Notes de cours Modèles de Calcul
Machines de Turing
L3 - Informatique

F. Prost

2021

1 Remarques préliminaires

Le problème de la caculabilité est assez récent et remonte, de manière formelle, aux questions qu'avait posé Hilbert lors du fameux congrès de Paris en 1900. En fait le problème existait avant mais n'était pas abordé : la méthode scientifique qui consiste à observer, poser une théorie et comparer les résultats théoriques avec les résultats pratiques, présuppose que les calculs dans le modèle théorique sont faisables. Or, comme Turing l'a montré, certains calculs ne sont pas réalisables par machine.

De plus on peut remarquer que toute production scientifique peut être vue comme le résultat d'un seul logiciel : un éditeur de texte. Tous les livres, vidéos, sons etc. enregistrés peuvent être digitalisés, c'est-à-dire, in fine, transformés en une suite plus ou moins longue de 0 et de 1 qui correspondent aux bits enregistrés pour ces documents. Ainsi toute la science se retrouve plongée à l'intérieur d'une partie d'elle même, car l'informatique est une science. Cette remarque n'est pas sans rapport avec la démonstration des théorèmes d'incomplétude de Gödel qui montrent que toute formule, et toute démonstration, peuvent se voir comme de simples entiers modulo un certain codage.

Une implication directe de cette remarque est que les limites de la calculabilité sont des limites qui s'imposent à tous les discours structurés, à toutes les sciences en particulier.

- Qu'est-ce que le calcul effectif : il faut pouvoir manier les symboles de manière mécanique. Il y a donc une **interprétation** de ces symboles que ce soit pour les entrées ou pour les sorties.
- Une implication est que l'on ne travaille que sur des entiers (modulo codage de N^2 dans N. Les entrées et sorties de la fonction implantée sont dénombrables.
- Jeu sur le fini/l'infini et les tailles non bornées. Le calcul doit se faire en un temps fini, la taille du programme est finie (rien ne doit être implicite ou magique) mais non bornée (on ne veut pas que des limites technologiques influent les définitions mathématiques). D'autre part l'espace de travail doit être fini (mais non borné): Nécessairement discret car un réel comporte une finité d'information (l'ensemble des chiffres décimaux suivant la virgule est infini).

Finalement ce qui nous intéresse est un sous-ensemble des fonctions sur les entiers.
On notera $\mathcal{F}^{(p)}$ l'ensemble des fonctions de $\mathbb{N}^{(p)}$ dans $\mathbb{N}$ et $\mathcal{F} = \cup_{p\in\mathbb{N}}\mathcal{F}^{(p)}$.
Des fonctions sur les entiers calculables on connait : addition, calcul du nième nombre premier, pgcd (on a des algos depuis la grèce antique) etc.
Des fonctions un peu moins évidentes :

$$f_1(x) = \begin{cases} 1 \text{ si le développement décimal de } \pi \text{ contient une sous-suite} \\ \quad \text{formée de } x \text{ '5', sans '5' à gauche ni à droite} \\ \\ 0 \text{ sinon} \end{cases}$$

$$f_2(x) = \begin{cases} 1 \text{ si le développement décimal de } \pi \text{ contient une sous-suite} \\ \quad \text{formée d'au moins } x \text{ '5'} \\ \\ 0 \text{ sinon} \end{cases}$$

Pour l'instant aucune méthode ou preuve mathématique ne permet de faire le calcul de f_1 dans un cadre général : on ne sait pas si f_1 est calculable.

Pour f_2, elle est certainement calculable,..., mais on ne sait pas la calculer. En effet deux cas sont possibles :

1. Il n'y a pas de borne à la longueur d'une sous-suite formée de '5' dans le développement décimal de π auquel cas c'est la fonction constante qui répond toujours 1
2. Il y a une borne k et l'implantation est facile si $x \leq k$ on répond 1 sinon on répond 0.

Dans les deux cas f_2 est calculable.

Partialité

- Quand on est devant une machine comment s'exprime la partialité ? Par exemple la division par 0 ?
- Il faut nécessairement que toute définition de fonctions intuitivement calculables comprenne des fonctions partielles. En effet supposons qu'on ait une définition de la calculabilité intuitive qui ne contienne que des fonctions totales.

  Soit $\mathcal{C} \subset \mathcal{F}^{(1)}$ une sous-classe de fonctions totales calculables (quelle que soit la définition de calculable qu'on ait). Comme il s'agit d'une classe de fonctions calculables il faut un programme écrit dans un langage quelconque. On a donc :
  $$\mathcal{C} = \{f_0, f_1, \ldots, f_n, \ldots\}$$

  On peut définir $g \in \mathcal{F}^{(1)}$ par $g(n) = f_n(n) + 1$. g est évidemment calculable et totale mais g n'est pas dans $\mathcal{C}$ car pour tout n dans $\mathbb{N}$ on a $g(n) \neq f_n(n)$. C'est la technique de diagonalisation.

En conclusion il faut accepeter la partialité pour avoir une définition cohérente de calculabilité effective. En effet dans le cas de fonctions partielles il est possible qu'il n'y ait pas de réponse pour $f_i(i)$ ainsi $f_i(i)+1$ est également indéfini et g et f_i peuvent très bien êtres "égaux".

2 Machines de Turing

L'avantage du modèle des machines de Turing comme modèle de calcul (car il y a des centaines d'autres modèles de calcul) :

- Simplicité de l'agent de calcul.
- Aspect mécanique très évident (pas le cas pour les fonctions récursives partielles ou le λ-calcul pour lequel le mécanisme de subsitution des variables libres est compliqué).

Il existe de nombreux formalismes, nous en verrons quelques un ultérieurement dans ce cours. Nous choisissons le modèle à quadruplet de Rodgers pour sa simplicité et par le fait qu'il permette d'aller jusque dans le détail de la numération de Gödel.

Définition 1 (Machine de Turing)**.** *On pose $Q = \{q_0, q_1, \ldots\}$ et $S = \{S_0, S_1, \ldots\}$ deux ensembles canoniques infinis dénombrables d'états et de symboles, d'intersection vide, et deux symboles particuliers L, R qui ne sont ni dans Q ni dans S.*

Une machine de turing Z est la donnée d'un ensemble non vide PZ et consistant de quadruplets de l'une des trois formes suivantes :

1. q_i, S_j, S_k, q_l
2. q_i, S_j, L, q_l
3. q_i, S_j, R, q_l

Un tel ensemble est consistant s'il n'y a pas deux quadruplets différents qui commencent par les deux premiers éléments q_i, S_j (pas d'ambigüité).
On notera $E_Z \subset Q$ et $S_Z \subset S$ les ensembles minimaux finis d'états et de symboles utilisés dans P_Z.

On distinguera deux symboles particuliers S_0, le "blanc", qu'on noter aussi B, et S_1 la "barre" ou le "un", noté également par "|".

La machine dispose d'une bande infinie sur laquelle elle inscrit et lit les symboles. En fait la bande est finie à chaque instant, pour cela on suppose qu'au départ la bande est presque partout blanche : donc toutes les cellules sauf un nombre fini d'entre elles portent le symbole B.

Définition 2 (Description de l'état de la machine de Turing)**.** *On définit les termes suivant :*

- *Une description instantanée : est un mot de $S^*.Q.S^+$*
- *Une description instantanée d'une machine Z : est un mot de $S_Z^*.E.S^+$*
- *Une expression de bande : est un mot de S^**
- *Une expression de bande d'une machine Z : est un mot S^**
- *$\langle\alpha\rangle$: le nombre de | dans le mot α, si c'est une description instantanée*

Une description instantanée d'une machine Z décrit sa configuration de façon unique si on ne considère pas les blancs "inutiles" (sinon on peut toujours rajouter des blancs à gauche ou à droite d'une description instantanée). Soit $\alpha = \gamma q_i S_j \delta$ une telle description minimale, elle correspond à la bande où toutes les cases sont blanches, sauf peut-être sur l'expression de bande déduite de α, c'est à dire $\gamma S_j \delta$, et où la machine est dans l'état q_i et sa tête de lecture pointe sur le symbole S_j indiqué.

Une machine va calculer pas à pas : chaque pas provoque le passage d'une configuration à une autre.

Définition 3. *Soit Z une machine de Turing, α, β deux descriptions instantanées de Z avec $\alpha = \gamma q_i S_j \delta$ et $\beta = \eta q S \theta$.*

1. *On dit que α dérive vers β par Z, et on note $\alpha \vdash_Z \beta$, si et seulement si on a un des cas suivants :*
   (a) *Soit $q_i, S_j, S_k, q_L \in P_Z$, et $\eta = \gamma$, $q = q_L$, $S = S_k$ et $\theta = \delta$.*
   (b) *Soit $q_i, S_j, R, q_L \in P_Z$, et $\eta = \gamma S_j$, $q = q_L$, et*
   $$\begin{cases} Si\ \delta = \epsilon & alors \quad \theta = \epsilon, S = B \\ Si\ \delta = S_k \delta' & alors \quad \theta = \delta', S = S_k \end{cases}$$
   (c) *Soit $\gamma, q_i, S_j, L, q_L \in P_Z$, et $\theta = \gamma S_j \delta$, $q = q_L$, et*
   $$\begin{cases} Si\ \gamma = \epsilon & alors \quad \eta = \epsilon, S = B \\ Si\ \gamma = \gamma' S_k & alors \quad \eta = \gamma', S = S_k \end{cases}$$
2. *α est terminale ou finale pour Z si pour aucun β on a $\alpha \vdash_Z \beta$. Autrement dit α est finale si aucun quadruplet de P_Z ne contient un quadruplet commençant par $q_i S_j$.*
3. *Un calcul de Z est une suite finie de descriptions instantanées $\alpha_1, \alpha_2, \ldots, \alpha_p$ telle que pour $1 \leq i < p$ on ait $\alpha_i \vdash_Z \alpha_{i+1}$ et que α_p est terminale pour Z. α_p est la résultante de α_1 par Z, et on note*
   $$\alpha_p = Res_Z(\alpha_1)$$

Nous allons maintenant définir la notion de fonction calculable relativement aux machines de Turing. Il faut pour cela fixer un codage des entrées sur la bande ainsi qu'une manière d'interpréter le résultat du calcul.

Pour tout entier x on associe l'expression de bande $\overline{x} = |^{x+1}$. Il faut mettre $x+1$ barres car il faut pouvoir coder le 0. On étend cette convention aux cortèges d'entiers $\vec{x} = (x_1, x_2, \ldots, x_n)$ auxquels on associe l'expression de bande $\overline{(x_1, x_2, \ldots, x_n)} = \overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$. Le cas particulier d'une fonction sans argument sera traité par une bande intégralement recouverte de symboles blancs.

Définition 4. *Soit Z une machine de Turing. Pour tout entier n, on associe à Z la fonction partielle Ψ_Z^n de $\mathbb{N}^n$ dans $\mathbb{N}$ en posant :*

$$\Psi_Z^{(n)}(x_1, x_2, \ldots, x_n) = \begin{cases} \langle Res_Z(q_0\overline{(x_1, \ldots, x_n)})\rangle & \text{si } Res_Z(q_0\overline{(x_1, \ldots, x_n)}) \text{ est défini} \\ \uparrow & \text{sinon} \end{cases}$$

On peut donc maintenant définir précisément ce que sont les fonctions calculables au sens de Turing.

Définition 5. *Une fonction f de $\mathbb{N}^n$ dans $\mathbb{N}$ est dit (partielle) T-calculable s'il existe une machine de Turing Z telle que $f = \Psi_Z^n$.*

On appelle T l'ensemble de toutes les fonctions T-calculables. On note $T^(n)$ les fonctions T-calculables d'arité n.

2.1 Ensemble, prédicats, Langages

Nous avons défini la notion de fonction calculable pour les fonctions des cortèges d'entiers dans les entiers. En fait cette notion est équivalente à d'autres présentation de la calculabilité. Au lieu de parler de fonctions on peut parler d'Ensembles calculables, de prédicats calculables ou de langages calculables (on emploie plutôt le terme de reconnaissable dans ce cas). Ce sont en fait 4 manières équivalentes de parler de la même chose.

Pour cela il faut considérer une technique centrale : on peut coder tout cortège d'entiers sur un entier. Le cas de base se fait en considérant un bijection de $\mathbb{N}^2$ dans $\mathbb{N}$. Il en existe de nombreuses (en fait une infinité). Donnons par exemple le codage de Cantor qui numérote les points de $\mathbb{N}^2$ en suivant les diagonnales.

La fonction $\mathcal{C} : \mathbb{N}^2 \to \mathbb{N}$ est définie par $\mathcal{C}(x, y) = (1+2+\ldots+(x+y)+x) = 1/2((x+y)^2+3x+y)$. On peut vérifier qu'il s'agit bien d'une bijection, on notera π_1, π_2 les fonctions de $\mathbb{N} \to \mathbb{N}$ telles que $\pi_1(\mathcal{C}(x, y)) = x$ et $\pi_2(\mathcal{C}(x, y)) = y$.

- Ensembles: un sous-ensemble d'entiers A sera dit calculable si sa fonction caractéristique $\Xi_A : \mathbb{N} \to \mathbb{N}$, définie par $\Xi_A(x) = 1$ si $x \in A$ et $\Xi_A(x) = 0$ si $x \notin A$, est T-calculable. Symétriquement on pourrait définir la calculabilité d'un ensemble comme un machine de Turing qui sur l'entrée x termine dans un état q_y si son entrée est dans A et q_N sinon. Si on considère des machines ne calculant que des ensembles on peut en fait définir une notion de fonction calculable de la façon suivante f est ensemble-calculable si l'ensemble $A_f = \{\mathcal{C}(x, f(x)) \mid x \in \mathbb{N}\}$ est calculable.

- Prédicats: un prédicat sur les entiers est une fonction qui pour chaque entier rend un booléen, donc soit vrai soit faux. Un prédicat p sera codé par une MT Z si $\Psi_Z^{(1)}$ est telle que $\Psi_Z^{(1)}(x) = 1$ si et seulement $p(x)$ est vrai, sinon $\Psi_Z^{(1)}(x) = 0$. Comme la fonction caractéristique d'un ensemble est un prédicat particulier, il est immédiat de constater que la définition de prédicat calculable est équivalente à celle d'un ensemble calculable (qui lui même est équivalent à la notion de fonction calculable).

- Langage: une autre manière de présenter les machines de Turing est de les présenter comme des machines qui, étant donné un mot m sur un alphabet Σ, répond si $m \in L$ ou $m \notin L$ avec L un langage (c'est à dire que $L \subseteq \Sigma^*$. On peut voir un langage L comme un ensemble d'entiers en passant par un codage : chaque lettre de Σ peut être codée par un nombre premier (on notera $pr(l)$ le l-ième nombre premier), et on code le mot $m = l_1 l_2\ ldotsl_n$ par $code(m) = \Pi_{i=1}^n pr(i)^{pr(l_i)}$, le théorème de l'unicité de la décomposition en facteur premier permet de décoder de manière non ambigüe tout entier en le mot correspondant (on étudiera cela plus précisément avec la numération de Gödel). Alors l'appartenance de m au langage L devient équivalente à l'appartenance de $code(m)$ à l'ensemble $\{code(m) \mid m \in L\}$.

3 Universalités

Le modèle des MT est simple et clairement mécanique. Cependant il se trouve qu'il est également universel à différents points de vues.

Trois types d'universalité :

- Universalité externe : changer le modèle de manière "raisonnable" ne sert à rien.
- Universalité interne : une seule machine peut simuler toutes les autres machines.
- Universalité en termes de modèles de calcul : la thèse de Church-Turing.

3.1 Universalité externe

Toute modification "censée" du modèle ne change pas la classe des fonctions calculables par MT. Donnons quelques exemples intuitifs et examinons en un précisément.

Quelques modèles alternatifs à celui présenté, on peut soit donner des capacités supplémentaires soit donner des limitations par rapport au modèle présenté :

- La tête de lecture peut en une seul étape écrire un nouveau symbole et se déplacer.
- La machine peut travailler sur plusieurs bandes en même temps.
- La machine est non-déterministe : c'est à dire que plusieurs quadruplets peuvent commencer par q, S et la machine choisit aléatoirement parmi les actions possibles lesquelles effectuer.
- On peut mixer les modifications précédentes : une machine à plusieurs bandes avec une tête de lecture qui peut écrire et se déplacer de manière atomique, le tout dirigé par un programme non déterministe.
- Le ruban n'est pas infini dans les deux sens mais seulement infini dans une certaine direction : disons par exemple qu'il y a une case de départ sur laquelle on ne peut jamais aller à gauche mais on peut aller aussi loin qu'on veut sur la droite de cette case de départ (ruban semi-infini).
- La machine a droit à plusieurs têtes de lecture/écriture qui travaillent en simultané sur la bande (en définissant proprement ce qui se passe quand les deux têtes pointent sur la même case).
- La machine n'utilise que deux symboles, le Blanc et la Barre.
- La machine n'utilise qu'un ensemble d'états de taille finie. Voire même on ne peut considérer qu'une seule machine spéciale (une machine de Turing universelle cf section 3.2
- ... (laissez libre cours à votre imagination)

Pour chacun de ses modèles ont peut reprendre le même schéma de définition : description instantanée de la machine, dérivation entre deux description par une machine, calcul, implantation d'une fonction de $\mathbb{N}^n \to \mathbb{N}$ etc. Il se trouve que si on considère la classe de fonctions qu'elles permettent d'implanter on trouvera une classe strictement équivalente à T.

La preuve se fait de la manière suivante. Considérons le modèle où la tête de lecture peut à la fois écrire un symbole et bouger. Cela nous donne le modèle MT_m. La définition de ce modèle se fait de manière similaire à celle des MT :

Définition 6 (Machine de Turing mobile)**.** *On pose $Q = \{q_0, q_1, \ldots\}$ et $S = \{S_0, S_1, \ldots\}$ deux ensembles canoniques infinis dénombrables d'états et de symboles, d'intersection vide, et trois symboles particuliers L, R, I qui ne sont ni dans Q ni dans S.*

Une machine de turing Z_m est la donnée d'un ensemble non vide PZ_m et consistant de quintuplets *d'une des trois formes suivante :*

1. q_i, S_j, S_k, I, q_l
2. q_i, S_j, S_k, L, q_l
3. q_i, S_j, S_k, R, q_l

Un tel ensemble est consistant s'il n'y a pas deux quintuplets différents qui commencent par les deux premiers éléments q_i, S_j (pas d'ambigüité).

On notera $E_{Z_m} \subset Q$ et $S_{Z_m} \subset S$ les ensembles minimaux finis d'états et de symboles utilisés dans P_{Z_m}.

On distinguera deux symboles particuliers S_0, le "blanc", qu'on noter aussi B, et S_1 la "barre" ou le "un", noté également par "|".

La machine dispose d'une bande infinie sur laquelle elle inscrit et lit les symboles. En fait la bande est finie à chaque instant, pour cela on suppose qu'au départ la bande est presque partout blanche : donc toutes les cellules sauf un nombre fini d'entre elles portent le symbole B.

Définition 7 (Description de l'état de la machine de Turing mobile)**.** *On définit les termes suivant :*

- *Une description instantanée : est un mot de $S^*.Q.S^+$*
- *Une descritption instantanée d'une machine Z_m : est un mot de $S^*_{Z_m}.E.S^+_m$*
- *Une expression de bande : est un mot de S^**
- *Une expression de bande d'une machine Z_m : est un mot S^**
- *$\langle\alpha\rangle$: le nombre de | dans le mot α, si c'est une description instantanée*

Définition 8. *Soit Z_m une machine de Turing, α, β deux descriptions instantanées de Z_m avec $\alpha = \gamma q_i S_j \delta$ et $\beta = \eta q S \theta$.*

1. *On dit que α dérive vers β par Z_m, et on note $\alpha \vdash_Z \beta$, si et seulement si on a un des cas suivants :*
   (a) *Soit $\gamma, q_i, S_j, S_k, I, q_L \in P_{Z_m}$, et $\eta = \gamma$, $q = q_L$, $S = S_k$ et $\theta = \delta$.*
   (b) *Soit $\gamma, q_i, S_j, S_k, R, q_L \in P_Z$, et $\eta = \gamma S_k$, $q = q_L$, et*

$$\begin{cases} Si\ \delta = \epsilon & alors \quad \theta = \epsilon, S = B \\ Si\ \delta = S_k\delta' & alors \quad \theta = \delta', S = S_k \end{cases}$$

   (c) *Soit $\gamma, q_i, S_j, S_k, L, q_L \in P_Z$, et $\theta = \gamma S_k \delta$, $q = q_L$, et*

$$\begin{cases} Si\ \gamma = \epsilon & alors \quad \eta = \epsilon, S = B \\ Si\ \gamma = \gamma' S_k & alors \quad \eta = \gamma', S = S_k \end{cases}$$

2. *α est terminale ou finale pour Z_m si pour aucun β on a $\alpha \vdash_{Z_m} \beta$. Autrement dit α est finale si aucun quintuplet de P_{Z_m} ne contient aucun quintuplet commençant par $q_i S_j$.*
3. *Un calcul de Z_0 est une suite finie de descriptions instantanées $\alpha_1, \alpha_2, \ldots, \alpha_p$ telle que pour $1 \leq i < p$ on ait $\alpha_i \vdash_{Z_m} \alpha_{i+1}$ et que α_p est terminale pour Z_m. α_p est la résultante de α_1 par Z_m, et on note*

$$\alpha_p = Res_{Z_m}(\alpha_1)$$

Nous allons maintenant définir la notion de fonction calculable relativement aux machines de Turing mobiles. Il faut pour cela fixer un codage des entrées sur la bande ainsi qu'une manière d'interpréter le résultat du calcul.

Pour tout entier x on associe l'expression de bande $\overline{x} = |^{x+1}$. Il faut mettre $x+1$ barres car il faut pouvoir coder le 0. On étend cette convention aux cortèges d'entiers $\vec{x} = (x_1, x_2, \ldots, x_n)$ auxquels on associe l'expression de bande $\overline{(x_1, x_2, \ldots, x_n)} = \overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$. Le cas particulier d'une fonction sans argument sera traité par une bande intégralement recouverte de symboles blancs.

Définition 9. *Soit Z_m une machine de Turing mobile. Pour tout entier n, on associe à Z_m la fonction partielle Ψ_Z^n de $\mathbb{N}^n$ dans $\mathbb{N}$ en posant :*

$$\Psi_Z^{(n)}(x_1, x_2, \ldots, x_n) = \begin{cases} \langle Res_{Z_m}(q_0\overline{(}x_1, \ldots, x_n))\rangle & \text{si } Res_{Z_m}(q_0\overline{(}x_1, \ldots, x_n)) \text{ est défini} \\ \uparrow & \text{sinon} \end{cases}$$

On peut donc maintenant définir précisément ce que sont les fonctions calculables au sens des machines de Turing mobiles.

Définition 10. *Une fonction f de $\mathbb{N}^n$ dans $\mathbb{N}$ est dit (partielle) MT-calculable s'il existe une machine de Turing mobile Z_m telle que $f = \Psi_{Z_m}^n$.*

On voit donc bien qu'on n'a fait que reprendre les définitions en les adaptants à ce nouveau modèle.

Théorème 1. *$T = T_m$. Autrement dit si une fonction est T-calculable alors elle est MT-calculabe et vice-versa.*

Proof. Il s'agit de démontrer une double inclusion.

- $T \subseteq T_m$: c'est le sens le plus simple car en quelques sortes toute MT est une MT mobile qui n'utilise qu'une partie de ses capacités. Plus précisément, soit Z une machine de Turing. On construit la machine de Turing mobile suivante : pour tout quadruplet (q_i, S_j, S_k, q_L) de Z on associe le quintuplet (q_i, S_j, S_k, I, q_L), pour tout quadruplet (q_i, S_j, X, q_L) (où X est soit R soit L) on associe le quintuplet (q_i, S_j, S_j, X, q_L). L'ensemble des quintuplets associés à PZ forme un ensemble de quintuplets PZ_m qui forment le programme de la machine de Turing mobile Z_m. On peut alors montrer par induction sur la longueur des dérivations et sur n que $Res_{Z_m}(q_0\overline{(}x_1, \ldots, x_n)) = Res_Z(q_0\overline{(}x_1, \ldots, x_n))$. Ce qui implique que toute fonction T-calculable est aussi Tm-calculable.

- $T_m \subseteq T$: dans ce sens il faut transformer un pas de MT mobile en plusieurs de MT "normale". Donc pour chaque quintuplet (q_i, S_j, S_k, X, q_l) il faut donner un équivalent. Si $X = I$ alors on associe le quadruplet (q_i, S_j, S_k, q_L) car la machine mobile ne bougeait pas. Si $X = L$ ou R on doit d'abord écrire le bon symbole puis bouger de la bonne manière, pour cela on va introduire un état intermédiaire q_l^w cela donnera les deux quadruplets suivants : $(q_i, S_i, S_j, q_l^w), (q_l^w), S_j, X, q_l)$. Il faut alors terminer la preuve comme dans le cas précédent sur l'égalité de la résultante des deux machines pour toute configuration initiale.

□

L'équivalence entre tous ces modèles se fait de la même manière : en montrant comment on *compile* un programme d'un modèle de machine 1 vers un modèle de machine 2.

Il existe cependant des limitations qui *restreignent* le modèle de calcul.

- Si la tête de lecture ne peut aller que dans une seule direction.
- S'il n'y a pas au moins deux symboles différents utilisés. En effet sinon cela signifie que la bande est blanche et on ne peut pas entrer de paramètres différents ni lire de résultats (peut être juste étudier le temps que prend le calcul, ou le temps moyen si on a une machine non déterministe de ce type là).
- Si la bande n'est pas de taille infinie (on a un nombre fini de configurations possibles et tout est codable par des automates d'états finis).

3.2 Universalité interne

Il s'agit de transformer toute machine de Turing en un nombre. L'avantage est que cela permet ensuite une manipulation plus abstraite de la notion de programme que de considérer des ensembles de quadruplets. Transformer un programme (l'équivalent d'un compilateur) sera aussi simple que de faire une fonction des entiers sur les entiers.

Définition 11. *Soit $V = S \cup Q \cup \{L, R\}$.*

1. *Le code des lettres de V est donné par la fonction γ :*

s	R	L	q_0	S_0	q_1	S_1	q_2	S_2	$\ldots$	q_i	S_i
$\gamma(s)$	3	5	7	9	11	13	15	17	$\ldots$	$4i+7$	$4i+9$

2. *Le nombre de Gödel d'une chaîne de lettres $M = a_1 a_2 \ldots a_n$, $a_i \in V$ est défini par :*

$$gn(M) = gn(a_1 \ldots a_n) = \begin{cases} \prod_1^n Pr(k)^{\gamma(a_k)} & si\ n > 0 \\ 1 & sinon \end{cases}$$

3. *Le nombre de Gödel d'une chaîne de mots $M = M_1 M_2 \ldots M_n$, $M_i \in V^*$ est défini par :*

$$gn(M) = gn(M_1 \ldots M_n) = \begin{cases} \prod_1^n Pr(k)^{gn(M_k)} & si\ n > 0 \\ 0 & sinon \end{cases}$$

4. *Les nombres de Goedel d'une machine de Turing Z définie par $P_Z = \{M_1, \ldots, M_n\}$ sont les $n!$ nombres $gn(M_{\pi(1)}, \ldots, M_{\pi(n)})$, où π est une permutation sur $\{1, \ldots, n\}$.*

On peut donc assimiler toute machine de Turing à un nombre. Dans la suite si z est un nombre de Gödel d'une machine de Turing on confondra ce nombre avec la machine elle même.

Les nombres de Gödel sont très particuliers. On peut se ramener à l'utilisation de tous les entiers en utilisant une technique d'indexation. On utilisera pour cela une fonction η, définie par induction de la façon suivante :

$$\begin{cases} \eta(0) & = & \mu z[MT(z)] \\ \eta(n+1) & = & \mu z[MT(z) \& z > \gamma(n)] \end{cases}$$

Avec la convention suivante : $\mu z[g(z)]$ est le plus petit z tel que $g(z) = 1$, et $MT(z)$ un prédicat qui est vrai si et seulement si z est un nombre de Gödel d'une machine de Turing. Nous prenons alors la convention suivante :

Définition 12. *On note $\varphi_n^{(k)}$ la fonction $\Psi_Z^{(k)}$, si $z = \gamma(n)$ avec z qui est un nombre de Gödel de Z.*

C'est la Gödelisation standard. Généralement, c'est à dire sauf si c'est explicitement détaillé, φ_n dénote $\varphi_n^{(1)}$.

On peut remarquer :

- Un indice est un indice pour une infinité de fonctions : $\varphi_n^{(1)}, \varphi_n^{(2),\ldots,\varphi_n^{(k)},\ldots}$
- Une même fonction a une infinité d'indices, car on peut toujours rajouter des instructions (quadruplets) inutiles (inaccessibles) à une machine pour en produire une différente qui calcule la même fonction.

Nous accepterons le théorème suivant :

Théorème 2 (Machine de Turing universelle)**.**

$$\exists u \in \mathbb{N} \forall x, y \in \mathbb{N}^2 . \varphi_u(x, y) = \varphi_x(y)$$

u est le code d'une machine de Turing universelle ! En effet, elle prend un nombre x en paramètre puis se comporte comme la machine de code x suivant la gödelisation standard.

3.3 La thèse de Church-Turing

La thèse de Church-Turing est l'énoncé selon lequel tout modèle de calcul effectif est équivalent au modèle des machines de Turing.

Il ne s'agit pas à proprement parler d'un énoncé mathématiques. En effet la thèse ne peut être démontée (dans quel système pourrait elle l'être). Elle a cependant plusieurs "justifications":

- Sociologique : tous les informaticiens et mathématiciens du monde depuis presqu'un siècle ont inventé de nouveaux modèles de calculs et à chaque fois les fonctions calculables dans ces modèles sont exactement le fonctions T-calculables (voire moins si le modèle et trop faible).
- Langage : pour toute machine il faut fournir un plan (sinon il y aurait une partie "magique" inexpliquée), or la manière de fournir un plan est d'écrire un texte (que ce soit une vidéo ou une image ne change rien on peut l'imaginer comme le texte de la suite des bits formants le fichier video ou image).
- Historique : même des modèles de calcul très anciens, comme les équations Diophantiennes. Il s'agit du modèle suivant : on considère des équations sur les entiers en utilisant addition et multiplication, ce sont des polynômes du type $E(x,y) = 47x^2 + 12y^3 - 234$. On peut construire des équations paramétrées $E_a(x,y) = 47x^2 + 12y^3 - a$, on a donc $E_{-234}(x,y) = E(x,y)$. On peut alors définir les ensembles Diophantiens par $\{a \mid E_a(x,y) = 0\}$, c'est l'ensemble des paramètre qui font que l'équations paramétrée a au moins une solution (on teste juste l'existence d'une solution, on ne s'intéresse pas à sa valeur). Les ensembles qu'on peut définir ainsi sont exactement les mêmes que ceux qui sont définissables par machine de Turing (c'est à dire les ensembles dont les fonctions caractéristiques sont Turing calculable).

Une implication particulière de cette thèse est que dès qu'on sait qu'une fonction f est programmable on peut supposer l'existence d'un entier n tel que $f = \varphi_n$. C'est une technique abondamment utilisée en calculabilité.

4 Calculabilité et problèmes insolubles

Il est maintenant possible de montrer formellement une limite fondamentale de l'informatique : on ne peut pas construire de programme qui prendrait en entrée un programme qu'on peut voir comme un entier p (on peut imaginer qu'il s'agit d'un codage du programme P dans un formalisme quelconque qu'on transforme en machine de Turing en vertu de la thèse de Church-Turing), un jeu d'entrées de P, et qui concluerait si oui ou non, P s'arrêtera un jour sur ce jeu d'entrée ou bouclera à tout jamais. Comme nous le verrons ces limites sont biens plus générales que ce simple problème. D'autre part nous verrons que la catégorie de problèmes liés à l'arrêt sont spécifiques d'un type de problème qui n'a pas d'équivalent en mathématiques standard : les problèmes de semi-décision.

Nous commençons par donner des résultats généraux en termes de calculabilité.

4.1 Premiers résultats

Théorème 3 (s-m-n)**.**

$$\begin{aligned}&\forall m \in \mathbb{N}.\forall n \geq 1 \in \mathbb{N}.\exists s_n^m \in T^{(m+1)}.\\ &\forall x, y_1, \ldots, y_m, z_1, \ldots, z_n \in \mathbb{N}^{n+m+1}.\\ &\varphi_x(y_1, \ldots, y_m, z_1, \ldots, z_n) = \varphi^{(n)}_{s_n^m(x,y_1,\ldots,y_m)}(z_1, \ldots, z_n)\end{aligned}$$

Proof. On peut faire une démonstration rapide par la technique de Church. Soit C une classe de fonctions de type $(z_1, \ldots, z_n) \mapsto \varphi_x(y_1, \ldots, y_m, z_1, \ldots, z_n)$. C'est évidemment une classe de fonctions calculables donc dans T. Par conséquent, il existe un moyen effectif (une fonction récursive totale $r^{(n+1)}$) qui permet de passer de cette caractérisation à la gödelisation standard, c'est à dire $(z_1, \ldots, z_n) \mapsto \varphi_x(y_1, \ldots, y_m, z_1, \ldots, z_n) = \varphi_{r(x,y_1,\ldots,y_n)}$.

La démonstration formelle se fait par récurrence sur m pour un n fixé. □

Théorème 4.

$$\exists g \in R^{(2)}.\forall x, y \in \mathbb{N}^2.\varphi_x \circ \varphi_y = \varphi_{g(x,y)}$$

Proof. La démonstration est assez directe en utilisant le théorème s-m-n : on pose $\theta(x, y, z) = \varphi_x(\varphi_y(z)) = \varphi_u(x, \varphi_u(, y, z))$, où u est un indice d'une machine de Turing universelle (cf théorème 2). Il est donc clair que θ est calculable, donc par Church il existe un entier n tel que $\theta = \varphi_n$, on a donc :

$$\theta(x, y, z) = \varphi_n(x, y, z) = \varphi_{s_1^2(n,x,y)}(z)$$

La dernière égalité provenant du théorème smn. On pose donc $g(x, y) = s_1^2(x, y)$ ce qui terine la preuve. □

Théorème 5 (point fixe)**.** *Toute fonction T-calculable totale a un point fixe sur les indices :*

$$\forall f \in R.\exists n \in \mathbb{N}.\varphi_{f(n)} = \varphi_n$$

Proof. Soit Ψ définie par :

$$\Psi(u, x) = \begin{cases} \varphi_{\varphi_u(u)}(x) & \text{si } \varphi_u(u) \downarrow \\ \uparrow & \text{sinon} \end{cases}$$

Ψ est T-calculable : en effet on calcule $\varphi_u(u)$, et, si et quand ce calcul termine on prend le résultat, disons t et on calcule $\varphi_t(x)$ (ce qui est possible d'après le théorème 2). Donc il existe n_0 tel que $\varphi_{n_0} = \Psi$. Par le théorème s-m-n on a l'existence d'une fonction totale calculable g telle que :

$$\Psi(u, x) = \varphi_{n_O}(u, x) = \varphi_{s_1^1(n_0,u)}(x) = \varphi_{g(u)}(x) = \begin{cases} \varphi_{\varphi_u(u)}(x) & \text{si } \varphi_u(u) \downarrow \\ \uparrow & \text{sinon} \end{cases}$$

La composée de f et g est une fonction totale (cf théorème 4)

□

4.2 Problèmes insolubles

Toutes ces définitions permettent maintenant de montrer de manière formelle une limitation fondamentale du calcul (de l'informatique mais comme si l'on tient la thèse de Church pour valide en fait ces résultats s'appliquent à n'importe quel calcul effectivement réalisable). Le problème emblématique de cette limite est le problème de l'arrêt : il ne peut pas exister de programme P qui prendrait en entrée un programme Q et des données D et qui pourrait décider si Q lancé avec comme entrée D va s'arrêter ou pas. Nous verrons que ce simple problème entraine que toute une classe de fonctions ne sont pas calculables. On dit que ce problème est indécidable, c'est à dire non décidable.

Il y a cependant une notion très spécifique à la calculabilité, et très déroutante, qui est celle de problème "semi-décidables" : informellement ce sont des problèmes qui s'arrêtent sur les instances positives et qui soit s'arrête et répondent "non", soit boucle, sur les instances négatives. Ce sont des cas particuliers de programmes indécidables : en fait le problème de l'arrêt est semi-décidable. Pour le résoudre il suffit de lancer le programme sur les bonnes entrées et, s'il s'arrête on répond "oui" sinon ... on attend pour l'éternité et on n'aura jamais la réponsen La subtilité est que cette notion n'est pas auto-duale dans le sens où si un problème est semi-décidable et que son complémentaire est semi-décidable alors il est décidable. Ce qui implique qu'un problème véritablement.

Nous utiliserons les notations suivantes :

- $\varphi_n(x) \downarrow$ signifie que la machine de Turing n s'arrête sur l'entrée x, autrement dit qu'il existe $z \in \mathbb{N}$ tel que $\varphi_n(x) = z$. On pourra également utiliser la notation $\varphi_n(x) = \uparrow$.
- $\varphi_n(x) \uparrow$ signifie que la machine de Turing n ne s'arrête pas sur l'entrée x, autrement dit qu'elle rentre dans une boucle infinie.

Théorème 6 (Indécidabilité du problème de l'arrêt)**.** *Il n'existe pas de fonction totale T-calculable g telle que*

$$g(x,y) = \begin{cases} 1 & si\ \varphi_x(x)\downarrow \\ 0 & si\ \varphi_x(x)\uparrow \end{cases}$$

Proof. La démonstration se fait par l'absurde. Supposons qu'il existe une telle fonction g T-calculable, alors la fonction Ψ définie par

$$\Psi(x) = \begin{cases} 1 & \text{si } g(x,x) = 0 \\ \uparrow & \text{si } g(x,x) = 1 \end{cases}$$

Il est clair qu'alors Ψ est T-caculable car pour la calculer il suffit de lancer une machine qui calcul g (elle existe car g est T-calculable) sur x, x, comme g est une fonction totale elle rend soit 0 soit 1. Dans le cas où c'est 0 il suffit d'effacer lande et de répondre 1, dans le second cas on rentre dans une boucle infinie.

Comme Ψ est calculable, il existe un indice n, tel que $\Psi = \varphi_n$. On calcule $\Psi(n,n)$, on a alors l'un des deux cas suivants :

- Soit $g(n,n) = 0$, c'est-à-dire que $\Psi(n) = \varphi_n(n) = 1$ $\varphi_n(n)\uparrow$. Ce qui n'est pas possible car on ne peut avoir simultanément $\varphi_n(n) = 1$ et $\varphi_n(n)\uparrow$ (car $g(n,n) = 0$).
- Soit $g(n,n) = 1$, c'est-à-dire que $\Psi(n) = \varphi_n(n) = \uparrow$. Ce qui n'est pas possible car on ne eut pas avoir simultanément $\varphi_n(n) = 1$ et $\varphi_n(n)\downarrow$ (car $g(n,n) = 1$).

□

Il s'agit de la technique de diagonalisation de Cantor dont nous avons déjà vu un exemple dasn l'introduction. Essentiellement il s'agit d'une version mathématique du paradoxe du menteur.

4.3 Réductibilité et Indécidabilité

On peut remarquer que dans la démonstration nous n'avons pas utilisé le second argument. En fait nous avons montré qu'un problème plus simple que celui de l'arrêt, car il s'agissait juste de voir si étant donné un entier n, est ce que $\varphi_n(n)$ converge ?, était déjà un problème indécidable.

Traditionnellement on a les notations suivantes :

- $K = \{x \mid x \in \mathbb{N}.\varphi_x(x)\downarrow\}$
- $K_0 = \{(x,y) \mid x, y \in \mathbb{N}.\varphi_x(y)\downarrow\}$

Clairement K est un sous ensemble de K_0. Pour la démonstration on en fait réduit le problème de la décision de K à celui de K_0. C'est à dire que si K_0 était décidable alors K serait décidable, car on pourrait utiliser l'algorithme de décision de K_0 pour décider K. Or nous avons vu dans la démonstration que K n'est pas décidable donc K_0 ne peut pas être décidable.

De façon générale un problème A est réductible à un problème B si une solution pour B donne aussi une solution pour A. En général un problème est identifié à un sous-ensemble de $\mathbb{N}^p$, par exemple le problème de l'arrêt est identifié à K_0. La technique de réduction est une des techniques principales pour montrer qu'un problème est indécidable.

Définition 13. *On dira que pour des ensembles d'entiers A et B, A se T-réduit (ou "se réduit" tout court quand il n'y a pas d'ambigüité) en B s'il existe une fonction f totale T-calculable telle que :*

$$x \in A \iff f(x) \in B$$

On notera ça $A \leq B$.

Proposition 1.

- *Si $A \leq B$ et B est décidable alors A est décidable.*
- *Si $A \leq B$ et A est indécidable alors B est indécidable.*

Proof. Les deux points sont essentiellements équivalents vu qu'ils sont la contraposée l'un de l'autre. Il suffit donc d'en prouver un. □

Nous allons maintenant voir quelques exemples d'utilisation de cette technique pour montrer que des problèmes sont indécidables.

Théorème 7. *$A = \{x \mid x \in \mathbb{N}.\varphi_x$ est une fonction constante $\}$ est un problème indécidable.*

Proof. Soit Ψ définie par :

$$\Psi(x,y) = \begin{cases} 0 & \text{si } \varphi_x(x)\downarrow \\ \uparrow & \text{sinon} \end{cases}$$

Ψ est calculable : il suffit de lancer la machine x sur x si le calcul termine on rend 0 sinon on est dans une boucle infinie ce qui est bien ce que "doit faire" Psi. Donc comme Ψ est calculable il y a une machie de Turing n telle que $\Psi(x,y) = \varphi_n(x,y)$ par la thèse de Church. Maintenant par le théorème s-m-n on a $\varphi_n(x,y) = \varphi_{s_1^1(n,x)}(y)$. Appelons $h = s_1^1(n,x)$, de par le théorème s-m-n h est une fonction totale calculable.

Nous avons les cas suivants :

- Supposons que $x \in K$ alors cela est équivalent à $h(x) \in A$ car $\varphi_{h(x)}(y) = \Psi(x,y) = 0$ c'est bien une fonction constante.
- Supposons que $s \notin K$ alors cela est équivalent $h(x) \notin A$ car $\varphi_{h(x)}(y) = \Psi(x,y) = \uparrow$ or la fonction qui diverge sur toute sses entrées n'est pas une fonction constante.

Autrement dit la fonction T-calculable h fait une réduction de K à A. □

Théorème 8. *Soit $B = \{x \mid \varphi_x(4) = 12\}$, B est indécidable.*

Proof. Considérons la fonction g définie par :

$$g(x,y) = \begin{cases} 12 & \text{si } \varphi_x(x)\downarrow \\ \uparrow & \text{sinon} \end{cases}$$

Cette fonction est évidemment calculable, elle a donc un indice n tel que $g(x,y) = \varphi_n(x,y)$. Par s-m-n on a $\varphi_n(x,y) = \varphi_{s_1^1(n,x)}(y)$. Soit f la fonction $f(x) = s_1^1(n,x)$.

Nous avons les cas suivants :

- Supposons que $x \in K$ alors $\varphi_{f(x)}(y) = \varphi_{s_1^1(n,x)}(y) = g(x,y) = 12$ en particulier quand $y = 4$ donc $f(x) \in B$.
- Supposons que $x \notin K$ alors $\varphi_{f(x)}(y) = \varphi_{s_1^1(n,x)}(y) = g(x,y) = \uparrow$, en particulier quand $y = 4$ on a une fonction qui diverge et qui donc ne vaut pas 12, donc au final $f(x) \notin B$.

Encore une fois nous avons fait une réduction de K à B au moyen d'une fonction, en l'occurrence f, totale et T-calculable. □

On s'aperçoit que cette preuve est très générique, on pourrait prouver de la même manière que $C = \{x \mid \varphi_x(45) = 868\}$ est indécidable en changeant dans la démonstration 4 par 45 et 12 par 868. Mais jusqu'où peut on généraliser cet type de preuves ? Considérons un exemple supplémentaire :

Théorème 9. *Soit $C = \{x \mid \varphi_x(23)\uparrow\}$, C est indécidable.*

Proof. Considérons encore une fois la fonction g définie par :

$$g(x,y) = \begin{cases} 12 & \text{si } \varphi_x(x)\downarrow \\ \uparrow & \text{sinon} \end{cases}$$

que nous avons déjà utilisée dans la démonstration du théorème 8. Si on reprend la même analyse on trouve les cas suivants :

- Supposons que $x \in K$ alors $\varphi_{f(x)}(y) = \varphi_{s_1^1(n,x)}(y) = g(x,y) = 12$ en particulier quand cette fonction ne diverge jamais donc $f(x) \notin C$.
- Supposons que $x \notin K$ alors $\varphi_{f(x)}(y) = \varphi_{s_1^1(n,x)}(y) = g(x,y) = \uparrow$, on a une fonction qui diverge sur toutes ses entrées et en particulier sur l'entrée 23, donc au final $f(x) \in C$.

On vient de faire un réduction du complémentaire de K, qu'on note $\overline{K}$, dans C. Or il est clair que si K n'est pas décidable, alors $\overline{K}$ non plus (la fonction $i : x \mapsto 1 - x$ étant une réduction triviale de K dans $\overline{K}$).

□

On peut en fait généraliser les deux preuves précédentes. Cela donne lieu à un théorème très puissant qui montre que tout ensemble d'indice d'une classe de fonctions calculables est indécidable.

Théorème 10 (Rice)**.** *Soit $\mathcal{C}$ une classe non-triviale (c'est à dire que $\mathcal{C}$ n'est ni T ni l'ensemble vide) de fonctions T-calculables. Alors l'ensemble des indices des machines de Turing qui implantent des fonctions de $\mathcal{C}$ est indécidable. Autrement dit, $\mathfrak{C}$ est indécidable avec :*

$$\mathfrak{C} = \{x \mid x \in \mathbb{N}.\varphi_x \in \mathcal{C}\}$$

Proof. La preuve se découpe en deux parties qui, essentiellement sont la généralisation des théorèmes 8 et 9.
On note $\perp$ la fonction qui diverge sur toute les entrées, $\forall x \in \mathbb{N}.\perp(x) \uparrow$.
On a deux cas: soit $\perp \in \mathcal{C}$ soit $\perp \notin \mathcal{C}$

1. $\perp \notin \mathcal{C}$: Dans ce cas on prend $v \in \mathcal{C}$, ce qui est possible car par hypothèse $\mathcal{C}$ n'est pas vide.

   Considérons la fonction g définie par :

   $$g(x,y) = \begin{cases} v(y) & \text{si } \varphi_x(x) \downarrow \\ \uparrow & \text{sinon} \end{cases}$$

   Cette fonction est évidemment calculable, elle a donc un indice n tel que $g(x,y) = \varphi_n(x,y)$. Par s-m-n on a $\varphi_n(x,y) = \varphi_{s_1^1(n,x)}(y)$. Soit f la fonction $f(x) = s_1^1(n,x)$.

   Nous avons les cas suivants :

   - Supposons que $x \in K$ alors $\varphi_{f(x)}(y) = \varphi_{s_1^1(n,x)}(y) = g(x,y) = v(y)$ donc $\varphi_{f(x)} \in \mathcal{C}$.
   - Supposons que $x \notin K$ alors $\varphi_{f(x)}(y) = \varphi_{s_1^1(n,x)}(y) = g(x,y) = \uparrow$, donc $\varphi_{f(x)} = \perp$ mais par hypothèse $\perp$ n'est pas dans $\mathcal{C}$.

   Nous avons fait une réduction de K à $\mathcal{C}$ au moyen d'une fonction, en l'occurrence f, totale et T-calculable.

2. $\perp \in \mathcal{C}$ Considérons la fonction h définie par, avec $v \notin \mathcal{C}$, ce qui est possible car par hypothèse $\mathcal{C}$ n'est pas l'ensemble des fonctions T-calculables, il y en a donc au moins une à l'extérieur :

   $$h(x,y) = \begin{cases} v(y) & \text{si } \varphi_x(x) \downarrow \\ \uparrow & \text{sinon} \end{cases}$$

   - Supposons que $x \in K$ alors $\varphi_{f(x)}(y) = \varphi_{s_1^1(n,x)}(y) = g(x,y) = v(y)$ donc $\varphi_{f(x)} = v \notin \mathcal{C}$
   - Supposons que $x \notin K$ alors $\varphi_{f(x)}(y) = \varphi_{s_1^1(n,x)}(y) = g(x,y) = \uparrow$, et donc $\varphi_{f(x)} = \perp \in C$.

   On vient de faire un réduction $\overline{K}$, dans C au moyen de la fonction totale T-calculable f.

□

5 Semi-décidabilité

Dans le chapitre précédent nous avons constaté que certains problèmes sont "plus indécidables" que d'autres. Par exemple K est indécidable mais il a la particularité suivante : il existe une manière de savoir si un programme termine, il suffit de le lancer et d'attendre. S'il termine un jour on sera ca able de le dire, par contre s'il ne termine pas on ne le saura jamais. On peut comparer cela avec la décision de $\overline{K}$ pour laquelle on est incapable de fournir avec certitude une réponse que ce soit ou non. On dit que le problème de l'arrêt (la décision de K) est semi-décidable. Cette notion est particulière à l'informatique et est très délicate à manier car fondamentalement asymétrique : "savoir que non" est fondamentalement différent de "ne pas savoir". Autrement dit la prouvabilité et la négation ne commutent pas.

Définition 14. *Un prédicat P sur $\mathbb{N}$ est dit semi-décidable si son extension (c'est à dire l'ensemble des valeurs pour lesquelles ce prédicat est vrai) est le domaine d'une fonction T-Calculable.*

Une manière intuitive de comprendre cette définition est de le voir comme ça, P est semi-décidable s'il existe une fonction T-calculable Ψ définie par:

$$\Psi(x) = \begin{cases} 1 & \text{si } P(x) \text{ est vrai} \\ \uparrow & \text{sinon} \end{cases}$$

Il est clair que tout prédicat décidable est aussi semi-décidable : il suffit de rentrer dans une boucle infinie si le calcul du prédica a rendu "faux".

Comme d'habitude on peut étendre cette notion à la reconnaissance d'ensemble en considérant le prédicat qui est la fonction caractéristique de l'ensemble. Dans ce cas on parle d'ensemble semi-décidable ou semi-récursif.

Cette notion mène à une notion similaire qui est celle d'ensemble récursivement énumérable.

Définition 15. *Un sous-ensemble A de $\mathbb{N}$ est dit récursivement énumérable s'il est vide ou image d'une fonction T-calculable totale.*

$$AR.E. \iff A = 0 \text{ ou } \exists f \in T.Im(f) = A$$

avec $Im(f) = \{y \mid \exists x \in \mathbb{N}.f(x) = y\}$.

C'est une idée très naturelle, quand on a un ensemble Récursivement Enumérable (RE) cela signifie qu'il existe une manière d'énumérer ses éléments en calculant successivement $f(0), f(1), f(2), \ldots$ etc. Le cas de l'ensemble vide est un cas particulier mais il est nécessaire pour avoir les résultats suivants :

Il est à noter que cette définition est la dual de celle d'un ensemble semi-décidable. En effet, la définition d'un ensemble semi-décicable se fait par le domaine alors que celle d'un ensemble récursivement énumérable se fait par l'image. Il se trouve que les deux définitions coïncident, c'est l'objet du théorème fondamental **??**.

Théorème 11. *Soit A un sous-ensemble de $\mathbb{N}$.*

- *Si A est T-décidable alors A est récursivement énumerable.*
- *A est T-décidable si et seulement si A et son complémentaire $\overline{A}$ sont tous les deux RE.*

Proof. □

Théorème 12. *Un ensemble est récursivement énumérable si et seulement s'il est l'extension d'un prédicat semi-décidable. C'est-à-dire :*

$$A \ RE \iff \exists x \in \mathbb{N}.Dom\varphi_x = A$$

avec $Dom(f) = \{x \mid \exists y \in \mathbb{N} f(x) = y\}$.

Proof. On démontre les deux sens séparéments.

- A RE $\implies \exists x \in \mathbb{N}.Dom\varphi_x = A$. On considère la fonction g définie par

$$\begin{cases} g(x,0) = 0 \\ g(x,n+1) = \begin{cases} 1 & \text{si } f(x) = 1 \\ g(x,n) & \text{sinon} \end{cases} \end{cases}.$$

  cette fonction est définie par récurrence et est bien T-calculable car f l'est. Alors la fonction $\Psi(x) = xsg(\mu n[(gx,n) = 1])$, qui calcule le plus petit n tel que $g(x,n) = 1$ applique la fonction signe sg telle que $sg(0) = 0$ et $sg(n+1) = 1$ pour tout entier n et multiplie ce résultat par x est aussi T-calculable. Il se trouve que $Dom(\Psi) = Im(f)$, car si un entier n'est pas dans l'image de f alors on n'aura jamais $g(x,n) = 1$ pour aucun n et la recherche du plus petit sera infinie.

- $\exists x \in \mathbb{N}.Dom\varphi_x = A \implies A$ RE. Pour ce sens nous allons utiliser la technique dite de la "queue d'arronde" (ou "dove tailing"). Il s'agit de lancer successivement la machine x sur toutes les entrées en parallèles, en alternant les calculs, si l'ensemble n'est pas vide on va bien trouver un cas où un calcul va s'arrêter, en itérant n fois ce procédé on peut construire une énumération des éléments de A. Un peu plus précisément (nous ne rentrerons pas dans la construction technique lourde de ce processus mais en donnons un aperçu de haut niveau) on peut considérer un pseudo-algorithme du type :

□