INF105
Contrôle de connaissances — Corrigé

Théorie des langages

15 juin 2023

Consignes.

Les exercices sont totalement indépendants. Ils pourront être traités dans un ordre quelconque, mais on demande de faire apparaître de façon très visible dans les copies où commence chaque exercice.

L'usage de tous les documents (notes de cours manuscrites ou imprimées, feuilles d'exercices, livres) est autorisé.

L'usage des appareils électroniques est interdit.

Durée : 1h30

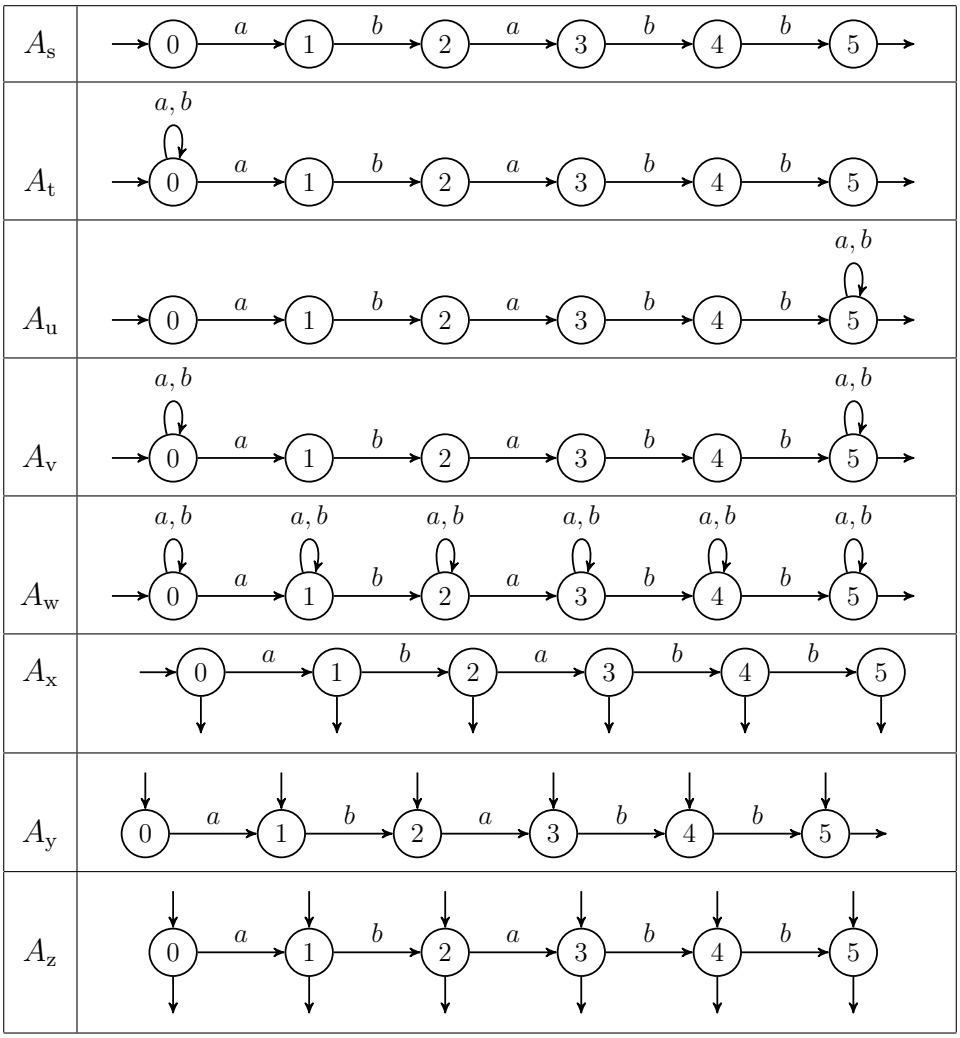
Barème *indicatif* et *approximatif* : exercice 1 : (a) 1.5, (b) 1.5, (c) 1, (d) 4, (e) 3, (f) 3 (soit au total 14) ; exercice 2 : (a) 1, (b) 2, (c) 1, (d) 2 (soit au total 6).

Ce corrigé comporte 8 pages (page de garde incluse).

Git : 8297c70 Thu Jun 29 16:30:06 2023 +0200

Exercice 1.

Dans cet exercice, on pose $\Sigma = \{a, b\}$. On considère les huit automates suivants ($A_{\mathrm{s}}, A_{\mathrm{t}}, A_{\mathrm{u}}, A_{\mathrm{v}}, A_{\mathrm{w}}, A_{\mathrm{x}}, A_{\mathrm{y}}, A_{\mathrm{z}}$) sur l'alphabet Σ :



On définit aussi les douze langages suivants, où w_0 désigne le mot $ababb$:

— $L_0 = \varnothing$: le langage vide.
— $L_1 = \{w_0\}$: le langage constitué du seul mot w_0.
— L_2 : le langage constitué des mots qui sont un sous-mot de w_0.
— L_3 : le langage constitué des mots qui ont w_0 comme sous-mot.
— L_4 : le langage constitué des mots qui sont un facteur de w_0.
— L_5 : le langage constitué des mots qui ont w_0 comme facteur.
— L_6 : le langage constitué des mots qui sont un préfixe de w_0.
— L_7 : le langage constitué des mots qui ont w_0 comme préfixe.
— L_8 : le langage constitué des mots qui sont un suffixe de w_0.
— L_9 : le langage constitué des mots qui ont w_0 comme suffixe.
— $L_{10} = \{w_0^i : i \in \mathbb{N}\}$: le langage constitué des mots qui sont une répétition d'un nombre quelconque (i) de copies de w_0 (par exemple, $ababbababb$).
— $L_{11} = \{b^i w_0 a^i : i \in \mathbb{N}\}$: le langage constitué des mots obtenus en précédant w_0 d'un nombre quelconque (i) de copies de la lettre b et en le suivant du même nombre de copies de la lettre a (par exemple, $bbababbaa$).

(a) Pour chacun des huit automates $A \in \{A_\mathrm{s}, A_\mathrm{t}, A_\mathrm{u}, A_\mathrm{v}, A_\mathrm{w}, A_\mathrm{x}, A_\mathrm{y}, A_\mathrm{z}\}$, dire lequel ou lesquels parmi les douze langages $L_0, \ldots, L_{11}$ est celui reconnu par l'automate A. On ne demande pas de justifier la réponse.

Corrigé. L'automate A_s reconnaît le langage $L_1 = \{w_0\}$.

L'automate A_t reconnaît le langage L_9 des mots ayant w_0 comme suffixe (car un chemin dans A_t finit par un chemin consommant w_0).

L'automate A_u reconnaît le langage L_7 des mots ayant w_0 comme préfixe (car un chemin dans A_u commence par un chemin consommant w_0).

L'automate A_v reconnaît le langage L_5 des mots ayant w_0 comme facteur (car un chemin dans A_u passe par un chemin consommant w_0).

L'automate A_w reconnaît le langage L_3 des mots ayant w_0 comme sous-mot (car un chemin dans A_u consomme les lettres a, b, a, b, b dans cet ordre, intercalées par un nombre quelconque de lettres quelconques).

L'automate A_x reconnaît le langage L_6 des préfixes de w_0 (car un chemin dans A_x consomme le début de w_0).

L'automate A_y reconnaît le langage L_8 des suffixes de w_0 (car un chemin dans A_y consomme la fin de w_0).

L'automate A_z reconnaît le langage L_4 des facteurs de w_0 (car un chemin dans A_z consomme un intervalle de lettres consécutives de w_0).

(Les justifications entre parenthèses n'étaient pas demandées.) ✓

(b) Pour chacun des sept automates $A \in \{A_\mathrm{s}, A_\mathrm{t}, A_\mathrm{u}, A_\mathrm{v}, A_\mathrm{w}, A_\mathrm{x}, A_\mathrm{y}\}$ (cette question-ci n'est pas posée pour A_z), donner une expression rationnelle dénotant le langage L reconnu par A. On ne demande pas de justifier la réponse, et il n'est pas obligatoire d'appliquer un algorithme vu en cours.

Corrigé. L'automate A_s reconnaît le langage dénoté par $ababb$.

L'automate A_t reconnaît le langage dénoté par $(a|b)*ababb$.

L'automate A_u reconnaît le langage dénoté par $ababb(a|b)*$.

L'automate A_v reconnaît le langage dénoté par $(a|b)*ababb(a|b)*$.

L'automate A_w reconnaît le langage dénoté par $(a|b)*a(a|b)*b(a|b)*a(a|b)*b(a|b)*b(a|b)*$.

L'automate A_x reconnaît le langage dénoté par $\underline{\varepsilon}|a|ab|aba|abab|ababb$ (simple énumération de tous les préfixes de w_0) ou, si on préfère, par $\underline{\varepsilon}|a(\underline{\varepsilon}|b(\underline{\varepsilon}|a(\underline{\varepsilon}|b(\underline{\varepsilon}|b))))$ (obtenue en éliminant les états de A_x dans l'ordre $5, 4, 3, 2, 1, 0$).

L'automate A_y reconnaît le langage dénoté par $\underline{\varepsilon}|b|bb|abb|babb|ababb$ (simple énumération de tous les suffixes de w_0) ou, si on préfère, par $\underline{\varepsilon}|(\underline{\varepsilon}|(\underline{\varepsilon}|(\underline{\varepsilon}|a)b)a)b)b$ (obtenue en éliminant les états de A_y dans l'ordre $0, 1, 2, 3, 4, 5$). ✓

(c) Pour chacun des huit automates $A \in \{A_\mathrm{s}, A_\mathrm{t}, A_\mathrm{u}, A_\mathrm{v}, A_\mathrm{w}, A_\mathrm{x}, A_\mathrm{y}, A_\mathrm{z}\}$, dire de quel type d'automate vu en cours (DFA complet, DFA à spécification incomplète, NFA ou bien NFA à transitions spontanées) est l'automate A. On donnera à chaque fois le type le plus particulier applicable.

Corrigé. L'automate A_s est un DFA à spécification incomplète (ou DFAi).

L'automate A_t est un NFA (à cause de la double transition étiquetée a depuis l'état 0).

L'automate A_u est un DFA à spécification incomplète (ou DFAi).

L'automate A_v est un NFA.

L'automate A_w est un NFA.

L'automate A_x est un DFA à spécification incomplète (ou DFAi).

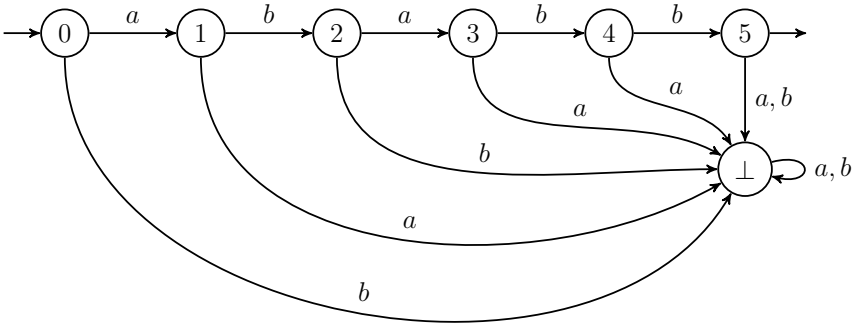
L'automate A_y est un NFA (à cause des multiples états initiaux).

L'automate A_z est un NFA. ✓

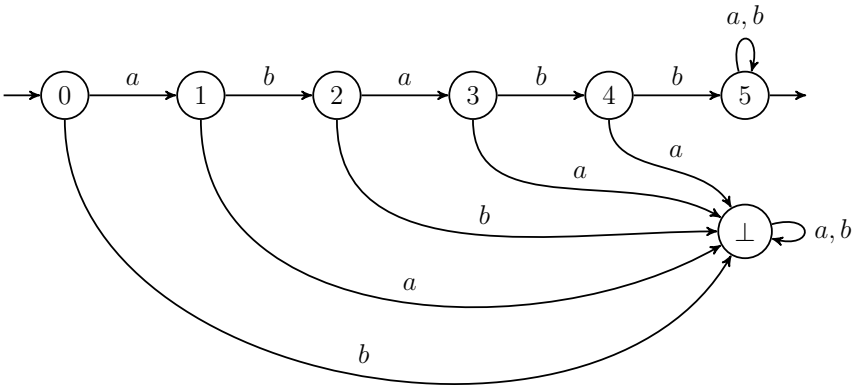
(d) Pour chacun des cinq automates $A \in \{A_\mathrm{s}, A_\mathrm{u}, A_\mathrm{v}, A_\mathrm{x}, A_\mathrm{z}\}$ (cette question-ci n'est pas posée pour $A_\mathrm{t}, A_\mathrm{w}, A_\mathrm{y}$), donner un DFA complet sans état inaccessible qui soit équivalent à A (c'est-à-dire, reconnaissant le langage L). On appliquera un algorithme vu en cours en le nommant.

Conseil sur la présentation graphique : pour s'éviter des maux de tête dans le placement des états (et en éviter aussi au correcteur), il est conseillé de commencer par placer horizontalement de gauche à droite les états successifs rencontrés lors de la consommation du mot $w_0 = ababb$ par l'automate construit, et d'ajouter ensuite les autres états éventuellement nécessaires.

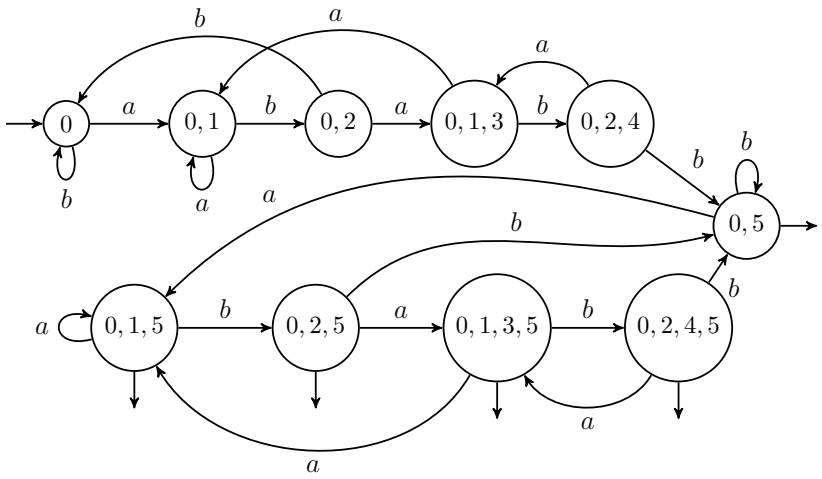
Corrigé. S'agissant de A_s, comme c'est un DFAi, il s'agit simplement de lui ajouter un état « puits », noté $\perp$ ci-dessous, où aboutissent toutes les transitions manquantes :



La construction est analogue pour A_u :

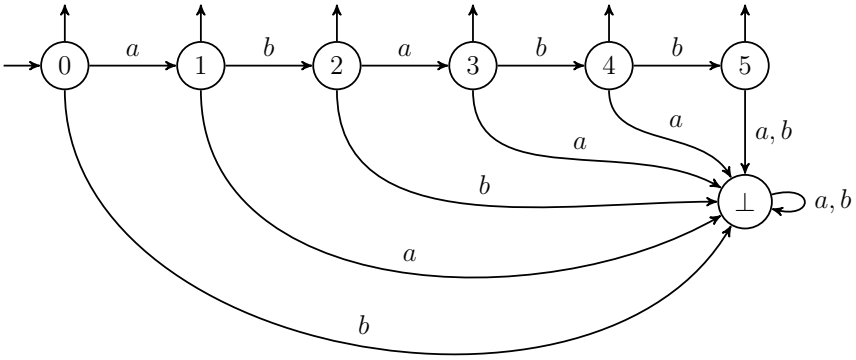


S'agissant de A_v, on a affaire à un « vrai » non-déterminisme et on applique donc l'algorithme de déterminisation vu en cours, qui donne (en omettant les accolades dans le nommage des états) :

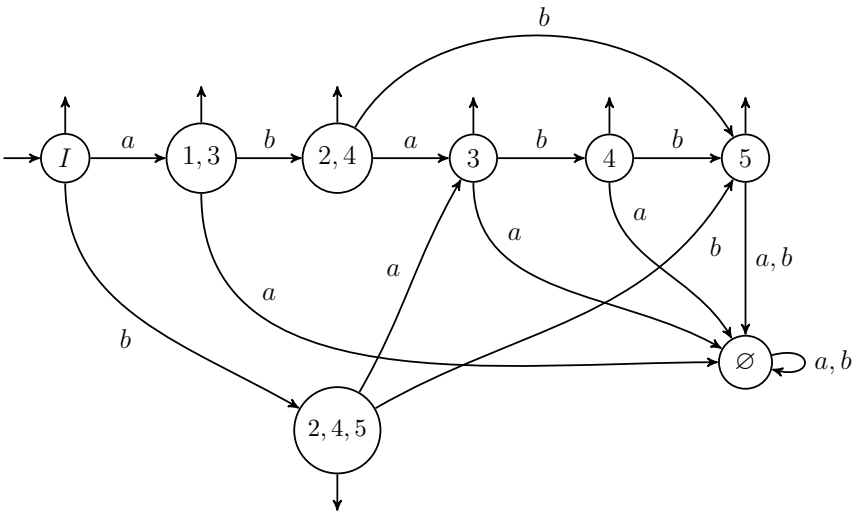


Note : Une version précédente de ce corrigé donnait un automate certes correct (DFA complet sans état inaccessible qui soit équivalent à A) mais qui n'est pas celui, représenté ci-dessus, obtenu en appliquant l'algorithme de déterminisation. (L'automate qui était représenté était, en fait, l'automate minimal obtenu en identifiant tous les états contenant 5 dans celui ci-dessus.)

La construction pour A_{x} est exactement comme pour A_{s} sauf que les états 0 à 5 sont tous marqués finaux :



S'agissant de A_{z}, on a de nouveau affaire à un « vrai » non-déterminisme et on applique donc l'algorithme de déterminisation vu en cours, qui donne (en omettant les accolades dans le nommage des états) :



où on a noté I au lieu de $\{0,1,2,3,4,5\}$. ✓

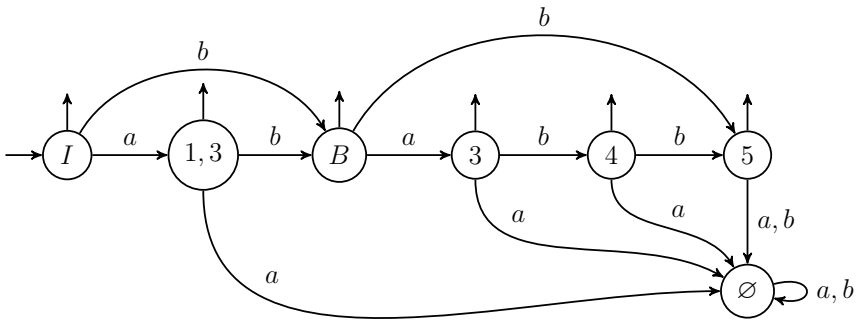
(e) Pour $A = A_{\text{z}}$ (et seulement celui-ci), minimiser le DFA trouvé à la question (d).

Corrigé. On a affaire à un DFA complet sans état inaccessible : on va le minimiser avec l'algorithme de Moore.

Dans une première étape, on fait deux classes d'état : celle des états finaux, c'est-à-dire $I, \{1,3\}, \{2,4\}, \{2,4,5\}, \{3\}, \{4\}, \{5\}$ et celle des états non-finaux, c'est-à-dire le seul état $\varnothing$.

Séparons maintenant les états selon la cible de la transition étiquetée par a : la classe formée de $I, \{1,3\}, \{2,4\}, \{2,4,5\}, \{3\}, \{4\}, \{5\}$ se scinde en deux : $I, \{2,4\}, \{2,4,5\}$ d'une part, où elle mène à un état final, et $\{1,3\}, \{3\}, \{4\}, \{5\}$ d'autre part, où elle mène à un état non-final. La cible de la transition étiquetée par b, pour sa part, sépare $\{5\}$ de tous les autres états (le seul où elle mène à un état non-final). À ce stade, on a quatre classes : la classe formée de $I, \{2,4\}, \{2,4,5\}$, celle de $\{1,3\}, \{3\}, \{4\}$, et deux classes singleton, $\{5\}$ et $\varnothing$.

À l'étape suivante, la cible de la transition étiquetée par a ne sépare rien (les états de la première classe mènent à la seconde et les états de la seconde mènent au puits). La cible de la transition étiquetée par b, en revanche, sépare la classe formée de $I, \{2,4\}, \{2,4,5\}$ en I d'une part (qui mène à cette même classe) et $\{2,4\}, \{2,4,5\}$ de l'autre (qui mène à $\{5\}$) ; et elle sépare la classe formée de $\{1,3\}, \{3\}, \{4\}$ en trois (puisque les cibles sont dans trois classes distinctes). À ce stade, les seuls états qui n'ont pas été séparés sont $\{2,4\}$ et $\{2,4,5\}$. Or comme ils ont exactement les mêmes cibles de transitions étiquetées par a et b, ils ne sont pas séparés, donc l'algorithme s'applique ici : il fusionne exactement deux états, à savoir $\{2,4\}$ et $\{2,4,5\}$, en un seul, que nous noterons B :



✓

(f) Parmi les douze langages $L_0, \ldots, L_{11}$, lesquels sont rationnels ? Lesquels sont algébriques ? Lesquels sont décidables ? Lesquels sont semi-décidables ? On justifiera la réponse à chaque fois (par exemple en donnant une expression rationnelle, un automate, une grammaire hors-contexte, un algorithme, ou n'importe quel autre type d'argument permettant de conclure).

Corrigé. Les langages $L_1, L_3, L_4, L_5, L_6, L_7, L_8, L_9$ sont rationnels car ils sont reconnaissables : on a vu que les automates $A_{\text{s}}, A_{\text{w}}, A_{\text{z}}, A_{\text{v}}, A_{\text{x}}, A_{\text{u}}, A_{\text{y}}, A_{\text{t}}$ respectivement les reconnaissent. Il reste donc à traiter le cas de L_0, L_2, L_{10}, L_{11}.

Les langages L_0 et L_2 sont rationnels car ils sont finis (on peut donner des expressions rationnelles pour les deux, à savoir $\perp$ pour L_0 et, en énumérant systématiquement tous les sous-mots, $\varepsilon|b|bb|a|ab|abb|bbb|ba|bab|babb|aa|aab|aabb|abbb|aba|abab|ababb$ pour L_2, mais ce n'est pas très intéressant et ce n'était pas demandé).

Le langage $L_{10} = \{w_0^i : i \in \mathbb{N}\} = L_1^*$ est rationnel car il est l'étoile de Kleene d'un langage rationnel (si on préfère, il est dénoté par l'expression rationnelle $(ababb)*$).

Tous les langages L_0 à L_{10} sont donc rationnels, et, en particulier, algébriques, décidables et semi-décidables.

Reste à évoquer le cas du langage $L_{11} = \{b^i w_0 a^i : i \in \mathbb{N}\}$. Montrons qu'il n'est pas rationnel, et montrons qu'il est algébrique.

Il n'est pas rationnel par le lemme de pompage. En effet, supposons par l'absurde que $\{b^i w_0 a^i : i \in \mathbb{N}\}$ soit rationnel, et soit k tel que donné par le lemme de pompage. Considérons le mot $t := b^k ababba^k \in L_{11}$. D'après la propriété de k, il existe une factorisation $t = uvw$ vérifiant les propriétés garanties par le lemme de pompage. Le fait que $|uv| \leq k$, comme le préfixe de longueur k de t est formé uniquement de la lettre b, assure que $u = b^{\ell_1}$ et $v = b^{\ell_2}$ pour certains ℓ_1, ℓ_2 avec, de plus, $\ell_2 > 0$ (car $v \neq \varepsilon$) et $\ell_1 + \ell_2 \leq k$. On a alors $w = b^{k-\ell_1-\ell_2} ababba^k$, et $uv^i w = b^{k+(i-1)\ell_2} ababba^k$. Comme les mots de L_{11} ont la propriété nécessaire que leur nombre initial de b (i.e., la longueur de leur plus long préfixe formé uniquement de la lettre b) est égale à leur nombre final de a (i.e., la longueur de leur plus long suffixe formé uniquement de la lettre a), on devrait avoir $k + (i-1)\ell_2 = k$, ce qui est une contradiction dès que $i \neq 1$.

En revanche, le langage L_{11} est algébrique, car il est engendré par la grammaire suivante d'axiome S :

$$S \to ababb \mid bSa$$

Étant algébrique, le langage L_{11} est décidable et semi-décidable. (Au demeurant, il est facile de donner un algorithme qui décide si un mot appartient à L_{11} : on vérifie que son nombre initial de b est égal à son nombre final de a et, une fois ce fait vérifié, on vérifie qu'une fois retiré le préfixe et le suffixe en question il reste exactement le mot $ababb$.)

Pour résumer, tous les langages dont on a parlé sont algébriques, décidables et semi-décidables, et tous sauf L_{11} sont rationnels. ✓

Exercice 2.

Dans cet exercice, les questions (1) et (2) sont indépendantes, et la question (4) ne dépend que de la question (2).

(1) La fonction $h\colon \mathbb{N} \to \mathbb{N}$ suivante est-elle calculable ?

$$h(0) = 1 \ , \ h(1) = 10 \ , \ h(2) = 10^{10} \ , \ h(3) = 10^{10^{10}} \ , \ h(4) = 10^{10^{10^{10}}} \ ,$$

$$h(n) = 10^{10^{\cdots^{10}}} \Big\} n$$

(Autrement dit, $h(n)$ est une tour d'exponentielle de hauteur n sur le nombre 10. Bien sûr, $10^{10^{10}}$ se comprend comme $10^{\left(10^{10}\right)}$.)

Corrigé. La fonction $g\colon n \mapsto 10^n$ est calculable : en effet, on peut exécuter une boucle à n itérations en multipliant par 10 la valeur d'une variable en partant de 1. On en déduit que la fonction h est calculable : en effet, on peut exécuter une boucle à n itérations en appliquant la fonction g à une variable en partant de 1. ✓

Fixons maintenant une numérotation (« codage de Gödel ») des algorithmes prenant en entrée un entier et renvoyant un entier, et appelons $\varphi_e(n)$ le résultat de l'exécution du programme codé par l'entier e sur l'entrée n, si cette exécution termine (et $\varphi_e(n)$ non défini si cette exécution ne termine pas). Considérons la fonction $b\colon \mathbb{N} \to \mathbb{N}$ suivante :

$$b(n) = \max\{\varphi_e(n) : 0 \leq e \leq n \text{ et } \varphi_e(n) \text{ défini}\}$$

(Autrement dit, $b(n)$ est la plus grande des valeurs $\varphi_e(n)$ qui sont définies lorsque e parcourt les entiers de 0 à n.)

(2) On veut montrer que pour toute fonction calculable $f\colon \mathbb{N} \to \mathbb{N}$, il existe un n_0 tel que pour tout $n \geq n_0$ on ait $b(n) > f(n)$. Supposons donc $f\colon \mathbb{N} \to \mathbb{N}$ calculable. **(a)** Expliquer

pourquoi il existe p tel que $\varphi_p = f + 1$ (c'est-à-dire $\varphi_p(n) = f(n) + 1$ pour tout n). **(b)** En déduire que $b(n) > f(n)$ lorsque $n \geq p$, et conclure.

Corrigé. **(a)** La fonction $n \mapsto f(n) + 1$ est calculable puisque f l'est et que l'addition l'est. Il existe donc un algorithme qui la calcule, c'est-à-dire un p tel que $\varphi_p = f + 1$.

(b) Sachant que $\varphi_p = f + 1$, si $n \geq p$, la valeur $b(n)$ est le maximum des valeurs $\varphi_e(n)$ qui sont définies lorsque e et n parcourent les entiers de 0 à n. En particulier, puisque $0 \leq p \leq n$, parmi ces entiers se trouve la valeur $\varphi_p(n) = f(n)+1$ (qui est bien définie car f est une fonction totale ici). On a donc $b(n) \geq \varphi_p(n) = f(n) + 1 > f(n)$. On a bien montré que si $n \geq p$ alors $b(n) > f(n)$, ce qui était voulu (pour $n_0 = p$). ✓

(3) Montrer que (pour les fonctions h et b introduites ci-dessus) :

$$\lim_{n \to +\infty} \frac{b(n)}{h(n)} = +\infty$$

Corrigé. On souhaite montrer que pour tout $C > 0$ il existe n_0 tel que si $n \geq n_0$ alors $\frac{b(n)}{h(n)} > C$. Pour cela, soit C un réel > 0, et, quitte à l'augmenter encore un peu, on peut supposer C entier. La fonction $C \cdot h$ (c'est-à-dire $n \mapsto C \cdot h(n)$) est calculable car h l'est (question (1)) et que la multiplication l'est. D'après la question (2), il existe n_0 tel que pour tout $n \geq n_0$ on ait $b(n) > C \cdot h(n)$: c'est ce qu'on voulait montrer. ✓

(4) On se propose ici de redémontrer l'indécidabilité du problème de l'arrêt de manière légèrement différente de ce qui a été vu en cours. Supposons donc par l'absurde que le problème de l'arrêt $H = \{(e, n) : \varphi_e(n) \text{ défini}\}$ soit décidable. **(a)** Montrer soigneusement, sous cette hypothèse, que b est calculable. **(b)** Conclure.

Corrigé. **(a)** Supposons par l'absurde qu'il existe un algorithme T qui décide le problème de l'arrêt : autrement dit, donnés e et n, cet algorithme est censé terminer en temps fini et répondre « oui » si le programme codé par e termine sur l'entrée n, « non » sinon. On peut alors calculer $b(n)$ de la manière suivante : on démarre avec $m = 0$ et on effectue une boucle pour e allant de 0 à n, et pour chacune des valeurs en question, on utilise T pour savoir si $\varphi_e(n)$ est défini, puis, *si* c'est le cas, on caclcule cette valeur $\varphi_e(n)$ au moyen de la machine universelle (ce calcul termine en temps fini justement puisque $\varphi_e(n)$ est défini), et on remplace la variable m par $\max(m, \varphi_e(n))$. À la fin de la boucle, on a bien calculé le max de toutes les valeurs $\varphi_e(n)$ qui sont définies, c'est-à-dire $b(m)$. La fonction b est donc calculable (de nouveau, sous l'hypothèse que le problème de l'arrêt l'est).

(b) On a vu en (2) que pour toute fonction calculable f il existe un n_0 tel que pour tout $n \geq n_0$ on ait $b(n) > f(n)$. Comme on vient de voir que (sous l'hypothèse que le problème de l'arrêt est calculable) la fonction b est calculable, en appliquant ce résultat à $f = b$, il doit exister un n_0 tel que tout $n \geq n_0$ on ait $b(n) > b(n)$, ce qui est absurde. C'est donc que l'hypothèse était absurde : le problème de l'arrêt n'est pas décidable. ✓