L3 Informatique
Modèles de Calcul
Machines de Turing

Frédéric Prost
frederic.prost@univ-grenoble-alpes.fr

UGA

2021

Intervenants

- Cours (8) - TD - Groupes 1, 2, 3 : Frédéric Prost ;
  e-mail: Frederic.Prost@univ-grenoble-alpes.fr
- TD (8*2) - Groupe 4 : Adrien Chaffangeon ;
  e-mail: Adrien.Chaffangeon@univ-grenoble-alpes.fr

Pages du cours :

- Page web du cours :
  `http://membres-lig.imag.fr/prost/L3_INFO_MCAL_MT/`
- Invitation Discord (TD) :
  `https://discord.com/invite/eKyG2CwUXf`

Plan

Calcul effectif - remarques préliminaires

Qu'est-il possible de calculer "pour de vrai" ?

- Il ne suffit pas de définir une fonction pour qu'on sache concrètement la calculer.

Calcul effectif - remarques préliminaires

Qu'est-il possible de calculer "pour de vrai" ?

- Il ne suffit pas de définir une fonction pour qu'on sache concrètement la calculer.

  $f(n) = 1$ si le n-ième tirage du loto est ...

Calcul effectif - remarques préliminaires

Qu'est-il possible de calculer "pour de vrai" ?

- Il ne suffit pas de définir une fonction pour qu'on sache concrètement la calculer.

  $f(n) = 1$ si le n-ième tirage du loto est ...

  L'équation de Schrödinger ?

Calcul effectif - remarques préliminaires

Qu'est-il possible de calculer "pour de vrai" ?

- Il ne suffit pas de définir une fonction pour qu'on sache concrètement la calculer.

  $f(n) = 1$ si le n-ième tirage du loto est ...

  L'équation de Schrödinger ?

- Il faut un programme (de taille finie), mais on ne veut pas se limiter vis-à-vis de contraintes physiques (taille finie mais non bornée de la mémoire par exemple).

Calculabilité intuitive

Des fonctions évidemment calculables :

- addition, calcul du nième nombre premier, pgcd etc.
- La composition de deux fonctions calculables est calculable.
- Les fonctions définies par récurrence le sont aussi, par exemple :

$$U_n = \begin{cases} U_0 \text{ si } n = 0 \\ (3U_{n-1} + 1)/2 \text{ si } U_n \text{ pair} \\ U_{n-1}/2 \text{ sinon} \end{cases}$$

Calculabilité intuitive

Des fonctions moins intuitivement calculables.

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

Calculabilité intuitive

Des fonctions moins intuitivement calculables.

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

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

Calculabilité intuitive

Des fonctions moins intuitivement calculables.

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

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

- Personne ne sait si f_1 est calculable.
- f_2 est calculable...mais personne ne sait la calculer.

Première difficultés : la partialité

- Supposons qu'on ait une définition : elle doit forcément inclure les fonctions partielles.

Première difficultés : la partialité

- Supposons qu'on ait une définition : elle doit forcément inclure les fonctions partielles.
- Soit $\mathcal{C} \subset \mathcal{F}^{(1)}$ une sous-classe de fonctions totales calculables.
- Comme il s'agit d'une classe de fonctions calculables il faut un programme écrit dans un langage quelconque. On a donc :

$$\mathcal{C} = \{f_0, f_1, \ldots, f_n, \ldots\}$$

Première difficultés : la partialité

- Supposons qu'on ait une définition : elle doit forcément inclure les fonctions partielles.
- Soit $\mathcal{C} \subset \mathcal{F}^{(1)}$ une sous-classe de fonctions totales calculables.
- 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 $\mathcal{C}$

Première difficultés : la partialité

- Supposons qu'on ait une définition : elle doit forcément inclure les fonctions partielles.
- Soit $\mathcal{C} \subset \mathcal{F}^{(1)}$ une sous-classe de fonctions totales calculables.
- 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 $\mathcal{C}$
- Seule possibilité : si on a des fonctions partielles il est possible que $f_n(n)$ ne soit pas un nombre.
- Problème de l'arrêt : absence de réponse ?

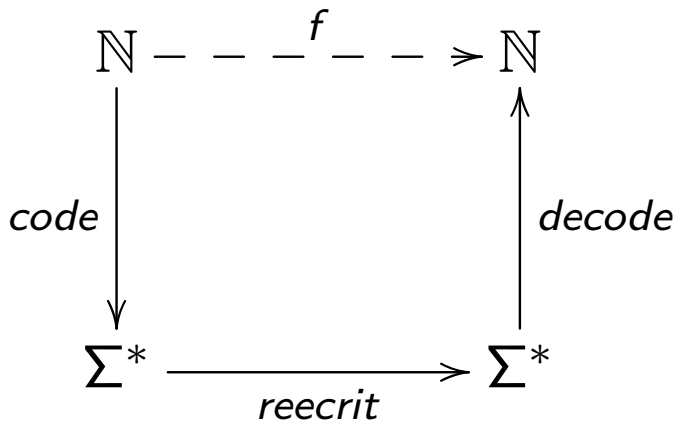
Plan du cours

1. Définition du modèle des machines de Turing.
2. Universalités.
3. Problèmes indécidables.
4. Complexités.

Plan

Formalisation précise d'un modèle de calcul

- Tout doit être explicite : pas de magie à n'importe quelle étape.
- Interpréter les entrées et les sorties:

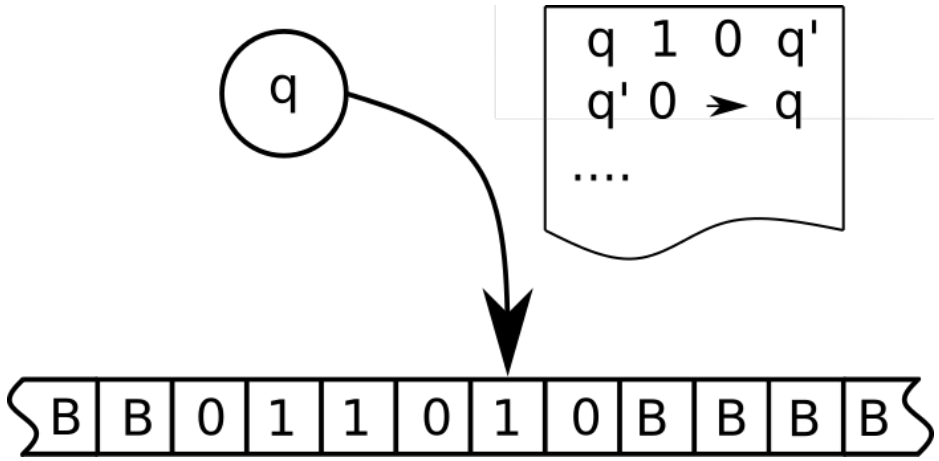


avec Σ un alphabet.

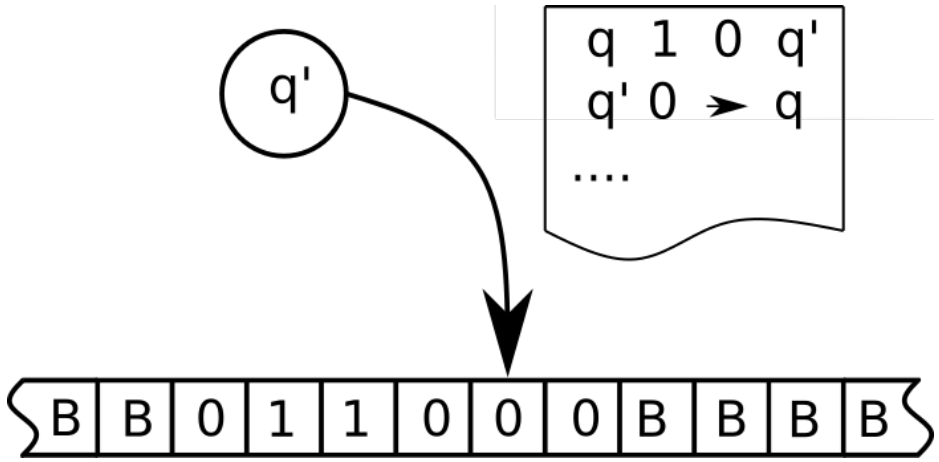
Machine de Turing - présentation informelle

Une machine de Turing est composée de trois parties :

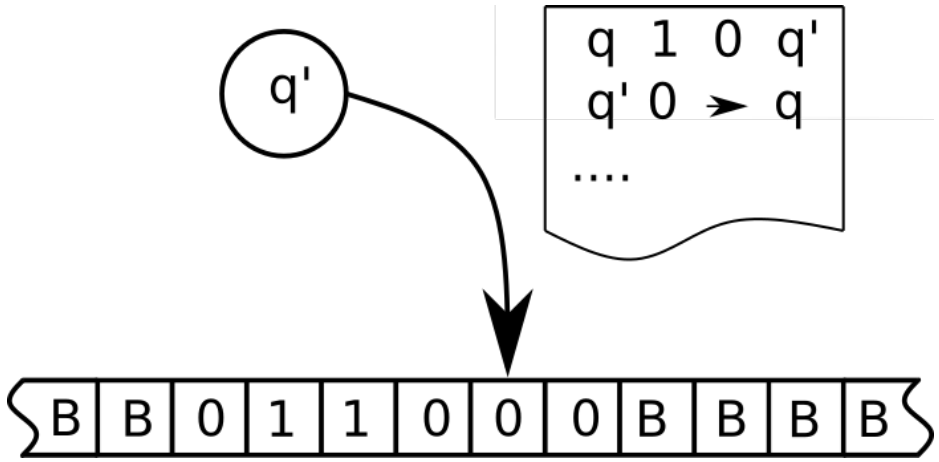
- Une tête de lecture/écriture qui est dans un **état** donné.
- Une bande (infinie) sur laquelle on peut écrire et lire des symboles.
- Un programme qui dit quoi faire quand on est dans un état q et que la tête de lecture pointe vers S.

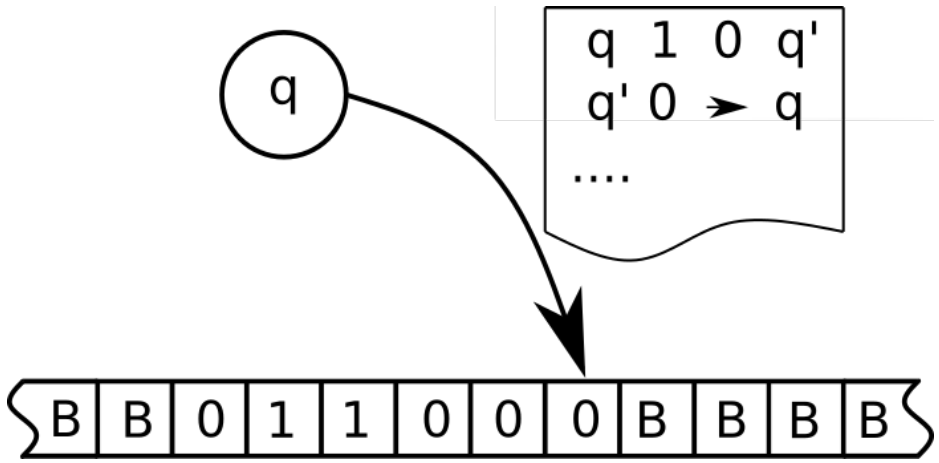


Machine de Turing - présentation informelle - éxécution



Machine de Turing - présentation informelle - éxécution





Machines de Turing - Définition mathématique

Definition (Machine de Turing basique)

On pose $Q = \{q_0, q_1, \ldots\}$ et $S = \{S_0, S_1, \ldots\}$, 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

Programme consistant s'il n'y a pas deux quadruplets différents qui commencent par q_i, S_j .
On note $S_0 = B$ (le blanc) et $S_1 = |$ (ou $S_1 = 1$).

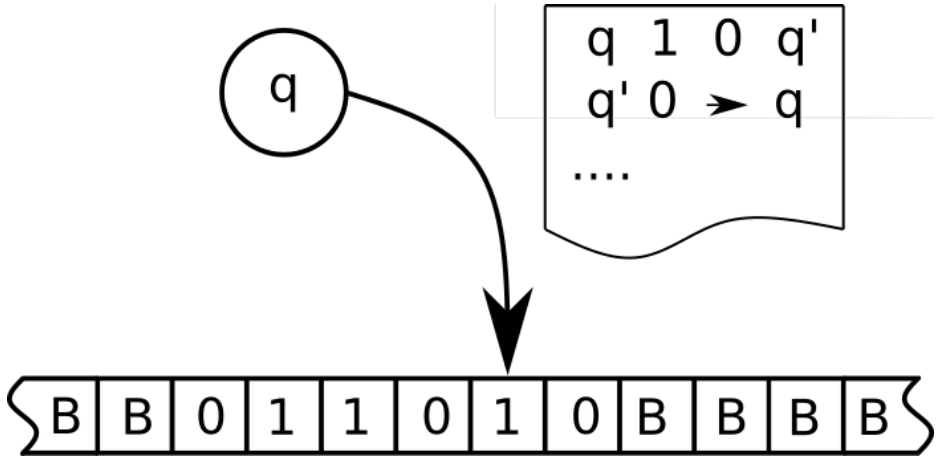
Formalisation du calcul - état de la machine

Definition (Description de l'état de la machine de Turing)

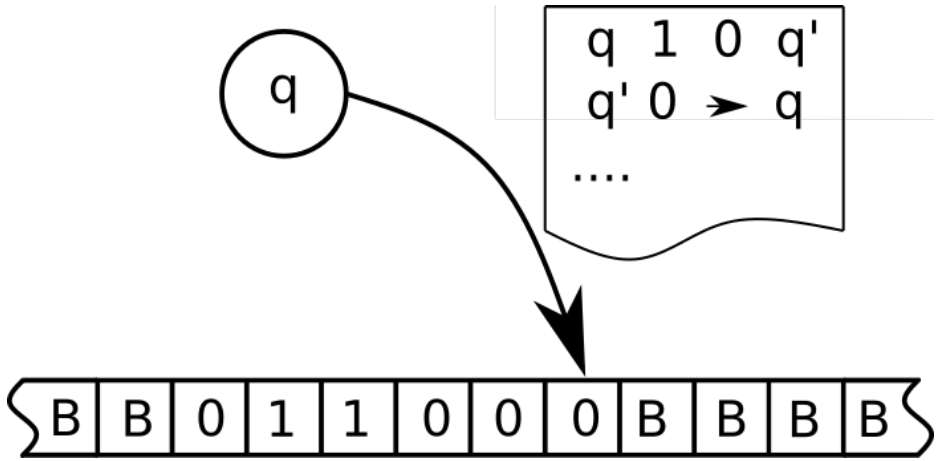
On définit les termes suivant :

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

Exemple de description de la machine



est décrit par $0110q10$



est décrit par $01100q'0$

Formalisation du calcul - un pas de calcul

Definition (Pas de calcul)

Soit Z une machine de Turing, α, β deux descriptions instantanées de Z avec $\alpha = \gamma q_i S_j \delta$ et $\beta = \eta q S \theta$.
$\alpha \vdash_Z \beta$, ssi on a un des cas suivants :

Formalisation du calcul - un pas de calcul

Definition (Pas de calcul)

Soit Z une machine de Turing, α, β deux descriptions instantanées de Z avec $\alpha = \gamma q_i S_j \delta$ et $\beta = \eta q S \theta$.
$\alpha \vdash_Z \beta$, ssi on a un des cas suivants :

1. Soit $q_i, S_j, S_k, q_L \in P_Z$, et $\eta = \gamma$, $q = q_L$, $S = S_k$ et $\theta = \delta$.

Formalisation du calcul - un pas de calcul

Definition (Pas de calcul)

Soit Z une machine de Turing, α, β deux descriptions instantanées de Z avec $\alpha = \gamma q_i S_j \delta$ et $\beta = \eta q S \theta$.
$\alpha \vdash_Z \beta$, ssi on a un des cas suivants :

1. Soit $q_i, S_j, S_k, q_L \in P_Z$, et $\eta = \gamma$, $q = q_L$, $S = S_k$ et $\theta = \delta$.
2. Soit $q_i, S_j, R, q_L \in P_Z$, et $\eta = \gamma S_j$, $q = q_L$, et

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

Formalisation du calcul - un pas de calcul

Definition (Pas de calcul)

Soit Z une machine de Turing, α, β deux descriptions instantanées de Z avec $\alpha = \gamma q_i S_j \delta$ et $\beta = \eta q S \theta$.
$\alpha \vdash_Z \beta$, ssi on a un des cas suivants :

1. Soit $q_i, S_j, S_k, q_L \in P_Z$, et $\eta = \gamma$, $q = q_L$, $S = S_k$ et $\theta = \delta$.
2. Soit $q_i, S_j, R, q_L \in P_Z$, et $\eta = \gamma S_j$, $q = q_L$, et

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

3. Soit $q_i, S_j, L, q_L \in P_Z$, et $\theta = \gamma S_j \delta$, $q = q_L$, et

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

Formalisation du calcul - un calcul de MT

- α est terminale ou finale pour Z si pour aucun β on a

$$\alpha \vdash_Z \beta$$

- 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

$$\alpha_p = Res_Z(\alpha_1)$$

- On note $\overline{x}$ le mot formé de $x+1$ symboles $|$.
  On note $\overline{x_1, x_2}$ le mot formé de $x_1 + 1$ symboles $|$ puis un blanc puis $x_2 + 1$ symboles $|$.
  Ces notations permettent de donner les arguments sur lesquels on lance les calculs de la machine de Turing.

Formalisation du calcul - fonction associée à une machine de Turing

Definition (Fonctions associées à une machine de Turing)

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

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

Formalisation du calcul - fonctions Turing-calculable

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

Definition (Fonction T-calculables)

1. Une fonction f de $\mathbb{N}^n$ dans $\mathbb{N}$ est dite (partielle) T-calculable s'il existe une machine de Turing Z telle que
$$f = \Psi_Z^{(n)}$$
2. On appelle T l'ensemble de toutes les fonctions T-calculables.
3. On note $T^{(n)}$ les fonctions T-calculables d'arité n.

Exemples

- Implanter la fonction d'addition $add : x, y \mapsto x + y$.
- Construire une machine de Turing Z qui répond n quand on lui fournit n arguments.
- Implanter une fonction qui teste la parité : répond 0 si son entrée est paire et 1 sinon.

Plan

Codages - Ensembles/Prédicats/Langages

Les notions suivantes sont équivalentes. Suivant le problème considéré on pourra utiliser l'une ou l'autre des notions.

- Fonctions *calculables* : comment utiliser une MT pour implanter une fonction.
- Ensembles *décidables* : utiliser une MT pour savoir si $x \in A$
- Prédicats *décidables* : ce sont des fonctions qui répondent un booléen "vrai" ou "faux".
- Langages *reconnaissables* : tel langage est il reconnaissable par machine de Turing?

Codages - Ensembles/Prédicats/Langages

Les notions suivantes sont équivalentes. Suivant le problème considéré on pourra utiliser l'une ou l'autre des notions.

- Fonctions *calculables* : comment utiliser une MT pour implanter une fonction.
- Ensembles *décidables* : utiliser une MT pour savoir si $x \in A$
- Prédicats *décidables* : ce sont des fonctions qui répondent un booléen "vrai" ou "faux".
- Langages *reconnaissables* : tel langage est il reconnaissable par machine de Turing?

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 note π_1, π_2 les fonctions de $\mathbb{N} \to \mathbb{N}$ telles que $\pi_1(\mathcal{C}(x, y)) = x$ et $\pi_2(\mathcal{C}(x, y)) = y$.
$\mathbb{N}^n$ est équipotent à $\mathbb{N}$ pour tout $n \geq 1$

Ensembles

- 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.

Ensembles

- 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.
- Symmétriquement 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

- 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 $\Psi_Z^{(1)}(x) = 0$.
- La fonction caractéristique d'un ensemble est un prédicat particulier, la notion de prédicat-calculable est équivalente à celle d'ensemble-calculable (qui lui même est équivalent à la notion de fonction calculable).

Langages

- Langage : étant donné un mot m sur un alphabet Σ, une MT décide 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 $m = l_1 l_2\ ldots l_n$ par $code(m) = \Pi_{i=1}^{n} pr(i)^{pr(l_i)}$ (bijection par réusltat fondamental de l'arithmétique).
- L'appartenance de m au langage L devient équivalente à l'appartenance de $code(m)$ à l'ensemble $\{code(m) \mid m \in L\}$.

Les Machines de Turing - un modèle universel

Le modèle des MT est simple et clairement mécanique. 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.

Universalité externe

- La tête de lecture peut écrire et se déplacer.
- La machine peut travailler sur plusieurs bandes en même temps.
- La machine est non-déterministe : plusieurs quadruplets peuvent commencer par q, S, la machine choisit aléatoirement parmi les actions possibles.
- On peut mixer les modifications précédentes : non-déterminisme, plusieurs bandes, des têtes lecture-écriture-déplacement...
- Le ruban n'est pas infini dans les deux sens mais seulement infini dans une certaine direction.
- La machine a plusieurs têtes de lecture/écriture qui travaillent en simultané sur une bande.
- La machine n'utilise que deux symboles, le Blanc et la Barre.
- La machine n'utilise qu'un ensemble d'états de taille finie (machine universelle).
- ...

Equivalence entre modèles de Machine de Turing - 1

Exemple du modèle où la tête de lecture peut à la fois écrire un symbole et bouger : modèle MT_m.

Equivalence entre modèles de Machine de Turing - 1

Exemple du modèle où la tête de lecture peut à la fois écrire un symbole et bouger : modèle MT_m.

Definition (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

Toutes les autres définitions suivent mutatis mutandi.

Equivalence entre modèles de Machine de Turing - 2

Il faut montrer qu'on peut simuler toute machine de Turing "normale" par une machine de Turing "mobile" et vice-versa.

Premier sens :
Soit Z une machine de Turing normale donnée. On définit Z_m par :
pour tout quadruplet $quad \in programmeZ$ on a trois possibilités distinctes :

1. $quad = (q_i, S_j, S_k, q_l)$ alors on a q_i, S_j, S_k, I, q_l dans Z_m.
2. $quad = (q_i, S_j, L, q_l)$ alors on a a q_i, S_j, S_l, L, q_l dans Z_m.
3. $quad = (q_i, S_j, R, q_l)$ alors on a a q_i, S_j, S_l, R, q_l dans Z_m.

On montre que

$$\forall n \in \mathbb{N}.\Psi_Z^{(n)} = \Psi_{Z_m}^{(n)}$$

Equivalence entre modèles de Machine de Turing - 3

Inversement pour tout quintuplet q_i, S_j, S_k, M, q_l de Z_m, on a les possibilités suivantes :

1. $M = I$ alors on a q_i, S_j, S_k, q_l dans Z
2. $M = L$ alors on a q_i, S_j, S_k, q'_l et q'_l, S_k, L, q_l dans Z
3. $M = R$ alors on a q_i, S_j, S_k, q'_l et q'_l, S_k, R, q_l dans Z

On montre aussi que dans ce cas

$$\forall n \in \mathbb{N}. \Psi_Z^{(n)} = \Psi_{Z_m}^{(n)}$$

la démonstration se fait par double récurrence sur n et k la longueur de la dérivation des calcul des machines de Turing.

Machine Universelle

Il existe une machine de Turing qui peut simuler toutes les autres : la machine de Turing Universelle... qu'on appelle "ordinateur".

- L'idée est d'associer un entier à une machine de Turing.
- Ainsi pour éxécuter la machine numéro n sur l'entrée x on a besoin que d'une machine universelle Z_{univ} telle que

$$\Psi_{Z_{univ}}(n, x) = \Psi_{Z_n}(x)$$

avec Z_n la machine de Turing de code n.

- Transformer un programme (compilation) sera vu comme une fonction des entiers dans les entiers.

Codage de Gödel - 1

Definition

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

- 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	$\ldots$	$4i+7$	$4i+9$

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

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

Codage de Gödel - 2

Definition

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

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

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

Codage de Gödel - 2

Definition

On note $\varphi_n^{(k)}$ la fonction $\Psi_Z^{(k)}$, si n 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.

Gödelisation standard

Definition

On note $\varphi_n^{(k)}$ la fonction $\Psi_Z^{(k)}$, si n 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.

Gödelisation standard - exemple

- La machine Z_{id} calculant l'identité est implantée par $\{(q_0, |, B, q_0)\}$. Son numéro de Gödel est :

$$gn(q_O, |, B, q_0) = 2^7 3^{13} 5^9 7^7 = 32824838659725000000$$

$$gn(Z_{id}) = 2^{32824838659725000000}$$

- La machine Z_{+2} calculant $x \mapsto x + 2$ est implantée par $\{(q_0, |, L, q_1), (q_1, B, |, q_1)\}$. Ses codes de Gödel sont :

$$\begin{aligned} gn(q_0, |, L, q_1) &= 2^7 3^{13} 5^5 7^{11} = 1260999001951995600000 \\ gn(a_1, B, |, q_1) &= 2^{11} 3^{13} 5^{13} 7^{11} = 7881243762199972500000000000 \end{aligned}$$

$$gn(Z_{+2}) = \begin{cases} 2^{1260999001951995600000}\, 3^{7881243762199972500000000000} \\ \\ 2^{7881243762199972500000000000}\, 3^{1260999001951995600000} \end{cases}$$

Théorème (Machine de Turing universelle)

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

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.

La thèse de Church

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

Différentes justifications de cette thèse qui n'est pas "scientifiquement" établie.

- Sociologiques.
- Historiques.
- Linguistiques.

La filière de Church

Une implication particulière de la thèse de Church est :

- Supposons qu'on puisse justifier que la fonction f est programmable,
- On peut inférer l'existence d'un entier n tel que $f = \varphi_n$.

C'est une technique abondamment utilisée en calculabilité.

Plan

Limites du calcul

- Quelles sont les limites de ce qui est calculable ?
- Attention : calculable en théorie, ce qui est différent de calculable en pratique.
  - Calculabilité : ce qu'aucun ordinateur (Thèse de Church) ne pourra jamais calculer en dehors d'aspects technologiques.
  - Complexité : une complexité exponentielle est, en pratique, incalculable (ordinateurs quantiques).
- On va raisonner sur l'ensemble des machines de Turing (énumération) pour montrer que certaines classes ne sont pas calculables.
  $\Longrightarrow$ A l'image de ce que vous avez vu pour les automates d'états finis et les langages réguliers (Lemme de la pompe).

Raisonnements sur les indices de MT

- Il faut penser qu'une machine est un programme est que toute machine est un entier.

  $\Longrightarrow$ les programmes sont juste des entiers (la liste des 0 et 1 qui constituent le fichier par exemple).
- La thèse de Church est un outil central : si une fonction est "visiblement" programmable cela implique qu'il existe une MT d'un certain indice qui la calcule.
- Exemples concrets :
  1. application partielle (théorème s-m-n).
  2. Théorème du point fixe.
  3. Problème de l'arrêt.

Théorème s-m-n

Théorème (s-m-n)

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

- Il s'agit d'une version théorique de l'application partielle, ou de la propagation de constante en compilation.
- La preuve est technique (une récurrence sur n et m).
- Théorème technique important.

Application du théorème s-m-n

Théorème

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

Proof.

Posons $\theta(x, y, z) = \varphi_x(\varphi_y(z)) = \varphi_u(x, \varphi_u(y, z))$, où u est un indice d'une machine de Turing universelle. θ est calculable, donc par Church il existe un entier n tel que $\theta = \varphi_n$, on a :

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

La dernière égalité provenant du théorème s-m-n. On pose donc $g(x, y) = s_1^2(x, y)$ ce qui termine la preuve. □

Théorème du point fixe

Théorème (point fixe)

Toute fonction T-calculable totale a un point fixe sur les indices :

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

Preuve :
Soit Ψ définie par :

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

Théorème du point fixe démonstration

Ψ est T-calculable : en effet on calcule $\varphi_v(v)$, et, si et quand ce calcul termine on prend le résultat, disons t et on calcule $\varphi_t(x)$ (en utilisant une machine de Turing universelle). 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 :

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

La composée de f et g est une fonction totale (cf théorème 3)

Problème de l'arrêt

- Semble être un problème bizarre mais en fait central : "est il possible de construire un programme qui prévoit si n'importe quel programme va s'arrêter à un moment donné?".
- A l'origine de l'invention du modèle des Machines de Turing.
- La réponse négative engendre une révolution industrielle/culturelle.
- Liens avec la logique : par exemple conjecture de Goldbach.

Problème de l'arrêt

- $\varphi_n(x)\downarrow$ signifie que la machine de Turing n s'arrête sur l'entrée x, autrement dit qu'il existe $z \in \mathbb{N}$ tel que $\varphi_n(x) = z$.
- $\varphi_n(x)\uparrow$ 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 (Indécidabilité du problème de l'arrêt)

Il n'existe pas de fonction totale T-calculable g telle que

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

Démonstration du théorème d'indécidabilité de l'arrêt

Proof.

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

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

Démonstration du théorème d'indécidabilité de l'arrêt

Proof.

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

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

Ψ est calculable, donc il existe un indice n, tel que $\Psi = \varphi_n$. On calcule $\Psi(n,n)$:

Démonstration du théorème d'indécidabilité de l'arrêt

Proof.

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

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

Ψ est calculable, donc il existe un indice n, tel que $\Psi = \varphi_n$. On calcule $\Psi(n,n)$:

- 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 $g(n,n) = 0$).

Démonstration du théorème d'indécidabilité de l'arrêt

Proof.

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

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

Ψ est calculable, donc il existe un indice n, tel que $\Psi = \varphi_n$. On calcule $\Psi(n,n)$:

- 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 $g(n,n) = 1$).

Diagonalisation

- Le problème de l'arrêt est une version "mécanique" de la preuve d'incomplétude de Gödel.
- Il s'agit d'une version mathématique du paradoxe du menteur: l'argument diagonal de Cantor.
- La diagonalisation est une technique de construction de paradoxe très puissante et montre la limite du langage : avec un texte de taille n on ne peut écrire "que" $|\Sigma|^n$ textes différents. Or l'ensemble des parties d'un ensemble à p éléments est 2^p.

Une théorie de tout est impossible

- Résultat technique : pas une spéculation philosophique.

Une théorie de tout est impossible

- Résultat technique : pas une spéculation philosophique.
- Deux ingrédients pour lancer la "machine à paradoxes" :
  1. Une explication est un lien entre une phrase et un "phénomène".
  2. Une explication est en elle-même un fait du monde : le morceau de papier où elle est écrite, la mémoire de l'ordinateur dans laquelle elle est enregistrée.

Une théorie de tout est impossible

- Résultat technique : pas une spéculation philosophique.
- Deux ingrédients pour lancer la "machine à paradoxes" :
  1. Une explication est un lien entre une phrase et un "phénomène".
  2. Une explication est en elle-même un fait du monde : le morceau de papier où elle est écrite, la mémoire de l'ordinateur dans laquelle elle est enregistrée.

1 $\Longrightarrow$ énumération des explications : $e_1, e_2, \ldots, e_n, \ldots$.

Une théorie de tout est impossible

- Résultat technique : pas une spéculation philosophique.
- Deux ingrédients pour lancer la "machine à paradoxes" :
  1. Une explication est un lien entre une phrase et un "phénomène".
  2. Une explication est en elle-même un fait du monde : le morceau de papier où elle est écrite, la mémoire de l'ordinateur dans laquelle elle est enregistrée.

1 $\Longrightarrow$ énumération des explications : $e_1, e_2, \ldots, e_n, \ldots$.

2 $\Longrightarrow$ nombre de phénomènes est au moins $\mathbb{N}^{\mathbb{N}}$:
"existe-t-il un lien entre l'explication j et l'ensemble des explications $\{x \mid x \in \mathbb{N}.\text{une propriété sur } x\}$

- Or $\mathbb{N}$ et $\mathbb{N}^{\mathbb{N}}$ ne sont pas équipotents (Cantor).

Comment montrer qu'une fonction n'est pas calculable ?

- On peut trouver une contradiction pour chaque cas comme pour le problème de l'arrêt (difficile).
- On peut utiliser la technique de réduction de problème :

  Pour montrer que f n'est pas calculable je montre que si je savais calculer f alors je pourrais calculer une autre fonction g dont on sait par ailleurs qu'elle 'n'est pas calculable $\Longrightarrow$ cela donne une contradiction.
- Un exemple : savoir si la valeur d'une variable change au cours de l'éxécution d'un programme.

Réductions de problèmes - 1

On a les notations suivantes :

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

La démonstration de l'indécidabilité de l'arrêt nous a montré que K est indécidable et que comme $K \subseteq K_0$ alors K_0 était aussi indécidable. Pour la démonstration du fait que K est indécidable, on a réduit le problème de l'appartenance à K à celui de de l'appartenance à K_0 :

- Si K_0 était décidable alors K serait décidable !
- Or 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.

Réductions de problèmes - 2

Un problème est un sous-ensemble de $\mathbb{N}^p$ (ensemble des instances pour lequel la réponse est oui).
Par exemple le problème de l'arrêt est identifié à K_0.

Definition

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 note ça $A \leq B$.

Proposition

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

Exemples de réductions - 1 - énoncé

Théorème

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

Exemples de réductions - 1 - preuve

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, donc il y a une machine de Turing n telle que $\Psi(x,y) = \varphi_n(x,y)$ par la thèse de Church. i
Par théorème s-m-n on a $\varphi_n(x,y) = \varphi_{s_1^1(n,x)}(y)$. Appelons $h(x) = s_1^1(n,x)$.
Nous avons les cas suivants :

- Si $x \in K$ alors cela est équivalent à $h(x) \in A$ car $\varphi_{h(x)}(y) = \Psi(x,y) = 0$.
- Si $s \notin K$ alors cela est équivalent $h(x) \notin A$ car $\varphi_{h(x)}(y) = \Psi(x,y) = \uparrow$

h est une réduction de K à A. □

Exemples de réductions - 2 - énoncé

Théorème

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

Exemples de réductions - 2 - preuve

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 calculable, donc il existe 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 $f(x) = s_1^1(n,x)$.

- 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 $f(x) \notin B$.

f est une réduction de K à B. □

Généralisation - Théorème de Rice

Théorème (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 :

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

La preuve se découpe en deux parties (généralisation des exemples précédents).
On note $\perp$ la fonction qui diverge sur toute les entrées, $\forall x \in \mathbb{N}.\perp(x) \uparrow$.
Les deux cas sont : $\perp \in \mathcal{C}$ ou $\perp \notin \mathcal{C}$

Théorème de Rice - Démonstration cas 1

Si $\perp \notin \mathcal{C}$: Dans ce cas on prend $v \in \mathcal{C}$ ($\mathcal{C} \neq \emptyset$).
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 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.

Théorème de Rice - Démonstration cas 2

Si $\perp \in \mathcal{C}$: soit $v \notin \mathcal{C}$ ($\mathcal{C} \neq T$).
Considérons la fonction h définie par

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

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

- 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 $\varphi_{f'(x)} = \perp \in \mathcal{C}$.

Réduction $\overline{K}$, dans $\mathcal{C}$ au moyen de la fonction totale T-calculable f.

Théorème de Rice - Signification

- Il est impossible d'avoir un programme qui certifie qu'un programme suive un cahier des charges défini de manière fonctionnelle.
  - Les IA ne remplaceront ni mathématiciens ni programmeurs.
  - Il n'y a pas d'anti-virus (informatique) qui marche à tous les coups.
- Attention le théorème de Rice porte sur les spécifications fonctionnelles entrée/sortie et pas sur le fonctionnement de la machine :
  savoir si un programme fait au moins k pas de calcul est décidable !

Problème de Rado

- Les problèmes indécidables vus jusque là sont abstraits: ensemble d'indices de fonctions....
- La fonction des "castors affairés" est plus concrète et semble calculable à première vue :

  la fonction des castors affairés est celle qui à tout entier n associe le nombre maximum de symboles qu'une machine de Turing (d'un modèle précis) peut écrire en démarrant sur une bande uniformément blanche en ayant n états au plus.

- Semble calculable : on lance toutes les machines de taille n (ça fait beaucoup mais c'est fini) et on note le plus grand résultat...

Problème de Rado

- Les problèmes indécidables vus jusque là sont abstraits: ensemble d'indices de fonctions....
- La fonction des "castors affairés" est plus concrète et semble calculable à première vue :

  la fonction des castors affairés est celle qui à tout entier n associe le nombre maximum de symboles qu'une machine de Turing (d'un modèle précis) peut écrire en démarrant sur une bande uniformément blanche en ayant n états au plus.

- Semble calculable : on lance toutes les machines de taille n (ça fait beaucoup mais c'est fini) et on note le plus grand résultat...
- Derrière se cache le problème de l'arrêt : quand est ce qu'on sait que toutes les machines qui s'arrêtent un jour sont arrêtées ?

Formalisation du problème - Modèle considéré

Les machines de Turing peuvent écrire et se déplacer en même temps. Les seuls symboles utilisés sont le blanc et la barre: $B, |$. Les programmes sont des quintuplets de la forme $(q_i, S_j, S_k, D_l, q_m)$, où D_l est soit R, soit L. La tête de lecture ne peut pas rester immobile.

Ces machines ont un état spécial, noté h, à partir duquel aucune transition n'est possible. Les états d'une machine de taille n sont dénotés par les entiers allant de 1 à n, plus l'état de halte h. Une machine de taille n comporte donc $n+1$ états en tout. L'état initial est l'état 1. Les calculs se font à partir d'une bande uniformément blanche.

Exemple

$$(1, B, |, R, 2) \quad (1, |, |, L, 3) \quad (2, B, |, L, 1)$$
$$(2, |, |, R, 2) \quad (3, B, |, L, 2) \quad (3, |, |, R, h)$$

En partant de la bande blanche on a

$$\begin{array}{llllll} 1B & \vdash_M |2B & \vdash_M 1|| & \vdash_M 3B|| & \vdash_M 2B||| & \vdash_M 1B|||| \\ ||2||| & \vdash_M |||2|| & \vdash_M ||||2| & \vdash_M |||||2B & \vdash_M |||||1| & \vdash_M ||||3|| \end{array}$$

On voit donc que pour la machine M il faut 13 transitions avant d'arriver dans l'état final et que le nombre de $|$ sur la bande quand la machine stoppe est de 6. Ainsi cette machine montre que $\Sigma(3) \geq 6$. Il a été montré que $\Sigma(3) = 6$.

Définition de fonctions calculables au sens de Rado

M calcule $f(n)$ sera une machine qui commence ses calculs avec n symboles | écrits consécutivement sur la bande. La tête de lecture-écriture pointant sur le premier | du bloc initial.
La machine s'arrête après un nombre fini d'étapes et il y a un bloc de $f(n)$ symboles | consécutifs sur la bande situé un blanc après l'argument qui été conservé. A la fin du calcul la tête pointe sur le premier symbole | du résultat (et sur un blanc si la valeur de la fonction est 0.
Schématiquement on peut représenter la situation comme cela :

- Configuration initiale (pour éviter les confusions on notera des parenthèses autour de l'état, qui est un entier, de la tête de lecture/écriture) : $(1)\overbrace{|\cdots|}^{x}$
- Configuration finale : $\overbrace{|\cdots|}^{x} Bq_{fin} \overbrace{|\cdots|}^{f(x)}$

La fonction de Rado n'est pas calculable

Théorème ($n \mapsto \Sigma(n)$ n'est pas calculable)

La fonction $n \mapsto \Sigma(n)$ n'est pas calculable par machine de Turing.

La stratégie de la preuve est de montrer que pour n'importe quelle fonction calculable f alors il existe une constante n_0 telle que pour tout n plus grand que n_0 on ait $\Sigma(n) > f(n)$.

Autrement dit Σ majore strictement toutes les fonctions calculables à partir d'un certain seuil.

Démonstration (1)

- Soit f une fonction calculable. On définit :

$$F(x) = \Sigma_{0 \leq i \leq x}(f(i) + i^2)$$

Démonstration (1)

- Soit f une fonction calculable. On définit :

$$F(x) = \Sigma_{0 \leq i \leq x}(f(i) + i^2)$$

- Comme f est calculable alors F aussi. Donc, par la filière de Church, il existe une machine M_F qui lançant ses calculs sur une bande où sont inscrits x symboles | s'arrête avec un bloc de $F(x)$ symboles | sur la bande.
  Supposons que M_F ait n états.

Démonstration (1)

- Soit f une fonction calculable. On définit :

$$F(x) = \Sigma_{0 \leq i \leq x}(f(i) + i^2)$$

- Comme f est calculable alors F aussi. Donc, par la filière de Church, il existe une machine M_F qui lançant ses calculs sur une bande où sont inscrits x symboles $|$ s'arrête avec un bloc de $F(x)$ symboles $|$ sur la bande.
  Supposons que M_F ait n états.
- Considérons une nouvelle machine M qui commence sur une bande uniformément blanche. M commence par écrire x symboles $|$, puis revient en arrière et pointe sur le premier symbole $|$.
  $\Longrightarrow$ Cela peut se faire avec $x + 2$ états. Le programme est $(i, B, |, R, i + 1)$ pour $i \in \{1, \ldots, x\}$, ensuite il faut ajouter le quintuplet $x + 1, |, |, L, x + 1$ et enfin $x + 1, B, B, R, x + 2$.

Démonstration (2)

- Après avoir lancé M on se retrouve dans la configuration suivante :

$$(x+2)\overbrace{|\cdots|}^{x}$$

- Maintenant on peut lancer la machine M_F : formellement il suffit d'ajouter $x+2$ à tous les états des quintuplets du programme de M_F. L'état de la bande est alors un bloc de x symboles $|$ suivi d'un blanc, suivi d'un bloc de $F(x)$ symboles barres, avec la tête de lecture écriture qui pointe vers le premier symbole $|$ du second bloc.

$$\overbrace{|\cdots|}^{x}\, B(x+2+n)\overbrace{|\cdots|}^{F(x)}$$

Démonstration (3)

- On peut relancer une nouvelle version de M_F : on va donc ajouter $x + 2 + n$ à tous les états des quintuplets du programme de M_F. La configuration instantanée sera formée d'un nouveau bloc supplémentaire de $F(F(x))$ symboles | sur la bande à la droite des deux blocs qui s'y trouvaient déjà. Schématiquement on a :
  - Configuration instantanée de la machine M :

$$\overbrace{|\cdots|}^{x} B \overbrace{|\cdots|}^{F(x)} B(x+2+2n) \overbrace{|\cdots|}^{F(F(x))}$$

- On remarquera que tout ce qui a été faite est une machine qui participe à la compétition des castors affairés. Donc la construction montre qu'on a au moins :

$$\Sigma(x + 2 + 2n) \geq x + F(x) + F(F(x))$$

Démonstration (4)

Si nous reprenons la définition de la fonction F, il est clair que $F(x) \geq x^2$ (c'est le dernier terme de la somme donc même si f est la fonction nulle, on a au moins cette inégalité vérifiée).
Il existe donc une constante c_1 telle que pour tout x plus grand que c_1 on ait $x^2 > x + 2 + 2n$. Donc, $F(x) > x + 2 + 2n$ pour tout $x \geq c_1$. Il s'ensuit les inégalités suivantes :

$$\begin{aligned} \Sigma(x + 2 + 2n) &\geq x + F(x) + F(F(x)) \\ &> F(F(x)) \\ &> F(x + 2 + 2n) \\ &\geq f(x + 2 + 2n) \end{aligned}$$

pour tout x plus grand que c_1. On a donc bien Σ qui majore strictement f, comme on n'a fait aucune supposition sur f mis à part le fait qu'elle soit calculable, cela implique que Σ n'est pas une fonction calculable.

Décidabilité - différents niveaux de gris

- Pour l'instant nous avons vu une dichotomie : calculable/non calculable.
- Les choses sont plus complexes. Le problème de l'arrêt est en fait "semi-décidable" :
  - Si la machine s'arrête, un jour on le saura.
  - Sinon on pourra toujours attendre...un peu plus.
- Une classe étrange: quand la réponse est oui, un jour la machine s'arrête et répond, et quand la réponse est "non" la machine peut boucler à l'infini.
- Difficile à maîtriser car la double négation n'est plus équivalente à une affirmation.

Semi-décidabilité - définition

Definition

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.

- Intuitivement P est semi-décidable s'il existe une fonction T-calculable Ψ définie par:

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

- Tout prédicat décidable est aussi semi-décidable.

Notion d'ensemble récursivement énumérable

Definition

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

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

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

- un ensemble est Récursivement Enumérable signifie qu'il existe une manière d'énumérer ses éléments en calculant successivement $f(0), f(1), f(2), \ldots$ etc.
- Cette définition est la définition duale de celle d'un ensemble semi-décidable.
  - La définition d'un ensemble semi-décicable se fait par le domaine de la fonction.
  - Celle d'un ensemble récursivement énumérable se fait par l'image (codomaine) de la fonction

Théorème fondamentale

Théorème (Théorème de Post)

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.*

Théorème (Théorème fondamentale)

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

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

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

Théorème fondamentale - démonstration (1)

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

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.
- Donc $\Psi(x) = xsg(\mu n[(gx,n) = 1])$, 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.

Théorème fondamentale - démonstration (2)

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

.

- On va lancer successivement la machine φ_x sur toutes les entrées en parallèles, en alternant les calculs. Procédé en .
- 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.

Plan

Impossibilité pratique

- Le modèle des MT est simple et donne une bonne idée du nombre d'opérations à faire (différent du λ-calcul).
- Il existe différents aspects qui coûtent quand on éxécute un programme.
  - Temps de calcul.
  - Mémoire utilisée.
  - Taille du programme (Kolmogorov/Chaitin).
  - Profondeur logique de Bennett.

Classes de complexité

- L'idée de classe de complexité permet de mesurer un type de difficulté pour résoudre un type de problème.
- Le fait qu'une fonction soit calculable n'est pas pertinent si le temps de calcul est plus grand que l'âge de l'univers.
- Exemple simple : énumérer les entiers qu'on peut écrire avec n bits. On part de 0 et on veut énumérer su rla bande tous les nombres binaires jusqu'à n 1.

Ordre d'idée temporel

- Supposons qu'on puisse écrire 10^{15} mots binaires par seconde (vitesse d'horloge d'un ordinateur puissant de 2020).
- Supposons que Google ou la NSA puisse acheter 1000 ordinateurs et les assigner à cette tâche

Taille en bits	temps d'éxécution
56	Moins de 1 sec
64	18 sec
128	$1,07 \times 10^{13}$ ans
256	$3,65 \times 10^{51}$ ans
512	$4,25 \times 10^{128}$ ans

Pour information l'âge estimé de l'univers est $13,7 \times 10^{9}$ ans.

Principe de Landauer

- Quelle énergie pour faire le calcul ?
- Loi de la mécanique quantique, la plus petite quantité d'énergie pour changer un état physique est :

$$kTln(2)$$

avec k la constante de Boltzmann $k = 1.28 \times 10^{-23} J \cdot K^{-1}$..
- Soit $2.810^{-21} J$ au minimum pour la modification d'un bit.
- Energie contenue dans l'univers visible :
$E = 1.45 \times 10^{53}(3 \times 10^8)^2 = 4,35 \times 10^{69} J$
$\Longrightarrow$ On peut énumérer au plus $1.45/2.8 \times 10^{90}$ entiers écrits en binaires soit 5×10^{89} donc les entiers sur 207 bits environ (26 caractères ASCII).

Définition de la complexité temporelle

- Une définition très intuitive : combien de mouvement/écriture de la tête de lecture sur la bande.
- Equivalence avec le modèle de calcul "Random access machine".
  $\Longrightarrow$ Essentiellement les MT sont un modèles "réalistes" de machines.
- Il existe un lien direct avec la complexité spatiale :
  - On ne peut visiter plus de n cases quand la complexité temporelle est n.
  - Si on se limite à n cases le nombre de configurations possibles pour une MT est fixé: $n \times |\Sigma^n| \times |Q|$
- Etude asymptotique : multiplier deux nombres de deux chiffres n'est pas pareil que multiplier deux nombres de 10^{100} chiffres.
  $\Longrightarrow$ étude asymptotique.

Modèle et classe de complexité.

- Modèle de MT décidant l'appartenance à un langage (un état code "oui" et un autre code "non"), avec deux bandes une contenant le mot à reconnaître et l'autre pour les calculs intermédiaires.
  $\Longrightarrow$ toutes les définitions suivent mutatis mutandis.

Definition

Soit $L \subset \Sigma^*$, M une machine de Turing décidant l'appartenance à L. On dit que $L \in \mathbf{TIME}(f(n))$ si pour tout x un mot sur Σ de taille n, le nombre de transitions pour atteindre la configuration du calcul de M en partant de x sur la bande est inférieur à $f(n)$.

Théorème d'accélération linéaire

Théorème (Accélération linéaire)

Soit $L \in \mathbf{TIME}(f(n))$. *Alors pour tout réel* $\epsilon > 0$, *on a* $L \in \mathbf{TIME}(f'(n))$, *avec* $f'(n) = \epsilon f(n) + n + 2$.

- Théorème qui justifie l'habitude de raisonner en O.
- Fondation de la complexité algorithmique. i
- Architectures RISC/CISC dans les processeurs.

Démonstration du théorème d'accélération linéaire

- L'idée de la démonstration est de considérer un alphabet qui va regrouper m letteres sur une seule case et donc simuler m étapes de calcul en une seule.
- On pose $\Sigma' = \Sigma \cup \Sigma^m$.
- La machine lit la bande à reconnaitre par paquets de m et enregistre ça dans un état (m est fixé il y a un nombre fini de cas). Elle écrit la "super" lettre sur la deuxième bande.
- Une fois le mot codé la machine prend la deuxième bande comme bande d'entrée. La machine peut simuler m pas de calculs en 6 étapes. (au début il faut aller à gauche et à droite pour voir l'état des cases voisines).
- Le temps de calcul total de M' est $|x| + 2 + 6\lceil \frac{f(|x|)}{m} \rceil$. Il suffit de choisir $m = \lceil \frac{6}{\epsilon} \rceil$ nous donne le théorème.

Classes polynomiales et exponentielles

- **PTIME** : contient tous les problèmes de décisions qui peuvent être calculés en temps polynomial, i.e. $O(n^{O(1)})a$.
- **EXPTIME** : contient tous les problèmes de décisions qui peuvent être calculés en temps exponentiel, i.e. $O(2^{nO(1)})$.
- PTIME : Multiplication, addition, tri, minimisation d'automates, produits d'automates.
- EXPTIME : déterminisation d'automates,
- EXPTIME ? (exemples pour lesquels on a pas mieux qu'un algorithme exponentiel pour le moment) : factorisation, isomorphisme de graphes, problème du voyageur de commerce, satisfaction d'un ensemble de clauses.

Classes polynomiales et exponentielles

- Ne pas se concentrer sur le temps ou l'espace mais sur la taille du programme.
- Idée proche (en fait équivalente) de la notion d'entropie pour mesurer la quantité d'information dans une suite de chiffres.
- Permet de donner une défintion non paradoxale d'une suite aléatoire : $\Longrightarrow$ pour toute définition du "hasard" on aurait la plus petite suite aléatoire...qui ne serait plus aléatoire.
- Le paradoxe ne tient plus à cause du problème de l'arrêt (ou du résultat de Rice): il est en général indécidable de savoir si un programme est bien le plus petit pour produire une certaine suite de chiffres!

Utilisation de la complexité de Kolmogorov

Théorème

Pour une infinité d'entiers n le nombre de nombres premiers plus petit que n est au minimum de :

$$\frac{\log n}{\log(\log n)}$$

Proof.

Soient $p_1, p_2, \ldots, p_m$ la liste des nombres premiers inférieurs à n, on a :

$$n = p_1^{e_1} p_2^{e_2} \ldots p_m^{e_m}$$

Il suffit de connaitre la liste des e_i pour recalculer n.

Utilisation de la complexité de Kolmogorov

Théorème

Pour une infinité d'entiers n le nombre de nombres premiers plus petit que n est au minimum de :

$$\frac{\log n}{\log(\log n)}$$

Proof.

Soient $p_1, p_2, \ldots, p_m$ la liste des nombres premiers inférieurs à n, on a :

$$n = p_1^{e_1} p_2^{e_2} \ldots p_m^{e_m}$$

Il suffit de connaitre la liste des e_i pour recalculer n.
Chaque exposant est au plus de taille $\log n$, et donc chaque exposant peut être représenté en $\log(\log n)$ bits. Cette description de n peut donc être donnée en $m \log(\log n)$ bits.

□

Profondeur logique de Bennett

- La complexité de Kolmogorov ne représente pas bien la "complexité" au sens d'un objet perfectionné :
  Par définition c'est l'absence de structure, donc quelque chose de simple, qui a la plus grande complexité de Kolmogorov.
- Proposition de Bennett :
  Calculer la complexité algorithmique à partir du plus petit programme permettant de produire un résultat.
- Du point de vue Bennett une suite aléatoire est de complexité nulle : le plus petit programme consiste juste à imprimer le résultat voulu ! Même chose pour la complexité d'un écran blanc (donc pas du tout aléatoire).

Utilisation de la profondeur logique de Bennett

- La complexité au sens de Bennett est une mesure objective : tout comme un objet à une masse etc. il a une profondeur de Bennett.

Utilisation de la profondeur logique de Bennett

- La complexité au sens de Bennett est une mesure objective : tout comme un objet à une masse etc. il a une profondeur de Bennett.
- Proposition (JP Delahaye) de baser une éthique sur la conservation et l'augmentation de la complexité de Bennett : une éthique objective pour par exemple.
  En reprenant l'idée que ce qui a une profondeur de Bennett a demandé beaucoup de ressources on retrouve des prescriptions éthiques traditionnelles.
  - Conservation de la vie.
  - Problèmes liés à l'environnement.
  - "Prix" d'oeuvre d'art ou d'objets (robots etc.)
  - Ouverture sur une manière non anthropocentrée de quantifier des impications morales.

Plan

RAPPEL - Semi-décidabilité - définition

Definition

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.

- Intuitivement P est semi-décidable s'il existe une fonction T-calculable ψ définie par:

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

- Tout prédicat décidable est aussi semi-décidable.

RAPPEL - Notion d'ensemble récursivement énumérable

Definition

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

$$A \text{ est RE} \iff A = \emptyset \text{ ou } \exists f \in T.Im(f) = A$$

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

- un ensemble est Récursivement Enumérable signifie qu'il existe une manière d'énumérer ses éléments en calculant successivement $f(0), f(1), f(2), \ldots$ etc.
- Définition duale de semi-décidable.
  - La définition d'un ensemble semi-décicable se fait par le domaine de la fonction.
  - Celle d'un ensemble récursivement énumérable se fait par l'image (codomaine) de la fonction.

Retour sur le théorème de Post

Théorème (Théorème de Post)

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

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

Preuve du théorème de Post (1)

1. Si $A = \emptyset$ alors par def A est RE.

Preuve du théorème de Post (1)

1. Si $A = \emptyset$ alors par def A est RE.
2. Si A fini alors $A = \{x_1, \ldots, x_n\}$, il suffit de poser :

$$f(k) = \begin{cases} x_k & \text{si } x \leq n \\ x_n & \text{sinon} \end{cases}$$

Preuve du théorème de Post (1)

1. Si $A = \emptyset$ alors par def A est RE.

2. Si A fini alors $A = \{x_1, \ldots, x_n\}$, il suffit de poser :

$$f(k) = \begin{cases} x_k & \text{si } x \leq n \\ x_n & \text{sinon} \end{cases}$$

3. Si A est infini, on définira f par :

$$\begin{cases} f(0) & = & \text{ppetit } y \text{ t.q. } \xi_A(y) = 1 \\ f(k+1) & = & \text{ppetit } y \text{ t.q. } \xi_A(y) = 1 \text{ et } y > f(k) \end{cases}$$

Preuve du théorème de Post (2)

- Condition nécessaire car si A est décidable $\overline{A}$ aussi car $\xi_{\overline{A}} = 1 - \xi_A$ or tout ensemble décidable est RE.

Preuve du théorème de Post (2)

- Condition nécessaire car si A est décidable $\overline{A}$ aussi car $\xi_{\overline{A}} = 1 - \xi_A$ or tout ensemble décidable est RE.
- Condition suffisante :
  1. Si $A = \emptyset$ ou $A = \mathbb{N}$ les fonctions caractéristiques sont des fonctions constantes donc T-décidables.

Preuve du théorème de Post (2)

- Condition nécessaire car si A est décidable $\overline{A}$ aussi car $\xi_{\overline{A}} = 1 - \xi_A$ or tout ensemble décidable est RE.
- Condition suffisante :
  1. Si $A = \emptyset$ ou $A = \mathbb{N}$ les fonctions caractéristiques sont des fonctions constantes donc T-décidables.
  2. Si $A, \overline{A}$ sont images de deux fonctions totatles f et g. On peut construire une fonction h telle que:

$$\begin{cases} h(2p) & = & f(p) \\ h(2p+1) & = & g(p) \end{cases}$$

Preuve du théorème de Post (2)

- Condition nécessaire car si A est décidable $\overline{A}$ aussi car $\xi_{\overline{A}} = 1 - \xi_A$ or tout ensemble décidable est RE.
- Condition suffisante :
  1. Si $A = \emptyset$ ou $A = \mathbb{N}$ les fonctions caractéristiques sont des fonctions constantes donc T-décidables.
  2. Si $A, \overline{A}$ sont images de deux fonctions totatles f et g. On peut construire une fonction h telle que:

$$\begin{cases} h(2p) & = & f(p) \\ h(2p+1) & = & g(p) \end{cases}$$

h énumère $f(0), g(0), f(1), g(1), \ldots, f(n), g(n)$, et $Im(h) = A \cup \overline{A} = \mathbb{N}$. Soit x donné la parité de son rang indique s'il est dans A ou non.

Retour sur le théorème fondamental

Théorème (Théorème fondamental)

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

Preuve du théorème fondamental - Sens $\Longrightarrow$

- Supposons que A est RE, on veut construire ψ telle que

$$\psi(x) = \begin{cases} x & \text{si } x \in Imf \\ \uparrow & \text{sinon} \end{cases}$$

Preuve du théorème fondamental - Sens $\Longrightarrow$

- Supposons que A est RE, on veut construire ψ telle que

$$\psi(x) = \begin{cases} x & \text{si } x \in Imf \\ \uparrow & \text{sinon} \end{cases}$$

- On commence par construire g définie par :

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

Preuve du théorème fondamental - Sens $\Longrightarrow$

- Supposons que A est RE, on veut construire ψ telle que

$$\psi(x) = \begin{cases} x & \text{si } x \in Imf \\ \uparrow & \text{sinon} \end{cases}$$

- On commence par construire g définie par :

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

alors $\psi(x) = x(\text{pptit } n \ t.q. \ g(x,n) = 1)$ et $Imf = Dom\psi$

Preuve du théorème fondamental - Sens $\Longleftarrow$

- Dans ce sens on cherche à énumérer $Dom\ \psi$, s'il est non vide, par une fonction totale f.

Preuve du théorème fondamental - Sens $\Longleftarrow$

- Dans ce sens on cherche à énumérer *Dom* ψ, s'il est non vide, par une fonction totale f.
- Pour cela on utilise la technique en "queue d'arronde ("dove tailing" ou encre) qui consiste à simuler 1 pas de calcul de $\psi(0)$, 2 pas de calcul de $\psi(0)$ puis un de $\psi(1)\ldots$.

Preuve du théorème fondamental - Sens $\Longleftarrow$

- Dans ce sens on cherche à énumérer *Dom* ψ, s'il est non vide, par une fonction totale f.
- Pour cela on utilise la technique en "queue d'arronde ("dove tailing" ou encre) qui consiste à simuler 1 pas de calcul de $\psi(0)$, 2 pas de calcul de $\psi(0)$ puis un de $\psi(1)\ldots$.
- A une certaine étape on peut rajouter à l'énumération (f) les entrés des calculs qui ont convergé.

Propriétés de stabilité

- On a les classes suivantes :
  - Ensembles décidables.
    Avec une hiérarchie : ensembles finis, reconnaissables par états-finis, ensembles algébriques (par automate à pile).
  - Ensembles indécidables.
    - semi-décidables/Récursivement énumérables.
    - les non RE.

Propriétés de stabilité

- On a les classes suivantes :
  - Ensembles décidables.
    Avec une hiérarchie : ensembles finis, reconnaissables par états-finis, ensembles algébriques (par automate à pile).
  - Ensembles indécidables.
    - semi-décidables/Récursivement énumérables.
    - les non RE.
- Considérons les opérations ensemblistes : Union, Intersection, complémentaire, différence propre...
- Comment se comportent les classes vis à vis de ces opérations ?

Exemple

- RE n'est pas toujours stable par rapport au complémentaire (cas 2 du théorème de Post).

Exemple

- RE n'est pas toujours stable par rapport au complémentaire (cas 2 du théorème de Post).
- Si A, B sont RE qu'en est il de $A \cap B$?

Exemple

- RE n'est pas toujours stable par rapport au complémentaire (cas 2 du théorème de Post).
- Si A, B sont RE qu'en est il de $A \cap B$?
- Si A, B sont RE qu'en est il de $A \cup B$?

Exemple

- RE n'est pas toujours stable par rapport au complémentaire (cas 2 du théorème de Post).
- Si A, B sont RE qu'en est il de $A \cap B$?
- Si A, B sont RE qu'en est il de $A \cup B$?
- Si A, B sont RE qu'en est il de $A \oplus B = \{x \mid (x \in A \& x \notin B) | (x \in B \& x \notin A)\}$?

Exemple

- RE n'est pas toujours stable par rapport au complémentaire (cas 2 du théorème de Post).
- Si A, B sont RE qu'en est il de $A \cap B$?
- Si A, B sont RE qu'en est il de $A \cup B$?
- Si A, B sont RE qu'en est il de $A \oplus B = \{x \mid (x \in A \& x \notin B) | (x \in B \& x \notin A)\}$?
- Si A est décidable est B est RE qu'en est-il de $A \cap B$?
- etc.

Plan

Certains problèmes sont plus indécidables que d'autres

- Les problèmes indécidables qui ne sont pas RE sont plus difficiles : $K vs \overline{K}$
- Il est possible d'avoir une hiérarchie qui a une correspondance étonnante dans les formules logiques.
  $\Longrightarrow$ Liens profonds entre informatique/calculabilité et logique.
- Machines avec oracles : calculabilité relative.
- Turing-réductibilité relative.

Machine de Turing à oracle

- Pour formaliser un processus de calcul inconnu on le représente comme un oracle.
- En plus du fonctionnement normal une machine à oracle peut poser une question à l'oracle qui lui répond.
- Un oracle A est un sous ensemble (infini) des entiers.
- On le formalise par des quadruplets de la forme $q_i S_j q_l q_k$.
- Un quadruplet de cette forme indique que dans l'état q_i si on pointe sur le symbole S_j la machine stoppe, on compte le nombre n de barres sur la bande.
  1. Si $n \in A$ alors la machine passe dans l'état q_i
  2. Si $n \notin A$ alors la machine passe dans l'état q_j
- Turing calculable relativement à A : T^A-calculable

Calculabilité relative

- On peut reprendre toutes les définitions vues mais d'une manière relative.
  - Un pas de calcul relativement à A
  - Un calcul relativement à A
  - Fonctions calculables relativement à A
  - ...

Calculabilité relative

- On peut reprendre toutes les définitions vues mais d'une manière relative.
  - Un pas de calcul relativement à A
  - Un calcul relativement à A
  - Fonctions calculables relativement à A
  - ...
- Si A est indécidable...il devient décidable dans les MT^A : Xi_A est implantable par le programme :

$$\{q_0|Bq_1, q_1Bq_yq_n\}$$

  Si la machine termine dans q_y on interprète "oui" sinon elle termine dans l'état q_n et on interprète par "non".
- Les MT normales sont des cas particuliers où l'oracle est $\emptyset$ ou $\mathbb{N}$!
- Mais il y a de nouveaux problèmes indécidables relativement à l'oracle !

Turing réductibilité relative

Definition

Soient A, B des ensembles. On dit que $A \leq_T B$ si la fonction caractéristique de A est B-calculable (calculable par une MT ayant accès à l'oracle B).
Si $A \leq_T B$ et $B \leq_T A$ on note $A \equiv_T B$.

Turing réductibilité relative

Definition

Soient A, B des ensembles. On dit que $A \leq_T B$ si la fonction caractéristique de A est B-calculable (calculable par une MT ayant accès à l'oracle B).
Si $A \leq_T B$ et $B \leq_T A$ on note $A \equiv_T B$.

Definition (Turing-jump)

Etant donné un ensemble A son "Turing-jump" est l'ensemble

$$A' = K^A = \{x \mid x \in W_x^A\}$$

W_x^A étant le domaine sur lequel la machine de Turing, relativement à A, de numéro x converge.
On note $A^0 = A$ et $A^n = (A^n)'$.

Par exemple $K = \{x \mid \varphi_x(x) \downarrow\} = \emptyset' = \emptyset^1$

Stabilités relatives

Proposition

1. *Un ensemble E est A-calculable ssi $E, \overline{E}$ sont A-récursivement énumérables.* bigskip
2. *Un ensemble E est A-récursivement énumérable si et seulement s'il est la projection d'une relation R, A-calculable :*

$$E = \{x \mid \exists y.R(x, y)\}$$

Liens entre problème de l'arrêt et logique

- La conjecture de Goldbach :

$$\forall n \in \mathbb{N}.$$
$$(pair(n) \wedge n \geq 4) \implies$$
$$\exists p, q \in \mathbb{N}.(Premier(p) \wedge Premier(q) \wedge p + q = n$$

Liens entre problème de l'arrêt et logique

- La conjecture de Goldbach :

$$\forall n \in \mathbb{N}.\\ (pair(n) \wedge n \geq 4) \implies\\ \exists p, q \in \mathbb{N}.(Premier(p) \wedge Premier(q) \wedge p + q = n$$

- Lien entre la logique et la terminaison de programmes.
- Toutes les formules logiques ne sont pas de cette forme $\forall x.\exists y.P(x, y)$.
- Qu'en est il de formules plus complexes ?
  $\implies$ Le lemme de l'étoile dans les langages réguliers

$$\exists n.\forall v.\exists xyz.\forall k.|v| \geq n \wedge xyz = v \ldots$$

Ensembles arithmétiques

Definition

- $\Pi_0 = \Sigma_0 = \Delta_0$ classe des ensembles décidables ($\emptyset$-décidables).
- Pour $n \geq 1$.

$$A \in \Sigma_n \iff \exists R : T - calculable.$$
$$A = \{x \mid \exists y_1 . \forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$

avec Q qui est soit $\forall$ soit $\exists$ (dépend de la parité de n).
Symétriquement :

$$A \in \Pi_n \iff \exists R : T - calculable.$$
$$A = \{x \mid \forall y_1 . \exists y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$

et l'interstection :

$$\Delta_n = \Sigma_n \cap \Pi_n$$

Ensembles arithmétiques

Definition

- $\Pi_0 = \Sigma_0 = \Delta_0$ classe des ensembles décidables ($\emptyset$-décidables).
- Pour $n \geq 1$.

$$A \in \Sigma_n \iff \exists R : T - calculable.$$
$$A = \{x \mid \exists y_1 . \forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$

avec Q qui est soit $\forall$ soit $\exists$ (dépend de la parité de n).
Symétriquement :

$$A \in \Pi_n \iff \exists R : T - calculable.$$
$$A = \{x \mid \forall y_1 . \exists y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$

et l'interstection :

$$\Delta_n = \Sigma_n \cap \Pi_n$$

Ensembles arithmétiques : Propriétés - 1

Pour $n \geq 1$, Σ_n et Π_n sont complémentaires.

Ensembles arithmétiques : Propriétés - 1

Pour $n \geq 1$, Σ_n et Π_n sont complémentaires.

Si A est dans Σ_n alors par définition il existe R totale calculable telle que :

$$A = \{x \mid \exists y_1 . \forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$

Ensembles arithmétiques : Propriétés - 1

Pour $n \geq 1$, Σ_n et Π_n sont complémentaires.

Si A est dans Σ_n alors par définition il existe R totale calculable telle que :

$$A = \{x \mid \exists y_1 . \forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$

En appliquant la négation sur les quantifieurs $\forall x.F = \neg\exists x.\neg F$ on obtient :

$$\overline{A} = \{x \mid \forall y_1 . \exists y_2 \ldots . Q y_n . \overline{R}(x, y_1, \ldots, y_n)\}$$

Où $\overline{R}$ est la négation de R, soit essentiellement $1 - R$.

Ensembles arithmétiques : Propriétés - 2

La hiérarchie est cumulative : $\Sigma_n \cup \Pi_n \subseteq \Delta_m$ pour $m > n$.

Ensembles arithmétiques : Propriétés - 2

La hiérarchie est cumulative : $\Sigma_n \cup \Pi_n \subseteq \Delta_m$ pour $m > n$.

Il suffit d'ajouter des quantificateurs qui ne servent à rien ! Si $A \in \Sigma_n$ alors :

$$A = \{x \mid \exists y_1 . \forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$

Pour une certaine relation R, soit $m > n$. Donc

$$\begin{aligned} A &= \{x \mid \exists y_1 . \forall y_2 \ldots . Q y_m . R'(x, y_1, \ldots, y_m)\} \\ &= \{x \mid \forall y_0 . \exists y_1 . \forall y_2 \ldots . Q y_m . R'(x, y_1, \ldots, y_{m-1}, y_0)\} \end{aligned}$$

avec $R' = \{(x, y_1, \ldots, y_m) \mid R(x, y_1, \ldots, y_n)\}$.
La première ligne montre A dans Σ_m, et la seconde montre A dans Π_m donc $\Sigma_n \subseteq \Sigma_m \cap \Pi_m$.

Ensembles arithmétiques : Propriétés - 3

Chaque classe dans la hiérarchie est close par intersection et union ($A, B \in \Sigma_n$ implique $A \cup B \in \Sigma_n$).

Ensembles arithmétiques : Propriétés - 3

Chaque classe dans la hiérarchie est close par intersection et union ($A, B \in \Sigma_n$ implique $A \cup B \in \Sigma_n$).

Soit A, B dans Σ_n donc :

$$A = \{x \mid \exists y_1 . \forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$
$$B = \{x \mid \exists z_1 . \forall z_2 \ldots . Q z_n . S(x, z_1, \ldots, z_n)\}$$

Ensembles arithmétiques : Propriétés - 3

Chaque classe dans la hiérarchie est close par intersection et union ($A, B \in \Sigma_n$ implique $A \cup B \in \Sigma_n$).

Soit A, B dans Σ_n donc :

$$A = \{x \mid \exists y_1.\forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$
$$B = \{x \mid \exists z_1.\forall z_2 \ldots . Q z_n . S(x, z_1, \ldots, z_n)\}$$

On a :

$$\begin{aligned} A \cup B \quad = \quad \{x \mid & (\exists y_1.\forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)) \vee \\ & (\exists z_1.\forall z_2 \ldots . Q z_n . S(x, z_1, \ldots, z_n))\} \end{aligned}$$

Ensembles arithmétiques : Propriétés - 3

Chaque classe dans la hiérarchie est close par intersection et union ($A, B \in \Sigma_n$ implique $A \cup B \in \Sigma_n$).

Soit A, B dans Σ_n donc :

$$A = \{x \mid \exists y_1.\forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$
$$B = \{x \mid \exists z_1.\forall z_2 \ldots . Q z_n . S(x, z_1, \ldots, z_n)\}$$

On a :

$$\begin{aligned} A \cup B &= \{x \mid (\exists y_1.\forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)) \vee \\ &\qquad (\exists z_1.\forall z_2 \ldots . Q z_n . S(x, z_1, \ldots, z_n))\} \\ &= \{x \mid (\exists y_1.\exists z1.\forall y_2 \forall z_2 \ldots . Q y_n . Q z_n . \\ &\qquad (R(x, y_1, \ldots, y_n) \vee S(x, z_1, \ldots, z_n)\end{aligned}$$

Ensembles arithmétiques : Propriétés - 3

Chaque classe dans la hiérarchie est close par intersection et union ($A, B \in \Sigma_n$ implique $A \cup B \in \Sigma_n$).

Soit A, B dans Σ_n donc :

$$A = \{x \mid \exists y_1.\forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)\}$$
$$B = \{x \mid \exists z_1.\forall z_2 \ldots . Q z_n . S(x, z_1, \ldots, z_n)\}$$

On a :

$$\begin{aligned} A \cup B &= \{x \mid (\exists y_1.\forall y_2 \ldots . Q y_n . R(x, y_1, \ldots, y_n)) \vee \\ &\qquad (\exists z_1.\forall z_2 \ldots . Q z_n . S(x, z_1, \ldots, z_n))\} \\ &= \{x \mid (\exists y_1.\exists z1.\forall y_2 \forall z_2 \ldots . Q y_n . Q z_n . \\ &\qquad (R(x, y_1, \ldots, y_n) \vee S(x, z_1, \ldots, z_n)) \\ &= \{x \mid (\exists u_1.\forall u_2 \ldots . Q u_n . T(x, u_1, \ldots, u_n)) \end{aligned}$$

Ensembles arithmétiques : Propriétés - 4

Les classes Σ_n et Π_n sont respectivement closes par quantifications existentielles et universelles.

C'est à dire que $R \in \Sigma_n$ alors $\{x \mid \exists y.R(x,y)\} \in \Sigma_n$.

Ensembles arithmétiques : Propriétés - 4

Les classes Σ_n et Π_n sont respectivement closes par quantifications existentielles et universelles.

C'est à dire que $R \in \Sigma_n$ alors $\{x \mid \exists y.R(x,y)\} \in \Sigma_n$.

C'est une conséquence directe du codage des n-uplets dans les entiers. On peut en effet voir que

$$\exists y \exists y.R(x,y)$$

est équivalent à

$$\exists \langle x,y \rangle.R(\pi_1(\langle x,y \rangle), \pi_2(\langle x,y \rangle)$$

avec $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$
et $\pi_i : \mathbb{N} \to \mathbb{N}$ telles que

$$\pi_i(\langle x_1, x_2 \rangle) = x_i$$

Ensembles arithmétiques : Propriétés - 5(1)

Les classes de la hiérarchie sont colses par quantifications (existentielles et universelles) bornées.

Ensembles arithmétiques : Propriétés - 5(1)

Les classes de la hiérarchie sont colses par quantifications (existentielles et universelles) bornées.

Par exemple si $R \in \Sigma_n$ alors $\{\langle x, y\rangle \mid \forall y < z.R(x, y, z)\} \in \Sigma_n$

Ensembles arithmétiques : Propriétés - 5(1)

Les classes de la hiérarchie sont colses par quantifications (existentielles et universelles) bornées.

Par exemple si $R \in \Sigma_n$ alors $\{\langle x, y\rangle \mid \forall y < z.R(x, y, z)\} \in \Sigma_n$

Il suffit de donner la preuve pour Σ_n (on a l'autre par complémentation).
La preuve se fait par récurrence (le cas $n = 0$ étant trivial : il s'agit juste de tester un nombre fini de cas).
Soit $R \in \Sigma_n$ tel que

$$\begin{aligned} R(x, y, z) &\iff \exists u_1 \forall u_2 \ldots Qu_n.R'(\langle x, y, z\rangle, u_1, \ldots, u_n) \\ &\iff \exists u_1.S(x, y, z) \end{aligned}$$

$S = \{\langle x, y, z, u\rangle \ mid \forall u_2 \exists u_3 \ldots Qu_n R'(\langle x, y, z\rangle, u_2, \ldots, u_n)\} \in \Pi_{n-1}$

Ensembles arithmétiques : Propriétés - 5(2)

Mais alors :

$$\begin{aligned} \{\langle x, y\rangle \mid \forall z < y.R(x, y, z)\}i &= \{\langle x, y\rangle \mid \forall z < y.\exists u.S(x, y, z, u)\} \\ &= \{\langle x, y\rangle \mid \exists \sigma.S'(x, y, z, \sigma\} \end{aligned}$$

avec $S' = \{\langle x, y, z, \sigma\rangle \mid \forall z < y.S(x, y, z, (\sigma)_z\}$ avec σ qui est un encodage de u correspondant à chaque z strictement inférieur à y tel que $S(x, y, z, u)$ soit vrai.

Mais par hypothèse d'induction S' est dans Π_{n-1} donc on en conclut que :

$$\langle x, y\rangle \mid \forall z < y.R(x, y, z)$$

est dans Σ_n

Un exemple : Cof

- Comment qualifier un ensemble dans cette hiérarchie ?
- Par exemple :

$$Cof = \{x \mid \overline{W_x} \text{ est de taille finie}\}$$

- Borne supérieure assez simple :

$$x \in Cof \iff \exists y_1 . \forall y_2 (y_2 \leq y_1 \wedge y_2 \in W_x$$

Comme la partie de la formule non quantifiée est une disjonction d'expressions Σ_0 et Σ_1 on a $Cof \in \Sigma_3$.

- Il est possible que Cof soit plus bas dans la hiérarchie: comment le montrer ?

Notion d'ensemble complet

Definition

Un ensemble A est dit Σ_n-complet (resp. Π_n complet) si chaque ensemble dans Σ_n (resp. Π_n) est 1-réductible à A.

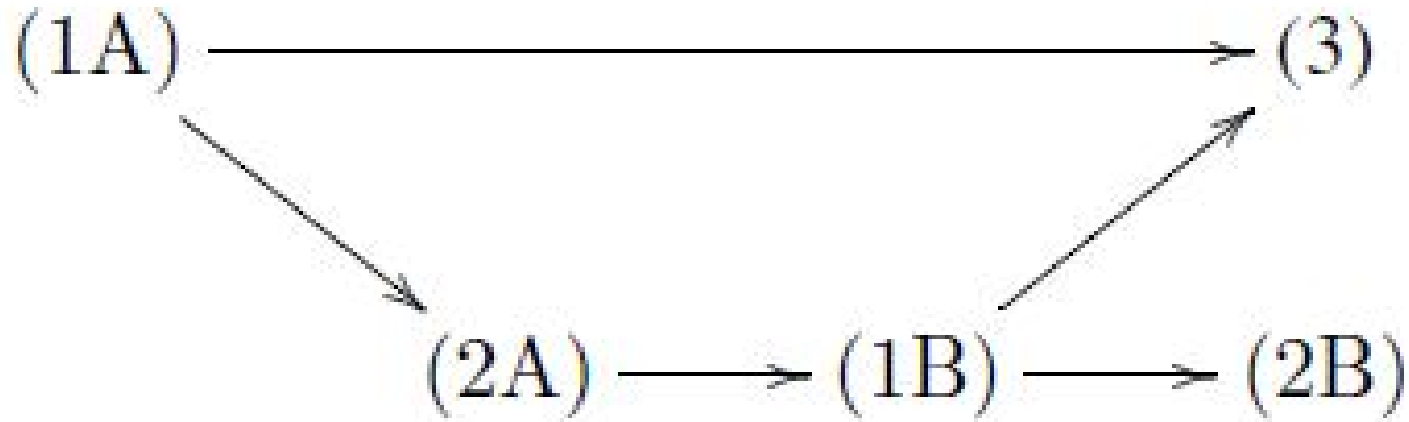
B est 1-réductible en A si, B se réduit en A par f et f est une bijection.

Thérème de Post : Hiérarchie Arithmétique

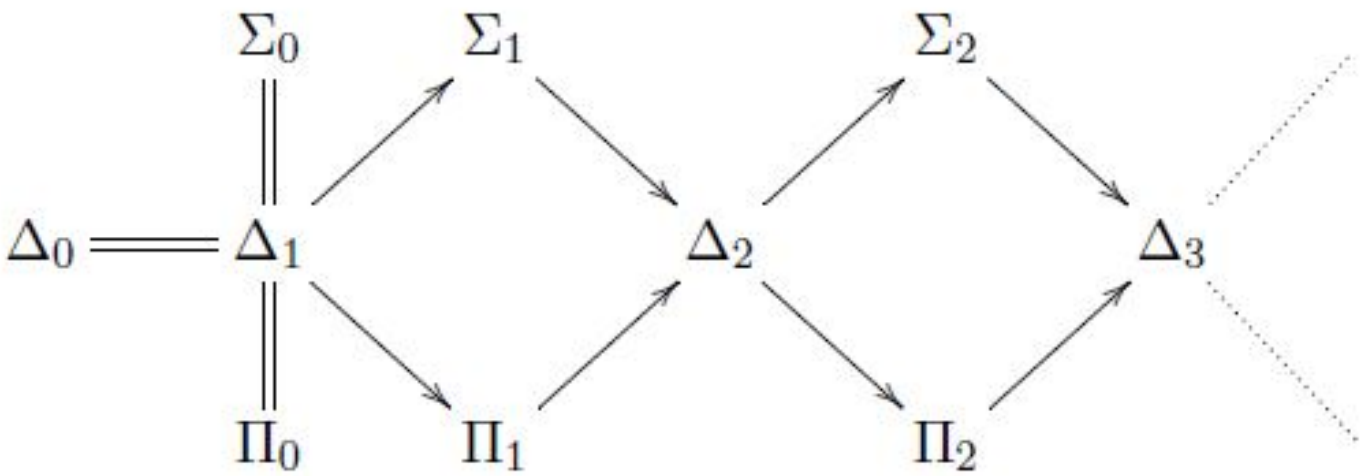
Théorème (Post)

1. *Un ensemble B est dans Σ_{n+1} si et seulement si B est récursivement énumérable en un ensemble Π_n (ou en un ensemble Σ_n).*
2. $\emptyset^{(n)}$ *est Σ_n-complet pour $n \geq 1$.*
3. *Un ensemble B est dans Δ_{n+1} si et seulement si B est décidable relativement à $\emptyset^{(n)}$.*

Schéma de preuve :



Hiérarchie arithmétique graphiquement.



Hiérarchie arithmétique signification

- Généralisation des fonctions calculables : calculabilité relative.

Hiérarchie arithmétique signification

- Généralisation des fonctions calculables : calculabilité relative.
- La hiérarchie arithmetique établit un lien formel etre la logique et la calculabilité effective (th. de Post).

Hiérarchie arithmétique signification

- Généralisation des fonctions calculables : calculabilité relative.
- La hiérarchie arithmetique établit un lien formel etre la logique et la calculabilité effective (th. de Post).
- Il existe des hiérarchies différentes :
  - Analytique : utilisation de l'ordre supérieur.
  - Polynomiale : en complexité à l'intérieur de ce qui est calculable.

Hiérarchie arithmétique signification

- Généralisation des fonctions calculables : calculabilité relative.
- La hiérarchie arithmetique établit un lien formel etre la logique et la calculabilité effective (th. de Post).
- Il existe des hiérarchies différentes :
  - Analytique : utilisation de l'ordre supérieur.
  - Polynomiale : en complexité à l'intérieur de ce qui est calculable.
- Course infinie contre la rélfexivitvé ? La conscience est elle Turing calculable ?
  - oui : il faut produire une théorie et un algorithme qui l'implante.
  - non : aucune théorie ne pourra le démontrer (sinon en implantant la théorie on aurait la solution)