RAPPORT DE STAGE DE L3
TRANSITIVITÉ ET MINIMALITÉ DES MACHINES DE TURING

∼

SIMON MIRWASSER

Encadré par NICOLAS OLLINGER (LIFO)

Du 5 juin au 19 juillet 2019

Résumé. Ce rapport présente les recherches effectuées au cours d'un stage de six semaines et demi au Laboratoire d'Informatique Fondamentale d'Orléans (LIFO) en fin d'année de L3, sous la direction de Nicolas Ollinger, de l'équipe Graphes, Algorithmique et Modèles de Calcul (GAMoC). Ces recherches abordent des problèmes de décision sur la dynamique des machines de Turing, ainsi que leur position dans la hiérarchie arithmétique.

1. Introduction

En informatique théorique, l'un des modèles de calcul les plus riche est le machine de Turing. C'est une machine abstraite introduite par Allan Turing en 1936, qui considère un ruban infini sur lequel se déplace une tête de lecture, qui lit les symboles et les modifie en fonction des instructions données à la machine. La richesse du modèle permet de démontrer de nombreux résultats, mais l'un des plus classiques est l'indécidabilité du problème de l'arrêt : c'est l'exemple principal de problème indécidable, c'est à dire de problème pour lequel il a été prouvé qu'aucun algorithme ne pourrait le résoudre. L'étude de ces problèmes indécidables à entraîné la conception d'une classification de ces problèmes, la hiérarchie arithmétique, qui sera présentée plus en détails en 2.3

Si de nombreux aspects des machines de Turing sont étudiés, le point de vue adopté ici sera celui des machines de Turing en tant que systèmes dynamiques, qui a été introduit par Moore [Moo91] et Kůrka [Kůr97]. De nombreuses propriétés dynamiques des machines de Turing ont étés étudiées, notamment leur place dans la hiérarchie arithmétique. Ainsi, l'objectif principal ici est de classer dans la cette hiérarchie un problème de transitivité, noté TR, et plus précisément, on cherche à démontrer la conjecture suivante :

Conjecture 1. *Le problème* TR *est* Π_2*-complet*

Ce résultat n'est pas prouvé ici, mais certaines pistes et tentatives sont données : la section 2 est dédiée à définir formellement les outils nécessaires, plusieurs techniques importantes seront données en section 3, des résultats utilisables en tant que lemmes seront présentés en section 4. Enfin, la section 5 reviendra sur les résultats obtenus pour les mettre en relation avec la conjecture précédente.

2. Sujet et premières définitions

2.1. Machines de Turing.

Une machine de Turing est définie formellement ainsi :

Définition 2.1. Une *machine de Turing* $\mathcal{M}$ est un triplet $\mathcal{M} = (Q, \Sigma, \delta)$ où Q est l'ensemble fini des *états* de la machine, Σ est son *alphabet* et $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, 0, 1\}$ est sa *fonction de transition*.

Une *configuration* d'une machine $\mathcal{M}$ est un triplet $(s, c, p) \in Q \times \Sigma^{\mathbb{Z}} \times \mathbb{Z}$ où s est l'état de la machine, c représente le ruban de la machine et p est la position de la tête de lecture. Si (s, c, p) est une configuration telle que $\delta(s, c(p))$ n'est pas défini, c'est une *configuration d'arrêt*

et $(s, c(p))$ est une *paire d'arrêt*. Enfin, lorsque la fonction c est partielle, et qu'elle n'est définie que sur une portion finie du ruban, on dit que (s, c, p) est une *configuration partielle*.

On dit qu'il y a une *transition* de la machine entre une première configuration (s, c, p) et une seconde configuration $(t, c', p + d)$, noté $(s, c, p) \vdash (t, c', p + d)$, si $\delta(s, c(p)) = (t, a, d)$, et si $c'(p) = a$ et $c'(i \neq p) = c(i)$. L'itération de la transition $\vdash$ est notée $\vdash^*$. Dans la suite, on notera aussi une transition (s, c, p) par $(\overset{\cdots\ c(p-1)\ \ c(p)\ \ c(p+1)\ \cdots}{\qquad\quad\ \ p})$

Une machine est dite *mortelle* si toutes ses configurations sont mortelles, c'est à dire que le calcul de la machine sur une telle configuration s'arrête. Une autre propriété importante est la totalité : une machine *totale* est une machine qui s'arrête sur chaque configuration *finie*.

Enfin, une machine *injective* (resp. *surjective*) est une machine dont la fonction de transition δ est injective (resp. injective).

2.1.1. *Machines réversibles.* Lors de l'étude des machines de Turing d'un point de vue dynamique, il est important d'introduire un certain type de machines :

Définition 2.2. Une machine de Turing $\mathcal{M}$ est dite *réversible* lorsque sa fonction de transition est injective, car cette fonction peut alors être inversée.

Ainsi, une machine réversible $\mathcal{M} = (Q, \Sigma, \delta)$, est caractérisée par deux fonctions ρ et μ où $\rho : Q \times \Sigma \to Q \times \Sigma$ est partielle et injective et $\mu : Q \to \{-1, 0, 1\}$ est aussi injective, telles que, si $(s, a) \in Q \times \Sigma$ n'est pas une paire d'arrêt, $\delta(s, a) = (t, b, \mu(t))$, où $\rho(s, a) = (t, b)$.

On définit alors la fonction partielle δ^{-1} telle que $\delta(t, b) = (s, a, -\mu(s))$ si et seulement si $\rho(s, a) = (t, b)$. Ainsi, la machine $\mathcal{M}^{-1} = (Q, \Sigma, \delta^{-1})$ est appelée *machine inverse* de $\mathcal{M}$.

Cette notion de machine inverse permet de définir les *paires de départ* d'une machine $\mathcal{M}$ réversible comme les paires d'arrêt de sa machine inverse $\mathcal{M}^{-1}$

2.1.2. *Définition usuelle.* Il existe différentes définitions de machines de Turing, mais elles ont toutes la même puissance de calcul. Celle donnée dans cette partie est une définition plus usuelle que la précédente, car elle permet de parler de calcul d'une machine de Turing, et du résultat d'un calcul.

Définition 2.3. Une machine de Turing $\mathcal{M}$ est un 6-uplet $(Q, \Sigma, \delta, q_0, q_f, B)$ tel que $\mathcal{M} = (Q, \Sigma \cup B, \delta)$ est une machine (selon la définition 2.1), $q_0,\ q_f \in Q$ sont respectivement les états initiaux et finaux de $\mathcal{M}$ et $B \in \Sigma$ est le symbole blanc.

Remarque. Dans toute la suite, on distinguera les deux différents types de machines de Turing présentés ici en notant $\mathcal{M}$ les machines de la définition 2.1 et $\mathcal{M}$ celles de la définition 2.3

Les configurations sont définies comme pour les machines $\mathcal{M}$, et le langage d'une machine $\mathcal{M}$ est défini par $L(\mathcal{M}) = \{u \in (\Sigma \setminus \{B\})^* \mid (\overset{\cdots\ B\ B\ u_0\ \cdots\ u_n\ B\ B\ \cdots}{\qquad\qquad\ \ q_0}) \vdash (\overset{\cdots\ a\ \cdots}{q_f}) \text{ avec } a \in \Sigma\}$

On définit enfin le résultat d'un calcul d'une machine $\mathcal{M}$ sur un mot $u \in L(\mathcal{M})$, qu'on notera $\mathcal{M}(u)$, par $\mathcal{M}(u) = v_1 \dots v_n \in \Sigma^*$ si et seulement si
$(\overset{\cdots\ B\ B\ u_0\ \cdots\ u_n\ B\ B\ \cdots}{\qquad\qquad\ \ q_0}) \vdash (\cdots\ B\ B\ v_1\ \cdots\ v_n\ B\ B\ \cdots)$ et $v_1, v_2 \neq B$.

2.2. **Problèmes de décision.** L'une des notions centrale étudiée ici est celle de problème informatique, et notamment de problème de décision. Un tel problème est une question à laquelle on peut répondre par "oui" ou par "non"

Pour formaliser cette idée, on peut représenter un problème de décision par un ensemble, et ici seuls des sous-ensembles de $\mathbb{N}$ nous intéresseront. En effet, les machines de Turing étant codables par des entiers (le code d'une machine $\mathcal{M}$ (resp. $\mathcal{M}$) étant noté $< \mathcal{M} >$ (resp. $< \mathcal{M} >$)), si l'on considère le problème de savoir si une machine $\mathcal{M}$ vérifie une propriété P, ce problème se formalise comme $\{< \mathcal{M} > \in \mathbb{N} \mid \mathcal{M} \text{ vérifie } P\}$ (et de même pour les machines $\mathcal{M}$).

Dans la suite, on s'autorisera une autre notation : si $\mathcal{M}$ est une machine, ayant deux arguments (il est possible de considérer qu'une machine calcule sur plusieurs mots, par exemple en séparant ces mots par un caractère spécial) alors $\exists x\ \mathcal{M}(x, y)$ correspond à $\{y \in \mathbb{N} \mid \exists x,\ (x, y) \in L(\mathcal{M})\}$, et $\forall x\ \mathcal{M}(x, y)$ correspond à $\{y \in \mathbb{N} \mid \forall x,\ (x, y) \in L(\mathcal{M})\}$.

2.3. **Hiérarchie arithmétique.** Un résultat classique de la théorie de la calculabilité est l'indécidabilité du problème de l'arrêt des machines de Turing. Cependant, même en se donnant un oracle qui déciderait de ce problème, il existerait toujours des questions indécidables, notamment se demander si une machine avec un tel oracle s'arrête. On peut alors continuer sur ce même modèle et se donner des oracles toujours plus puissants, mais l'arrêt restera toujours indécidable au rang supérieur. Cette propriété permet de classer les problèmes indécidables selon la puissance (si elle existe) d'oracle nécessaire à rendre le problème décidable. Cette hiérarchie parmi les problèmes est présentée en FIGURE 1

2.3.1. *Les classes Δ_n^0, Σ_n^0 et Π_n^0.* Suite à ce constat, on appelle oracle de degré n un oracle qui peut décider du problème de l'arrêt des machines à oracles de degré n, où l'oracle de degré 0 et l'oracle qui ne décide rien (et ainsi, l'oracle de degré 1 décide du problème de l'arrêt). On définit alors trois types de classes de problèmes :

Définition 2.4. On pose $\Delta_0^0 = \Sigma_0^0 = \Pi_0^0$ comme la classe des problèmes récursifs. Pour $n \geq 1$ on définit par induction les ensembles suivants :

- Σ_n^0 est la classe des problèmes récursivement énumérables avec un oracle de degré $n-1$
- Π_n^0 est la classe des problèmes co-récursivement énumérables avec un oracle de degré $n-1$
- $\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$ et c'est la classe des problèmes récursifs avec un oracle de degré $n-1$

Remarque. Par la suite, pour plus de clarté, on écrira Δ_n, Σ_n et Π_n plutôt que Δ_n^0, Σ_n^0 et Π_n^0.

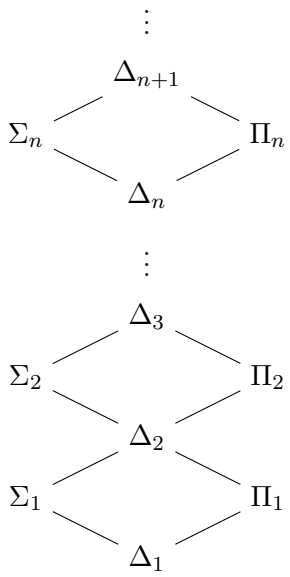

FIGURE 1. Hiérarchie arithmétique

2.3.2. *La hiérarchie arithmétique par les formules logiques.* Il existe une autre définition utile des classes de la hiérarchie arithmétique. On peut en effet les voir comme des formules utilisant des alternances de quantificateurs universels et existentiels et des fonctions récursives :

Proposition 1. *On a les égalités suivantes :*

- *Pour tout entier $n \geq 1$, $\Sigma_n = \{\exists y, \phi(x,y) \mid \phi \in \Delta_{n-1}\}$, donc les problèmes dans Σ_n sont ceux que l'ont peut écrire comme alternance de n quantificateurs $\exists\forall\exists\ldots\phi(\vec{x})$ ou ϕ est récursive.*

- *Pour tout entier* $n \geq 1$, $\Pi_n = \{\forall y, \phi(x, y) \mid \phi \in \Delta_{n-1}\}$, *donc les problèmes dans* Π_n *sont ceux que l'ont peut écrire comme alternance de* n *quantificateurs* $\forall\exists\forall \ldots \phi(\vec{x})$ *ou* ϕ *est récursive.*
- *Dans cette écriture, on définit toujours* Δ_n *comme* $\Delta_n = \Sigma_n \cap \Pi_n$, *ce qui permet de définir correctement les classes* Σ_n, Π_n *et* Δ_n *par induction.*

Cette façon de considérer la hiérarchie arithmétique sera très pratique pour nous, notamment lorsque l'on cherchera à montrer qu'un problème est dans une certaine classe.

2.3.3. *Complétude dans un classe.* Une notion importante à ajouter à ces classes est la notion de complétude, car si l'on veut comparer la difficulté de différents problèmes, dire qu'ils sont dans la même classe apporte peu d'informations. En effet, puisqu'on a de nombreuses inclusions dans la hiérarchie, pour caractériser correctement un problème, on cherchera à quelles classes il appartient, mais aussi pour quelle classe (si il en existe) il est complet, c'est à dire qu'il est plus difficile que tous les problèmes de cette classe

Définition 2.5. Soient deux problèmes A et B. On dit qu'on peut *réduire* A à B (noté $A \leq_m B$ [1]) si il existe une fonction f calculable telle que $x \in A \Leftrightarrow f(x) \in B$

Cette définition de réduction permet de définir rigoureusement le fait d'être plus difficile que tous les problèmes d'une classe :

Définition 2.6. Soient C une classe de problèmes, et A un problème

- Si : $\forall R \in C,\ R \leq_m A$, on dit que A est *C-difficile*
- Si A est C-difficile et que $A \in C$, on dit que A est *C-complet*

2.4. **Systèmes dynamiques.** Il existe différentes façons d'étudier les machines de Turing en tant que système dynamique. Tous d'abord, un tel système se définit ainsi :

Définition 2.7. Un *système dynamique* est une paire (X, T) où X est un espace topologique, et $T : X \to X$ est la fonction de transition du système

A un point $x \in X$ de l'espace du système est associée une *orbite* définie par $\mathcal{O}(x) = (T^n(x))_{n\in\mathbb{N}}$. Lorsque $\overline{\mathcal{O}(x)} = X$, x est un point *transitif* du système dynamique, qui est alors un *système transitif*. Si tous les points sont transitifs, c'est un *système minimal*

Pour étudier les machines de Turing d'un point de vue dynamique, il faut considérer un espace topologique. Pour cela, on fixe une distance :

Définition 2.8. La distance d sur $\Sigma^{\mathbb{Z}}$ est définie par $d(x, y) = 2^{-\min\{n\in\mathbb{N} | x_n \neq y_n \vee x_{-n} \neq y_{-n}\}}$ où x_i (pour $i \in \mathbb{Z}$) est la lettre d'indice i dans le mot infini x

La façon la plus naturelle d'associer un système dynamique à une machine de Turing est alors de poser $X = Q \times \Sigma^{\mathbb{Z}} \times \mathbb{Z}$ l'espace des configuration et T telle que $T(s, c, p) = (t, c', p')$ si et seulement si $(s, c, p) \vdash (t, c', p')$. Cependant, X ainsi défini n'est pas compact, donc d'autres systèmes associés à une machine de Turing on étés introduits par Kůrka [Kůr97]

2.4.1. *Modèle TMH.* Dans ce modèle, on considère que la tête de lecture de la machine se déplace sur une bande fixe [2]. Pour formaliser cela, l'espace du système est $X_h = \{(x \in (\Sigma \cup Q)^{\mathbb{Z}} \mid |\{i \in \mathbb{Z} \mid x_i \in Q\}| \leq 1\}$ (notant $|E|$ le cardinal d'un ensemble E), et la distance associée est la distance d définie précédemment sur l'alphabet $\Sigma \cup Q$. La fonction T_h s'applique alors localement aux points contenant un état, en considérant que le sous-mot précédent l'état est le mot avant la tête de lecture et que le mot après l'état est le mot à partir de la tête de lecture. Sur les mots ne contenant pas d'état, T_h agit comme l'identité

1. Le m dans $\leq_m$ signifie *many-one*, car il existe d'autres types de réductions, les réductions one-one par exemple
2. TMH signifie ainsi Turing machine with Moving Head

Remarque. D'autres systèmes, non présentés ici, sont aussi possibles, notamment le système TMT qui considère lui une tête de lecture fixe avec une bande qui se déplace [3].

3. Techniques de réduction

3.1. **La machine SMART.** Pour étudier la minimalité ou la transitivité des machines de Turing, une première étape intéressante peut être de chercher différents exemples de ce type de machines. Une des seules machines vérifiant ces propriétés connue aujourd'hui à étée introduite par Kari et Ollinger [COT17]. C'est la machine SMART [4] qui est décrite en Figure 2. Cette machine, est non seulement un exemple permettant de comprendre mieux les propriétés dynamiques des machines de Turing, mais certaines techniques de réduction utilisant la machine SMART, présentées dans [GOT15] permettent de démontrer certains théorèmes. Cette machine vérifie

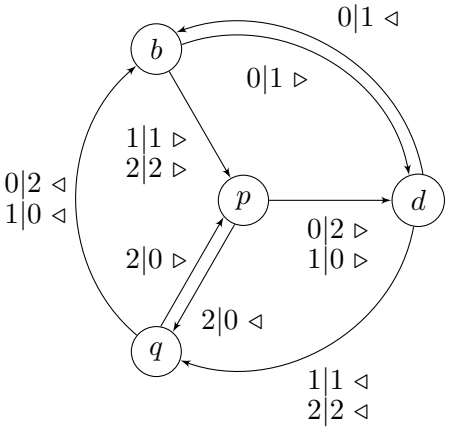

Figure 2. La machine SMART

donc la propriété suivante, qui est démontrée dans [COT17] :

Proposition 2. *La machine SMART est transitive et minimale*

3.2. **Inversion du temps.** Un première technique importante pour classer dans la hiérarchie arithmétique des problèmes sur la dynamique des machines de Turing, est la technique d'inversion du temps. Cette technique utilise la notion de machine réversible, pour créer une nouvelle machine qui ne modifie pas le ruban, mais qui a un comportement dynamique intéressant.

Pour une machine $\mathcal{M} = (Q, \Sigma, \delta)$ réversible, deux machines $\mathcal{M}_+ = (Q \times \{+\}, \Sigma, \delta^+)$ et $\mathcal{M}_- = (Q \times \{-\}, \Sigma, \delta^-)$ sont crées, où $\delta^+((s,+),a) = \delta(s,a)$ et $\delta^-((s,-),a) = \delta^-(s,a)$. Ces deux machines correspondent donc respectivement à $\mathcal{M}$ et à sa machine inverse $\mathcal{M}^{-1}$ mais avec des états distincts, ce qui permet de créer une nouvelle machine à partir de $\mathcal{M}_+$ et de $\mathcal{M}_-$, comme représenté en Figure 3.

La construction de la machine $\mathcal{M}_{IT}$, qui est l'objectif de 3.2, consiste à connecter les machines $\mathcal{M}_+$ et $\mathcal{M}_-$, en connectant les paires d'arrêt de $\mathcal{M}_+$ avec les paires de départ de $\mathcal{M}_-$, afin que la machine $\mathcal{M}_{IT}$ se comporte premièrement comme $\mathcal{M}^+$ (et donc comme $\mathcal{M}$), puis lorsque le calcul est fini, la machine $\mathcal{M}^-$ est utilisée afin "d'annuler" l'effet de $\mathcal{M}^+$ sur le ruban. Cette technique sera donc utilisée dans des réductions, avec la technique décrite dans 3.3, de tester la dynamique de $\mathcal{M}$ sans agir sur le ruban.

La machine $\mathcal{M}_{IT}$ est alors définie formellement par $\mathcal{M}_{IT} = (Q \times \{+\} \cup Q \times \{-\}, \Sigma, \delta_{IT})$ où δ_{IT} se comporte comme δ^+ sur $(Q \times \{+\} \times \Sigma)$, comme δ^- sur $(Q \times \{-\} \times \Sigma)$, à l'exception que si $((s,+),q)$ est une paire d'arrêt de $\mathcal{M}^+$, on pose $\delta_{IT}((s,+),a) = (s,-),a,-\mu(s))$

3. TMT signifie donc Turing machine with Moving Tape
4. SMART pour Small Minimal Aperiodic Reversible Turing machine

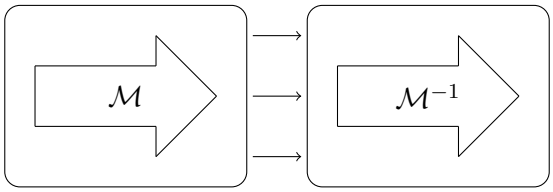

FIGURE 3. La technique d'inversion du temps

3.3. **Plongement.** Cette seconde technique est appelée technique de plongement [5]. Elle s'utilise avec la technique précédente et nécessite des machines vérifiant une certaine propriété :

Définition 3.1. Une machine $\mathcal{M}$ est dite inoffensive, si chaque paire de départ (s, a) est associée à une unique paire d'arrêt (t, a), telle que tout calcul de $\mathcal{M}$ sur une configuration $(s, c, p + \mu(s))$ avec $c(p) = a$ est soit infini, soit s'arrête sur (t, c, p), donc le calcul ne modifie pas le ruban.

Remarque. Si $\mathcal{M}$ est une machine, $\mathcal{M}_{IT}$ est inoffensive

Le plongement consiste à insérer une machine inoffensive entre chaque transition d'une autre machine. Pour cela, on considère deux machines : la machine hôte $\mathcal{H}$ et la machine invitée $\mathcal{I}$, qui est inoffensive, afin de créer une machine $\mathcal{H}_{\mathcal{I}}$. Cette machine est construite en prenant tout d'abord l'union des deux machines $\mathcal{H}$ et $\mathcal{I}$, puis en notant $(s_1, a_1), \ldots, (s_n, a_n)$ les paires de départ de $\mathcal{I}$, et $(t_1, a_1), \ldots, (t_n, a_n)$ les paires d'arrêt associés ($\mathcal{I}$ étant inoffensive), chaque transition $\delta(p, a) = (q, b, d)$ est supprimée de $\mathcal{H}_{\mathcal{I}}$, et les transitions $\delta(p, a) = (s_1, a_1, \mu(s_1)), \delta(t_1, a_1) = (s_2, a_2, \mu(s_2)), \delta(t_2, a_2) = (s_3, a_3, \mu(s_3)), \ldots \delta(t_n, a_n) = (q, b, d)$ sont ajoutées, comme illustré en FIGURE 4.

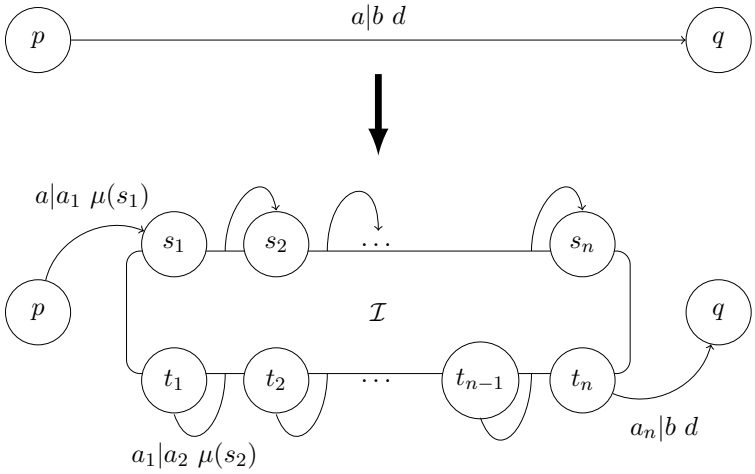

FIGURE 4. La technique de plongement

4. Classification de problèmes dans la hiérarchie arithmétique

4.1. **Problèmes sur les langages des machines de Turing.** Un premier résultat important pour classer certains problèmes dans la hiérarchie et qui est prouvé dans [Odi99] est la proposition suivante :

Proposition 3. *Le problème* TOT *: "Une machine de Turing $\mathcal{M}$ est-elle totale ?" est Π_2-complet*

5. Technique d'embedding en anglais

Il peut être utile de chercher à n'utiliser que des machines réversibles, et donc de s'intéresser à une version restreinte du problème TOT : on considère le problème $\mathsf{TOT_R}$: *"Une machine $\mathcal{M}$ réversible est-elle totale ?"*

Proposition 4. *Le problème* $\mathsf{TOT_R}$ *est* Π_2*-complet*

Démonstration. Tout d'abord, il est trivial que ce problème est Π_2 car c'est une restriction de TOT qui est lui même Π_2. La difficulté est de montrer que $\mathsf{TOT_R}$ est complet et pour cela, on va réduire le problème TOT.

Soit une machine de Turing $\mathcal{M}$, qu'on va chercher à transformer en une machine injective, pour la rendre réversible. Pour cela, tant qu'il existe des transitions non injectives dans $\mathcal{M}$, on se donne une transition non injective, on note k le nombre d'antécédents, et on créé k copies de $\mathcal{M}$, où on ne garde qu'un antécédent à cette transition de chaque copie (et de sorte à ce que chaque copie corresponde à un antécédents distinct). La machine $\mathcal{M}$ devient alors toutes ces machines disjointes.

Ce procédé termine car chaque transition a au plus $|Q| \times |\Sigma|$ antécédents (en notant $|E|$ le cardinal d'un ensemble E), mais lorsque l'on a transformé une transition, elle est définitivement injective dans toutes les machines copies. Ainsi, après au plus $(|Q| \times |\Sigma|)^{|Q|\times|\Sigma|}$ étapes ce procédé termine (car il y a au plus $|Q| \times |\Sigma|$ transitions), et la machine obtenue est notée $\mathcal{M}'$.

Pour que la réduction décrite ici fonctionne, il faut que $\mathcal{M}'$ (qui est injective donc réversible) soit totale si et seulement si $\mathcal{M}$ l'est.

- $\Rightarrow$ Supposons $\mathcal{M}'$ totale. Pour une configuration partielle (s, c, p) de $\mathcal{M}$, l'état s est copié plusieurs fois dans $\mathcal{M}'$ mais pour chacun de ces états s', le calcul de $\mathcal{M}'$ sur (s', c, p) s'arrête car $\mathcal{M}'$ est totale. Or l'un de ces calculs correspond à celui effectué par $\mathcal{M}$ sur la configuration (s, c, p) donc ce calcul s'arrête. Ainsi, $\mathcal{M}$ est totale.
- $\Leftarrow$ Supposons alors $\mathcal{M}$ totale. Si (s', c, p) est une configuration de $\mathcal{M}'$, le calcul de $\mathcal{M}$ sur la configuration (s, c, p) (où s' à été copié depuis s) est le même que celui de $\mathcal{M}'$ sur (s', c, p). $\mathcal{M}'$ est alors totale.

∎

A présent, il peut être utile de classer d'autres problèmes classiques sur les machines de Turing dans la hiérarchie arithmétique :

- EQU "Pour deux machines $\mathcal{M}_1$ et $\mathcal{M}_2$, a-t-on $L(\mathcal{M}_1) = L(\mathcal{M}_2)$ ainsi que pour tout $u \in L(\mathcal{M}_1)$, $\mathcal{M}_1(u) = \mathcal{M}_2(u)$?"
- INCL "Pour deux machines $\mathcal{M}_1$ et $\mathcal{M}_2$, a-t-on $L(\mathcal{M}_1) \subseteq L(\mathcal{M}_2)$?"
- INF "Pour une machine $\mathcal{M}$, le cardinal de $L(\mathcal{M})$ est-t-il infini ?"
- EQU$^-$ "Pour deux machines $\mathcal{M}_1$ et $\mathcal{M}_2$, a-t-on $\forall u \in L(\mathcal{M}_1) \cap L(\mathcal{M}_2),\ \mathcal{M}_1(u) = \mathcal{M}_2(u)$?"

Proposition 5. *Les problèmes* EQU, INCL *et* INF *sont* Π_2 *et le problème* EQU$^-$ *est* Π_1*-complet.*

Démonstration. Pour prouver ces différents résultats, le point intéressant n'est généralement pas de montrer que ces problèmes appartiennent à une classe mais plutôt qu'ils sont difficiles pour cette classe. Pour cela, on va les réduire :

- EQU Le problème R qui, pour deux machines $\mathcal{M}_1$, $\mathcal{M}_2$, un mot u et un temps t, décide si $\mathcal{M}_1$ s'arrête sur u en moins de t étapes si et seulement si $\mathcal{M}_2$ s'arrête sur u en moins de t étapes, et si en ce cas, le résultat de ces deux calculs est le même, est décidable, car il suffit de simuler les deux machines pendant t étapes. Ainsi,
  $(\mathcal{M}_1, \mathcal{M}_2) \in$ EQU $\Leftrightarrow \forall u \exists t\ R(\mathcal{M}_1, \mathcal{M}_2, u, t)$ donc EQU s'écrit $\forall\exists R$ avec R décidable : EQU est Π_2. On réduit EQU à TOT pour prouver sa complétude :
- INCL Considérant le problème R qui décide si, pour deux machines $\mathcal{M}_1$ et $\mathcal{M}_2$, un mot u et un temps t, si la machine $\mathcal{M}_1$ s'arrête s'arrête en moins de t étapes sur le mot sur le mot u alors $\mathcal{M}_2$ s'arrête en moins de t étapes sur u, ce problème est décidable car il suffit de

simuler $\mathcal{M}_1$ et $\mathcal{M}_2$ sur u pendant t étapes, ainsi :
$(\mathcal{M}_1, \mathcal{M}_2) \in \mathsf{INCL} \Leftrightarrow \forall u \exists t\ R(\mathcal{M}_1, \mathcal{M}_2, u, t)$ donc INCL s'écrit $\forall\exists R$ avec R décidable : INCL est Π_2

INF Pour prouver que INF est Π_2, on utilise le problème R qui décide si, pour une machine $\mathcal{M}$, un mot u, un temps t et une taille n si u est de taille supérieure à n et si $\mathcal{M}$ s'arrête sur u en moins de t étapes. Ce problème est décidable, et
$\mathcal{M} \in \mathsf{INF} \Leftrightarrow \forall n \exists u \exists t\ R(\mathcal{M}, u, t, n)$ donc INF est Π_2.

$\mathsf{EQU^-}$
- Ici, on considère le problème (décidable) R qui décide, pour deux machines $\mathcal{M}_1$ et $\mathcal{M}_2$, un mot u et un temps t si, dans le cas où $\mathcal{M}_1$ et $\mathcal{M}_2$ s'arrêtent en moins de t étapes dans le calcul sur u, le résultat du calcul est le même pour $\mathcal{M}_1$ et $\mathcal{M}_2$. On peut ainsi écrire :
$(\mathcal{M}_1, \mathcal{M}_2) \in \mathsf{EQU^-} \Leftrightarrow \forall (u, v)\ R(\mathcal{M}_1, \mathcal{M}_2, u, t)$ donc $\mathsf{EQU^-}$ s'écrit $\forall R$ avec R décidable : $\mathsf{EQU^-}$ est Π_1.
- Le problème ATT : "Pour une machine $\mathcal{M}$ et deux configurations u et v de cette machine, a-t-on $u \vdash^* v$?" est Σ_1-complet [GOT15]. On va donc réduire le complémentaire de $\mathsf{EQU^-}$ à ATT. Soit $\mathcal{M}$ une machine, u et v deux configurations et \$ un symbole qui n'est pas dans l'alphabet de $\mathcal{M}$. Alors on pose $\mathcal{M}_1$ comme la machine qui efface tout entrée, écrit \$ puis s'arrête, et on construit une machine $\mathcal{M}_2$ qui passe dans la configuration u, puis agit comme $\mathcal{M}$ mais qui, après chaque transition, vérifie si la machine est dans la configuration v (cette construction est proche de celle du plongement décrite plus tôt), et en ce cas efface le contenu de la bande, écrit \$ puis s'arrête. La seule difficulté pour faire cela est qu'il faut se souvenir de la position de la tête de lecture lors des phases de vérification, mais il suffit de considérer un alphabet Σ' disjoint de $\Sigma \cup \{\$\}$, et de remplacer la lettre sur laquelle pointe la tête de lecture par son analogue dans Σ'.
Ainsi, $\mathcal{M}$ atteint v à partir de u si et seulement si il existe un mot w tel que $\mathcal{M}_2$ s'arrête sur w et $\mathcal{M}_2(\mathrm{w}) = \$$. On a ainsi bien réduit notre problème et $\mathsf{EQU^-}$ est Π_1-complet.

■

Remarque. Le problème $\mathsf{ATT_R}$ qui est le même problème que ATT mais avec une machine réversible est lui aussi Σ_1-complet, donc la même preuve que la précédente donne que $\mathsf{EQU^-_R}$ qui considère des machines réversibles est lui aussi Π_1-complet.

4.2. **Machines à compteurs.** Les machines à compteurs, introduites par Minsky, sont un modèle de calcul alternatif aux machines de Turing. Elles sont Turing complètes [Hoo66] (c'est-à-dire que leur puissance de calcul est la même que celle des machines de Turing), et nous intéressent ici car on espère pouvoir simuler une machine à compteurs entre les transitions de SMART, sur la même idée que le plongement mais avec des machines à compteurs. Pour définir ces objets, on pose tout d'abord les ensembles de *valeurs* $\Upsilon = \{Z, P\}$, qui feront la distinction entre un compteur non nul et un, compteur (strictement) positif, et d'*opérations* $\Phi = \{-, 0, +\}$ qui nous serons utiles dans la définition

Définition 4.1. Une *k-machine à compteurs* M est un triplet (S, k, T) où S est l'ensemble des *états*, k est le nombre de compteurs et $T \subseteq S \times \{1, \ldots, k\} \times (\Upsilon \cup \Phi) \times S$ est l'ensemble des *transitons*.

On distingue deux types de transitions : si $(s, i, \phi, t) \in T$ alors c'est une *transition de test* si $\phi \in \Upsilon$, et c'est une *transition de modification* si $\phi \in \Phi$. On interdit ici que les compteurs soient strictement négatifs, dont les transitions du type $(s, u, i, -, t)$ avec $u(i) = 0$ sont illégales. De façon analogue aux machines de Turing, une *configuration* d'une machine est une paire $(s, v) \in Q \times \mathbb{N}^k$ où s est l'état de la machine et v est la valeur de ses compteurs.

Hooper à donnée, dans [Hoo66], la construction explicite à partir d'une machine de Turing réversible $\mathcal{M}$, d'une machine à compteur M ayant 2 compteurs et le même comportement que $\mathcal{M}$. L'ensemble des machines à compteurs simulant une machine de Turing réversible, suivant la construction de Hooper sera noté $\mathcal{MC}_{\mathcal{R}}$

Comme on l'a vu précédemment, le problème de la totalité sur les machines de Turing réversibles est Π_2-complet, posant alors $\mathsf{TOT}_{\mathsf{R}}^{\mathsf{MC}}$: "Une machine à compteurs $M \in \mathcal{MC}_{\mathcal{R}}$ est-elle totale ?", le résultat suivant peut être conjecturé :

Conjecture 2. *Le problème* $\mathsf{TOT}_{\mathsf{R}}^{\mathsf{CM}}$ *est* Π_2*-complet*

5. Transitivité et minimalité

Le problème central ici est le problème TR : "Pour une machine $\mathcal{M}$, le système dynamique (TMH) associé est-il transitif ?", mais les ingrédients données dans les parties précédentes ne permettent pas de prouver la conjecture 1. Cependant, certains résultats intéressants peuvent être démontrés. Tou d'abord, la proposition suivante donne une caractérisation pratique de la transitivité d'une machine $\mathcal{M}$:

Proposition 6. *Pour une machine* $\mathcal{M}$*, le système dynamique (TMH) associé* (X_h, T_h) *est transitif si et seulement si pour toute paires* $(u.u')$ *et* $(v.v')$ *de configurations du système, il existe* w *tel que* $u.u'$ *est un sous mot de* w *(soit* $u.u' = w_i \dots w_j$ *avec* $0 \leq i \leq j$*) et* $n \in \mathbb{N}$ *tels que* $v.v'$ *est un sous mot de* $T_h^n(w)$

Remarque. Un autre problème auquel on peut aussi s'intéresser est le problème MIN : "Pour une machine $\mathcal{M}$, le système dynamique (TMT) associé est-il minimal ?". Une proposition analogue à celle-ci peut être donnée pour caractériser les machines dont le système (TMT) est minimal.

Pour montrer ce résultat, on utilise le lemme suivant :

Lemme. *Un système dynamique* (X, T) *est transitif si et seulement si pour toute paire d'ouverts* U *et* V*, il existe* t *tel que* $T^t(U) \cap V \neq \emptyset$

Démonstration de la proposition 6. Pour une machine $\mathcal{M}$ et une configuration u de X_h, l'ensemble des configurations w telles que u est un sous mot de w forme un ouvert de X_h, donc le lemme précédent permet d'affirmer que si $\mathcal{M}$ est transitive, pour toute paire de configurations $(u.u'), (v.v')$, il existe w et t tel que $(v.v')$ est un sous mot de $T_h^t(w)$. Réciproquement, si $\mathcal{M}$ n'est pas transitive, il existe (par le lemme) deux ouverts U et V tels que pour tout t, $T_h^t(U) \cap V = \emptyset$, donc pour $u \in U$ et $v \in V$, pour tout mot w tel que u est un sous mot de w, v n'est pas un sous mot de $T_h^n(w)$. ∎

A l'aide de cette proposition, un premier résultat sur la position dans la hiérarchie arithmétique du problème TR peut être donné :

Proposition 7. TR *est* Π_2

Démonstration. On considère le problème R décidant, pour une machine $\mathcal{M}$, deux configurations TMH de cette machine $(u.u')$ et $(v.v')$, un mot w et un temps n si $u.u'$ est un sous mot de w et si $v.v'$ est un sous mot de $T_h^n(w)$. Ce problème étant décidable, la proposition 6 nous donne : $\mathcal{M} \in \mathsf{TR} \Leftrightarrow \forall((u.u'), (v.v'))\exists(w, n)\ R((u.u'), (v.v'), w, n)$ donc TR est Π_2. ∎

Un second résultat, qui permet ici de s'approcher de la conjecture 1 peut être prouvé grâce aux outils développés précédemment :

Théorème 1. TR *est* Π_1*-difficile.*

Remarque. De façon analogue, il est possible de prouver que MIN est Σ_1-difficile.

Ce théorème est démontré dans [GOT15], ici la preuve présentée est la réduction d'un autre problème basée sur une construction plus simple.

Démonstration. On va réduire le problème TR au problème $\mathsf{EQU}_{\mathsf{R}}^{-}$: soit deux machines $\mathcal{M}_1$ et $\mathcal{M}_2$. Un résultat classique sur les machines de Turing permet de considérer que l'alphabet de $\mathcal{M}_1$ et de $\mathcal{M}_2$ a deux caractères, 0 et 1, et que les propriétés dynamiques sont conservées (notamment la réversibilité). Posant $\mathcal{M}$ la machine qui, comme dans la technique d'inversion du temps, connecte la machine $\mathcal{M}_1$ avec la machine $\mathcal{M}_2$. Cette connexion peut être faite car les seules paires de départ qui nous intéressent sont celles avec les états de départ des machines (car on considère les mots sur lesquels les machines vont s'arrêter en partant de l'état initial). La connexion se fait alors en connectant la paire d'arrêt de $\mathcal{M}_1$ liée à $(q_0^1, 0)$ (où q_0^1 est l'état initial de $\mathcal{M}_1$, et q_0^2 est celui de $\mathcal{M}_2$) avec $(q_0^2, 0)$, ainsi que de même pour le caractère 1. Enfin, si lors d'un calcul la machine lit un 2, elle refait le calcul en sens arrière (ce qui est possible grâce à la réversibilité).
Suite à cette construction, on va montrer que la machine $\mathrm{SMART}_{\mathcal{M}}$ (qui est le plongement de $\mathcal{M}$ dans SMART) est transitive si et seulement si $(\mathcal{M}_1, \mathcal{M}_2) \in \mathsf{EQU}_{\mathsf{R}}^{-}$.
Si $\mathrm{SMART}_{\mathcal{M}}$ est transitive, et que $u \in L(\mathcal{M}_1) \cap L(\mathcal{M}_2)$, il existe $n \in \mathbb{N}$ tel que le calcul pour $\mathcal{M}_1$ et pour $\mathcal{M}_2$ sur u n'utilise qu'une portion de la bande qui est de la forme $0^n\ u\ 0^n$, or la machine $\mathrm{SMART}_{\mathcal{M}}$ étant transitive, le calcul de cette machine sur la configuration $(2\ 0^n, q_0^1, u\ 0^n\ 2)$ atteint toute configuration finie, donc il atteint la configuration qui "suit" la précédente dans le calcul dans SMART. Or si $\mathcal{M}_1(u) \neq \mathcal{M}_2(u)$, lorsque le calcul sera dans $\mathcal{M}_2^{-1}$, la bande ne reviendra pas sous la forme $(2\ 0^n, q, u\ 0^n\ 2)$ (où q est l'état juste avant de repasser dans SMART), et ainsi, la configuration qui "suit" ne sera pas vu, et ce à chaque fois car si cette configuration était atteinte, la configuration $(2\ 0^n, q_0^1, u\ 0^n\ 2)$ serait la précédente : ainsi, la transitivité donne $\mathcal{M}_1(u) = \mathcal{M}_2(u)$.
A présent, supposant que $\forall u \in \mathsf{L}(\mathcal{M}_1) \cap L(\mathcal{M}_2),\ \mathcal{M}_1(u) = \mathcal{M}_2(u)$, pour deux configurations u et v, chaque calcul de $\mathcal{M}$ dans $\mathrm{SMART}_{\mathcal{M}}$ est fini, car si l'une des deux machines avait un calcul infini, puisqu'elle est réversible, elle utiliserait une portion infinie de la bande, donc le calcul dans $\mathrm{SMART}_{\mathcal{M}}$ s'arrêtera car la tête de lecture arrivera sur un symbole 2. Ainsi, si v est une configuration avec un état de SMART, puisque SMART est transitive, cette configuration sera forcément atteinte. Sinon, elle a un état de $\mathcal{M}$, et $\mathcal{M}$ étant réversible, il existe une configuration de SMART u' telle que, dans $\mathrm{SMART}_{\mathcal{M}}$, $u' \vdash^* v$ (u' s'obtient par le calcul de $\mathcal{M}^{-1}$ sur v). Et dans $\mathrm{SMART}_{\mathcal{M}}$, on a $u \vdash^* u' \vdash^* v$ (toujours grâce à la transitivité de SMART). ∎

6. Conclusion

Finalement, on a cherché à classer le problème TR dans la hiérarchie arithmétique : pour cela, les techniques de d'inversion du temps et de plongement dans la machine SMART permettent de prouver que ce problème est Π_1-difficile (et permettent de même de montrer que MIN est Σ_1-difficile). Pour cela, il a aussi fallu classer un autre problème dans la hiérarchie : le problème $\mathsf{EQU}_{\mathsf{R}}^{-}$.

Afin de prouver la conjecture 1, plusieurs pistes sont possibles : tout d'abord, une réduction de problème Π_2-complets sur les langages des machines (comme les problèmes EQU, INCL ou INF, mais qui n'ont pas étés prouvés comme étant complets) est possible, sur un modèle proche de la réduction avec le problème $\mathsf{EQU}_{\mathsf{R}}^{-}$. Une autre façon d'obtenir ce résultat serait de prouver que le problème $\mathsf{TOT}_{\mathsf{R}}^{\mathsf{CM}}$ est Π_2-complet, ce qui n'a pas pu être creusé ici, faute de temps, puis de chercher comment simuler une machine à deux compteurs à l'aide de la machine SMART, en utilisant le fait que cette machine créée des plages de "0" aussi grande que l'on veut [COT17].

Références

[COT17] Julien Cassaigne, Nicolas Ollinger et Rodrigo Torres-Avilés. "A Small Minimal Aperiodic Reversible Turing Machine". In : *Journal of Computer and System Sciences (JCSS)* (2017).

[GOT15] Anahí Gajardo, Nicolas Ollinger et Rodrigo Torres-Avilés. "The transitivity problem of Turing machines". In : *Mathematical Foundations of Computer Science 2015 - 40th International Symposium (MFCS 2015)*. T. 9234. Lecture Notes in Computer Science. Milan, Italy : Springer, août 2015, pp. 231-242.

[Hoo66] Philip K. Hooper. "The Undecidability of the Turing Machine Immortality Problem". In : *The Journal of Symbolic Logic* (1966).

[Kůr97] Petr Kůrka. "On topological dynamics of Turing machines". In : *Theoretical Computer Science* 174.1 (1997), p. 203–216.

[Moo91] C Moore. "Generalized shifts : unpredictability and undecidability in dynamical systems". In : *Nonlinearity* 4.2 (mai 1991), p. 199–230.

[Odi99] Piergiorgio Odifreddi. "Classical Recursion Theory Vol II". In : *Studies in Logic and the Foundations of Mathematics* (1999).