Calculabilité

TP10

Y. Deville
C-H. Bertrand Van Ouytsel - V. Coppé - A. Gerniers
N. Golenvaux - M. Parmentier

Mai 2021

Questions du test

Les affirmations suivantes sont-elles vraies ?

1. Si un problème A peut être réduit à un problème B et que A est calculable, alors B est calculable.

   Réponse : Faux
   $A \leq B$ signifie si B est calculable, alors on peut calculer A.
   Par contre, la calculabilité de A ne dit rien sur celle de B.

2. Soient deux ensembles $A, B \subseteq \mathbb{N}$. Si $A \leq_a B$ et si B est récursif, alors A est récursif.

   Réponse : Vrai
   Définition de $\leq_a$.

3. Soient deux ensembles $A, B \subseteq \mathbb{N}$. Si $A \leq_a B$ et si B n'est pas récursif, alors A n'est pas récursif.

   Réponse : Faux
   Contre-exemple : $K_{\text{BLOOP}} \leq_a K$

Questions du test

Les affirmations suivantes sont-elles vraies ?

4. Soient deux ensembles $A, B \subseteq \mathbb{N}$. Si A est récursif, alors $A \leq_f B$.

   Réponse : Vrai
   $A \leq_f B$ est vrai dans le cas où si on peut décider B, on peut aussi décider A. Or A est déjà décidable, donc A est réductible à n'importe quel B.

5. Soient deux ensembles $A, B \subseteq \mathbb{N}$. Alors $A \leq_f B \Leftrightarrow \bar{A} \leq_f \bar{B}$.

   Réponse : Vrai
   S'il existe une fonction totale calculable telle que $a \in A \Leftrightarrow f(a) \in B$, cette même fonction implique que $a \notin A \Leftrightarrow f(a) \notin B$, et donc $a \in \bar{A} \Leftrightarrow f(a) \in \bar{B}$.

Questions du test

Les affirmations suivantes sont-elles vraies ?

6. Soit C une classe de problèmes. Si on trouve un algorithme permettant de décider un des problèmes $A \in C$, alors il est possible de décider n'importe quel autre problème dans C.

   Réponse : Faux
   Pour que cette affirmation soit vraie, il faut que A soit C-complet, i.e. que tout autre problème de C puisse être réduit à A :

   $$\forall B \in C : B \leq A$$

   A doit donc être le problème "le plus compliqué" de la classe C.

Questions du test

Les affirmations suivantes sont-elles vraies ?

7. Soient $A, B \subseteq \mathbb{N}$. Si $A \leq_a B$ et qu'on connait la complexité de B, alors on peut en déduire la complexité de A.

   Réponse : Faux
   La réduction algorithmique permet de déduire la récursivité de A en connaissant celle de B, mais ne donne aucune information sur la complexité (car on peut utiliser autant de fois que l'on veut la récursivité de B pour calculer A).

8. Soient $A, B \subseteq \mathbb{N}$. Si $A \leq_f B$, qu'on connait la complexité de B et qu'on a accès à un programme P_f qui calcule f, alors on peut en déduire quelque chose au sujet de la complexité de A.

   Réponse : Vrai
   Par définition de $\leq_f$, on utilise un unique test $f(a) \in B$ pour décider $a \in A$. On peut donc déduire la complexité de A à partir de celle de la fonction de décision de B et de celle de f.

Questions du test

Les affirmations suivantes sont-elles vraies ?

9. Étant donné que les machines de Turing sont un modèle complet de calculabilité et un modèle de complexité (i.e. on peut calculer précisément la complexité d'une machine de Turing), tout problème peut se calculer sur une machine de Turing avec la même complexité qu'en Python.

   Réponse : Faux
   Tous les modèles de complexité ont des complexités reliées de manière *polynomiale*. Une MT peut donc s'exécuter avec une complexité supérieure à l'équivalent en Python (mais cette exécution restera toujours pratiquement faisable si c'est le cas avec Python).

Questions du test

Les affirmations suivantes sont-elles vraies ?

10. La classe NP (pour "Non Polynomial") est définie comme l'ensemble des problèmes ne pouvant pas être résolus par des programmes de complexité polynomiale.

    Réponse : Faux
    La classe NP (pour "*Non-déterministe* Polynomial") est définie comme l'ensemble des problèmes pouvant être résolus par des programmes *non-déterministes* de complexité *polynomiale* :

$$NP = \bigcup_{i \geq 0} NTIME(n^i)$$

Question 1 du TP

Algorithmic versus functional reduction.

1. Prove that for every set A, $A \leq_a \bar{A}$.
2. Prove that for all sets A, B, if $A \leq_f B$ and B is recursively enumerable, then A is recursively enumerable.
3. Show an example for which the relation $A \leq_f \bar{A}$ does not hold.
4. Show an example for which the following relation does not hold : if $A \leq_a B$ and B is recursively enumerable, then A is recursively enumerable.

Question 1 du TP

Algorithmic versus functional reduction.

1. Prove that for every set A, $A \leq_a \bar{A}$.

   Réponse : Soit $\bar{A}$ récursif. Alors il existe un programme $P_{\bar{A}}(a)$ qui calcule :

   $$f_{\bar{A}}(a) = \begin{cases} 1 & \text{si } a \in \bar{A} \\ 0 & \text{sinon} \end{cases}$$

   À partir de ce programme, on construit :

   $$P_A(a) \equiv \texttt{return } 1 - P_{\bar{A}}(a)$$

   $\Longrightarrow f_A$ est total calculable, donc A est récursif
   $\Longrightarrow A \leq_a \bar{A}$

Question 1 du TP

Algorithmic versus functional reduction.

2. Prove that for all sets A, B, if $A \leq_f B$ and B is recursively enumerable, then A is recursively enumerable.

   Réponse : Soit B récursivement énumérable. Alors il existe un programme P_B qui calcule :

$$f_B(b) = \begin{cases} 1 & \text{si } b \in B \\ \perp & \text{sinon} \end{cases}$$

   Si $A \leq_f B$, alors il existe une fonction f totale calculable t.q. $a \in A \Leftrightarrow f(a) \in B$. Cette fonction peut être calculée par un programme P_f. On peut construire :

$$P_A(a) \equiv \left[\begin{array}{l} b = P_f(a) \\ \texttt{return } P_B(b) \end{array} \right.$$

   $\Longrightarrow A$ est récursivement énumérable.

Question 1 du TP

Algorithmic versus functional reduction.

3. Show an example for which the relation $A \leq_f \bar{A}$ does not hold.

   Réponse : $\bar{K} \not\leq_f K$
   En effet, K est récursivement énumérable. Si $\bar{K} \leq_f K$, alors $\bar{K}$ serait aussi récursivement énumérable, ce qui impliquerait que K soit récursif.

Question 1 du TP

Algorithmic versus functional reduction.

3. Show an example for which the following relation does not hold : if $A \leq_a B$ and B is recursively enumerable, then A is recursively enumerable.

   Réponse : Par définition, $\bar{K} \leq_a K$. K est récursivement énumérable, mais pas $\bar{K}$.

Question 2 du TP

Prove the following relations.

1. Does $A \leq_a B$ imply $A \leq_f B$?
2. Does $A \leq_f B$ imply $A \leq_a B$?

Question 2 du TP

Prove the following relations.

1. Does $A \leq_a B$ imply $A \leq_f B$?

   Réponse : Faux
   Par définition, $\bar{K} \leq_a K$. On sait déjà que $\bar{K} \leq_f K$ est faux (voir exercice 1.3).

Question 2 du TP

Prove the following relations.

2. Does $A \leq_f B$ imply $A \leq_a B$?

   Réponse : Vrai
   Supposons $A \leq_f B$. Alors il existe une fonction f totale calculable tel que $a \in A \Leftrightarrow f(a) \in B$. Si on suppose B récursif, alors on peut déterminer si $f(a) \in B$ ou non, ce qui revient à décider $a \in A$. A est donc aussi récursif.
   $\Longrightarrow A \leq_a B$.

• On peut écrire le programme

$P_A(a) \equiv$ [return $P_B(P_f(a))$

Question 3 du TP

Let $re = \{A \mid A \text{ is recursively enumerable}\}$.

1. State the conditions to prove that HALT is re-complete wrt $\leq_a$.
2. Prove the conditions.

Question 3 du TP

Let $re = \{A \mid A \text{ is recursively enumerable}\}$.

1. State the conditions to prove that $HALT$ is re-complete wrt $\leq_a$. **Réponse :**
   i. $HALT \in re$
   ii. $\forall B \in re, B \leq_a HALT$

$HALT(m, x) \equiv \left[\begin{array}{l} P_m(x) \\ \text{return } 1 \end{array} \right.$

2. Prove the conditions. **Réponse :**
   i. $P_{HALT_{re}}(n, x) \equiv \left[\begin{array}{l} P_n(x) \\ \texttt{return } 1 \end{array} \right.$
   ii. Si $B \in re$, $\exists P_{B_{re}}$ qui calcule $f_{B_{re}}(b) = \left\{ \begin{array}{ll} 1 & \text{si } b \in B \\ \perp & \text{sinon} \end{array} \right.$

   Si on suppose $HALT$ récursif, alors on peut calculer la fonction de décision de B :

$$P_{B_{rec}}(b) \equiv \left[\begin{array}{l} \texttt{if } halt(B_{re}, b) \texttt{ return } 1 \\ \texttt{else return } 0 \end{array} \right.$$

   ce qui impliquerait B récursif. $\Longrightarrow B \leq_a HALT$

Question 4 du TP

Does the class $DTIME(n^2)$ depend of your calculus model (e.g. Java language) ? And what about the class P ? Justify your answers.

Question 4 du TP

Does the class $DTIME(n^2)$ depend of your calculus model (e.g. Java language) ?

Réponse :
La classe $DTIME(n^2)$ est spécifique à un modèle de calculabilité :

- Définie comme l'ensemble des problèmes pouvant être résolus par un programme Java ayant une complexité $\mathcal{O}(n^2)$.
- Les modèles de complexités sont liés entre eux par un facteur polynomial.
  Pour les mêmes problèmes, on aura donc une complexité $\mathcal{O}(n^i)$ (avec potentiellement $i \neq 2$) avec une machine de Turing par exemple.

Question 4 du TP

And what about the class P ?

Réponse :
Tous les modèles de complexité étant liés entre eux par un facteur polynomial, tout problème résolu en $\mathcal{O}(n^i)$ avec Java sera forcément résolu en $\mathcal{O}(n^j)$ par un autre modèle de complexité.

La définition de P demeure donc inchangée :

$$P = \bigcup_{i \geq 0} DTIME(n^i)$$

Question 5 du TP

Let A be a recursive set and f a function from $\mathbb{N}$ to $\mathbb{N}$. Is the following implication true ? Explain.

$$A \in NTIME(f) \Rightarrow \exists c > 1 : A \in DTIME(c^f)$$

Question 5 du TP

Let A be a recursive set and f a function from $\mathbb{N}$ to $\mathbb{N}$. Is the following implication true ? Explain.

$$A \in NTIME(f) \Rightarrow \exists c > 1 : A \in DTIME(c^f)$$

Réponse : Vrai

On peut représenter l'ensemble des exécutions possibles d'un programme non-déterministe par un arbre d'exécution, où chaque nœud représente un choix d'exécution.

Si une exécution se fait en $\mathcal{O}(f)$, parcourir cet arbre (i.e. simuler l'ensemble des exécutions possibles) se fait avec un facteur exponentiel. On est donc en $\mathcal{O}(c^f)$.

Question 6 du TP

Let A be a recursive set and f a function from $\mathbb{N}$ to $\mathbb{N}$. Is the following implication true ? Explain.

$$A \in DTIME(f) \Rightarrow A \in DSPACE(f)$$

Question 6 du TP

Let A be a recursive set and f a total function from $\mathbb{N}$ to $\mathbb{N}$. Is the following implication true ? Explain.

$$A \in DTIME(f) \Rightarrow A \in DSPACE(f)$$

Réponse : Vrai

En effet, un programme décidant A peut utiliser au plus 1 emplacement en mémoire par instruction.

La complexité spatiale est donc bornée par la complexité temporelle.