mots $z \in L$ en $z = xwy$.
- Dans tous les cas, le rayon $xw^n y$ n'est pas inclus dans L.
- Donc notre hypothèse que $a^n b^n$ est régulier, est fausse !
- C'est donc un langage non-régulier !

Encore un exemple de langage non-réguliers

- $L = \{w \mid w \in \Sigma^* \#_a(w) = \#_b(w)\}$, nombre de a égal au nombre de b.
- Toujours le même principe de réduction à l'absurde : on suppose que L est régulier.
- Alors pour tout autre langage régulier R, $L \cap R$ devrait aussi être régulier !
- En particulier, $R = \{a^n b^m \mid m, n \in \mathbb{N}\}$.
- Tout le mode sait prouver que R est régulier ?...
- Alors

$$L \cap R = \{a^n b^n \mid n \in \mathbb{N}\}$$

- Et on sait déjà que $L \cap R$ n'est pas régulier !
- **Morale :** Intersecter avec des langages réguliers peut simplifier la preuve de non-régularité.

Au delà des langages réguliers

- Donc le monde n'est pas régulier dans sa totalité...
- Et $a^n b^n$ n'est pas un langage sans importance !
  - On peut imaginer que a = parenthèse ouvrante et b = parenthèse fermante.
  - Donc $a^n b^n$ apparaît tout à fait naturellement dans les langages de programmation !
- A-t-on d'autres manières de définir des langages ?
  - Il nous les faut, car on voudrait être capables de vérifier quand un programme a le même nombre de parenthèses fermées que les parenthèses ouvertes !
  - C'est la moindre des choses que l'analyse syntaxique d'un programme doit vérifier !
  - Et c'est aussi important pour la traduction en code machine !

Une première idée

- Prenons les équations de langages :

$$X = a \cdot X + b \quad \text{avec solution} \quad X = a^* b$$

- On peut imaginer des équations linéaires à droite :

$$X = X \cdot a + b \quad \text{avec solution} \quad X = ba^*$$

- Et si on permettait des “constantes” à gauche ou à droite ?

$$X = aXb + c$$

- Quelle serait la solution ?...
- On cherche le langage L qui satisfait la propriété

$$L = a \cdot L \cdot b \cup \{c\}$$

Une première idée

- Solution pour $X = aXb + c$:

$$L = \left\{ a^n c b^n \mid n \in \mathbb{N} \right\}$$

- Et si on voulait $a^n b^n$?

$$X = aXb + \varepsilon$$

- Forcément, ce qu'on obtient ne sont plus des langages réguliers !
- On pourrait aussi chercher des généralisations des automates finis pour accepter de tels langages.
- Idée : permettre un ensemble infini d'états !
- ... mais bien-sûr, ensemble qui soit généré de manière finitaire !