École polytechnique — X2018 — INF412 Fondements de l'informatique

PC6 notée

Sujet proposé par Samuel Mimram et Amaury Pouly et Bruno Salvy
(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 Graphes connexes

On considère dans cette section des graphes non-orientés, c'est-à-dire des paires d'ensembles (V, E) avec $E \subseteq V \times V$ telles que si $(x, y) \in E$ alors $(y, x) \in E$. On rappelle que la longueur d'un chemin est le nombre de ses arêtes (c'est un entier naturel) et qu'un graphe est connexe lorsqu'il existe un chemin entre toute paire de sommets.

Nous allons montrer qu'il n'existe pas de théorie du premier ordre dont les modèles sont les graphes connexes. Nous raisonnons par l'absurde et supposons qu'il existe une théorie du premier ordre $\mathcal{C}$ sur la signature constituée d'un unique symbole de relation binaire E, dont les modèles sont les graphes connexes.

Question 1.1. *Décrivez*

1. *étant donné $n \in \mathbb{N}$, une théorie du premier ordre $\mathcal{T}_n$ dont les modèles sont les triplets (G, a, b) où G est un graphe connexe et a et b sont des sommets de G tels qu'il n'existe pas de chemin de longueur au plus n de a à b,*
2. *une théorie du premier ordre $\mathcal{T}$ dont les modèles (s'il en existe) sont les triplets (G, a, b) où G est un graphe connexe et a et b sont des sommets de G tels qu'il n'existe aucun chemin de a à b.*

Solution :

1. On considère la théorie obtenue à partir de $\mathcal{C}$ en ajoutant deux symboles de constantes a et b, ainsi que les axiomes (C_i), pour $0 \leq i \leq n$ qui permettent de s'assurer qu'il n'y a pas de chemin de longueur i de a à b. Pour un entier naturel i, l'axiome (C_i) est défini par

$$\neg(\exists x_0.\exists x_1 \ldots \exists x_n.a = x_0 \wedge x_n = b \wedge E(x_0, x_1) \wedge E(x_1, x_2) \wedge \ldots \wedge E(x_{n-1}, x_n))$$

2. Il suffit de prendre la théorie définie de la même façon que précédemment, mais en prenant tous les (C_i), pour $i \in \mathbb{N}$, comme axiomes.

□

Question 1.2. *Montrez que la théorie $\mathcal{T}_n$ admet un modèle, pour tout $n \in \mathbb{N}$.*

Solution : On considère le graphe G avec $\mathbb{N}$ comme ensemble de sommets et dont les arêtes de la forme $(i, i+1)$ pour $i \in \mathbb{N}$. Ce graphe est clairement connexe : étant donnés deux sommets i et j avec $i \leq j$, on a le chemin $i, i+1, i+2, \ldots, j$. En interprétant a par le sommet 0 et b par le sommet $n+1$, on obtient un modèle de la théorie $\mathcal{T}_n$. □

Question 1.3. *Montrez que la théorie $\mathcal{T}$ admet un modèle.*

Solution : Étant donnée une sous-théorie $\mathcal{T}'$ de $\mathcal{T}$ avec un ensemble fini d'axiomes, il existe un entier n tel que $\mathcal{T}'$ ne contient des axiomes C_i que pour $i \leq n$. La théorie $\mathcal{T}'$ est alors une sous-théorie de $\mathcal{T}_n$ et admet donc un modèle par la question 1.2. Par le théorème de compacité, on déduit l'existence d'un modèle de $\mathcal{T}$. □

Question 1.4. *Montrez qu'il ne peut pas exister de théorie $\mathcal{C}$ dont les modèles sont les graphes connexes.*

Solution : On a montré à la question précédente l'existence d'un modèle pour $\mathcal{T}$, or cette théorie n'admet pas de modèle car l'hypothèse qu'il n'existe pas de chemin de a à b contredit la connexité du graphe. □

2 Une variante des machines de Turing

Les machines de Turing définies dans le cours ont une fonction de transition

$$\delta : Q \times \Gamma \to \Gamma \times \{\leftarrow, |, \rightarrow\} \times Q$$

qui, à un état et un caractère présent sur le ruban, associe un caractère à écrire sur le ruban, un déplacement de la tête de lecture et un nouvel état. Le cycle de travail est donc :

lecture → écriture → déplacement → transition vers un autre état.

On s'intéresse dans cet exercice à une variante de machines dont le cycle de travail serait

écriture → déplacement → lecture → transition vers un autre état,

avec une fonction de transition de signature

$$\delta' : Q \to \Gamma \times \{\leftarrow, |, \rightarrow\} \times (\Gamma \to Q)$$

qui écrit un caractère, déplace la tête de lecture, et change d'état en fonction du caractère qui se trouve alors sous la tête de lecture. Initialement la tête de lecture est toujours sur le premier caractère blanc (B) à gauche du mot à traiter. On appelera ces machines des machines de Turing modifiées.

Question 2.1. *En se limitant au cas où l'alphabet de travail Γ est réduit à $\{a, b, B\}$, écrire une machine de Turing modifiée qui déplace la tête de lecture jusqu'au premier b sur la droite de sa position initiale, sans modifier le mot. (Ces machines ne seront pas représentées graphiquement ; donner juste l'ensemble des états et la fonction de transition.)*

Solution : La machine a quatre états :

1. un état initial q_0, pour lequel la fonction de transition vaut $\delta'(q_0) = (B, \rightarrow, \phi)$ avec $\phi(a) = q_1$, $\phi(b) = q_a$, $\phi(B) = q_r$;
2. un état q_1 dans lequel la machine arrive après avoir lu un a, et $\delta'(q_1) = (a, \rightarrow, \phi)$;
3. un état d'acceptation q_a avec $\delta'(q_a) = (b, |, \psi)$ avec $\psi(b) = q_a$;
4. un état de rejet q_r avec $\delta'(q_r) = (B, |, \chi)$ et $\chi(B) = q_r$.

□

Question 2.2. *Montrer comment simuler une machine de Turing modifiée par une machine de Turing.*

Solution : Il s'agit de décaler les opérations de sorte que la lecture ait lieu en premier. Pour tout état $q \in Q$, on note $\delta'(q) = (\ell_q, d_q, \phi_q)$. La machine de Turing possède un état Q_q pour chaque $q \in Q$ et a une fonction de transition définie par

$$\delta(Q_q, m) = (\ell_{\phi_q(m)}, d_{\phi_q(m)}, Q_{\phi_q(m)}), \quad \text{pour tous } q \in Q, m \in \Gamma.$$

Enfin, on ajoute un état initial Q_0 avec fonction de transition $\delta(Q_0, B) = (Q_{q_0})$ pour traiter séparément la première action de la machine. On vérifie que les transitions effectuées sont les mêmes, ainsi que l'état d'acceptation et celui de rejet. □

Question 2.3. *Montrer comment simuler une machine de Turing (dont la tête de lecture est initialement sur le premier caractère blanc (B) à gauche du mot à traiter) par une machine de Turing modifiée. [Indication : la machine de Turing modifiée peut avoir plus d'états que la machine de Turing initiale.]*

Solution : Soit $Q := \{q_0, q_1, \ldots, q_k\}$ l'ensemble des états de la machine de Turing et δ sa fonction de transition. Une solution consiste à créer une machine de Turing modifiée avec $|\Gamma| \times 3 \times k$ états pour mémoriser ce que la machine de Turing vient de lire. En nommant ces états $Q_{\ell,d,q}$ pour $\ell \in \Gamma$, $d \in \{\leftarrow, |, \rightarrow\}$ et $q \in Q$, la fonction de transition est définie par

$$\delta'(Q_{\ell,d,q}) = \left(\ell, d, \left\{u \mapsto Q_{\delta(q,u)} \mid u \in \Gamma\right\}\right).$$

Pour que les états Q_{ℓ,d,q_a} où q_a est l'état d'acceptation soient considérés comme des états d'acceptation, on rajoute les transitions

$$\delta'(Q_{\ell,d,q_a}) = Q_a,$$

où Q_a est un nouvel état, d'acceptation. On procède de même pour les états de rejet. Enfin, l'état initial est $Q_{B,\rightarrow,q_0}$. □

3 Problèmes de décision

Parmi les problèmes suivants, lesquels sont décidables, lesquels sont indécidables, et lesquels sont semi-décidables ? (Argumentez votre réponse).

Question 3.1. *Déterminer si une machine de Turing M accepte au moins 2019 mots.*

Solution : La propriété est non triviale : Σ^* la possède et le langage vide ne la possède pas. Le théorème de Rice permet de conclure à l'indécidabilité.

La propriété est quand même semi-décidable : il suffit de simuler la machine sur des entrées de longueur croissante pendant des nombres d'étapes croissants et de stopper dès que 2019 mots ont été acceptés. □

Question 3.2. *Déterminer si une machine de Turing M accepte au moins 2019 mots en au plus 2020 étapes.*

Solution : Cette propriété est décidable : en au plus 2020 étapes, la machine ne peut lire que des entrées de longueur au plus 2020. Il suffit donc de simuler la machine pendant 2020 étapes sur toutes les entrées de longueur 2020 (qui sont en nombre fini puisque l'alphabet Σ est fini), et de compter le nombre de mots acceptés, pour enfin le comparer à 2019. □

Question 3.3. *Déterminer si le langage reconnu par une machine de Turing M est reconnu par une machine (éventuellement différente de M) dont le nombre d'états est un multiple de 2019.*

Solution : La réponse est que c'est décidable, car c'est tout le temps vrai : on construit une machine de Turing M' qui est une copie de M avec des états supplémentaires inutiles pour arriver à un nombre d'états multiple de 2019. □

4 Systèmes d'étiquettes

Les systèmes d'étiquettes [1] sont des machines très simples inventées par Emil Post en 1943. Malgré leur apparente simplicité, nous allons voir que leurs comportements peuvent être très compliqués.

Un système d'étiquettes est défini par un triplet (m, Σ, P) où :
— m est un entier naturel indiquant le nombre de lettres à effacer à chaque étape ;
— Σ est l'alphabet ;
— $P : \Sigma \to \Sigma^*$ est une règle de production associant un mot à chaque lettre.

Le paramètre m va jouer un rôle crucial, ainsi un système de paramètre m est appelé un m-système. Un calcul fonctionne de la façon suivante : à partir d'un mot initial (« l'entrée »), on modifie le mot à chaque étape en utilisant la règle de production. Le système s'arrête lorsque la taille du mot devient strictement plus petite que m. Comme pour une machine de Turing, il est possible pour un système de ne jamais s'arrêter. Plus précisémment, à chaque étape :
— on regarde la première lettre $a \in \Sigma$ du mot ;
— on supprime m lettres au début du mot ;
— on ajoute $P(a)$ à la fin du mot.

Si un mot w se transforme en w' après une étape de calcul, on note $w \vdash w'$. Si w se transforme en w' après k étapes de calculs, on note $w \vdash^k w'$. S'il existe k tel que w se transforme en w' après k étapes de calculs, on note $w \vdash^* w'$. Enfin on note ε le mot vide.

Voici un exemple de 3-système sur l'alphabet $\Sigma = \{a, b\}$ avec les règles de transformation $P(a) = aa$ et $P(b) = bbab$ et partant de l'entrée $aaaba$. Pour simplifier la lecture, on a mis en gras les 3 lettres effacées à chaque étape.

$$\mathbf{aaa}ba \vdash \mathbf{baa}a \vdash \mathbf{abb}ab \vdash \mathbf{aba}a \vdash \mathbf{aaa} \vdash aa \quad .$$

Le problème de l'arrêt pour les m-systèmes consiste à déterminer, étant donnés la description d'un m-système et un mot w, s'il existe $k \in \mathbb{N}$ tel que $w \vdash^k w'$ avec $|w'| < m$. Le but de cet exercice est de montrer que le problème de l'arrêt pour les m-systèmes est décidable pour $m = 1$ et indécidable pour $m \geq 2$.

4.1 Indécidabilité des 2-systèmes

Nous allons maintenant montrer que l'on peut simuler une machine de Turing par un 2-système. Pour cela, on considère des machines à deux compteurs a et b. Initialement, $b = 0$ et $a \in \mathbb{N}$ est l'entrée. Une telle machine comporte un nombre fini L d'instructions. Pour $i \in \{1, \ldots, L\}$, l'instruction i est de l'une des formes

1. $\mathsf{Incr}(c, j)$ qui incrémente $c \in \{a, b\}$ puis va à l'instruction $j \neq i$;
2. $\mathsf{Decr}(c, j)$ qui décrémente $c \in \{a, b\}$, puis va à l'instruction $j \neq i$; **cette instruction ne peut être utilisée que lorsque** $c > 0$;
3. $\mathsf{IsZero}(c, j, k)$ qui teste si $c \in \{a, b\}$ est nul, va à l'instruction $j \neq i$ si c'est le cas, et à l'instruction $k \neq i$ sinon ;
4. Halt qui arrête le calcul.

Pour chaque instruction $i \in \{1, \ldots, L\}$, on introduit les lettres $a_i, b_i, A_i, B_i, a'_i, b'_i, A'_i, B'_i, Z_i$ et on ajoute aussi le symbole $*$. L'alphabet est donc

$$\Sigma = \{*\} \cup \{a_i, b_i, A_i, B_i, a'_i, b'_i, A'_i, B'_i, Z_i : i = 1, \ldots, L\}.$$

1. En anglais « tag system ».

Étant donné un état (a, b, i) d'une machine à compteur, un encodage valide de cet état est n'importe quel mot de la forme

$$\overbrace{\mathsf{a}_i? \cdots \mathsf{a}_i?}^{2^a \text{ fois}} A_i? \overbrace{\mathsf{b}_i? \cdots \mathsf{b}_i?}^{2^b \text{ fois}} B_i?$$

où chaque ? peut-être remplacé par une lettre quelconque. Par exemple, le mot

$$\underbrace{a_i*}_{a_i?}\underbrace{a_i a_i}_{a_i?}\underbrace{a_i a_j}_{a_i?}\underbrace{a_i B_j}_{a_i?}\underbrace{A_i A_i}_{A_i?}\underbrace{b_i*}_{b_i?}\underbrace{b_i b_j}_{b_i?}\underbrace{B_i*}_{B_i?}$$

est un encodage valide de $(2, 1, i)$.

Question 4.1. *Montrer que si l'instruction i est $\mathsf{Incr}(a, j)$ alors les règles $P(a_i) = a_j * a_j *$, $P(A_i) = A_j *$, $P(b_i) = b_j *$ et $P(B_i) = B_j *$ sont correctes : tous les encodages valides de l'état (a, b, i) se transforment (en plusieurs étapes) en un encodage valide de $(a + 1, b, j)$.*

Solution : Après application des règles $2^a + 2^b + 2$ fois, on obtient le mot

$$\overbrace{a_j * \cdots a_j *}^{2^{a+1} \text{ fois}} A_j * \overbrace{b_j * \cdots b_j *}^{2^b \text{ fois}} B_j *$$

qui encode bien $(a + 1, b, j)$. □

Question 4.2. *Proposer une règle pour les instructions Halt qui produit un mot vide.*

Solution : On prend la règle $P(x_i) = \varepsilon$ pour toutes les lettres x. Ainsi on va effacer le mot par la gauche jusqu'à ce qu'il soit vide. □

Question 4.3. *Proposer une règle pour les instructions $\mathsf{Decr}(c, j)$, on rappelle que l'on suppose que $c > 0$. Quel problème se pose lorsque $c = 0$? [Indication : souvenez-vous que « ? » peut être n'importe quelle lettre dans l'encodage.]*

Solution : Si $c = b$ par exemple, on prend les règles $P(a_i) = a_j *$, $P(A_i) = A_j *$, $P(b_i) = b_j$ et $P(B_i) = B_j *$. Après application des règles sur tout le mot, on obtient

$$\overbrace{a_j * \cdots a_j *}^{2^a \text{ fois}} A_j * \overbrace{b_j \cdots b_j}^{2^{b-1} \text{ fois}} B_j *$$

qui encode bien $(a, b - 1, j)$ puisque un b_j sur deux correspond à un ? dans l'encodage. Lorsque $b = 0$, nos règles produisent

$$\overbrace{a_j * \cdots a_j *}^{2^a \text{ fois}} A_j * b_j B_j *$$

qui n'est pas un encodage valide : le mot est de longueur impaire, on s'attend à avoir $b_j? B_j?$ à la fin mais on a $b_j B_j?$. □

On suppose que l'instruction i est $\mathsf{IsZero}(b, j, k)$ et on introduit les règles

$$\begin{array}{llll} P(a_i) = a'_i * \quad , & P(A_i) = A'_i * \quad , & P(b_i) = b'_i \quad , & P(B_i) = B'_i Z_i \quad , \\ P(a'_i) = a_j a_k \quad , & P(A'_i) = A_j A_k \quad , & P(b'_i) = b_j b_k \quad , & P(B'_i) = B_j * \quad , \\ P(Z_i) = b_k B_k * \quad . & & & \end{array}$$

Question 4.4. *Montrer que si l'état est un encodage de (a, b, i) avec $b > 0$ alors les règles ci-dessus produisent bien un encodage de (a, b, j).*

Solution : Après application des règles sur tout le mot, on obtient

$$\overbrace{a'_i * \cdots a'_i *}^{2^a \text{ fois}} A'_i * \overbrace{b'_i \cdots b'_i}^{2^b \text{ fois}} B'_i Z_i$$

dont on note qu'il est de longueur paire car 2^b est pair, car $b > 0$. En appliquant les règles une fois de plus sur tout le mot, on obtient

$$\overbrace{a_j a_k \cdots a_j a_k}^{2^a \text{ fois}} A_j A_k \overbrace{b_j b_k \cdots b_j b_k}^{2^b \text{ fois}} B_j *$$

qui encode bien (a, b, j). □

Question 4.5. *Montrer que si l'état est en encodage de $(a, 0, i)$ alors les règles ci-dessus produisent bien un encodage de $(a, 0, k)$. [Indication : remarquer que les règles produisent un mot de longueur impaire, ce qui change l'interprétation de l'encodage.]*

Solution : En partant d'un mot de la forme

$$\overbrace{\mathtt{a}_i ? \cdots \mathtt{a}_i ?}^{2^a \text{ fois}} A_i ? \mathtt{b}_i ? B_i ?$$

on obtient

$$\overbrace{a'_i * \cdots a'_i *}^{2^a \text{ fois}} A'_i * b'_i B'_i Z_i$$

dont on note qu'il est de longueur impaire. Ainsi après $2^a + 1$ étapes on obtient

$$b'_i B'_i Z_i \overbrace{a_j a_k \cdots a_j a_k}^{2^a \text{ fois}} A_j A_k$$

qui devient

$$Z_i \overbrace{a_j a_k \cdots a_j a_k}^{2^a \text{ fois}} A_j A_k b_j b_k$$

qui produit (attention au décalage, on supprime $Z_i a_j$)

$$\underbrace{\overbrace{a_k a_j \cdots a_k a_j}^{2^a - 1 \text{ fois}} a_k A_j}_{2^a \text{ fois } a_k ?} \underbrace{A_k b_j}_{A_k ?} \underbrace{b_k b_k}_{b_k ?} \underbrace{B_k *}_{B_k ?}$$

qui est de la forme

$$\overbrace{\mathtt{a}_k ? \cdots \mathtt{a}_i ?}^{2^a \text{ fois}} A_k ? \mathtt{b}_k ? B_k ?$$

et donc encode $(a, 0, k)$. □

Question 4.6. *Conclure.*

Solution : Le problème de l'arrêt pour les machines à deux compteurs du sujet est indécidable. En effet, par rapport aux machines à deux compteurs « standards », nous avons seulement ajouté la condition que $\mathsf{Decr}(c, j)$ ne peut être utilisé que lorsque $c > 0$. On peut facilement modifier une machine standard pour que ce soit le cas : on remplace chaque instruction $\mathsf{Decr}(c, j)$ par un test à zéro (IsZero) puis on ne décrémente que si $c > 0$.

Or nous avons construit un 2-système qui simule une machine à deux compteurs et produit le mot vide si et seulement si la machine s'arrête. Donc le problème de l'arrêt pour les 2-systèmes est indécidable. □

4.2 Décidabilité des 1-systèmes

Considérons un 1-système sur l'alphabet Σ avec une fonction de production P. Pour chaque mot $w \in \Sigma^*$ et lettre $x \in \Sigma$, on note $|w|_x$ le nombre de fois où la lettre x apparait dans le mot w. Par exemple $|aabcac|_a = 3$. Afin de simplifier les notations, on suppose que $\Sigma = \{a_1, \ldots, a_n\}$. Soit $f : \Sigma^* \to \mathbb{N}^n$ la fonction définie par $f(w) = (|w|_{a_1}, \ldots, |w|_{a_n})$. Par exemple si $\Sigma = \{a, b, c\}$, alors $f(w) = (|w|_a, |w|_b, |w|_c)$ et donc $f(aabcac) = (3, 1, 2)$.

Question 4.7. *Montrer qu'il existe une fonction linéaire $g : \mathbb{N}^n \to \mathbb{N}^n$, qui ne dépend que de P, telle que pour tout mot w, si $w \vdash^{|w|} w'$ alors $f(w') = g(f(w))$.*

Solution : On pose $g(v)_i = \sum_{j=1}^n v_j |P(a_j)|_{a_i}$, qui est clairement linéaire en v. Dans un 1-système, un mot w se transforme, après $|w|$ étapes, en $w' = P(w_1) \cdots P(w_{|w|})$. On a donc

$$f(w')_i = \sum_{k=1}^{|w|} |P(w_k)|_{a_i} = \sum_{j=1}^{n} \sum_{k : w_k = a_j} |P(w_k)|_{a_i} = \sum_{j=1}^{n} |w|_{a_j} |P(a_j)|_{a_i} = g(f(w))_i \ .$$

□

On note $B = \{0, 1\}$ l'ensemble des booléens, muni des opérations usuelles (conjonction, disjonction, négation). Soit $\pi : \mathbb{N} \to B$ la projection définie par $\pi(n) = 1$ si $n > 0$ et $\pi(0) = 0$. On étend π aux vecteurs composante par composante, ainsi $\pi : \mathbb{N}^n \to B^n$ satisfait $\pi(v_1, \ldots, v_n) = (\pi(v_1), \cdots, \pi(v_n))$. Par exemple, $\pi(4, 0, 1) = (1, 0, 1)$.

Question 4.8. *Montrer qu'il existe une fonction $h : B^n \to B^n$, telle que pour tout $v \in \mathbb{N}^n$, $\pi(g(v)) = h(\pi(v))$. [Indication : étudier $\pi(n + m)$ et $\pi(nm)$ pour $n, m \in \mathbb{N}$, puis utiliser la linéarité de g.]*

Solution : On observe immédiatement que si $n, m \in \mathbb{N}$ alors $\pi(n + m) = \pi(n) \vee \pi(m)$ et $\pi(nm) = \pi(n) \wedge \pi(m)$. Soit $e_1, \ldots, e_n$ la base canonique de $\mathbb{N}^n$. On a alors pour tout vecteur v que $g(v) = \sum_{i=1}^n v_i g(e_i)$. On a donc pour chaque i :

$$\pi(g(v))_i = \pi\left(\sum_{j=1}^{n} v_j g(e_j)_i\right) = \bigvee_{j=1}^{n} \pi(v_j) \wedge \pi(g(e_j)_i) = \bigvee_{j=1}^{n} \pi(v)_j \wedge \pi(g(e_j)_i) = h(\pi(v))$$

où $h(b)_i = \bigvee_{j=1}^n b_j \wedge \pi(g(e_j)_i))$. □

Question 4.9. *Montrer que le problème de l'arrêt des 1-systèmes est décidable.*

Solution : Si $w \vdash^* \varepsilon$ alors il existe une suite $w(0) = w, w(1), \ldots, w(k) = \varepsilon$ telle que

$$w = w(0) \vdash^{|w(0)|} w(1) \vdash^{|w(1)|} w(2) \cdots \vdash^{|w(k-1)|} w(k) = \varepsilon \quad .$$

Mais alors $g(f(w)) = f(w(1))$, donc $g(g(f(w))) = g(f(w(1))) = f(w(2))$, donc $g^{[k]}(f(w)) = f(\varepsilon) = 0$ où $g^{[k]}$ désigne la k^{eme} itérée de g. On se convainc qu'il s'agit d'une équivalence, c'est-à-dire que $w \vdash^* \varepsilon$ si et seulement s'il existe $k \in \mathbb{N}$ tel que $g^{[k]}(f(w)) = 0$. Or pour un vecteur $v \in \mathbb{N}^n$, on a bien que $v = 0$ si seulement si $\pi(v) = 0$. Donc $g^{[k]}(f(w)) = 0$ si et seulement $\pi(g^{[k]}(f(w))) = 0$ si et seulement si $h^{[k]}(\pi(f(w))) = 0$. Il suffit donc déterminer si un tel k existe. Or l'ensemble B^n est fini, de taille 2^n. Donc après 2^n itérations de h sur $\pi(f(w))$, on a soit atteint 0 (le système s'arrête) soit bouclé sur un vecteur non nul (le système ne s'arrête pas). Le problème de l'arrêt est donc bien décidable. □