École polytechnique — X2021 — INF412 Fondements de l'informatique

PC6 notée

(corrigé)

Cet énoncé comporte quatre parties indépendantes et qui pourront être résolues dans n'importe quel ordre. Dans chaque partie, on pourra, pour répondre à une question, admettre les résultats dont on demande la démonstration aux questions *précédentes*. Il n'est pas nécessaire de traiter toutes les questions pour avoir la note maximale. Les correcteurs vous remercient d'avance d'écrire lisiblement.

1 Quelques questions de calculabilité

Les problèmes de décision suivants sont-ils décidables ? Justifier.

Question 1.1. *Déterminer si le langage reconnu par une machine de Turing est fini.*

Solution : Par le théorème de Rice, ça n'est pas décidable (c'est bien non trivial puisqu'on a par exemple la machine qui ne reconnaît aucun mot et la machine qui reconnaît tous les mots). □

Question 1.2. *Déterminer si une machine de Turing M accepte une entrée w après un nombre pair de transitions.*

Solution : On ne peut pas utiliser le théorème de Rice, car ça n'est pas une propriété des langages semi-décidables. On montre que c'est indécidable par réduction du problème de l'arrêt. Étant donnée une machine M et un mot w, on note M' la machine qui simule M sur w en un nombre pair de transitions (par exemple en ajoutant une transition inutile à chaque transition, on en effectuant un nombre pair de transitions chaque fois qu'on simule une transition de M). La machine M' est calculable à partir de M et on a bien que M' accepte après un nombre pair de transitions si et seulement si M accepte w. □

Question 1.3. *Déterminer si le langage reconnu par une machine de Turing est aussi reconnu par une machine avec plus de 10^{42} états.*

Solution : C'est toujours vrai : il suffit de rajouter des états qui ne sont ni la source ni le but de transition pour obtenir une machine avec plus de 10^{42} états. □

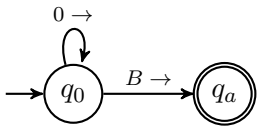
2 Machines de Turing et mots binaires

On considère des machines de Turing sur l'alphabet $\Sigma = \{0, 1\}$. L'objectif est d'implémenter certaines opérations sur les entiers codés en binaire. On rappelle par exemple que le codage binaire de 13 est 1101, ou bien 001101 (on autorisera des codages qui débutent par le bit 0, et le codage de 0 sera toujours non vide).

On demande des machines de Turing explicitement décrites. Pour chaque machine, on donnera quelques phrases expliquant leur fonctionnement. Toute réponse de type « une machine avec 50 états sans explication » sera refusée.

Question 2.1. *Proposer une machine de Turing qui détermine si un entier est nul.*

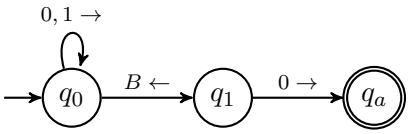
Solution : On vérifie que tous les caractères sont des 0 :



□

Question 2.2. *Proposer une machine de Turing qui détermine si un entier est pair.*

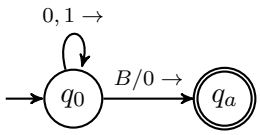
Solution : On déplace la tête de lecture au dernier caractère et on accepte lorsque celui-ci est 0 :



□

Question 2.3. *Proposition une machine de Turing qui multiplie un entier par deux.*

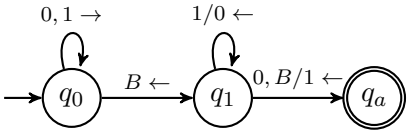
Solution : On ajoute un 0 à la fin de l'entrée :



□

Question 2.4. *Proposer une machine de Turing qui calcule le successeur d'un entier.*

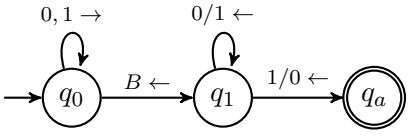
Solution : En partant de la droite, on change tous les 1 en 0, jusqu'au premier 0 (ou blanc) qu'on change en 1 :



□

Question 2.5. *Proposer une machine de Turing qui calcule le prédécesseur d'un entier supposé non nul.*

Solution : En partant de la droite, on change tous les 0 en 1, jusqu'au premier 1 qu'on change en 0 :

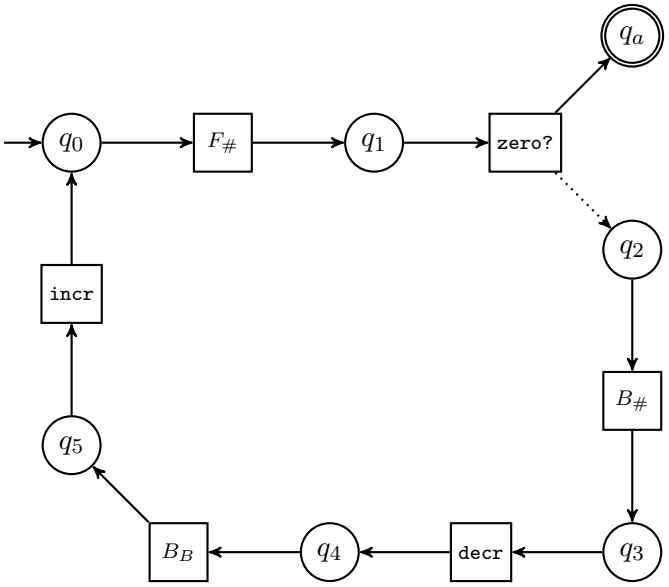


□

Question 2.6. *Proposer une machine de Turing qui calcule la somme de deux entiers séparés par un caractère spécial « # ». Par exemple, le résultat sur l'entrée* 110#101 *sera* 1011 *(car* $6+5=11$*). On s'autorisera à utiliser des machines auxiliaires qu'on introduira explicitement si elle ne l'ont pas déjà été à des questions précédentes.*

Solution : On note
- B_B la machine qui met la tête de lecture sur le caractère juste après le blanc de gauche,
- $B_\#$ la machine qui met la tête sur le caractère juste après le précédent #,
- $F_\#$ la machine qui met la tête sur le caractère juste après le prochain #,
- `zero?` la machine qui teste si l'entrée est nulle.

La machine suivante teste si le second nombre est 0, tant que ça n'est pas le cas on le décrémente et on incrémente le premier nombre :



Le résultat est le premier nombre (on pourrait aisément enlever le # et les 0 qui suivent). □

3 Groupes sans torsion

On considère la signature avec une constante « 1 », une fonction d'arité 2 « × » et l'égalité comme unique symbole de relation, d'arité 2. On notera $\mathcal{T}$ la théorie des groupes sur cette signature. Pour n entier, on s'autorisera la notation x^n pour $((x\times x)\times\ldots)\times x$, avec n occurrences de x.

On rappelle qu'il faut justifier les réponses aux questions suivantes. En particulier, veilliez à bien nommer explicitement les théorèmes utilisés.

Question 3.1. *Rappeler les axiomes de la théorie des groupes.*

Solution : Les axiomes sont
- associativité : $\forall x.\forall y.\forall y.(x\times y)\times z = x\times(y\times z)$
- unité : $\forall x.1\times x = x\wedge x\times 1 = x$
- existence d'un inverse : $\forall x.\exists y.y\times x = 1\wedge x\times y = 1$

□

Question 3.2. *La formule* $\exists x.x^5 = 1$ *est-elle prouvable dans cette théorie ?*

Solution : Oui, on peut toujours prouver $1^5 = 1$. □

Question 3.3. *La formule* $\exists x.\neg(x = 1) \wedge x^5 = 1$ *est-elle prouvable dans cette théorie ?*

Solution : Non, si elle l'était, elle serait vraie dans tout modèle, or le groupe additif $\mathbb{R}$ ne satisfait pas cette formule. □

Question 3.4. *La formule* $\neg(\exists x.\neg(x = 1) \wedge x^5 = 1)$ *est-elle prouvable dans cette théorie ?*

Solution : Non, si elle l'était, elle serait vraie dans tout modèle, or on peut exhiber des groupes dans lesquels ça n'est pas le cas.

— Le groupe multiplicatif $\mathbb{C}$ ne la satisfait pas : en notant $a = \mathrm{e}^{\mathrm{i}2\pi/5}$, on a $a \neq 1$ et $a^5 = 1$.
— Le groupe additif $\mathbb{Z}/5\mathbb{Z}$ ne la satisfait pas : on a $1 \neq 0$ mais $1 + 1 + 1 + 1 + 1 = 0$.

□

Étant donné un entier n, un élément a d'un groupe est dit d'*ordre* n lorsque $a^n = 1$. Un élément a d'un groupe est dit d'*ordre fini* lorsqu'il est d'ordre n pour un certain entier $n > 0$. Un groupe est *sans torsion* lorsqu'il ne contient aucun élément d'ordre fini à l'exception de l'unité.

Question 3.5. *Donner une théorie du premier ordre dont les modèles sont les groupes sans torsion. On notera* $\mathcal{U}$ *cette théorie.*

Solution : On ajoute à la théorie $\mathcal{T}$ des groupes les axiomes

$$\psi_n = \forall x.\neg(x = 1) \Rightarrow \neg(x^n = 1)$$

pour tout n entier avec $n > 0$. □

Question 3.6. *Cette théorie est-elle cohérente ?*

Solution : Oui : par le théorème de correction, il suffit d'exhiber un modèle. Les groupes additifs $\mathbb{Z}$, $\mathbb{Q}$ et $\mathbb{R}$ sont des modèles (et multiplicatifs aussi pour les deux derniers). □

Question 3.7. *Étant donné un entier* n*, proposer une théorie* $\mathcal{U}_n$ *dont les modèles sont les groupes qui contiennent au moins un élément d'ordre* n*, et aucun élément d'ordre* k *pour* $1 < k < n$*.*

Solution : En reprenant les notations de la question 3.5, on définit

$$\mathcal{U}_n = \mathcal{T} \cup \{\psi_1, \psi_2, \ldots, \psi_{n-1}, \neg\psi_n\}$$

□

On admettra dans la suite que, pour tout entier *premier* n, la théorie $\mathcal{U}_n$ admet le groupe additif $\mathbb{Z}/n\mathbb{Z}$ comme modèle.

Nous cherchons maintenant à montrer qu'on ne peut pas donner de théorie avec un nombre fini d'axiomes dont les modèles sont les groupes sans torsion. Nous allons raisonner par l'absurde et supposer qu'il existe une telle théorie $\mathcal{V}$.

Question 3.8. *Montrer qu'on peut supposer que* $\mathcal{V}$ *est réduite à une formule, c'est-à-dire qu'elle est de la forme* $\mathcal{V} = \{\phi\}$ *pour une certaine formule* ϕ*.*

Solution : La théorie $\mathcal{V} = \{\phi_1, \ldots, \phi_n\}$ a les mêmes modèles que $\mathcal{V}' = \{\bigwedge_{i=1}^n \phi_i\}$. □

On considère la théorie $\mathcal{W} = \mathcal{U} \cup \{\neg\phi\}$ (on rappelle que $\mathcal{U}$ est la théorie de la question 3.5 et ϕ est la formule de la question précédente).

Question 3.9. *Montrer que l'existence de la théorie $\mathcal{V}$ implique que théorie $\mathcal{W}$ n'a pas de modèle.*

Solution : Par définition, cette théorie n'a pas de modèle : un modèle satisfait $\mathcal{U}$, c'est donc un groupe sans torsion, et $\neg\phi$ c'est n'est donc pas un groupe sans torsion. □

Question 3.10. *Montrer qu'il existe un modèle de $\mathcal{W}$.*

Solution : Par le théorème de compacité, il suffit de montrer que tout sous-ensemble fini de $\mathcal{W}$ a un modèle. Considérons $\mathcal{W}' \subseteq \mathcal{W}$ fini. On a

$$\mathcal{W} = \mathcal{U} \cup \{\neg\phi\} = \mathcal{T} \cup \{\psi_1, \psi_2, \ldots\} \cup \{\neg\phi\}$$

On note n un entier premier tel que ψ_k n'apparaît pas dans $\mathcal{W}'$ pour $k \geq n$. Considérons un modèle G de $\mathcal{U}_n$: un tel modèle existe d'après la remarque qui suit la question 3.7. G
- valide $\mathcal{T}$ car $\mathcal{T} \subseteq \mathcal{U}_n$,
- valide les ψ_k pour $k < n$ (car $\psi_k \in \mathcal{U}_n$),
- n'est pas sans torsion donc ne valide pas ϕ donc valide $\neg\phi$.

G est donc un modèle de $\mathcal{W}'$. □

Question 3.11. *Conclure qu'il n'existe pas de théorie avec un nombre fini d'axiomes dont les modèles sont les groupes sans torsion.*

Solution : On a raisonné par l'absurde et montré que $\mathcal{W}$ n'avait pas de modèle et en avait. Contradiction. □

4 Problème de correspondance de Post et langages non contextuels

Le *problème de correspondance de Post* est le problème de décision suivant :

Donnée : Une liste $(s_1, t_1), \ldots, (s_n, t_n)$ de n paires de mots sur un alphabet Σ.

Réponse : Décider s'il existe une liste $i_1, \ldots, i_k$ d'indices telle que $s_{i_1} s_{i_2} \cdots s_{i_k} = t_{i_1} t_{i_2} \cdots t_{i_k}$.

Par exemple, étant données les paires suivantes de mots sur l'alphabet $\Sigma = \{a, b\}$

$$(aba, ab), (ba, aaba), (b, bab), (baba, ab)$$

la réponse est « oui », car la liste des indices $1, 4, 1, 2, 3$ donne une correspondance :

aba	$baba$	aba	ba	b	$=$	$abababaababab$
ab	ab	ab	$aaba$	bab		$abababaababab$

Question 4.1. *Montrer que le problème de correspondance de Post est décidable dans le cas d'un alphabet unaire $|\Sigma| = 1$.*

Solution : La liste de paires $(s_1, t_1), \ldots, (s_n, t_n)$ étant finie, on peut rapidement identifier dans laquelle des trois situations mutuellement exclusives suivantes on se trouve :
- Il existe un index i tel que $s_i = t_i$. Dans ce cas la réponse est trivialement "Oui", car la liste singleton i donne une correspondance.
- $|s_i| < |t_i|$ pour tout $i \in \{1, \ldots, n\}$, ou réciproquement $|s_i| > |t_i|$ pour tout i. Dans ce cas la réponse est "Non", car forcement $|s_{i_1} s_{i_2} \cdots s_{i_k}| < |t_{i_1} t_{i_2} \cdots t_{i_k}|$, ou réciproquement $|s_{i_1} s_{i_2} \cdots s_{i_k}| > |t_{i_1} t_{i_2} \cdots t_{i_k}|$, pour n'importe quelle liste d'indices $i_1, \ldots, i_k$.

— Il existe deux indices i et j tel que $|s_i| < |t_i|$ et $|s_j| > |t_j|$. Dans ce cas la réponse est "Oui", car en effet, si on note les différences $p = |t_i| - |s_i|$ et $q = |s_j| - |t_j|$, alors la liste

$$\underbrace{i, \ldots, i}_{q \text{ fois}}, \underbrace{j, \ldots, j}_{p \text{ fois}}$$

donne une correspondance.

□

Nous admettrons dans la suite que le problème de correspondance de Post est indécidable si $|\Sigma| \geq 2$.

Comme mentionné en PC4, les langages *non contextuels* sont une généralisation des langages rationnels, qui incluent par exemple le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ et le langage des mots bien parenthésés. Un langage non contextuel est généré par une *grammaire non contextuelle* $G = (\Sigma, N, S, P)$ composée :

— d'un ensemble fini Σ dont les éléments sont appelés *terminaux* ;
— d'un ensemble fini N (disjoint de Σ) dont les éléments sont appelés *non-terminaux* ;
— d'un non-terminal distingué $S \in N$ appelé le *symbole initial* ;
— et d'une liste P de *productions* de la forme $R \to \alpha$ où $R \in N$ est un non-terminal et $\alpha \in (\Sigma \cup N)^*$ est un mot sur les terminaux et les non-terminaux.

On définit une relation $\Rightarrow$ entre mots dans $(\Sigma \cup N)^*$ comme suit : $\beta \Rightarrow \beta'$ s'il existe des mots $\gamma, \delta \in (\Sigma \cup N)^*$ et une production $R \to \alpha$ dans P telle que $\beta = \gamma R \delta$ et $\beta' = \gamma \alpha \delta$. Le langage engendré par la grammaire G est l'ensemble des mots

$$L(G) = \{w \mid w \in \Sigma^*, S \Rightarrow^+ w\}$$

où $\Rightarrow^+$ est la fermeture transitive de $\Rightarrow$.

Exemple 1. *Le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ est généré par une grammaire avec un seul non-terminal S et deux productions*

$$S \to aSb \qquad\qquad S \to \epsilon$$

où ϵ est le mot vide.

Exemple 2. *Le langage des mots bien parenthésés est généré par une grammaire avec un non-terminal S et deux productions*

$$S \to (S)S \qquad\qquad S \to \epsilon$$

Le *problème d'intersection des langages non contextuels* est le problème de décision suivant :

Donnée : Deux grammaires non contextuelles $G_1 = (\Sigma, N_1, S_1, P_1)$ et $G_2 = (\Sigma, N_2, S_2, P_2)$ sur le même alphabet de terminaux Σ.

Réponse : Décider s'il existe un mot $w \in \Sigma^*$ telle que $w \in L(G_1) \cap L(G_2)$.

Question 4.2. *Montrer que le problème d'intersection des langages non contextuels est indécidable si on ne met pas de contrainte sur l'alphabet Σ, par réduction à partir du problème de correspondance de Post.*

Solution : Étant donnée une liste de paires $(s_1, t_1), \ldots, (s_n, t_n)$ de mots sur un alphabet Σ, on construit deux grammaires $G_1 = (\Sigma', N_1, S_1, P_1)$ et $G_2 = (\Sigma', N_2, S_2, P_2)$ comme suit :

— Pour l'alphabet des terminaux, on prend l'alphabet auquel on a ajouté n symboles distincts, $\Sigma' = \Sigma \uplus \{x_1, \ldots, x_n\}$.

— Les deux grammaires ont un seul non-terminal nommé S, qui est donc le symbole initial. C'est-à-dire, on prend $N_1 = N_2 = \{S\}$ et $S_1 = S_2 = S$.
— Pour les listes de productions P_1 et P_2, on prend respectivement :

$$S \to x_i S s_i \qquad\qquad S \to x_i s_i$$

et

$$S \to x_i S t_i \qquad\qquad S \to x_i t_i$$

avec une paire de productions dans chaque grammaire pour chaque couple indexé (s_i, t_i) de la liste d'entrée.

Par construction, on voit que la liste d'indices $(i_1, \ldots, i_k)$ donne une correspondance $s_{i_1} \ldots s_{i_k} = t_{i_1} \ldots t_{i_k}$ si et seulement s'il existe un mot $w = x_{i_k} \ldots x_{i_1} s_{i_1} \ldots s_{i_k} = x_{i_k} \ldots x_{i_1} t_{i_1} \ldots t_{i_k}$ dans l'intersection $L(G_1) \cap L(G_2)$. Nous avons donc réussi à réduire le problème de correspondance de Post au problème d'intersection des langages non contextuels, ce qui établit l'indécidabilité de ce dernier. (Notons au passage que les grammaires que nous avons construites sont linéaires, au sens défini ci-dessous. Cette preuve établit donc que le problème d'intersection des langages non contextuels est indécidable même dans le cas linéaire, si on ne met pas de contrainte sur l'alphabet.) □

Question 4.3. *Utiliser le résultat précédent pour conclure que les langages non contextuels ne sont pas clos par intersection de façon calculable, c'est-à-dire qu'il n'existe pas d'algorithme qui, étant données deux grammaires non contextuelles G_1 et G_2, renvoie une grammaire non contextuelle G telle que $L(G) = L(G_1) \cap L(G_2)$. (En fait, il existe des langages non contextuels L_1 et L_2 tels que $L_1 \cap L_2$ n'est pas non contextuel, mais nous ne vous demandons pas de le montrer.)*

Solution : En conséquence du résultat précédent, il suffit de montrer qu'il est décidable de savoir si une grammaire non contextuelle G génère un langage non vide $L(G) \neq \emptyset$. En effet, s'il existait un algorithme pour calculer l'intersection G de deux grammaires G_1 et G_2, on pourrait obtenir un algorithme pour décider le problème de correspondance de Post en prenant les grammaires G_1 et G_2 construites ci-dessus et en testant $L(G) \neq \emptyset$.

Étant donnée une grammaire non contextuelle $G = (\Sigma, N, S, P)$ arbitraire, pour décider si $L(G) \neq \emptyset$, on peut utiliser une procédure itérative qui marque progressivement les non-terminaux comme "utiles" :

1. Au début, on marque seulement les non-terminaux R pour lesquels il y a une production de la forme $R \to w$ où w est un mot sur les terminaux dans Σ.
2. Ensuite, pour chaque non-terminal $R \in N$ non marqué, on considère chaque production $R \to \alpha$ et on factorise le membre droit $\alpha = w_0 R_1 w_1 \ldots R_n w_n$ en une liste de mots des terminaux $w_0, \ldots, w_n$ séparés par des non-terminaux $R_1, \ldots, R_n$. Si tous les $R_1, \ldots, R_n$ sont déjà marqués utiles, alors on marque R utile.
3. On répète l'étape (2) tant qu'il y a des non-terminaux nouvellement marqués utiles.

Remarquons que l'étape (2) peut être répétée au plus $|N|$ fois, et qu'à la fin du processus, un non-terminal R aurait été marqué utile si et seulement s'il existe un mot w tel que $R \Rightarrow^+ w$ (ce qu'on peut montrer par induction). Pour décider si $L(G)$ est non vide, il suffit donc de regarder si le symbole initial S a été marqué utile. □

Une grammaire est dite *linéaire* si pour chaque production $R \to \alpha$, le membre droit α contient au plus un non-terminal.

Question 4.4. *Montrer que le problème d'intersection est décidable pour les langages non contextuels linéaires sur un alphabet unaire $|\Sigma| = 1$.*

Solution : Tout d'abord il faut observer que l'opération de concaténation des mots sur un alphabet unaire est toujours commutative (on peut le voir comme l'addition des entiers). Cette commutativité implique qu'une grammaire linéaire peut être transformée en une autre grammaire linéaire générant le même langage mais avec la propriété supplémentaire que pour chaque production qui contient un non-terminal dans le membre droit, le non-terminal se trouve au début (c'est-à-dire, le plus à gauche) : il faut juste remplacer chaque production de la forme $R \to uR'v$ par $R \to R'uv$, ce qui ne change pas le langage en raison de la commutativité. Le résultat découle alors :

1. du fait que le langage engendré par une telle grammaire *linéaire gauche* est rationnel ;
2. du fait que les langages rationnels sont clos par intersection ;
3. et du fait qu'il est décidable si un langage rationnel est non vide.

Plus précisément, à partir d'une grammaire linéaire gauche G on peut facilement construire un automate fini non déterministe avec ϵ-transitions qui reconnaît $L(G)$:

— Il y a un état initial q_0 et un état q_R pour chaque non-terminal R, plus quelques états supplémentaires décrits ci-dessous.

— Chaque production de la forme $R \to R'a_1 \dots a_n$ ou $R \to a_1 \dots a_n$ avec $n > 0$ est traduite respectivement par une suite de n transitions

$$q_{R'} \xrightarrow{a_1} q'_1 \xrightarrow{a_2} \dots \xrightarrow{a_{n-1}} q'_{n-1} \xrightarrow{a_n} q_R$$

ou

$$q_0 \xrightarrow{a_1} q'_1 \xrightarrow{a_2} \dots \xrightarrow{a_{n-1}} q'_{n-1} \xrightarrow{a_n} q_R$$

via $n-1$ états supplémentaires $q'_1, \dots, q'_{n-1}$.

— Chaque production de la forme $R \to R'$ ou $R \to \epsilon$ est traduite respectivement par une ϵ-transition $q_{R'} \to q_R$ ou $q_0 \to q_R$.

— On prend q_S pour l'état acceptant.

Cet automate peut ensuite être déterminiser de façon standard. Après avoir construit deux automates finis A_1 et A_2 qui reconnaissent les langages $L(G_1)$ et $L(G_2)$ générés par deux grammaires linéaires G_1 et G_2 sur un alphabet unaire, on peut construire un automate fini A qui reconnaît l'intersection, dont les états sont des paires (q_1, q_2) d'états $q_1 \in A_1$ et $q_2 \in A_2$, et avec une transition $(q_1, q_2) \xrightarrow{a} (q'_1, q'_2)$ pour chaque paire de transitions $q_1 \xrightarrow{a} q'_1 \in A_1$ et $q_2 \xrightarrow{a} q'_2 \in A_2$. Enfin, pour décider si cet automate A reconnaît un langage non vide il suffit de parcourir le graphe de transition pour voir s'il y a un chemin de l'état initial vers l'état d'acceptation.

(Notons au passage que le problème d'intersection des langages non contextuels est décidable dans le cas d'un alphabet unaire, même sans la condition de linéarité. C'est un corollaire du théorème de Parikh, qui dit que les langages non contextuels sont équivalents aux langages rationnels si l'on oublie l'ordre et ne compte que le nombre d'occurrences de lettres dans les mots. Pourtant, la démonstration du théorème de Parikh n'est pas triviale.) □