Jeudi 23 Février 2006

Langages réguliers (3) et Fonctions séquentielles

Reprise d'exercices antérieurs.

Exercice 1 (Automates universels)
Un automate fini $\mathcal{A}$ peut-être interprété comme étant universel : un mot est accepté si tous les calculs sur ce mot sont accepteurs. On note $L_\forall(\mathcal{A})$ son langage.

(i) Montrer que si $\mathcal{A}$ est un automate fini, $L_\forall(\mathcal{A})$ est rationnel.

(ii) On suppose L rationnel, montrer que les langages suivants sont rationnels :
$\{w \mid \exists(u,v)\ w = uv \text{ et } v \in L\}$ et $\{w \mid \forall(u,v)\ w = uv \Rightarrow v \in L\}$

Exercice 2 (Double renversement)

1. Soit L un langage rationnel. Montrer que le déterminisé d'un automate co-déterministe co-accessible qui reconnaît L est l'automate minimal de L.
2. On note $\mathcal{A}^t$ l'automate $\mathcal{A}$ dans lequel on a inversé le sens des flèches et échangé états initiaux et finaux. Montrer que $(((\mathcal{A}^t)_{det})^t)_{det}$ est l'automate minimal de $L(\mathcal{A})$.

Morphismes et monoïdes.

Exercice 3 (Monoïde syntaxique)
Calculer le monoïde syntaxique du langage des mots se terminant par ab.

Exercice 4 (Langages sans étoile)
Le langage $L = ((a + cb^*a)c^*b)^*$ est-il sans étoile ?

Exercice 5 (Congruence à droite)
Rappels : Soit $L \subseteq \Sigma^*$ et $\equiv$ une congruence sur Σ^*.
Le langage L est *saturé par* $\equiv$ si $\forall u \in \Sigma^*, \forall v \in L, u \equiv v$ implique $u \in L$.
La congruence $\equiv$ est dite *congruence à droite* si $\forall u, v, w \in \Sigma^*, u \equiv v$ implique $uw \equiv vw$.
La congruence $\equiv$ est dite *d'index fini* si elle possède un nombre fini de classes d'équivalence.
Enfin, étant donné un langage L, on définit la congruence $\equiv_L^r$ sur Σ^* par $u \equiv_L^r v$ si $\forall y \in \Sigma^*$, $uy \in L \iff vy \in L$.

1. Soit $L \subseteq \Sigma^*$. Montrer que L est reconnaissable si et seulement si L est saturé par une congruence à droite d'index fini.
2. Montrer que $\equiv_L^r$ est la congruence à droite la plus grossière qui sature L.

3. Faire le lien entre $\equiv_L^r$ et l'automate minimal de L.
4. Application : calculer $\equiv_L^r$ pour le langage de l'exercice 3.

Exercice 6 (Reconnaissables et concaténation)

1. En utilisant la représentation par monoïdes, montrer que les reconnaissables sont stables par concaténation.
   Indication : Si L_i $(i = 1, 2)$ est reconnu par $\phi_i : A^* \to M_i$, on pourra considérer l'application
   $$\begin{array}{rccl} \phi : & A^* & \to & 2^{M_1 \times M_2} \\ & w & \mapsto & \{(\phi_1(u), \phi_2(v)) \mid w = uv\} \end{array}$$
2. En déduire que tout langage sans étoile est apériodique (ce qui correspond à une implication du théorème de Shützenberger).

Fonctions séquentielles

Exercice 7 (Opérations sur les entiers)

Dans cet exercice on considère deux représentations des entiers en base 2 (ou plus généralement en base β) : la représentation *classique* (avec le bit de poids fort en premier) et la représentation *inverse* (avec le bit de poids faible en premier).

1. Donner un transducteur séquentiel qui réalise la multiplication par 3 en base 2 (représentation inverse).
2. Donner un transducteur séquentiel qui réalise la division entière par 3 en base 2 (représentation classique).
3. On définit l'opérateur comparaison comme la fonction prenant en argument deux entiers x et y en représentation inverse et qui retourne $\top$ si $x \geq y$ et $\bot$ sinon. Donner un transducteur séquentiel qui réalise la comparaison. (on suppose que les codages de x et y ont même longueur)
4. On définit la soustraction comme la fonction prenant deux entiers x et y en représentation inverse et qui retourne l'entier $x - y$ en base 2 inverse si $x \geq y$ et le caractère $\#$ en dernier sinon. Donner un transducteur séquentiel qui réalise la soustraction.
5. Donner un transducteur séquentiel qui reconnaît l'ensemble des représentations classiques d'entiers vérifiant $2x + 3y \equiv 1[6]$. De même pour $\exists y \mid 2x + 3y \equiv 1[6]$.

Exercice 8 (Codes à délai de déchiffrage fini)

1. Soit $\beta_1 : \{x, y\}^* \to A^*$ le morphisme défini par $\beta_1(x) = a$ et $\beta_1(y) = aba$. La relation β_1^{-1} est-elle une fonction séquentielle ?
2. Même question avec $\beta_2 : \{x, y, z\}^* \to A^*$ défini par $\beta_2(x) = ab$, $\beta_2(y) = abb$ et $\beta_2(z) = baab$.

3. Généralisation. Soit X un sous-ensemble fini de A^* ; X est dit *préfixe* si aucun mot de X n'est préfixe d'un autre mot de X. Soit $B = \{x_1, \ldots, x_n\}$ un ensemble en bijection avec X ; cette bijection induit un morphisme $\beta : B^* \to A^*$. Par définition, X est un *code* si ce morphisme est injectif. Un code X est dit *à délai de déchiffrage* d si quand un mot $f = x_1x_2 \cdots x_{d+1} \in X^{d+1}$ est *préfixe* (en tant que mot de A^*) d'un mot $g = y_1y_2 \cdots y_r \in X^*$, alors on a $x_1 = y_1$.
   (a) Vérifier qu'un code préfixe est un code à délai de déchiffrage 0.
   (b) Donner un exemple de code qui n'est pas à délai de déchiffrage fini.
   (c) Montrer que si X est préfixe, β^{-1} est une fonction séquentielle pure.
   (d) Montrer que si X est un code à délai de déchiffrage fini, β^{-1} est une fonction séquentielle.
   (e) Réciproquement, montrer que si β^{-1} est une fonction séquentielle, alors X est un code à délai de déchiffrage fini.

   On a démontré la :

   Proposition 1
   Soit $\beta : B^* \to A^*$ un morphisme. L'ensemble $X = \beta(B)$ est un code à délai de déchiffrage fini si et seulement si β^{-1} est une fonction séquentielle.