Logique et calculabilité

Pierron Théo

ENS Ker Lann

Table des matières

Chapitre 1

Calculabilité

Thèse de Church (1936) : tout ce qui est calculable l'est par fonction récursive.

1.1 Machine de Turing

1.1.1 Machine de Turing déterministe

Définition 1.1 Une machine de Turing est constituée d'un ruban infini d'un côté, d'une tête de lecture, et d'un ensemble d'états, dont un état initial et un ensemble fini d'état accepteurs. On considère aussi un symbole spécial #.

Lors de l'éxécution, initialement, on est dans l'état initial, la tête de lecture sur la première case et un mot d'entrée est inscrit sur le ruban, suivi d'une infinité de #.

À chaque étape, la machine lit le symbole sous la tête, remplace ce symbole, change d'état et déplace la tête à gauche ou à droite

Définition 1.2 Une machine de Turing correspond à la donnée d'un heptuplet $(Q, \Gamma, \Sigma, \delta, q_0, B, F)$ avec

- Q ensemble d'états
- Γ alphabet fini de ruban
- $\Sigma \subset \Gamma$ alphabet fini des mots d'entrée
- $B \in \Gamma$ un symbole blanc
- $F \subset Q$ un ensemble d'états accepteurs
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{G, D\}$ fonction partielle

Une configuration est notée (q, α_1, α_2) avec q l'état courant, α_1 le mot strictement à gauche de la tête et α_2 le mot à droite de la tête juqu'au dernier caractère non blanc.

Remarque 1.1 *À tout moment de l'exécution, il existe une position à partir de laquelle il n'y a que des blancs.*

Définition 1.3 Dérivations Si $\delta(q, b) = (q'b', D)$, alors

$$(q, \alpha_1, \underbrace{b\alpha_2'}_{\alpha_2}) \vdash (q', \alpha_1 b', \alpha_2')$$

(avec $b = \#$ si $\alpha_2 = \varepsilon$). Si $\delta(q, b) = (q', b', G)$ $(\alpha_1 \neq \varepsilon)$,

$$(q, \underbrace{\alpha_1' a}_{\alpha_1}, b\alpha_2') \vdash (q', \alpha_1', ab'\alpha_2')$$

On notera $\vdash^*$ la fermeture réflexive transitive de $\vdash$.

1.1.2 Langage accepté et langage décidé

Définition 1.4 Un mot u est dit accepté par une machine de Turing M ssi l'exécution de M à partir d'une configuration initiale avec u sur le ruban mène à un état accepteur.

$$L(M) = \{u \in \Sigma^*, \exists q \in F, (q_0, \varepsilon, u) \vdash^* (q, \alpha_1, \alpha_2)\}$$

Exemple 1.1 $Q = \{q_0, \ldots, q_4\}$, $\Sigma = \{a, b\}$, $F = \{q_4\}$ et $\Gamma = \{a, b, X, Y, \#\}$ avec

δ	a	b	X	Y	$\#$
q_0	(q_1, X, R)			(q_3, Y, R)	$(q_4, \#, R)$
q_1	(q_1, a, R)	(q_2, Y, L)		(q_1, Y, R)	
q_2	(q_2, a, L)		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	$(q_4, \#, R)$

On passe du mot $aabb\#\ldots$ à $XXYY\#\ldots$. On a $L(M) = \{a^n b^n, n \geqslant 0\}$.

On a trois cas pour les transitions à partir de (q_0, ε, u) :

- La suite de transitions contient une configuration avec un état accepteur : u est accepté
- δ n'est pas défini ou déplacement gauche impossible : blocage
- suite infinie de configurations ne passant jamais par un état accepteur : boucle

Définition 1.5 On appelle exécution sur un mot u une suite de configurations partant de (q_0, ε, u) maximale.

Définition 1.6 On dit que L est décidé par M ssi M accepte L et M n'a as d'exécution infinie.

1.1.3 Autres définitions de machines de Turing

- Sans état accepteur, avec un état d'arrêt et δ définie partout. Dans l'état d'arrêt, M accepte ou non le mot suivant que la première case du ruban contient 0 ou 1. On dit que M décide de ce langage si elle atteint toujours l'état d'arrêt.
- Avec deux états d'arrêt q_Y et q_N.

1.1.4 Thèse de Turing-Church

« Les langages reconnus/calculés par procédure effective sont ceux décidés par machine de Turing ».

Le modèle est pertient :

- les autres modèles sont équivalents
- on peut simuler l'exécution d'un ordinateur
- le modèle est robuste (en enrichissant le modèle, on n'ajoute pas de langage)

1.2 Extension des machines de Turing

1.2.1 Ruban bi-infini

Pour passer d'un ruban bi-infini M à infini d'un seul côté M, on pose

$$\Gamma' = \Gamma \times (\Gamma \cup \{\$\}), \Sigma' = \Sigma \times \{B, \$\}, B' = (B, B),$$
$$Q' = Q \times \{b, b\} \cup \{q_0'\}, F' = F \times \{b, b\}$$

où \$ est un nouveau symbole.

L'idée est de dupliquer les états et d'écrire deux choses sur la même case (a_{-i} et a_i).

1.2.2 Rubans multiples

Par exemple, deux rubans avec deux têtes de lecture.

Il suffit de considérer un ruban avec des quadruplets : deux composantes pour le contenu des rubans et deux pour les positions des têtes.

1.2.3 Machines à RAM

On considère une machine avec une RAM et des registres. La machine répète le cycle :

- se déplacer dans le RAM jusqu'à l'adresse dans le Program Counter
- lire et décoder l'nstruction à cette adresse
- trouver les opérandes
- exécuter l'instruction (modifier la RAM et le registres)
- mettre à jour Program Counter

1.2.4 Machine non déterministe

On considère Δ une relation de $(Q \times \Gamma) \times (Q \times \Gamma \times \{G, D\})$ au lieu de δ. Il n'y a alors plus unicité de l'exécution.

Définition 1.7 Un mot est dit accepté ssi il existe une exécution avec un état accepteur.

THÉORÈME 1.1 *Tout langage accepté par une machine non déterministe est accepté par une machine déterministe.*

1.2.5 Machines de Turing universelles

Il existe des machines de Turing qui simulent des machines de Turing. On parle de machine de Turing universelle.

1.2.6 Fonction calculable par machine de Turing

Définition 1.8 Une machine de Turing calcule une fonction $f : \Sigma^* \to \Sigma^*$ ssi pout tout mot d'entrée u, la machine s'arrête dans une configuration où $f(u)$ se trouve sur le ruban.

Une fonction f est calculable par machine de Turing ssi il existe une machine de Turing qui la calcule.

1.3 Fonctions récursives

On considère des fonctions de $\mathbb{N}^k \to \mathbb{N}$ avec $k \geqslant 0$.

1.3.1 Fonctions primitives récursives

Définition 1.9 Les fonctions primitives récursives de bases sont :

- La fonction $0()$ qui renvoie 0.
- La fonction successeur $\sigma : n \mapsto n + 1$.
- Les projecteurs $\pi_i^k : (n_1, \ldots, n_k) \mapsto n_i$ pour $1 \leqslant i \leqslant k$.

Définition 1.10 Composition Soient $h_1, \ldots, h_l : \mathbb{N}^k \to \mathbb{N}$ et $g : \mathbb{N}^l \to \mathbb{N}$.

La composée f de g et $h_1, \ldots, h_l$ est définie par $f(\overline{n}) = g(h_1(\overline{n}), \ldots, h_l(\overline{n}))$ si $\overline{n} = (n_1, \ldots, n_k)$.

Définition 1.11 Récursion primitive Soit $g : \mathbb{N}^k \to \mathbb{N}$, $h : \mathbb{N}^{k+2} \to \mathbb{N}$.

$f : \mathbb{N}^{k+1} \to \mathbb{N}$ est définie par récursion primitive à partir de g et h par

$$f(\overline{n}, 0) = g(\overline{n}) \text{ et } f(\overline{n}, m+1) = h(\overline{n}, m, f(\overline{n}, m))$$

Définition 1.12 Les fonctions primitives récursives sont les fonctions primitives récursives de base et les fonctions obtenues à partir des fonctions récursives primitives par composition et récursion primitive.

Exemple 1.2

- La fonction constante $j() = \sigma^j \circ 0()$.
- L'identité π_1^1.
- La somme $+(n_1, n_2)$ par récursion primitive avec $g = \mathrm{Id}$ et $h = \sigma \circ \pi_3^3$.
- Le produit $\times(n_1, n_2)$ par récursion avec $g = 0()$ et $h(n_1, n_2, \times(n_1, n_2)) = +(\pi_3^1, \pi_3^3)$.
- La puissance de même.
- On généraliser à $n \uparrow^k 0 = 1$ et $n \uparrow^k (m+1) = n \uparrow^{k-1} (n \uparrow^k m)$ ie $f(k+1, n, m+1) = f(k, n, f(k+1, n, m))$ qui récursionne sur deux indices. De plus, si on enlève le n inutile, on tombe sur la fonction d'Ackermann.
- Prédécesseur : $\mathrm{pred}(0) = 0$ et $\mathrm{pred}(n+1) = n$.
- Soustraction : $n \dot{-} 0 = n$ et $n \dot{-} (m+1) = \mathrm{pred}(n \dot{-} m)$.
- Signe : $\mathrm{Sgn}(m) = 0$ si $m = 0$, sinon 1.
- Produit borné $\prod_{i=0}^{n} g(\overline{n}, i)$.

Définition 1.13 La fonction d'Ackermann est définie par

$$\begin{cases} A(0, m) & = m+1 \\ A(k+1, 0) & = A(k, 1) \\ A(k+1, m+1) & = A(k, A(k+1, m)) \end{cases}$$

Elle n'est pas primitive récursive.

1.3.2 Prédicats primitifs récursifs

Définition 1.14 Un prédicat P à k arguments est une partie de $\mathbb{N}^k$.

Définition 1.15 Un prédicat est dit récursif primitif ssi son indicatrice l'est.

Exemple 1.3

- zéro ?$(0) = 1$ et zéro ?$(m+1) = 1$.
- $n < m = \mathrm{Sgn}(m \dot{-} n)$.
- $(n == m) = 1() \dot{-} (\mathrm{Sgn}(m \dot{-} n) + \mathrm{Sgn}(n \dot{-} m))$.
- opérations logiques : ET= $\times$, OU= Sgn(+), NON= $1() \dot{-}$.
- quantification bornée : $\forall i \leqslant n, p(\overline{n}, i)$ correspond au produit borné $\prod_{i=0}^{n} p(\overline{n}, i)$

  $\exists i \leqslant n, p(\overline{n}, i)$ c'est $1() \dot{-} \prod_{i=0}^{n} (1 \dot{-} p(\overline{n}, i))$.
- définition par cas : $f(\overline{n}) = g_i(\overline{n})$ si $p_i(\overline{n})$. Si les p_i sont mutuellements exclusifs, $f(\overline{n} = \sum_{i=1}^{k} p_i(\overline{n}) g_i(\overline{n})$.
- $\mu(m, q(\overline{n}, \cdot))$ est le plus petit entier $i \leqslant m$ tel que $q(\overline{n}, i)$ et 0 si un tel i n'existe pas. On la définit par $\mu(0, q(\overline{n}, \cdot)) = 0$ et

$$\mu(m+1, q(\overline{n}, \cdot)) = \begin{cases} \mu(m, q(\overline{n}, \cdot)) & \text{si } \mu(m, q(\overline{n}, \cdot)) = 1 \\ m+1 & \text{si } \mu(m, q(\overline{n}, \cdot)) = 0 \text{ et } q(\overline{n}, m+1) \\ 0 & \text{sinon} \end{cases}$$

1.3.3 Fonctions récursives générales

Remarque 1.2

- *Il existe des fonctions $\mathbb{N} \to \mathbb{N}$ qui ne sont pas primitives récursives.*
- *Les fonctions primitives récursives sont dénombrables car on sait les décrire sur un alphabet.*
- *$\mathcal{P}(\mathbb{N})$ n'est pas dénombrable. En effet, s'il l'était, on aurait une énumération $(S_n)_n$. On pose $D = \{n, n \notin S_n\}$. Il existe k tel que $D = S_k$. Si $k \in S_k$, $k \notin S_k$ et si $k \notin S_k$, $k \in S_k$. Contradiction.*
- *Il existe des fonctions calculables qui ne sont pas primitives récursives. On considère f_n une énumération des fonctions primitives récursives. Si f a k arguments, on considère $f(n, \ldots, n)$.*
  On pose $g(n) = f_n(n) + 1$. Si g était primitive récursive, ce serait un f_k et donc $f_k(k) = g(k) = f_k(k) + 1$. Pourtant, on sait calculer g de façon effective via l'énumération des f_i pour $i \leqslant n$ et le calcul de $f_n(n)$.

On étend la notion de fonction récursive en définissant la minimisation non bornée :

$$\mu(i, q(\overline{n}, i)) = \min\{i, q(\overline{n}, i) = 1\}$$

avec 0 si un tel i n'existe pas.

Définition 1.16 Un prédicat $q(\overline{n}, 1)$ est dit sûr ssi pour tout $\overline{n}$, il existe i tel que $q(\overline{n}, i) = 1$.

Définition 1.17 Les fonctions μ-récursives sont obtenues à partir des fonctions récursives primitives de base par composition, récursion primitive, et minimisation non bornée de prédicats sûrs.

1.4 Fonctions calculables par machine de Türing et fonctions récursives

Les fonctions μ-récursives sont les fonctions calculables par machine de Türing.

1.4.1 Codage

On veut passer de façon effective des entiers aux chaînes de caractères et réciproquement.

Soit Σ un alphabet à k caractères, on prend $\mathrm{gd} : \Sigma \to [\![1, k]\!]$ (Gödelisation) et on l'étend à Σ^* par

$$\mathrm{gd}(u_0 \dots u_n) = \sum_{i=0}^{n} (k+1)^{n-i} \mathrm{gd}(u_i)$$

qui est calculable par une procédure effective. Pour la transformation inverse il suffit de diviser donc on sait aussi le faire par calcul effectif.

1.4.2 Toute fonction récursive est calculable par MT

- Les fonctions primitives primitives récursives de base sont calculables par MT.
- La composition passe bien aux MT
- La récursion primitive passe aux MT (il suffit de considérer plussieurs rubans, dont un qui garde l'entier sur lequel on récursive)
- La minimisation non bornée de prédicats sûrs passe aussi.

Remarque 1.3 On aurait pu passer par une compilation sur machine à RAM.

1.4.3 Toute fonction calculable par MT est μ-récursive

Soit M une MT claculant $f_M : \Sigma^* \to \Sigma^*$. On prend gd un encodage des mots de Σ^* par des entiers.

Existe-t-il f récursive telle que $f_M(\omega) = \mathrm{gd}^{-1}(f(\mathrm{gd}(\omega)))$? On définit les fonctions primitives récursives

- configuration de M : (q, α_1, α_2) est codé par 3 entiers

- init(x) : configuration initiale pour un mot d'entrée
- config_suivante(y) : codage de la configuration successeur de celle codée par y
- config(x, n) définie par config($x, 0$) = x et

$$\text{config}(x, n+1) = \text{config\_suivante}(\text{config}(x, n))$$

- stop(y) qui vaut 1 si y code une configuration finale et 0 sinon
- sortie(y) la valeur calculée par la machine dans une configuration finale y.

On a alors la définition de f :

$$f(x) = \text{sortie}(\text{config}(\text{init}(x), \mu_i \text{stop}(\text{config}(\text{init}(x), i))))$$

On a bien un prédicat sûr puisque la MT s'arrête.

1.5 Non calculabilité

Il existe des fonctions non calculables par MT.

1.5.1 Problèmes et classes de décidabilité

On considère des problèmes binaires (la réponse est oui ou non). On encode chaque instance du problème sur un alphabet Σ.

On dira que les instances positives sont les mots pour lesquels la réponse est oui. La réponse ne dépend pas de l'encodage. Un problème est équivalent au langage de ses instances.

Définition 1.18 La classe de décidabilité R est l'ensemble des langages décidables par MT (ie décidables, calculables, récursifs).

La classe de décidabilité RE est l'ensemble des langages acceptés par MT (ie partiellement décidables, partiellement récursifs).

Proposition 1.1 $R \subset RE$.

1.5.2 Premières briques

On veut exhiber une langage n'appartenant pas à RE. On énumère les MT $(M_n)_n$ en supposant que les MT travaillent sur le même alphabet. On énumère aussi les mots w_n. On construit ainsi un tableau infini A tel que $A[i, j]$ =vrai ssi la machine M_i accepte le mot w_j. Ainsi, $A[i, j]$ =faux ssi on a un rejet ou une boucle infinie.

Le langage L_0 est défini par

$$L_0 = \{w, w = w_i \text{ et } A[i,i] = \text{faux}\}$$

Proposition 1.2 $L_0 \notin RE$.

Démonstration. Si $L_0 \in RE$, il existe M_i qui accepte L_0. Si M_i accepte w_i alors $w_i \in L_0$ donc M_i n'accepte pas w_i.

Si M_i n'accepte pas w_i, $w_i \notin L_0$ donc M_i accepte w_i. Contradiction. ■

Lemme 1.1.1
Si $L \in R$ alors $\overline{L} \in R$.

Lemme 1.1.2
Si $L \in RE$ et $\overline{L} \in RE$ alors $L \in R$.

Démonstration. Si M accepte L et $\overline{M}$ accepte $\overline{L}$, on construit M' qui simule en parallèle l'exécution de M et $\overline{M}$ sur un mot. ■

On a trois cas possibles :
- $L \in R$ et $\overline{L} \in R$ (cas gentil)
- $L \notin RE$ et $\overline{L} \notin RE$
- $L \notin RE$ et $\overline{L} \in RE \setminus R$.

Proposition 1.3 $\overline{L_0} \in RE$.

Démonstration. On construit une MTU qui accepte $\overline{L_0}$. Elle prend w en entrée, elle énumère les w_i jusqu'à trouver i tel que $w_i = w$ puis les MT jusqu'à M_i et enfin on simule l'exécution de M_i sur w_i. ■

1.5.3 Réductions

On veut montrer qu'un langage L_2 est indécidable sachant que L_1 l'est. La preuve générale consiste à supposer L_2 décidable et à en déduire que L_1 l'est.

Exemple 1.4 Soit $LU = \{\langle M, w\rangle, M \text{ accepte } w\}$ (où $\langle M, w\rangle$ est un mot codant M et w). $LU \in RE$ car il est reconnu par MTU.

$LU \notin R$. Pour le montrer, on va réduire $\overline{L_0}$ à LU. Supposons qu'on ait un algorithme qui décide LU. On construit un algorithme qui décide $\overline{L_0}$:
- On détermine i tel que $w = w_i$
- On détermine la MT d'indice i
- On code $\langle M_i, w_i\rangle$ et on applique LU ($LU \in R$)
- Si le résultat est positif, accepter w sinon le refuse.

Donc $\overline{L_0} \in R$. Contradiction. On en déduit que $\overline{LU} \notin R$ et $\overline{LU} \notin RE$.

1.5.4 Exemples de problèmes indécidables

Problème de l'arrêt

Proposition 1.4 $H = \{\langle M, w\rangle, M \text{ s'arrête sur } w\} \notin R$.

Démonstration. On suppose qu'on a un algorithme qui décide H.
- On l'applique à $\langle M, w\rangle$
- Si la réponse est non, on répond non
- Sinon, simuler l'exécution de M sur w et renvoyer la réponse.

Donc on a un algorithme pour LU. ∎

Problème de l'arrêt sur mot vide

Pour une instance $\langle M, w\rangle$ du problème de l'arrêt, on construit une machine de Turing M' qui écrit w sur son ruban et se comporte comme M après.

On applique l'algorithme du problème de l'arrêt sur mot vide à M' et on a un problème.

Exemple 1.5 D'autres exemples : l'arrêt existentiel/universel

Problème du langage accepté vide

Une instance est une machine M. On cherche si $L(M) = \varnothing$. On réduit à partir de $\overline{LU}$.

Pour une instance $\langle M, w\rangle$ de $\overline{LU}$, on construit la MT M' qui simule l'exécution de M sur w sans tenir compte du mot d'entrée.

On dira que M' accepte son mot d'entrée ssi M accepte w. On résout alors le problème du langage accepté vide pour M'. On a $L(M') = \varnothing$ ssi M n'accepte pas w, ce qui résout $\overline{LU}$

1.5.5 Propriétés des langages RE

Langages calculés par MT

Définition 1.19 Soit M une MT. Le langage calculé par M est $L_c(M) = \{w, \exists u, M \text{ s'arrête pour } u \text{ et } w = f_M(u)\}$.

THÉORÈME 1.2 *Un langage est calculé par MT ssi il est RE.*

Démonstration.

⇐ Si $L \in RE$, il existe M qui l'accepte. On construit M' qui l'accepte : on stocke le mot d'entrée, on simule l'exécution de M et si le mot est

accepté par M, on l'efface et on recopie le mot de départ. Sinon, M' boucle.
On remarque alors que L est calculé par M'.

$\Rightarrow$ Si L est calculé par M. On construit M' qui accepte w ssi il existe u tel que $w = f_M(u)$. On va utiliser le non-déterminisme pour simuler l'exécution sur tous les u possibles. ■

Langages énumérés par MT

Si on a une MT M, on peut énumérer les mots acceptés par M en utilisant du non-déterminisme.

Langages générés par une grammaire

THÉORÈME 1.3 *Les langages RE sont les langages générés par les grammaires.*

1.5.6 Problèmes indécidables divers

- Étant donnée une grammaire hors-contexte G, on ne sait pas décider si $L(G) = \Sigma^*$.
- On ne sait pas décider si l'intersection de de deux langages algébriques est vide ou non.
- On ne sait pas dire si une formule du calcul des prédicats du premier ordre est satisfiable.
- On ne sait pas dire si $P \in \mathbb{Z}[X_1, \ldots, X_n]$ a une racine dans $\mathbb{Z}^n$.
- Toute propriété non triviale sur les langages RE est indécidable. En langage plus rigoureux, soit X un ensemble de fonctions récursives partielles à une variable.
  Si X et $\overline{X}$ sont non vides, alors l'ensemble des indices des fonctions de X (à chaque élément de X on associe un entier) n'est pas récursif.

Chapitre 2

Induction

2.1 Construction et et preuve par induction

Définition 2.1 Une définition inductive d'une partie X d'un ensemble E est la donnée d'un sous ensemble B de E et d'un ensemble K de fonctions partielles $\Phi : E^{a(\Phi)} \to E$ où $a(\Phi)$ est l'arité de Φ.

X est alors défini comme le plus petit ensemble vérfiant

(i) $B \subset X$

(ii) $\forall \Phi \in K$ et $x_1, \ldots, x_{a(\Phi)} \in X$, $\Phi(x_1, \ldots, x_{a(\Phi)}) \in X$.

Exemple 2.1 On peut définir les entiers pairs par $0 \in P$ et si $n \in P$, $n + 2 \in P$.

Exemple 2.2 L'ensemble AB des arbres binaires sur l'alphabet A est construit avec $E = (A \cup \{\varnothing, (,), ;\})^*$, $B = \{\varnothing\}$ et si $g, d \in AB$, pour tout $a \in A$, $(a; g; d) \in AB$.

THÉORÈME 2.1 *Si X est défini inductivement, tout élément de X s'obtient à partie de la base en appliquant un nombre fini d'étapes :*

$$X = \bigcup_{n \in \mathbb{N}} X_n$$

où X_n est l'ensemble des éléments obtenus par n itérations.

THÉORÈME 2.2 *Soit X un ensemble défini inductivement et soit $P(x)$ un prédicat exprimant une propriété de $x \in X$. Si $P(x)$ est vraie pour $x \in B$ et si pour tout $x_1, \ldots, x_{a(\Phi)} \in X$ vérifiant $P(x_1), \ldots, P(x_{a(\Phi)})$ sont vraies, alors $P(\Phi(x_1, \ldots, x_{a(\Phi)}))$ est vraie pour tout $\Phi \in K$ alors $P(x)$ est vraie pour tout $x \in X$.*

2.2 Définition non-ambiguë

Définition 2.2 La définition inductive de X est dite non-ambiguë si

- pour tout $x_1, \ldots, x_{a(\Phi)} \in X$ et $\Phi \in K$, $\Phi(x_1, \ldots, x_{a(\Phi)}) \notin B$
- pour tout $x_1, \ldots, x_{a(\Phi)}, x'_1, \ldots, x'_{a(\Phi')} \in X$ et $\Phi, \Phi' \in K$,

$$\Phi(x_1, \ldots, x_{a(\Phi)}) = \Phi'(x'_1, \ldots, x'_{a(\Phi')})$$

implique $\Phi = \Phi'$ et $x_i = x'_i$ pour tout i.

THÉORÈME 2.3 *Soit $X \subset E$ défini inductivement de manière non-ambiguë. Soit F un ensemble, $f_B : B \to F$ et une famille $f_\Phi : E^{a(\Phi)} \times F^{a(\Phi)} \to F$. Il existe une unique fonction $f : X \to F$ telle que*

- *pour tout $x \in B$, $f(x) = f_B(x)$*
- *pour tout $x_1, \ldots, x_{a(\Phi)} \in X$ et $\Phi \in K$,*

$$f(\Phi(x_1, \ldots, x_{a(\Phi)})) = f_\Phi(x_1, \ldots, x_{a(\Phi)}, f(x_1), \ldots, f(x_{a(\Phi)}))$$

Chapitre 3

Calcul propositionnel

3.1 Syntaxe

Soit P un ensemble dénombrable de symboles appelés variables propositionnelles, C un ensemble fini de connecteurs $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$.

Définition 3.1 L'ensemble des formules propositionnelles $\mathcal{F}$ est un langage sur $C \cup P \cup \{(,)\}$ défini par induction

(i) $P \subset \mathcal{F}$.

(ii) Pour tout $\alpha \in C$ tel que $\alpha \neq \neg$, et pour tout $x, y \in \mathcal{F}^2$, $(x\alpha y) \in \mathcal{F}$.

(iii) Pour tout $x \in \mathcal{F}$, $\neg x \in \mathcal{F}$.

On a donc $\mathcal{F} = \bigcup_{n \in \mathbb{N}} \mathcal{F}_n$.

Définition 3.2 La hauteur $h(f)$ d'une formule f est le plus petit entier n tel que $f \in \mathcal{F}_n$.

THÉORÈME 3.1 *La définition de $\mathcal{F}$ est non ambiguë.*

COROLLAIRE 3.1 *Si F est une formule, soit $F \in P$, soit il existe une unique G telle que $F = \neg G$, soit il existe un unique $(G, H) \in F^2$ et $\alpha \in C \setminus \{\neg\}$ tel que $F = (G\alpha H)$.*

Définition 3.3 Une occurrence de la variable p dans la formule F est la donnée de cette variable et de la place où elle apparaît dans F.

Définition 3.4 La formule $F[G/p]$ (substitution de G à p dans F) est définit par induction sur la formule F :

(i) Si $F = p$, alors $F[G/p] = G$

(ii) Si $F \in P \setminus \{p\}$ alors $F[G/p] = F$

(iii) Si $F = \neg H$, $F[G/p] = \neg H[G/p]$

(iv) Si $F = (H_1 \alpha H_2)$ alors $F[G/p] = (H_1[G/p] \alpha H_2[G/p])$

3.2 Sémantique

Définition 3.5 Une distribution de valeurs de vérité (ou valuation) est une application $\varphi : P \to \mathbb{Z}/2\mathbb{Z}$.

THÉORÈME 3.2 *Il existe un unique prolongement d'une valuation φ à $\mathcal{F}$ noté $\overline{\varphi}$ qui satisfait les conditions*

- $\overline{\varphi}|_P = \varphi$
- *Si* $F = \neg G$, $\overline{\varphi}(F) = 1 - \overline{\varphi}(G)$
- *Si* $F = (H_1 \wedge H_2)$, $\overline{\varphi}(F) = \overline{\varphi}(H_1)\overline{\varphi}(H_2)$
- *Si* $F = (H_1 \vee H_2)$, $\overline{\varphi}(F) = 1 - (1 - \overline{\varphi}(H_1))(1 - \overline{\varphi}(H_2))$
- *Si* $F = (H_1 \Rightarrow H_2)$, $\overline{\varphi}(F) = 1 - \overline{\varphi}(H_1)(1 - \overline{\varphi}(H_2))$
- *Si* $F = (H_1 \Leftrightarrow H_2)$, $\overline{\varphi}(F) = 1 - (\overline{\varphi}(H_1) + \overline{\varphi}(H_2))$

Définition 3.6

- Une formule F est satisfaite par une valuation φ ssi $\varphi(F) = 1$.
- Une tautologie est une formule satisfaite par toute valuation.
- Les formules F et G sont équivalentes ssi pour tout φ, $\varphi(F) = \varphi(G)$. C'est une relation d'équivalence notée $\equiv$.

Exemple 3.1

$$((p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r)))$$

est une tautologie.

Proposition 3.1 F et G sont équivalentes ssi $(F \Rightarrow G)$ et $(G \Rightarrow F)$ sont des tautologies.

Définition 3.7 Soit Σ un ensemble de formules et F une formule. On dit que F est conséquence de Σ et on note $\Sigma \vDash F$ si toute valuation qui satisfait toutes les formules de Σ satisfait aussi F.

Proposition 3.2 $\Sigma \vDash F$ ssi $\Sigma \cup \{\neg F\}$ n'est pas satisfiable.

Proposition 3.3 Soit $F(p_1, \ldots, p_n)$ une formule et φ une valuation. La valeur $\varphi(F)$ ne dépend que de $\varphi(p_1), \ldots, \varphi(p_n)$.

Proposition 3.4 Soient F et G deux formules, φ une valuation. On a $\varphi(F[G/p]) = \varphi'(F)$ avec $\varphi'(p) = \varphi(G)$ et pour tout $q \neq p$, $\varphi'(q) = \varphi(q)$.

COROLLAIRE 3.2 *Si $F \equiv F'$ alors $F[G/p] \equiv F'[G/p]$ et si $G \equiv G'$, $F[G/p] \equiv F[G'/p]$.*

Exemple 3.2

$$((F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H)))$$

est une tautologie.

THÉORÈME 3.3 *Il y a une bijection entre les applications de* $\{0,1\}^n \to \{0,1\}$ *et l'ensemble des formules à n variables quotienté par la relation d'équivalence sémantique. Les représentants sont appelés formes normales.*

Définition 3.8 Forme normale disjonctive On dit que F est sous forme normale disjonctive ssur $F = G_1 \vee \dots G_k$ avec $G_i = B_{i,1} \wedge \dots B_{i,l_i}$ avec $B_{i,j} = p$ ou $B_{i,j} = \neg p$.

Remarque 3.1 Les formes normales conjonctives c'est pareil en permutant $\wedge$ *et* $\vee$.

Définition 3.9 Un système complets de connecteurs est un sous-ensemble de C qui permet d'engendre toutes les applications de $\{0,1\}^n \to \{0,1\}$.

Définition 3.10 Un ensemble Σ de formules est dit finiment satisfiable ssi tout sous-ensemble fini de Σ est satisfiable.

Un ensemble Σ de formules finiment satisfiable est dit maximal ssi pour toute formule f, f ou $\neg f$ appartiennent à Σ.

THÉORÈME 3.4 COMPACITÉ *Un ensemble de formules* Σ *est satisfiable ssi il est finiment satisfiable.*

Démonstration. Le sens $\Rightarrow$ est clair. Montrons l'autre. On sait qu'il y a une bijection entre les valuations et les ensembles finiment satisfiables maximaux.

Pour $p \in \mathcal{P}$, on pose $\varphi(p) = 1$ ssi $p \in \Sigma$. On vérifie que $\Sigma = \{F, \varphi(F) = 1\}$.

Si Σ_0 est finiment satisfiable, on construit Σ finiment satifiable maximal le contenant.

On part de Σ_0 et d'une énumération des formules $(F_n)_n$ et on pose $\Sigma_{n+1} = \Sigma_n \cup \{F_n\}$ si $\Sigma_n \cup \{F_n\}$ est finiment satisfiable et $\Sigma_{n+1} = \Sigma_n \cup \{\neg F_n\}$ sinon. On prend alors $\Sigma = \bigcup_{n \in \mathbb{N}} F_n$. ∎

3.3 Systèmes de déduction

Définition 3.11 Un système formel est la donnée d'un alphabet, un procédé de formation des formules sur cet alphabet, un ensemble d'axiomes, un ensemble de règles de déduction.

3.3.1 Déduction par coupure

On se limite à un sous-ensemble de formules : les clauses.

Définition 3.12 Une clause C est une formule $B_1 \vee \ldots \vee B_k$ où chaque B_i est soit une variable soit sa négation.

Les variables apparaissant sans néggation dans C sont appelées variables positives, les autres sont les variables négatives.

Proposition 3.5 Toute formule du calcul propositionnel est sématiquement équivalente à une conjonction finie de clauses.

Remarque 3.2 Toute clause contenant au moins une variable positive et une variable négative est équivalente à une formule de la forme

$$(a_1 \wedge \ldots \wedge a_k) \Rightarrow (b_1 \vee \ldots \vee b_p)$$

avec a_i négatifs et b_i positifs. On notera $C = (\Gamma, \Delta)$ avec Γ l'ensemble des variables négatives et Δ l'ensemble des variables positives.

Remarque 3.3 Si $\Delta = \varnothing$, on parle de clause négative. Si $\Gamma = \varnothing$, on parle de clause positives. Si $\Gamma = \Delta = \varnothing$, on obtient la clause vide notée $\square$, qui n'est satisfaite par aucune valuation.

Définition 3.13 Soit C_1, C_2 deux clauses et $p \in \Delta_1 \cap \Gamma_2$. On dit que C se déduit par coupure sur p de C_1 et C_2 ssi $C = (\Gamma, \Delta)$ avec $\Gamma = \Gamma_1 \cup (\Gamma_2 \setminus \{p\})$ et $\Delta = \Delta_2 \cup (\Delta_1 \setminus \{p\})$.

On notera $\frac{C_1 C_2}{C}$.

Lemme 3.4.1 Validité
Si $\frac{C_1 C_2}{C}$ alors $\{C_1, C_2\} \vDash C$

Démonstration. Soit φ tel que $\varphi \vDash C_1$ et $\varphi \vDash C_2$ et supposons que $\varphi \nvDash C$.

Pour tout $q \in \Gamma_1 \cup (\Gamma_2 \setminus \{p\})$, $\varphi(q) = 1$ et pour tout $r \in (\Delta_1 \setminus \{p\}) \cup \Delta_2$, $\varphi(r) = 0$.

Si $\varphi(p) = 1$, alors $\varphi(C_2) = 0$ et symétriquement, si $\varphi(p) = 0$, $\varphi(C_1) = 0$, ce qui est absurde. ∎

Définition 3.14 Soit S un ensemble de clauses, C une clause. Une preuve par coupure de C à partir de S est une suite finie $C_1, \ldots, C_k = C$ telle que pour tout i, $C_i \in S$ ou il existe $j < i$ et $l < i$ tel que $\frac{C_j C_l}{C_i}$.

On note $S \vdash C$ l'existence d'une telle preuve. Une réfutation de S par coupure est une preuve de $\square$ à partir de S par coupure.

Remarque 3.4 On utilise la preuve par coupure pour prouver si $\Sigma \vDash F$.

On met les formules de $\Sigma \cup \{\neg F\}$ sous forme de clauses et on cherche une réfutation par coupure. Si on arrive à $\square$, en contraposant le lemme précédent, on trouve que $\Sigma \cup \{\neg F\}$ n'est pas satisfiable donc $\Sigma \vDash F$ (voir corollaire 3.3).

Lemme 3.4.2
Si $S \vdash C$ alors $S \vDash C$.

Démonstration. Récurrence sur la longueur de la suite des C_i. ■

COROLLAIRE 3.3 *Si $S \vdash \square$, alors S n'est pas satisfiable.*

Complétude de la démonstration par coupure

Proposition 3.6 Tout ensemble de clauses non satisfiable possède une réfutation par coupure.

Lemme 3.4.3
Soit S un ensemble de clauses non statisfiable, qui ne contient pas la clause vide (sinon on a fini), alors il existe une variable p et deux clauses C_1 et C_2 de S tel que $p \in \Delta_1 \cap \Gamma_2$ (Γ : positif, Δ : négatif).

Démonstration. Par l'absurde, si pour toute variable p :

1. p apparait positivement dans toute clause de S, on pose $\varphi(p) = 1$
2. p apparait négativement dans toute clause de S, on pose $\varphi(p) = 0$

On construit ainsi une valuation qui satisfait S : absurde. ■

Définition 3.15 Résolvant En notant S_p (on reprend les notations du lemme précédent) le sous-ensemble des clauses contenant p, le résolvant de S_p, $\mathrm{Res}(S_p)$ est l'ensemble des clauses obtenues par coupure à partir de deux clauses de S_p (avec p négatif dans l'une et positif dans l'autre).

Remarque 3.5 Si S non satisfiable, il existe p tel que $\mathrm{Res}(S_p) \neq \varnothing$.

Lemme 3.4.4
S est satisfiable ssi $(S \setminus S_p) \cup \mathrm{Res}(S_p)$ satisfiable.

Démonstration.

- $\Rightarrow$ facile : comme on enlève des clauses dans $(S \setminus S_p)$ par rapport à S, la valuation de S reste valable pour $(S \setminus S_p)$.
- $\Leftarrow$ Soit φ satisfaisant $(S \setminus S_p) \cup \mathrm{Res}(S_p)$ On va prolonger φ sur $\{p\}$. Comme φ satisfait toutes les clauses de la forme $C_1 \vee C_2$ obtenues par coupure sur p à partir des clauses $(C_1 \vee p)$ et $(C_1 \vee \neg p)$ de S_p.

  S'il existe une clause $C_1 \vee p$ de S_p telle que $\varphi(C_1) = 0$ on a alors $varphi(C_2) = 1$. Et on pose $\varphi(p) = 1$ ce qui nous donne :

$$\varphi(C_2 \vee \neg p) = 1$$
$$\varphi(C_1 \vee p) = 1$$

  On a donc satisfait toutes les clauses.

  si pour toute $C_1 \vee p$), on a $\varphi(C_1) = 1$, on pose $\varphi(p) = 0$ et on a $\varphi(C_2 \vee \neg p) = 1$.

  φ ainsi prolongée satisfait toutes les clauses de S_p. ■

THÉORÈME 3.5 *Tout ensemble fini de clauses non satisfiables possède une réfutation par coupure.*

Démonstration. Comme on a un ensemble fini de variables et que l'on sait les éliminer une par une (grâce aux deux lemmes précédent), une récurrence montre le théorème. ■

Démonstration du théorème de complétude. Théorème de compacité. ■

Le problème est la construction de la coupure : pour l'instant, on ne peut tomber que par hasard dessus. On va donc plutôt travailler sur les clauses de HORN, où il existe des stratégies efficaces pour construire la clauses vide.

3.3.2 Déduction naturelle (GENTZEN)

À partir d'un ensemble de formules Γ et A une formule, on cherche à déduire A de Γ : $\Gamma \vdash A$ On note

$$\Gamma, A \text{ pour } \Gamma \cup \{A\}$$
$$\Gamma, \Delta \text{ pour } \Gamma \cup \Delta$$

Logique minimale NM

Si A est dans Γ on peut déduire A de Γ, noté :

$$\frac{}{\Gamma, A \vdash A}$$

On note intro une règle qui permet d'introduire un opérateur et élim une règle qui permet de l'éliminer.

- Pour $\Rightarrow$, on a les constructeurs/destructeurs :

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\text{intro}_{\Rightarrow} \text{ et } \frac{\Gamma \vdash A \quad \Delta \vdash A \Rightarrow B}{\Gamma, \Delta \vdash B}\text{élim}_{\Rightarrow}$$

- Pour $A \wedge B$ sous les hypothèses Γ :

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\text{intro}_{\wedge} \text{ et } \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\text{élim}_{\wedge 1} \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\text{élim}_{\wedge 2}$$

- Pour $A \vee B$,

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\text{intro}_{\vee 1} \text{ et } \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\text{intro}_{\vee 2}$$

et

$$\frac{\Gamma \vdash A \vee B \quad \Delta, A \vdash C \quad \Delta', B \vdash C}{\{\Gamma, \Delta, \Delta'\} \vdash C}\text{élim}_{\vee}$$

Un exemple d'arbre de preuve :

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{(A \wedge B) \vdash (A \wedge B)}^{\text{ax}}}{(A \wedge B) \vdash B}\text{élim}_{\wedge 2} \quad \dfrac{\overline{(A \wedge B) \vdash (A \wedge B)}^{\text{ax}}}{(A \wedge B) \vdash A}\text{élim}_{\wedge 1}}{(A \wedge B) \vdash (B \wedge A)}\text{intro}_{\wedge}}{\vdash (A \wedge B) \Rightarrow (B \wedge A)}\text{intro}_{\Rightarrow}$$

Logique intuitionniste (NJ)

On ajoute un symbole au langage naturel : $\bot$ (absurde) et la règle

$$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}\text{élim}_{\bot}$$

ex falso quodibet sequitur $\neg A$ est une abréviation pour $A \Rightarrow \bot$. Pour travailler sur des formules avec négation, on utilise les règles

$$\dfrac{\Gamma, A \vdash \neg B \quad \Gamma, A \vdash B}{\Gamma \vdash \neg A}\text{intro}_{\neg}$$

$$\dfrac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \bot}\text{élim}_{\neg}$$

Exemple 3.3 On peut prouver $((A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A))$ en logique minimale. C'est un cas particulier de $((A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C)))$.

Remarque 3.6 On a :
- *élim$_\neg$ dérivable en NM.*
- *élim$_\neg$ utilise la règle élim$_\bot$.*

Proposition 3.7 NJ est plus forte que NM : si $\Gamma \vdash_{NM} A$, alors $\Gamma \vdash_{NJ} A$.

Logique classique (NK)

On ajoute un nouveau moyen d'inférence.

Définition 3.16 Il suffit de choisir l'une des trois règles suivantes :

(i) Absurde

$$\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}$$

(ii) Double négation :

$$\dfrac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$$

(iii) Tiers-exclu :

$$\overline{\Gamma \vdash A \vee \neg A}$$

Ces trois règles sont équivalentes.

Remarque 3.7

- *Les preuves utilisant le tiers-exclu sont plus « difficiles » (car c'est un processus non déterministe de choisir la propriété).*
- *NK est plus fort que NJ (évident car on a rajouté des axiomes).*
- *encore plus fort : il existe des formules prouvables avec NK mais pas avec NJ.*

Traduction de NK vers NJ

NK permet de prouver davantage de tautologies que NJ.

Exemple 3.4 Exemples classiques :

$$\vdash_{\mathrm{NK}} : (A \wedge B) \Leftrightarrow \neg\,(\neg A \vee \neg B)$$
$$\vdash_{\mathrm{NK}} : (A \Rightarrow B) \Leftrightarrow (\neg A \vee B)$$
$$\vdash_{\mathrm{NK}} : (A \Rightarrow B) \Leftrightarrow (\neg A \wedge B)$$

On peut remarquer que souvent la logique NK n'est nécessaire que dans un seul sens.

Définition 3.17 Soient $\mathcal{L}$ et $\mathcal{L}'$ deux logiques, $\mathcal{L}$ plus forte que $\mathcal{L}'$. φ une application qui à toute formule F de $\mathcal{L}$ associe une formule $\varphi(F)$ de $\mathcal{L}'$ φ est une traduction de $\mathcal{L}$ dans $\mathcal{L}'$ si pour toute formule F de $\mathcal{L}$

$$\vdash_{\mathcal{L}} F \quad \text{implique} \quad \vdash_{\mathcal{L}'} \varphi(F)$$
$$\vdash_{\mathcal{L}} F \equiv \varphi(F)$$

Traduction de NK dans NJ (Glivenko 1929)

$$\varphi(F) = \neg\neg F$$

Il faudra attendre Gödel pour avoir le résultat avec les symboles d'équivalence et d'existence.

Chapitre 4

Calcul des prédicats

4.1 Syntaxe

Définition 4.1 Un langage du premier ordre est la donnée d'un ensemble de symboles L qui se compose en deux parties :

- La partie commune à tous les langages : un ensemble dénombrable de symboles de variables V, $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$, (,), $\forall$ et $\exists$.
- La partie spécifique à chaque langage : un ensemble C de symboles de constantes, deux suites (F_n) et (R_n) de symboles de fonctions et de prédicats d'arité n.

Définition 4.2 L'ensemble $\tau(L)$ des termes du langage L est défini inductivement par

- $C \cup V \subset \tau(L)$
- Si $f \in F_n$ et $t_1, \ldots, t_n \in \tau(L)$, $ft_1 \ldots t_n \in \tau(L)$.

Exemple 4.1 $L = \{c_{(0)}, f_{(1)}, g_{(3)}\}$,

$$M = ggfffv_0gv_2v_0cfcfffgfcgv_2fv_0ffcfcfc \in \tau(L)$$

Proposition 4.1 La définition des termes est non ambiguë.

Définition 4.3 Un terme sans occurrence de symbole de variable est un terme clos. On note $t[v_{i_1}, \ldots, v_{i_n}]$ pour indiquer que toutes les variables ayant une occurrence dans t sont parmi $\{v_{i_1}, \ldots, v_{i_n}\}$.

Définition 4.4 Soit $t \in \tau(L)$, $v_1, \ldots, v_k \in V$ et $t_1, \ldots, t_k \in \tau(L)$. On définit $t[\frac{t_1}{v_1}, \ldots, \frac{t_k}{v_k}]$ la substitution simultanée des v_i par les t_i par induction sur t :

- Si $t \in C$, $t[\frac{t_i}{v_i}] = t$.
- Si $t = v_i$, $t[\frac{t_i}{v_i}] = t_i$.
- Si $t \in V$, $t[\frac{t_i}{v_i}] = t$.
- Si $t = fu_1 \ldots u_n$ alors $t[\frac{t_i}{v_i}] = fu_1[\frac{t_i}{v_i}] \ldots u_n[\frac{t_i}{v_i}]$.

Définition 4.5 Une formule atomique $M \in L^*$ est de la forme $M = Rt_1 \dots t_n$ où $R \in R_n$ et $t_1, \dots, t_n \in \tau(L)$.

Définition 4.6 L'ensemble $\mathcal{F}$ des formules du premier ordre est construit inductivement :

- $\mathcal{F}$ contient les formules atomiques
- Si $f, g \in \mathcal{F}$ alors $(f \wedge g)$, ... appartiennent à $\mathcal{F}$
- Si $f \in \mathcal{F}$, pour tout $v \in V$, $\forall v f$ et $\exists v f$ appartiennent à $\mathcal{F}$.

C'est une définition est non ambiguë

Définition 4.7 Les occurrences libres d'une variable v dans une formule F sont celles qui ne sont pas sous la portée d'un quantificateur. On dit qu'une variable est libre ssi elle a une occurrence libre.

Une formule close est une formule sans variable libre. On notera

$$F[v_{i_1}, \dots, v_{i_n}]$$

le fait que les variables libres de F sont parmi $\{v_{i_1}, \dots, v_{i_n}\}$.

La clôture universelle de $F[v_{i_1}, \dots, v_{i_n}]$ est $\forall v_{i_1} \dots \forall v_{i_n} F$.

Définition 4.8 On définit la substitution de termes à des occurrences libres de variables dans les formules par induction. Le seul cas posant problème est celui de $\forall$ et $\exists$: si $\square$ est un quantificateur, et $F = \square w G$, $F[\frac{t}{v}] = \square w G$ si $v = w$ et $\square w G[\frac{t}{v}]$ sinon.

4.2 Sémantique

Définition 4.9 Une structure $\mathcal{M}$ sur un langage L est la donnée de

- un ensemble M non vide (domaine)
- un élément $c^{\mathcal{M}}$ de M pour chaque $c \in C$
- pour chaque $f \in F_k$, une application $\Gamma^{\mathcal{M}} : M^k \to M$.
- pour chaque $r \in R_k$, un sous ensemble $R^{\mathcal{M}}$ de M^k.

Définition 4.10 Une valuation est une application $\rho : V \to M$. On note $\rho[v \to d]$ une valuation ρ' telle que $\rho'(w) = \rho(w)$ si $v \neq w$ et $\rho'(v) = d$.

Définition 4.11 La valeur du terme t dans la structure $\mathcal{M}$ par rapport à la valuation ρ, notée $[\![r]\!]_\rho^{\mathcal{M}}$ est défini par induction sur t :

- Si $t \in C$, $[\![t]\!]_\rho^{\mathcal{M}} = c^{\mathcal{M}}$
- Si $t \in V$, $[\![t]\!]_\rho^{\mathcal{M}} = \rho(t)$
- Si $t = f t_1 \dots t_n$, $[\![t]\!]_\rho^{\mathcal{M}} = f^{\mathcal{M}}([\![t_1]\!]_\rho^{\mathcal{M}}, \dots, [\![t_n]\!]_\rho^{\mathcal{M}})$

Définition 4.12 Si F est une formule, $[\![F]\!]_\rho^{\mathcal{M}} \in \{0, 1\}$ est défini inductivement sur F par :

- Si $F = Rt_1 \dots t_n$, $[\![F]\!]_\rho^{\mathcal{M}} = 1$ ssi $([\![F]\!]_\rho^{\mathcal{M}}, \dots, [\![F]\!]_\rho^{\mathcal{M}}) \in R^{\mathbb{M}}$

- connecteurs propositionnels
- si $F = \forall v G$, $[\![F]\!]^{\mathcal{M}}_{\rho} = 1$ ssi $[\![G]\!]^{\mathcal{M}}_{\rho[v\to d]} = 1$ pour tout $d \in M$.
- si $F = \exists v G$, $[\![F]\!]^{\mathcal{M}}_{\rho} = 1$ ssi $[\![G]\!]^{\mathcal{M}}_{\rho[v\to d]} = 1$ pour au moins un $d \in M$.

Si F est une formule close, on dira que $\mathcal{M}$ est un modèle de F et on notera $\mathcal{M} \vDash F$ pour $[\![F]\!]^{\mathcal{M}}_{\rho} = 1$ (ρ n'a pas d'importance).

Définition 4.13 Soit $t[w_1, \dots, w_n]$ un terme, $F[v, w_1, \dots, w_n, u_1, \dots, u_p]$ une formule. La substitution $F[\frac{t}{v}]$ est licite ssi aucune occurrence libre de v ne se trouve sous un quantificateur $\forall w_i$ ou $\exists w_i$.

Proposition 4.2 Si la substitution est licite, $[\![F[\frac{t}{v}]]\!]^{\mathcal{M}}_{\rho} = [\![F]\!]^{\mathcal{M}}_{\rho[v\to[\![t]\!]^{\mathcal{M}}_{\rho}]}$.

Définition 4.14

- Une formule close est universellement valide ssi elle est satisfaite dans toute structure. On note $\vDash^* F$.
- Une formule (close) F est contradictoire ssi $\vDash^* \neg F$.
- Deux formules (closes) sont équivalentes ssi $\vDash^* F \Leftrightarrow G$.
- Une formule est universellement valide ssi sa clôture universelle est universellement valide.

Proposition 4.3 Si $F \equiv F'$, $G \equiv G'$ alors $\forall v F \equiv \forall v F'$, $\exists v F \equiv \exists v F'$, $F \circ G \equiv F' \circ G'$ et $\neg F \equiv \neg F'$.

Proposition 4.4 Si w n'a aucune occurrence dans F alors $\forall v F$ et $\forall w F[w/v]$ sont équivalentes (idem avec $\exists$)

Proposition 4.5 Toute formule du premier ordre est universellement équivalente à une formule ne contenant pas de symbole de quantificateurs ou de connecteurs autres que $\neg, \vee$ et $\exists$.

Définition 4.15 F est prénexe ssi il existe k, $v_1, \dots, v_k \in V$, $Q_1, \dots, Q_k$ des quantificateurs et G une formule sans quantificateurs tels que $F = Q_1 v_1 \dots Q_k v_k G$. (ie les quantificateurs sont au début)

On dit qu'une formule prénexe est polie ssi il y a au plus une occurrence de chaque variable dans le préfixe.

THÉORÈME 4.1 *Toute formule admet une forme prénexe polie.*

Démonstration. Par induction sur les formules.

Si $G \equiv Q_1 v_1 \dots Q_k v_k G''$ et $H \equiv Q'_1 v'_1 \dots Q'_{k'} v'_{k'} H''$.

On prend $k + k'$ variables distinctes $u_1, \dots, u_{k+k'}$ sans occurrence dans G et H. On substitue :

$$G_1 \equiv G''[u_i/v_i] \text{ et } G_2 \equiv H_1 \equiv H''[u_{k+i}/v_i]$$

et on a $G \circ H \equiv Q_1 u_1 \dots Q_k u_k Q'_1 u_{k+1} \dots Q'_{k'} u_{k+k'} (G_1 \circ H_1)$. ∎

4.3 Formes de Skolem

Si G est sous FNC ou FND, on dit que F est sous forme préfixe conjonctive ou disjonctive.

On part d'une formeule prénexe polie et on va faire disparaître les $\exists$ en ajoutant des nouvelles fonctions au langage.

Dès qu'on voit un $\exists v$ avec k $\forall x_i$ avant, on ajoute une fonction (de Skolem) f d'arité k et on remplace chaque occurrence de v par $f(x_i)$.

Exemple 4.2 $\forall x_1 \forall x_2 \exists y_1 \forall x_3 \exists y_2 G$ devient

$$\forall x_1 \forall x_2 \forall x_3 G[y_1/f_1(x_1, x_2), y_2/f_2(x_1, x_2, x_3)]$$

Définition 4.16 Ainsi, à F on associe $L_{Sk}(F)$, enrichissement de L par p symbiles. La formule obtenue est appellée forme de Skolem de F, notée F_{Sk}. On généralise au cas où F est une formule quelconque en prenant la forme prénexe de F.

Remarque 4.1 *F et F_{Sk} ne sont pas toujours universellement équivalentes. Par exemple, si $F = \forall x \exists y Pxy$, $F_{Sk} = \forall x Pxfx$.*

Pour le modèle $M = (\mathbb{Z}, P = \leqslant, f = n - -)$, $M \vDash F$ mais $M \nvDash F_{Sk}$.

Lemme 4.1.1
Soit $v_1, \ldots, v_n$ des variables deux à deux distinctes et $F = F[v_1, \ldots, v_n]$ une formule prénexe polie. Alors $F_{S_k} \Rightarrow F$ est universellement valide du langage $L_{Sk}(F)$.

THÉORÈME 4.2 *Une formule close admet un modèle ssi l'une quelcoqnue de ses formes de Skolem admet un modèle.*

Lemme 4.2.1
Avec les hypothèses précédentes et $\mathcal{M} = (M, \ldots)$ une structure pour L. Soit ρ une valuation telle que $[\![F]\!]_\rho^{\mathcal{M}} = 1$.

Il est possible d'enrichir $\mathcal{M}$ en une structure $\mathcal{M}_{Sk}$ de $L_{Sk}(F)$ tel que $[\![F_{Sk}]\!]_\rho^{\mathcal{M}_{Sk}} = 1$.

Démonstration. On fait une récurrence sur le nombre de fonctions de Skolem.

Supposons qu'on soit arrivés à

$$F = \forall x_1 \ldots \forall x_n \exists x G[v_1, \ldots, v_m, x_1, \ldots, x_n, x]$$

avec k symboles $\exists$ dans G. On fixe des $v_1, \ldots, v_n$ aux valeurs $b_1, \ldots, b_n$ dans ρ.

$$\{b \in M, [\![G]\!]_{\rho[x_i \to a_i, x \to b]}^{\mathcal{M}} = 1\}$$

est non vide pour un n-uplet $(a_1, \ldots, a_m)$ donc on peut choisir un élément de cet ensemble, ce qui nous permet de définir une application $\varphi : M^m \to M$ interprétation de la fonction de Skolem au rang $k + 1$. ∎

4.4 Théories

Définition 4.17 Une théorie T est un ensemble de formules closes.

Définition 4.18 Soit M une structure. M est une modèle de T (noté $M \vDash T$) ssi pour toute $F \in T$, $M \vDash F$.

Définition 4.19 Une théorie est dite consistante si elle admet au moins un modèle, contradictoire sinon.

Définition 4.20 Une formule close F est conséquence de T ssi tout modèle de T est un modèle de F. On note $T \vDash^* F$.

Si F est non close, on prend sa clôture universelle.

Proposition 4.6 $T \vDash^* F$ ssi $T \cup \{\neg F\}$ est contradictoire.

Proposition 4.7 T est contradictoire ssi il existe une formule universellement valide F telle que $T \vDash^* \neg F$.

Définition 4.21 T et T' sont équivalentes ssi elles admettent les mêmes modèles.

Proposition 4.8 T est équivalente à la théorie vide ssi toutes les formules de T sont universellement valides.

Chapitre 5

Déduction naturelle

Un exemple de sytème de déduction est donné par les règles de la logique classique et on ajoute : Si x n'est pas libre dans Γ ni dans G :

$$\frac{\Gamma \vdash F[x]}{\Gamma \vdash \forall x F[x]} \text{intro}\forall$$

$$\frac{\Gamma \vdash \exists x F[x], (\Gamma, F[x]) \vdash G}{\Gamma \vdash G} \text{élim}\exists$$

Si t est un terme tel que la substitution est licite :

$$\frac{\Gamma \vdash \forall x, F[x]}{\Gamma \vdash F[t/x]} \text{élim}\forall$$

$$\frac{\Gamma \vdash F[t/x]}{\Gamma \vdash \exists x, F[x]} \text{intro}\exists$$

Exemple 5.1 Si $\Gamma \vdash \exists x \neg F$ alors $\Gamma \vdash \neg \forall x F$. On pose pour alléger $\Gamma_1 = (\Gamma, \neg F[x], \forall x F[x])$.

$$\frac{\overline{\Gamma \vdash \exists x \neg F}, \dfrac{\overline{\Gamma_1 \vdash \neg F[x]}, \dfrac{\overline{\Gamma_1 \vdash \forall x F[x]}}{\Gamma_1 \vdash F[x]} \text{élim}\forall}{(\Gamma, \neg F[x]) \vdash \neg \forall x F} \text{intro}\neg}{\Gamma \vdash \neg \forall x F} \text{élim}\exists$$

Sémantique	Déduction
T consistante	T cohérente
$T \vDash F$ ssi $T \cup \{\neg F\}$ inconsistante	$T \vdash F$ ssi $T \cup \neg F$ incohérente
T consistante ssi toute partie de T l'est	Si toute partie de T est cohérentes, T l'est
$T \vDash F$	$T \vdash F$

Proposition 5.1 Si $\Gamma \vdash F$ alors il existe une partie finie Γ_0 de Γ telle que $\Gamma_0 \vdash F$.

Définition 5.1 Une théorie T est dite cohérente ssi il n'existe pas de F telle que $T \vdash F$ et $T \vdash \neg F$.

Proposition 5.2 $T \vdash F$ ssi $T \cup \{\neg F\}$ est incohérente.

Démonstration.
$\Rightarrow$ Si $T \vdash F$ alors $T, \neg F \vdash \neg F$ et $T, \neg F \vdash F$ donc $T \cup \{\neg F\}$ est incohérente.
$\Leftarrow$

$$\dfrac{\dfrac{\overline{T, \neg F \vdash G} \quad \overline{T, \neg F \vdash \neg G}}{T \vdash \neg\neg F}}{T \vdash F}$$

■

THÉORÈME 5.1 *Si T est une théorie dont toutes les parties sont cohérentes alors T est cohérente.*

Démonstration. Si T est incohérente, il existe F tel que $T \vdash F$ et $T \vdash \neg F$.
Donc $T \vdash F \wedge \neg F$. Il existe donc T_0 fini tel que $T_0 \vdash (F \wedge \neg F)$. ■

5.1 Validité de la déduction

THÉORÈME 5.2 *Soit T une théorie, F une formule, F' une clôture universelle de F. On a*

(i) *Si $T \vdash F$ alors $T \vdash^* F$*

(ii) *Si $\vdash F$ alors F' est universellement valide.*

Démonstration. Par induction sur la structure de dérivation $T \vdash F$, on montre que si ρ est une valuation satisfaisant toutes les formules de Γ alors ρ satisfait F pour une structure donnée. ■

COROLLAIRE 5.1 *Si T a un modèle alors T est cohérente.*

5.2 Complétude

Définition 5.2 Soit T une théorie. T est syntaxiquement complète ssi elle est cohérente et pour toute formule close F, $T \vdash F$ ou $T \vdash \neg F$.

Remarque 5.1 La complétude assure que toute formule close est vraie ou fausse et qu'on peut le prouver.

5.3 Témoins de Henkin

Définition 5.3 Une théorie T sur un langage L est dite de Henkin ssi pour toute formule à une variable libre $F[x]$, il existe un symbole de constante $c \in L$ tel que si $T \vdash \exists x F[x]$ alors $T \vdash T[c/x]$.

THÉORÈME 5.3 *Si T est une théorie de Henkin sur L syntaxiquement complète alors T admet un modèle.*

Démonstration. On construit une structure $\mathcal{M}$:

- Ensemble de base : l'ensemble des termes clos de L.
- Les constantes sont interprétées par elles-mêmes
- Si f est un symbole de fonction d'arité n, $f^{\mathcal{M}}$ est une application de $H^n \to H$ qui à $(t_1, \dots, t_n)$ associe $f t_1 \dots t_n$.
- Si R est un symbole de prédicat d'arité n alors $R^{\mathcal{M}} t_1 \dots t_n$ ssi $T \vdash R t_1 \dots t_n$.

Il faut alors vérifier que $\mathcal{M}$ est un modèle de T, ie de toute formule de T. Par induction sur F close dans T, on montre que $\mathcal{M} \vDash F$ ssi $T \vdash F$.

- Si F est atomique close, c'est bon
- On se restreint à $\neg$, $\wedge$ et $\exists$. Supposons le résultat vrai pour F et G.
  $T \vdash \neg F$ ssi $T \nvdash F$ (syntaxiquement complète) ssi $\mathcal{M} \nvDash F$ ssi $\mathcal{M} \vDash \neg F$.
  $T \vdash (F \wedge G)$ ssi $T \vdash F$ et $T \vdash G$ ssi $\mathcal{M} \vDash F$ et $\mathcal{M} \vDash G$ ssi $\mathcal{M} \vDash F \wedge G$.
  Si $T \vdash \exists x F$, il existe c tel que $T \vdash F[c/x]$ donc $\mathcal{M} \vDash F(c)$ par hypothèse d'induction, mais c est une constante qui s'interprète par elle-même dans la structure donc $\mathcal{M} \vDash \exists x F[x]$.
  Réciproquement, si $\mathcal{M} \vDash \exists x F[x]$, il existe un terme clos t tel que $[\![F]\!]^{\mathcal{M}}_{\rho[x \to t]} = 1$ donc $\mathcal{M} \vDash F[t/x]$ donc $T \vdash F[t/x]$ donc $T \vdash \exists x F$. ∎

THÉORÈME 5.4 *Pour toute théorie cohérente sur un langage L, il existe un unique langage $L' \supset L$ et T' théorie sur L' qui contient T telle que T' soit une théorie de Henkin syntaxiquement complète sur L'.*

Démonstration. L' est défini à partir de L en ajoutant un nombre dénombrable de constantes C' : une par formule dans la théorie. L est dénombrable donc on peut énumérer toutes les formules closes de L' : F_n. On construit alors la suite croissante de théories T_n définie par $T_0 = T$ et en posant $G_n = F_n$ si $T_n \cup \{F_n\}$ est cohérente et $G_n = \neg F_n$ sinon.

Si G_n n'est pas de la forme $\exists x G'$, on pose $T_{n+1} = T_n \cup \{G_n\}$, sinon on choisit un symbole de constante $c \in C$ qui n'apparaît dans aucune formule de $T_n \cup \{G_n\}$. Dans ce cas, on pose $T_{n+1} = T_n \cup \{G_n, G_n[c/x]\}$.

Pour tout n, T_n est cohérente,$T_n \subset T_{n+1}$ et $T_n \setminus T$ est fini.

De plus, $F_n \in T_{n+1}$ ou $\neg F_n \in T_{n+1}$. Si F est de la forme $\exists x G'$ et $F \in T_n$ alors il existe c tel que $F[c/x] \in T_n$.

On pose alors $T' = \bigcup_{n\in\mathbb{N}} T_n$. T' est cohérente car toutes ses parties le sont. T' est syntaxiquement correcte. T' est de Henkin par construction. ■

Remarque 5.2 On devrait faire une récurrence plutot que de faire une induction.

THÉORÈME 5.5 COMPACITÉ AU PREMIER ORDRE *Si T est une théorie dont toute partie finie a un modèle alors T a un modèle.*

Démonstration. Soit T_0 une partie finie de T. T_0 a un modèle par hypothèse dont T_0 est cohérente donc T est cohérente donc T a un modèle. ■

5.4 Théories

Définition 5.4 Une théorie est un ensemble de formules closes appelées axiomes.

Définition 5.5 L'ensemble $\mathrm{Thm}(T)$ (ou théorie engendrée par T) est l'ensemble des formules F tels que $T \vdash F$.

Définition 5.6 T est récursive ssi l'ensemble des formules de T est récursif.

Définition 5.7 T est décidable ssi $\mathrm{Thm}(R)$ est récursif.

5.4.1 Calcul des prédicats

On prend la théorie sans axiomes, engendrée par $\varnothing$. $\varnothing$ est consistante et récursive.

5.4.2 Théorie de l'égalité

Quand R_2 contient un symbole de relation binaire. On prend les axiomes

- $A^1 : \forall x, x = x$
- $A^2 : \forall x \forall y, x = y \Rightarrow y = x$
- $A^3 : \forall x \forall y, \forall z, x = y \wedge y = z \Rightarrow x = z$
- $A^f : \forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n, x_1 = y_1 \wedge \ldots \wedge x_n = y_n \Rightarrow f x_1 \ldots x_n = f y_1 \ldots y_n$.
- $A^R : \forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n, x_1 = y_1 \wedge \ldots \wedge x_n = y_n \Rightarrow (R x_1 \ldots x_n \Rightarrow R y_1 \ldots y_n)$.

Pour simplifier, on pourra déplacer ces axiomes dans les règles de déduction.

THÉORÈME 5.6 CHURCH *Toute théorie contenant l'égalité et au moins un autre prédicat binaire est indécidable.*

5.4.3 Arithmétique

On considère $L = \{0, S, +, \times, <, =\}$ avec les axiomes P_0

- $\forall x, \neg(Sx = 0)$
- $\forall x \exists y \neg(x = 0) \Rightarrow Sy = x$
- $\forall x \forall y (Sx = Sy) \Rightarrow x = y$
- $\forall x, x + 0 = x$
- $\forall x \forall y x + Sy = S(x + y)$
- $\forall x, x \times 0 = 0$
- $\forall x \forall y, x \times Sy = x + (x \times y)$
- $\forall x, \neg(x < 0)$
- $\forall x \forall y, x < Sy \Leftrightarrow (w < y \wedge x = y)$

On ajoute l'axiome de récurrence : pour toute formule F et toute variable libre x,

$$(F[0/x] \wedge \forall y (F[y/x] \Rightarrow F[Sy/x])) \Rightarrow \forall x F$$

On note $P = P_0 \cup \{\text{récurrence}\}$. Le modèle standard de P est $\mathbb{N}$.

Proposition 5.3 Il existe des modèles non standard.

Démonstration. Pour $n \in \mathbb{N}$, on note $\overline{n} = S \dots S0$ (S apparaît n fois). On ajoute un symbole de constante c au langage.

On pose $T = P \cup \{\neg(c = \overline{n}), n \in \mathbb{N}\}$.

Soit T_0 une partie finie de T. Il existe N tel que $T_0 \subset P \cup \{\neg(c = n), n \leqslant N\}$. On peut interpréter c par un entier plus grand que N. T_0 admet donc un modèle.

Ainsi, T admet un modèle. ∎

5.4.4 Décidabilité

THÉORÈME 5.7 *Une théorie syntaxiquement complète récursive est décidable.*

Démonstration. Si T est récursive, $\mathrm{Thm}(T)$ est RE et si T est récursive complète, $\mathrm{Thm}(T)^c$ est RE donc $\mathrm{Thm}(T)$ est récursif. ∎

THÉORÈME 5.8 *Une théorie consistante contenant P est indécidable.*

COROLLAIRE 5.2 GÖDEL *Une théorie récursive consistante contenant P_0 est incomplète.*

5.4.5 Axiomatisation finie

THÉORÈME 5.9 *L'arithmétique de Peano n'est pas finiment axiomatisable.*

Définition 5.8 Soit L un langage du premier ordre et $\mathbb{P}$ une propriété sur les structures de L.

$\mathbb{P}$ est axiomatisable ssi il existe une théorie T de L telle que $\mathcal{M}$ vérifie $\mathbb{P}$ ssi $\mathcal{M} \vDash T$.

Proposition 5.4 La propriété « être un ensemble à au moins n éléments » est finiment axiomatisable.

Démonstration. On prend $F_n = \exists x_1 \dots \exists x_n \bigwedge_{i<j} \neg(x_i = x_j)$. ■

Proposition 5.5 La propriété « être un ensemble à exactement n éléments » est finiment axiomatisable.

Démonstration. On prend $F_n \wedge \neg F_{n+1}$. ■

Proposition 5.6 La propriété « être un ensemble fini » n'est pas (finiment) axiomatisable.

Démonstration. Par l'absurde si T est (finie) et vérifie $\mathcal{M}$ fini ssi $\mathcal{M} \vDash T$.

$T \cup \{F_n, n \in \mathbb{N}^*\}$ est une théorie infinie contradictoire donc il existe une sous-théorie T' finie contradictoire. Il existe N tel que $T' \subset T \cup \{F_n, n \leqslant N\} =: T_N$. T_N est contradictoire mais on sait lui trouver un modèle (les entiers de 1 à N). Absurde. ■

Ainsi, « être un ensemble infini » n'est pas axiomatisable, donc Peano non plus.

Chapitre 6

Résolution

6.1 Mise sous forme de clauses

On passe d'un ensemble de formules Σ à un ensemble S de clôtures universelles de clauses via

(i) mise sous forme prénexe normale conjonctive

(ii) mise sous forme de Skolem

(iii) distribution des $\forall$ sur les $\wedge$

(iv) décomposition en un ensemble de clôtures universelles de clauses

Exemple 6.1 Si L contient deux prédicats unaires P et Q et

$$\Sigma = \{(\exists Px \Rightarrow \forall yPy), \forall x(Px \vee Qx), \neg((\exists x \neg Qx) \Rightarrow (\forall yPy))\}$$

On obtient S :

$$S = \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \neg Qa, \neg Pb\}$$

Proposition 6.1 Σ admet un modèle ssi S admet un modèle.

6.2 Unification

6.2.1 Substitutions

Définition 6.1 Une substitution est une application $V \to T$. Le domaine d'une substitution α est $\{x, \alpha(x) \neq x\}$ et sera supposé fini. On prolonge les substitutions aux termes et aux formules.

Définition 6.2 Un ensemble fini de formules atomiques $\{A_1, \ldots, A_n\}$ est dit unifiable ssi il existe une substitution α telle que $\alpha A_1 = \ldots = \alpha A_n$.

Exemple 6.2 $\{Rxf(y), Rg(y,c)z, Rg(d,v)f(d)\}$ est unifiable via

$$[g(d,c)/x, d/y, f(d)/z, c/v]$$

Définition 6.3 Soit S un ensemble fini de couples non ordonnés de termes $\langle t_i, t'_i \rangle$.

Une substitution α est un unificateur de S ssi pour tout i, $\alpha t_i = \alpha t'_i$.

α est un unificateur principal de S ssi pour tout unificateur σ de S, il existe β tel que $\sigma = \beta \circ \alpha$.

Proposition 6.2 Deux unificateurs principaux sont égaux à permutation près des variables.

6.2.2 Algorithmes d'unification

Si $A = R(t_1, \ldots, t_n)$ et $B = R(t'_1, \ldots, t'_n)$, on pose $S = \{(t_i, t'_i)\}$.

Pour tout $(t, t') \in S$ tel que $t = fu_1 \ldots u_m$ et $t' = f'u'_1 \ldots u'_{m'}$.

- Si $f \neq f'$, S n'a pas d'unificateur
- Sinon, on remplace dans S les couples (t, t') par $\{(u_i, u'_i), i \in [\![1, m]\!]\}$.

Et on itère le procédé.

Cet algorithme termine car la hauteur des termes décroît à chaque itération.

On supprime les couples identiques et les couples (t, t). On note S' le système obtenu après simplification et réduction.

Proposition 6.3 S' et S ont les mêmes unificateurs.

Remarque 6.1 *S' est de hauteur nulle : tous les couples sont de la forme (constante,terme) ou (variable,terme).*

Définition 6.4 Les couples réductibles sont ceux de la forme (variable,terme) où la variable n'intervient pas dans le terme.

Les couples irréductibles sont donc les (x, t) avec x variable de t, $(c, fx_1 \ldots x_n)$ et (c, c') avec $c \neq c'$.

Proposition 6.4 Si S' contient un couple irréductible, il n'est pas unifiable.

On a donc un algorithme d'unification :

- On part de $\sigma = \varnothing$
- On construit S à partir de deux formules atomiques
- On réduit et simplifie S en S'
- Si S' contient un irréductible, on a fini et S n'est pas unifiable
- Sinon on choisit un couple (x, t) dans S' et on recommence avec $\sigma_1 = [t/x] \circ \sigma$ et $S_1 = \{(\sigma_1(x'), \sigma_1(t')), (x', t') \in S' \setminus \{(x, t)\}\}$ tant que $S_1 \neq \varnothing$.

THÉORÈME 6.1 *L'algorithme donne un unificateur principal.*

Démonstration. L'algorithme s'arrête car on a un nombre fini de variables et chaque itération en fait disparaître une.

Si S' possède un unificateur α alors on montre qu'il existe β unificateur de S_1 tel que $\alpha = \beta \circ [t/x]$. On pose $\beta(x) = x$ et $\beta(y) = \alpha(y)$ si $y \neq x$.

On a $\beta \circ [t/x](y) = \beta(y) = \alpha(y)$ et $\beta \circ [t/x](x) = \beta(t) = \alpha(t) = \alpha(x)$ car t n'apparaît pas dans x.

On conclut par récurrence sur le nombre de variables. ■

6.3 Résolution

Soient $C_1 = (\Gamma_1, \Delta_1)$ et $C_2 = (\Gamma_2, \Delta_2)$ deux clauses sans variables communes (c'est toujours possible car on travaille avec les clôtures universelles donc on peut renommer).

Définition 6.5 C est un résolvant de C_1 et C_2 ssi il existe $P_1 \subset \Delta_1$ et $N_2 \subset \Gamma_2$ non vides telles que $P_1 \cup N_2$ unifiable (via un unificateur principal σ) et tels qu'on ait $C = (\Gamma, \Delta)$ avec

$$\Gamma = \sigma(\Gamma_1 \cup (\Gamma_2 \setminus N_2)) \text{ et } \Delta = \sigma(\Delta_2 \cup (\Delta_1 \setminus P_1))$$

Exemple 6.3 $S = \{Px \vee Qx, \neg Qf(d), \neg Px \vee Py, \neq Pc\}$.

$$\dfrac{\dfrac{\dfrac{Px \vee Qx \quad \neg Qf(d)}{Pf(d)} \quad \neg Px \vee Py}{Py} \quad \neg Pc}{\square}$$

Lemme 6.1.1
Si C est un résolvant de C_1 et C_2 alors la clôture universelle de C notée $\forall C$ est conséquence sémantique de $\{\forall C_1, \forall C_2\}$.

Corollaire 6.1 *S'il existe une réfutation de S alors S n'a pas de modèle.*

Théorème 6.2 *Si S n'a pas de modèle alors il existe une réfutation de S.*

Démonstration. On a la complétude sur les clauses de type $p \wedge \neg q$.

On commence par construire l'ensemble de tous les prédicats appliqués à des termes clos (base de Herbrand).

Lemme 6.2.1
Une clause close F admet un modèle ssi elle admet un modèle de Herbrand.

F est de la forme $\forall x_1 \ldots \forall x_n G$, l'ensemble des instances de F est

$$\{G[t_1/x_1, \ldots, t_n/x_n], (t_1, \ldots, t_n) \text{ clos}\}$$

On interprête $Rt_1 \dots t_n$ comme une variable propositionnelle.

Dans le cadre du calcul propositionnel, par complétude de la coupure, on peut construire une réfutation par coupure de l'ensemble des instances. Cette réfutation utilise un nombre fini d'instances. On peut alors la transformer en une réfutation par résolution (dans le cadre du premier ordre). ■

