Fondements de l'informatique. Examen
Durée: 3h

Sujet proposé par Olivier Bournez
Version 7
(corrigé)

Les 5 parties sont indépendantes, et peuvent être traitées dans un ordre quelconque. On pourra admettre le résultat d'une question pour passer aux questions suivantes. On pourra utiliser tous les résultats et les théorèmes démontrés ou énoncés en cours ou en petite classe ou dans le polycopié sans chercher à les redémontrer, sauf si cela est explicitement demandé.

Il est possible d'avoir la note maximale sans répondre à toutes les questions. La difficulté des questions n'est pas une fonction linéaire ni croissante de leur numérotation.

La ***qualité et clarté de votre argumentation et de votre rédaction*** *sera une partie importante de votre évaluation.*

Il peut être clair pour vous que certaines notions impliquent d'autres notions, mais à une question du type "le problème est-il A ? est-il B ? " ***il est demandé de répondre aux deux questions****, soit en répondant à chacune, soit en rendant explicite que vous savez que A implique B ou le contraire pour avoir la note maximale, possiblement une fois pour toute dans votre copie.*

On note $|S|$ pour le nombre d'éléments de l'ensemble (fini) S.

1 Décidabilité

On considère des machines de Turing qui travaillent sur un alphabet suffisant pour couvrir tous les symboles utilisés dans un traitement de texte.

Question 1. *Parmi les problèmes de décision suivants, lesquelles sont décidables ? Récursivement énumérable ? Dans NP, NP-complet, dans P ? Justifier votre réponse.*

1. *Déterminer si un sujet d'examen contient le mot "machine de Turing".*
2. *Déterminer si une machine de Turing produit sur l'entrée correspondant au mot vide un sujet d'examen qui contient le mot "machine de Turing".*

Solution : Etre polynomial implique être NP, qui implique être décidable, qui implique être récursivement énumérable.

Le problème 1. est dans P, car il suffit de parcourir le sujet.

Le problème 2. Est récursivement énumérable, car il suffit de simuler la machine de Turing, et de s'arrêter dès que l'événement se produit pour semi-décider le problème. Il n'est pas décidable car le langage universel L_u s'y réduit. En effet, soit M^+ la machine de Turing qui produit ce sujet d'examen. Cette machine existe car un sujet d'examen est un texte fini. Etant donnés une machine de Turing M et un mot w, on peut construire la machine M_w qui simule M sur w et si M accepte w simule M^+. M accepte w si et seulement si M_w produit un sujet qui contient le mot "machine de Turing", et comme on peut produire effectivement M_w à partir du code de M et de w, cela constitue une réduction du problème universel vers ce problème.

Par conséquent, le problème 2 n'est ni dans P, ni dans NP car indécidable. □

2 Logique et décidabilité

Question 2. *On considère le problème $SAT_{propositionnel}$ de décision suivant :*

— ***Donnée**: Une formule F du calcul propositionnel.*

— ***Réponse**: F est-elle satisfiable.*

Ce problème est-il décidable ? Récursivement énumérable ? Dans NP, NP-complet, dans P ? Justifier votre réponse.

Solution : Le problème est NP-complet. Voir le cours. (SAT ou $3-SAT$ en sont des cas particuliers, et le problème est clairement dans NP, car la donnée d'une valeur des variables de F constitue un certificat vérifiable en temps polynomial). La question de savoir s'il est dans P est par conséquent la question ouverte $P =^? NP$. Un problème de NP est décidable, et donc aussi récursivement énumérable. □

Le but est d'étudier la difficulté du problème similaire $SAT_{\text{prédicat}}$ pour la logique du premier ordre [1].

On appelle *machine de Turing à demi-ruban infini*, une machine qui n'écrit jamais aucun symbole à gauche de son entrée : la partie à gauche de l'entrée reste donc constituée à jamais uniquement de symboles de blanc.

Question 3. *Montrer que le problème de décision suivant est indécidable : Etant donnée M une machine de turing à demi-ruban infini, décider si M accepte le mot vide.*

Solution : On peut réduire le problème L_{univ} à cette question : étant donnée une machine de Turing M et un mot w, on considère la machine de Turing à demi-ruban infini M_w qui commence par écrire $\#w$, où $\#$ est un nouveau symbole qui n'est pas dans l'alphabet de M, et qui simule ensuite M : à chaque fois que M tente d'aller/écrire à gauche (ce qu'on peut détecter car elle lit le symbole $\#$), M_w remplace le contenu de son ruban de la forme $\# < contenu >$ par $\#B < contenu >$, puis reprend la simulation de M. En d'autres termes M_w simule M, mais décale le ruban vers la droite si besoin, de telle sorte que son propre ruban reste toujours à droite du marqueur $\#$.

On aura M accepte w si et seulement si M_w accepte le mot vide. Par ailleurs, le programme de M_w s'obtient facilement (de façon calculable) à partir du programme de M et donc c'est bien une réduction. □

Etant donné un mot $w = a_1 a_2 \dots a_n$ sur l'alphabet Σ, on notera $\overline{w}$ pour le mot w renversé : $\overline{w} = a_n a_{n-1} \dots a_2 a_1$. Pour a un symbole fixé quelconque de Σ, on note a^T pour le mot de longueur T dont toutes les lettres sont le symbole a. En particulier, $a^0 = \epsilon$.

Etant donnée une machine de Turing M, on considère une signature Σ_M constituée du symbole de constante ϵ, d'un symbole de fonction a d'arité 1 pour chaque lettre a de Σ, et d'un symbole f_q d'arité 2 pour chaque état interne $q \in Q$ de M.

Une telle signature Σ_M possède un modèle $\mathfrak{M}_M$ particulier : son ensemble de base est constitué des mots sur l'alphabet Σ (et donc les variables x s'interprètent comme des mots sur l'alphabet Σ), ϵ s'interprète comme le mot vide, $a(w)$ est défini comme le mot aw pour tout mot w, et $f_q(x, y)$ est défini comme vrai si et seulement si M sur l'entrée vide atteint une configuration où elle est dans l'état q, et où le contenu du ruban est $\overline{x}y$ avec la tête de lecture en face de la première lettre de y.

Bien entendu, Σ_M peut posséder d'autres modèles. Mais le théorème de complétude permet d'affirmer qu'une formule F du calcul des prédicats (logique du premier ordre) est prouvable si et seulement si elle est vraie dans tout modèle, c'est-à-dire quelle que soit l'interprétation des symboles de constantes, fonctions et relations.

1. Rappelons qu'on dit qu'une formule close est satisfiable si et seulement si elle possède un modèle.

Question 4. *Soit M une machine de Turing à demi-ruban infini. Montrer que l'on peut construire une formule close F_M qui est prouvable (vraie dans tout modèle) si et seulement si la machine de Turing à demi-ruban infini M accepte le mot vide.*

Solution : On peut écrire que la machine part avec le mot vide par $F_{q_0}(\epsilon, \epsilon)$. On peut décrire le programme de M de la façon suivante :

— Si le programme de M dit quand on est dans l'état q et qu'on lit un a, d'aller dans l'état r, et d'écrire un b et d'aller à droite :

$$\forall x \forall y (f_q(x, a(y)) \Rightarrow f_r(b(x), y))$$

— Si le programme de M dit quand on est dans l'état q et qu'on lit un a, d'aller dans l'état r, et d'écrire un b , et d'aller à gauche : on ajoute

$$\forall x \forall y (f_q(c(x), a'(y)) \Rightarrow f_r(x, c(b(y)))$$

pour chaque symbole $c \in \Sigma$.
On remarquera qu'il est important dans ce cas que la machine soit à semi-ruban infini, de telle sorte qu'il y ait bien un symbole à gauche.

— On doit couvrir aussi des variantes des formules précédentes pour cas de la lecture d'un blanc (d'une case visitée pour la première fois).
pour un déplacement à droite :

$$\forall x \forall y (f_q(x, \epsilon) \Rightarrow f_r(b(x), \epsilon)).$$

pour un déplacement à gauche :

$$\forall x \forall y (f_q(c(x), \epsilon) \Rightarrow f_r(x, c(b(\epsilon))));$$

$$\forall x \forall y (f_q(t, \epsilon, \epsilon) \Rightarrow f_r(\epsilon, b(\epsilon))).$$

On prend comme formule T_M la conjonction de toutes ces formules (qui sont bien en nombre fini), et prendre comme formule F_M la Formule $T_M \Rightarrow \exists x \exists y f_{q_a}(x, y)$ qui dit qu'un calcul doit être acceptant.

Par construction, si M accepte le mot vide, chacune des implications suivantes sera vraie dans n'importe quel modèle. Réciproquement, si la formule est vraie dans n'importe quel modèle, elle doit l'être dans le modèle particulier $\mathfrak{M}_M$, et donc M accepte le mot vide. □

Question 5. *Montrer qu'une formule close F du calcul des prédicats (logique du premier ordre) est satisfiable si et seulement si sa négation $\neg F$ n'est pas prouvable.*

Solution : Cela découle du théorème de complétude qui démontre qu'un formule est prouvable si et seulement si elle est vraie dans tout modèle. Si $\neg F$ est prouvable, dans tout modèle, on doit avoir $\neg F$ vrai, et donc tout modèle de F serait nécessairement modèle de F et $\neg F$, ce qui est impossible. Réciproquement si $\neg F$ n'est pas prouvable, la preuve du théorème de complétude fournit un modèle qui satisfait F. □

Question 6. *On considère le problème $SAT_{prédicat}$ de décision suivant :*

— ***Donnée**: Une formule close F du calcul des prédicats (logique du premier ordre).*

— ***Réponse**: F est-elle satisfiable.*

Ce problème est-il décidable ? Récursivement énumérable ? Dans NP, NP-complet, dans P ? Justifier votre réponse.

Solution : Il est indécidable. En effet, c'est la même chose que décider si la négation de F est prouvable (en inversant la réponse), et la question 4 réduit le problème indécidable de la question 3 à celui-ci.

On sait que F est satisfiable si et seulement si $\neg F$ n'est pas prouvable. Pour décider si $\neg F$ est prouvable, on peut énumérer toutes les preuves possibles jusqu'à trouver cette preuve. Donc son complémentaire est récursivement énumérable : il est récursivement énumérable de savoir si F n'est pas satisfiable. Comme on sait qu'un problème qui est récursivement énumérable et donc le complémentaire est récursivement énumérable est nécessairement décidable, et qu'on sait qu'il est indécidable, il n'est pas récursivement énumérable.

Il n'est pas dans NP, donc pas NP-complet, ni dans P car toutes ces classes sont des classes de langages décidables. □

Question 7. *La signature précédente possède des symboles d'arité* 1 *et* 2.

Etant donné un entier k, on considère le problème $SAT_{prédicat,k}$ de décision suivant :

— ***Donnée**: Une formule close F du calcul des prédicats (logique du premier ordre) sur une signature avec des symboles d'arité au plus k.*

— ***Réponse**: F est-elle prouvable.*

Montrer que pour certains entier k que l'on précisera, le problème $SAT_{prédicat,k}$ est RE-complet : il est récursivement énumérable, et pour tout problème A récursivement énumérable, $A \leq_m SAT_{prédicat,k}$.

On admettra qu'il est décidable pour $k = 1$.

Solution : Pour tout entier k, le problème est récursivement énumérable : pour savoir si F est prouvable, on peut énumérer toutes les preuves possibles jusqu'à trouver cette preuve.

Pour $k \geq 2$, il est RE-difficile : étant donné un problème A récursivement énumérable, on sait qu'il existe une machine de Turing M qui accepte A. La question 4 produit une formule F_M sur une signature avec des symboles d'arité 1 et 2 qui est prouvable si et seulement si M accepte w. Pour $k > 2$, on peut ajouter des symboles d'arité k, sans que cela ne change rien. Cela constitue une réduction de A vers ce problème, si l'on observe qu'à chaque fois il est facile de produire F_M à partir de M.

Pour $k = 1$, il ne peut pas être RE-complet, car on sait qu'il est décidable, et qu'il y a des problèmes récursivement énumérables qui ne sont pas décidables. □

3 NP-complétude

On s'intéresse dans cette question au problème de décision suivant : Set Packing :

— **Donnée**: Un ensemble fini U, un ensemble $\mathcal{C}$ composé de sous-ensembles $S_1, \ldots, S_t$ de U, et un entier z.

— **Réponse**: Existe-il z éléments de $\mathcal{C}$ deux-à-deux disjoints ?

Rappelons qu'une *couverture de sommets* S d'un graphe G est un sous-ensemble de sommets tel que toutes les arêtes de G ont au moins une extrémité dans S.

Partant d'un graphe $G = (V, E)$, nous allons construire un ensemble $\mathcal{C}$ composé de $|V|$ éléments de la façon suivante : pour chaque sommet $v \in V$, on définit l'ensemble S_v comme l'ensemble de toutes les arêtes qui ont v comme extrémité.

Question 8. *Montrer que le graphe possède une couverture de sommets avec k sommets si et seulement si $n - k$ éléments de $\mathcal{C}$ sont deux-à-deux disjoints.*

Solution : Suppposons que A soit une couverture de sommets avec k sommets. Considérons $B = V - A$ (le complémentaire de A).

Nous allons prouver par contradiction que si v et u appartiennent à B, alors S_v et S_u sont deux ensembles disjoints. En effet, supposons que S_v et S_u ne soient pas deux ensembles disjoints. Cela signifie qu'il existe une arête e qui est dans S_v et S_u. Cette arête e est nécessairement l'arête (u, v), mais on a supposé que ni u ni v n'appartiennent à A, et donc cette arête ne serait pas couverte par A.

Les ensembles S_v et S_u sont donc disjoints deux-à-deux, et donc les $n-k$ éléments de $\mathcal{C}$ sont deux-à-deux disjoints.

Réciproquement, supposons qu'il existe $n-k$ éléments de $\mathcal{C}$ deux-à-deux disjoints. Notons les S_{v_1}, ..., $S_{v_{n-k}}$. Considérons $B = \{v_1, \ldots, v_{n-k}\}$ et $A = V - B$ (son complémentaire). Nous allons prouver que A est une couverture sommet de G. En effet, pour chaque arête $e = (u, v)$, il n'est pas possible que ni u ni v soient dans B. En effet, par construction, on aurait $e \in S_u$ et $e \in S_v$. Ceci contredirait que S_u, ..., S_v soient des ensembles disjoints.

Puisqu'il est impossible que ni u ni v soient dans B, au moins l'un des deux est dans A. Autrement dit, A est une couverture de sommets de G,

□

Question 9. *Montrer que le problème* SET PACKING *est NP-complet.*

On utilisera la NP-complétude du problème Couverture de sommets (VC)

— ***Donnée**: Un graphe non-orienté $G = (V, E)$ et un entier k.*

— ***Réponse**: G admet il une couverture de sommets $\mathcal{S}$ avec au plus k sommets ?*

Solution : Le problème SET PACKING est dans NP car la donnée des z éléments de $\mathcal{C}$ constitue un certificat vérifiable facilement en temps polynomial.

En effet, on peut vérifier en temps polynomial si deux éléments J et I de C sont deux-à-deux disjoints : un algorithme naïf (comparer deux-à-deux les éléments) peut se faire en $\mathcal{O}(|J| \cdot |I|)$ opérations. Décider si z éléments de $\mathcal{C}$ sont deux-à-deux disjoints peut se faire en temps polynomial ($\mathcal{O}(|U|^2 z^2)$ opérations).

On peut réduire VC à SET PACKING de la façon suivante : Soit (G, k) une instance du problème VC. Soit n le nombre de sommets de G.

On construit une instance SET PACKING de la façon suivante

1. l'ensemble U de u éléments correspond à l'ensemble des arêtes : $U = E$;
2. l'ensemble $\mathcal{C}$ est composé de sous-ensembles $S_1, \ldots, S_n$: S_v est composé de toutes les arêtes adjacentes à v ;
3. $z = n - k$.

La transformation se fait bien en temps polynomial.

Par la question précédente, le graphe G possède une couverture de sommets S de taille k si et seulement s'il existe $n-k$ éléments de $\mathcal{C}$ deux-à-deux disjoints.

On a donc bien une réduction de VC vers SET PACKING. □

4 Théorème de Hall : du cas fini au cas infini

On note $\mathbb{N}^+$ pour l'ensemble des entiers naturels non-nuls. On considère $I = \mathbb{N}^+$ ou $I = \{1, 2, \ldots, n\}$ dans tout ce qui suit.

Etant donné un ensemble S et une famille $(S_i)_{i \in I}$ de sous-ensembles de S, un *système de représentants distincts* est un choix d'éléments (c'est-à-dire une famille $(x_i)_{i \in I}$) avec pour tout i, $x_i \in S_i$ tel que pour $i \neq j \in I$, on a $x_i \neq x_j$.

Par exemple, pour $S = \mathbb{N}^+$, $I = \mathbb{N}^+$, et la famille $(S_i)_{i \in \mathbb{N}^+}$, définie par $S_i = \{i, i+1\}$, on a $S_1 = \{1, 2\}$, $S_2 = \{2, 3\}$, ..., on peut prendre comme système de représentants distincts $x_i = i$.

On admettra le résultat suivant [2], pour le cas où la famille (c'est-à-dire I) est finie :

Théorème [Hall 1935]. Considérons un entier $n > 0$. Une condition nécessaire est suffisante pour l'existence d'un système de représentants distincts pour une famille $(S_i)_{1 \leq k \leq n}$ est la suivante :

(H_n) Pour tout $1 \leq k \leq n$, et tout choix d'indices distincts $1 \leq i_1, \ldots, i_k \leq n$, on a $|S_{i_1} \cup \cdots \cup S_{i_k}| \geq k$.

Dans cette partie, on cherche à étendre ce théorème au cas d'une famille infinie.

Question 10. *Une première tentative serait de penser qu'une condition nécessaire et suffisante pour l'existence d'un système de représentants distincts pour une famille $(S_i)_{i \in \mathbb{N}^+}$ est la suivante :*

(H_∞) *Pour tout $k \in \mathbb{N}^+$, et tout choix d'indices distincts $i_1, \ldots, i_k \in \mathbb{N}^+$, on a*

$$|S_{i_1} \cup \cdots \cup S_{i_k}| \geq k.$$

Montrer que cela ne suffit pas, en exhibant une famille de sous-ensembles de $\mathbb{N}$ (les entiers naturels) pour lesquels il n'existe pas de système de représentants distincts. On pourra choisir $S_1 = \mathbb{N}$, et les autres S_i finis.

Solution : Prendre par exemple $S_1 = \mathbb{N}$, et $S_2 = \{0\}$, $S_{n+2} = \{n\}$ pour tout n. Il ne peut pas y avoir de système de représentant distincts, car si on a choisi l'indice i pour S_1, on ne peut pas le choisir pour S_{i+2}, et donc S_{i+2} n'a pas d'indice. □

Par contre, cela fonctionne si l'on suppose que chaque S_i est **finie**. En effet, nous allons prouver le résultat suivant :

Théorème. On considère une famille $(S_i)_{i \in \mathbb{N}^+}$ d'ensembles **finis** de S .

Une condition nécessaire et suffisante pour l'existence d'un système de représentants distincts pour une famille $(S_i)_{i \in \mathbb{N}^+}$ est la suivante :

(H_∞) Pour tout $k \in \mathbb{N}^+$, et tout choix d'indices distincts $i_1, \ldots, i_k \in \mathbb{N}^+$, on a

$$|S_{i_1} \cup \cdots \cup S_{i_k}| \geq k.$$

Question 11. *Prouver que (H_∞) est une condition nécessaire.*

Solution : Si on a un système de représentants distincts pour la famille $(S_i)_{i \in \mathbb{N}^+}$, cela en fournir un pour toute sous-famille extraite de taille n et par le théorème de Hall, on doit avoir l'hypothèse H_n pour tout n. Ce qui implique la propriété. □

Question 12. *Prouver l'autre direction du théorème. Indication : On pourra chercher à définir une théorie qui est satisfiable si et seulement si la famille admet un système de représentants distincts.*

Solution : On introduit la famille de formules $C_{n,x}$, avec $x \in S_n$.

On considère l'ensemble Σ constitué de toutes les formules :

$$(a) \neg(C_{n,x} \wedge C_{m,x}) \text{ pour } n \neq m \in \mathbb{N}^+, x \in S_n \cap S_m.$$

$$(b) \neg(C_{n,x} \wedge C_{n,y}) \text{ pour } n \in \mathbb{N}^+, x \neq y \in S_n \cap S_m.$$

$$(c)\ C_{n,x_1} \vee \cdots \vee C_{n,x_k} \text{ pour } n \in \mathbb{N}^+, \text{ où } S_n = \{x_1, \ldots, x_k\}.$$

2. Qui n'est pas difficile à démontrer.

Supposons qu'on arrive à satisfaire Σ. On peut alors obtenir un système de représentants distincts en choisissant x dans S_n si et seulement si $C_{n,x}$ est vrai dans l'affectation qui satisfait Σ.

En effet, par (b) et (c), chaque S_n se verra affecté d'un unique représentant. Par (a), on affectera pas le même représentant à plusieurs ensembles.

Σ est finiment satisfiable. En effet, pour cela, considérons un sous-ensemble Σ_0 de Σ, et appelons $i_1, \ldots, i_l$ les indices qui apparaissent dans Σ_0. La famille $\{S_{i_1}, S_{i_2}, \ldots, S_{i_l}\}$ doit alors vérifier l'hypothèse H_∞. Par le théorème de Hall, il y a un système de représentants distincts pour $\{S_{i_1}, S_{i_2}, \ldots, S_{i_l}\}$. Considérer $C_{i_r,x}$ vrai si et seulement si $x = x_r$. Cela satisfait Σ_0.

Par le théorème de Compacité, puisque Σ est finiment satisfiable, Σ est satisfiable, et donc on a un système de représentants distincts.

□

5 Hiérarchie arithmétique : vision logique

Dans cette partie, on considère des formules sur la signature $\mathcal{L} = (0, s, +, \times, <, =)$ de l'arithmétique.

Une formule sur cette signature est dite Σ_0 si elle appartient au plus petit ensemble contenant les formules atomiques et clos par :

— conjonction $\wedge$ (finie), disjonction $\vee$ (finie), négation $\neg$;
— quantification universelle bornée : si φ est Σ_0, si t est un terme, et x une variable n'apparaissant pas dans t, alors $\forall x \leqslant t\ \varphi$ est Σ_0 (ici, $\forall x\ \leqslant t\ \varphi$ est une abréviation pour $\forall x\ (x \leqslant t \Rightarrow \varphi)$) ;
— quantification existentielle bornée, que l'on définit de façon similaire.

Par convention, on considère que Π_0 et Σ_0 sont des synonymes.

Pour $n \in \mathbb{N}$, on dit d'une formule qu'elle est :

— Σ_{n+1} si elle est de la forme $\exists x\ \varphi$ pour φ une formule Π_n ;
— Π_{n+1} si elle est de la forme $\forall x\ \varphi$ pour φ une formule Σ_n.

L'intuition est que Σ (respectivement : Π) signifie que l'on commence par une quantification existentielle (respectivement universelle), et l'indice indique le nombre d'alternances de quantificateurs, sans compter les quantifications bornées. Par exemple, $\exists n \forall m \leq n\ n \leq m \times m$ est une formule Σ_1, et $\exists n \forall j \forall m \leq n\ m \leq j \times n$ est une formule Σ_2.

On cherche à comprendre les sous-ensembles de $\mathbb{N}^p$ que l'on arrive à définir avec ces formules : on dit qu'une formule $\phi(x_1, \ldots, x_p)$ a p-variables libres définit un sous-ensemble de $\mathbb{N}^p$: cet ensemble correspond à l'ensemble des p-uplets qui la satisfait. Un ensemble est alors dit Σ_n (respectivement : Π_n) définissable si ϕ est une formule Σ_n (respectivement : Π_n). On dit qu'il est Δ_n si il est à la fois Σ_n et Π_n définissable.

Pour $n = 0$, on peut se convaincre par (induction sur la forme des formules) que tout ensemble définit par une formule $\Pi_0 = \Sigma_0$ est décidable. Les sous-ensembles Δ_0 sont donc décidables.

Question 13. *Montrer que si une partie A de $\mathbb{N}^p$ est Σ_1 alors A est récursivement énumérable.*

Solution : On note $\vec{x}$ pour $x_1, \ldots, x_p$.

Supposons que $A \subseteq \mathbb{N}^p$ soit Σ_1 : cela implique que déterminer si $\vec{x} \in A$ est équivalent à déterminer si $\exists u\ \phi$ pour une formule $\phi = \phi(\vec{x}, u)$ qui est Π_0, et donc décidable. Soit M la machine de Turing qui, étant donné $\vec{x}$ et u décide si $\phi(\vec{x}, u)$. A est semi-décidé par l'algorithme qui consiste, sur l'entrée $\vec{x}$, a tester pour chaque u si $\phi(\vec{x}, u)$, et accepte dès qu'elle trouve un tel u. □

On admettra que toute partie A de $\mathbb{N}^p$ décidable est Σ_1. On pourra aussi admettre qu'on peut [3] construire une bijection $\pi : \mathbb{N} \to \mathbb{N}^2$ telle que si l'on note $\pi(r) = (\pi_1(r), \pi_2(r))$, les relations $u = \pi_1(r)$ et $v = \pi_2(r)$ sont Δ_0.

Question 14. *Montrer la réciproque : une partie A de $\mathbb{N}^p$ est Σ_1 si elle est récursivement énumérable.*

Solution : Supposons que A soit récursivement énumérable. A est reconnu par une machine de Turing M. On a $\vec{x} \in A$ si et seulement s'il existe un temps t tel que M accepte $\vec{x}$ au temps t. Puisque "M accepte $\vec{x}$ au temps t" est décidable, et qu'on admet que décidable implique Σ_1, donc peut s'écrire sous la forme $\exists u\ \phi(\vec{x}, t, u)$ pour une certaine formule $\Pi_0 = \Sigma_0$. On a alors $\vec{x} \in A$ qui s'écrit $\exists r\ \phi(\vec{x}, \pi_1(r), \pi_2(u))$. □

Question 15. *En déduire que les ensembles Δ_1 de $\mathbb{N}^p$ sont exactement les ensembles décidables de $\mathbb{N}^p$.*

Solution : Les ensembles décidables correspondent aux ensembles semi-décidables dont le complémentaire est semi-décidable. Le complémentaire d'un ensemble Σ_1 correspond à un ensemble Π_1. Cela découle donc de la question précédente, et de la définition de Δ_1. □

Fixons $p \geq 1$. On dit qu'un ensemble $U \subseteq \mathbb{N}^{p+1}$ est Σ_n-universel s'il est dans Σ_n et si pour tout $A \subseteq \mathbb{N}^p$ qui est dans Σ_n, il existe $i \in \mathbb{N}$ tel que $A = U_i$, où $U_i = \{(x_1, \ldots, x_p) | (i, x_1, \ldots, x_p) \in U\}$. On dit qu'un ensemble est Π_n universel, si on a la même propriété en remplaçant Σ_n par Π_n dans la phrase précédente.

On fixe un codage [4] des machines de Turing par les entiers, de telle sorte que l'on puisse parler de la machine de Turing numéro n.

Question 16. *Montrer que $U = \{(i, x_1, \ldots, x_p) |$ la machine de Turing i accepte $(x_1, \ldots, x_p)\}$ est Σ_1-universel.*

Solution : U correspond à un problème de décision récursivement énumérable, donc Σ_1. D'autre part, soit A un ensemble Σ_1 de $\mathbb{N}_p$: A est donc accepté par une machine de Turing M, qui possède un numéro i. L'ensemble A correspond alors à U_i. □

Question 17. *Montrer que pour tout $n \geq 1$, et pour tout $p \geq 1$, il existe un ensemble Σ_n-universel, et un ensemble Π_n-universel dans $\mathbb{N}^{p+1}$.*

Solution : Pour $U \subseteq \mathbb{N}^{p+1}$, notons $\overline{U}$ pour son complémentaire. Observons que $\overline{U}_i = \{(\vec{x}) | (i, \vec{x}) \notin U\}$ est le complémentaire de $U_i = \{(\vec{x}) | (i, \vec{x}) \in U\}$.

Notons U^1 pour le langage de la question précédente.

Ayant définit le langage U^n, définissons $U^{n+1} = \{(n, \vec{x}) | \exists u\ (n, u, \vec{x}) \in \overline{U^n}\}$.

On montre par récurrence que U^n est Σ_n-universel et son complémentaire $\overline{U^n}$ est Π_n-universel.

Pour $n = 1$, il reste à voir que $\overline{U^1}$ est Π_1-universel. Mais si A est Π_1, son complémentaire est Σ_1, et donc correspond à U_i^1 pour un certain entier i, et donc A correspond à $\overline{U^1}_i$.

Supposons la propriété pour n. Un langage A de Σ_{n+1} s'écrit sous la forme $\exists u\ \phi$, avec ϕ qui définit un langage qui est Π_n. Par hypothèse de récurrence, cela correspond à $\overline{U^n}_i$ pour un certain entier i. On a donc bien que A est U_i^{n+1}. Pour un langage A de Π_{n+1}, on peut raisonner sur son complémentaire qui est Σ_{n+1} et obtenir la conclusion.

Par ailleurs, on se persuade facilement par récurrence sur n que U^n correspond bien à une formule Σ_n et symétriquement pour son complémentaire. □

3. C'est très facile.

4. Raisonnable, c'est-à-dire comme dans le cours, tel que l'on puisse bien retrouver chacun des ingrédients de la machine (programme, état initial, etc..) à partir de l'entier qui le code.

Question 18. *Montrer que les inclusions* $\Delta_n \subseteq \Sigma_n$ *et* $\Delta_n \subseteq \Pi_n$ *sont strictes pour* $n \geq 1$.

Solution : Il suffit de prouver que le langage universel U^n (qui est dans Σ_n) n'est pas dans Δ_n. Cela prouve la première inclusion stricte, et la seconde, puisque son complémentaire (qui est dans Π_n) ne saurait être dans Δ_n car Δ_n est close par complémentaire.

Montrons donc que le langage universel U^n (qui est dans Σ_n) n'est pas dans Δ_n : supposons par l'absurde qu'il l'était. On peut considérer $A = \{i | (i, i) \in U^n\}$. Δ_n étant close par complément, le complémentaire $\overline{A}$ de A serait aussi dans Δ_n. Puisque $\Delta_n \subseteq \Sigma_n$ et que Σ_n est universel, il doit exister un entier i_0 tel que $\overline{A} = U^n_{i_0}$. Mais on obtient $i_0 \notin A$ si et seulement si $(i_0, i_0) \in U^n$ si et seulement si $i_0 \in A$ ce qui est impossible. □

Question 19. *Montrer que chacun des niveaux* Σ_n *et* Π_n *pour* $n \geq 1$ *possède des problèmes complets. Donner un exemple de tels problèmes pour chacun des niveaux.*

Solution : Un problème U Σ_n-universel est nécessairement Σ_n-complet, . En effet, il est dans Σ_n par définition, et d'autre part, si on a un problème A qui est Σ_n, il se réduit à U par la fonction qui à $\vec{x}$ associe $(i, \vec{x})$, où i est l'indice correspondant à A.

Symétriquement pour Π_n.

On connait donc des exemples par les questions précédentes pour chacun des niveaux. □

Il est aussi possible de donner des caractérisations de chacun des niveaux de cette hiérarchie en utilisant des machines de Turing étendues (avec des oracles).