CH.3 Propriétés des langages réguliers

- 3.1 Le lemme de pompage
- 3.2 Les propriétés de fermeture
- 3.3 Les problèmes de décidabilité

3.1 Le lemme de pompage

Théorème (lemme de pompage) :
Soit L un langage régulier reconnu par un automate à n états.
Soit z un mot de L de longueur $\geq$ n. Alors z se factorise en
z = uvw, où $|uv| \leq n$, $v \neq \varepsilon$ et, pour tout $i \geq 0$, on a
$uv^iw \in L$.

Démonstration :
Si $z = a_1a_2\ldots a_m$ est accepté par un automate à n états et que
$m \geq n$, le chemin correspondant à z dans l'automate passe deux fois par un même état (principe des tiroirs). Soit q le premier état qui est revisité. On a alors z = uvw, où $\delta(q_0, u) = q$, et
$\delta(q, v) = q$. On a $v \neq \varepsilon$ et le chemin correspondant à uv visite l'état q deux fois et lui seul. Donc $|uv| \leq n$. Enfin, $\delta(q_0, uv^i) = q$ donc $\delta(q_0, uv^iw) = \delta(q_0, z)$ donc $uv^iw \in L$.

Application : montrer que certains langages ne sont pas rationnels.

Exemples :

$L = \{0^n 1^n : n \geq 0\}$ n'est pas régulier.
Si L est reconnu par un automate à n états, la factorisation de $0^n 1^n$ conduit à une contradiction.

L = {écritures binaires de p, où p est premier} n'est pas régulier.
Si L est reconnu par un automate à n états, soit $p > 2^n$, p premier.
Soit z = uvw son écriture binaire et soient n_u, n_v et n_w les nombres d'écriture u, v et w respectivement. On sait alors que, pour tout i, uv^iw est l'écriture d'un nombre premier. Choisissons i = p. La valeur numérique de uv^pw est :

$q = n_u\, 2^{|w| + p|v|} + n_v\, 2^{|w|}(1 + 2^{|v|} + \cdots + 2^{(p-1)|v|}) + n_w.$
Puisque $1 \leq |v| \leq n$, alors $2 \leq 2^{|v|} \leq 2^n < p$. Donc $2^{|v|} \neq 1 \bmod p$.
Dans l'expression de q, la parenthèse vaut $(2^{p|v|} - 1) / (2^{|v|} - 1)$.
(Série géométrique). Mais $2^{p-1} \equiv 1 \bmod p$ (Petit théorème de Fermat).
Donc la parenthèse dans q vaut 1 modulo p. On a donc que, modulo p,
$q \equiv n_u\, 2^{|w| + |v|} + n_v\, 2^{|w|} + n_w = p.$
Donc p divise q, ce qui est une contradiction.

Le lemme de pompage est insuffisant pour montrer que d'autres langages ne sont pas réguliers (par exemple les mots ayant un nombre différent de a et de b).

3.2 Les propriétés de fermeture

On peut utiliser les définitions équivalentes des langages réguliers, par expressions régulières ou par automates.

Théorème : Les langages réguliers sont fermés par les opérations régulières.

Démonstration : évident avec les expressions régulières.

Théorème : Les langages réguliers sont fermés par complémentation.

Démonstration : Soit M un AFD complet qui reconnaît L. Soit M' l'AFD obtenu en échangeant les états terminaux et non-terminaux de M. Alors M' reconnaît le complémentaire L' de L.

Théorème : Les langages réguliers sont fermés par intersection.

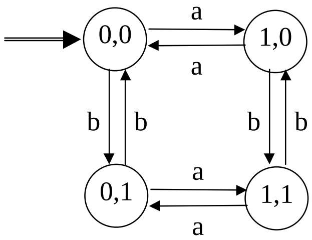
Démonstration : utiliser la fermeture par complémentaire et les relations de de Morgan.

En pratique, si L_1 et L_2 sont reconnus par des AFD M_1 et M_2, on peut construire directement un AFD reconnaissant $L_1 \cup L_2$ et un AFD reconnaissant $L_1 \cap L_2$ par produit cartésien des automates. L'ensemble des états terminaux est alors l'ensemble des couples d'états dont l'un est terminal pour l'union, et l'ensemble des couples d'états terminaux pour l'intersection.

Exemple :

Soit L_1 = mots ayant un nombre impair de a.
Soit L_2 = mots ayant un nombre impair de b.

Produit cartésien



Pour $L_1 \cup L_2$, prendre $F = \{(0, 1), (1, 0), (1, 1)\}$;
Pour $L_1 \cap L_2$, prendre $F = \{(1, 1)\}$.

Exemple :
Le langage L constitué de mots ayant un nombre différent de a et de b n'est pas régulier.
S'il l'était, il en serait de même de son complémentaire, mots ayant autant de a et de b. Ce dernier n'est pas régulier, car on peut "pomper" sur le mot $a^n b^n$.

Soit L un langage représenté par une expression régulière r. Soit f une application qui associe à chaque lettre a une expression régulière f(a). La substitution de f dans L est le langage f(L) représenté par l'expression régulière f(r). Cette définition ne dépend que du langage L et des langages représentés par les f(a), et non des expressions régulières.

Théorème : Les langages réguliers sont fermés par substitution.

Démonstration : évident par la définition.

Cette opération est utilisée pour créer des langages réguliers dont les lettres seront ensuite remplacées par des mots appartenant à des langages spécifiés à partir de ces lettres (variables).

Cas particulier de substitution : homomorphisme.
Soit L un langage sur un alphabet {a, b, ... }. On considère pour chaque lettre a un mot h(a) sur un autre alphabet (ou le même). L'ensemble des mots de la forme h(w) constitue le langage h(L). Comme c'est une substitution particulière, on a :

Théorème : Si L est régulier et si h est un homomorphisme, alors h(L) est régulier.

Homomorphisme inverse :
Si h est un homomorphisme, $h^{-1}(L) = \{x : h(x) \in L\}$.

Exemple :

Soit L représenté par (ab + ba)*a et h(0) = aa, h(1) = aba.

On vérifie directement que $h^{-1}(L) = \{1\}$.

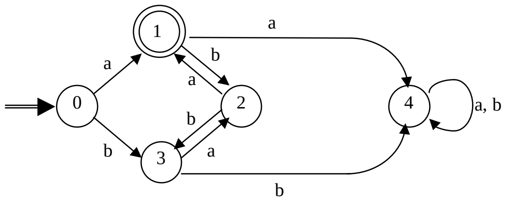
Homomorphisme et homomorphisme inverse sont les outils essentiels du codage. Le théorème précédent et le suivant montrent que le codage et le décodage de langages réguliers sont encore réguliers.

Théorème : Si L est régulier et si h est un homomorphisme, alors $h^{-1}(L)$ est régulier.

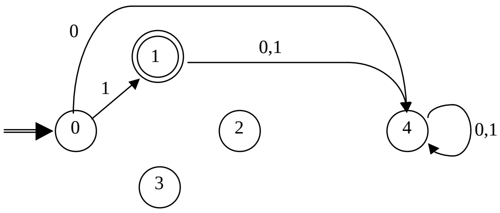
Démonstration : Prendre un automate M reconnaissant L et effectuer sur M les transitions correspondant à h. On obtient ainsi un automate reconnaissant $h^{-1}(L)$.

Exemple :
Soit L représenté par $(ab + ba)^*a$ et $h(0) = aa$, $h(1) = aba$.

M



Pour $h^{-1}(L)$:



On peut utiliser ces théorèmes pour montrer que certains langages ne sont pas réguliers.

Exemple :
Soit L l'ensemble des mots ayant deux fois plus de a que de b.
Si L était régulier, il en serait de même de $L \cap a^*b^* = \{a^{2n}b^n : n \geq 0\}$.
Soit $h(0) = aa$ et $h(1) = b$. Alors $h^{-1}(L \cap a^*b^*) = \{0^n1^n : n \geq 0\}$ serait aussi régulier.

Quotients :
L_1 et L_2 sont deux langages. On pose :
$L_1 / L_2 = \{x : \exists y \in L_2 , xy \in L_1\}$ et
$L_2 \backslash L_1 = \{x : \exists y \in L_2 , yx \in L_1\}$.
Ce sont les quotients à droite et à gauche de L_1 par L_2.

Théorème : Si L_1 est régulier et L_2 quelconque, alors les deux quotients de L_1 par L_2 sont réguliers.

Démonstration : Soit M un AFD qui reconnaît L_1.
Pour le quotient à droite, il suffit de trouver les états de M qui sont reliés à un état terminal par un mot de L_2 et de les prendre comme nouveaux états terminaux. Pour le quotient à gauche, on prend comme nouveaux états initiaux les états auxquels on arrive depuis l'état initial de M par un mot de L_2. Comme on n'a droit qu'à un seul état initial, on en ajoute un, relié aux états précédemment trouvés par une transition vide.

Cette démonstration n'indique pas comment trouver ces états (cela dépend de L_2). Elle n'est donc pas constructive. Si L_2 est régulier, on peut la rendre constructive.

Les langages réguliers sont également fermés par les opérations suivantes : facteurs, sous-mots, permutations circulaires, image-miroir, conjugaison, produit d'intercalement, ...

3.3 Les problèmes de décidabilité

Théorème : Soit M un automate à n états. Le langage accepté par M est :

1.— non vide $\Leftrightarrow$ M accepte un mot de longueur $< n$;

2.— infini $\Leftrightarrow$ M accepte un mot de longueur l, $n \leq l < 2n$.

Démonstration :

1.— C'est une propriété générale des chemins dans les graphes ;

2.— Si x est un mot accepté de longueur $\geq n$, le lemme de pompage montre que le langage accepté est infini.

Réciproquement, si le langage accepté est infini, il existe des mots de longueur $\geq n$. Soit x un tel mot de longueur minimum l. Par le lemme de pompage, $x = uvw$ et uw est aussi reconnu. On a donc $|uw| < n$ (l minimum) ; comme $|v| \leq |uv| \leq n$ (lemme de pompage), on a $l < 2n$.

En fait, pour montrer effectivement que le langage accepté est non-vide, il est plus judicieux de tester s'il existe des états terminaux accessibles. De même, le langage accepté est infini si et seulement si il existe des circuits greffés sur des chemins allant de l'état initial à un état terminal. Pour cela, on peut ne garder que les transitions utiles, en remontant depuis les états terminaux, et tester si le graphe ainsi obtenu possède des circuits (par exploration en profondeur).

Problèmes décidables.

Un problème relatif à un ensemble d'objets {x} est une question telle que "la propriété P(x) est-elle vraie ?". Un problème est décidable s'il existe un algorithme prenant x (ou plutôt un codage de x) en entrée et s'arrêtant au bout d'un temps fini avec une réponse par oui ou par non.

Les différents codages possibles pour les éléments x sont sans importance tant qu'on peut passer de l'un à l'autre pa un algorithme effectif :

codage de x → algorithme → OUI, P(x) est vrai
→ NON, P(x) est faux

Théorème : Les problèmes suivants sont décidables :

1.— L'automate M reconnaît un langage non-vide ;
2.— L'automate M reconnaît un langage infini ;
3.— Le langage reconnu par M est inclus dans celui reconnu par N ;
4.— Les automates M et N reconnaissent le même langage.

Démonstration : Les deux derniers points peuvent se faire par produit des automates avec un ensemble terminal approprié et test su un certain langage comme étant vide.