Examen terminal - Théorie de la complexité

THX

17 décembre 2019

La précision et la clarté de la rédaction est prise en compte dans l'évaluation. Certaines questions peuvent demander une formalisation très lourde : favorisez alors la pédagogie. Écrivez assez grand. N'hésitez pas à réutiliser des résultats vus en cours ou en TD/DM. Rédigez soigneusement.

1 Inclusion de langages rationnels

Étant donné une expression rationnelle e, $L(e)$ est le langage dénoté par l'expression rationnelle. **INCLUSION** est le problème :

- entrée : un alphabet fini Σ, deux expressions rationnelles e_1, e_2 sur Σ ;
- sortie : oui, si $L(e_1) \subseteq L(e_2)$; non sinon.

Question 1. Donner $L(ab^*)$.

$L(ab^*) := \{ab^n \mid n \in \mathbb{N}\}$.

Question 2. Montrer que **INCLUSION** est dans PSPACE.

On décide le dual de **INCLUSION** avec l'algorithme non-déterministe suivant. On construit des NFA A_1 et A_2 pour e_1 et e_2. On choisit alors de manière non-déterministe une lettre. On avance un ensemble d'états E_1 et E_2 dans A_1 et A_2. On accepte si $E_1 \cap F_1 \neq \emptyset$ et $E_2 \cap F_2 = \emptyset$.

Question 3. Montrer que **INCLUSION** est PSPACE-dur.

On réduit depuis l'universalité en temps polynomial : $tr(e) = (\Sigma^*, e)$.

2 UNIQUE SAT

Une formule φ de la logique propositionnelle est satisfaite par une unique valuation s'il existe une unique valuation ν tel que $\nu \models \varphi$ et $\nu(p) = 0$ pour toute variable propositionnelle p qui n'apparaît pas dans φ.

UNIQUE SAT est le problème :

- entrée : une formule φ de la logique propositionnelle ;
- sortie : oui, si φ est satisfaite par une unique valuation ; non, sinon.

En TD, vous avez démontré que **UNIQUE SAT** est dans Δ_2^p. On ne sait pas s'il est Δ_2^p-complet.

Question 4. Montrer que **UNIQUE SAT** est coNP-dur.

UNSAT, l'ensemble des formules non satisfiables est coNP-complet. On réduit UNSAT à **UNIQUE SAT** en temps polynomial :
$tr(\varphi(x_1, \ldots, x_n)) := (x_0 \wedge x_1 \wedge \ldots x_n) \vee (\neg x_0 \wedge \varphi(x_1, \ldots, x_n))$.
φ est non satisfiable ssi $tr(\varphi(x_1, \ldots, x_n))$ admet une unique valuation.
Si φ est non satisfiable alors la clause de droite ne peut pas être vraie et seule la valuation $x_0 := \ldots x_n := 1$ rend $tr(\varphi(x_1, \ldots, x_n))$ vraie.
Réciproquement, si $tr(\varphi(x_1, \ldots, x_n))$ admet une unique valuation, comme $x_0 := \ldots x_n := 1$ rend vraie $tr(\varphi)$, c'est que c'est la seule. Et donc impossible de rendre φ vraie. Donc φ est non satisfiable.

On pose la classe de complexité $DIF = \{L_1 \setminus L_2 \mid L_1, L_2 \in \text{NP}\}$.

Question 5. Montrer que **UNIQUE SAT** est dans DIF.

UNIQUE SAT $:= L_1 \setminus L_2$ où $L_1 := SAT$ et L_2 est l'ensemble des formules admettant au moins deux valuations distinctes.

3 Problème de pavage

On considère le problème de pavage suivant, que l'on appelle **BINARY SQUARE TILING** :

- entrée : un ensemble fini T de types de tuiles, une tuile $t \in T$ et un entier N écrit en binaire ;
- sortie : oui s'il est possible de paver le rectangle $\{0, \ldots, N-1\} \times \{0, \ldots, N-1\}$ avec des tuiles de T et avec la tuile t en $(0, 0)$; non sinon.

Question 6. Montrer que **BINARY SQUARE TILING** est dans NEXPTIME.

Il faut deviner une tuile t_{ij} pour toute coordonnée $\{0, \ldots, N-1\} \times \{0, \ldots, N-1\}$ puis vérifier les contraintes.

Question 7. Expliquer pourquoi **BINARY SQUARE TILING** est NEXPTIME-dur (on ne demande pas une preuve complétement détaillée mais les idées principales doivent figurer dans votre argumentation).

On code l'exécution d'une machine de Turing qui s'exécute en temps exponentiel dans un carré $\{0, \ldots, N-1\} \times \{0, \ldots, N-1\}$.

4 Dependence QBF

Dependence QBF (DQBF) généralise QBF (quantified binary formulas) en indiquant les dépendances entre variables. Par exemple,

$$\forall p_1, \forall p_2, \exists q_1(\{p_1\}) \exists q_2(\{p_2\})(p_1 \leftrightarrow q_1) \wedge (p_2 \leftrightarrow q_2)$$

se lit 'pour toute valeur booléenne de p_1, pour toute valeur booléenne de p_2, et il existe une valeur de q_1 qui ne dépend que de p_1 et il existe une valeur de q_2 qui ne dépend que de p_2, qui rendent $(p_1 \leftrightarrow q_1) \wedge (p_2 \leftrightarrow q_2)$ vraie'. Plus généralement, une formule de DQBF est de la forme

$$\forall p_1, \ldots, \forall p_n \exists q_1(D_1) \ldots \exists q_m(D_m) \psi(p_1, \ldots, p_n, q_1, \ldots, q_m)$$

où $p_1, \ldots, p_n, q_1, \ldots, q_m$ sont des variables propositionnelles, pour tout $i \in \{1, \ldots, m\}$, $D_i \subseteq \{p_1, \ldots, p_n\}$, et $\psi(p_1, \ldots, p_n, q_1, \ldots, q_m)$ est une formule propositionnelle sur les variables $p_1, \ldots, p_n, q_1, \ldots, q_m$.

Étant donné un ensemble fini V de variables propositionnelles, on note $val(V)$ l'ensemble des valuations sur V. Étant donné une valuation $\nu \in val(V)$, on note $\nu_{|V'}$ la restriction de la valuation aux variables de V'. Par exemple, la restriction de la valuation $[p_1 := 0, p_2 := 1, p_3 := 1]$ à $\{p_1, p_3\}$ est $[p_1 := 0, p_3 := 1]$.

Une formule de DQBF $\forall p_1, \ldots, \forall p_n \exists q_1(D_1) \ldots \exists q_m(D_m) \psi(p_1, \ldots, p_n, q_1, \ldots, q_m)$ est *vraie* ssi il existe une famille de fonctions $(f_i)_{i \in \{1, \ldots, m\}}$ avec $f_i : val(D_i) \to \{0, 1\}$, tel que pour tout valuation ν sur les propositions $\{p_1, \ldots, p_n\}$, on a

$$\nu \cup [q_1 := f_1(\nu_{|D_1}), \ldots, q_m := f_m(\nu_{|D_m})] \models \psi.$$

On considère le problème **TDQBF** (pour *True Dependence Quantified Binary Formulas*) :

- entrée : une formule de DQBF φ ;
- sortie : oui si φ est vraie ; non sinon.

Question 8. Donner une réduction en temps polynomial de **TQBF** à **TDQBF**.

A toute **TQBF**-instance $\exists q_1 \forall p_1 \exists q_n \forall p_n \psi$ on associe l'instance de **TDQBF** suivante :

$$\forall p_1 \ldots \forall p_n \exists q_1(D_1) \exists q_n(D_n)\ \psi$$

où $D_k := \{q_1, \ldots, q_{k-1}\}$.

Question 9. Montrer que **TDQBF** est dans NEXPTIME.

On devine les fonctions s_i.

On considère une instance (T, t, N) de **BINARY SQUARE TILING** de l'exercice précédent. On utilise le jeu coopératif suivant avec deux joueurs nommés a et b et un maître du jeu :

1. le maître du jeu attribue de manière privée une case c_a de $\{0, \ldots, N-1\}^2$ au joueur a ;
2. le maître du jeu attribue de manière privée une case c_b de $\{0, \ldots, N-1\}^2$ au joueur b ;
3. le premier joueur a choisit secrètement une tuile $t_a \in T$;
4. le deuxième joueur b choisit secrètement une tuile $t_b \in T$;
5. le maître du jeu recueille t_a et t_b puis déclare les deux joueurs gagnants exactement lorsque les conditions suivantes sont satisfaites :
   (a) si c_a (resp. c_b) est la case $(0, 0)$, alors t_a (resp. t_b) est la tuile t qui doit être en $(0, 0)$;
   (b) si les cases c_a, c_b sont adjacentes horizontalement (resp. verticalement), alors les tuiles t_a et t_b respectent la contrainte de couleur horizontale (resp. verticale).

Question 10. En s'inspirant du jeu ci-dessus, montrer que **TDQBF** est NEXPTIME-dur (on ne demande pas une preuve complétement détaillée mais les idées principales doivent figurer dans votre argumentation).

A une instance de **BINARY SQUARE TILING**, on construit la formule DQBF suivante :

$$\forall \vec{a} \forall \vec{b} \exists \vec{\alpha}(\vec{a}) \exists \vec{\beta}(\vec{b}) \psi(\vec{a}, \vec{b}, \vec{\alpha}, \vec{\beta})$$

où $\vec{a}$ correspond aux coordonnées d'une case de $\{0, \ldots, N-1\} \times \{0, \ldots, N-1\}$, pareil pour $\vec{b}$, $\vec{\alpha}$ est la représentation binaire d'une tuile, pareil pour $\vec{\beta}$.
La formule booléenne ψ vérifie les contraintes.