Langages formels, calculabilité et complexité

Examen du 2 février 2012

Corrigé, version $\beta 1$

Exercice 1 – *Grammaires : un petit exercice*

On considère le problème ESTFINI suivant

> Étant donné une grammaire hors contexte, décider si elle engendre un langage fini.

1. Analyser la décidabilité/complexité de ce problème.

Correction. Mettons la grammaire en forme normale de Chomsky. Construisons un graphe G dont les sommets sont les non-terminaux de la grammaire et il y a un arc de A à B s'il existe une règle $A \to BC$ ou $A \to CB$ dans la grammaire.

Lemme 1*Le langage est infini si et seulement si le graphe* G *contient un cycle.*

Preuve. A COMPLÉTER.
Le parcours de graphe en largeur ou en profondeur permet de détecter un tel cycle, donc le problèmes est **décidable**.
Toutes les étapes de l'algorithme (mise en forme de Chomsky, construction de graphe, recherche de cycle) sont polynomiaux, donc le problème est dans la classe P.

Exercice 2 – *Calculabilité : problème de Collatz et fonctions analytiques*

Une séquence de Collatz générale est définie par l'affectation

$$x := \mathrm{Col}(x) = \begin{cases} a_1 x + b_1 & \texttt{si} \quad x \mod p = r_1 \\ \quad \ldots \quad \ldots & \ldots \\ a_k x + b_k & \texttt{si} \quad x \mod p = r_k \end{cases}$$

Les coefficients a_i et b_i sont des constantes rationnelles ; p et r_i des constantes entières, toutes les r_i sont différentes. On itère cette affectation jusqu'a ce que x devienne 1(ou jusqu'à l'infini). Si x devient non-entier, on s'arrête aussi (une pathologie à éviter).

On analyse le problème COLLATZ suivant :

> Étant donnés un système de Collatz général et une valeur initiale de x (naturelle), décider si la séquence commençant par x atteint éventuellement la valeur 1.

1. Montrer que COLLATZ est semi-décidable (récursivement énumérable).

Correction. Le semi-algorithme pour COLLATZ est comme ceci : pour un x donné itérer la fonction jusqu'à ce qu'on obtienne 1. Dans ce cas répondre "OUI". Donc le problème est semi-décidable.

2. Montrer que COLLATZ est indécidable

Correction. Étant donnée une Machine à 2 compteurs (de Minsky) M on la simulera par Collatz en choisissant S premier et supérieur au nombre d'instructions de M, et en codant le fait que la machine se trouve dans l'état q_i avec compteurs m et n par $x = 2^m 3^n S + i - S$ (c'est un codage injectif).
Faisons quelques observations :
–l'état q_1 avec $m = n = 0$ correspond à $x = 1$
–pour connaître le numéro de l'état où se trouve M il suffit de calculer $x \mod S$
–Dans l'état q_i on a $m = 0$ ssi $x - i + S \mod 2 \neq 0$. Pareil, $n = 0$ ssi $x \mod 3 \neq 0$.
–Pour incrémenter (décrémenter m) il suffit de multiplier x par 2 (resp. par $1/2$) et ajouter une constante qui corrige le numéro d'instruction active. Pour n on multiplie par 3 ou $1/3$.
Pour traduire M en système de Collatz on représente chaque instruction de la machine à compteurs par une ou 2 lignes de système de Collatz selon la table (on omet les 3 instructions pour n qui sont similaires) :

Compteurs	Collatz
dans q_i faire m++, aller à q_j	$x := (x - i) * 2 + j$ si $x \mod S = i$
dans q_i faire m–, aller à q_j	$x := (x - i)/2 + j$ si $x \mod S = i$
dans q_i si m=0, aller à q_j sinon q_k	$x := x - i + j$ si $x \mod S = i$ et $x - i + S \mod 2 = 1$
	$x := x - i + k$ si $x \mod S = i$ et $x - i + S \mod 2 = 0$

(les conditions sur les modulos peuvent toutes être exprimées avec $\mod p$ où $p = 6S$.)
la séquence de x du système de Collatz ainsi obtenu correspond à la séquence des (q_i, m, n) de la machine M. En particulier,

M à partir de (q_0, m, n) atteindra $(q_1, 0, 0)$ ssi Collatz de $2^m 3^n S - S$ atteindra 1.

Si COLLATZ était décidable, l'atteignabilité dans les Machines à 2 compteurs le serait aussi. Or ce dernier n'est pas décidable, on conclut que COLLATZ aussi.

3. Construire une fonction $f : \mathbb{R} \to \mathbb{R}$ définissable par une formule trigonométrique explicite telle que pour les x naturels $f(x) = 1$ si $x \mod p = 0$ et $f(x) = 0$ sinon.

Correction. La fonction

$$h(x) = \prod_{k=1}^{p-1} \sin^2(k - x)\pi/p$$

pour tout $x \equiv k(\mod p)$ avec $k = 1..p - 1$ est nulle puisque son facteur $\sin^2(k - x)\pi/p$ est nul. Pour tous les x entiers multiples de p la valeur $h(x)$ est la même et non nulle

$$h(x) = h(0) = \prod_{k=1}^{p-1} \sin^2 k\pi/p.$$

Donc la fonction $f(x) = h(x)/h(0)$ satisfait l'énoncé.

4. Construire une fonction $g : \mathbb{R} \to \mathbb{R}$ définissable par une formule trigonométrique explicite qui coïncide avec Col sur tous les x naturels.

Correction.

$$g(x) = \sum_{i=0}^{p} (a_i x + b_i) f(x)$$

avec f définie dans le point précédent.

5. Déduire un théorème d'indécidabilité pour les itérations des fonctions trigonométriques. Formuler précisément l'énoncé et achever la preuve.

Correction.
Définition 1*Une fonction trigo est une fonction de x qu'on peut obtenir à partir de*

$$\sin a\pi x, \cos a\pi x, \sin a\pi, \cos a\pi, x, a$$

(avec $a \in \mathbb{Q}$) en n'appliquant que des opérations arithmétiques.
On définit le problème TRIGO comme ceci

Étant donné une fonction trigo f et un entier x, est-ce qu'il il'éxiste n tel que l'itération $f^n(x)$ égale 1 ?

Théorème 1*Le problème* TRIGO *est indécidable.*
Effectivement, grâce au point 2 COLLATZ est indécidable, et grâce au point 4 COLLATZ se réduit à TRIGO. Donc TRIGO est aussi indécidable.

Exercice 3 – *Complexité : étude d'un problème*

On analyse le problème EVALUER-FAMILLE suivant :

On a une famille C de sous-ensembles d'un ensemble A et un entier J. Est-il possible d'obtenir tous les éléments de C à partir de singletons en utilisant l'opération $\cup$ (union) au plus J fois ?

1. Montrer que EVALUER-FAMILLE est dans la classe NP.

Correction. L'algorithme non-déterministe consiste à deviner dans quel ordre on efféctue les J unions à partir des singletons, et où se trouve chaque ensemble de C dans la séquence obtenue, à faire ces unions, et à vérifier qu'effectivement tous les ensembles de C sont obtenus. Tous les calculs nécessaires sont polynomiaux, donc le problème est dans NP.

2. Montrer que EVALUER-FAMILLE est NP-complet.

Correction. Comme indiqué, on réduira le problème TRANSVERSAL (problème NP-complet) à EVALUER-FAMILLE. Une instance de ce problème est un graphe G et un entier K. Pour faire la réduction, on construit une famille C qui contient $\{a_0, u, v\}$ pour chaque arête (u, v) de G, et on prend $J = K+n$ où n est le nombre d'arêtes de G. Cette construction est clairement polynomiale, il reste à démontrer que c'est effectivement une réduction. Ca se ramène à deux lemmes :

Lemme 2*Si le graphe* G *possède un transversal de* K *éléments, alors on peut construire* C *en* J *unions.*

Preuve. Soit T ce transversal. Pour chaque $w \in T$ on réunit v avec a_0 et on obtient $\{a_0, w\}$ (ça prendra K opération. Pour chaque arête (u, v) soit u soit v est dans T, supposons que c'est u. Donc l'ensemble $\{a_0, u\}$ est déjà construit. En faisant union avec v on obtient $\{a_0, u, v\}$. Ça necessite une union par arête, et on obtient C en $K + n = J$ opérations. □

Lemme 3*Si on peut construire* C *en* J *unions, alors le graphe* G *possède un transversal de* K *éléments.*

Chaque ensemble $\{a_0, u, v\}$ est obtenu d'une de deux façons :
Aréunir a_0 et u, puis ajouter v (ou la symétriques) ;
Bréunir u et v, puis ajouter a_0.
Preuve. Dans le cas A on ne change rien, dans le cas B on modifie la construction - on réunit d'abord a_0 et u, puis ajouter v (ça n'a aucune influence sur le reste de la construction puisque $\{u, v\}$ qui disparaît est inutile pour les autres éléments de C). Ainsi on obtient une nouvelle construction de la famille C en J opérations utilisant seulement la méthode A. Dans cette nouvelle construction il y a n union de type $\{a_0, u\} \cup \{v\}$ (une pour chaque arête) et $L - n = K$ opérations de type $\{a_0\} \cup \{u\}$. En plus, cette dernière opération est faite pour un sommet de chaque arête.
Soit T l'ensemble de K sommets u telle que $\{a_0\} \cup \{u\}$ est faite. Il est transversal. □

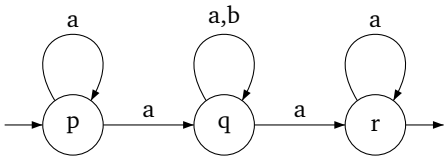
Exercice 4 – *Automates et langages : mots biinfinis*

Soit $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ un automate fini (I, F sont deux ensembles d'états). Un calcul sur un mot bi-infini

$$u = \ldots a_{-1} a_0 a_1 \ldots$$

est une suite bi-infinie d'états $(q_i)_{i\in\mathbb{Z}}$ telle que : $\forall i, (q_i, a_i, q_{i+1}) \in \Delta$. Un calcul est *accepteur* s'il existe une infinité de $i \leqslant 0$ tels que $q_i \in I$ et une infinité de $i \geqslant 0$ tels que $q_i \in F$. Un mot bi-infini est *reconnu* par l'automate $\mathcal{A}$ s'il possède un calcul accepteur. On définit de façon immédiate l'ensemble des langages *reconnaissables* de mots bi-infinis sur un alphabet Σ.

1. Donner un automate sur l'alphabet $\{a, b\}$ qui reconnaît l'ensemble des mots bi-infinis ayant un nombre fini de b.



Correction.
Pour tout mot w avec un nombre fini de b le calcul accepteur es fait comme ceci : il reste dans p jusqu'au symbole précédant le premier b, puis reste dans q jusqu'au dernier b , puis va dans r et y reste pour toujours (sur le mot $a^{\mathbb{Z}}$ qui ne contient aucun b le calcul reste dans p jusqu'à la position 0, puis va dans q, puis tout de suite dans r). Donc il est accepté par l'automate.
Tout calcul accepteur doit rester dans p sur toutes les positions $-\infty..m$, et dans r sur toutes les positions $n..\infty$. Donc il passe sur q au plus $n - m$ pas. Mais les lettres b sont possibles seulement sur q, donc le mot accepté contient un nombre fini de q.

2. Soit L_0 l'ensemble de tous les mots bi-infinis sur $\{a, b\}$ avec la lettre b en position 0. Est-il reconnaissable ?

Correction. Si on décale un calcul accepteur de k positions, il reste un calcul accepteur. Le mot accepté est ainsi décalé de k positions. On déduit le lemme suivant :
Lemme 4*Un langage bi-infini reconnaissable est invariant par décalage ; avec chaque mot w il contient le mot $v = dec(v)$ tel que $\forall i(v_i = w_{i+k})$.*
Or le langage de l'énoncé n'est pas invariant par décalage : il contient un seul mot, mais pas ses décalés. Donc il n'est pas reconnaissable.

3. Étudier les propriétés de clôture des langages bi-infinis reconnaissables. *Indication: Commencer par l'union, si le temps permet traiter d'autres opérations.*

Correction.
Les langages reconnaissables infinies sont clos par union, intersection et même complément.
Pour faire un automate qui reconnait l'union de $L(A)$ et $L(B)$ il suffit de faire une union disjointe de deux automates A et B.
Pour faire intersection de $L(A)$ et $L(B)$ il faudra faire l'automate produit $A \times B$ et puis l'enrichir un peu pour assurer que I_A et I_B ainsi que F_A et F_B sont visités infiniment souvent en alternance. A COMPLÉTER.
La clôture par complément est sans doute trop compliquée pour la faire pendant un examen.
On ne peut pas concaténer deux ou plusieurs mots bi-infinis et obtenir un mot bi-infini. On ne parlera donc pas de clôture par concaténation ou autre itération. On pourrait étudier shuffle, morphisme, morphisme inverse.

4. Définir (formellement) les expressions régulières bi-infinies (BRE) et leur sémantique. Donner une BRE pour le langage du premier point de cet exercice.

Correction. Soit RE les expression rationnelles usuelles sans ε sur un alphabet Σ (pour les langages réguliers de mots finis), on rappelle la définition de la syntaxe (mais on omet la sémantique)

$$RE ::== a|RE + RE|RE \cdot RE|RE^*,$$

où $a \in \Sigma$. On peut maintenant définir les BRE

$$BRE ::== RE^{-\omega} \cdot RE \cdot RE^{\omega}|BRE + BRE.$$

La sémantique des BRE est définie par induction structurelle avec deux cas.
–l'union est facile $[e + f] = [e] \cup [f]$ pour BRE e et f.
–l'autre opération et un peu plus subtile. Soient e, f, g trois RE. Alors le langage

$$e^{-\omega} \cdot f \cdot g^{\omega}$$

consiste de tous les mots bi-infinis w tels qu'il existe une séquence d'indices bi-infinie strictement croissante i_n avec trois propriétés

–pour tout $n < 0$ le sous-mot de w de la position i_n jusqu'à $i_{n+1} - 1$ appartient à $[e]$;
–(pour $n = 0$) le sous-mot de w de la position i_0 jusqu'à $i_1 - 1$ appartient à $[ef]$;
–pour tout $n > 0$ le sous-mot de w de la position i_n jusqu'à $i_{n+1} - 1$ appartient à $[g]$.
(On remarque que cette définition est bien invariante par décalage)
La BRE pour le langage de point 1 : $a^{-\omega} \cdot (a + b)^* \cdot a^{\omega}$.

5. Démontrer qu'un langage de mots bi-infinis est reconnaissable si et seulement si il peut être exprimé par une BRE.

Correction.

Reconnaissable $\Rightarrow$ **rationnel** Dans un automate A, pour deux états p et q soit L_{pq} l'ensemble de tous les mots finis non-vides qui mènent de p à q. Par théorème de Kleene ce langage peut être défini par une RE.

Le langage de mots bi-infinis reconnus par A est :

$$\bigcup_{i \in I, f \in F} {L_{ii}}^{-\omega} \cdot L_{if} \cdot {L_{ff}}^{\omega}.$$

En remplaçant chaque L_{pq} par son RE et le $\bigcup$ par une somme on obtient une BRE pour L.

Rationnel $\Rightarrow$ **reconnaissable** Induction structurelle. Il y a deux cas
–l'union est facile l'automate pour BRE $e + f$ est l'union disjointe des automates pour $e + f$.
–étant donnée une BRE $e^{-\omega} \cdot f \cdot g^{\omega}$ on construit d'abord les automates finis A, B, C pour les RE e, f, g. On les fait "normaux", c-à-d avec un seul état initial sans transitions entrantes, et avec un seul état final sans transition sortantes. On fait l'union disjoint de A, B et C, et puis on identifie certains états :
–i_A avec f_A ;
–f_A avec i_B ;
–f_B avec i_C ;
–i_C avec f_C.
Le seul état initial de l'automate obtenu et i_A, le seul état final est f_C. A COMPLÉTER. : faire un dessin.