

FDI 2

L3 – 2nd semestre de l'année 2008-2009

Natacha Portier

École Normale Supérieure de Lyon

Chapitre 1

Calculabilité

1.1 Algorithmes

- Un algorithme est exactement une recette de calcul.
- Enumérer les algorithmes.
- Un algorithme calcule une fonction.
- Une fonction est calculée par un ensemble compliqué d'algorithmes (th. de Rice).
- Une énumération des algos : f_n la fonction calculée par le n -ième algorithme ($f_n : \mathbb{N} \rightarrow \mathbb{N}$). Quid de $g : n \mapsto f_n(n) + 1$?
- \aleph_0 : cardinal du dénombrable alors que $|\{f : \mathbb{N} \rightarrow \mathbb{N}\}| = 2^{\aleph_0}$: cardinal de \mathbb{R} . En pratique on ne peut pas calculer $\{f : \mathbb{N} \rightarrow \mathbb{N}\}$ mais $\{f : \text{dom}(f) \subseteq \mathbb{N} \rightarrow \mathbb{N}\}$

1.1.1 Mais des fonctions de quoi dans quoi ?

- historique : $f : \mathbb{N} \rightarrow \mathbb{N}$.
- moderne : $f : \Sigma^* \rightarrow \Sigma^*$ où Σ est un alphabet fini.

Passage de l'un à l'autre Si $\Sigma = \{1\}$, $f : n \mapsto \underbrace{1 \dots 1}_n$.

Si $\Sigma = \{0, 1\}$, $f : n \rightarrow \bar{n}^2$ n'est *pas* une bijection¹ ! On pourra par contre prendre $f^{-1} : u \mapsto 1.\bar{u}^2 - 1$.

Exemple :

- $\epsilon \mapsto 1 - 1 \rightsquigarrow 0$
- $0 \mapsto 10 - 1 \rightsquigarrow 1$
- $1 \mapsto 11 - 1 \rightsquigarrow 2$
- $01 \mapsto 101 - 1 \rightsquigarrow 4$

1.1.2 Des fonctions non calculables existent ?

$$|\{f : \mathbb{N} \rightarrow \mathbb{N}\}| \geq |]0, 1[| > |\mathbb{N}|$$

Programme de Hilbert (Paris 1900) Dixième problème : résoudre les équations diophantiennes². Exemple d'équation diophantienne : $3x_1x_2^2 - 6x_4^3 = 5$.

Cela revient à donner un algorithme qui permet de savoir si une équation diophantienne admet une solution.

- 1 variable \rightsquigarrow 1 polynôme. On sait faire.
- Plusieurs variables \rightsquigarrow pas de méthode générale (non calculable à partir d'un nombre suffisant de variables³).

¹Quelle est l'antécédent de 01 par exemple ?

²Équations polynomiales à plusieurs inconnues à résoudre dans \mathbb{N} .

³Pour information il existe une preuve utilisant un peu plus de 26 variables.

1.1.3 Les définitions des algorithmes

Machines de Turing RAM Fonctions récursives Algorithmes de Markov	$\left. \vphantom{\begin{array}{l} \text{Machines de Turing} \\ \text{RAM} \\ \text{Fonctions récursives} \\ \text{Algorithmes de Markov} \end{array}} \right\} \begin{array}{l} 1930 : \text{Thèse de Church.} \\ \text{C'est bien cela la définition des fonctions calculables.} \\ \text{Toutes les définitions seront équivalentes.} \end{array}$
--	---

1.2 Machines de Turing

Définition :

Une **machine de Turing** est un 7-uplet $M = (K, \Sigma, B, Q, q_0, Q_f, \delta)$ où :

- $K \in \mathbb{N}^*$ est le nombre de ruban(s) de la machine.
- Σ est un alphabet fini. On notera $\Sigma_B = \Sigma \cup \{B\}$.
- Q un ensemble fini (d'états) qui contient q_0 (l'état initial) et Q_f (les états finaux).
- $\delta : Q \times \Sigma_B^K \rightarrow Q \times \Sigma_B^K \times \{-1, 0, 1\}^K$ est appelée fonction de transition.

Exemple :

$\begin{aligned} K &= 1 \\ \Sigma &= \{0, 1\} \\ Q &= \{q_0, q_1, q_2\} \ \& \ Q_f = \{q_2\} \\ \delta(q_0, 0) &= (q_0, 0, +1) \\ \delta(q_0, 1) &= (q_0, 1, +1) \\ \delta(q_0, B) &= (q_1, 0, -1) \\ \delta(q_1, 0) &= (q_1, 0, -1) \\ \delta(q_1, 1) &= (q_1, 1, -1) \\ \delta(q_1, B) &= (q_2, B, +1) \end{aligned}$	$\left. \vphantom{\begin{aligned} K &= 1 \\ \Sigma &= \{0, 1\} \\ Q &= \{q_0, q_1, q_2\} \ \& \ Q_f = \{q_2\} \\ \delta(q_0, 0) &= (q_0, 0, +1) \\ \delta(q_0, 1) &= (q_0, 1, +1) \\ \delta(q_0, B) &= (q_1, 0, -1) \\ \delta(q_1, 0) &= (q_1, 0, -1) \\ \delta(q_1, 1) &= (q_1, 1, -1) \\ \delta(q_1, B) &= (q_2, B, +1) \end{aligned}} \right\} \begin{array}{l} \text{On multiplie le nombre écrit en binaire par 2.} \\ \text{On se replace au début.} \end{array}$
---	---

Exemple :

Avec 2 rubans.

$\begin{aligned} K &= 2 \\ \Sigma &= \{0, 1\} \\ Q &= \{q_0, q_1\} \ \& \ Q_f = \{q_1\} \\ \delta(q_0, 0, 0) &= \delta(q_0, 1, 1) = (q_0, 0, 0, +1, +1) \\ \delta(q_0, 1, 0) &= \delta(q_0, 0, 1) = (q_0, 1, 1, +1, +1) \\ \delta(q_0, B, B) &= (q_1, B, B, 0, 0) \end{aligned}$	$\left. \vphantom{\begin{aligned} K &= 2 \\ \Sigma &= \{0, 1\} \\ Q &= \{q_0, q_1\} \ \& \ Q_f = \{q_1\} \\ \delta(q_0, 0, 0) &= \delta(q_0, 1, 1) = (q_0, 0, 0, +1, +1) \\ \delta(q_0, 1, 0) &= \delta(q_0, 0, 1) = (q_0, 1, 1, +1, +1) \\ \delta(q_0, B, B) &= (q_1, B, B, 0, 0) \end{aligned}} \right\} \begin{array}{l} \text{Calcule le XOR des deux rubans.} \\ \text{Le recopie sur les deux.} \end{array}$
--	--

Définition :

Une **configuration** d'une machine de Turing est constituée du contenu des rubans (les mots), des positions des têtes de lecture / écriture et de l'état courant. Donc c'est éventuellement infini.

Une configuration finie est une configuration telle que sur chaque ruban toutes les cases (sauf un nombre fini) contiennent B . On dit alors que le ruban contient un mot fini.

Si $\Sigma = \{a_1, \dots, a_\rho\}$, on code une configuration finie par :

$$(q, a_{i_1} \dots a_{j_1-1} \$ a_{j_1} \dots a_{i'_1}, a_{i_2} \dots a_{j_2-1} \$ a_{j_2} \dots a_{i'_2}, \dots, a_{i_K} \dots a_{j_K-1} \$ a_{j_K} \dots a_{i'_K})$$

C'est un mot sur l'alphabet $\{(\cdot, \cdot); \$\} \cup Q \cup \Sigma_B$.

On notera C l'ensemble des configurations finies (i.e. l'ensemble des mots correspondants).

Définition :

La **fonction globale de transition** $\Delta : C \rightarrow C$, $c \mapsto \Delta(c)$ est définie de la manière suivante :

⁴Les fonctions programmables sont exactement les fonctions calculables par des algorithmes.

Nom de l'instruction	Instance	Sémantique
(1 _j)	$X : \text{add}_j Y$	$0 \leq j \leq n-1, Y \mapsto Y.a_j$
(2)	$X : \text{dell } Y$	Efface la lettre de gauche du registre Y
(3)	$X : \text{clr } Y$	Efface le contenu du registre Y
(4)	$X : Y \leftarrow Z$	Copie le contenu de Z dans Y . Z reste inchangé.
(5)	$X : \text{jump } X'$	X' un entier (numéro de ligne à laquelle on saute)
(6 _j)	$X : Y \text{ jump}_j X'$	Si $Y = a_j.Y'$, sauter à X'
(7)	$X : \text{stop}$	S'arrêter

TAB. 1.1: Instructions d'une RAM

Si $\delta(q, a_{j_1}, \dots, a_{j_K}) = (q', b_1, \dots, b_K, \epsilon_1, \dots, \epsilon_K)$ et $c = (q, a_{j_1}, \dots, a_{j_K})$ alors

$$\Delta(c) = \begin{cases} (q, \dots, a_{i_l} \dots \$a_{j_l-1} b_l \dots a_{i'_l}, \dots) & \text{si } \epsilon_l = -1 \\ (q, \dots, a_{i_l} \dots a_{j_l-1} \$b_l \dots a_{i'_l}, \dots) & \text{si } \epsilon_l = 0 \\ (q, \dots, a_{i_l} \dots a_{j_l-1} b_l \$ \dots a_{i'_l}, \dots) & \text{si } \epsilon_l = 1 \end{cases}$$

Remarque :

Si $c' = \Delta^n(c)$, on note $c \rightarrow^n c'$.

Définition :

c **conduit** à c' s'il existe $n \in \mathbb{N}$ tel que $c \rightarrow^n c'$.

Définition :

Soient $M = (1, \Sigma, B, Q, q_0, Q_f, \delta)$ et $\Gamma \subset \Sigma$. On dit que M calcule la fonction $f : \Gamma^* \rightarrow \Gamma^*$ si pour tout mot $u = u_1 \dots u_n$, $u \in \text{dom}(f) \Leftrightarrow$ il existe $q' \in Q_f$ tel que $(q_0, \$u_1 \dots u_n) \rightarrow (q', \$v_1 \dots v_n)$ et dans ce cas $v_1 \dots v_n = f(u_1 \dots u_n)$

Remarque :

La définition est **robuste** : toute fonction calculable avec une machine à K rubans, l'est également avec une machine à 1 ruban.

1.3 Les RAM (Random⁵ Access Memory)

- John von Neumann
- Machine de von Neumann : Mettre le programme en données.
Ici une machine correspond à un programme.

Machine de Turing \rightarrow Machine universelle
RAM \rightarrow RASP

Définition :

On fixe un alphabet $\Sigma = \{a_0, a_1, \dots, a_{n-1}\}$ et un ensemble dénombrable de registres $\{R_0, R_1, \dots\}$. Une machine (un programme de RAM) est une suite finie d'instructions (cf. tableau ??), numérotées dans l'ordre, la dernière étant un **stop**.

Remarque :

La définition est **robuste**. En particulier : il existe quelquefois un registre spécial appelé **accumulateur** qui permet de faire les tests.

Définition :

Une **configuration** d'une RAM est la donnée :

⁵Random se traduit ici par arbitraire et non aléatoire. Ce sont des machines à "accès par adresse".

- d'un numéro de ligne (à exécuter)
- du contenu de tous les registres (ensemble dénombrable de mots infinis)

Définition :

Une **configuration** d'une RAM est **finie** s'il n'existe qu'un nombre fini de registres non vides, chacun contenant un mot fini.

Définition :

Un **pas de calcul** est le passage d'une configuration finie à une autre en exécutant une instruction.

Définition :

Un **calcul** est une suite de pas de calcul. Une fonction $f : \text{dom}(f) \subseteq \Sigma^* \rightarrow \Sigma^*$ est calculée par une RAM \mathfrak{R} si pour tout $u \in \Sigma^*$, si on effectue un calcul en partant de la configuration initiale (ligne 1, u dans R_0 , rien dans les autres), alors :

1. $u \in \text{dom}(f)$ ssi le calcul arrive à une instruction **stop**;
2. si $u \in \text{dom}(f)$, alors à la fin du calcul, tous les registres sont vides sauf R_1 qui contient $f(u)$.

Définition :

On dit qu'une fonction f est **calculable** par une RAM s'il existe une RAM \mathfrak{R} telle que f est calculée par \mathfrak{R} .

Théorème :

Une fonction f est calculable par machine de Turing ssi elle est calculable par une RAM.

DÉMONSTRATION :

1. \Leftarrow

Soit une fonction calculée par la RAM \mathfrak{R} . On va simuler le calcul de \mathfrak{R} par le calcul d'une machine de Turing M à délimiteurs.

- K = nombre de registres nommés dans la RAM
- $Q = \{X^1, X^2, \dots\}$ où X est un numéro de ligne.
- $\Sigma(M) = \Sigma(\mathfrak{R}) \cup \{G, D\}$ ⁶
- δ est définie de la manière suivante :
 - X : $\text{add}_j R_i$ donne (où la lettre indiquée est la $i^{\text{ème}}$) :
 - $\delta(X^1, \dots \alpha \dots) = (X^1, \dots, 0 \dots 010 \dots 0)$ où $\alpha \neq D$
 - $\delta(X^1, \dots D \dots) = (X^2, \dots a_j \dots, 0 \dots 010 \dots 0)$
 - $\delta(X^2, \dots B \dots) = ((X + 1)^1, \dots D \dots, 0 \dots 0(-1)0 \dots 0)$
 - X : $\text{dell } R_i$ donne :
 - $\delta(X^1, \dots G \dots) = (X^2, \dots B \dots, 0 \dots 010 \dots 0)$
 - $\delta(X^1, \dots \alpha \dots) = (X^1, \dots, 0 \dots 0(-1)0 \dots 0)$ où $\alpha \neq D$
 - $\delta(X^2, \dots D \dots) = (X^3, \dots, 0 \dots 0(-1)0 \dots 0)$
 - $\delta(X^2, \dots \alpha \dots) = ((X + 1)^1, \dots G \dots, 0 \dots 0)$ où $\alpha \neq D$
 - $\delta(X^3, \dots) = ((X + 1)^1, \dots G \dots, 0 \dots 010 \dots 0)$
 - ...⁷

2. \Rightarrow

Soit f calculée par une machine de Turing sans délimiteurs à 1 ruban. Le programme de la RAM qu'on lui associe est alors écrit sur le même alphabet. À chaque transition $\delta(q, a_i)$, on va associer une suite d'instructions situées aux lignes $X_{q, a_i, 1}, \dots, X_{q, a_i, k_{q, a_i}}$. De plus, le registre R_1 contient toute la partie du ruban qui est à gauche de la tête de lecture, tandis que R_2 contient la partie à droite de la tête de lecture.

- $\delta(q, a_i) = (q', a_j, +1)$

$$X_{q, a_i} \left\{ \begin{array}{l} \text{dell } R_2 \\ \text{add}_j R_1 \\ R_2 \text{ jump}_1 X_{q', a_1} \\ \vdots \\ R_2 \text{ jump}_m X_{q', a_m} \end{array} \right.$$

⁶Donc $\Sigma_B(M) = \Sigma(\mathfrak{R}) \cup \{B, G, D\}$ avec les notations introduites précédemment.

⁷Suite laissée au lecteur.

Briques de base	Historique	Moderne
fonction nulle	$0^n : (x_1, \dots, x_n) \mapsto 0$	$0^n : x \in (A_k^*)^n \mapsto \epsilon$
fonction successeur	$s : x \mapsto x + 1$	$s_i : x \mapsto x.a_i$
i ème projecteur $1 \leq i \leq n$	$p_i^n : (x_1, \dots, x_n) \mapsto x_i$	$p_i^n : (x_1, \dots, x_n) \mapsto x_i$

TAB. 1.2: Briques de base des fonctions récursives

– $\delta(q, a_i) = (q', a_j, 0)$ ⁸

X_{q,a_i} $\left[\begin{array}{l} \text{Effacer } a_i, \text{ faire le miroir de } R_2, \text{ coller } a_j, \text{ refaire le miroir.} \\ \text{Aller à } X_{q',a_j} \text{ si } \delta(q', a_j) \text{ existe.} \end{array} \right.$

– $\delta(q, a_i) = (q', a_j, -1)$

X_{q,a_i} $\left[\begin{array}{l} \text{Miroir de } R_1. \text{ Effacer une lettre. Miroir de } R_1. \text{ Effacer une lettre.} \\ \text{Miroir de } R_2. \text{ Coller } a_j \text{ et } a_{i-1}. \text{ Refaire le miroir.} \\ \text{Aller à } X_{q',a_{i-1}} \text{ si } \delta(q', a_{i-1}) \text{ existe.} \end{array} \right.$

□

1.4 Les fonctions récursives

1.4.1 Les fonctions récursives primitives

1. Des briques de base
2. Des opérations pour fabriquer de nouvelles fonctions

Définition :

La **clôture mathématique** est le plus petit sous-ensemble de fonctions qui contient les briques de base et est stable par les opérations.

Définition :

La **clôture** (Cl) en informatique se définit de la manière suivante :

$$Cl^0 = \{\text{briques de base}\}$$

$$Cl^{n+1} = \text{fonctions construites avec une opération sur des fonctions de } \bigcup_{0 \leq i \leq n} Cl^i$$

$$Cl = \bigcup_{n \in \mathbb{N}} Cl^n$$

Définition :

L'ensemble des **fonctions récursives primitives** est la clôture obtenues en utilisant les briques de base (cf. tableau ??) et les opérations (cf. tableau ??) décrites ci-dessous.

Il existe deux versions de chaque fonction : une version historique ($f : \mathbb{N}^n \rightarrow \mathbb{N}^*$) et une version moderne ($f : (A_k^*)^n \rightarrow A_k^*$ avec $A_k = \{a_1, \dots, a_k\}$).

Exemple :

L'addition (F) est récursive primitive. $F : \mathbb{N}^2 \rightarrow \mathbb{N}, (x, y) \mapsto x + y$ est en effet caractérisée par :

$$\begin{cases} F(0, y) = y \\ F(s(x), y) = s(F(x, y)) \end{cases}$$

D'où :

$$F = \text{Rec}^2(p_1^1, \text{Sub}_3^1(s, p_2^3))$$

⁸Exercice : l'écrire directement.

Opérations	Historique	Moderne
Substitution	$m, n \in \mathbb{N}$ $g : (\mathbb{N}^*)^m \rightarrow \mathbb{N}^*$ $f_1 : (\mathbb{N}^*)^n \rightarrow \mathbb{N}^*$ \vdots $f_m : (\mathbb{N}^*)^n \rightarrow \mathbb{N}^*$ $\text{Subst}_n^m(g, f_1, \dots, f_m) : (\mathbb{N}^*)^n \rightarrow \mathbb{N}^*$ $\bar{x} \mapsto g(f_1(\bar{x}), \dots, f_m(\bar{x}))$	$m, n \in \mathbb{N}$ $g : (A_k^*)^m \rightarrow A_k^*$ $f_1 : (A_k^*)^n \rightarrow A_k^*$ \vdots $f_m : (A_k^*)^n \rightarrow A_k^*$ $\text{Subst}_n^m(g, f_1, \dots, f_m) : (A_k^*)^n \rightarrow A_k^*$ $\bar{x} \mapsto g(f_1(\bar{x}), \dots, f_m(\bar{x}))$
Récursion primitive	$n \in \mathbb{N}$ $g : \mathbb{N}^{n-1} \rightarrow \mathbb{N}$ $f : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ $\text{Rec}^n(g, f) :$ $(0, x_2, \dots, x_n) \mapsto g(x_2, \dots, x_n)$ $(s(x'_1), x_2, \dots, x_n) \mapsto f(x'_1, \text{Rec}^n(g, f))$ $(x'_1, x_2, \dots, x_n), x_2, \dots, x_n$	$n \in \mathbb{N}$ $g : (A_k^*)^{n-1} \rightarrow A_k^*$ $f_1, \dots, f_k : (A_k^*)^{n+1} \rightarrow A_k^*$ $\text{Rec}^n(g, f_1, \dots, f_k) :$ $(\epsilon, x_2, \dots, x_n) \mapsto g(x_2, \dots, x_n)$ $(y, a_i, x_2, \dots, x_n) \mapsto f_i(y, \text{Rec}^n(g, f_1, \dots, f_k))$ $(y, x_2, \dots, x_n), x_2, \dots, x_n$

TAB. 1.3: Opérations utilisées pour définir les fonctions récursives

Remarque :

Toute fonction $f : (A_k^*)^m \rightarrow (A_k^*)^n$ est récursive primitive si et seulement si chacune de ses fonctions composantes est récursive primitive.

Exemple :

- Les fonctions constantes
- Les fonctions concaténation : $\text{con}_n : (x_1, \dots, x_n) \mapsto x_1 \dots x_n$
- La fonction puissance : $[x]^n : \underbrace{x \dots x}_n$
- La fonction lev : $\epsilon \mapsto \epsilon, x \neq \epsilon \mapsto a_1$
- La fonction lev' : $\epsilon \mapsto a_1, x \neq \epsilon \mapsto \epsilon$
 $\text{lev}' = \text{Rec}(s_1 0^0, 0^2, \dots, 0^2)$
- La fonction end_j :

$$x \mapsto \begin{cases} a_1 & \text{si } x \text{ finit par } a_j \\ \epsilon & \text{sinon} \end{cases}$$

- La fonction rev qui à tout x associe son miroir
- La fonction del_i :

$$x \mapsto \begin{cases} x' & \text{si } x = a_i \cdot x' \\ \epsilon & \text{sinon} \end{cases}$$

- ...

Remarque :

On a donné ici un ensemble non minimal de briques de base mais cela n'est pas gênant dans la mesure où tout ensemble de briques de base devra être infini.

Définition :

Un ensemble $E \subseteq (A_k^*)^n$ (respectivement \mathbb{N}^n) est récursif primitif si sa fonction caractéristique l'est.

$$\chi_E(x_1, \dots, x_n) = \begin{cases} \epsilon & \text{si } \exists i \in \llbracket 1, n \rrbracket, x_i \notin E & (\text{resp. } 0) \\ a_1 & \text{sinon} & (\text{resp. } 1) \end{cases}$$

Remarque :

Un prédicat est un ensemble ($\chi_E \leftrightarrow E$). Le prédicat est récursif primitif si l'ensemble correspondant l'est.

Proposition :

- Si E et F sont des ensembles récurrents primitifs de $(A_k^*)^n$ alors $E \cup F$, $E \cap F$ et \mathbb{C}_E sont récurrents primitifs.
- Si P et Q sont des prédicats de $(A_k^*)^n$ récurrents primitifs alors $P \vee Q$, $P \wedge Q$ et $\neg P$ sont récurrents primitifs.

DÉMONSTRATION :

- $\chi_{\mathbb{C}_E}$ en fonction de $\chi_E : \chi_{\mathbb{C}_E} = \text{lev}' \circ \chi_E$
 - $\chi_{E \cup F} = \text{Rec}^2(p_1^1, a_1)(\chi_E(x), \chi_F(x)) = \text{lev}(\text{con}_2(\chi_E(x), \chi_F(x)))$
 - $\chi_{E \cap F} = \text{dell}(\text{con}_2(\chi_E(x), \chi_F(x)))$
-

Proposition :

Si P_1, \dots, P_n sont des prédicats récurrents primitifs disjoints et f_1, \dots, f_{n+1} des fonctions récurrentes primitives alors la fonction suivante est récurrente primitive.

$$g(x_1, \dots, x_m) = \begin{cases} f_1(x_1, \dots, x_m) & \text{si } P_1(x_1, \dots, x_m) \\ \vdots \\ f_n(x_1, \dots, x_m) & \text{si } P_n(x_1, \dots, x_m) \\ f_{n+1}(x_1, \dots, x_m) & \text{sinon} \end{cases}$$

DÉMONSTRATION :

On définit la fonction \sharp (notée en infixe par la suite) :

$$\sharp : \begin{cases} (A_k^*)^2 & \rightarrow A_k^* \\ (x, y) & \mapsto \begin{cases} \epsilon & \text{si } x = \epsilon \\ y & \text{sinon} \end{cases} \end{cases}$$

\sharp est récurrente primitive. En effet : $\sharp = \text{Rec}^2(0^1, p_3^3, \dots, p_3^3)$.

$$F = \text{con}_{n+1}(P_1 \sharp f_1, \dots, P_n \sharp f_n, \mathbb{C}_{P_1 \cup \dots \cup P_n} \sharp f_{n+1})$$

□

Définition :

On définit la **quantification bornée** de la manière suivante : Soit P un prédicat $(n+1)$ -aire.

$\exists y|x, P(y, z_1, \dots, z_n) \Leftrightarrow \exists y$ segment initial de x tel que $P(y, z_1, \dots, z_n)$

$\forall y|x, P(y, z_1, \dots, z_n) \Leftrightarrow \forall y$ segment initial de x , on a $P(y, z_1, \dots, z_n)$

$\exists y|x, P(y, z_1, \dots, z_n) = \{(x, z_1, \dots, z_n) | \exists y \text{ segment initial de } x \text{ tel que } P(y, z_1, \dots, z_n)\} \subseteq (A_k^*)^{n+1}$

Pour les entiers : $y|x \Leftrightarrow y \leq x$

Proposition :

Si P est récurrente primitive alors $\exists y|x, P$ et $\forall y|x, P$ aussi.

DÉMONSTRATION :

$$\begin{cases} \chi_{\exists y|x, P}(\epsilon, z_1, \dots, z_n) = \chi_P(\epsilon, z_1, \dots, z_n) \\ \chi_{\exists y|x, P}(x, a_i, z_1, \dots, z_n) = \text{lev}(\text{con}_2(\chi_P(x, a_i, z_1, \dots, z_n), X_{\exists y|x, P}(x, z_1, \dots, z_n))) \end{cases}$$

$\forall y|x, P = \neg \exists y|x, (\neg P)$

□

1.4.2 Les fonctions récurrentes

Exemple :

La fonction d'Ackermann.

Définition :

$$A : \begin{cases} \mathbb{N}^2 & \rightarrow \mathbb{N} \\ (0, x) & \mapsto x + 1 \\ (p + 1, 0) & \mapsto A(p, 1) \\ (p + 1, x + 1) & \mapsto A(p, A(p + 1, x)) \end{cases}$$

Exemple :

- $A(1, 0) = A(0, 1) = 2$
- $A(2, 0) = A(1, 1) = A(0, A(1, 0)) = A(0, 2) = 3$
- $A(1, 1) = 3$
- $A(1, x) = A(0, A(1, x-1)) = A(1, x-1) + 1 = x + 2$
- $A(2, x) = 2x + 3$
- $A(3, x) = 8 \cdot 2^x - 3$
- $A(4, x) = 8 \cdot 2^{A(3, x-1)} - 3$

Proposition :

- A n'est pas récursive primitive.
- A est récursive.

DÉMONSTRATION :

Voir le TD. \square

Définition :

On ajoute un nouvel opérateur, le schéma μ , et on s'intéresse aux fonctions partielles.

- **Version moderne :**

Soit f une fonction partielle $(\text{dom } f) \subseteq (A_k^*)^{n+1} \rightarrow A_k^*$.

Alors $\text{Sch}_i \mu(f)(x_1, \dots, x_n)$ est défini si et seulement s'il existe m tel que :

1. $f(\epsilon, x_1, \dots, x_n), f(a_i, x_1, \dots, x_n), \dots, f(a_i^m, x_1, \dots, x_n)$ sont définis
2. $f(a_i^m, x_1, \dots, x_n) = \epsilon$

Dans ce cas, si m est le plus petit possible, on a $\text{Sch}_i \mu(f)(x_1, \dots, x_n) = a_i^m$

- **Version classique :**

Soit f une fonction partielle $(\text{dom } f) \subseteq \mathbb{N}^{n+1} \rightarrow \mathbb{N}$.

Alors $\text{Sch } \mu(f)(x_1, \dots, x_n)$ est défini si et seulement s'il existe $y \in \mathbb{N}$ tel que :

1. $f(0, x_1, \dots, x_n), f(1, x_1, \dots, x_n), \dots, f(y, x_1, \dots, x_n)$ sont définis
2. $f(y, x_1, \dots, x_n) = 0$

Dans ce cas, si y est le plus petit possible on a $\text{Sch } \mu(f)(x_1, \dots, x_n) = y$

On doit maintenant adapter les anciennes définitions pour tenir compte des fonctions partielles :

Définition :- **Version moderne :**

Soit $(m, n) \in \mathbb{N}^2$. Si :

- $g : (\text{dom } g) \subseteq (A_k^*)^m \rightarrow A_k^*$
- $f_1 : (\text{dom } f_1) \subseteq (A_k^*)^n \rightarrow A_k^*$

- ...

- $f_m : (\text{dom } f_m) \subseteq (A_k^*)^n \rightarrow A_k^*$

Alors $\text{Subst}_n^m(g, f_1, \dots, f_m)(x_1, \dots, x_n)$ est défini si et seulement si :

1. $f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)$ sont définis
2. $g(f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n))$ est défini

Dans ce cas :

$$\text{Subst}_n^m(g, f_1, \dots, f_m)(x_1, \dots, x_n) = g(f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n))$$

- **Version classique :**

Soit $(m, n) \in \mathbb{N}^2$. Si :

- $g : (\text{dom } g) \subseteq \mathbb{N}^m \rightarrow \mathbb{N}$
- $f_1 : (\text{dom } f_1) \subseteq \mathbb{N}^n \rightarrow \mathbb{N}$

- ...

- $f_m : (\text{dom } f_m) \subseteq \mathbb{N}^n \rightarrow \mathbb{N}$

Alors $\text{Subst}_n^m(g, f_1, \dots, f_m)(x_1, \dots, x_n)$ est défini si et seulement si :

1. $f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)$ sont définis
2. $g(f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n))$ est défini

Dans ce cas :

$$\text{Subst}_n^m(g, f_1, \dots, f_m)(x_1, \dots, x_n) = g(f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n))$$

Définition :

– **Version moderne :**

Soit $n \in \mathbb{N}$. Si :

- $g : (\text{dom } g) \subseteq (A_k^*)^{n-1} \rightarrow A_k^*$
- $f_1 : (\text{dom } f_1) \subseteq (A_k^*)^{n+1} \rightarrow A_k^*$
- ...
- $f_k : (\text{dom } f_k) \subseteq (A_k^*)^{n+1} \rightarrow A_k^*$

Alors $\text{Rec}^n(g, f_1, \dots, f_k)(\epsilon, x_2, \dots, x_n)$ est défini si et seulement si $g(x_2, \dots, x_n)$ est défini.

Dans ce cas :

$$\text{Rec}^n(g, f_1, \dots, f_k)(\epsilon, x_2, \dots, x_n) = g(x_2, \dots, x_n)$$

Et $\text{Rec}^n(g, f_1, \dots, f_k)(a_i y, x_2, \dots, x_n)$ est défini si et seulement si :

1. $\text{Rec}^n(g, f_1, \dots, f_k)(y, x_2, \dots, x_n)$ est défini
2. Si $z = \text{Rec}^n(g, f_1, \dots, f_k)(y, x_2, \dots, x_n)$, alors $f_i(y, z, x_2, \dots, x_n)$ est défini

Dans ce cas :

$$\text{Rec}^n(g, f_1, \dots, f_k)(a_i y, x_2, \dots, x_n) = f_i(y, \text{Rec}^n(g, f_1, \dots, f_k)(y, x_2, \dots, x_n), x_2, \dots, x_n)$$

– **Version classique :**

Soit $n \in \mathbb{N}$. Si :

- $g : (\text{dom } g) \subseteq \mathbb{N}^{n-1} \rightarrow \mathbb{N}$
- $f : (\text{dom } f) \subseteq \mathbb{N}^{n+1} \rightarrow \mathbb{N}$

Alors $\text{Rec}^n(g, f)(0, x_2, \dots, x_n)$ est défini si et seulement si $g(x_2, \dots, x_n)$ est défini.

Dans ce cas :

$$\text{Rec}^n(g, f)(0, x_2, \dots, x_n) = g(x_2, \dots, x_n)$$

Et $\text{Rec}^n(g, f)(y + 1, x_2, \dots, x_n)$ est défini si et seulement si :

1. $\text{Rec}^n(g, f)(y, x_2, \dots, x_n)$ est défini
2. Si $z = \text{Rec}^n(g, f)(y, x_2, \dots, x_n)$, alors $f(y, z, x_2, \dots, x_n)$ est défini

Dans ce cas :

$$\text{Rec}^n(g, f)(y + 1, x_2, \dots, x_n) = f(y, \text{Rec}^n(g, f)(y, x_2, \dots, x_n), x_2, \dots, x_n)$$

Définition :

L'ensemble des fonctions récursives est le plus petit ensemble contenant les fonctions constantes nulles et les projections, et stable par récursion primitive, substitution et schéma μ .

Une fonction récursive totale est une fonction récursive définie partout.

Remarque :

Deux fonctions égales ont le même domaine.

Exemple :

- $f : (\text{dom } f) \subsetneq \mathbb{N} \rightarrow \mathbb{N}$
- $f - f$ n'est pas la fonction nulle car elle n'est pas définie sur tout \mathbb{N} .

Théorème :

Les fonctions récursives sont calculables.

DÉMONSTRATION :

Idée : Par induction

- Les briques de base sont calculables.

- Si f, g_1, \dots, g_m sont calculables alors $\text{Subst}_m^n(f, g_1, \dots, g_m)$ l'est également. Il s'agit juste de composer deux machines de Turing.
- Si g, f_1, \dots, f_k sont calculables alors $\text{Rec}^n(g, f_1, \dots, f_k)$ l'est également. Il s'agit de construire une machine qui sur l'entrée $(y = y_1 \dots y_n, x_2, \dots, x_n)$:
 1. Calcule $g(x_2, \dots, x_n)$ ce qui est possible par induction car g est calculable
 2. Puis calcule $f(y_n, g(x_2, \dots, x_n), x_2, \dots, x_n)$ ce qui est possible par induction car f est calculable
 3. ...
 4. Enfin calcule $f(y_1, f(y_2, \dots, f(y_n, g(x_2, \dots, x_n), x_2, \dots, x_n) \dots, x_2, \dots, x_n), x_2, \dots, x_n)$
- Si f est calculable, $\text{Sch}_i \mu f(y)$ l'est également. Il s'agit de construire une machine qui sur l'entrée (x_1, \dots, x_n) :
 1. Calcule $f(\epsilon, x_1, \dots, x_n)$
 2. Calcule $f(a_i, x_1, \dots, x_n)$
 3. ...
 4. Calcule $f(a_i^m, x_1, \dots, x_n)$ jusqu'à ce que $f(a_i^m, x_1, \dots, x_n) = \epsilon$

□

Théorème :

Les fonctions calculables sont rékursives.

Remarque :

- **Ordre lexicographique :**

On définit $<_{\text{lex}}$ ainsi :

$$x_1 \dots x_n <_{\text{lex}} y_1 \dots y_m \Leftrightarrow \begin{cases} \exists i \in \llbracket 1, \min(n, m) \rrbracket, \text{ si } x_1 = y_1, \dots, x_{i-1} = y_{i-1} \text{ et } x_i < y_i \\ \text{ou} \\ n < m \text{ et } x_1 = y_1, \dots, x_n = y_n \end{cases}$$

- **Ordre hiérarchique :**

On définit \prec ainsi :

$$x_1 \dots x_n \prec y_1 \dots y_m \Leftrightarrow \begin{cases} n < m \text{ ou} \\ n = m \text{ et } x <_{\text{lex}} y \end{cases}$$

C'est un ordre total, un bon ordre (tout sous-ensemble a un plus petit élément) et il n'y a pas de chaînes infinies décroissantes.

- Les deux ordres coïncident sur $(A_k^*)^n$.

- **Codages :**

Il existe $\chi : \mathbb{N}^2 \rightarrow \mathbb{N}$ bijective et réursive primitive. On note $\langle x, y \rangle = \chi(x, y)$.

On note (π_1, π_2) les réciproques qui sont elles aussi rékursives primitives.

On pose

$$- K_2 = \chi$$

$$- K_{n+1}(x_1, \dots, x_{n+1}) = \langle x_1, K_n(x_2, \dots, x_{n+1}) \rangle.$$

$$- \xi(x_1 \dots x_n) = \langle n, K_n(x_1, \dots, x_n) \rangle$$

Alors pour tout $n \in \mathbb{N}$, K_n est une bijection de \mathbb{N}^n dans \mathbb{N} .

De plus, ξ est une bijection de \mathbb{N}^* dans \mathbb{N} .

Proposition :

La fonction d'Ackermann est réursive.

DÉMONSTRATION :

On veut calculer $A(p+1, x+1) = A(p, A(p+1, x))$.

$(p, A(p+1, x)) < (p+1, x+1)$ et $(p+1, x) < (p+1, x+1)$ donc la portion à calculer est finie.

On cherche le plus petit code d'un tableau rectangulaire à L lignes et C colonnes $y = K_3(L, C, K_{(L+1)(C+1)}(a_{00}, a_{01}, \dots, a_{0C}, a_{10}, \dots, a_{LC}))$ tel que :

1. La première ligne est bien remplie et la première colonne aussi
2. La case $(p'+1, x'+1)$ contient $a(p'+1, x'+1)$ tel que si $a(p'+1, x') \leq C$ et $a(p', a(p'+1, x')) \neq 0$ alors $a(p'+1, x'+1) = a(p', a(p'+1, x'))$.
3. $p+1 \leq L$, $x+1 \leq C$ et $a(p+1, x+1) \neq 0$

Toutes ces conditions sont rékursives primitives.

1. Pour tout $0 \leq j \leq C$: $a(0, j) = j + 1$ et $\pi_1 \pi_2^{2+j} y = j + 1$ car :

$$\begin{aligned}\pi_1 y &= L \\ \pi_1 \pi_2 y &= C \\ \pi_1 \pi_2^2 &= a_{00} \\ \pi_1 \pi_2^3 &= a_{01} \\ \pi_1 \pi_2^4 &= a_{02}\end{aligned}$$

Puis pour tout $0 \leq i \leq L$: $\pi_1 \pi_2^{2+i\pi_1\pi_2 y} y = \pi_1 \pi_2^{3+(i-1)\pi_1\pi_2 y} y$ car $A(i, 0) = A(i-1, 1)$

2. Pour tout $p' + 1 \leq L$ et $x' + 1 \leq C$ on a :
- soit $(a(p' + 1, x') > C$ et $a(p' + 1, x' + 1) = 0)$
 - soit $(a(p' + 1, x') \leq C$ et $a(p', a(p' + 1, x')) = 0$ et $a(p' + 1, x' + 1) = 0)$
 - soit $(a(p' + 1, x') \leq C$ et $a(p', a(p' + 1, x')) \neq 0$ et $a(p' + 1, x' + 1) = a(p', a(p' + 1, x')))$
- à écrire avec $a_{i,j-1} = \pi_1 \pi_2^{2+i\pi_1\pi_2 y+(j-1)} y \dots$
3. $\pi_1 y \geq p + 1$ et $\pi_1 \pi_2 y \geq x + 1$ et $\pi_1 \pi_2^{2+(x+1)\pi_1\pi_2 y+p} y \neq 0$

On obtient une forme récursive primitive pour $f(y, x, p)$.

$$y = K_4(a_{p+1,x+1}, L, C, K_{(L+1)(C+1)}(a_{00}, \dots))$$

Dans les conditions, on ajoute que le $a_{p+1,x+1}$ est celui du tableau.

On a $A(p + 1, x + 1) = \pi_1 \text{Sch } \mu(y) f(y, x, p) \square$

Théorème :

Toute fonction récursive f s'écrit sous la forme $f(x) = \pi_1 \text{Sch } \mu(g)(x)$ avec g récursive primitive.

DÉMONSTRATION :

Soit M une machine de Turing à un seul ruban.

On code une configuration $a_0, a_1, \dots, a_i, \dots, a_l, B, \dots$ par le mot $a_0 a_1 \dots a_i q a_{i+1} \dots a_l$ pour q l'état de la machine.

Il existe une fonction récursive primitive qui reconnaît une configuration initiale (en checkant si la deuxième lettre est q_0).

Il existe une fonction récursive primitive qui reconnaît une configuration finale (plus compliqué).

Il existe une fonction récursive primitive qui vérifie qu'une configuration est bien la suivante d'une autre.

Donc il existe une fonction récursive primitive qui, en ayant en entrée le code d'un diagramme espace-temps, vérifie que c'est un bon calcul. \square

Chapitre 2

Les systèmes acceptables de programmation et leur propriétés

2.1 Codage et énumération des machines et des configurations

Notation :

On note \bar{n}^2 l'écriture en binaire de n .

Remarque :

Dans toute la suite, on construit des codes sur $\{0; 1; ; \heartsuit\}^*$. Toutefois en appliquant la transformation suivante, on peut transformer tout code sur $\{0; 1; ; \heartsuit\}^*$ en un code sur $\{0; 1; \}$:

$$\begin{array}{lcl} 0 & \mapsto & 00 \\ 1 & \mapsto & 01 \\ , & \mapsto & 10 \\ \heartsuit & \mapsto & 11 \end{array}$$

Soit M une machine à un seul ruban bi-infini.

Soit $Q = \{q_1, \dots, q_n\}$ l'ensemble de ses états et $F \subseteq Q$ l'ensemble de ses états finaux.

Soit $\Sigma = \{a_0 = B, a_1, \dots, a_\sigma\}$ son alphabet.

Soit δ sa fonction de transition.

On code alors M sur l'alphabet $\{0; 1; ; \heartsuit\}^*$:

Objet	Codage
Transition $\delta : (q_i, a_j) \mapsto (q_k, a_l, \varepsilon)$	$\bar{i}^2, \bar{j}^2, \bar{k}^2, \bar{l}^2$
Fonction δ	$\text{trans}_1 \heartsuit \text{trans}_2 \heartsuit \dots \heartsuit \text{trans}_N$ Où trans_p désigne l'encodage de la $p^{\text{ième}}$ transition de δ
Machine M	$\bar{n}^2 \heartsuit \underbrace{0 \heartsuit 1 \heartsuit \dots \heartsuit \varepsilon_i \heartsuit \dots}_{\varepsilon_i = \begin{cases} 1 & \text{si } q_i \in F \\ 0 & \text{sinon} \end{cases}} \heartsuit \bar{N}^2 \heartsuit \bar{\sigma}^2 \heartsuit \text{trans}_1 \heartsuit \dots \heartsuit \text{trans}_N$

Lemme :

Le langage $L \subseteq \{0; 1\}^*$ des codes des machines à un ruban bi-infini est reconnu par une machine de Turing appelée analyseur.

Définition :

On peut associer à une machine codée par le mot m le numéro α qui est son numéro dans l'ordre hiérarchique des mots qui sont des codes de machines.

On appellera φ_α la fonction calculée par la machine de numéro α .

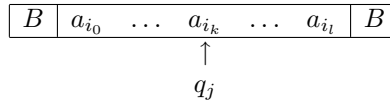
Lemme :

Il existe une machine qui sur l'entrée $u \in \{0;1\}^*$ s'arrête seulement si u est le code d'une machine et dans ce cas calcule son numéro α .

Il existe une machine qui sur l'entrée α calcule le code de φ_α .

Définition :

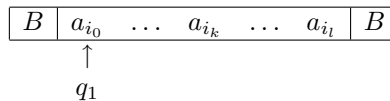
Soit une configuration :



On définit son code dans $\{0;1;,\;;\heartsuit\}$ par :

$$\overline{i_0}^{-2}, \overline{i_1}^{-2}, \dots, \overline{i_k}^{-2} \heartsuit \overline{j}^{-2}, \overline{i_{k+1}}^{-2}, \dots, \overline{i_l}^{-2}$$

Soit une configuration initiale :



On définit son code dans $\{0;1;,\;;\heartsuit\}$ par :

$$\overline{i_0}^{-2}, \dots, \overline{i_l}^{-2}$$

Lemme :

Le langage $L \subseteq \{0;1\}^*$ des codes des configurations (resp. configurations initiales) est reconnu par une machine de Turing.

Définition :

On peut associer à une configuration (resp. configuration initiale) codée par le mot m le numéro α qui est son numéro dans l'ordre hiérarchique des mots qui sont des codes de configurations (resp. configurations initiales).

On note $\varphi_\alpha(i)$ le résultat du calcul de la machine numéro α sur la i -ième configuration initiale.

Lemme :

Il existe une machine qui sur l'entrée $u \in \{0;1\}^*$ s'arrête seulement si u est le code d'une configuration (resp. configuration initiale) et dans ce cas calcule son numéro i .

Il existe une machine qui sur l'entrée i calcule le code la i -ième configuration (resp. configuration initiale) de la machine φ_α .

Théorème de l'arrêt :

Il n'existe pas de machine de Turing qui décide sur l'entrée $\langle i, j \rangle$ si $\varphi_i(j)$ est défini.

DÉMONSTRATION :

Supposons que le problème de l'arrêt soit décidable.

Soit M la machine qui :

- renvoie 1 si $\varphi_i(i)$ termine ;
- renvoie 0 si $\varphi_i(i)$ boucle.

Ceci est possible par hypothèse.

Soit M' la machine qui :

- renvoie 0 si M renvoie 0 sur la même entrée ;
- boucle si M renvoie 1 sur la même entrée.

Soit a le numéro de M' , alors $M' = \varphi_a$.

Il y a alors deux cas possibles :

- si $\varphi_a(a) = 0$ alors par définition $M'(a) = 0$ donc $M(a) = 0$ donc $\varphi_a(a)$ boucle : absurde.
- si $\varphi_a(a)$ non défini alors par définition $M(a) = 1$ donc $\varphi_a(a)$ termine : absurde.

□

2.2 Machines de Turing universelles

Définition :

Une machine de Turing φ_α est dite universelle si pour tous $i, j \in \mathbb{N}$, $\varphi_\alpha(\langle i, j \rangle)$ s'arrête si et seulement si $\varphi_i(j)$ s'arrête et dans ce cas les valeurs calculées sont égales.

Proposition :

Les machines de Turing universelles existent.

Exemple :

Un ordinateur (à mémoire infinie...)

Contre-exemple :

Une machine à laver

DÉMONSTRATION :

Soit la machine qui :

- Décode $\langle i, j \rangle$ et met i sur le premier ruban, j sur le deuxième;
- Trouve le code de φ_i ;
- Trouve le code de la j -ième configuration initiale de φ_i ;
- Simule φ_i sur l'entrée j .

Chacun de ces opérations est calculable d'après les lemmes précédents. \square

Notation :

On note φ_{univ} une machine universelle.

2.3 Théorème smn

Théorème :

Pour tous $n, m \in \mathbb{N}$ il existe une fonction s_n^m à $(m+1)$ variables, récursive totale telle que pour tous $z, x_1, \dots, x_m, y_1, \dots, y_n$ on ait $\varphi_z(\langle x_1, \dots, x_m, y_1, \dots, y_n \rangle) = \varphi_{s_n^m(z, x_1, \dots, x_m)}(\langle y_1, \dots, y_n \rangle)$.

DÉMONSTRATION :

Pour m et n fixés :

1. Considérons pour z, x_1, \dots, x_m la machine M qui fait ce qui suit sur l'entrée $\langle y_1, \dots, y_n \rangle$:
 - décode y_1, \dots, y_n
 - écrit $z, x_1, \dots, x_m, y_1, \dots, y_n$
 - le code en $\langle z, \langle x_1, \dots, x_m, y_1, \dots, y_n \rangle \rangle$
 - simule φ_{univ} sur cette entrée

Alors sur l'entrée $\langle y_1, \dots, y_n \rangle$, M calcule $\varphi_z(\langle x_1, \dots, x_m, y_1, \dots, y_n \rangle)$

2. Considérons la fonction f qui à $\langle z, x_1, \dots, x_m \rangle$ associe le numéro du code de M .

Alors f est récursive totale. Posons $s_n^m = f$.

Alors $\varphi_{s_n^m(z, x_1, \dots, x_m)}(y_1, \dots, y_n) = M(y_1, \dots, y_n) = \varphi_z(\langle x_1, \dots, x_m, y_1, \dots, y_n \rangle)$

\square

Théorème :

Il existe une fonction c récursive totale telle que $\forall(i, j), \varphi_{c(i, j)} = \varphi_j \circ \varphi_i$

Corollaire :

Il existe α récursive totale telle que $\forall(i, j), \varphi_{\alpha(j)}(i) = \varphi_i(j)$

DÉMONSTRATION :

- **Pour les machines de Turing :** Considérons la machine c qui :
 - écrit le code de φ_i ;
 - écrit le code qui nettoie le ruban;
 - écrit le code de φ_j en décalant les états;
 - construit le code de la machine contenant ces deux codes;

- cherche le numéro de ce code.
- Alors sur l'entrée $\langle i, j \rangle$, cette machine calcule le code de $\varphi_j \circ \varphi_i$.
- **Dans le cas général :** La fonction $(i, j, x) \mapsto \varphi_j \circ \varphi_i(x)$ est récursive car elle est la composée de deux fonctions récursives.
- Soit b un de ses numéros. Alors :

$$\varphi_b(i, j, x) = \varphi_j \circ \varphi_i(x)$$

Alors d'après le théorème smn :

$$\varphi_b(i, j, x) = \underbrace{\varphi_{s_1^2(b, i, j)}(x)}_{c(i, j)} = \varphi_i \circ \varphi_j(x)$$

De plus, c est bien récursive totale, toujours d'après le théorème smn.

□

DÉMONSTRATION :

du corollaire

- **Pour les machines de Turing :**

Soit j fixé, considérons la machine M_j de code suivant :

- sur l'entrée i écrit i, j
- le code en $\langle i, j \rangle$
- simule φ_{univ}

Le résultat du calcul de cette machine sur l'entrée i est $\varphi_i(j)$.

Soit α la fonction qui à j associe le numéro du code de cette machine.

Alors α est bien récursive totale et on a $\varphi_{\alpha(j)}(i) = \varphi_i(j)$ par construction.

- **Dans le cas général :**

On a :

$$\begin{aligned} \varphi_i(j) &= \varphi_{\text{univ}}(\langle i, j \rangle) = \varphi_{\text{univ}} \circ \underbrace{K(\pi_2, \pi_1)}_{=\varphi_a}(\langle j, i \rangle) \\ &= \underbrace{\varphi_{\text{univ}} \circ \varphi_a}_{=\varphi_b(a)}(\langle j, i \rangle) \\ &= \varphi_b(\langle a, j, i \rangle) \\ &= \underbrace{\varphi_{s_1^2(b, a, j)}}_{\alpha(j)}(i) \end{aligned}$$

De plus, α est récursive totale d'après le théorème smn.

□

2.4 Théorème de Rice

Théorème :

Soit \mathcal{F} l'ensemble des fonctions calculables de \mathbb{N} dans \mathbb{N} et soit $X \subseteq \mathcal{F}$.

Soit $A = \{i \in \mathbb{N} \mid \varphi_i \text{ calcule une fonction dans } X\}$.

Alors soit χ_A est la fonction totale nulle, soit χ_A est la fonction totale unité, soit χ_A n'est pas récursive.

Exemple :

- $X = \{ \text{la fonction nulle} \}$;
- $X = \{ \text{les fonctions totales} \}$;
- $X = \{ \text{les fonctions croissantes} \}$.

DÉMONSTRATION :

On suppose $A \neq \emptyset$ et $A \neq \mathbb{N}$

Soit a tel que φ_a est la fonction de domaine vide.

On peut supposer que $a \in A$ car χ_A est récursive $\Leftrightarrow \chi_{\mathbb{C}_A}$ l'est et $a \in A \cup \mathbb{C}_A$.

Soit $b \in \mathbb{N} \setminus A$.

Soit $\psi(x, y, z) = \varphi_b(z) + \varphi_x(y) - \varphi_x(y)$.

Attention : on ne simplifie pas !

Alors ψ est récursive donc il existe c tel que $\varphi_c = \psi$ et donc $\psi(x, y, z) = \varphi_{s_1^2(c, x, y)}(z)$.

- Si $\varphi_x(y)$ est défini alors $\varphi_{s_1^2(c,x,y)}(z) = \varphi_b(z)$.
Donc $s_1^2(c,x,y) \in \mathbb{N} \setminus A$ par définition de b .
Donc $\chi_A(s_1^2(c,x,y)) = 0$.
- Si $\varphi_x(y)$ n'est pas défini alors $\varphi_{s_1^2(c,x,y)} = \varphi_a$.
Donc $s_1^2(c,x,y) \in A$ par définition de a .
Donc $\chi_A(s_1^2(c,x,y)) = 1$

Donc $\chi_A(s_1^2(c,x,y))$ décide de l'arrêt de $\varphi_x(y)$ donc n'est pas récursive i.e. χ_A n'est pas récursive. \square

2.5 Théorème du point fixe

Théorème :

Pour toute fonction récursive totale f il existe $n \in \mathbb{N}$ tel que $\varphi_n = \varphi_{f(n)}$

DÉMONSTRATION :

On considère la fonction $f : (x, y) \mapsto \varphi_{f(s_1^1(y,y))}(x)$.

Elle est récursive car égale à $\varphi_{\text{univ}}(\langle f(s_1^1(y,y)), x \rangle)$.

Soit a un de ses numéros.

Alors : $\varphi_{f(s_1^1(y,y))}(x) = \varphi_a(y, x) = \varphi_{s_1^1(a,y)}(x)$

On pose $y = a$ et $n = s_1^1(a, a)$. Et alors :

$$\varphi_{f(n)}(x) = \varphi_{f(s_1^1(a,a))}(x) = \varphi_{s_1^1(a,a)}(x) = \varphi_n(x)$$

\square

Exemple :

On appelle **quine** une fonction qui écrit son propre code.

Soit f la fonction qui à i associe le numéro du code d'une machine M_i qui sans tenir compte de son entrée calcule le code de la machine numéro i .

Alors f est récursive totale donc il existe n tel que $\varphi_n = \varphi_{f(n)}$.

Alors :

$$\varphi_n(i) = \varphi_{f(n)}(i) = M_n(i) = \text{le code de la machine } n$$

Donc φ_n écrit son propre code !

Théorème du point fixe effectif :

Il existe une fonction récursive totale g telle que pour toute fonction f de numéro j on ait $\varphi_{g(j)} = \varphi_{f(g(j))}$.

Exemple :

Soit fact la fonction factorielle :

$$\begin{cases} \text{fact}(0) &= 1 \\ \text{fact}(n) &= n \times \text{fact}(n-1) \end{cases}$$

Soit f la fonction qui à i associe le numéro de la machine suivante :

- Sur l'entrée 0 la machine calcule 1
- Sur l'entrée n la machine calcule $n \times \varphi_i(n-1)$

Si $\varphi_{f(i)} = \varphi_i$ alors :

- Si $n > 0$ alors $\varphi_i(n) = \varphi_{f(i)}(n) = n \times \varphi_i(n-1)$
- $\varphi_i(0) = 1$

Donc φ_i est le fonction factorielle !

Si j est un numéro de f alors il existe g récursive totale telle que $g(j)$ est de numéro pour fact.

Exemple :

Soit f la fonction qui à i associe le numéro de la machine suivante :

- $\langle 0, x \rangle \mapsto x + 1$
- $\langle p, 0 \rangle \mapsto \varphi_i \langle p-1, 1 \rangle$
- $\langle p+1, x+1 \rangle \mapsto \varphi_i \langle p, \varphi_i \langle p+1, x \rangle \rangle$

f est bien récursive totale car définie par cas et composition de fonction récursives.
De plus, si $\varphi_i = \varphi_{f(i)}$ alors φ_i est la fonction d'Ackermann.

Définition :

Soit Φ_0, Φ_1, \dots une suite de fonctions partielles de \mathbb{N} dans \mathbb{N} .

C'est un système acceptable de programmation si :

1. L'ensemble des Φ_i est exactement l'ensemble des fonctions récursives
2. Il existe une fonction universelle Φ_{univ} dans la liste telle que $\forall (i, x) \in \mathbb{N}^2, \Phi_{\text{univ}}(\langle i, x \rangle) = \Phi_i(x)$
3. Il existe une fonction récursive totale c de composition telle que $\forall (i, j) \in \mathbb{N}^2, \Phi_i \circ \Phi_j = \Phi_{c(i, j)}$

Remarque :

Le théorèmes smn, le théorème de l'arrêt et le théorème de Rice sont vrais dans tout système acceptable de programmation.

Définition :

En mathématiques, les systèmes acceptables de programmation sont appelés les nombres de Gödel des fonctions récursives.

2.6 Le théorème de l'isomorphisme de Rogers

Lemme :

Soient (Φ_i) un système acceptable de programmation et h une fonction récursive totale.

Il existe i tel que :

- $x \mapsto s_1^1(i, x)$ est une fonction injective
- $\forall x, \Phi_{h(x)} = \Phi_{s_1^1(i, x)}$

DÉMONSTRATION :

Soit k tel que $\Phi_k = h$. Alors pour tout y :

$$\begin{aligned} \Phi_{h(x)}(y) &= \Phi_{\text{univ}}(h(x), y) \\ &= \Phi_{\text{univ}}(\Phi_k(x), y) \\ &= \Phi_{\text{univ}}(\Phi_{\text{univ}}(k, x), y) \end{aligned}$$

Donc $k \mapsto x \mapsto y \mapsto \Phi_{h(x)}(y)$ est récursive : soit l un de ses numéros. Alors :

$$\Phi_{h(x)}(y) = \Phi_l(k, x, y) = \underbrace{\Phi_{s_1^1(l, k)}}_i(x, y) = \Phi_i(x, y) = \Phi_{s_1^1(i, x)}(y)$$

Problème : $s_1^1(i, x)$ n'est pas injective pour tout i .

On utilise le théorème de Kleene pour montrer qu'il existe i tel que :

$$\Phi_i(j, y) = \begin{cases} 0 & \text{si } \exists k < j, s_1^1(i, k) = s_1^1(i, j) \\ 1 & \text{si } \forall k < j, s_1^1(i, k) \neq s_1^1(i, j) \text{ et } \exists j < k < y, s_1^1(i, k) = s_1^1(i, j) \\ \Phi_{h(j)}(y) & \text{sinon} \end{cases}$$

En effet, soit f la fonction qui à i associe le numéro de la machine qui calcule $\Phi_{f(i)}(j, y)$ tq

- $\Phi_{f(i)}(j, y) = 0$ si $\exists k < j, s_1^1(i, k) = s_1^1(i, j)$
- $\Phi_{f(i)}(j, y) = 1$ si $\forall k < j, s_1^1(i, k) \neq s_1^1(i, j)$ et $\exists j < k < y, s_1^1(i, k) = s_1^1(i, j)$
- $\Phi_{f(i)}(j, y) = \Phi_{h(j)}(y)$ sinon

Montrons alors que $x \mapsto s_1^1(i, x)$ injective et $\Phi_{s_1^1(i, x)} = \Phi_{h(x)}$.

- Supposons que $x \mapsto s_1^1(i, x)$ n'est pas injective. Alors :

$$\exists k < j, s_1^1(i, k) = s_1^1(i, j)$$

Alors :

$$\begin{aligned}
 \Phi_i(j, y) &= 0 \text{ par définition} \\
 &= \Phi_{s_1^1(i,j)}(y) \text{ par smn} \\
 &= \Phi_{s_1^1(i,k)}(y) \text{ par hypothèse} \\
 &= \Phi_i(k, y)
 \end{aligned}$$

Prenons j minimum et $y = k + 1$ et alors :

$$\begin{aligned}
 \Phi_i(k, y) &= 1 \text{ par définition et minimalité de } j \\
 &= \Phi_{s_1^1(i,k)}(y) \text{ par smn} \\
 &= \Phi_{s_1^1(i,k)}(y) \text{ par hypothèse} \\
 &= \Phi_i(j, y)
 \end{aligned}$$

Donc $0 = \Phi_i(j, y) = 1$ ce qui est absurde.

– Pour tout y :

$$\begin{aligned}
 \Phi_{s_1^1(i,x)}(y) &= \Phi_i(x, y) \text{ par smn} \\
 &= \Phi_{h(x)}(y) \text{ par injectivité de } s_1^1(i, x) \text{ et définition de } \Phi_i
 \end{aligned}$$

□

Théorème :

Soient (Φ_i) et (Ψ_i) deux systèmes acceptables de programmation.

Il existe une bijection récursive totale f telle que $\forall x, \Phi_x = \Psi_{f(x)}$.

Lemme :

Il existe une f récursive totale telle que $\forall x, \Phi_x = \Psi_{f(x)}$.

DÉMONSTRATION :

Il existe Φ_{univ} récursive universelle.

Il existe k tq $\Phi_{\text{univ}} = \Psi_k$.

Alors

$$\forall (x, i) \in \mathbb{N}^2, \Phi_x(i) = \Phi_{\text{univ}}\langle x, i \rangle = \Psi_k\langle x, i \rangle = \Psi_{s_{\Psi_1^1}(k,x)}(i)$$

On pose $f(x) = s_{\Psi_1^1}$ qui est récursive totale. □

Remarque :

Attention : A priori $s_{\Psi_1^1} \neq s_{\Phi_1^1}$!

Lemme :

Il existe une f récursive totale injective telle que $\forall x, \Phi_x = \Psi_{f(x)}$.

DÉMONSTRATION :

Si g est récursive totale alors d'après le lemme, il existe i tel que $x \mapsto s_{\Psi_1^1}(i, x)$ soit injective et $\Psi_{g(x)} = \Psi_{s_{\Psi_1^1}(i,x)}$.

Prenons $g(x) = s_{\Psi_1^1}(k, x)$.

On pose $f(x) = s_{\Psi_1^1}(i, x)$ et alors f est injective d'après le lemme précédent. Alors :

$$\begin{aligned}
 \Phi_x &= \Psi_{s_{\Psi_1^1}(k,x)} \text{ d'après le lemme précédent} \\
 &= \Psi_{g(x)} \text{ par définition de } f \\
 &= \Psi_{s_{\Psi_1^1}(i,x)} \text{ d'après ce qui précède} \\
 &= \Psi_{f(x)}
 \end{aligned}$$

Donc f convient. □

Lemme de Bourrage :

Pour tout (Ψ_i) système acceptable de programmation, il existe une fonction p récursive totale injective telle que $\forall (x, y), \Psi_x = \Psi_{p(x,y)}$.

DÉMONSTRATION :

– **Pour les machines de Turing :**

Si le nombre d'état de la machine de numéro x est $q(x)$, on ajoute au code $\chi_3(q(x), x, y) - q(x)$ états inutiles. On a alors une machine qui a $\chi_3(q(x), x, y)$ état et comme χ_3 est injective et que deux machines ayant un nombre différent d'états sont différentes, si l'on note $p(x, y)$ le numéro de cette machine alors p est injective.

– **Dans le cas général :**

Soit (Φ_i) le système acceptable de programmation des machines de Turing et (Ψ_i) un autre système acceptable de programmation.

D'après le lemme précédent, il existe f et g récursives totales injectives telles que :

$$\Phi_x = \Psi_{f(x)} \text{ et } \Psi_x = \Phi_{g(x)}$$

Soit p_M la fonction de bourrage pour les machines de Turing.

Posons $p(x, y) = f(p_M(g(x), y))$. Alors :

1. p est injective car g, p_M et f le sont.
2. $\Psi_{p(x,y)} = \Psi_{f(p_M(g(x),y))} = \Phi_{p_M(g(x),y)} = \Phi_{g(x)} = \Psi_x$

□

Lemme :

Il existe une f récursive totale injective et strictement croissante telle que $\forall x, \Phi_x = \Psi_{f(x)}$ et $f(0) > 0$.

DÉMONSTRATION :

On prend h du lemme précédent et p une fonction de bourrage du système acceptable de programmation. On pose :

$$\begin{cases} f(0) &= p(h(0), \min\{y | p(h(0), y) > 0\}) \\ f(x+1) &= p(h(x+1), \min\{y | p(h(x+1), y) > f(x)\}) \end{cases}$$

□

DÉMONSTRATION DU THÉORÈME :

D'après le lemme précédent, il existe g et h récursives totale strictement croissantes telles que :

$$\begin{cases} \forall x, \Phi_x = \Psi_{g(x)} \\ \forall x, \Psi_x = \Phi_{h(x)} \\ f(0) > 0 \\ g(0) > 0 \end{cases}$$

D'où :

$$\forall x, g(x) > x \text{ et } h(x) > x$$

Soit x , on définit $f(x)$ en alternant l'application de h^{-1} et g^{-1} jusqu'à ce que le résultat obtenu ne soit ni dans l'image de g ni dans l'image de h . Autrement dit, on "zig-zag" entre le monde Ψ et le monde Φ et on regarde où l'on s'arrête : si l'on s'arrête en bas alors on pose $f(x) = h^{-1}(x)$ et sinon on pose $f(x) = g(x)$.

Formellement on pose :

$$u_0(x) = x \quad u_{2n}(x) = \begin{cases} g^{-1}(u_{2n-1}(x)) & \text{s'il existe} \\ \perp & \text{sinon} \end{cases} \quad u_{2n+1}(x) = \begin{cases} h^{-1}(u_{2n}(x)) & \text{s'il existe} \\ \perp & \text{sinon} \end{cases}$$

On pose alors :

$$k(x) = \min \{n \geq 0 \mid u_n(x) = \perp\} - 1$$

On est assuré que $k(x)$ existe car g et h étant strictement croissantes, g^{-1} et h^{-1} sont strictement décroissantes et donc $(u_n(x))_{n \in \mathbb{N}}$ est une suite strictement décroissante de $\mathbb{N}^{\mathbb{N}}$ jusqu'à ce qu'elle atteigne \perp .

On pose enfin :

$$f(x) = \begin{cases} h^{-1}(x) & \text{si } k \text{ est impair} \\ g(x) & \text{si } k \text{ est pair} \end{cases}$$

DESSIN !

Montrons que f est injective : supposons que $f(x) = f(y)$ et $x \neq y$

– **Premier cas :** Si $f(x) = g(x)$.

DESSIN

Alors par définition de f , on a nécessairement $f(y) = h^{-1}(y)$ car sinon on aurait $x = y$. Mais alors, le "zig-zag" partant de y rejoint celui de x puisque :

$$g^{-1}(h^{-1}(y)) = g^{-1}(f(y)) = g^{-1}(f(x)) = g^{-1}(g(x)) = x$$

Donc le "zig-zag" partant de y passe par x donc finit au même endroit que x donc $f(y) = g(y)$ par définition de f . Ainsi, $f(y) = f(x) = g(x) = g(y)$ donc $x = y$ par stricte croissante de g ce qui est absurde.

- **Deuxième cas** : Si $f(x) = h^{-1}(x)$.

DESSIN

De même, $x \neq y$ impose $f(y) = g(y)$. Mais alors :

$$g^{-1}(h^{-1}(x)) = g^{-1}(f(x)) = g^{-1}(f(y)) = g^{-1}(g(y)) = y$$

Donc le “zig-zag” partant de x passe par y donc finit au même endroit que y donc $f(x) = g(x)$ par définition de f . Ainsi $f(x) = f(y) = g(x) = g(y)$ donc $x = y$ par stricte croissante de g ce qui est absurde.

Montrons que f est surjective : soit $y \in \mathbb{N}$.

Considérons le “zig-zag” partant de $z = h(y)$:

- **Premier cas** : S’il finit en haut (*i.e.* si $f(z) = g(z)$) alors nécessairement $u = g^{-1}(y)$ existe et le “zig-zag” partant de u finit lui aussi en haut. Donc $f(u) = g(u) = g(g^{-1}(y)) = y$. Donc y a un antécédent par f .
- **Deuxième cas** : S’il finit en bas (*i.e.* si $f(z) = h^{-1}(z)$). Alors $f(z) = h^{-1}(z) = h^{-1}(h(y)) = y$. Donc y a un antécédent par f .

□

2.7 Propriétés des ensembles d’entiers

Définition :

- $\text{Seq } \mathbb{N}$ est l’ensemble des suites finies de \mathbb{N} : $\text{Seq } \mathbb{N} = \cup_{k \in \mathbb{N}} \mathbb{N}^k$.
- Une partie A de $\text{Seq } \mathbb{N}$ est dite réursive (ou décidable ou calculable) si sa fonction caractéristique χ_A est réursive totale.

Proposition :

Si A est finie alors A est réursive. Si A est décidable alors \bar{A} aussi.

Définition :

$A \subseteq \text{Seq } \mathbb{N}$ est dite réursivement énumérable si A est le domaine de définition d’une fonction réursive.

Proposition :

1. Si A est réursive alors A est réursivement énumérable
2. Si A et \bar{A} sont réursivement énumérables alors A et \bar{A} sont réursives.
3. Si A est réursivement énumérable et infini alors il existe une bijection réursive totale entre \mathbb{N} et A .

DÉMONSTRATION :

1. Soit M une machine qui calcule χ_A sur l’entrée x .
Soit M' qui sur l’entrée x calcule $M(x)$ puis boucle si $M(x) = 0$ et renvoie 1 sinon.
Alors $\text{dom } M' = \{x \mid M(x) \neq 0\} = \{x \mid \chi_A(x) = 1\} = A$
2. Soit M calculant une fonction de domaine A et M' calculant une fonction de domaine \bar{A} .
Soit M'' une machine qui sur l’entrée x fait simule M et M' en parallèle (*i.e.* alterne un pas de l’une puis un pas de l’autre), et qui renvoie 1 si M termine en premier et 0 sinon M' termine en premier.
Comme $\forall x \in \mathbb{N}$, $M(x)$ ou $M'(x)$ termine, M'' est termine toujours et $M'' = \chi_A$.
3. **Supposons A réursivement énumérable et infini :**
Soit M une machine qui calcule une fonction de domaine A .
Soit M' une machine qui :
 - simule 1 pas de calcul de $M(0)$
 - puis simule 2 pas de calcul de $M(0)$ et 1 pas de calcul de $M(1)$
 - puis simule 3 pas de calcul de $M(0)$ et 2 pas de calcul de $M(1)$ et 1 pas de calcul de $M(3)$
 - *etc.*
 Si l’on note :
 - $f(0)$ le résultat du premier calcul qui termine
 - $f(1)$ le résultat du second calcul qui termine
 - $f(1)$ le résultat du troisième calcul qui termine
 - *etc.*

Sur l'entrée n , M' renvoie $f(n)$.

Alors M' va énumérer tous les éléments de A donc M' est bijective de $\text{dom } M'$ dans A et comme A est infini, $\text{dom } M' = \mathbb{N}$. Donc M' est une bijection de \mathbb{N} dans A .

S'il existe une bijection récursive totale de \mathbb{N} dans A :

Soit M une telle bijection. Soit M' une machine qui sur l'entrée a :

- calcule $M(0)$ et renvoie 1 si $M(0) = a$
- puis calcule $M(1)$ et renvoie 1 si $M(1) = a$
- etc.

Alors, si $a \in A$, $M'(a)$ fini car comme M est bijective, $\exists n, M(n) = a$ donc M' finira par le calculer.

De plus, si $a \notin A$, alors M' ne trouvera jamais n tel que $M(n) = a$ et comme \mathbb{N} est infini, elle bouclera.

Donc $\text{dom } M' = A$.

□

Chapitre 3

Logique

3.1 Calcul propositionnel

3.1.1 Syntaxe

Définition :

Le langage est constitué :

- d'un ensemble de variables propositionnelles \mathcal{P} (dénombrable ou pas) ;
- de connecteurs logiques \neg (unaire), \wedge , \vee , \Rightarrow , \Leftrightarrow (binaires) ;
- de parenthèses.

Les briques de base des formules \mathcal{F} sont les variables.

Si F_1 et F_2 sont des formules alors $\neg F_1$, $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \Rightarrow F_2)$ et $(F_1 \Leftrightarrow F_2)$ aussi.

Définition :

L'ensemble \mathcal{F} des formules du calcul propositionnel est le plus petit ensemble de mots sur $\mathcal{P} \cup \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)\}$ tel que $\mathcal{P} \subseteq \mathcal{F}$ et si $(F_1, F_2) \in \mathcal{F}^2$ alors $\neg F_1 \in \mathcal{F}$ et $(F_1 \circ F_2) \in \mathcal{F}$ pour tout $\circ \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$.

Définition :

On pose $\mathcal{F}_0 = \mathcal{P}$ et pour $n \in \mathbb{N}$, $\mathcal{F}_{n+1} = \mathcal{F}_n \cup \{\neg F_1, (F_1 \circ F_2) \mid (F_1, F_2) \in (\mathcal{F}_n)^2, \circ \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}\}$.

Alors $\mathcal{F} = \bigcup_{n \in \mathbb{N}} \mathcal{F}_n$.

Intérêt : preuves par induction.

Théorème de lecture unique :

Pour tout $F \in \mathcal{F}$, on se trouve dans un et un seul des cas suivants :

1. $F \in \mathcal{P}$;
2. il existe un unique $F_1 \in \mathcal{F}$ tel que $F = \neg F_1$;
3. il existe deux uniques $(F_1, F_2) \in \mathcal{F}^2$ et un unique $\circ \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$ tels que $F = (F_1 \circ F_2)$.

DÉMONSTRATION :

En TD. \square

Définition :

On appelle valuation toute fonction $\nu \in \{0, 1\}^{\mathcal{P}}$.

Lemme de lecture unique :

Pour toute valuation ν , il existe une unique fonction $\bar{\nu} \in \{0, 1\}^{\mathcal{F}}$ qui prolonge ν et telle que pour tous $(F_1, F_2) \in \mathcal{F}^2$:

1. $\bar{\nu}(\neg F_1) = 1 - \bar{\nu}(F_1)$
2. $\bar{\nu}(F_1 \wedge F_2) = \bar{\nu}(F_1) \cdot \bar{\nu}(F_2)$
3. $\bar{\nu}(F_1 \vee F_2) = \max(\bar{\nu}(F_1), \bar{\nu}(F_2))$
4. $\bar{\nu}(F_1 \Rightarrow F_2) = \max(1 - \bar{\nu}(F_1), \bar{\nu}(F_2))$

$$5. \bar{\nu}(F_1 \Leftrightarrow F_2) = 1 \text{ ssi } \bar{\nu}(F_1) = \bar{\nu}(F_2)$$

DÉMONSTRATION :

On montre par induction sur \mathcal{F} que $\bar{\nu}$ est définie de manière unique.

- Si $F \in \mathcal{F}_0$, alors $\bar{\nu}(F) = \nu(F)$;
- Soit $n \in \mathbb{N}$. Supposons que $\bar{\nu}$ est définie de manière unique sur \mathcal{F}_n . Soit $F \in \mathcal{F}_{n+1}$.
 - Si $F \in \mathcal{F}_n$: fini ;
 - Sinon, d'après le lemme de lecture unique, on se trouve dans exactement un des cas de construction de F à partir d'éléments de \mathcal{F}_n , pour chacun desquels $\bar{\nu}$ est définie de manière unique.

□

Remarque :

Par abus de notation, on notera pour tout $\nu \in \{0, 1\}^{\mathcal{P}}$, $\nu \equiv \bar{\nu} \in \{0, 1\}^{\mathcal{F}}$.

3.1.2 Tables de vérité

La valuation $\nu(F)$ ne dépend que de la valeur de ν sur les variables apparaissant dans F .

Définition :

- On dit qu'une formule F est satisfaite par une valuation ν si $\nu(F) = 1$.
- Une tautologie est une formule satisfaite pour toute valuation.
- Un ensemble de formules E est satisfiable s'il existe une valuation ν qui satisfait chacune des formules de E .
- Un ensemble de formules E est finiment satisfiable si tout sous-ensemble fini de E est satisfiable.
- Une formule ϕ est conséquence sémantique d'un ensemble de formules E si toute valuation satisfaisant E satisfait ϕ .

Théorème de complétude du calcul propositionnel :

Les notions de conséquence sémantique et syntaxique sont les mêmes.

Théorème de compacité :

Un ensemble E de formules est satisfiable ssi il est finiment satisfiable.

A faire

(Ne doit-on pas échanger \exists et \forall ? : $\exists(\varepsilon_n)_{n \in \mathbb{N}} \forall n \dots$)

DÉMONSTRATION DANS LE CAS \mathcal{P} DÉNOMBREABLE :

Posons $\mathcal{P} = \{x_n, n \in \mathbb{N}\}$. Soit A un ensemble de formules sur \mathcal{P} finiment satisfiable. On raisonne par récurrence sur $n \in \mathbb{N}$ pour montrer H_n : pour tout $n \in \mathbb{N}$, il existe $(\varepsilon_0, \dots, \varepsilon_n) \in \{0, 1\}^{n+1}$ tels que pour tout $B \subseteq A$ fini, il existe une valuation ν qui satisfait B et telle que pour tout $i \leq n$, $\nu(x_i) = \varepsilon_i$.

- Cas de base : de deux choses l'une.

1. Pour tout $B \subseteq A$ fini, il existe une valuation ν satisfaisant B telle que $\nu(x_0) = 0$. On pose alors $\varepsilon_0 = 0$;
2. Il existe $B_0 \subseteq A$ fini, tel que pour toute valuation ν , si ν satisfait B_0 alors $\nu(x_0) = 1$. On pose alors $\varepsilon_0 = 1$.

Soit $B \subseteq A$ fini. Si on est dans le premier cas, H_n est vérifiée. Sinon, $\varepsilon_0 = 1$. Par hypothèse, comme $B \cup B_0$ est fini, soit ν une valuation qui satisfait $B \cup B_0$. En particulier, $\nu(x_0) = 1$ puisque ν satisfait B_0 . On a trouvé le ν recherché.

- Soit $n \in \mathbb{N}$, on suppose H_n vérifiée. On se retrouve encore dans l'un des deux cas suivants :

1. Pour tout $B \subseteq A$ fini, il existe une valuation ν satisfaisant B telle que $\nu(x_{n+1}) = 0$ et pour tout $i \leq n$, $\nu(x_i) = \varepsilon_i$. On pose alors $\varepsilon_{n+1} = 0$;
2. Il existe $B_{n+1} \subseteq A$ fini, tel que pour toute valuation ν , si ν satisfait B_{n+1} et si pour tout $i \leq n$, on a $\nu(x_i) = \varepsilon_i$, alors $\nu(x_{n+1}) = 1$. On pose alors $\varepsilon_{n+1} = 1$.

Dans chacun des cas, on applique le même raisonnement que celui du cas de base.

On a donc défini une valuation ν telle que pour tout $i \in \mathbb{N}$, $\nu(x_i) = \varepsilon_i$. Reste à prouver que ν satisfait A .

Soit $F \in A$. Soit $n \in \mathbb{N}$ tel que les variables de F sont dans $\{x_0, \dots, x_n\}$. D'après l'hypothèse de récurrence, il existe une valuation ν_F qui satisfait $\{F\}$ et telle que pour tout $i \leq n$, on ait $\nu_F(x_i) = \varepsilon_i$.

Comme ν_F et ν coïncident sur $\{x_i, i \leq n\}$, alors $\nu_F(F) = \nu(F)$, donc ν satisfait F . □

Définition :

On dit qu'un ensemble ordonné (X, R) est inductif si pour tout sous-ensemble $Y \subseteq X$, si Y est totalement ordonné par R , alors Y admet un majorant dans X .

Lemme de Zorn :

Si (X, R) est un ensemble ordonné inductif non vide, alors X admet au moins un élément maximal.

Théorème :

Les trois énoncés suivants sont équivalents :

1. Lemme de Zorn ;
2. Axiome du choix : tout produit d'ensembles non-vides est non-vidé ;
3. Axiome du bon ordre : tout ensemble admet un bon ordre.

DÉMONSTRATION DU THÉORÈME DE COMPACTITÉ DANS LE CAS GÉNÉRAL :

Soit \mathcal{P} un ensemble de variables. Soit A un ensemble de formules sur \mathcal{P} finiment satisfiable.

$$\text{Posons } E = \left\{ \phi \in \bigcup_{X \subseteq \mathcal{P}} \{0, 1\}^X, \forall B \subseteq A \text{ fini, } \exists \hat{\phi} \text{ tel que } \hat{\phi}(B) \text{ et } \hat{\phi} \text{ prolonge } \phi \right\}.$$

Montrons que E est inductif, et qu'un de ses éléments maximaux satisfait A .

On munit E de l'ordre \sqsubseteq tel que pour tous $(\phi, \psi) \in E^2$, $\phi \sqsubseteq \psi$ ssi ψ est un prolongement de ϕ . En particulier, si $\phi \sqsubseteq \psi$, alors $\text{dom } \phi \subseteq \text{dom } \psi$.

Soit $Y \subseteq E$ totalement ordonné. On définit la fonction λ par :

1. $\text{dom } \lambda = \bigcup_{\phi \in Y} \text{dom } \phi$
2. Pour $x \in \text{dom } \lambda$, soit $\phi \in Y$ tel que $x \in \text{dom } \phi$, on pose alors $\lambda(x) = \phi(x)$.

Le choix du ϕ n'influe pas sur la définition de λ . En effet, si $\psi \in Y$ est tel que $x \in \text{dom } \psi$, on peut supposer sans perte de généralité que $\psi \sqsubseteq \phi$, et dans ce cas $\phi(x) = \psi(x)$.

Pour tout $\phi \in Y$ on a bien $\phi \sqsubseteq \lambda$, puisque :

1. $\text{dom } \phi \subseteq \text{dom } \lambda$
2. Pour tout $x \in \text{dom } \phi$, $\lambda(x) = \phi(x)$.

Ainsi λ est un majorant de E . Reste à prouver que $\lambda \in E$.

Soit $B \subseteq A$ fini. Soient $\{x_0, \dots, x_n\}$ les variables apparaissant dans B et qui sont dans $\text{dom } \lambda$.

Il existe $(\phi_i) \in Y^{n+1}$ tels que pour tout $i \leq n$, $x_i \in \text{dom } \phi_i$. Puisque Y est totalement ordonné, on peut supposer $\phi_0 \sqsubseteq \dots \sqsubseteq \phi_n$. Alors $\{x_0, \dots, x_n\} \subseteq \text{dom } \phi_n$.

Comme $\phi_n \in E$, il existe $\hat{\phi}_n$ qui prolonge ϕ_n et qui satisfait B . Cherchons un prolongement $\hat{\lambda}$ de λ à \mathcal{P} qui satisfasse B . On définit $\hat{\lambda}$ par :

1. Si $x \in \mathcal{P}$ apparaît dans B , alors $\hat{\lambda}(x) \equiv \hat{\phi}_n(x)$;
2. Si $x \in \mathcal{P}$ n'apparaît pas dans B mais $x \in \text{dom } \lambda$, alors $\hat{\lambda}(x) \equiv \lambda(x)$;
3. Sinon, on s'en tamponne, et on pose par exemple $\hat{\lambda}(x) \equiv \hat{\phi}_n(x)$.

Ainsi, $\hat{\lambda}$ satisfait B et prolonge λ .

Donc E est inductif. Par ailleurs, E n'est pas vide car la fonction de domaine nul appartient à E , puisque A finiment satisfiable.

Par le lemme de Zorn, on exhibe $\omega \in E$ un élément maximal de E . Reste à montrer que $\text{dom } \omega = \mathcal{P}$.

Par l'absurde, supposons $\text{dom } \omega \neq \mathcal{P}$. Soit $x \in \mathcal{P} \setminus \text{dom } \omega$. De deux choses l'une.

1. Soit pour tout $B \subseteq A$ fini, il existe $\hat{\omega} \in \{0, 1\}^{\text{dom } \omega \cup \{x\}}$ qui prolonge ω tel que $\hat{\omega}(x) = 0$. On pose alors $\hat{\omega}(x) = 0$.
2. Soit il existe $\hat{B} \subseteq A$ fini, tel que pour tout $\hat{\omega} \in \{0, 1\}^{\text{dom } \omega \cup \{x\}}$, si $\hat{\omega}$ prolonge ω , alors $\hat{\omega} = 1$. On pose alors $\hat{\omega} = 1$.

Pour $x \in \text{dom } \omega$, on pose par ailleurs $\hat{\omega}(x) = \omega(x)$.

Alors on a clairement $\hat{\omega} \in E$, mais dans les deux cas $\omega \sqsubset \hat{\omega}$, donc ω n'est pas maximal. C'est donc absurde.

□

Proposition :

Le théorème de complétude implique celui de compacité.

DÉMONSTRATION :

Soit A un ensemble de formules non satisfiables. On va montrer que A n'est pas finiment satisfiable.

La formule $x \wedge \neg x$ est une conséquence logique de A . Donc $x \wedge \neg x$ admet une preuve à partir de A

Cette preuve n'utilise qu'un nombre fini de formules de A , notons-les B . Par conséquent, B n'est pas satisfiable, et A n'est pas finiment satisfiable. \square

Proposition :

Le théorème de compacité implique celui de complétude.

DÉMONSTRATION :

Cf. Buscarel. \square

3.2 Calcul des prédicats

3.2.1 Syntaxe

Définition :

Un langage \mathcal{L} est un ensemble formé :

- d'un ensemble de relations \mathcal{R} , munies de leur arité ;
- d'un ensemble de symboles de fonctions \mathcal{F} , munies de leur arité ;
- d'un ensemble de symboles de constantes C .

On dit qu'un langage est égalitaire s'il contient l'égalité dans ses relations.

Définition :

L'ensemble des termes T est le plus petit ensemble contenant C , un ensemble de variables V et tel que pour tout $k \in \mathbb{N}$, si $(t_1, \dots, t_k) \in T^k$ et $f \in \mathcal{F}^{(k)}$ alors $f(t_1, \dots, t_k) \in T$.

Un terme est dit clos s'il ne contient pas de variables.

Définition :

Une formule atomique est un mot de la forme $R(t_1, \dots, t_k)$ où $R \in \mathcal{R}^{(k)}$ et $(t_1, \dots, t_k) \in T^k$.

L'ensemble des formules sur \mathcal{L} est le plus petit ensemble contenant les formules atomiques, et tel que pour tous F_1, F_2 formules et $x_0 \in V$, les mots suivants sont des formules : $\neg F_1, F_1 \wedge F_2, F_1 \vee F_2, \forall x_0. F_1, \exists x_0. F_1$.

Exemple :

Pour $\mathcal{L} = \{O, S, +, \times, =, \leq\}$:

- termes : SSO, Sx_0 .
- formules atomiques : $SSO \leq SSSx_0$.
- formules : $(SO \leq x_0) \wedge \exists x_1. (Sx_1 = SSO)$.

Théorème de lecture unique :

À écrire et à prouver pour les termes et les formules.

Définition :

Une occurrence d'une variable $x \in V$ dans la formule F est une de ses apparitions.

Par induction on définit les occurrences libres ou liées :

- dans une formule atomique, toutes les occurrences sont libres ;
- si une occurrence est libre (resp. liée) dans F , alors elle est libre (resp. liée) dans $\neg F, F \wedge G, G \wedge F, F \vee G$ et $G \vee F$.
- une occurrence de x est libre dans $\exists y. F$ ssi $x \neq y$ et x est libre dans F . Sinon, elle est liée.

Remarque :

On se ramène toujours à des formules dans lesquelles une variable n'a pas à la fois des occurrences libres et liées.

3.2.2 Sémantique

Définition :

Étant donné un langage \mathcal{L} , une \mathcal{L} -structure \mathcal{M} est constituée :

- d'un ensemble non vide de base M ;
- d'une relation $R^{\mathcal{M}} \subseteq M^n$ pour tout $R \in \mathcal{R}^{(n)}$;
- d'une fonction $f^{\mathcal{M}} : M^n \rightarrow M$ pour tout $f \in \mathcal{F}^{(n)}$;
- d'une constante $c^{\mathcal{M}} \in M$ pour tout $c \in C$.

Exemple :

- Pour $\mathcal{L} = \{R\}$, une \mathcal{L} -structure est un graphe orienté.
- Pour $\mathcal{L} = \{f_1^{(1)}, f_2^{(2)}, c\}$, une \mathcal{L} -structure est le langage des groupes : $\mathcal{M} = \langle G, i, \cdot, e \rangle$.
- Pour $\mathcal{L} = \{f_1^{(1)}, f_2^{(2)}, f_3^{(2)}, c_1, c_2\}$, une \mathcal{L} -structure est le langage des anneaux : $\mathcal{M} = \langle A, -, +, \times, 0, 1 \rangle$.
- Pour $\mathcal{L} = \{f_1^{(1)}, f_2^{(2)}, f_3^{(2)}, c\}$, une \mathcal{L} -structure est le langage de l'arithmétique : $\mathcal{M} = \langle \mathbb{N}, S, +, \times, O \rangle$.

Définition :

Soit \mathcal{M} une \mathcal{L} -structure. On définit par induction sur l'ensemble des termes l'interprétation en $(a_0, \dots, a_n) \in M^{n+1}$ d'un terme t dont les variables sont dans $(x_0, \dots, x_n) \in V^{n+1}$, notée $t^{\mathcal{M}}(x_0 \mapsto a_0, \dots, x_n \mapsto a_n) \in M$ ainsi :

- si $t = c \in C$, alors $t^{\mathcal{M}}(x_0 \mapsto a_0, \dots, x_n \mapsto a_n) = c^{\mathcal{M}}$;
- si $t = x_i$ pour $0 \leq i \leq n$, alors $t^{\mathcal{M}}(x_0 \mapsto a_0, \dots, x_n \mapsto a_n) = a_i$;
- si $t = f(t_1, \dots, t_k)$, alors $t^{\mathcal{M}}(x_0 \mapsto a_0, \dots, x_n \mapsto a_n) = f^{\mathcal{M}}(t_1^{\mathcal{M}}(x_0 \mapsto a_0, \dots, x_n \mapsto a_n), \dots, t_k^{\mathcal{M}}(x_0 \mapsto a_0, \dots, x_n \mapsto a_n))$.

Exemple :

Pour $t = f_2(f_1(x_0), f_1(f_1(C)))$ en 1 dans le langage de l'arithmétique, $t^{\mathcal{M}}(x_0 \mapsto 1) = 4$.

Définition :

On définit par induction sur l'ensemble des formules la satisfaction d'une formule $F(x_0, \dots, x_n)$ dont les variables libres sont dans $\{x_0, \dots, x_n\}$ pour $(a_0, \dots, a_n) \in M^{n+1}$, notée $\mathcal{M} \models F(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$ ainsi :

- si F atomique, $F = R(t_1, \dots, t_k)$, alors $\mathcal{M} \models F(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$ ssi $(t_1^{\mathcal{M}}(x_0 \mapsto a_0, \dots, x_n \mapsto a_n), \dots, t_k^{\mathcal{M}}(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)) \in R^{\mathcal{M}}$;
- si $F = F_1 \wedge F_2$, alors $\mathcal{M} \models F(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$ ssi $\mathcal{M} \models F_1(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$ et $\mathcal{M} \models F_2(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$;
- si $F = F_1 \vee F_2$, alors $\mathcal{M} \models F(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$ ssi $\mathcal{M} \models F_1(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$ ou $\mathcal{M} \models F_2(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$;
- si $F = \neg F'$, alors $\mathcal{M} \models F(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$ ssi $\mathcal{M} \not\models F'(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$;
- si $F = \exists x.F'$, alors $\mathcal{M} \models F(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$ ssi il existe $b \in M$ tel que $\mathcal{M} \models F'(x_0 \mapsto a_0, \dots, x_n \mapsto a_n, x \mapsto b)$;
- si $F = \forall x.F'$, alors $\mathcal{M} \models F(x_0 \mapsto a_0, \dots, x_n \mapsto a_n)$ ssi pour tout $b \in M$, $\mathcal{M} \models F(x_0 \mapsto a_0, \dots, x_n \mapsto a_n, x \mapsto b)$.

Exemple :

Pour $\mathcal{L} = \{0, S, +, \times, \leq\}$ dans $\mathcal{M}_1 = \langle \mathbb{N}, 0, S, +, \times, \leq \rangle$ et $\mathcal{M}_2 = \langle \mathbb{Z}, 0, S, +, \times, \leq \rangle$, avec $F(x) = \exists y.(x + y \leq 0)$.

On a $\mathcal{M}_1 \models F(0)$ mais $\mathcal{M}_1 \not\models F(2)$.

D'autre part $\mathcal{M}_2 \models F(0)$ et $\mathcal{M}_2 \models F(2)$.

Remarque :

Si F close (sans variable libre), alors la satisfaction de F dans \mathcal{M} ne dépend d'aucun n -uplet.

On écrit alors $\mathcal{M} \models F$ ou $\mathcal{M} \not\models F$.

Définition :

- On appelle énoncé toute formule close.
- Soit Σ un ensemble d'énoncés. On dit que \mathcal{M} satisfait Σ (noté $\mathcal{M} \models \Sigma$) si pour tout $F \in \Sigma$, on a $\mathcal{M} \models F$.
- Une formule F est conséquence sémantique d'un ensemble d'énoncés Σ si pour tout \mathcal{M} , si $\mathcal{M} \models \Sigma$ alors $\mathcal{M} \models F$.
On note alors $\Sigma \vdash F$.

Définition :

- Un ensemble d'énoncés Σ est satisfiable s'il existe \mathcal{M} tel que $\mathcal{M} \models \Sigma$.

- Un ensemble d'énoncés Σ est finiment satisfiable si pour tout $\Sigma' \subseteq \Sigma$ fini, Σ' est satisfiable.

Théorème de compacité :

Pour tout ensemble d'énoncés Σ , Σ est satisfiable ssi Σ finiment satisfiable.

Remarque :

- Certaines formules ne sont pas exprimables dans le système étudié actuellement (cf. second ordre).
- Si $\Sigma \not\vdash F$ et $\Sigma \not\vdash \neg F$, on dit que F est indécidable dans Σ .

Exemple :

L'axiome du choix (C) est indécidable dans Zermelo-Fraenkel (ZF).
L'hypothèse du continu (HC) selon laquelle $2^{\aleph_0} = \aleph_1$ est indécidable dans ZFC.

3.2.3 Théorème de complétude

Définition :

- (**Ax 1**) Toute tautologie du calcul propositionnel génère une tautologie du calcul des prédicats, en substituant ses variables par des formules du calcul du prédicat.
- (**Ax 2**) Pour tout $\phi(v)$, on a : $\exists v.\phi(v) \Leftrightarrow \neg\forall v.\neg\phi(v)$
- (**Ax 3**) Pour tous $\phi(v)$ et θ clos, on a : $(\forall v.(\theta \Rightarrow \phi(v))) \Rightarrow (\theta \Rightarrow \forall v.\phi(v))$
- (**Ax 4**) Pour tous $\phi(v)$ et t clos, on a : $\forall v.\phi(v) \Rightarrow \phi(t)$.
- (**Modus ponens**) Pour tous ϕ et ψ formules, de $\{\phi, \phi \Rightarrow \psi\}$ on déduit ψ .
- (**Généralisation**) Pour tout $\phi(v)$, si c est un symbole de constante qui n'apparaît pas dans $\phi(v)$, alors de $\phi(c)$ on déduit $\forall v.\phi(v)$.

Définition :

Soit Σ un ensemble d'énoncés de \mathcal{L} et ψ un énoncé.

On dit que ψ est une conséquence syntaxique (ou formelle) de Σ , ce que l'on note $\Sigma \Vdash \psi$, s'il existe une démonstration formelle de ψ à partir de Σ , i.e. s'il existe une suite finie $(\psi_i)_{i \leq k}$ d'énoncés de $\mathcal{L} \cup \tilde{C}$, où \tilde{C} est un ensemble fini de constantes n'apparaissant pas dans Σ , telle que :

- $\psi = \psi_k$;
- Pour tout $i \leq k$, l'une des conditions suivantes est remplie :
 1. $\psi_i \in \Sigma$;
 2. ψ_i est un axiome ;
 3. il existe $m, l < i$ tels que $\psi_l = \psi_m \Rightarrow \psi_i$;
 4. ψ_i est de la forme $\forall v.\phi(v)$, et il existe $m < i$ tel que $\psi(m) = \phi(c)$, où $c \in \tilde{C}$ n'apparaît ni dans $\phi(v)$ ni dans Σ .

Définition :

On dit que Σ est contradictoire s'il existe θ tel que $\Sigma \Vdash \theta$ et $\Sigma \Vdash \neg\theta$. Dans le cas contraire, on dit que Σ est cohérent.

Remarque :

Si Σ est contradictoire, alors pour tout ϕ , $\Sigma \Vdash \phi$. En effet, il suffit d'appliquer la tautologie du calcul propositionnel suivante : $A \Rightarrow \neg A \Rightarrow B$.

Remarque :

Si Σ est contradictoire, alors Σ n'admet pas de modèle.

En effet, supposons que Σ admette un modèle \mathcal{M} , alors pour tout énoncé θ , de deux choses l'une, soit $\mathcal{M} \models \theta$, soit $\mathcal{M} \not\models \theta$. Or d'après le théorème de complétude qui suit, si $\Sigma \Vdash \theta$, alors $\Sigma \vdash \theta$. Or $\mathcal{M} \models \Sigma$ donc $\mathcal{M} \models \theta$. De même, si $\Sigma \Vdash \neg\theta$, on tire $\mathcal{M} \not\models \theta$: absurde.

Théorème de complétude (1) :

Soit \mathcal{L} un langage, Σ un ensemble d'énoncés et θ un énoncé sur \mathcal{L} . Si $\Sigma \Vdash \theta$, alors $\Sigma \vdash \theta$.

Théorème de complétude (2) :

Soit \mathcal{L} un langage et Σ un ensemble d'énoncés sur \mathcal{L} . Si Σ est cohérent, alors Σ admet un modèle.

Proposition :

Les deux théorèmes sont équivalents.

DÉMONSTRATION :

Reste à prouver que (2) implique (1).

Lemme de déduction :

Si $\Sigma \cup \{\psi\} \vdash \theta$, alors $\Sigma \vdash \psi \Rightarrow \theta$.

Soit Σ cohérent. Supposons $\Sigma \vdash \theta$, alors $\Sigma \not\vdash \neg\theta$.

Par conséquent, $\Sigma \cup \{\neg\theta\}$ n'a pas de modèle. D'après le théorème (2), on tire que $\Sigma \cup \{\neg\theta\}$ n'est pas cohérent. En particulier, $\Sigma \cup \{\neg\theta\} \Vdash \theta$, d'où par le lemme $\Sigma \Vdash \neg\theta \Rightarrow \theta$.

Comme $(\neg A \Rightarrow A) \Rightarrow A$ est une tautologie du calcul propositionnel, on en conclut que $\Sigma \Vdash \theta$. \square

DÉMONSTRATION DU THÉORÈME DE COMPACTITÉ :

Soit Σ non satisfiable. En particulier, $\Sigma \Vdash (\theta \wedge \neg\theta)$ pour un certain θ . Or la preuve $\Sigma \Vdash (\theta \wedge \neg\theta)$ fait intervenir un nombre d'hypothèses de Σ fini. Soient Σ' ces hypothèses.

Alors $\Sigma' \Vdash (\theta \wedge \neg\theta)$, donc $\Sigma' \vdash (\theta \wedge \neg\theta)$ par le théorème de complétude. Donc Σ' est contradictoire, et non satisfiable.

Ainsi Σ n'est pas finiment satisfiable.

\square

3.3 Théorèmes d'incomplétude de Gödel

3.3.1 L'arithmétique avec l'exponentielle : le théorème de Tarski

On pose : $\mathcal{L}_E = \{0, S, +, *, E, \leq, =\}$

On définit les termes, les formules atomiques et les formules.

On va coder les symboles de \mathcal{L}_E et les connecteurs logiques.

Les formules sont des mots sur $\mathcal{L}_E \cup \{\wedge, \vee, \rightarrow, \forall, \exists, \neg\} \cup \{v_0, v_1, \dots\}$.

On va utiliser les lettres $0, S, (,), f, v, ', \neg, \rightarrow, \forall, = :$

- $v_0 = v, v_1 = v', v_x = v''' \dots$

- $+$: f

- $*$: f'

- E : f''

Exercice :

Ecrire la conjecture des nombres premiers jumeaux :

$$\text{Premier}(v) = \forall v' \forall v'' (v = v' f' v'' \Rightarrow \neg(v' = S0) \Rightarrow v'' = S0)$$

$$\forall v \exists v' (v \leq v' \wedge \text{Premier}(v') \wedge \text{Premier}(SSv'))$$

Notation :

Si $n \in \mathbb{N}$, on note \bar{n} le mot $0SSS \dots S = 0S^n$, et v_i le mot $v''' \dots = v'^i$

Définition :

Si $F(v_i)$ est une formule avec une seule variable libre v_i , on notera $F(\bar{n})$ le mot obtenu en remplaçant chaque occurrence libre de v_i par \bar{n} .

De même pour $F(v_1, v_2, \dots, v_k)$ et $F(\bar{n}_1, \bar{n}_2, \dots, \bar{v}_k)$.

A faire

(Manque un tout petit quelque chose...)

Définition :

- Deux formules closes F et G sont arithmétiquement équivalentes si elles sont simultanément vraies ou fausses dans $(\mathbb{N}, (0, S, +, *, E, \leq, =))$.
- Deux formules $F(v_1, v_2, \dots, v_k), G(v_1, v_2, \dots, v_k)$ avec les mêmes variables libres sont arithmétiquement équivalentes si pour tous $(n_1, \dots, n_k) \in \mathbb{N}^k, F(\bar{n}_1, \bar{n}_2, \dots, \bar{v}_k)$ et $G(\bar{n}_1, \bar{n}_2, \dots, \bar{v}_k)$ sont arithmétiquement équivalentes.

Définition :

- une formule $F(v_i)$ exprime(représente) l'ensemble des entiers n tels que $F(\bar{n})$ est vrai.
- Une formule $F(v_1, v_2, \dots, v_k)$ représente $\{(n_1, \dots, n_k) \in \mathbb{N}^k / \mathbb{N} \models F(\bar{n}_1, \bar{n}_2, \dots, \bar{v}_k)\}$.
- Une formule $F(n_1, n_2, \dots, n_k)$ représente la relation $R(x_1, \dots, x_k)$ si $\{(n_1, \dots, n_k) \in \mathbb{N}^k / R(n_1, \dots, n_k)\} = \{(n_1, \dots, n_k) \in \mathbb{N}^k / F(\bar{n}_1, \bar{n}_2, \dots, \bar{v}_k)\}$.
- Une propriété P est représentée par la formule $F(v_1, v_2, \dots, v_k)$ si un k -uplet (n_1, \dots, n_k) a la propriété P si et seulement si $\mathbb{N} \models F(\bar{n}_1, \bar{n}_2, \dots, \bar{v}_k)$.

Définition :

- Un ensemble (resp. une propriété, une relation) est E -arithmétique s'il existe une formule F dans \mathcal{L}_E qui le (resp. la) représente.
- Un ensemble (resp. une propriété, une relation) est arithmétique s'il existe une formule F dans $\mathcal{L}_E \setminus E$ qui le (resp. la) représente.

Remarque :

On montrera que x^y est définissable, et donc que arithmétique est équivalent à E -arithmétique.

Définition :

Une fonction f est E -arithmétique (resp. arithmétique) si et seulement si la relation $y = f(x_1, \dots, x_n)$ l'est ;

Définition :

Numérotation de Gödel : La puissance sur le i ème nombre premier code le i ème caractère.

0	s	()	f	l	v	\neg	\Rightarrow	\forall	$=$	\leq	$\#$
1	0	2	3	4	5	6	7	8	9	η	ε	δ

Exemple :

$OSvf$ est codé par $2^1 3^0 5^6 7^4$.
 \bar{n} est codé par 13^n .

Définition :

Concaténation en base $b, b \geq 2$
 Si n et m sont deux entiers et $l_b(n)$ la longueur de l'écriture en base b de n , alors $m *_b n = mb^{l_b(n)} + n$.

Exemple :

$53 *_1 0792 = 53792$
 $cinq *_2 trois = \overline{10111}^2 = 23$

Théorème :

Pour tout $b \geq 2$, la relation $x *_b y = z$ est E -arithmétique.

DÉMONSTRATION :

- $Pow_b(x)$ est la relation $\exists y (x = b^y)$.
- la relation $y = b^{l_b(x)}$ est E -arithmétique.
 En effet, $S(x, y) = Pow_b(y) \wedge (x < y) \wedge \forall z ((x < z \wedge Pow_b(z)) \Rightarrow y \leq z)$ (y est la plus petite puissance de b plus grande que x) est E -arithmétique, et $y = b^{l_b(x)}$ est $(x = 0 \wedge y = b) \vee (x \neq 0 \wedge S(x, y))$.
- $x *_b y = z$ est $x b^{l_b(y)} + y = z$
i.e. $\exists z_1 \exists z_2 (b^{l_b(y)} = z_1 \wedge x *_b z_1 = z_2 \wedge z_2 + y = z)$

□

Remarque :

$*_b$ n'est pas associative : $(5 *_b 0) *_b 3 = 503 \neq 53 = 5 *_b (0 *_b 3)$

Corollaire :

Pour tout $n \geq 2$ $x_1 *_b (x_2 *_b (\dots *_b x_n) \dots)$ est E -arithmétique.

Notation :

on appelle expression tout mot de $\mathcal{L}_E \cup \{\#\}$ qui ne commence pas par S .

A faire

(La notation qui suit est-elle bien placée ? cohérence avec le reste ?)

Notation :

\mathcal{E}_x sera l'expression dont le numéro de Gödel est x . Pour toute sous-expression X de \mathcal{L}_E , on notera $[X]$ son numéro de Gödel.

Proposition :

1. Il existe une fonction E -arithmétique $x \circ y$ telle que pour toutes expressions X et Y de numéros x et y , le numéro de XY est $x \circ y$.
2. Le numéro de Gödel de \bar{n} est une fonction E -arithmétique de n .

DÉMONSTRATION :

1. C'est $*_{13}$.
2. La relation $f(x) = y$, avec $f(x) = 13^x$ est représentée par $F(v_1, v_2) : v_2 = 0S^n E v_1$.

□

Définition :

On pose $T = \{n/n \text{ est le nombre de Gödel d'un énoncé de } \mathcal{L}_E \text{ vrai dans } \mathbb{N}\}$.

Théorème de Tarski :

T n'est pas E -arithmétique.

Définition :

Un énoncé de Gödel pour un ensemble $A \subseteq \mathbb{N}$ est un énoncé X tel que X est faux dans \mathbb{N} si et seulement si son numéro n'est pas dans A .

Théorème :

Tout ensemble E -arithmétique possède un énoncé de Gödel.

DÉMONSTRATION : CE THÉORÈME IMPLIQUE CELUI DE TARSKI :

Supposons T E -arithmétique.

Alors $\tilde{T} = \{n \in \mathbb{N}/n \text{ est le nombre de Gödel d'un énoncé de } \mathcal{L}_E \text{ faux dans } \mathbb{N}\}$ l'est aussi :

$$n \in \tilde{T} \iff 7 *_b n \in T$$

D'après le théorème précédent, il existe un énoncé de Gödel X de \tilde{T} .

$[X] \in \tilde{T} \iff X$ est faux dans \mathbb{N} , par définition de \tilde{T} . Or X est faux $\iff [X] \notin \tilde{T}$, ce qui mène à une contradiction. □

Notation :

Pour une expression ξ , on note $\xi[\bar{n}]$ l'expression $\forall v_1 (v_1 = \bar{n} \Rightarrow \xi)$.

Lemme :

Si ξ est la formule $F(v_1)$, alors $F(\bar{n})$ et $\xi[\bar{n}]$ sont équivalentes. **A faire** (ce n'est pas ce qu'il y a dans le cours... est-ce bien celà ?)

Notation :

Si ξ est de numéro e , on note $r(e, n)$ le numéro de $E[\bar{n}]$.

Lemme :

r est E -arithmétique.

DÉMONSTRATION :

Si ξ_x est l'expression de numéro x , alors $r(x, y) = z$ est le numéro de $\forall v' (v' = \bar{y} \Rightarrow \xi_x)$.

Donc $r(s, y) = z$ est : $\exists w (w = 13^y \wedge z = 9652652\eta * w * 8 * x * 3)$ \square

Définition :

Si $A \subseteq \mathbb{N}$, on note $A^* = \{n \in \mathbb{N} / r(n, n) \in A\}$.

Lemme :

Si A est E -arithmétique, alors A^* l'est aussi.

DÉMONSTRATION : EXISTENCE D'ÉNONCÉS DE GÖDEL :

Soit A un ensemble E -arithmétique. Donc A^* l'est aussi. Soit $H(v_1)$ représentant A^* , et h son numéro de Gödel.

$H[\bar{h}]$ est vrai dans \mathbb{N} si et seulement si $r(h, h) \in A$, et $r(h, h)$ est bien le numéro de Gödel de $H[\bar{h}]$: c'est un énoncé de Gödel pour A . \square

3.3.2 Incomplétude de l'arithmétique de Péano avec exponentielle**Notation :**

On note dans la suite F, G, H des formules, v_i, v_j des variables et t un terme quelconques.

Groupe 1 : les schémas d'axiomes de la logique propositionnelle.

$$\mathbf{L1} : F \Rightarrow (G \Rightarrow F)$$

$$\mathbf{L2} : (F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H))$$

$$\mathbf{L3} : ((\neg F \Rightarrow \neg G) \Rightarrow (G \Rightarrow F))$$

Groupe 2 : les schémas d'axiomes pour l'égalité.

$$\mathbf{L4} : (\forall v_i (F \Rightarrow G) \Rightarrow (\forall v_i F \Rightarrow \forall v_i G))$$

$$\mathbf{L5} : (F \Rightarrow \forall v_i F) \text{ si } v_i \notin \text{FVar}(F)$$

$$\mathbf{L6} : \exists v_i (v_i = t) \text{ si } v_i \notin \text{Var}(t)$$

$$\mathbf{L7} : (v_i = t \Rightarrow (X_1 v_i X_2 \Rightarrow X_1 t X_2)) \text{ avec } (X_1 v_i X_2) \text{ une formule atomique}$$

Groupe 3 : les axiomes arithmétiques.

$$\mathbf{N1} : (v_1 s = v_2 s \Rightarrow v_1 = v_2)$$

$$\mathbf{N2} : \neg(0 = v_1 s)$$

$$\mathbf{N3} : v_1 + 0 = v_1$$

$$\mathbf{N4} : v_1 + v_2 s = (v_1 + v_2) s$$

$$\mathbf{N5} : v_1 \times 0 = 0$$

$$\mathbf{N6} : v_1 \times (v_2 s) = (v_1 \cdot v_2) + v_1$$

$$\mathbf{N7} : v_1 \leq 0 \Leftrightarrow v_1 = 0$$

$$\mathbf{N8} : (v_1 \leq v_2 s \Rightarrow (v_1 = v_2 s \vee v_1 \leq v_2))$$

$$\mathbf{N9} : (v_1 \leq v_2 \vee v_2 \leq v_1)$$

$$\mathbf{N10} : v_1 \text{ E } 0 = 0s$$

$$\mathbf{N11} : v_1 \text{ E } (v_2 s) = (v_1 \text{ E } v_2) \times v_1$$

Groupe 4 : le schéma d'induction.

Si $F(v_1)$ est une formule qui peut avoir d'autres variables libres que v_1 on note $F[v_1 s]$ la formule suivante équivalente à $F(v_1 s)$, avec $v_i \notin \text{FVar}(F)$:

$$\forall v_i (v_i = v_1 s \Rightarrow \forall v_1 (v_1 = v_i \Rightarrow F))$$

N12 : $(F[0] \Rightarrow (\forall v_1 (F(v_1) \Rightarrow F[v_1s]) \Rightarrow \forall v_1 F(v_1)))$

Règles de déduction : utilisées dans les preuves.

1 : Par le *modus ponens* de F et $(F \Rightarrow G)$ on déduit G

2 : Par la généralisation de F on déduit $\forall v_i F$

Définition :

Une preuve dans \mathcal{P}_E (Peano avec exponentielle) est une suite de formules telle que chaque formule est soit un axiome, soit se déduit d'une des règles de déduction de formules la précédant dans la suite.

Une formule est prouvable dans \mathcal{P}_E s'il existe une preuve dont la formule est un élément.

Une formule est réfutable si sa négation est prouvable.

Proposition :

L'ensemble P_E des numéros de Gödel des formules prouvables dans \mathcal{P}_E est E-arithmétique.

Rappels :

0	s	()	f	l	v	¬	⇒	∀	=	≤	#
1	0	2	3	4	5	6	7	8	9	η	ε	δ

Lemme :

Les prédicats suivants sont E-arithmétiques.

- $x \star_b y = z$;
- $x_1 \star_b x_2 \star_b \dots \star_b x_n = y$;
- $\text{Pow}_b(w)$: w est une puissance de b .

Notation :

Désormais $b = 13$ et xy remplace $x \star_b y$, à ne pas confondre avec le produit $x \times y$.

Définition :

On définit les relations suivantes :

1. On note $x \smile_b y$ si l'écriture en base b de x est un segment initial de celle de y .
2. On note $x \frown_b y$ si l'écriture en base b de x est un segment final de celle de y .
3. On note $x \asymp_b y \equiv (\exists z \leq y)(x \smile_b z \wedge z \frown_b y)$, i.e. l'écriture en base b de x est contenue dans celle de y .

Définition :

On notera K l'ensemble des entiers dont l'écriture en base 13 n'utilise pas le chiffre δ .

Définition :

Soient $a_1, \dots, a_n \in K$. Le nombre-suite de a_1, \dots, a_n est $\delta a_1 \delta \dots \delta a_n \delta$.

Définition :

On définit les relations suivantes :

1. $\text{Seq}(x)$ si x est un nombre-suite.
2. $x \in y$ si $\text{Seq}(y)$ et x est dans la suite.
3. $x \prec_z y$ si $\text{Seq}(z)$ et x et y sont dans la suite dans cet ordre.

Lemme :

Toutes les relations précédentes sont E-arithmétiques.

DÉMONSTRATION :

1. $x \frown y \equiv (\exists z \leq y)(zx = y) \vee (x = y)$
2. $x \smile y \equiv (x \neq 0 \wedge (\exists z \leq y)(\exists w \leq y)(\text{Pow}_{13}(w) \wedge (x \times w)z = y)) \vee (x = y)$

3. $x \asymp y \equiv (\exists z \leq y)(x \smile z \wedge z \smile y)$
4. $\text{Seq}(x) \equiv (\delta \smile x) \wedge (\delta \smile x) \wedge (\delta \delta \neq x) \wedge (\forall y \leq x)((\delta[0]y \asymp x) \Rightarrow (\delta \smile y))$
5. $x \in y \equiv \text{Seq}(x) \wedge (\delta x \delta \asymp y) \wedge (\delta \neq x)$
6. $x \prec_z y \equiv \text{Seq}(z) \wedge (x \in z) \wedge (y \in z) \wedge (\exists w \leq z)((w \smile z) \wedge (x \in w) \wedge (y \notin w))$

□

Lemme :

L'ensemble des $n \in \mathbb{N}$ tel que n est le numéro de Gödel d'un terme de \mathcal{L}_E est E-arithmétique.

Définition :

On définit la relation de formation des termes R_t :

$$R_t(X, Y, Z) \equiv (Z = (X[+]Y)) \vee (Z = X[\times]Y) \vee (Z = X[E]Y) \vee (Z = X[s])$$

Une suite de formation de termes est une suite X_1, \dots, X_n d'expressions telle que pour tout $i \leq n$, soit X_i est une variable, soit un entier, soit il existe $j, k < i$ tels que $R_t(X_j, X_k, X_i)$.

X est un terme s'il appartient à une suite de formation de termes.

Définition :

On définit de même la relation de formation des formules R_f :

$$R_f(X, Y, Z) \equiv (Z = [\neg]X) \vee (Z = X[\Rightarrow]Y) \vee (Z = [\forall v_i]X)$$

Une suite de formation de formules est une suite X_1, \dots, X_n d'expressions telle que pour tout $i \leq n$, soit X_i est une formule atomique, soit il existe $j, k < i$ tels que $R_f(X_j, X_k, X_i)$.

Définition :

Le nombre de Gödel d'une suite $(\mathcal{E}x_1, \dots, \mathcal{E}x_n)$ avec $x_1, \dots, x_n \in K$ est le nombre-suite de (x_1, \dots, x_n) , *i.e.* $[\#x_1\# \dots \#x_n\#]$.

Proposition :

Les conditions suivantes sont E-arithmétiques :

1. $\text{Sb}(x) : \mathcal{E}_x$ est une suite de ι .

$$\text{Sb}(x) \equiv (\forall y \leq x)(y \asymp x \Rightarrow [\iota] \asymp y)$$

2. $\text{IsVar}(x) : \mathcal{E}_x$ est une variable.

$$\text{IsVar}(x) \equiv (\exists y \leq x)((x = [v]y) \wedge \text{Sb}(y))$$

3. $\text{Num}(x) : \mathcal{E}_x$ est un entier.

$$\text{Num}(x) \equiv \text{Pow}_{13}(x)$$

4. $R_t(x, y, z) : R_t(\mathcal{E}_x, \mathcal{E}_y, \mathcal{E}_z)$

$$R_t(x, y, z) \equiv (z = x[+]y) \vee (z = x[\times]y) \vee (z = x[E]y) \vee (z = x[s])$$

5. $\text{Seq}_t(x) : \mathcal{E}_x$ est une suite de formation de termes.

$$\text{Seq}_t(x) \equiv \text{Seq}(x) \wedge (\exists y \in x)(\text{IsVar}(y) \vee \text{Num}(y) \vee (\exists z, w \prec_x y)(R_t(z, w, y)))$$

6. $\text{Term}(x) : \mathcal{E}_x$ est un terme.

$$\text{Term}(x) \equiv \exists y(\text{Seq}_t(y) \wedge (x \in y))$$

7. $\text{AF}(x) : \mathcal{E}_x$ est une formule atomique.

$$\text{AF}(x) \equiv (\exists y \leq x)(\exists z \leq x)(\text{Term}(y) \wedge \text{Term}(z) \wedge ((x = y[=]z) \vee (x = y[\leq]z)))$$

8. $\text{Gen}(x, y) : \text{il existe } v_i \text{ une variable telle que } \mathcal{E}_y = \forall v_i \mathcal{E}_x.$

9. $R_f(x, y, z) : R_f(\mathcal{E}_x, \mathcal{E}_y, \mathcal{E}_z)$

10. Seqf(x) : \mathcal{E}_x est une suite de formation de formules.
11. Form : \mathcal{E}_x est une formule de \mathcal{L}_E .
12. Ax : \mathcal{E}_x est un axiome de \mathcal{P}_E .
13. MP(x, y, z) : \mathcal{E}_z se prouve par *modus ponens* à partir de \mathcal{E}_x et \mathcal{E}_y .
14. Der(x, y) : pareil avec la deuxième règle d'inférence.
15. Proof(x) : \mathcal{E}_x est une preuve de \mathcal{P}_E .
16. Pe : \mathcal{E}_x est prouvable dans \mathcal{P}_E .

Théorème :

Théorème d'incomplétude de Gödel pour l'arithmétique de Peano avec exponentielle :
Le système \mathcal{P}_E est incomplet *i.e.* il y a des choses vraies dans \mathbb{N} non prouvables. DÉMONSTRATION :

A faire \square

3.3.3 Incomplétude de l'arithmétique de Péano sans exponentielle

Définition :

On note \mathcal{P} les axiomes de l'arithmétique qui n'ont pas E. On note P_A l'ensemble des numéros de Gödel des énoncés de $\mathcal{L}_E \setminus \{E\}$ prouvables dans \mathcal{P} .

Théorème d'incomplétude de Gödel :

Le système \mathcal{P} est incomplet.

Pour montrer ce résultat, on va se débarrasser de E.

Lemme :

Pour p premier, les prédicats suivants sont arithmétiques.

1. $\text{div}(x, y) \equiv x \mid y$
2. $\text{Pow}_p(x)$
3. $\text{MPow}_p(x, y) \equiv \left(y = \min_{n \in \mathbb{N}^*} \{p^n > x\} \right)$

DÉMONSTRATION :

1. $\text{div}(x, y) \equiv (\exists z \leq y)(y = x \times z)$
2. $\text{Pow}_p(x) \equiv (\forall y \leq x)(\text{div}(y, x) \Rightarrow (y \neq 1) \Rightarrow \text{div}(p, y))$
3. $\text{MPow}_p(x, y) \equiv \text{Pow}_p(y) \wedge (y > x) \wedge (y > p) \wedge (\forall z < y)(\neg \text{Pow}_p(z) \vee (z \leq x) \vee (z \leq p))$

\square

Lemme :

Pour p premier, le prédicat $x \star_p y = z$ est arithmétique.

DÉMONSTRATION :

$$(x \star_p y = z) \equiv (\exists w \leq z)(\text{MPow}_p(y, w) \wedge (z = w \times x + y)) \quad \square$$

Lemme :

Pour p premier, les prédicats $\simeq_p, \frown_p, \succsim_p$ et $x_1 \star_p \dots \star_p x_n = y$ sont arithmétiques.

DÉMONSTRATION :

On reprend les mêmes formules que dans le cas avec E : elle n'utilisaient pas E. \square

Proposition :

Le prédicat $x^y = z$ est arithmétique.

Pour montrer le lemme, on va transformer une récurrence en une suite $(0, 1)\#(1, x)\#\dots\#(y, x^y)$.

Lemme :

Il existe une relation κ telle que :

1. Pour toute suite $(a_i, b_i)_{i \leq n}$, il existe z tel que pour tous x, y , $\kappa(x, y, z)$ ssi il existe $i \leq n$ tel que $(x, y) = (a_i, b_i)$.
2. Pour tous x, y, z , si $\kappa(x, y, z)$ alors $x \leq z$ et $y \leq z$.

Définition :

Un cadre est un nombre de la forme $21 \dots 12$. Soit θ une suite $(a_i, b_i)_{i \leq n}$. Soit f un cadre plus long que tous ceux apparaissant dans θ . On associe à θ le nombre-suite $ffa_1fb_1ff \dots ffa_nfb_nff$.

On appelle cadre maximal de y tout nombre x tel que x est un cadre, $x \asymp y$ et c'est le plus long cadre de y . Dans ce cas, on note $x \text{ mf } y$.

Lemme :

Le prédicat mf est arithmétique.

DÉMONSTRATION :

Posons $I(x) : x$ est une suite de 1. Alors $I(x) \equiv (x \neq 0) \wedge (\forall y \leq x)(y \asymp x \Rightarrow 1 \asymp y)$. On exprime alors :

$$x \text{ mf } y \equiv (x \asymp y) \wedge (\exists z \leq x)(I(z) \wedge (x = 2z2) \wedge (\forall w \leq y)(I(w) \Rightarrow 2zw2 \neq y))$$

□

On a ainsi trouvé le κ cherché :

$$\kappa(x, y, z) \equiv (\exists f \leq z)((f \text{ mf } z) \wedge (ffxffyff \asymp z) \wedge (f \neq x) \wedge (f \neq y))$$

A faire

(pas fini!)**Définition :**

Une théorie est un ensemble d'énoncés.

On note $Th(T)$ l'ensemble des numéros de Gödel des théorèmes de T .

i.e. $F \in Th(T) \iff T \vdash F$ (*i.e.* F est vrai dans tous les modèles de T , d'après le théorème de complétude)

Définition :

On dit que T est décidable lorsque $Th(T)$ est récursif.

Proposition :

Si T est récursive, alors $Th(T)$ est récursivement énumérable.

DÉMONSTRATION :

Il suffit d'énumérer les preuves. □

Proposition :

Si T est complète, (*i.e.* Pour tout énoncé F , F est un théorème de T , ou $\neg F$ est un théorème de T) et récursive, alors T est décidable.

Théorème :

Soit \mathcal{L} un langage contenant le langage de l'arithmétique.

Si T est une théorie cohérente sur \mathcal{L} contenant P_0 , alors T est décidable.

DÉMONSTRATION :

cf. Cori-Lasca. □

Théorème de Church :

L'ensemble des théories d'un langage contenant celui de l'arithmétique n'est pas récursif.

DÉMONSTRATION :

Soit $T = \emptyset$ et G la conjonction des axiomes de P_0 .

$$G \vdash F \iff \emptyset \vdash (G \Rightarrow F)$$

Si l'ensemble des théorèmes était récursif, P_0 serait décidable, ce qui est faux d'après le théorème précédent. □

Théorème d'incomplétude de Gödel-Rosser :

Une théorie cohérente, récursive et contenant P_0 est incomplète.

DÉMONSTRATION :

Si elle était complète, elle serait décidable. \square

Théorème d'incomplétude de Gödel (2^e) :

Soit \mathcal{L} un langage contenant le langage de l'arithmétique et T une théorie cohérente sur \mathcal{L} , récursive et contenant P_0 .

Si $Coh(T)$ est : il n'existe pas de formule F et de preuve de $\neg F$ dans T , alors

$$T \not\vdash Coh(T)$$

3.3.4 Le problème de correspondance de Post (PCP)

Définition :

Données : un alphabet Σ

Un ensemble fini de couples de mots de $\Sigma \{(w_1, x_1), \dots, (w_k, x_k)\}$

Question : Est-ce qu'il existe une suite i_1, \dots, i_m avec $m \in \mathbb{N}^*$ et $i_2, \dots, i_m \in \{1, \dots, k\}$ tq $w_{i_1}w_{i_2} \dots w_{i_m} = x_{i_1}x_{i_2} \dots x_{i_m}$

PCP = codes binaires de $\Sigma, (w_1, x_1), \dots, (w_k, x_k)$ tq $\exists i_1, \dots, i_m$ et $w_{i_1}w_{i_2} \dots w_{i_m} = x_{i_1}x_{i_2} \dots x_{i_m}$

Exemple :

TODO

Définition :

On appelle solution partielle d'un pb PCP une suite i_1, \dots, i_m tq l'un des $w_{i_1}w_{i_2} \dots w_{i_m}$ et $x_{i_1}x_{i_2} \dots x_{i_m}$ soit un préfixe de l'autre.

Si i_1, \dots, i_m est une solution toute sous-suite préfixe d'elle est une solution partielle.

Exemple :

211 est une solution partielle

Théorème :

PCP est indécidable

Définition :

MPCP = $\Sigma, (w_1, x_1), \dots, (w_k, x_k)$ tq $\exists i_1, \dots, i_m$ et $w_{i_1}w_{i_2} \dots w_{i_m} = x_{i_1}x_{i_2} \dots x_{i_m}$

Lemme :

MPCP se réduit à PCP i.e. il existe une fonction calculable f telle que pour tout mot x codant une entrée de MPCP, $f(x)$ est le code d'une entrée de PCP et $f(x) \in PCP \Leftrightarrow x \in MPCP$. DÉMONSTRATION :

Soient $x = (\Sigma, (w_1, x_1), \dots, (w_k, x_k)) \in MPCP$ et $\star \notin \Sigma$

$f(x)$ est Σ

$\cup \{\star, \$\}$

$\cup w'_0 = \star a_1, \dots, \star a_n \star$ avec $w_1 = a_1, \dots, a_n$

$\cup x'_0 = \star b_1, \dots, \star b_n \star$ avec $x_1 = b_1, \dots, b_n$

et pour $1 \leq i \leq k$:

- $w'_i = a_1 \star a_2, \dots, \star a_n \star$ avec $w_i = a_1, \dots, a_n$

- $x'_i = \star b_1, \dots, \star b_n \star$ avec $x_i = b_1, \dots, b_n$

et :

- $w'_{k+1} = \$$

- $x'_{k+1} = \star \$$

i	w'	x'
0	$\star 1 \star$	$\star 1 \star 1 \star 1$
1	$1 \star$	$\star 1 \star 1 \star 1$
2	$1 \star 0 \star 1 \star 1 \star 1 \star$	$\star 1 \star 0$
3	$1 \star 0 \star$	$\star 0$
4	$\$$	$\star \$$

$x \in MPCP \Rightarrow f(x) \in PCP :$

Soit i_1, \dots, i_n une solution pour x .

On a $w_{i_1} w_{i_2} \dots w_{i_m} = x_{i_1} x_{i_2} \dots x_{i_m}$

On montre que $0i_1, \dots, i_n(k+1)$ est une solution pour $f(x) :$

$w'_0 w'_{i_1} w'_{i_2} \dots w'_{i_m} w'_{k+1} = x'_0 x'_{i_1} x'_{i_2} \dots x'_{i_m} x'_{k+1}$

Exemple :

$f(x) \in PCP \Rightarrow x \in MPCP :$

Soit $1_1, \dots, 1_m$ une solution de $f(x)$.

On a $i_1 = 0$ donc en enlevant les \star et les $\$$ de $w'_{i_1} w'_{i_2} \dots w'_{i_m}$ on obtient $w_{i_1} w_{i_2} \dots w_{i_m}$ avec pour convention $w_0 = w_1$ et $w_{k+1} = \Sigma$

Donc une solution pour x est la suite $1_1, \dots, 1_m$ dans laquelle on enlève les $(k+1)$ et on remplace 0 par 1. \square

Théorème :

$MPCP$ est indécidable.

Corollaire :

PCP est indécidable.

DÉMONSTRATION :

Soit L l'ensemble des codes de (M, w) où M est une machine avec un demi-ruban qui n'écrit jamais B et tel que $M(w)$ termine.

L est indécidable.

On va montrer que L se réduit en $MPCP :$

f tq $x \in L \Leftrightarrow f(x) \in MPCP$

x est $(Q, \Sigma, \Gamma, q_0, Q_f, \delta, B)$ et $w \in \Sigma^*$

Idée : dans le mot $w = \# \alpha_1, \dots, \# \alpha_n$ suite de config $x = \# \alpha_1, \dots, \# \alpha_n + 1$

Rappel d'un codage d'une config : xqy où on est dans l'état q avec le pointeur entre x et y .

TODO : règles

f est calculable.

Si $x \in L$ i.e. $M(w)$ termine alors $f(x) \in MPCP$

Si $f(x) \in MPCP$ alors toutes les solutions partielles sans état final sont de la forme $(\chi \# \alpha, \chi \# \alpha' \# \beta)$ où :

- χ est une suite de configurations correspondant au calcul
- α est le début du code d'une configuration
- α' est une configuration, la suivant dans le calcul dans χ
- β est le début du code d'un calcul

On montre ceci par récurrence sur la longueur de la solution partielle.

Si on a une solution c'est qu'on a un état final : donc $x \in L$. \square

3.4 Un exemple de théorie décidable : l'élimination des quanteurs dans les corps réels cols

3.4.1 De quoi parle t-on ?

$= \{0; 1; +; \cdot; \leq\}$

Définition :

Un corps réel cols est un corps ordonné qui vérifie pour les polynômes univariés le théorème des valeurs intermédiaires.

Les axiomes :

- Corps :
 - $0 \neq 1$
 - $\forall x \forall y : x + y = y + x$
 - $\forall x \exists y x + y = 0$
- Corps ordonné :
 - \leq relation d'ordre totale
 - $\forall x : x \leq x$
 - $\forall x \forall y : (x \leq y \vee y \leq x)$
 - $\forall x \forall y \forall z : ((x \leq y \wedge y \leq z) \Rightarrow (x \leq z))$
 - $\forall x \forall y : ((x \leq y \wedge y \leq x) \Rightarrow (x = y))$
 - $\forall x \forall y \forall z : ((x \leq y \Rightarrow x + z \leq y + z) \wedge ((x \leq y \wedge 0 \leq z) \Rightarrow xz \leq yz) \wedge ((x \leq y \wedge z \leq 0) \Rightarrow yz \leq xz))$
 - $\forall n \geq 1 : \forall a_0 \dots \forall a_n \forall x \forall y [(a_0, \dots, a_n x^n \geq 0 \wedge a_0, \dots, a_n y^n \leq 0) \Rightarrow \exists z (x \leq z \leq y, a_0, \dots, a_n z^n = 0)]$

Tous ces axiomes sont récursifs.

Exemple :

\mathbb{R} et $\mathbb{Q} \cap \mathbb{R}$

Définition :

Formules et termes de \mathcal{L} :

Les termes sont les polinômes à coefficients dans \mathbb{Z} .

Les formules atomiques sont les $P(\bar{x})\Delta Q(\bar{x})$ avec $\Delta \in \{\leq, \geq, =\}$ équivalents à une formules de la forme $P(\bar{x})\Delta 0$ avec $P \in \mathbb{Z}[\bar{X}]$.

Les formules sont équivalents à des formules de la forme $Q_1 x_1 \dots Q_n x_n (\vee_i (\wedge_j P_{ij}(\bar{x}\bar{y})\Delta_{ij} 0))$ avec $Q_i \in \{\exists, \forall\}$, $\Delta_{ij} \in \{\leq, \geq, =\}$, $P_{ij} \in \mathbb{Z}[\bar{X}\bar{Y}]$.

Exemple :

$$\phi(a, b, c) : \exists y (ay^2 + byc = 0)$$

$$\psi(a, b, c) : (((a \neq 0 \vee b \neq 0) \wedge (b^2 - 4ac \geq 0)) \vee (a = 0 \wedge b = 0 \wedge c = 0))$$

$$\forall a, b, c : (\psi(a, b, c) \Leftrightarrow \phi(a, b, c))$$

$$\exists y_1 \forall y_2 : (ay_1^2 y_2 + 3bay_2^3 + y_2^4 = 0 \wedge y_1 y_2^2 - y_2 \geq 0) ?$$

Définition :

Une théorie T' élimine les quanteurs si pour toute formule $\phi(x_1, \dots, x_n)$ il existe une formule $\psi(x_1, \dots, x_n)$ sans quanteur et telle que $T \vdash \forall x_1, \dots, x_n : (\phi(x_1, \dots, x_n) \Leftrightarrow \psi(x_1, \dots, x_n))$.

Théorème Le théorème de Tarski :

La théorie des CRC élimine les quanteurs. Elle est donc complète et décidable.

Remarque :

La complétude est une conséquence de l'élimination.

Une théorie complète et récursive est décidable.

Théorème :

La théorie des corps algébriques clos de caractéristique fixée élimine les quanteurs.

Dans \mathbb{C} , les ensembles définis avec des équations polynomiales sont les ensembles semi-algébriques.

Remarque :

La projection d'un semi-algébrique est un semi-algébrique.

Lemme :

T élimine les quanteurs \Leftrightarrow pour toute formule $\phi(y, \bar{x})$ sans quanteur il existe $\psi(\bar{x})$ sans quanteur et $T \vdash \forall \bar{x} : (\exists y : \phi(y, \bar{x}) \Leftrightarrow \psi(\bar{x}))$.

DÉMONSTRATION :

$$\exists y : (a \vee b) \Leftrightarrow (\exists y : a \vee \exists y : b) \quad \square$$

Lemme :

T élimine les quanteurs \Leftrightarrow pour toute formule $\phi(y, \bar{x})$ de la forme $\wedge_i P_i(y, \bar{x})\Delta_i 0$ il existe $\psi(y, \bar{x})$ sans quanteur et $T \vdash \forall \bar{x} : (\exists y : \phi(y, \bar{x}) \Leftrightarrow \psi(\bar{x}))$.

3.4.2 La méthode d'élimination

Donnée : les P_i et les Δ_i

Sortie : ψ

Définition :

Pour $P \in \mathbb{Z}[\bar{Y}][X]$ avec X à éliminer, $P(X) = a_0, \dots, a_n X^n, n \geq 1, a_n \neq 0$ on définit :

- $D(P)$ la dérivée $\frac{\partial P}{\partial X}$
- $E(P) = a_n$
- $O(P) = a_0, \dots, a_{n-1} X^{n-1}$
- $MR(P, Q)$ avec $Q = b_0, \dots, b_m X^m, n \geq m$ et $P \neq Q$ est l'unique polynôme tel qu'il existe un polynôme L de degré $< m$ avec $b_m^{n-m+1} . P = Q.L + R$

Si S et S' sont des ensembles de polynômes de $\mathbb{Z}[\bar{Y}][X]$ on définit :

- $D(S) = \{D(P) | P \in S \wedge \text{deg}P \geq 1\}$
- $E(S) = \{E(P) | P \in S \wedge \text{deg}P \geq 1\}$
- $O(S) = \{O(P) | P \in S \wedge \text{deg}P \geq 1\}$
- $MR(S, S') = \{MR(P, Q) | (P, Q) \in S \times S' \wedge P \neq Q \wedge \text{deg}P \geq \text{deg}Q \geq 1\}$
- $C(S, S') = D(S') \cup E(S') \cup O(S') \cup MR(S, S') \cup MR(S', S)$
- $\text{deg}S = \max_{P \in S} \text{deg}P$

Lemme :

1. $\#D(S), \#O(S), \#E(S) \leq \#S$
2. $\#MR(S, S') \leq (\#S)(\#S')$
3. $\text{deg}C(S, S') < \text{deg}S'$

Définition :

$$S_0 = S$$

$$S_n = C(\cup_{i=0}^{n-1} S_i, S_{n-1})$$

$$CS = \cup_{\mathbb{N}} S_n$$

Lemme :

1. CS est la clôture de S par O, E, D et MR
2. CS est fini
3. CS est une union finie de S_n

Définition :

BCS est l'ensemble des polynômes de CS de degré 0.

Une condition de signe sur $BCS = \{t_1(\bar{Y}), \dots, t_s(\bar{Y})\}$ est une formule de la forme $t_1(\bar{Y})\Delta_1 0 \wedge \dots \wedge t_s(\bar{Y})\Delta_s 0$ avec $\Delta_i \in \{<, >, =\}$.

On peut voir $\Delta = (\Delta_1, \dots, \Delta_s)^t$ comme une colonne.

Une condition de signe Δ sur BCS est satisfiable par \bar{a} si $\mathbb{R} \vdash t_1(\bar{a})\Delta_1 0 \wedge \dots \wedge t_s(\bar{a})\Delta_s 0$ qui est noté $\Delta(\bar{a})$.

Une condition de signe Δ sur BCS est satisfiable s'il existe \bar{a} tel que $\mathbb{R} \vdash \Delta(\bar{a})$.

Soient $S = \{P_1(\bar{Y}, X), \dots, P_l(\bar{Y}, X)\}$ et $\gamma_0 = -\infty < \gamma_1 < \gamma_2 < \dots < \gamma_n < \gamma_{n+1} = +\infty$ avec $\gamma_i \in \mathbb{R}$:

Considérons un tableau avec l lignes et $2n + 1$ colonnes :

] - \infty; \gamma_1[\gamma_1]\gamma_1; \gamma_2[\gamma_2	\dots	\gamma_n]\gamma_n; +\infty[
$P_1(\bar{Y}, X)$	>	Δ_{ij}					
\vdots							
$P_l(\bar{Y}, X)$							

A faire

Lemme :

Soient S un ensemble fini de polynômes de $\mathbb{Z}[\bar{Y}][X]$ et BCS l'ensemble de polynômes de degré nul correspondant. Il existe une fonction calculable qui à une condition de signe Δ pour BCS associe un tableau de changement de signe T pour CS tel que $\mathbb{R} \vdash \forall \bar{Y} : \Delta(\bar{Y}) \Leftrightarrow T(\bar{Y})$.

A faire l'exemple

Exemple :

$$\exists y : (ay^2 + by + c = 0), S = \{ay^2 + by + c\}$$

$$\Delta = (=, >, >, >)$$

$$\left\{ \begin{array}{l} a = 0 \\ b > 0 \\ c > 0 \\ b^2 - 4ac > 0 \end{array} \right.$$

	$] -\infty; \gamma_1[$	γ_1	$] \gamma_1; +\infty[$
a	$=$	$=$	$=$
b	$>$	$>$	$>$
c	$>$	$>$	$>$
$b^2 - 4ac$	$>$	$>$	$>$
$ay^2 + by + c$	$<$	$=$	$>$