Langages formels, calculabilité et complexité

Examen de rattrapage

25 avril 2013, durée 3h, sans documents

1 On applique les cours

Exercice 1 – *Récursives primitives*

Soient `petit(x)` et `grand(x)` le plus petit et le plus grand facteurs premiers de x. Montrer que les fonctions `petit` et `grand` sont récursives primitives.

Correction.

$$\texttt{petit}(x) = \mu y \leqslant x.(\texttt{estPremier}(y) \wedge y|x)$$

Les prédicats "`estPremier`" et "|" sont **rp**, opération $\wedge$ préserve la récursivité primitive, et la minimisation bornée appliquée à un prédicat **rp** donne une fonction **rp**. Donc, la fonction `petit` est **rp**.

$$\texttt{grand}(x) = \mu y \leqslant x.(\texttt{estPremier}(y) \wedge y|x \wedge \neg\exists z \leqslant x(z > y \wedge \texttt{estPremier}(z) \wedge z|x))$$

(en fait c'est l'unique y satisfaisant cette condition). Les prédicats "`estPremier`", "$<$", et "|" sont **rp**, opération $\wedge$, $\neg$ et la quantification existentielle bornée préservent la récursivité primitive, et la minimisation bornée appliquée à un prédicat **rp** donne une fonction **rp**. Donc, la fonction `petit` est **rp**.

Exercice 2 – *Mots infinis*

On considère le langage T de tous les mots infinis sur $\{a, b\}$ qui contiennent un nombre infini de b sur des positions impaires.

1. Trouver une expression ω-régulière définissant T.
2. Trouver un automate de Büchi reconnaissant T.
3. Trouver une formule MSO définissant T.

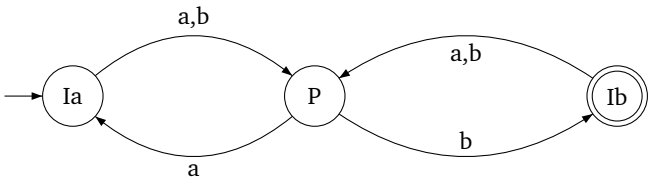
Correction. On suppose que le mot commence par la position 0.

1.Une expression ω-régulière définissant L

$$\Big((a+b)((a+b)^2)^*b\Big)^\omega$$

Les b sont ici intercalés par des quantités impaires de caractères arbitraires.

2.*Trouver un automate de Büchi reconnaissant* L.



Chaque fois quand il y a un b en position impaire on visite l'état accepteur.

3. *Trouver une formule MSO définissant* L. L'idée serait d'introduire une variable X de second ordre qui représente l'ensemble de positions impaires, et de dire qu'il y a des b dans X dans une infinité de positions.

$$\exists X\,(\neg X(0) \wedge (\forall i\,(X(i) \Leftrightarrow \neg X(S(i)))) \wedge (\forall m \exists n\,(n > m \wedge X(n) \wedge b(n))))$$

Exercice 3 – *Un langage non-régulier*

Le langage L consiste de tous les mots sur l'alphabet $\{a, b\}$ qui contiennent deux fois plus de a que de b.

1. Montrer que L n'est pas régulier.
2. Trouver une grammaire hors contexte qui génère L.
3. Trouver un automate à pile qui accepte L.

Correction.

1. On suppose le contraire : L est régulier. Soit n la taille de l'automate déterministe le reconnaissant. Prenons le mot $w = b^n a^{2n} \in L$. Par *pumping lemma* il admet une décomposition $w = xyz$ telle que $w' = xz \in L$ et encore $|xy| \leqslant n$ et $y \neq \varepsilon$. Donc y a la forme $y = b^k$ avec $k > 0$. On déduit que w' contient $2n - k$ lettres b et n lettres a, par définition du langage L il est impossible que $w' \in L$. Contradiction termine la preuve.
2. On propose la grammaire suivante :

$$\begin{aligned} S &\to \varepsilon | aAbS | aSbA | bSaA \\ A &\to aS | bAaA \\ B &\to aAbB | aSbS | bA \end{aligned}$$

   Pour un mot w soit $f(w) = \#a - 2\#b$. Le langage engendré par S est $L = \{w | f(w) = 0\}$, celui par A est $\{w | f(w) = 1\}$, celui par A est $\{w | f(w) = -1\}$. Démontrons-le.
3. On fera un automate à pile non-déterministe avec un seul état qui acceptera L par pile vide. Il lira le mot d'entrée et mémorisera dans sa pile le nombre de a observés moins deux fois le nombre de b. Pour désigner un nombre positif k la pile contiendra k fois $+$; pour un négatif $-k$ elle contiendra k fois $-$. Voici le programme commenté (avec Z le symbole du fond de pile) :

$$\begin{aligned} q, + &\xrightarrow{a} q, ++ && \text{pile} > 0, a \\ q, Z &\xrightarrow{a} q, + && \text{pile} = 0, a \\ q, - &\xrightarrow{a} q, \varepsilon && \text{pile} < 0, a \\ q, ++ &\xrightarrow{b} q, \varepsilon && \text{pile} > 1, b \\ q, Z+ &\xrightarrow{b} q, Z- && \text{pile} = 1, b \\ q, Z &\xrightarrow{b} q, Z-- && \text{pile} = 0, b \\ q, - &\xrightarrow{b} q, Z--- && \text{pile} < 0, b \end{aligned}$$

2 On réfléchit un peu plus

Exercice 4 – *Machines à file — indécidabilité*

Une file (FIFO) est un modèle de mémoire capable de stocker une chaîne de caractères d'un alphabet, avec deux opérations : `enqueue(a:char)`, qui ajoute a dans la file en tant que le dernier élément et `dequeue():char` qui lit le premier élément et l'enlève de la file (si la file est vide cette fonction retourne 0). La machine à file (MàF) a un ensemble fini d'états de contrôle et une seule file. L'entrée x est représentée par la file initialisée à 1^x. Le résultat de calcul est le nombre de 1 dans la file après l'arrêt de la MàF

1. **Formalisation :** Donner une définition formelle de MàF. Soyez succincts.
   (a) La syntaxe. *Indication: une MàF est un tuple de forme...*
   (b) La sémantique. *Indication: une configuration, une transition, une exécution, la fonction calculée*
2. **Programmation :** Décrire les MàF qui calculent les fonctions suivantes :
   (a) $f(x) = 2x$
   (b) $g(x, y) = x \dot{-} y$ (l'expression $x \dot{-} y$ vaut $x - y$ si $x \geqslant y$ et 0 sinon.
3. **Simulation :** Simuler une machine de Turing (ou, si vous préférez, une machine à 2 compteurs) avec une MàF.
4. **R.e. :** Montrer qu'un ensemble $A \subseteq \mathbb{N}$ est récursivement énumérable si et seulement s'il existe une MàF M telle que $A = \{n | M \text{ s'arrête à partir de la file } 1^n\}$
5. **Indécidabilité du problème d'arrêt :**Le problème d'arrêt pour MàF est le suivant : étant donnés une MàF M et un entier n, décider si M s'arrête à partir de la file 1^n. Prouver que ce problème d'arrêt n'est pas décidable.

Correction.

1.**Formalisation :**

(a)La syntaxe : Un MàF est un tuple de la forme Q, Σ, q_0, Δ. Ici Q est un ensemble fini d'états, $\Sigma \ni 1$ un alphabet fini, $q_0 \in Q$ l'état initial, Δ le programme (relation de transition) - un ensemble fini de tuples $(p, a \to q, w) \in Q \times \Sigma \cup \varepsilon \times Q, \Sigma*$ (ça se lit comme ça : "dans l'état p on peut défiler a, enfiler w et passer vers l'état q"). On exige que la machine soit déterministe :
–s'il y a une transition $p, \varepsilon \to \ldots$, alors il n'y a aucune autre transition de p
–il ne peut pas y avoir deux transitions avec les mêmes p, a.

(b)La sémantique.

une configurationest un couple $c = (q, f) \in Q \times \Sigma^*$ - un état et un contenu de la file.

une transition$(p, af) \to (q, fw)$ telle qu'il existe une ligne de programme qui le permet : $(p, a \to q, w) \in \Delta$

une exécution sur $x \in \mathbb{N}$une chaîne de configurations et transitions $c_0 \to c_1 \to \cdots \to c_T$ telle que
–$c_0 = (q_0, 1^x)$: on démarre dans l'état initial avec x symboles 1 dans la files ;
–pour tout i on a que $c_i \to c_{i+1}$ est une transition ;
–On s'arrête parce qu'il n'y pas de transitions qui peuvent s'appliquer à c_T
–le résultat de cette exécution est le nombre total de 1 dans la file de c_T.

la fonction $f : \mathbb{N} \to \mathbb{N}$ **calculée**$f(x)$ est le résultat de l'exécution de la machine sur x (si cette exécution s'arrête).

2.**Programmation :**

(a)$f(x) = 2x$ Initialement dans la file il y a x symboles 1. On enfile 0, puis systématiquement on défile 1 et on enfile 11. Quand on tombe sur 0 le calcul est terminé.
FAIRE UN DESSIN ! ! !
Formellement la machine a deux états $\mathbb{Q} = \{i, b\}$ et Δ contient deux instructions

$i, \varepsilon \to b, 0$	initialisation :on enfile 0 pour séparer
$b, 1 \to b, 11$	boucle principale : à chaque tour on défile 1 et on enfile 11

(b)$g(x, y) = x \dot{-} y$ (l'expression $x \dot{-} y$ vaut $x - y$ si $x \geqslant y$ et 0 sinon. On suppose qu'initialement la file contient $1^x 0 1^y$. On enfile 0, puis on commence na boucle principale : faire un tour complet de la file en decrémentant x et y. Après le premier tour (qui prendra $\approx x + y$ transitions) on aura $1^{x-1}01^{y-1}$, après le second $1^{x-2}01^{y-2}$ etc. Si initialement $x \geqslant y$, alors à la fin on aura $1^{x-y}0$ et on s'arrêtera. Dans le cas contraire, à la sortie de la boucle principale on aura 01^{y-x} et on procédera à une boucle de nettoyage qui videra la file. Dans les deux cas le résultat sera $x \dot{-} y$.
FAIRE UN DESSIN ! ! !

Le programme est comme suit :

$i, \varepsilon \to b_0, 0$	initialisation :on enfile 0 pour séparer
$b_0, 1 \to b_1, \varepsilon$	début de boucle :on decrémente x
$b_1, 1 \to b_1, 1$	on défile le reste de x et on l'enfile de l'autre côté
$b_1, 0 \to b_2, 0$	on traverse la séparation entre x et y
$b_2, 1 \to b_3, \varepsilon$	on decrémente y
$b_3, 1 \to b_3, 1$	on défile le reste de y et on l'enfile de l'autre côté
$b_3, 0 \to b_0, 0$	fin de boucle principale
$b_0, 0 \to n, 0$	puisque $x < y$ on commence le nettoyage
$n, 0 \to n, \varepsilon$	on efface tous les 0
$n, 0 \to n, \varepsilon$	on efface tous les 1

(si $x > y$ on s'arrêtera dans b_2 en observant 0 dans la file).

3.**Simulation :**

On va simuler une machine à 2 compteurs M par une MàF F. On représentera les valeurs de compteurs x, y par la file 1^x01^y. L'état de la machine à compteurs sera représenté par l'état de F. On simulera chaque instruction de la machine à compteurs par une instruction de la machine à file. Certains cas sont faciles

–Si M contient l'instruction $p \overset{y++}{\to} q$ on doit transformer la file 1^x01^y en 1^x01^{y+1}. Pour cela on mettra dans le programme de F l'instruction $p, \varepsilon \to q, 1$ (c-à-d enfiler 1 à la fin de 1^y).

–Si M contient l'instruction $p \overset{x--}{\to} q$ on doit transformer la file 1^x01^y en $1^{x-1}01^y$. Pour cela on mettra dans le programme de F l'instruction $p, 1 \to q, \varepsilon$ (c-à-d défiler 0 au début de 1^yx).

D'autres sont plus difficiles, et ils sont représentés par un "tour de file" comme dans le programme pour $x \dot{-} y$.

–Si M contient l'instruction $p \overset{x++}{\to} q$ on doit transformer la file 1^x01^y en $1^{x+1}01^y$. Pour ceci on enfilera 01 (o pour séparer, 1 pour incrémnter x, puis, lettre par lettre on défilera 1^x et on le remettra à la fin de la file, et finalemnt, de la même manière on reportera y à la fin de la file. On ajoutera donc au programme de F les instructions suivantes :

$p, \varepsilon \to b_1, 01$	initialisation :on enfile 0 pour séparer et 1 pour incrémenter x
$b_1, 1 \to b_1, 1$	on défile x et on l'enfile de l'autre côté
$b_1, 0 \to b_2, 0$	on traverse la séparation entre x et y
$b_2, 1 \to b_2, 1$	on défile y et on l'enfile de l'autre côté
$b_2, 0 \to q, \varepsilon$	fin de boucle principale, on va vers q

–Si M contient l'instruction $p \overset{y--}{\to} q$ on procède de manière similaire avec un tout de file qui décrémentera y.

–le test $(p \overset{x=0}{\to} q, \text{ sinon } r)$ et l'autre pour y se font facilement avec un tour de boucle.

Ainsi on construira pour M donné une machine à file F qui la simule. Plus précisément, si M en démarrant en (q_0, x, y) s'arrête, alors F en démarrant de l'état q_0 avec la file 1^x01^y s'arrête aussi ; et la réciproque est aussi vraie.

4.**R.e. :**

Si A est r.e., alors il existe une fonction récursive partielle $\phi : \mathbb{N} \to \mathbb{N}$ telle que $A = dom(\phi)$. Toute fonction r.p. peut être calculée par une MàC, soit M_0 une telle machine pour ϕ. Par définition elle s'arrête à partir de la configuration $(q_0, n, 0)$ ssi $n \in A$. Soit F_0 la machine qui la simule construite dans le point précédent. En appliquant la dernière phrase du corrigé du point précédent, F en démarrant de l'état q_0 avec la file 1^n0 s'arrête ssi $n \in A$. Ça correspond à l'énoncé à l'exception d'un dernier 0 dans la pile. On construira donc une MàF un peu modifiée F. Elle commence par enfiler 0 et puis lance la machine F_0, il est immédiat que F satisfait l'énoncé.

5.**Indécidabilité du problème d'arrêt :**

Soit K un ensemble r.e. indécidable. Soit M une machine à file qui s'arrête à partir de de la file 1^n (on l'a construite dans le point précédent). Donc si on savait décider l'arrêt pour M on saurait décider K ce qui n'est pas le cas. Donc l'arrêt des MàF est indécidable

3 Un vrai théorème

Exercice 5 – *Encore des réels*

L'objectif de cet exercice consiste à démontrer la décidabilité d'une théorie et trouver un algorithme de décision (un tel algorithme est appliqué pour la vérification de programmes). Il s'agit de la théorie $\mathcal{T}$ de premier ordre avec la signature suivante :

$$(S) = (0, 1, \leqslant, +, Z).$$

On l'interprète sur le domaine de réels $\mathbb{R}$. L' interprétation de $0, 1, \leqslant, +$ est usuelle. Le prédicat $Z(x)$ est vrai si et seulement si x est un entier. On souhaite pour chaque formule close décider si elle est vraie dans cette interprétation.

1. Décrire en termes usuels l'ensemble de $x \in \mathbb{R}$ satisfaisant la formule $exo(x)$ ci-dessous :

$$\exists y \exists z\, (y + y + y = 1 \wedge Z(z))$$

2. Proposer une méthode pour représenter un nombre réel par un mot infini. Un vecteur réel de dimension k par un mot infini. Un ensemble de vecteurs de dimension k par un langage de mots infinis. N'oubliez pas de préciser quels sont les alphabets utilisés.
3. Associer un langage $S(f)$ de mots infinis à chaque formule dans la signature.
4. Montrer que le langage $S(exo)$ associé à la formule $exo(x)$ introduite au début de ce problème est ω-régulier (en exhibant un automate de Büchi ou une expression ω-régulière).
5. Donner un plan de preuve par induction structurelle du lemme principal suivant :

   Lemme 1 *Pour toute formule f, le langage $S(f)$ est ω-régulier.*
6. Démontrer le cas de base.
7. Démontrer le cas inductif.
8. Déduire la décidabilité de la théorie $\mathcal{T}$ et expliquer l' algorithme de décision.
9. Vous avez sans doute oublié que certains nombres réels admettent plus d'un développement en fraction décimale/ou binaire. Par exemple

$$\begin{aligned} 2,5 &= 2,4999999999999999999\ldots \quad \text{base 10} \\ 10,1 &= 10,011111111111111111\ldots \quad \text{base 2} \end{aligned}$$

   Comment corriger la preuve pour tenir compte de ce phénomène désagréable ?

Correction.

1.*Décrire en termes usuels l'ensemble de $x \in \mathbb{R}$ satisfaisant la formule $exo(x)$ ci-dessous :*

$$\exists y \exists z\, (y + y + y = 1 \wedge Z(z))$$

Il y avait un typo dans cette question qui la rend trop facile. La formule est vraie pour toutes les valeurs de x (on peut choisir $y = 1/3; z = 5$). L'ensemble de x est donc tout le $\mathbb{R}$.

2.*Proposer une méthode pour représenter un nombre réel par un mot infini.* On le représente en système décimal. Le mot commence par un signe, quelques chiffres pour la partie entière, virgule, et une infinité de chiffres pour la partie fractionnelle. L'alphabet utile serait $\Sigma = \{01\ldots9+-,\}$. Par exemple $+7,66666666666666\ldots$ représente $7\frac{2}{3}$

Un vecteur réel de dimension k *par un mot infini.* Pour un vecteur $(x_1,\ldots,x_k)$ on écrit les mots pour x_1, pour x_2, pour x_k un au-dessus de l'autre, en alignant les signes et les virgules. Par exemple, pour $(-25,5;7\frac{2}{3})$ on écrit

-	2	5	,	5	0	0	0	0	0	0	0	0	0	...
+	0	7	,	6	6	6	6	6	6	6	6	6	6	...

Une telle table est en fait un mot infini sur l'alphabet Σ^k, où chaque colonne correspond à une lettre.

Un ensemble E *de vecteurs de dimension* k *par un langage de mots infinis.* On prend le langage $L(E)$ de tous les mots sur l'alphabet Σ^k qui correspondent aux vecteurs de E.

3.*Associer un langage* $L(f)$ *de mots infinis à chaque formule dans la signature* $\mathcal{S}$. Une formule f avec k variables libres définit un ensemble E de vecteurs de dimension k. Dans la question précédente on y a associé un langage L de mots infinis. On appellera ce langage $L(f)$. **Petit problème : comment faire pour une formule close ?**

4.*Montrer que le langage* $L(exo)$ *associé à la formule* $exo(x)$ *introduite au début de ce problème est* ω*-régulier (en exhibant un automate de Büchi ou une expression* ω*-régulière).*

$L(exo) = (++-)(0..9)^*,(0..9)^\omega$

5.*Donner un plan de preuve par induction structurelle du lemme principal suivant :*

Lemme 2*Pour toute formule* f *le langage* $L(f)$ *est* ω*-régulier.*

On passe tout d'abord à une théorie $\mathcal{T}'$ de premier ordre avec la signature suivante :

$$\mathcal{S}' = (0,1,\leqslant,P,Z).$$

On a remplacé la fonction $+$ par un prédicat $P(x,y,z)$ avec le sens "$x+y=z$". On remarque que la théorie $\mathcal{T}$ se réduit à $\mathcal{T}'$.

L'avantage de cette nouvelle théorie est qu'elle possède trois types de termes seulement : 0, 1 et des variables.

On va donc démontrer le lemme pour chaque formule f en signature S', en utilisant l'induction structurelle sur la construction de f.

Il faut démontrer :

–**Le cas de base**. $L(f)$ est ω-régulier pour les formules atomiques. Il y a les cas suivants à traiter.
  –$Z(0), Z(1), Z(x)$;
  –les formules de la forme $s \leqslant t$ où s,t sont des termes (il suffit de considérer 16 cas : $s,t \in \{0,1,x,y\}$). Les seuls cas non-triviaux sont $0 \leqslant x; x \leqslant 0; 1 \leqslant x; x \leqslant 1; x \leqslant y$.
  –et encore toutes les formules possible de la forme $P(s,t,r)$ où s,t,r sont des termes (il suffit de considérer un nombre fini de cas : $s,t,r \in \{0,1,x,y,z\}$).
–**Le cas inductif**. Si $L(f)$ et $L(g)$ sont ω-réguliers, alors $L(\neg f)$, $L(f \vee g)$, $L(\exists x f)$ sont aussi ω-réguliers.

6.*Démontrer le cas de base.* Il faut exhiber des automates (ou des expressions régulières pour les formules énumérées précédemment. On le fera au tableau pour $Z(x)$, $x \leqslant y$ et $P(x,y,z)$. Les autres cas sont plus simples et similaires.

7.*Démontrer le cas inductif* Supposons que $L(f)$ et $L(g)$ sont ω-réguliers.

–$L(\neg f) = overline L(f)$, il est ω-régulier en tant que complément d'un langage ω-régulier (résultat énoncé sans preuve en cours.)**Problème : cette construction est un peu fausse, comme elle donne un langage qui contient des mots mal formés, tels que** $+9, -899,,,++$**. Corrigez la preuve.**
–$L(f \vee g) = L(f) \cup L(g)$, il est ω-régulier en tant qu'union de deux langages ω-réguliers. **Problème : ce raisonnement marche si** f **et** g **ont les mêmes variables libres. Que faire dans le cas général.**
–$L(\exists x f)$ est une image homomorphe de $L(x)$ **Pourquoi ? Décrire le morphisme utilisé.** Donc, ce langage est aussi ω-régulier. On a utilisé un résultat suivant : une image homomorphe d'un langage ω-régulier est aussi ω-régulière. textbfdémontrer cette propriété pour les homomorphismes remplaçant une lettre par une lettre

8. *Déduire la décidabilité de la théorie* $\mathcal{T}$ *et expliquer l' algorithme de décision.*

L'algorithme est le suivant : pour une formule close f donnée construire , en appliquent la méthode ci-dessus un automate de Büchi (A) pour $L(f)$. Tester en appliquant un algo connu si le langage de (A) est vide. S'il est vide f est fausse, sinon elle est vraie

9. *Vous avez sans doute oublié que certains nombres réels admettent plus d'un développement en fraction décimale/ou binaire. Par exemple*

$$\begin{aligned} 2,5 &= 2,4999999999999999999\ldots \quad \textit{base 10} \\ 10,1 &= 10,01111111111111111111\ldots \quad \textit{base 2} \end{aligned}$$

Comment corriger la preuve pour tenir compte de ce phénomène désagréable ?

–Une solution possible : **interdire** les développements avec 999 : on définit le langage de tous les mots "jolis", c'est-a-dire qui correspondent à un vecteur de réels sans une infinité de 9 à la fin. Pour la dimension 1 ça s'écrit

$$J = (+ \vee -)(0..9)^*, ((0..9)^*(0..8))^\omega$$

Pour une dimension arbitraire c'est un peu plus difficile, mais J est toujours ω-régulier.
Ensuite, à chaque étape de la construction de l'automate pour $L(f)$ on intersecte avec J.

–Une autre possibilité : **autoriser** les séquences infinies de 9. Il faudra corriger légèrement l'automate pour $Z(x)$ qui doit accepter, par exemple $+7,9^\omega$ et l'automate pour $x \leqslant y$, qui doit accepter $x = 2,39999\ldots$ et $y = 2,40000\ldots$. **Comment ?**