École polytechnique — X2016 — INF412 Fondements de l'informatique

PC5 notée

Sujet proposé par Stéphane Graham-Lengrand et Samuel Mimram et Bruno Salvy

Cet énoncé comporte trois parties indépendantes et qui pourront être résolues dans n'importe quel ordre. Dans chaque partie, on pourra, pour répondre à une question, admettre les résultats dont on demande la démonstration aux questions *précédentes*. Il n'est pas nécessaire de traiter toutes les questions pour avoir la note maximale. Les correcteurs vous remercient d'avance d'écrire lisiblement.

1 Le théorème de Ramsey

Un *graphe* (S, A) est une paire constituée d'un ensemble S (de *sommets*) et d'un ensemble $A \subseteq S \times S$ (d'*arêtes*). Lorsque $(x, y) \in S \times S$ est une arête, on dira que x *est relié à* y. On supposera que tous les graphes que l'on considère sont

— *symétriques* : pour tous sommets x et y, si x est relié à y alors y est relié à x, et
— *reflexifs* : tout sommet est relié à lui-même.

Question 1.1. *Donnez une théorie dont les modèles sont les graphes.*

Un *sous-graphe complet* d'un graphe (S, A) est un sous-ensemble $S' \subseteq S$ tel qu'un sommet de S' est relié à tout autre sommet de S'. Sa *taille* est le cardinal de S'. On rappelle qu'une théorie est *égalitaire* lorsqu'elle contient un symbole de relation binaire « $=$ » qui est interprété par l'égalité usuelle dans les modèles.

Question 1.2. *Étant donné $n \in \mathbb{N}$, donnez une théorie égalitaire dont les modèles sont les graphes ne contenant pas de sous-graphe complet de taille n.*

De même, un *sous-graphe discret* d'un graphe (S, A) est un sous-ensemble $S' \subseteq S$ tel qu'un sommet de S' n'est relié à aucun autre sommet distinct de S'. Encore une fois, *taille* est le cardinal de S'.

Question 1.3. *Étant donné $n \in \mathbb{N}$, donnez une théorie égalitaire dont les modèles sont les graphes ne contenant pas de sous-graphe discret de taille n.*

Notre objectif est de démontrer les deux variantes suivantes du théorème de Ramsey :

Théorème 1 (Théorème de Ramsey finitaire)**.** *Pour tout $n \in \mathbb{N}$, il existe $r \in \mathbb{N}$ tel que tout graphe avec plus de r sommets admet*

— *un sous-graphe complet de taille n, ou*
— *un sous-graphe discret de taille n.*

Théorème 2 (Théorème de Ramsey infinitaire)**.** *Tout graphe avec un nombre infini de sommets admet*

— *un sous-graphe complet infini, ou*
— *un sous-graphe discret infini.*

Question 1.4. *Dans le théorème de Ramsey finitaire, montrez que pour $n = 3$, on a nécessairement $r \geq 5$.*

Nous allons commencer par montrer que la variante finitaire peut se déduire de la variante infinitaire :

Question 1.5. *En supposant que le théorème de Ramsey infinitaire est vrai, montrez le théorème de Ramsey finitaire. On pourra raisonner par l'absurde.*

Question 1.6. *Montrez le théorème de Ramsey infinitaire. Étant donné un graphe (S, A) avec un nombre infini de sommets, on pourra distinguer deux cas selon que l'assertion suivante est vraie ou non dans le graphe :* il existe $T \subseteq S$ infini tel que pour tout $x \in T$ il existe $U_x \subseteq T$ fini tel que x est relié à tout élément de $T \setminus U_x$.

2 Temps de calcul des machines de Turing

On mesure le temps qu'il faut à une machine de Turing pour accepter ou rejeter un mot par le nombre de transitions qu'elle effectue avant d'atteindre un état d'acceptation ou de rejet. Pour une machine qui termine sur toutes ses entrées de taille n, $T(n)$ note le maximum des temps de calcul sur toutes ces entrées. L'objectif de cet exercice est d'étudier sur un exemple simple la façon dont la croissance de cette quantité avec n (la complexité) peut dépendre du nombre de rubans de la machine.

L'exemple est le suivant : on considère l'alphabet $\Sigma = \{0, 1, 2\}$ et le langage

$$\mathcal{L} \quad := \quad \{m \in \Sigma^\star \mid \exists w \in \{0,1\}^\star, \exists a \in \{2\}^\star, m = waw \text{ et } |a| = |w|\},$$

où $|w|$ représente la longueur d'un mot w ($\Sigma^\star$ dénotant, comme d'habitude, l'ensemble des mots finis sur l'alphabet Σ). Par exemple, 012201 est dans le langage $\mathcal{L}$, mais 01220, 0220 ou 0202 n'y sont pas.

Question 2.1. *Décrire informellement une machine de Turing à deux rubans reconnaissant les mots de $\mathcal{L}$ en un nombre maximum de transitions $T(n)$ linéaire en leur longueur n.*

Question 2.2. *Décrire informellement une machine de Turing à un seul ruban reconnaissant les mots de $\mathcal{L}$ en un nombre maximum de transitions $T(n)$ quadratique en leur longueur n.*

La suite de l'exercice vise à montrer qu'aucune machine de Turing à un ruban ne peut reconnaître $\mathcal{L}$ avec $T(n)$ inférieur à $O(n^2)$ transitions. (Autrement dit, il existe $c > 0$ tel que pour tout $n \geq 1$, $T(n) > cn^2$).

On suppose pour simplifier que les positions sur le ruban sont indexées par les entiers naturels, que le mot d'entrée occupe les positions 1 à n, et on se restreint aux machines qui n'écrivent jamais à gauche de leur entrée. Pour tout mot w, et tout entier $i \in \mathbb{N}$, on définit une suite d'états T_i^w : lorsqu'une transition déplace la tête de lecture de la position i à la position $i+1$ ou de la position $i+1$ à la position i, l'état q auquel elle aboutit est ajouté à cette suite. Ces suites sont toutes finies pour les mots acceptés ou rejetés.

Par exemple, pour la machine simple du polycopié p. 99, et le mot $w = 0011$ pour lequel le diagramme espace-temps est donné à l'Exemple 7.6 p. 100, les suites correspondantes sont

$$\begin{aligned}
T_1^{0011} &= (q_1, q_2, q_0),\\
T_2^{0011} &= (q_1, q_2, q_1, q_2, q_0),\\
T_3^{0011} &= (q_1, q_2, q_3),\\
T_4^{0011} &= (q_3),\\
T_5^{0011} &= (q_4),
\end{aligned}$$

et les autres sont vides.

Question 2.3. *Soient $A = a_1a_2$ et $B = b_1b_2$ (avec a_1, a_2, b_1, b_2 des mots dans $\Sigma^\star$) deux mots qui sont soit tous deux rejetés soit tous deux acceptés par une machine M et tels que les suites d'états $T^A_{|a_1|}$ et $T^B_{|b_1|}$ sont identiques. Montrer qu'alors les mots $C = a_1b_2$ et $D = b_1a_2$ sont acceptés si et seulement si A l'est.*

Question 2.4. *On suppose que la machine M reconnaît le langage $\mathcal{L}$, et soit $\mathcal{L}_{3n}$ l'ensemble des mots de $\mathcal{L}$ de longueur $3n$. Montrer que toutes les suites T_i^w pour $w \in \mathcal{L}_{3n}$ et $i \in \{n, \ldots, 2n\}$ sont distinctes.*

Question 2.5. *Montrer qu'il existe un mot w de $\mathcal{L}$ de longueur $3n$ tel que toutes les suites d'états d'indices n à $2n$ pour w ont longueur au moins $n/\log_2 Q$, où Q est le nombre d'états de la machine. Conclure.*

3 Théorie des tableaux

Cette partie concerne le calcul des prédicats égalitaire, à savoir la version du calcul des prédicats utilisée en cours et en PCs où le symbole $=$ a le statut particulier d'être toujours interprété par l'égalité sémantique. [1]

La théorie des tableaux s'y exprime avec la signature

$$\Sigma \quad = \quad (\{\mathsf{undef}\}, \{\mathsf{read}, \mathsf{write}\}, \{\mathsf{array}, \mathsf{value}, =\})$$

où read et write sont respectivement des symboles de fonction d'arité 2 et 3, array et value sont des symboles de relation d'arité 1. Nous utiliserons les abbréviations suivantes : $t \neq u$ pour $\neg(t = u)$, $a[i]$ pour $\mathsf{read}(a, i)$, qui représente le contenu de la case i du tableau a, et $a[i \leftarrow v]$ pour $\mathsf{write}(a, i, v)$, qui représente le tableau qui a le même contenu que le tableau a sauf dans la case i dont le contenu est v. [2]

L'ensemble $\mathcal{A}$rrays est l'ensemble des axiomes suivants :

$$\begin{array}{ll}
(\mathrm{A}_1) & \forall t\ \neg(\mathsf{value}(t) \wedge \mathsf{array}(t)) \\
(\mathrm{A}_2) & \forall t\ (t = \mathsf{undef} \Leftrightarrow \neg(\mathsf{value}(t) \vee \mathsf{array}(t))) \\
(\mathrm{A}_3) & \forall a \forall i\ \neg(\mathsf{array}(a[i])) \\
(\mathrm{A}_4) & \forall a \forall i\ (\mathsf{value}(a[i]) \Leftrightarrow (\mathsf{array}(a) \wedge \mathsf{value}(i))) \\
(\mathrm{A}_5) & \forall a \forall i \forall v\ \neg(\mathsf{value}(a[i \leftarrow v])) \\
(\mathrm{A}_6) & \forall a \forall i \forall v\ (\mathsf{array}(a[i \leftarrow v]) \Leftrightarrow (\mathsf{array}(a) \wedge \mathsf{value}(i) \wedge \mathsf{value}(v))) \\
(\mathrm{A}_7) & \forall a \forall i \forall j \forall v\ ((\mathsf{value}(a[i \leftarrow v][j]) \wedge (i = j)) \Rightarrow a[i \leftarrow v][j] = v) \\
(\mathrm{A}_8) & \forall a \forall i \forall j \forall v\ ((\mathsf{value}(a[i \leftarrow v][j]) \wedge (i \neq j)) \Rightarrow a[i \leftarrow v][j] = a[j])
\end{array}$$

On peut comprendre les axiomes ainsi :

— (A_1) dit que les valeurs et les tableaux sont disjoints.

— (A_2) dit que l'élément undéfini est l'unique élément qui n'est ni une valeur ni un tableau.

— (A_3)-(A_6) expriment une forme de "typage" :
$a[i]$ n'est pas un tableau, et est une valeur si et seulement si a est un tableau et i est une valeur ; et $a[i \leftarrow v]$ n'est pas une valeur, et est un tableau si et seulement si a est un tableau et i et v sont des valeurs.

— (A_7) et (A_8) décrivent ce qu'il se passe quand on écrit v dans la case i du tableau a, puis qu'on lit la valeur de la case j du tableau résultant : si $i = j$, on récupère v ; si $i \neq j$, on récupère la valeur qu'on aurait lue dans la case i du tableau a.

Question 3.1. *Indiquez parmi les 4 formules suivantes celles qui sont insatisfiables dans $\mathcal{A}$rrays. Justifiez brièvement pourquoi elle sont insatisfiables.*

$$\begin{array}{ll}
\mathsf{value}(a[i][j]) & \mathsf{array}(a[i][j \leftarrow v]) \\
\mathsf{value}(a[a[j]]) & \mathsf{array}(a[b[i] \leftarrow a])
\end{array}$$

1. C'est-à-dire qu'une formule $t = u$ est satisfaite dans une structure si et seulement si t et u sont interprétés dans son ensemble de base comme le même élément.

2. Par nature de la représentation en logique, les tableaux dans cette modélisation peuvent être vus comme *immutables*, même si à partir de la question 3.5 on verra comment s'en servir pour modéliser des programmes où les tableaux sont mutables.

Question 3.2. *Donnez un modèle de la théorie Arrays qui satisfait également et simultanément* $\exists a\ \mathsf{array}(a)$ *et* $\exists v\ \mathsf{value}(v)$. *Quel est le plus petit cardinal que l'ensemble de base d'un tel modèle peut avoir ?*

On s'intéresse à une classe de structures sur Σ bien particulières, définies ainsi : on interprète les valeurs dans $\mathbb{N}$, les tableaux comme des paires composées d'un "tag" s identifiant le tableau et d'une fonction f de $\mathbb{N}$ dans $\mathbb{N}$ qui à tout indice i associe le contenu de sa case d'indice i,[3] et les autres termes comme $\star$.

Le tag s utilisé dans l'interprétation d'un tableau permet par exemple de représenter l'adresse mémoire du tableau, et plus généralement de déterminer quand deux tableaux doivent être considérés égaux (ce que les axiomes de *Arrays* ne spécifient pas !) : il faudra non seulement que les interprétations des deux tableaux aient les mêmes contenus, mais aussi les mêmes tags. La plupart des langages de programmation considèrent par exemple que les adresses mémoire doivent être les mêmes.

L'opération critique est de déterminer, quand le tableau a est interprété par (s, f) et les valeurs i et v sont interprétées par les entiers n et m, quel tag on utilise pour interpréter $a[i \leftarrow v]$, dénoté $\mathsf{address}(s, n, m)$. Plusieurs choix sont possibles, que l'on modélise par différentes fonctions $\mathsf{address}$ de $\mathcal{T} \times \mathbb{N} \times \mathbb{N}$ vers $\mathcal{T}$, où $\mathcal{T}$ est l'ensemble des tags que l'on se donne.

Pour tout ensemble de tags $\mathcal{T}$ et toute fonction $\mathsf{address}$ de $\mathcal{T} \times \mathbb{N} \times \mathbb{N}$ vers $\mathcal{T}$, on définit la structure $\mathfrak{M}_{\mathcal{T},\mathsf{address}}$ sur Σ ainsi :

L'ensemble de base de $\mathfrak{M}$, noté $|\mathfrak{M}|$, est $\{\star\} \cup \mathbb{N} \cup \{(s, f) \mid s \in \mathcal{T}, f : \mathbb{N} \to \mathbb{N}\}$,
où $\star$ n'est ni un entier ni une paire

$\mathsf{value}^{\mathfrak{M}}(c)$ ssi $c \in \mathbb{N}$
$\mathsf{array}^{\mathfrak{M}}(c)$ ssi $c \in \mathcal{T} \times (\mathbb{N} \to \mathbb{N})$
$\mathsf{undef}^{\mathfrak{M}}$ est $\star$
$\mathsf{read}^{\mathfrak{M}}((s, f), n)$ est $f(n)$ si $n \in \mathbb{N}$ et $(s, f) \in \mathcal{T} \times (\mathbb{N} \to \mathbb{N})$, et
$\mathsf{read}^{\mathfrak{M}}(c_1, c_2)$ est $\star$ dans les autres cas,
$\mathsf{write}^{\mathfrak{M}}((s, f), n, m)$ est $(\mathsf{address}(s, n, m), f')$ si $n, m \in \mathbb{N}$ et $(s, f) \in \mathcal{T} \times (\mathbb{N} \to \mathbb{N})$,
et où $f'(n) = m$ et $f'(n') = f(n')$ si $n' \neq n$, et
$\mathsf{write}^{\mathfrak{M}}(c_1, c_2, c_3)$ est $\star$ dans les autres cas.

Question 3.3. *Quels que soient* $\mathcal{T}$ *et* $\mathsf{address}$, *montrez que* $\mathfrak{M}_{\mathcal{T},\mathsf{address}}$ *est un modèle de Arrays.*

Question 3.4.

1. *Donnez un ensemble* $\mathcal{T}_{\mathsf{ext}}$ *et une fonction* $\mathsf{address}_{\mathsf{ext}}$ *tels que* $\mathfrak{M}_{\mathcal{T}_{\mathsf{ext}},\mathsf{address}_{\mathsf{ext}}}$ *satisfait la formule suivante (appelée* axiome d'extensionalité*) :*

$$\forall a \forall a'\ ((\forall i\ (a[i] = a'[i])) \Rightarrow a = a')$$

2. *Donnez un ensemble* $\mathcal{T}_{\mathsf{prog}}$ *et une fonction* $\mathsf{address}_{\mathsf{prog}}$ *tels que* $\mathfrak{M}_{\mathcal{T}_{\mathsf{prog}},\mathsf{address}_{\mathsf{prog}}}$ *ne satisfait pas l'axiome d'extensionalité, mais satisfait les formules suivantes :*

$$\forall a \forall i \forall i' \forall v \forall v'\ ((i \neq j) \Rightarrow (a[i \leftarrow v][i' \leftarrow v'] = a[i' \leftarrow v'][i \leftarrow v]))$$
$$\forall a \forall i \forall v\ (a[i \leftarrow v][i \leftarrow a[i]] = a)$$

3. *Donnez un ensemble* $\mathcal{T}_{\mathsf{copy}}$ *et une fonction* $\mathsf{address}_{\mathsf{copy}}$ *tels que* $\mathfrak{M}_{\mathcal{T}_{\mathsf{copy}},\mathsf{address}_{\mathsf{copy}}}$ *ne satisfait pas l'axiome d'extensionalité, mais satisfait la formule suivante :*

$$\forall a \forall a' \forall i \forall i' \forall v \forall v'\ ((a[i \leftarrow v] = a'[i' \leftarrow v']) \Rightarrow (a = a' \wedge i = i' \wedge v = v'))$$

3. Pour éviter les questions de bornes de tableaux, notre classe de structures considère que les tableaux ont un nombre de case infini, ce à quoi on peut toujours se ramener en affectant une valeur par défaut à toutes les cases au-delà d'une certaine case.

On cherche maintenant à exprimer, puis à établir, qu'un programme réalise l'échange de deux cases d'un tableau. Plus précisément, le programme prend en entrée un tableau a et deux indices i et j, et renvoie un tableau où les valeurs des cases i et j ont été échangées, et toute autre case contient la même valeur que dans a.

Question 3.5. *Ecrivez une formule* $\mathsf{notswap}(a, i, j, r, k)$ *sur la signature* Σ*, de variables libres* a*,* i*,* j*,* r*,* k*, sans quantificateurs, qui exprime le fait que* k *est un indice qui montre que* r *n'est pas le tableau résultant de l'échange des cases* i *et* j *dans* a *(i.e. la valeur de* r *dans la case* k *n'est pas ce qu'elle devrait être).*

On veut maintenant montrer que le programme suivant, écrit en pseudo code, renvoie bien un tableau qui ne diffère du tableau a passé en argument que par l'échange de ses cases i et j :

```
swap(a,i,j){
  v=a[i];
  a[i]<-a[j];
  a[j]<-v;
  return a;
}
```

On exprime par la formule $\mathsf{progrun}(a_0, a_1, a_2, i, j, v, k)$ ci-dessous une exécution du programme `swap` :

$$\begin{aligned}&\mathsf{array}(a_0) \wedge \mathsf{value}(i) \wedge \mathsf{value}(j)\\ &\wedge\ v = a_0[i]\\ &\wedge\ a_1 = a_0[i \leftarrow a_0[j]]\\ &\wedge\ a_2 = a_1[j \leftarrow v]\end{aligned}$$

et on exprime par la formule Bug ci-dessous le fait que le programme `swap` puisse renvoyer une mauvaise valeur :

$$\mathsf{progrun}(a_0, a_1, a_2, i, j, v, k) \wedge \mathsf{notswap}(a_0, i, j, a_2, k)$$

Dans cette question, on rappelle qu'un *littéral* est une formule atomique ou la négation d'une formule atomique.

Question 3.6.

1. *Donnez une formule* C_1 *qui soit une conjonction de littéraux et qui soit équivalente à* $k = j \wedge \mathsf{notswap}(a_0, i, j, a_2, k)$ *dans le calcul des prédicats égalitaire.*
2. *Donnez une formule* C_2 *qui soit une conjonction de littéraux et qui soit équivalente à* $k = i \wedge k \neq j \wedge \mathsf{notswap}(a_0, i, j, a_2, k)$ *dans le calcul des prédicats égalitaire.*
3. *Donnez une formule* C_3 *qui soit une conjonction de littéraux et qui soit équivalente à* $k \neq i \wedge k \neq j \wedge \mathsf{notswap}(a_0, i, j, a_2, k)$ *dans le calcul des prédicats égalitaire.*

Pour chaque C_q parmi C_1, C_2, C_3, on note D_q la formule $\mathsf{progrun}(a_0, a_1, a_2, i, j, v, k) \wedge C_q$.

On rappelle que $\bot$ est la constante logique "faux", interprété comme le Booléen 0 quelle que soit la structure.

Question 3.7.

Supposez que $\mathcal{A}rrays \cup \{D_1\} \vdash \bot$*, que* $\mathcal{A}rrays \cup \{D_2\} \vdash \bot$*, et que* $\mathcal{A}rrays \cup \{D_3\} \vdash \bot$*.*

Justifiez pourquoi Bug *est insatisfiable dans* $\mathcal{A}rrays$*.*

Question 3.8. *Pour chaque* D_q *parmi* D_1, D_2, D_3*, montrez que* $\mathcal{A}rrays \cup \{D_q\} \vdash \bot$*.*

Déduisez-en que Bug *est insatisfiable dans* $\mathcal{A}rrays$*.*

Pour chaque D_q*, on exhibera une preuve de* $\mathcal{A}rrays \cup \{D_q\} \vdash \bot$ *selon le format de preuve suivant : on donnera une suite finie de formules* $B_1, \ldots, B_n$*, où* B_n *est la formule* $\bot$*, et pour tout* $0 < i \leq n$*, soit* $(D_q \wedge B_1 \wedge \cdots \wedge B_{i-1}) \Rightarrow B_i$ *est valide dans le calcul des prédicats égalitaire, soit* $(A \wedge B_1 \wedge \cdots \wedge B_{i-1}) \Rightarrow B_i$ *est valide dans le calcul des prédicats égalitaire, pour un axiome* A *de* $\mathcal{A}rrays$*. On indiquera à chacune de ces étapes l'axiome de* $\mathcal{A}rrays$ *utilisé.*