université de BORDEAUX	**Année Universitaire 2020–2021** Master CSI, UE 4TCY701U, Théorie de la complexité. Jeudi 12 novembre 2020. 9h30–12h30. Durée : 3h. **Documents non autorisés.**	Collège Sciences et Technologies

Toute réponse non justifiée n'apportera pas de point. Le barème est indicatif.
Vous pouvez utiliser une question précédente même si elle n'a pas été traitée.

Exercice 1 : Questions d'application du cours — 3 Points

Répondre aux questions ci-dessous en **justifiant précisément** (une réponse qui n'est pas correctement justifiée **n'apporte pas de point**) :

1) Le problème **42-Somewhere** suivant est-il décidable ?
Entrée : Le code d'une machine de Turing M.
Question : La machine M accepte-t-elle tous les mots contenant la chaîne 101010 ?

2) Est-il vrai que tout problème de la classe **P** se réduit de façon polynomiale au problème de l'arrêt sur le mot vide ?

3) Le problème **Arrêt universel borné** suivant est-il décidable ?
Entrée : Une machine de Turing M.
Question : Existe-t-il un entier k tel que la machine M s'arrête sur toute entrée en au plus k étapes ?

Exercice 2 : Variations autour de 3SAT — 6 Points

On s'intéresse à trois variations du problème **3SAT**. On considère d'abord le problème **3SAT-FALSIFY** suivant :
Entrée : Une formule propositionnelle φ en forme 3CNF.
Question : Existe-t-il une affectation des variables de φ rendant la formule φ **fausse** ?

1) Donnez la plus petite classe de complexité que vous connaissez contenant ce problème, en **justifiant**.

On considère maintenant le problème **OIT-3SAT** suivant (OIT signifie « One in Three ») :
Entrée : Une formule propositionnelle φ en forme 3CNF.
Question : Existe-t-il une affectation des variables de φ pour laquelle, dans chaque clause $(\ell \vee \ell' \vee \ell'')$ de φ, **exactement un** des littéraux ℓ, ℓ', ℓ'' est rendu vrai par cette affectation ?

2) Montrer que le problème **OIT-3SAT** est dans la classe **NP**.

On veut montrer que **OIT-3SAT** est **NP**-complet. On propose la réduction suivante de **3SAT** à **OIT-3SAT**. Soit φ une formule en forme 3CNF. Pour chaque clause C_i de φ, de la forme $(\ell \vee \ell' \vee \ell'')$ où ℓ, ℓ', ℓ'' sont des littéraux, on ajoute 4 variables x_i, x_i', y_i, y_i' et on produit 3 clauses de la formule résultat :

$$(\ell \vee x_i \vee y_i), \qquad (\neg\ell' \vee x_i \vee x_i'), \quad \text{et} \quad (\neg\ell'' \vee y_i \vee y_i').$$

Si m est le nombre de clauses de φ et n son nombre de variables, cette réduction produit donc une nouvelle formule avec $n+4m$ variables et $3m$ clauses.

3) Montrer que cette transformation définit bien une **réduction polynomiale** de **3SAT** à **OIT-3SAT**, en justifiant votre réponse (il y a deux aspects : c'est bien une réduction, et elle est polynomiale).

4) Que peut-on déduire des questions précédentes concernant le problème **OIT-3SAT** ? Justifiez.

On considère cette fois le problème **NAE-SAT** suivant (NAE signifie « Not All Equal ») :
Entrée : Une formule propositionnelle φ en forme CNF.
Question : Existe-t-il une affectation des variables de φ pour laquelle, dans chaque clause $(\ell_1 \vee \cdots \vee \ell_k)$ de φ, **au moins un** des littéraux est rendu **vrai** par l'affectation et **au moins un** d'entre eux rendu **faux** ?

5) Montrer que le problème **NAE-SAT** est dans **NP**.

6) Montrer que le problème **NAE-SAT** est **NP**-complet par réduction à partir de **3SAT**.

Indication

La réduction attendue est simple. Elle nécessite (globalement) une seule variable additionnelle.

On considère enfin le problème **NAE-3SAT**, qui est la restriction de **NAE-SAT** aux formules 3CNF.

Entrée : Une formule propositionnelle φ en forme **3**CNF.
Question : Existe-t-il une affectation des variables de φ pour laquelle, dans chaque clause $(\ell \vee \ell' \vee \ell'')$ de φ, **au moins un** des littéraux ℓ, ℓ', ℓ'' est **vrai** et **au moins un** d'entre eux est **faux**?

7) Montrer que le problème **NAE-3SAT** est **NP**-complet.

Exercice 3 : Machines reset — 6 Points

Dans cet exercice, toutes les machines de Turing considérées fonctionnent sur une seule bande, infinie à droite seulement (les cases sont numérotées par les entiers naturels, la case la plus à gauche par 0). L'entrée des machines est écrite à partir de la case 0, et initialement la tête est sur la case 0.

Une *machine reset* est une machine de Turing à une bande lecture-écriture, infinie à droite, potentiellement non déterministe, et disposant de 2 types d'instructions :

— Type 1 : la machine remplace le symbole sous la tête, change d'état, et déplace la tête d'une case à droite.
— Type 2 : la machine remplace le symbole sous la tête, change d'état, et repositionne la tête sur la case 0 (c'est-à-dire, sur la case la plus à gauche de la bande).

Par contre, la machine n'a pas d'instruction pour déplacer la tête d'une case vers la gauche. Une telle machine peut être *non déterministe*. Avec les notations vues en cours, on a donc une fonction de transition $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\triangleright, \lhd\lhd\})$, où $\triangleright$ désigne le déplacement de la tête d'une case à droite et $\lhd\lhd$ le repositionnement de la tête en début de bande (et où, si E est un ensemble, $\mathcal{P}(E)$ désigne l'ensemble des sous-ensembles de E).

1) Écrire une machine reset effectuant la multiplication par 2 du nombre binaire donné en entrée.

2) Écrire une machine reset effectuant la division par 2 du nombre binaire donné en entrée.

3) On se donne maintenant une machine de Turing habituelle M, à une bande infinie à droite : M a des instructions déplaçant la tête de lecture d'une case vers la gauche, mais pas d'instruction permettant de ramener directement la tête de lecture en position 0. Construire une machine *reset* M_{reset} qui simule M, c'est-à-dire telle que pour tout mot d'entrée x, M accepte x si et seulement si M_{reset} accepte x.

Indication. Il s'agit d'expliquer comment simuler une instruction de M qui déplace la tête d'une case à gauche. Vous pouvez ici vous aider du non-déterminisme. Vous pouvez aussi marquer des cases, par exemple en écrivant un couple $(b, \#)$ là où la machine d'origine écrit b. Vous pouvez enfin utiliser plusieurs types de marques.

4) Le problème suivant est-il décidable? Justifier précisément la réponse.
ENTRÉE : Le code d'une machine reset.
QUESTION : Cette machine reset s'arrête-t-elle sur le mot vide?

5) Peut-on simuler une machine reset par une machine de Turing habituelle? Justifier la réponse.

On considère maintenant les machines reset déterministes. Une telle machine est donnée par une fonction de transition $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\triangleright, \lhd\lhd\}$.

6) Écrire une machine reset déterministe effectuant la division par 2 du nombre binaire donné en entrée.

7) Expliquer comment simuler une machine de Turing à une bande habituelle par une machine reset déterministe.

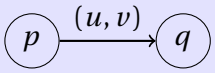
8) Montrer que si un problème est dans la classe **P**, il peut être décidé par une machine reset déterministe effectuant un nombre polynomial de pas de calcul par rapport à la taille de l'entrée.

Exercice 4 : Transducteurs — 8 Points

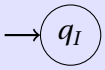
On considère un alphabet fini A ayant au moins 2 lettres. On appelle ***transducteur*** un tuple $\mathcal{T} = (Q, q_I, F, \delta)$, où Q est un ensemble fini d'états, $q_I \in Q$ un état initial, $F \subseteq Q$ un ensemble d'états finaux et $\delta \subseteq Q \times A^* \times A^* \times Q$ un ensemble fini de transitions.

Remarque

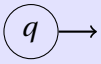
Si (p, u, v, q) est une transition, on la notera aussi $p \xrightarrow{(u,v)} q$, et on la dessinera de la façon suivante :



Vous pourrez ainsi dessiner des transducteurs plutôt que de donner la liste de leurs transitions. Dans ce cas, l'état initial devra être indiqué par une flèche entrante :



et les états finaux par une flèche sortante, de la façon suivante (où q est final) :



Un transducteur $\mathcal{T} = (Q, q_I, F, \delta)$ ***accepte*** des paires de mots $(u, v) \in A^* \times A^*$. Une telle paire $(u, v) \in A^* \times A^*$ est ***acceptée par*** $\mathcal{T}$ quand il existe une séquence de la forme suivante :

$$q_1 \xrightarrow{(x_1,y_1)} q_2 \xrightarrow{(x_2,y_2)} \cdots q_{n-1} \xrightarrow{(x_{n-1},y_{n-1})} q_n \quad \text{avec } n \geqslant 1 \text{ et,}$$

— $q_1, \ldots, q_n \in Q$ sont des états tels que $q_1 = q_I$ et $q_n \in F$.

— Pour tout i, (q_i, x_i, y_i, q_{i+1}) est une transition de $\mathcal{T}$ (c'est-à-dire appartenant à δ).

— $u = x_1 \cdots x_{n-1}$ et $v = y_1 \cdots y_{n-1}$.

Si $\mathcal{T}$ est un transducteur, on notera $L(\mathcal{T})$ l'ensemble des paires de mots (u, v) acceptées par $\mathcal{T}$, et on dira que $L(\mathcal{T})$ est le ***langage accepté*** par $\mathcal{T}$.

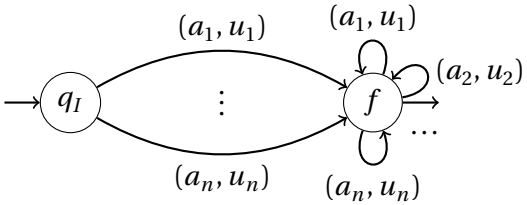
1) Montrer que le problème suivant est indécidable.

Identité non vide

Entrée : Un transducteur $T = (Q, q_I, F, \delta)$.
Question : Existe-t-il un mot $u \in A^*$ **non vide** tel que la paire (u, u) est acceptée par T ?

2) Même question pour le problème dans lequel on supprime la restriction que le mot u est non vide.

Soit un entier $n > 0$. On considère l'alphabet $I_n = \{a_1, a_2, \ldots, a_n\}$, et on suppose, quitte à renommer A, que $A \cap I_n = \varnothing$ (on rappelle que A est l'alphabet fixé au départ). À partir d'un ensemble fini de n mots $u_1, \ldots, u_n$ sur l'alphabet A, on construit le transducteur $\mathcal{T}_u$ suivant sur l'alphabet $A \cup I_n$. Il a deux états : q_I (qui est l'état initial) et f (qui est final), et $2n$ transitions.



Autrement dit, il y a une transition $q_I \xrightarrow{(a_i,u_i)} f$ pour tout i, et une boucle $f \xrightarrow{(a_i,u_i)} f$ également pour tout i. On rappelle que a_i est une lettre (la i^{e} de l'alphabet I_n), et que u_i est un mot sur l'alphabet A (l'alphabet du transducteur $\mathcal{T}_u$ est donc $A \cup I_n$).

3) Montrer que l'ensemble des couples (v, w) qui **ne sont pas dans le langage** $L(\mathcal{T}_u)$, c'est-à-dire qui **ne sont pas acceptés par ce transducteur**, est accepté par un autre transducteur.

Indication

Vous pouvez énumérer les raisons pour lesquelles un couple n'est pas accepté par $\mathcal{T}_u$, et pour chacune d'elles, construire un transducteur. Il faudra ensuite combiner les transducteurs obtenus.

4) Montrer que le problème **INTERSECTION-TRANSDUCTEURS** suivant est indécidable :

Entrée : Deux transducteurs $\mathcal{T}_1$ et $\mathcal{T}_2$.
Question : Existe-t-il une paire de mots acceptée à la fois par $\mathcal{T}_1$ et par $\mathcal{T}_2$?
Autrement dit, a-t-on $L(\mathcal{T}_1) \cap L(\mathcal{T}_2) \neq \varnothing$?

Indication

Vous pouvez utiliser des transducteurs similaires à celui de la question **3)**, mais la question **3)** elle-même n'est pas utile pour cette question.

5) Montrer que ce problème reste indécidable si l'alphabet des transducteurs comporte **deux lettres**.

Une précision...

Vous devez donner **deux** justifications (qui sont indépendantes) :

— qu'on peut utiliser un alphabet à deux lettres sur la **première composante** des couples de mots,
— qu'on peut utiliser un alphabet à deux lettres sur la **seconde composante** des couples de mots.

6) En utilisant la question **3)**, montrer que le problème **INCLUSION-TRANSDUCTEURS** suivant est indécidable :

Entrée : Deux transducteurs $\mathcal{T}_1$ et $\mathcal{T}_2$.
Question : Existe-t-il une paire de mots acceptée par $\mathcal{T}_1$ mais pas par $\mathcal{T}_2$?
Autrement dit, a-t-on $L(\mathcal{T}_1) \not\subseteq L(\mathcal{T}_2)$?

7) Montrer que le problème **UNIVERSALITÉ** suivant est indécidable :

Entrée : Un transducteur $\mathcal{T}$ sur un alphabet A.
Question : Tout couple de mots de $A^* \times A^*$ est-il accepté par $\mathcal{T}$?

8) Montrer que le problème **REJET-FINI** suivant est indécidable :

Entrée : Un transducteur $\mathcal{T}$ sur un alphabet A.
Question : Le transducteur $\mathcal{T}$ accepte-t-il tous les couples de mots de $A^* \times A^*$ sauf un nombre fini ?