Notes de cours Option Info MP/MP* : Motifs, langages et automates

Motifs

Un motif est un ensemble dont les éléments ont une propriété qui est reconnaissable. Par exemple:

- 0, 2, 4, 6, 8,... l'ensemble des entiers naturels pairs
- `let`, `for`, `while`, `match`,... sont des mots du langage Caml
- Les codes barres, les QR codes...

La reconnaissance de motifs a de nombreuses applications, notamment en informatique. Lors de la compilation (traduction des instructions écrites dans un langage de programmation en langage machine), la première phase est l'analyse lexicale, qui consiste à identifier les séquences de caractères qui forment les mots-clés du langage, les noms de variables, etc.. Vient ensuite l'analyse syntaxique, c'est-à-dire vérifier que les instructions sont syntaxiquement correctes (expressions bien parenthésées, `;;` à la fin, `done` après un `do`...).

On peut aussi citer les applications en bio-informatique: chercher un gène dans un brin d'ADN (codé par une succession de A, C, T, G), détecter des ressemblances entre plusieurs brins d'ADN pour établir une parentalité...

Nous nous contenterons de chercher des motifs en dimension 1, c'est-à-dire dans des chaînes de caractères.

Pour cela nous aurons plusieurs objectifs:

1. Avoir un moyen simple de définir un motif.
2. Avoir un moyen efficace de reconnaître un motif.

Cela nous mènera aux définitions de langage et d'expression rationnelle.

Par la suite, on construira une machine appelée automate qui reconnaîtra si un mot vérifie un certain motif en temps linéaire en la longueur du mot (une seule lecture du mot suffit).

Exemple 1 : les nombres entiers

Décrivons d'abord les chaînes de caractères représentant les entiers (à compléter): ...

Exemple 2 : nombres écrits en binaire divisibles par 3 écrits en binaire

Comment déterminer si un nombre écrit en binaire est divisible par 3 ? (à compléter) ...

Exemple 3 : mots dont les voyelles apparaissent la 1ère fois dans l'ordre alphabétique

Par exemple, le mot bateau vérifie ce motif, le mot binaire ne le vérifie pas. On a un algorithme en temps linéaire qui permet de vérifier le motif (à compléter) : ...

Mots et langages

Dans cette partie A désigne un alphabet, c'est-à-dire un ensemble fini. Ses éléments sont appelés lettres.

Définitions:

- Un mot sur l'alphabet A est une liste finie d'éléments de A, le mot vide se note ϵ. La longueur d'un mot m est notée $|m|$, et si $a \in A$, $|m|_a$ est le nombre d'occurrences de la lettre a dans le mot m.
- On note A^* l'ensemble des mots sur l'alphabet A. Cet ensemble est muni d'une loi de composition interne, la concaténation . , qui est asssociative et admet le mot vide pour élément neutre. On dit que A^* muni de cette loi est un monoïde.
- Étant donnés deux mots u et v, on dit que:
  - u est un préfixe de v s'il existe un mot t tel que $v = u.t$.
  - u est un suffixe de v s'il existe un mot t tel que $v = t.u$.
  - u est un facteur de v s'il existe deux mots x et y tels quel $v = x.u.y$.
- Étant donné un mot $u = a_1 \dots a_n$ (les a_i étant ses lettres), un sous-mot de u est un mot de la forme $a_{\varphi(1)} \dots a_{\varphi(p)}$ où $p \leqslant n$ est $\varphi : [[1, p]] \to [[1, n]]$ est strictement croissante. Par exemple, "ami" est un sous-mot de "amabilité" mais ce n'est pas un facteur.
- Un langage est un sous-ensemble de A^*.

Opérations sur les langages

Étant données deux langages L_1 et L_2, on peut définir:

- Leur réunion $L_1 \cup L_2$ que l'on note aussi $L_1 + L_2$ ou $L_1|L_2$.
- Leur intersection $L_1 \cap L_2$.
- Leur concaténation $L_1.L_2 = \{u.v, u \in L_1, v \in L_2\}$.

La concaténation étant associative, on peut définir par récurrence sur n le langage L^n en posant $L^0 = \{\epsilon\}$ et $L^{n+1} = L.L^n = L^n.L$ (attention à ne pas confondre $L^2 = \{u.v, u \in L, v \in L\}$ avec l'ensemble plus petit des carrés des mots de L qui est $\{u^2, u \in L\}$). On définit enfin la fermeture de Kleene (ou l'étoile) L^* du langage L par:

$$L^* = \bigcup_{n \in \mathbb{N}} L^n.$$

On note aussi $L^+ = \bigcup_{n \in \mathbb{N}^*} L^n$.

Expressions rationnelles et langages rationnels

Définition: Les expressions rationnelles sur un alphabet A sont définies inductivement par:

- $\emptyset$ et ϵ sont des expressions rationnelles.
- Pour tout $a \in A$, a est une expression rationnelle.
- Pour toutes expressions rationnelles e_1 et e_2, $e_1 + e_2$ et $e_1.e_2$ sont des expressions rationnelles.

- Pour toute expression rationnelle e, e^* est une expression rationnelle.

Remarque: On peut représenter une expression rationnelle par un arbre.

Définition: On définit de manière inductive le langage $L(e)$ associé à une expression rationnelle e:

- $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$, $L(a) = \{a\}$ pour tout $a \in A$.
- $L(e_1 + e_2) = L(e_1) + L(e_2)$.
- $L(e_1.e_2) = L(e_1).L(e_2)$.
- $L(e^*) = L(e)^*$.

Un langage rationnel est un langage défini par une expression rationnelle.

Remarque: On peut aussi définir les expressions rationnelles étendues en ajoutant l'intersection et la différence ensembliste.

Exemples: Sur l'alphabet $A = \{a, b\}$:

- a^*b^* est une expression rationnelle dont le langage associé est (à compléter) ...
- $(ab)^*$ est une expression rationnelle dont le langage associé est (à compléter) ...
- Retour sur l'exemple des nombres entiers.

Définition: Deux expressions rationnelles e_1 et e_2 sont dites équivalentes si elles définissent le même langage, autrement dit si $L(e_1) = L(e_2)$. On note $e_1 \equiv e_2$.

Exemple: $(ab)^* \equiv \epsilon + a(ba)^*b$.

Exercices: Écrire des expressions rationnelles qui définissent les langages suivants:

1. Toutes les chaînes de 0 et 1 se terminant par 0
2. Toutes les chaînes de 0 et 1 contenant au moins un 0
3. Toutes les chaînes de 0 et 1 contenant au plus un 1
4. Toutes les chaînes de 0 et 1 dans lesquelles toutes les séries de 1 ont une longueur paire. (ex: 0011011110011 est dans le langage mais pas 0110100)
5. Toutes les chaînes de 0 et 1 contenant un nombre pair de 1

Décrire les langages définis par les expressions rationnelles suivantes:

1. $(a|b)^*$
2. $(a^*b^*)^*$
3. $(a^*ba^*b)^*a^*$

Automates finis

Définition: Un automate fini $\mathcal{A}$ est un quintuplet (A, Q, I, T, E) où:

- A est un alphabet.
- Q est un ensemble fini appelé ensemble des états.
- I et T sont des sous-ensembles de Q appelés ensembles des états initiaux et terminaux (ou états d'acceptation), en général I est un singleton.
- E est une partie de $Q \times A \times Q$ appelée ensemble des transitions (ou flèches). Pour une transition (o, a, d), o est l'origine de la transition, a l'étiquette et d la destination.

Calcul et langage reconnu

Un calcul de longueur n sur l'automate $\mathcal{A}$ est une séquence de flèches (q_i, a_i, q_{i+1}), pour i variant de 0 à $n-1$ avec $q_0 \in I$. Le calcul est dit réussi si $q_n \in T$. Le mot $a_0 \dots a_{n-1}$ est appelée étiquette du calcul. Un mot est dit reconnu par l'automate $\mathcal{A}$ s'il étiquette un calcul réussi sur $\mathcal{A}$. Le langage reconnu par $\mathcal{A}$, noté $L(\mathcal{A})$, est l'ensemble des mots reconnus par l'automate. Deux automates sont dits équivalents s'ils reconnaissent le même langage.

Exemples

1. Donner un automate reconnaissant l'ensemble des mots sur $\{a, b\}$ avec un nombre impair de a.
2. Donner un automate reconnaissant l'ensemble des mots sur $\{0, 1\}$ contenant un nombre pair de 1 et un nombre impair de 0.
3. Donner un automate reconnaissant l'ensemble des mots sur $\{0, 1\}$ contenant trois 1 à la suite.
4. Donner un automate reconnaissant l'ensemble des entiers pairs.
5. Exemples de deux automates reconnaissant tous les deux le langage défini par l'expression rationnelle $(a + b)c^*b$.

Automate complet

Un automate est dit complet si pour tout état q et toute étiquette a il existe une transition du type (q, a, q'), autrement dit si tous les calculs terminent.

Il est très facile de compléter un automate, il suffit de rajouter un état "puits" vers lequel on fait aller toutes les transitions manquantes.

Exemples:

1. Donner un automate complet reconnaissant l'ensemble des nombres exprimés par l'expression rationnelle $0^*1^*2^*$ (on travaille sur l'alphabet des chiffres de 0 à 9).
2. Compléter les automates de la partie précédente.

Remarque: Avoir un automate complet permet de construire un automate reconnaissant le complémentaire d'un langage.

Automate déterministe

Un automate est dit déterministe si:

- I est un singleton.
- Si $q \in Q$ et $a \in A$ il existe au plus une transition du type (q, a, q'),

en d'autres termes un mot de A^* étiquette au plus un calcul de l'automate.

Fonction de transition

Dans le cas d'un automate déterministe complet, on peut définir la fonction de transition $\gamma : Q \times A \to Q$ qui à (q, a) associe l'unique état q' tel que $(q, a, q') \in E$.

Déterminisation d'un automate

Déterminiser un automate, c'est trouver un automate déterministe qui lui est équivalent. C'est toujours possible.

La méthode générale est la suivante: soit $\mathcal{A} = (A, Q, I, T, E)$ l'automate de départ. Son déterminisé est l'automate $\mathcal{D}$ tel que:

- Son alphabet est A (le même que $\mathcal{A}$).
- L'ensemble de ses états est $\mathcal{P}(Q)$.
- Il a un unique état inital: I.
- Ses transitions sont de la forme $(e, a, \{q' \in Q, \exists q \in e, (q, a, q') \in E\})$.
- Les états terminaux sont les sous-ensembles contenant au moins un étal terminal de $\mathcal{A}$.

Exercice: Montrer que ces deux automates reconnaissent le même langage (procéder par double-inclusion et récurrence sur la taille du mot).

Exemple: Donner un automate déterministe reconnaissant l'ensemble des mots sur $\{a, b\}$ commençant par a et finissant par b.

Remarque: la déterminisation peut avoir un coût exponentiel en la taille de l'automate de départ.

Exemple: Soit $n \in \mathbb{N}^*$. Donner un automate déterministe reconnaissant le langage défini par l'expression rationnelle $(a + b)^* a (a + b)^n$.

Montrons qu'un tel automate doit nécessairement comporter 2^n états à l'aide de la méthode dite de séparation d'états. A tout mot u de n lettres on peut associer l'état $q(u)$ atteint en lisant le mot u. On définit ainsi une application $q : \{0, 1\}^n \to Q$ et nous allons montrer qu'elle est injective. Soient deux mots u, v distincts de n lettres. On peut les décomposer $u = u'aw$ et $v = v'bw$ où w est leur plus long suffixe commun. La taille de w est inférieure ou égale à $n - 1$ donc on peut compléter w en un mot ww' de longueur $n - 1$. Alors le mot $uw' = u'aww'$ est reconnu par l'automate mais pas le mot $vw' = v'bww'$. Or l'automate est déterministe donc on ne peut pas parvenir sur le même état en lisant u et en lisant v, donc $q(u) \neq q(v)$.