LOGIQUE MATHÉMATIQUE : INTRODUCTION

Arnaud Durand - Paul Rozière
Paris 7 – 31HU03MI

9 avril 2018

(*version de travail, ne pas hésiter à indiquer coquilles et erreurs, le polycopié est encore incomplet certaines parties seront développées*)

Chapitre 1

Calcul propositionnel

1.1 Syntaxe et sémantique de la logique propositionnelle

Il y a deux aspects principaux dans la description d'un langage : sa *syntaxe* et sa *sémantique*. Intuitivement, la syntaxe concerne la structure du langage et la forme de ses énoncés; la sémantique s'intéresse au "sens" (ici mathématique) à donner au langage.

1.1.1 Syntaxe

Soit un ensemble de constantes propositionnelles $\mathscr{P}$ (on parlera souvent de variables propositionnelles dans la suite). L'ensemble des formules du calcul propositionnel sur $\mathscr{P}$ — nous dirons formules propositionnelles sur $\mathscr{P}$ — que l'on notera $\mathfrak{F}_{\mathscr{P}}$, ou simplement $\mathfrak{F}$ s'il n'y a pas d'ambiguïté, est l'ensemble engendré *inductivement* et *librement* par les règles de construction suivantes :

atomes si $p \in \mathscr{P}$, p est une formule propositionnelle.

absurde $\bot$ est une formule propositionnelle.

négation Si F est une formule propositionnelle $\neg F$ est une formule propositionnelle.

implication Si F et G sont des formules propositionnelles $(F \to G)$ est une formule propositionnelle.

conjonction Si F et G sont des formules propositionnelles $(F \wedge G)$ est une formule propositionnelle.

disjonction Si F et G sont des formules propositionnelles $(F \vee G)$ est une formule propositionnelle.

Par *engendré inductivement*, on entend qu'une formule ne peut être obtenue que par l'une de ces règles de construction, soit par la règle initiale, soit par l'une des règles suivantes à partir de formules déjà construites. Cela revient à définir $\mathfrak{F}$ comme l'union $\bigcup_{n\in\mathbb{N}} \mathfrak{F}_n$ où :

— $\mathfrak{F}_0 = \mathscr{P}$ et,

— pour tout $n \in \mathbb{N}$, $\mathfrak{F}_{n+1} = \mathfrak{F}_n \cup \{\neg F : F \in \mathfrak{F}_n\} \cup \{(F \odot G) : F, G \in \mathfrak{F}_n, \odot \in \{\wedge, \vee \to\}\}$.

La *hauteur* d'une formule F est le plus petit $n \in \mathbb{N}$ tel que $F \in \mathfrak{F}_n$.

Pour démontrer des propriétés de l'ensemble des formules propositionnelles, on peut raisonner par récurrence sur la hauteur de la formule, mais on utilise plutôt le principe équivalent de *raisonnement par induction sur la structure* des formules.

Raisonnement par induction. Soit une propriété à vérifier sur l'ensemble de toutes les formules propositionnelles $\mathfrak{F}_{\mathscr{P}}$. Alors

— si la propriété est vérifiée pour toutes les variables propositionnelles et l'absurde,

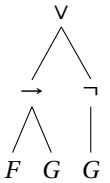
— si, quand elle est vérifiée pour F elle est encore vérifiée pour $\neg F$

— si, quand elle est vérifiée pour F et G elle est encore vérifiée pour $(F \to G)$, $(F \wedge G)$ et $(F \vee G)$

alors elle est vérifiée pour toutes les formules propositionnelle.

Ce principe de raisonnement est une conséquence directe de la définition inductive de l'ensemble des formules, et du fait qu'une formule ne peut être obtenue que par l'une des règles indiquée. On verra plus loin des utilisations du raisonnement par induction.

On peut représenter de façon arborescente la suite des constructions qui mène à une formule, c'est l'*arbre de décomposition de la formule*[1]. Par exemple la formule $((F \to G) \vee \neg G)$ a pour arbre de décomposition :



Par *engendré librement* on entend que l'on ne peut pas arriver à la même formule par deux constructions différentes, ou encore qu'une formule possède un seul arbre de décomposition. On exprime ceci plus précisément par la *propriété de lecture unique*.

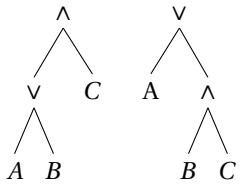
Propriété de lecture unique. Pour toute formule F de $\mathfrak{F}_{\mathscr{P}}$, exactement un des cas suivants est réalisé :

1. $F \in \mathscr{P}$
2. il existe unique $G \in \mathfrak{F}_{\mathscr{P}}$ telle que $F = \neg G$
3. il existe un unique $\odot \in \{\wedge, \vee \to\}$ et deux uniques $G, H \in \mathfrak{F}_{\mathscr{P}}$ tels que $F = (G \odot H)$.

1. On définit cette notion formellement plus loin en s'appuyant sur celle d'arbre orienté étiqueté

Une réalisation concrète. Une façon de réaliser concrètement l'ensemble des formules propositionnelles est de voir les formules propositionnelles comme des mots sur l'alphabet $\Sigma = \mathscr{P} \cup \{\wedge, \vee, \rightarrow,), (, \neg, \bot\}$, l'ensemble des mots sur Σ étant noté Σ^*. L'ensemble $\mathfrak{F}$ peut alors être réalisé comme le plus petit sous-ensemble de Σ^* contenant $\mathscr{P}$ et clos, pour tout mot F et G par les opérations $F \mapsto \neg F$, $(F,G) \mapsto (F \wedge G)$, $(F,G) \mapsto (F \vee G)$, $(F,G) \mapsto (F \rightarrow G)$. Là encore c'est une autre façon de dire que l'ensemble est défini inductivement par les règles ci-dessus. Pour que la réalisation soit correcte, il faut alors démontrer un *théorème de lecture unique* — la propriété de lecture unique ci-dessus — pour s'assurer que ce sous-ensemble de Σ^* est bien librement engendré.

Le parenthésage indiqué dans les opérations de clôture est utile pour montrer cette propriété, par exemple pour distinguer les formules $((A \vee B) \wedge C)$ et $(A \vee (B \wedge C))$ d'arbres de décomposition



On se rend compte cependant que les parenthèses extérieures peuvent toujours être omises, chose que l'on se permettra désormais. On pourrait également obtenir la lecture unique par des règles de priorité, comme pour les expressions arithmétiques où le signe $\times$ est prioritaire sur le signe $+$. On s'en abstiendra dans la suite (une convention courante est que le $\wedge$ est prioritaire sur les autres connecteurs). Une autre convention est de décider arbitrairement de décomposer une expression non parenthésée d'un certaine façon. Dans la suite on décide par exemple de lire $A \vee B \vee C$ comme $(A \vee B) \vee C$, donc $A \vee B \vee C \vee D$ comme $((A \vee B) \vee C) \vee D$, etc. De même pour le $\wedge$. On verra que cela est de peu d'importance car d'un point de vue sémantique ces connecteurs sont associatifs. Une convention parfois utilisée est de lire $A \rightarrow B \rightarrow C$ comme $A \rightarrow (B \rightarrow C)$, donc $A \rightarrow B \rightarrow C \rightarrow D$ comme $A \rightarrow (B \rightarrow (C \rightarrow D)$, etc. On verra pourtant que le connecteur n'est pas associatif et la convention est importante.

Toutes ces choses sont indispensables à prendre en compte si on s'intéresse à l'implémentation du calcul propositionnel sur machine. On doit définir une *grammaire* pour le calcul propositionnel, qui est une grammaire *context free* (indépendante du contexte). Il existe des outils spécialisés pour l'*analyse syntaxique* de ce genre d'expressions, soit reconnaître si un mot est bien une expression du langage (ici une formule propositionnelle), et dans ce cas de produire une représentation concrète de son arbre de décomposition.

Sous-formule, hauteur et arbre de décomposition. La propriété de lecture unique permet de montrer que les définitions par induction, comme celle des sous-formules ci-dessous, sont correctes. Ce sont l'analogue des définitions par récurrence pour les entiers naturels.

Définition 1.1.1 (sous-formule) *On définit inductivement l'ensemble* $\mathsf{sf}(F)$ *des sous-formules de* $F \in \mathfrak{F}$ *par :*

— *Si* $F \in \mathscr{P}$*, alors* $\mathsf{sf}(F) = \{F\}$
— *Si* F *s'écrit* $\neg G$*, alors* $\mathsf{sf}(F) = \mathsf{sf}(G) \cup \{F\}$ *;*
— *Si* F *s'écrit* $G \odot H$ *(avec* $\odot \in \{\wedge, \vee \rightarrow\}$*), alors* $\mathsf{sf}(F) = \mathsf{sf}(G) \cup \mathsf{sf}(H) \cup \{F\}$ *.*

La propriété de lecture unique est clairement nécessaire pour que ces cas soient disjoints et que la définition soit donc non ambiguë.

On montre facilement que (du fait du théorème de lecture unique) la hauteur h d'une formule (voir ci-dessus) peut se définir inductivement :

— si $F \in \mathscr{P}$, alors $\mathsf{h}(F) = 0$;
— si F s'écrit $\neg G$, alors $\mathsf{h}(F) = 1 + \mathsf{h}(G)$;
— si F s'écrit $G \odot H$ (avec $\odot \in \{\wedge, \vee \rightarrow\}$), alors $\mathsf{h}(F) = 1 + \sup(\mathsf{h}(F), \mathsf{h}(G))$.

Du fait de la propriété de lecture unique, une formule F possède un unique arbre de décomposition $\mathsf{arb}(F)$, que l'on peut définir de la façon suivante.

Définition 1.1.2 (arbre de décomposition) *On définit inductivement l'arbre de décomposition* $\mathsf{arb}(F)$ *associé à* F *par :*

1. *si* $F \in \mathscr{P}$*, alors* $\mathsf{arb}(F)$ *se réduit à un point étiqueté* F *;*
2. *si* $F = \neg G$*, alors* $\mathsf{arb}(F)$ *est l'arbre de racine étiquetée par* $\neg$ *ayant pour unique sous-arbre* $\mathsf{arb}(G)$ *;*

3. $F = (G \odot H)$, *alors* $\mathsf{arb}(F)$ *est l'arbre de racine étiquetée par* $\odot$, *ayant pour sous-arbre gauche* $\mathsf{arb}(G)$ *et pour sous-arbre droit* $\mathsf{arb}(H)$ *;*

Noter la disparition du parenthésage qui n'a plus lieu d'être dans un arbre orienté. On identifiera souvent F avec son arbre dans la suite. Noter que chaque sous-arbre de $\mathsf{arb}(F)$ est l'arbre de décomposition d'une sous-formule de F et réciproquement chaque sous-formule de F admet pour arbre de décomposition un sous-arbre de $\mathsf{arb}(F)$.

Dans cette section, on ne s'est préoccupé que de syntaxe c'est à dire de la forme des énoncés du calcul propositionnel. Si l'on veut formaliser un problème dans ce langage, on prendra pour $\mathscr{P}$ un ensemble qui a une certaine structure mais on n'a pas besoin de connaître cette structure pour étudier la sémantique.

Notations. On utilisera également des notations supplémentaires, qui ne font pas partie directement de la syntaxe :

— $\top$ est une abréviation pour $(\bot \to \bot)$ (le "vrai") ;

— $(F \leftrightarrow G)$ est une abréviation pour $(F \to G) \wedge (G \to F)$ (l'équivalence).

On aurait tout aussi bien pu les ajouter à la syntaxe.

On introduit quelques notations qui sont des abréviations plus élaborées (les entiers interviennent, on changerait complètement la syntaxe en les y introduisant directement).

Si $A_1, \ldots, A_n$ sont des formules de $\mathfrak{F}$, $I = \{i_1, ..., i_k\} \subseteq \{1, ..., n\}$, alors on note

$$\bigwedge_{i=1}^{n} A_i = A_1 \wedge A_2 \wedge \cdots \wedge A_n \; ; \; \bigwedge_{j \in I} A_j = A_{i_1} \wedge A_{i_2} \wedge \cdots \wedge A_{i_k} \; ;$$

de même

$$\bigvee_{i=1}^{n} A_i = A_1 \vee A_2 \vee \cdots \vee A_n \; ; \; \bigvee_{j \in I} A_j = A_{i_1} \vee A_{i_2} \vee \cdots \vee A_{i_k} \; .$$

Une convention (assez arbitraire) d'association pour les connecteurs $\wedge$, $\vee$ a été a été vue au dessus. Elle permettrait une définition par récurrence de ces notations.

1.1.2 Sémantique : valuations, tables de vérité, ensemble de modèles

La seule chose qui nous intéresse est la valeur de vérité de l'interprétation des constantes propositionnnelles. Une constante propositionnelle est soit vraie, soit fausse. On appelle donc *valuation* une fonction de $\mathscr{P}$ dans $\{0,1\}$ (0 pour "Faux", 1 pour "Vrai").

On peut formuler la définition de l'interprétation d'une formule propositionnelle à l'aide des valuations. Étant donnée une valuation v de $\mathscr{P}$ dans $\{0,1\}$, on définit l'interprétation induite sur les formules du calcul propositionnel, que l'on note $\overline{v}$ et que l'on notera ensuite (abusivement) v, par induction sur la définition des formules propositionnelles :

atome $\overline{v}(p) = v(p)$.

absurde $\overline{v}(\bot) = 0$.

négation $\overline{v}(\neg G) = 1$ ssi $\overline{v}(G) = 0$.

conjonction $\overline{v}(G \wedge H) = 1$ ssi $\overline{v}(G) = 1$ et $\overline{v}(H) = 1$.

disjonction $\overline{v}(G \vee H) = 0$ ssi $\overline{v}(G) = 1$ et $\overline{v}(H) = 1$.

implication $\overline{v}(G \to H) = 0$ ssi $\overline{v}(G) = 1$ et $\overline{v}(H) = 0$.

De façon équivalente, on peut aussi définir $\overline{v}$ en utilisant les opérations usuelles "+" et "·" de $\mathbb{Z}/2\mathbb{Z}$:

négation $\overline{v}(\neg G) = 1 + \overline{v}(G)$.

conjonction $\overline{v}(G \wedge H) = \overline{v}(G) \cdot \overline{v}(H)$.

disjonction $\overline{v}(G \vee H) = \overline{v}(G) + \overline{v}(H) + \overline{v}(G) \cdot \overline{v}(H)$.

implication $\overline{v}(G \to H) = 1 + \overline{v}(G) + \overline{v}(G) \cdot \overline{v}(H)$.

On voit que pour chaque connecteur n-aire la sémantique est définie par une fonction de $\{0,1\}^n$ dans $\{0,1\}$. On a déjà décrit ces fonctions, de deux façons différentes. On peut également les décrire par des tableaux de 2^n lignes :

G	$\neg G$
0	1
1	0

G	H	$G \wedge H$	$G \vee H$	$G \rightarrow H$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	1

Il faut se persuader que pour chacun de ces connecteurs, qui sont empruntés à la langue courante, la sémantique dérive bien de la logique intuitive (non spécifiquement mathématique), mais simplifiée drastiquement par le fait que seul deux valeurs de vérité sont possibles (le tiers-exclu).

Il est clair que l'interprétation d'une formule F pour une valuation v ne dépend que des valeurs de v pour les constantes propositionnelles qui apparaissent effectivement dans F et qui sont donc en nombre fini.

Étant donnée une formule qui n'utilise que les constantes propositionnelles $p_1, \ldots, p_n$, la *table de vérité* de la formule F pour $p_1, \ldots, p_n$ est la fonction de l'ensemble des valuations sur $p_1, \ldots, p_n$, soit $\{0,1\}^{\{p_1,\ldots,p_n\}}$, dans $\{0,1\}$, qui à chaque valuation v associe $\overline{v}(F)$. L'ensemble des tables de vérités sur les variables propositionnelles $p_1, \ldots, p_n$ est donc $\{0,1\}^{\{0,1\}^{\{p_1,\ldots,p_n\}}}$.

On peut représenter unetable de vérité comme ci-dessus par un tableau de 2^n lignes. Par exemple voici un calcul de la table de vérité de la formule $p \leftrightarrow q$ définie par $(p \rightarrow q) \wedge (q \rightarrow p)$ (pour chacune des 4 valuations v on utilise bien la définition du prolongement $\overline{v}$) :

p	q	$p \rightarrow q$	$q \rightarrow p$	$p \leftrightarrow q$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	1	1	1

La table de vérité d'une formule décrit entièrement la sémantique d'une formule propositionnelle, puisqu'elle décrit toutes les interprétations possibles.

Définition 1.1.3 (satisfaisabilité, validité, équivalence)

— *Une formule propositionnelle F est* satisfaisable *s'il existe au moins une valuation v telle que $\overline{v}(F) = 1$. Par abus de langage, on dira qu'une telle valuation $\overline{v}$ est un modèle de F.*

— *Une formule propositionnelle F est valide (i.e. est une* tautologie*) si pour toute valuation v on a $\overline{v}(F) = 1$.*

— *Deux formules propositionnelles F et G sont équivalentes, on note $F \equiv G$, si elles sont satisfaites par les mêmes valuations :*

$$F \equiv G \text{ ssi pour toute valuation } v,\ \overline{v}(F) = \overline{v}(G).$$

On vérifie que :

$$F \equiv G \text{ ssi } F \leftrightarrow G \text{ est une tautologie,}$$

en utilisant la table de vérité de $\leftrightarrow$. On remarque également (c'est juste une reformulation de ce qui précède) que deux formules sont équivalentes si et seulement si elles ont les mêmes tables de vérité. On a aussi facilement le résultat suivant.

Proposition 1.1.4 ***(satisfaisabilité et validité)*** *Pour toute formule $F \in \mathfrak{F}$: F est satisfaisable si et seulement si $\neg F$ n'est pas valide.*

1.2 Exemples de formalisation en calcul propositionnel.

Il n'est pas très confortable de devoir formaliser en calcul propositionnel. Quand cela est possible on y gagne un langage beaucoup plus simple. Le langage s'avère néanmoins pratique pour formaliser un certain nombre de problèmes combinatoires quand l'espace des solutions est fini et de taille raisonnable. De plus on peut vérifier mécaniquement la satisfaisabilité d'une formule propositionnelle (donc la faisabilité d'un ensemble de contraintes qu'elle exprime). On verra ce qu'il en est plus tard de l'existence de méthodes qui soient efficaces (en théorie ou en pratique) pour tester la satisfaisabilité.

1.2.1 Contraintes de compatibilité ou d'exclusion

On considère le problème suivant. On dispose d'un ensemble de n produits chimiques que l'on doit ranger dans k conteneurs ($k \leqslant n$). On sait aussi que le stockage dans un même conteneur de certaines combinaisons de produits n'est pas possible (pour des raisons de sécurité). Ces contraintes sont données sous la forme d'un ensemble $\mathscr{L}$ de sous-ensembles de $\{1,...,n\}$ tel que tout $I = \{i_1,...,i_h\} \in \mathscr{L}$ s'interprète par : tous les produits $i_1,...,i_h$ de la liste I ne peuvent être ensemble dans le même conteneur.

On exprime petit à petit l'ensemble des contraintes du problème. Pour cela on va utiliser des variables propositionnelles $p(i,j)$, $i \leqslant n$, $j \leqslant k$ dont la signification intuitive est : *le produit i se trouve dans le conteneur j*.

— Chaque produit doit être rangé dans un conteneur (et un seul)

$$F = (\bigwedge_{i\leqslant n} \bigvee_{j\leqslant k} p(i,j)) \wedge (\bigwedge_{i\leqslant n} \bigwedge_{\substack{j,j'\leqslant k \\ j\neq j'}} \neg(p(i,j) \wedge p(i,j')))$$

— Le rangement doit respecter la liste d'incompatibilité :

$$G = \bigwedge_{I\in\mathscr{L}} \bigwedge_{j\leqslant k} \neg(\bigwedge_{i\in I} p(i,j)).$$

Il est clair que le problème de départ admet une solution (de rangement) si la formule $F \wedge G$ est satisfaisable [2]. Une valuation v telle que $v(F) = 1$ décrira une solution possible et l'ensemble des valuations satisfaisantes, l'ensemble des solutions possibles

1.2.2 Le Sudoku

Le jeu du Sudoku consiste à compléter une grille 9×9 partiellement remplie par des chiffres de 1 à 9 avec les contraintes qui suivent.

1. Chaque case contient un et un seul chiffre
2. Les chiffres de 1 à 9 apparaissent sur chacune des lignes. Autrement dit, aucune ligne ne contient deux fois le même chiffre.
3. Les chiffres de 1 à 9 apparaissent sur chacune des colonnes. Autrement dit, aucune colonne ne contient deux fois le même chiffre.
4. Les chiffres de 1 à 9 apparaissent dans chacun des sous-carrés 3×3 qui subdivisent la grille. Autrement dit, aucun de ces carrés ne contient deux fois le même chiffre.

On va modéliser les contraintes de ce jeu en calcul propositionnel. Du fait de la finitude des contraintes, celui ci s'y prête assez bien. Soit $p(i,j,k)$, avec $i,j,k \in \{1,...,9\}$, les variables propositionnelles désignant le fait que la case à l'intersection de la ligne i et de la colonne j contient k.

1. $\displaystyle\bigwedge_{1\leqslant i\leqslant 9} \bigwedge_{1\leqslant j\leqslant 9} \big(p(i,j,1) \vee p(i,j,2) \vee \ldots \vee p(i,j,k)) \wedge \bigwedge_{\substack{k,k'\leqslant 9 \\ k\neq k'}} (\neg p(i,j,k) \vee \neg p(i,j,k'))\big)$
2. $\displaystyle\bigwedge_{1\leqslant i\leqslant 9} \bigwedge_{1\leqslant j<j'\leqslant 9} \bigwedge_{1\leqslant k\leqslant 9} \neg p(i,j,k) \vee \neg p(i,j',k)$
3. $\displaystyle\bigwedge_{1\leqslant j\leqslant 9} \bigwedge_{1\leqslant i<i'\leqslant 9} \bigwedge_{1\leqslant k\leqslant 9} \neg p(i,j,k) \vee \neg p(i',j,k)$
4. $\displaystyle\bigwedge_{i\in\{1,4,7\}} \bigwedge_{j\in\{1,4,7\}} \bigwedge_{(i',j')\neq(i'',j'')\in I} \bigwedge_{1\leqslant k\leqslant 9} \neg p(i+i',j+j',k) \vee \neg p(i+i'',j+j'',k)$
   où $I = \{(i,j) : i,j \in \{0,1,2\}\}$

Bien entendu, le jeu n'a de sens que si la grille est partiellement remplie au départ. Il faut donc ajouter un certain nombre de contraintes (que l'on dira *unitaire*) de la forme $p(i,j,k)$ pour indiquer que le nombre k se trouve, au départ, sur la case d'indice (i,j).

2. Le prouver formellement

1.3 Retour sur la sémantique : ensemble de modèles et fonctions Booléennes

On reprend l'idée d'associer à une formule propositionnelle F l'ensemble de ses modèles c'est à dire l'ensemble des valuations v telles que $\overline{v}(F) = 1$. Soit $\mathscr{P}$ un ensemble de variables propositionnelles, on définit la fonction

$$\mathsf{mod} : \mathfrak{F} \longrightarrow \{0,1\}^{\{0,1\}^{\mathscr{P}}}$$

de la façon suivante :

— Si $F \in \mathscr{P}$, $\mathsf{mod}(F) = \{v : \overline{v}(F) = 1\}$
— Si $F = \neg G$, $\mathsf{mod}(F) = \{0,1\}^{\mathscr{P}} \backslash \mathsf{mod}(G)$.
— Si $F = G \vee H$, $\mathsf{mod}(F) = \mathsf{mod}(G) \cup \mathsf{mod}(H)$.
— Si $F = G \wedge H$, $\mathsf{mod}(F) = \mathsf{mod}(G) \cap \mathsf{mod}(H)$.
— Si $F = G \rightarrow H$, $\mathsf{mod}(F) = (\{0,1\}^{\mathscr{P}} \backslash \mathsf{mod}(G)) \cup \mathsf{mod}(H)$.

La fonction mod associe à tout $F \in \mathfrak{F}$ l'ensemble de ses modèles. On peut montrer facilement par induction le résultat suivant.

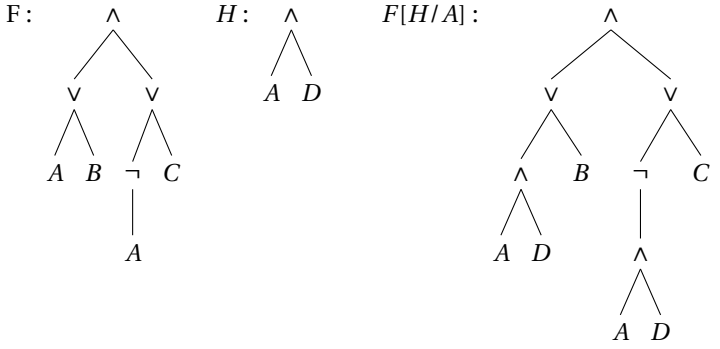
Proposition 1.3.1 *Pour toute formule $F \in \mathfrak{F}$, $\mathsf{mod}(F) = \{v : \overline{v}(F) = 1\}$.*

Une *fonction Booléenne* en n variables est une fonction de $\mathbb{B}^n = \{0,1\}^n$ dans $\mathbb{B} = \{0,1\}$ [3]. Les fonctions Booléennes jouent un rôle important dans de nombreux domaines de l'informatique (aide à la décision, recherche opérationnelle, contraintes, codes, cryptographie, ...) et de l'électronique. L'ensemble des valuations v de $\mathscr{P} = \{x_1, ..., x_n\}$ dans $\{0,1\}$ est en bijection avec $\mathbb{B}^n$ et donc, pour tout $F \in \mathfrak{F}$, mod$F$ est une fonction Booléenne que l'on dira *représentée* par F.

1.4 Quelques équivalences logiques et tautologies usuelles.

La substitution simultanée de formules propositionnelles $H_1, \ldots, H_n$ aux variables propositionnelles $p_1, \ldots, p_n$ d'une formule propositionnelle F s'écrit $F[H_1/p_1, \ldots, H_n/p_n]$ (F dans laquelle on remplace simultanément [4] la variable p_1 par H_1, p_2 par H_2 etc.) se définit par induction sur F sans difficultés.

L'exemple suivant montre une telle subsitution.



La propriété de substitution énoncée ci-dessous permet de voir les tautologies comme des schémas de formules propositionnelles valides. Par exemple On sait que $p \rightarrow p$ est une tautologie, on en déduit immédiatement que $(p \rightarrow q) \rightarrow (p \rightarrow q)$, est une tautologie.

Proposition 1.4.1 (substitution) *Soit F et G des formules propositionnelles où n'apparaissent que les constantes propositionnelles $p_1, \ldots, p_n$. Soit $H_1, \ldots, H_n$ des formules propositionnelles sur un ensemble $\mathscr{P}$ de variables propositionnelles, alors*

— *si F est une tautologie, la formule*

$$F[H_1/p_1, \ldots, H_n/p_n] \text{ est une tautologie.}$$

3. Dans la suite, on utilisera souvent la structure d'algèbre de Boole de $(\mathbb{B}^n, \wedge, \vee, \mathbf{0}, \mathbf{1}, (.)')$ (isomorphe à l'algèbre de Boole des parties d'un ensemble à n éléments)

4. On insiste sur *simultanément* car rien n'empêche les formules H_i, $i \leqslant n$, de contenir certaines des variables $p_1, ..., p_n$ et on veut éviter les imbrications dans les substitutions

— *si* $F \equiv G$,

$$F[H_1/p_1,\ldots,H_n/p_n] \equiv G[H_1/p_1,\ldots,H_n/p_n]$$

Noter que les réciproques des résultats ci-dessus ne sont pas vraies. Si $H_1 = \top$, et $F = p_1$ alors $F[H_1/p_1] = H_1 = \top$ qui est une tautologie, alors que F n'en est pas une.

Démonstration. Il suffit de démontrer la première partie de l'énoncé puisque $F \equiv G$ ssi $F \leftrightarrow G$ est une tautologie. Pour cette première partie on utilise le lemme suivant.

Lemme 1.4.2 *Soit F une formule propositionnelle où n'apparait que les variables propositionnelles $p_1,\ldots,p_n$. Soit $H_1,...,H_n$ des formules propositionnelles sur un ensemble $\mathscr{P}$ de variables (qui peut contenir ou non $p_1,\ldots,p_n$). Soit v une valuation sur $\mathscr{P}$ telle que $v(H_i) = \delta_i$ ($\delta_i \in \{0,1\}$, $i \in \{1,\ldots,n\}$). Alors pour la valuation v' définie sur $\{p_1,\ldots,p_n\}$ par $v'(p_i) = \delta_i$, pour $i \in \{1,...,n\}$, vérifie*

$$v(F[H_1/p_1,\ldots,H_n/p_n]) = v'(F)\ .$$

Démonstration (lemme). Le lemme se montre par induction sur F. Par souci de lisibilité, on note $F[\bar{H}/\bar{p}]$ la formule $F[H_1/p_1,\ldots,H_n/p_n]$

— Si F est $\bot$, $v(\bot) = v'(\bot) = 0$.
— Si F est une variable propositionnelle p_i, $i \leqslant n$, on a : $F[\bar{H}/\bar{p}]$ est H_i, et $v'(p_i) = \delta_i = v(H_i)$.
— Si F est $\neg G$, avec par hypothèse d'induction $v(G[\bar{H}/\bar{p}]) = v'(G)$, alors $v(F[\bar{H}/\bar{p}]) = 1 - v(G[\bar{H}/\bar{p}]) = 1 - v'(G) = v'(F)$.
— Si F est $G_1 \wedge G_2$, avec par hypothèse d'induction $v(G_1[\bar{H}/\bar{p}]) = v'(G_1)$, $v(G_2[\bar{H}/\bar{p}]) = v'(G_2)$, alors $v(F[\bar{H}/\bar{p}]) = v(G_1[\bar{H}/\bar{p}]) \cdot v(G_2[\bar{H}/\bar{p}]) = v'(G_1) \cdot v'(G_2) = 1$. Le raisonnement est le même pour les autres connecteurs. ■

Fin de la démonstration (propriété de substitution). Supposons que F est une tautologie. Soit v une valuation quelconque sur $\mathscr{P}$. Par le lemme précédent, il existe v' valuation sur $\{p_1,\ldots,p_n\}$ telle que

$$v(F[H_1/p_1,\ldots,H_n/p_n]) = v'(F)\ .$$

Comme F est une tautologie, $v'(F) = 1$ et donc $v(F[H_1/p_1,\ldots,H_n/p_n]) = 1$. Toute valuation v satisfait bien $F[H_1/p_1,\ldots,H_n/p_n]$ qui est donc une tautologie. ■

Il s'agit d'une propriété finalement très intuitive et que l'on applique spontanément au cas par cas (voir les équivalences qui suivent). La démonstration du lemme permet de faire fonctionner une démonstration par induction sur la structure des formules. Il doit être bien clair qu'elle ne pose aucune difficulté autre que d'écriture. Dans la suite on ne rédige pas forcément ce genre de démonstration.

La propriété de substitution explique que l'on parle de variables propositionnelles, alors que, dès que l'on formalise il s'agit plutôt de constantes.

Dans la suite A, B et C désignent des formules quelconques. Voici quelques équivalences usuelles, qu'il suffit donc de vérifier pour des variables propositionnelles par substitution.

Négation des connecteurs usuels.

$$\neg\neg A \equiv A$$

Lois de de Morgan :

$$\neg(A \wedge B) \equiv (\neg A \vee \neg B)$$

$$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$$

$$\neg(A \rightarrow B) \equiv A \wedge \neg B$$

$$\neg(A \leftrightarrow B) \equiv (A \leftrightarrow \neg B)$$

$$\neg(A \leftrightarrow B) \equiv (\neg A \leftrightarrow B)$$

Expression des connecteurs avec $\neg$, $\wedge$ et $\vee$.

$$\bot \equiv (A \wedge \neg A)$$

$$\top \equiv (A \vee \neg A)$$

$$(A \rightarrow B) \equiv (\neg A \vee B)$$

$$(A \leftrightarrow B) \equiv ((A \rightarrow B) \wedge (B \rightarrow A)) \equiv ((\neg A \vee B) \wedge (\neg B \vee A))$$

$$(A \leftrightarrow B) \equiv ((A \wedge B) \vee (\neg A \wedge \neg B))$$

Propriétés de la disjonction et de la conjonction.

commutativité :

$$(A \wedge B) \equiv (B \wedge A)$$

$$(A \vee B) \equiv (B \vee A)$$

associativité :

$$(A \wedge (B \wedge C)) \equiv ((A \wedge B) \wedge C)$$

$$(A \vee (B \vee C)) \equiv ((A \vee B) \vee C)$$

distributivité :

$$(A \wedge (B \vee C)) \equiv ((A \wedge B) \vee (A \wedge C))$$

$$(A \vee (B \wedge C)) \equiv ((A \vee B) \wedge (A \vee C))$$

idempotence :

$$(A \wedge A) \equiv A$$

$$(A \vee A) \equiv A$$

absorption :

$$(A \wedge \bot) \equiv \bot$$

$$(A \wedge (A \vee B)) \equiv A$$

$$(A \vee \top) \equiv \top$$

$$(A \vee (A \wedge B)) \equiv A$$

neutre :

$$(A \wedge \top) \equiv A$$

$$(A \vee \bot) \equiv A$$

$$(\neg A \wedge (A \vee B)) \equiv (\neg A \wedge B)$$

$$(\neg A \vee (A \wedge B)) \equiv (\neg A \vee B)$$

Remarque : ces deux dernières équivalences ainsi que celles d'absorption et d'idempotence permettent de simplifier les formes normales.

Propriétés de l'implication et de l'équivalence.

$$((A \wedge B) \rightarrow C) \equiv (A \rightarrow (B \rightarrow C))$$

contraposée :

$$(A \rightarrow B) \equiv (\neg B \rightarrow \neg A)$$

$$(A \leftrightarrow B) \equiv (\neg B \leftrightarrow \neg A)$$

Distributivité à droite :

$$(A \rightarrow (B \wedge C)) \equiv ((A \rightarrow B) \wedge (A \rightarrow C))$$

$$(A \rightarrow (B \vee C)) \equiv ((A \rightarrow B) \vee (A \rightarrow C))$$

Pseudo-distributivité à gauche :

$$((A \wedge B) \rightarrow C) \equiv ((A \rightarrow C) \vee (B \rightarrow C))$$

$$((A \vee B) \rightarrow C) \equiv ((A \rightarrow C) \wedge (B \rightarrow C))$$

Remarque : il n'y a pas de propriété analogue à ces 4 dernières pour l'équivalence.

$\bot$ et $\top$:

$$(A \rightarrow \bot) \equiv \neg A \quad (\bot \rightarrow A) \equiv \top \quad (A \rightarrow \top) \equiv \top \quad (\top \rightarrow A) \equiv A$$

$$(A \leftrightarrow \bot) \equiv \neg A \quad (A \leftrightarrow \top) \equiv A$$

Encore quelques équivalences.

$$(\neg A \rightarrow A) \equiv A \quad ((A \rightarrow B) \rightarrow A) \equiv A \quad ((A \rightarrow B) \rightarrow B) \equiv (A \vee B)$$

1.5 Formes normales.

1.5.1 Définitions.

Soit $\mathscr{P} = \{x_1, ..., x_n\}$ un ensemble de variable propositionnelle. On appelle *littéral* une constante propositionnelle ou une négation de constante propositionnelle. On parle de littéral négatif pour un littéral de la forme $\neg x_i$, $x_i \in \mathscr{P}$.

Une *clause* C est une formule propositionnelle de la forme :

$$C = \bigvee_{i \in A} x_i \vee \bigvee_{i \in B} \neg x_i$$

où $A, B \subseteq \{1, ..., n\}$. La *longueur* d'une clause est son nombre de littéraux (ici $|A| + |B|$). On le note $|C|$. Lorsque $|C| = 1$, on parle de clause *unitaire*.

De même, une *conjonction élémentaire* C est une formule propositionnelle de la forme [5] :

$$C = \bigwedge_{i \in A} x_i \wedge \bigwedge_{i \in B} \neg x_i$$

où $A, B \subseteq \{1, ..., n\}$. A nouveau, la *longueur* d'une conjonction élémentaire est son nombre de littéraux (ici $|A| + |B|$). On le note aussi $|C|$.

Une formule F est sous *forme normale disjonctive*, notée FND, (on dit parfois forme normale disjonctive-conjonctive) si elle s'écrit comme une disjonction de conjonctions de littéraux i.e. comme une disjonction de conjonctions élémentaires. Dans ce cas, F est sous FND si elle est de la forme :

$$F = \bigvee_{i=1}^{m} \left(\bigwedge_{i \in A_k} x_i \wedge \bigwedge_{i \in B_k} \neg x_i \right)$$

Si la longueur de chaque conjonction élémentaire est bornée par l, on dira que la formule est en l-FND.

5. Quand il n'y a pas ambiguité, on omet le parenthésage

Une formule est sous *forme normale conjonctive*, notée FNC, (on dit parfois forme normale conjonctive-disjonctive) si elle s'écrit comme une conjonction de disjonctions de littéraux i.e. comme une conjonction de clauses. Elle est de la forme :

$$F = \bigwedge_{i=1}^{m} (\bigvee_{i \in A_k} x_i \vee \bigvee_{i \in B_k} \neg x_i)$$

Si la longueur de chaque clause est bornée par l, on dira que la formule est en l-FNC.

La longueur d'une formule F en FNC (resp. FND) est égale à la somme de la longueur de ses clauses (resp. de ses conjonctions élémentaires). Ainsi, pour $F = \bigwedge_{i=1}^{m} C_i$, on a :

$$|F| = \sum_{i=1}^{m} |C_i|.$$

Enfin, on notera $\mathsf{cl}(F)$ l'ensemble $\{C_1, ..., C_m\}$ des clauses de F et $\mathsf{var}(F)$ l'ensemble des variables de F (qui est un sous-ensemble de $\mathscr{P}$).

Exemple. Supposons que p, q, r soient des constantes propositionnelles, alors $p, q, r, \neg p, \neg q, \neg r$ sont des littéraux.

La formule $(p \wedge q \wedge r) \vee (\neg p \wedge r) \vee \neg r$ est sous forme normale disjonctive (la troisième conjonction est réduite à un élément).

La formule $(p \vee \neg q) \wedge \neg p \wedge (p \vee q \vee \neg r)$ est sous forme normale conjonctive (la deuxième disjonction est réduite à un élément).

La formule $p \vee \neg q$ est à la fois sous forme normale disjonctive (deux conjonctions réduites à un élément) et disjonctive (une conjonction réduite à une disjonction de deux littéraux), de même la formule $p \wedge q \wedge \neg r$.

1.5.2 Table de vérité et formes normales.

Voyons tout d'abord l'idée de la démonstration sur un exemple. Il s'agit tout simplement de décrire la table de vérité ligne par ligne, on obtient naturellement ainsi une forme normale disjonctive ou conjonctive suivant la façon dont on procède. Prenons la table de vérité de l'équivalence :

p	q	$p \leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

On peut la décrire de deux façons. En énumérant les valuations qui rendent la formule vraie :

$$\begin{aligned} v(p \leftrightarrow q) = 1 \quad & \text{ssi } [v(p) = 0 \text{ et } v(q) = 0] \text{ ou } [v(p) = 1 \text{ et } v(q) = 1] \\ & \text{ssi } [v(\neg p) = 1 \text{ et } v(\neg q) = 1] \text{ ou } [v(p) = 1 \text{ et } v(q) = 1] \\ & \text{ssi } v(\neg p \wedge \neg q) = 1 \text{ ou } v(p \wedge q) = 1 \\ & \text{ssi } v((\neg p \wedge \neg q) \vee (p \wedge q)) = 1 \end{aligned}$$

on obtient une forme normale disjonctive de $p \leftrightarrow q$ qui est $(\neg p \wedge \neg q) \vee (p \wedge q)$.

En énumérant les valuations qui rendent la formule fausse :

$$\begin{aligned} v(p \leftrightarrow q) = 0 \quad & \text{ssi } [v(p) = 0 \text{ et } v(q) = 1] \text{ ou } [v(p) = 1 \text{ et } v(q) = 0] \\ & \text{ssi } [v(p) = 0 \text{ et } v(\neg q) = 0] \text{ ou } [v(\neg p) = 0 \text{ et } v(q) = 0] \\ & \text{ssi } v(p \vee \neg q) = 0 \text{ ou } v(\neg p \vee q) = 0 \\ & \text{ssi } v((p \vee \neg q) \wedge (\neg p \vee q)) = 0 \end{aligned}$$

on obtient une forme normale conjonctive de $p \leftrightarrow q$ qui est $(p \vee \neg q) \wedge (\neg p \vee q)$.

On montre maintenant que ce procédé fonctionne pour n'importe quelle table de vérité.

Les deux propriétés suivantes sont des conséquences immédiates de la définition de l'interprétation des conjonctions et disjonctions :

Lemme 1.5.1 *Soient $F_1, \ldots, F_n$ des formules propositionnelles. On a :*

$$\begin{aligned} v(F_1 \wedge \cdots \wedge F_n) = 1 \ \ \textit{ssi} \ \ v(F_1) = 1 \text{ et } \ldots \text{ et } v(F_n) = 1 \\ v(F_1 \vee \cdots \vee F_n) = 0 \ \ \textit{ssi} \ \ v(F_1) = 0 \text{ et } \ldots \text{ et } v(F_n) = 0 \end{aligned}$$

On utilise ces propriétés pour le lemme suivant :

Lemme 1.5.2 *Soit a une valuation, et n constantes propositionnelles $p_1,\dots,p_n$. Il existe des formules F^1 et F^0 telles que pour toute valuation v :*

$$F^1 \text{ est une conjonction de littéraux et } v(F^1)=1 \text{ ssi pour tout } i\in\{1,\dots,n\}\ v(p_i)=a(p_i)$$
$$F^0 \text{ est une disjonction de littéraux et } v(F^0)=0 \text{ ssi pour tout } i\in\{1,\dots,n\}\ v(p_i)=a(p_i)$$

Démonstration. On pose $p_i^1=p_i$ si $a(p_i)=1$, $p_i^1=\neg p_i$ si $a(p_i)=0$. On pose $F^1=p_1^1\wedge\cdots\wedge p_n^1$. On a bien que $a(p_i^1)=1$ pour tout i, donc $a(F^1)=1$.

On a que $v(F^1)=1$ ssi $v(p_i^1)=1$ pour chaque $i\in\{1,\dots,n\}$, d'où le résultat.

De même en posant $p_i^0=p_i$ si $a(p_i)=0$, $p_i^0=\neg p_i$ si $a(p_i)=1$, et$F^0=p_1^0\vee\cdots\vee p_n^0$, on a bien, $a(F^0)=0$ et $v(F^0)=0$ ssi $v(p_i^0)=0$ pour chaque $i\in\{1,\dots,n\}$, d'où le résultat. ■

Pour en déduire le théorème de mise sous forme normale, on utilise maintenant les propriétés duales du lemme 1.5.2 :

Lemme 1.5.3 *Soient $F_1,\dots,F_n$ des formules propositionnelles. On a :*

$$v(F_1\vee\cdots\vee F_n)=1 \ \textit{ssi}\ \ v(F_1)=1 \text{ ou } \dots \text{ ou } v(F_n)=1$$
$$v(F_1\wedge\cdots\wedge F_n)=0 \ \textit{ssi}\ \ v(F_1)=0 \text{ ou } \dots \text{ ou } v(F_n)=0$$

Théorème 1.5.4 *Toute table de vérité, est la table de vérité d'une formule. Qui plus est on peut choisir cette formule sous forme normale conjonctive, ou sous forme normale disjonctive.*

Plus précisément, pour tout ensemble de n constantes propositionnelles $\{p_1,\dots,p_n\}$, toute fonction de $\{0,1\}^{\{p_1,\dots,p_n\}}$ dans $\{0,1\}$, il existe une formule sous forme normale conjonctive et une formule sous forme normale disjonctive dont c'est la table de vérité.

Démonstration. Soit t une table de vérité sur $p_1,\dots,p_n$. Soit $v_0,\dots,v_q$ toutes les valuations dont l'image est 1 par t. A chacune de ces valuations v_i on associe la formule F_i^1 donnée par le lemme 1.5.2. La formule $F_0^1\vee\cdots\vee F_q^1$ est sous forme normale disjonctive et répond à la question. En effet $v(F_0^1\vee\cdots\vee F_q^1)=1$ ssi $v(F_0^1)=1$ ou …ou $v(F_q^1)=1$, et donc ssi v est l'une des valuation $v_0,\dots,v_q$ d'après le lemme 1.5.2.

De façon analogue si $u_0,\dots,u_r$ sont les valuations dont l'image est 0 par t, on associe à chacune d'entre elles la formule F_i^0 donnée au lemme 1.5.2. La formule $F_0^0\wedge\cdots\wedge F_q^0$ est sous forme normale conjonctive et répond à la question, car $v(F_0^0\wedge\cdots\wedge F_q^0)=0$ ssi $v(F_0^0)=0$ ou …ou $v(F_0^0)=0$. ■

Là encore on s'aperçoit que l'on a essentiellement utilisé les propriétés de la conjonction et de la disjonction dans le meta-langage. C'est à dire que l'on peut décrire une table de vérité à l'aide uniquement de conjonctions de disjonctions et de négations en décrivant toutes les valuations qui prennent la valeur "vrai" (forme normale disjonctive) ou la valeur "faux" (forme normale conjonctive).

La forme normale que l'on trouve à la lecture de la table de vérité, en suivant la méthode indiquée par la preuve du théorème précédent, n'est en général pas la plus courte! Ainsi $p\wedge q$ est sous forme normale conjonctive et disjonctive. La lecture de la table de vérité donnera bien $p\wedge q$ comme forme normale disjonctive, mais $(p\vee q)\wedge(p\vee\neg q)\wedge(\neg p\vee q)$ comme forme normale conjonctive.

1.5.3 Systèmes complets de connecteurs.

On dit qu'un système de connecteurs est *fonctionnellement complet*, ou parfois plus simplement *complet* si toute table de vérité est représentée par une formule utilisant ces seuls connecteurs. Un corollaire immédiat du théorème de mise sous-forme normale est que :

Corollaire 1.5.5 *Le système de connecteurs $\{\neg,\wedge,\vee\}$ est fonctionnellement complet.*

Des diverses équivalences parmi celles du paragraphe 1.4 on déduit que :

Corollaire 1.5.6 *Les systèmes de connecteurs $\{\neg,\wedge\}$, $\{\neg,\vee\}$, $\{\neg,\to\}$, $\{\bot,\to\}$ sont fonctionnellement complets.*

Démonstration. Pour chaque système S il suffit de montrer que chacun des connecteurs d'un système complet connu, pour le moment le seul connu est $\{\neg, \wedge, \vee\}$, s'exprime avec les connecteurs du système S.

Le système $\{\neg, \wedge\}$ est complet car $A \vee B \equiv \neg(\neg A \wedge \neg B)$ (loi de de Morgan).
Le système $\{\neg, \vee\}$ est complet car $A \wedge B \equiv \neg(\neg A \vee \neg B)$ (loi de de Morgan).
Le système $\{\neg, \rightarrow\}$ est complet car $A \vee B \equiv \neg A \rightarrow B$ et $\{\neg, \vee\}$ est complet.
Le système $\{\bot, \rightarrow\}$ est complet car $\neg A \equiv A \rightarrow \bot$ et $\{\neg, \rightarrow\}$ est complet. ■

1.5.4 Mise sous forme normale.

Étant donnée une formule, la mettre sous forme normale conjonctive, resp. disjonctive, consiste à trouver une formule équivalente sous forme normale conjonctive, resp. disjonctive.

Une première méthode est de chercher la table de vérité puis d'en déduire la forme normale cherchée comme dans la preuve du théorème 1.5.4.

Une seconde méthode un peu plus directe est d'utiliser convenablement les équivalences logiques des paragraphes 1.4, 1.4 et 1.4.

Ces méthodes ne sont pas très efficaces algorithmiquement. La méthode des tables de vérité nécessite d'inspecter l'ensemble des valuations et, utilisée telle quelle, son coût est prohibitif.

Pratiquement on a souvent besoin seulement d'une forme normale non pas équivalente mais *equisatisfaisable* à une formule donnée (l'une est satisfaite ssi l'autre l'est). Là il existe des algorithmes simples et efficaces que l'on décrira par la suite.

1.6 Formes normales complètes et bornes inférieures de FND

1.6.1 Impliquants, absorption et impliquants premiers

Soient $F \in \mathfrak{F}$ et C une conjonction élémentaire non vide de littéraux, sur $\mathscr{P}$. On dit que C est un *impliquant* de F si $(C \rightarrow F)$ est valide i.e. si pour toute valuation $\overline{v} \in \{0,1\}^{\mathscr{P}}$, $\overline{v}(C) = 1$ implique $\overline{v}(F) = 1$. De façon équivalente, C est un impliquant de F si $C \vee F \equiv F$.

Soient C_1, C_2 des impliquants de F. On dit que C_1 *absorbe* C_2 si $C_2 \vee C_1 \equiv C_1$ autrement dit si C_2 est aussi un impliquant de C_1 (c'est une façon de dire que C_1 est *plus général* que C_2). On dit que C_1 est un *impliquant premier* de F si C_1 est un impliquant de F et n'est absorbé par aucun autre impliquant C_2 de F.

Pour résumer :

$$\begin{array}{rl} C_1 \text{ est un impliquant de } C_2 \text{ ssi} & (C_1 \rightarrow C_2) \text{ est valide} \\ \text{ssi} & (C_1 \vee C_2) = C_2 \\ \text{ssi} & C_2 \text{ absorbe } C_1. \end{array}$$

De même, C_1 est un impliquant premier de F ssi
— C_1 est un impliquant de F et
— pour toute conjonction élémentaire C_2 sur $\mathscr{P}$, $(C_1 \rightarrow C_2)$ n'est pas valide (i.e. $(C_1 \vee C_2) \not\equiv C_2$)

Par définition, tout impliquant est absorbé par un impliquant premier. En d'autres termes, les impliquants premiers sont en quelque sorte les impliquants les plus "généraux" d'une formule.

Exemple. $C_2 = (x_1 \wedge \neg x_2 \wedge x_3)$ est un impliquant de $C_1 = (x_1 \wedge \neg x_2)$ (C_1 absorbe C_2). Ils sont tous les deux impliquants de la formule $F = x_1 \vee (\neg x_2 \wedge \neg x_3)$. La clause $C = x_1$ est un impliquant premier de F.

Exemple. Soient $C_1 = x$, $C_2 = (x \wedge y)$, $C_3 = (\neg x \wedge y)$ et $F = C_1 \vee C_2 \vee C_3$. C_1, C_2 et C_3 sont des impliquants de F. C_1 est un impliquant premier de F mais C_2 n'est pas un impliquant premier (car il implique C_1). On a donc $F \equiv C_1 \vee C_3$. La conjonction C_3 n'est pas non plus un impliquant premier de F car :
— $C_4 = y$, est un impliquant de F (le vérifier)
— C_3 implique C_4
On a donc : $F \equiv x \vee y$.

On laisse l'exercice suivant au lecteur. Bien que facile à démontrer, le résultat sera utilisé de nombreuses fois dans la suite.

Exercice 1 Soit $C_1 = \bigwedge_{i \in A_1} x_i \wedge \bigwedge_{i \in B_1} \neg x_i$ et $C_2 = \bigwedge_{i \in A_2} x_i \wedge \bigwedge_{i \in B_2} \neg x_i$. Montrer que C_1 est un impliquant de C_2 (i.e. que C_2 absorbe C_1) si et seulement si $A_2 \subseteq A_1$ et $B_2 \subseteq B_1$.

Une conséquence immédiate de l'exercice 1 est que si C_1 est un impliquant de C_2 alors $|C_2| \leqslant |C_1|$.

Exercice 2 Soit $C = \bigwedge_{i\in I} l_i$ où chaque l_i est un littéral et F une formule propositionnelle. Montrer que C est un impliquant premier de F si et seulement si C est un impliquant de F et aucune des conjonctions $C_j = \bigwedge_{i\in I\setminus\{j\}} l_i$ n'est un impliquant de F.

La proposition suivante complète et raffine l'approche pour construire des FND par les tables de vérités. On remarque tout d'abord que pour toute FND

$$F = \bigvee_{i\in I} C_i,$$

C_i est un impliquant de F.

Proposition 1.6.1 *Toute formule peut être mise sous forme normale disjonctive dont les conjonctions élémentaires sont précisément ses impliquants premiers.*

Une telle forme FND est dite *complète.*

Démonstration. Soit F une formule propositionnelle sur un ensemble $\mathscr{P}$ fini. A équivalence logique près, le nombre de conjonctions élémentaires sur $\mathscr{P}$ est fini et donc le nombre d'impliquants premiers de F est fini. Soit $D_1, ..., D_m$ la liste des impliquants premiers de F. On sait que F admet au moins une FND. Posons $F \equiv \bigvee_{i\in I} C_i$. Comme F absorbe chacun de ses impliquants (donc les impliquants premiers) et par associativité de la disjonction, on a :

$$F \equiv \bigvee_{i\in I} C_i \vee \bigvee_{j=1}^{m} D_j.$$

Chaque C_i, $i \in I$, est un impliquant de F, il est donc absorbé par au moins un impliquant premier D_j, pour $j \in \{1, ..., m\}$. On a donc $C_i \vee D_j \equiv D_j$. Par absorption successive de tous les C_i on obtient que $F \equiv \bigvee_{j=1}^{m} D_j$. ■

Soit $F \equiv \bigvee_{i\in I} C_i$ une formule en FND. On dit que cette FND est *première* si chaque C_i est un impliquant premier de F et qu'elle est *non redondante* si, quelque soit $j \in I$:

$$F \not\equiv \bigvee_{i\in I\setminus\{j\}} C_i.$$

Une FND F est donc non redondante si, pour tout $j \in I$, il existe une valuation qui satisfait C_j mais pas $\bigvee_{i\in I\setminus\{j\}} C_i$. A contrario, F est *redondante* s'il existe $j \in I$ tel que toute valuation satisfaisant C_j satisfait aussi $\bigvee_{i\in I\setminus\{j\}} C_i$ i.e. si C_j est absorbée par $\bigvee_{i\in I\setminus\{j\}} C_i$. La notion de redondance généralise donc celle d'absorption.

Si une FND est complète alors elle est première. Par contre, elle peut être première sans être complète. En effet, une FND complète (ou simplement première) peut être redondante : certains de ses impliquants premiers peuvent être superflus. On peut alors éliminer certains impliquants premiers tout en conservant une formule première équivalente à la formule de départ.

Exemple. La FND,

$$F = (x \wedge y) \vee (y \wedge z) \vee (z \wedge \neg x)$$

est première (vérifiez que chacune de ses clauses sont (parmi) ses impliquants premiers) mais elle est redondante. En effet, la clause $(y \wedge z)$ est superflue : toutes les valuations qui satisfont $(y \wedge z)$ satisfont aussi soit $(x \wedge y)$, soit $(z \wedge \neg x)$. Il s'en suit que $F \equiv (x \wedge y) \vee (z \wedge \neg x)$.

Toute FND d'une formule peut-être transformée en une FND première en remplaçant successivement chaque conjonction C_i qui n'est pas première par un impliquant premier qui l'absorbe. De même, à partir d'une FND, on peut obtenir une FND équivalente et non redondante en éliminant les clauses redondantes. On se sert de ces notions pour caractériser les formes FND "minimales" de certaines formules (ou tables de vérités).

1.6.2 Formules propositionnelles positives

Soit $\mathscr{P}$ un ensemble de variables propositionnelle et v_1, v_2 deux valuations. On dit que $v_1 \leqslant v_2$ si, pour tout $x \in \mathscr{P}$, $\overline{v}_1(x) \leqslant \overline{v}_2(x)$.

Définition 1.6.1 *Une formule $F \in \mathfrak{F}$ est positive si, pour toutes valuations v_1, v_2 telles que $v_1 \leqslant v_2$, on a :*

$$\overline{v}_1(F) \leqslant \overline{v}_2(F)$$

Proposition 1.6.2 *Une formule propositionnelle F sur $\mathscr{P}$ est positive si et seulement si tous les impliquants premiers de F sont positifs i.e. aucun ne contient un littéral $\neg x$ avec $x \in \mathscr{P}$.*

Démonstration. Supposons que F est positive et qu'un de ses impliquants premier C contient des littéraux négatifs. C est de la forme $C = \bigwedge_{i\in A} x_i \wedge \bigwedge_{i\in B} \neg x_i$. On sait (cf exercice 2) que dans ce cas, $C' = \bigwedge_{i\in A} x_i$ n'est pas un impliquant de F i.e. qu'il existe une valuation v telle que $\overline{v}(C') = 1$ et $\overline{v}(F) = 0$. Si v est telle que $\overline{v}(C') = 1$ alors elle vérifie, $v(x_i) = 1$ pour $i \in A$ et les autres variables peuvent être assignées indifféremment. Or, on sait que C est un impliquant de F donc la valuation v_0 définie par $v_0(x_i) = 1$ pour $i \in A$ et $v_0(x_j) = 0$ pour $j \in B$ est telle que $\overline{v}_0(C) = \overline{v}_0(F) = 1$. Comme F est positive, toute assignation v, telle que $v_0 \leqslant v$ satisfait $\overline{v}(F) = 1$. En particulier, toutes les assignations v telles que $v(x_i) = 1$ pour $i \in A$ sont dans ce cas. Donc C' est un impliquant de F.

Pour l'autre direction, on sait que F est équivalente à $\bigvee_{j\in I} C_j$ où les C_j sont les impliquants premiers, tous positifs, de F. Soit v une valuation et posons $A_v = \{i \in \{1,...,n\} : v(x_i) = 0\}$. Par définition, $\overline{v}(C_j) = 0$ si C_j contient un littéral x_i avec $i \in A_v$ et $\overline{v}(C_j) = 1$ sinon. Soit maintenant $v' \neq v$ tel que $v \leqslant v'$. A nouveau, $\overline{v}'(C_j) = 0$ si C_j contient un littéral x_i avec $i \in A_{v'}$ et $\overline{v}'(C_j) = 1$ sinon. Mais, comme $v \leqslant v'$, on a $A_{v'} \subseteq A_v$ donc $\{C_j : \overline{v}(C_j) = 1\} \subseteq \{C_j : \overline{v}'(C_j) = 1\}$. Donc $\overline{v}(F) \leqslant \overline{v}'(F)$. ■

Par extension, on dira qu'une FND est positive si elle ne contient aucun littéral négatif. Le résultat précédent implique que pour construire la FND complète d'une formule positive on ne peut s'aider des négations de variables.

Proposition 1.6.3 *Soit F une formule propositionnelle positive. Alors la FND complète de F est positive et non redondante. Elle est donc l'unique FND première de F.*

Démonstration. Soit $F \in \mathfrak{F}$ et $\mathsf{var}(F) = \{x_1, ..., x_n\}$. Comme les impliquants premiers d'une formule positive sont positifs (proposition 1.6.2) et que toute formule a une FND dont les conjonctions élémentaires sont ses impliquants premiers (proposition 1.6.1), la FND complète de F est positive. Soit $F \equiv \bigvee_{i\in I} C_i$ cette FND et supposons qu'elle est redondante. Soit $j \in I$ tel que :

$$F \equiv \bigvee_{i\in I\setminus\{j\}} C_i.$$

Posons $C_j = \bigwedge_{a\in A} x_a$, $A \subseteq \{1,..,n\}$, et considérons la valuation v définie par : $v(x_a) = 1$ pour $a \in A$, et $v(x_a) = 0$ si $a \notin A$. La valuation v vérifie $\overline{v}(F) = 1$. Comme $F \equiv \bigvee_{i\in I\setminus\{j\}} C_i$, on a $\overline{v}(\bigvee_{i\in I\setminus\{j\}} C_i) = 1$. En particulier, il existe $j' \neq j$, telle que $\overline{v}(C_{j'}) = 1$. Or $C_{j'}$ est un impliquant premier de F et est donc positive i.e. ne contient pas de littéraux négatif. Posons $C_{j'} = \bigwedge_{b\in B} x_b$. On ne peut avoir $A \subseteq B$ ou $B \subseteq A$ car $C_j \neq C_{j'}$ et C_j et $C_{j'}$ sont des impliquants premiers (cf exercice 2). Donc $B \setminus A \neq \emptyset$ i.e. il existe $b \in B$ tel que x_b apparait dans $C_{j'}$ mais pas dans C_j. Par définition de v : $v(x_b) = 0$ et donc $\overline{v}(C_{j'}) = 0$. Contradition. ■

Noter que cette proposition n'est plus vraie si F n'est pas positive.

On peut se sevir de cette propriété pour montrer que certaines formules positive ont une FND dont le nombre minimal de conjonction élémentaire est exponentiel en le nombre de variables.

Soit $\mathscr{P} = \{x_0, ..., x_{2n-1}\}$ et $F = \bigwedge_{i=0}^{n-1}(x_{2i} \vee x_{2i+1})$. Une analyse de la formule (ou de sa table de vérité) donne la forme normale disjonctive complète G équivalente F suivante :

$$G = \bigvee_{\epsilon=(\epsilon_1,...,\epsilon_n)\in\{0,1\}^n} C_\epsilon = \bigvee_{\epsilon=(\epsilon_1,...,\epsilon_n)\in\{0,1\}^n} x_{\epsilon_1} \wedge x_{2+\epsilon_2} \wedge \ldots \wedge x_{2(n-1)+\epsilon_n}$$

La formule F est positive. Il n'y a pas d'autres impliquant premier que les C_ϵ. L'expression G ci-dessus est donc une FND complète de F. Par la proposition 1.6.3, l'expression donnée ci-dessus est donc non redondante et il s'agit de l'unique FND première de F. Toute autre FND G_0 équivalente à F est donc non première. Chaque clause conjonctive C de G_0 peut donc être remplacée par un impliquant premier C^* de F afin d'obtenir une nouvelle FND G^*, première, équivalente à F. Comme $|D^*| \leqslant |D|$, on a que $|G^*| \leqslant |G_0|$. Or, comme l'unique FND première de F est G, on a $G = G^*$. Toute FND G_0 de F est donc plus longue que G. et nécessite donc un nombre de conjonctions élémentaires supérieur ou égal à 2^n.

Exercice 3 Une FND

$$F = \bigvee_{i=1}^{m} \Big(\bigwedge_{i \in A_k} x_i \wedge \bigwedge_{i \in B_k} \neg x_i \Big)$$

est dite orthogonale si pour tout $k, l \in \{1, ..., m\}$ tels que $k \neq l$: $(A_k \cap B_l) \cup (A_l \cap B_k) \neq \emptyset$.

1. Donner un exemple de FND orthogonale avec au moins trois variables et trois conjonctions élémentaires.
2. Combien y-a-t-il de valuations satisfaisant F ?

1.7 Résolution

Dans ce chapitre, on s'intéresse à une méthode effective différente des tables de vérités, appelée *résolution*, pour décider de la satisfaisabilité d'une formule propositionnelle.

1.7.1 Introduction

Nous avons vu précédemment que le calcul d'une forme normale, FNC ou FND, équivalente pour une formule propositionnelle F peut être très couteux. Dans bien des cas, pourtant, ce n'est pas le problème de validité d'une formule qui nous intéresse mais celui de satisfaisabilité. Si on repense aux exemples que l'on a modélisé, le problème était souvent de montrer qu'un ensemble de contraintes (par exemple celle du Sudoku) sont réalisables (c'est-à-dire satisfisables). On n'a donc pas forcément besoin de construire une forme normale G *équivalente* à F mais simplement une forme normale qui est satisfaisable si F l'est et qui n'est pas satisfisable si F ne l'est pas. En d'autres terme, les formules F et G doivent etre ce que l'on appelle *équisatisfaisables*.

Dans le résultat ci-dessous, on montre que construire une forme normale conjonctive préservant la satisfaisabilité peut se faire très "rapidement".

Proposition 1.7.1 *Soit F une formule propositionnelle sur un ensemble de variables $\mathscr{P}_1$, alors il existe une formule propositionnelle en FNC, $G = \bigwedge_{i\in I} C_i$ sur un ensemble de variables $\mathscr{P}_2$ telle que F et G sont équisatisfaisables et telle que :*

— $|\mathscr{P}_2| = |\mathsf{sf}(F)|$
— $|I| \leqslant 3 \times |\mathsf{sf}(F)| + 1$
— *Chaque clause C_i, $i \in I$, est de longueur au plus* 3

Démonstration. Soit $F_1(=F), F_2, ..., F_m$ la liste des sous-formules de F. Notez que chaque variable propositionnelle apparait au plus une fois comme sous-formule. On introduit pour chaque F_i, $i \leqslant m$ une variable propositionnelle p_{F_i}. On pose alors : $\mathscr{P}_2 = \{p_{F_1}, ..., p_{F_m}\}$. Pour toute sous-formule F_i non réduite à une variable propositionnelle i.e. dans $\mathsf{sf}(F)\backslash\mathsf{var}(F)$, on définit une formule ϕ_i de la façon suivante :

— Si $F_i = \neg F_j$, on pose $\phi_i = (p_{F_i} \leftrightarrow \neg p_{F_j})$
— Si $F_i = F_j \vee F_k$, on pose $\phi_i = (p_{F_i} \leftrightarrow (p_{F_j} \vee p_{F_k}))$
— Si $F_i = F_j \wedge F_k$, on pose $\phi_i = (p_{F_i} \leftrightarrow (p_{F_j} \wedge p_{F_k}))$

Enfin, G est défini par :

$$G = p_{F_1} \wedge \bigwedge_{F_i \in \mathsf{sf}(F)} \phi_i$$

Soit $v_1 : \mathscr{P}_1 \to \{0,1\}$. On définit $v_2 : \mathscr{P}_2 \to \{0,1\}$ de la façon suivante :

$$v_2(p_{F_i}) = \bar{v}_1(F_i), \text{ pour } F_i \in \mathsf{sf}(F).$$

En particulier, si $F_i = x \in \mathscr{P}_1$, $v_1(p) = v_2(p_{F_i})$. On montre facilement, par induction sur la structure de F que si v_1 satisfait F alors v_2 satisfait G. Réciproquement, de toute valuation v_2 satisfaisant G, on peut extraire une valuation v_1 satisfaisant F par :

$$v_1(x) = v_2(p_{F_i}) \text{ pour } F_i = x \in \mathscr{P}_1$$

Il reste, pour terminer la preuve, à mettre chacun des ϕ_i en FNC grâce aux équivalences de la section 1.4. Par exemple, une équivalence du type $(p \leftrightarrow q \vee r)$ devient :

$$(p \leftrightarrow q \vee r) \equiv (p \to q \vee r) \wedge (q \vee r \to p) \equiv (\neg p \vee q \vee r) \wedge (\neg q \vee p) \wedge (\neg r \vee p)$$

Chaque équivalence est donc transformée en une conjonction d'au plus 3 clauses de longueur au plus 3. ■

Si on considère que la taille d'une formule F est le nombre de noeuds de son arbre de décomposition $\mathsf{arb}(F)$ alors la 3-FNC donnée par la proposition ci-dessus a un nombre de clauses linéaire en la taille de F. Cela vient du fait qu'il y a une injection [6] de l'ensemble des sous-formules de F vers l'ensemble des sous-arbres (donc de noeuds) de $\mathsf{arb}(F)$.

6. "injection" seulement (et pas forcément bijection) car une même sous-formule peut apparaitre plusieurs fois comme sous-arbre : c'est le cas par exemple des occurrences multiples d'une même variable

1.7.2 La méthode de résolution

La *résolution* est une méthode de preuve dédiée aux formules propositionnelles sous forme normale conjonctive construites sur un ensemble de variables propositionnelles $\mathscr{P}$. Dans la suite, pour alléger les notations, une clause $C = l_1 \vee \cdots \vee l_k$ où chaque l_i, $i \leqslant k$, est un littéral sur $\mathscr{P}$ sera parfois représentée sous forme ensembliste $C = \{l_1, ..., l_k\}$. De même toute formule $\varphi = \bigwedge_{i=1}^{n} C_i$ sera assimilée à son ensemble de clauses $\mathsf{cl}(\varphi) = \{C_1, ..., C_n\}$ (et on notera, par abus, $\varphi = \{C_1, ..., C_n\}$). Suivant la nature des objets considérés (clause ou formule) l'interprétation donnée à la notation ensembliste sera donc différente. Suivant le contexte, on alternera entre la présentation classique et la présentation ensembliste des formules en FNC.

La méthode de résolution a une seule règle de déduction qui est la suivante. Soient $p \in \mathscr{P}$ et $D_1 = C_1 \cup \{p\}$ et $D_2 = C_2 \cup \{p\}$ deux clauses, telles que $p, \neg p \notin C_1 \cup C_2$. On écrit :

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\neg p\}}{C_1 \cup C_2}$$

pour dire qu'on obtient la clause $C_1 \cup C_2$ par résolution à partir des clauses D_1 et D_2. La nouvelle clause $C_1 \cup C_2$ produite par la règle s'appelle le *résolvant* sur la variable p de $C_1 \cup \{p\}$ et $C_2 \cup \{\neg p\}$. Elle est une conséquence logique de ces deux dernières clauses. En effet, supposons qu'une valuation v satisfait $C_1 \vee p$ et $C_2 \vee \neg p$. Alors, soit $v(p) = 1$ et dans ce cas $v(C_2) = 1$, soit $v(p) = 0$ et dans ce cas $v(C_1) = 1$. Dans tous les cas $v(C_1 \vee C_2) = 1$. On peut donc ajouter $C_1 \cup C_2$ sans changer la satisfaisabilité de $\{C_1 \cup \{p\}, C_2 \cup \{\neg p\}\}$ (mais par contre pas remplacer ces deux clauses par leur résolvant). En d'autre termes, pour toute valuation v,

$$v \text{ satisfait } \{D_1, D_2\} \text{ si et seulement si } v \text{ satisfait } \{D_1, D_2, C_1 \cup C_2\}.$$

Définition 1.7.1 *Une preuve de réfutation par résolution d'une formule en FNC, $\varphi = \{C_1, ..., C_n\}$ est une suite $r_1, r_2, ..., r_m$ telle que, chaque r_i, $i \leqslant m$ est :*

— *soit une clause C_j, $j \leqslant n$*

— *soit une clause obtenue par résolution à partir de deux clauses r_{i_0}, r_{i_1} avec $i_0 < i$ et $i_1 < i$.*

et telle que r_m est la clause vide, que l'on note $\square$. On dira que φ est réfutable par résolution dans ce cas.

De façon alternative on dira que la clause vide peut être dérivée par résolution à partir de $C_1, ..., C_n$. Si on obtient la clause vide $\square$ après résolution de deux clauses C_1, C_2 cela signifie qu'il existe $p \in \mathscr{P}$ tels que $C_1 = p$ et $C_2 = \neg p$. Comme un ensemble de clause contenant à la fois la clause p et la clause $\neg p$ ne peut être satisfaisable, et qu'ajouter une clause obtenue par résolution préserve la satisfaisabilité, il est donc cohérent de poser que $v(\square) = 0$, pour toute valuation v..

Soit $\varphi = \{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}, \{\neg p, \neg q\}\}$ (vu sous forme ensembliste). φ n'est clairement pas satisfaisable. On va en donner une preuve par résolution ci-dessous. Une façon assez claire de présenter les preuves de ce type est de les disposer en colonne avec à droite, pour rappel, les nunéros des clauses utilisées pour la résolution.

1.	p, q	
2.	$p, \neg q$	
3.	$\neg p, q$	
4.	$\neg p, \neg q$	
5.	p	résol. 1 et 2
6.	q	résol 1 et 3
7.	$\neg p$	résol 4 et 6
8.	$\square$	résol 5 et 7

Définition 1.7.2 *Un système de réfutation est cohérent (ou consistent) si toute formule qui peut-être réfutée n'est pas satisfaisable. Il est complet si toute formule non satisfaisable peut être réfutée.*

Théorème 1.7.3 *La méthode de résolution est cohérente et complète pour les formules propositionnelles en FNC. Plus précisément, soit φ une formule propositionnelle en FNC. Les deux assertions suivantes sont équivalentes :*

1. *φ est réfutable par résolution*
2. *φ n'est pas satisfaisable*

Démonstration. Supposons que φ est réfutable par résolution. Soit $r_1, r_2, ..., r_m$ une réfutation de φ ($r_m = \square$). On sait que le résolvant de deux clauses est une conséquence logique de celles-ci. Donc, comme tout r_i, $i \leqslant m$, est soit une clause de φ soit une clause obtenue par une chaine de résolutions à partir de clause de φ, il vient que pour toute valuation ν, si $\nu(\varphi) = 1$ alors $\nu(r_i) = 1$, pour tout $i \leqslant m$. Or r_m est la clause vide, donc $\nu(r_m) = 0$. Il vient que φ n'est pas satisfaisable.

Pour la réciproque, supposons que φ n'est pas satisfaisable. On montre en raisonnant par induction sur le nombre k de variables, que la clause $\square$ peut-être dérivée par résolution. On assimile φ a son ensemble de clauses $\{C_1, ..., C_n\}$. Si $k = 0$, alors soit $\varphi = \emptyset$, soit $\varphi = \{\square\}$. Le premier cas contredit l'hypothèse que φ n'est pas satisfaisable. Le deuxième amène la conclusion souhaitée.

Supposons maintenant l'équivalence vraie jusqu'à un certaine valeur $k \geqslant 0$ de $|\mathsf{var}(\varphi)|$ et montrons là pour $k+1$. Soit $p \in \mathsf{var}(\varphi)$. On isole les clauses obtenues par résolution sur la variable p ainsi que celles de φ ne contenant ni p ni $\neg p$. Au préalable on élimine aussi les clauses trivialement satisfaites i.e. celles qui contiennent à la fois p et $\neg p$. On pose ainsi $S = \varphi \backslash \{C_i : p \in C_i \text{ et } \neg p \in C_i\}$ puis :

$$\mathsf{Res}_p = \{C \cup D : C \cup \{p\} \in S \text{ et } D \cup \{\neg p\} \in S\}$$

$$\varphi_p = \mathsf{Res}_p \cup \{C \in S : p \notin C \text{ et } \neg p \notin C\}$$

Noter que $p \notin \mathsf{var}(\varphi_p)$. On montre :

Fait 1.7.4 *Si φ n'est pas satisfaisable alors φ_p n'est pas satisfaisable*

Démonstration (Fait 1.7.4). Supposons que φ n'est pas satisfaisable mais que φ_p l'est. Considérons $\nu : \mathsf{var}(\varphi_p) \longrightarrow \{0,1\}$ une valuation telle que $\bar{\nu}(\varphi_p) = 1$. Comme $p \notin \mathsf{var}(\varphi_p)$, on peut étendre ν en deux assignations ν_0, ν_1 telles que $\nu_0(q) = \nu(q)$ si $q \neq p$ et $\nu_0(p) = 0$, $\nu_1(q) = \nu(q)$ si $q \neq p$ et $\nu_1(p) = 1$. De façon évidente, toujours parce que $p \notin \mathsf{var}(\varphi_p)$: $\bar{\nu}_0(\varphi_p) = \bar{\nu}_1(\varphi_p) = \bar{\nu}(\varphi_p) = 1$. Comme, par hypothèse, φ n'est pas satisfaisable : $\bar{\nu}_0(\varphi) = 0$ et $\bar{\nu}_1(\varphi) = 0$. Donc il existe $C \in S$, $D \in S$, telles que : $\nu_0(C) = 0$ et $\nu_1(D) = 0$. On a aussi $C, D \notin \varphi_p$ car $\bar{\nu}_0(\varphi_p) = \bar{\nu}_1(\varphi_p) = \bar{\nu}(\varphi_p) = 1$. En particulier, puisque $C, D \in \varphi$, soit p, soit $\neg p$ apparaissent dans C et D. Comme $\nu_0(p) = 0$, il vient que $p \in C$ (si $\neg p \in C$, alors on aurait $\nu_0(C) = 1$). Donc C est de la forme $C' \cup \{p\}$ avec $\nu_0(C') = \nu(C') = 0$. De même, de $\nu_1(p) = 1$ il vient que $\neg p \in D$ et D est de la forme $D' \cup \{\neg p\}$ avec $\nu_1(D') = \nu(D') = 0$. Donc $\nu(C' \cup D') = 0$, mais ceci contredit le choix de ν qui devrait satisfaire $C' \cup D'$ puisque $C' \cup D' \in \mathsf{Res}_p$. Contradiction. ■

Comme $|\mathsf{var}(\varphi_p)| = |\mathsf{var}(\varphi)| - 1$, par hypothèse d'induction, il vient que φ_p est réfutable. Mais les clauses de φ_p sont, soit des clauses de φ, soit des clauses obtenues de celles de φ par résolution. Donc, la preuve de réfutation de φ_p peut être étendue en une preuve de réfutation de φ. ■

La preuve de complétude ci-dessus donne un algorithme (cf algorithme 1.1) pour tester la satisfaisabilité d'un ensemble de clauses : on élimine petit à petit les variables mais au prix de la génération par résolution d'un nombre de clauses qui peut croitre de façon quadratique à chaque étape. Au final, on peut être amené à produire un nombre exponentiel (en la longueur de la formule) de résolvantes. On remarque au passage que le processus est donc bien fini même dans le cas où la formule n'est pas réfutable : au delà d'un certain nombre d'étapes, on ne peut plus créer de nouveau résolvant donc en particulier la clause $\square$ si cela n'a pas déjà été fait. On peut alors arrêter le processus et conclure à la satisfaisabilité.

```
1: function SAT-RESOLUTION(φ = {C_1, ..., C_n})
2:     S ← φ
3:     while var(S) ≠ ∅ et □ ∉ S do
4:         choose p ∈ var(S)
5:         S = S\{C ∈ S : p ∈ C et ¬p ∈ C}
6:         Res ← {C ∪ D : C ∪ {p} ∈ S et D ∪ {¬p} ∈ S}
7:         S ← Res ∪ {C ∈ S : p ∉ C et ¬p ∉ C}
8:     if □ ∈ S then return 0    # φ n'est pas satisfaisable
9:     else return 1    # φ est satisfaisable
```

FIGURE 1.1 – Algorithme de satisfaisabilité par résolution

Si la preuve peut être tournée en un algorithme de test de satisfaisabilité elle n'est néanmoins pas très constructive :

— Dans le cas où φ est satisfaisable, elle ne permet pas de trouver immédiatement une valuation satisfaisante
— Dans le cas où φ n'est pas satisfaisable, elle ne donne pa vraiment de preuve par réfutation.

Des preuves plus constructives existent, que l'on verra en exercice.

Test de validité d'une formule par résolution Pour décider de la validité d'une formule φ, on peut donc appliquer l'algorithme suivant :

1. Transformer φ en $(\neg\varphi)$
2. Mettre $(\neg\varphi)$ sous forme FNC (par une méthode, peu coûteuse, préservant seulement l'équisatisfaisabilité)
3. Montrer que $(\neg\varphi)$ est non satisfaisable en produisant une preuve de réfutation par la méthode de résolution.

Sur l'efficacité de la résolution

Soit $r_1, r_2, ..., r_m$ une suite témoignant de la réfutation par résolution d'une formule $\varphi = \bigwedge_{i=1}^{n} C_i$. On appelle *longueur* d'une telle preuve, le nombre de r_j qui sont des clauses obtenues par résolution i.e. le nombre de r_j tels que $r_j \notin \{C_1, ..., C_n\}$.

Comme on l'a vu, la résolution est complète pour le calcul propositionnel mais on ne sait rien sur la longueur des preuves c'est-à-dire sur l'efficacité de cette méthode de preuve qu'elle soit appliquée avec ou sans stratégie particulière. On peut montrer que la résolution n'est pas une méthode très efficace et ce même pour des énoncés simples dont la validité est évidente d'un point de vue mathématique.

Le principe des tiroirs énonce que, pour tout entier positifs n, si on cherche à ranger $n+1$ objets dans n boites, deux objets partageront la même boite. Autrement dit, qu'il n'existe pas d'injection de $\{1, ..., n+1\}$ dans $\{1, ..., n\}$. Pour tout $n \in \mathbb{N}^*$, on appelle $\neg\mathbf{PHP}_n$ (PHP pour *Pigeon Hole Principle*), la formule obtenue par conjonction des clauses suivantes :

— $\varphi = \bigwedge_{i=1}^{n+1} p_{i,1} \vee \ldots \vee p_{i,n}$
— $\psi = \bigwedge_{j=1}^{n} \bigwedge_{1 \leqslant i < i' \leqslant n+1} (\neg p_{i,j} \vee \neg p_{i',j})$

En interprétant chaque $p_{i,j}$, pour $1 < j < n$ et $1 < i < n+1$, par *"l'objet i est dans la boite j"*, la formule $\neg\mathbf{PHP}_n$ affirme la négation du principe des tiroirs à savoir : que chaque objet $i \leqslant n+1$ est dans une boite j, $j \leqslant n$ (formule φ), et qu'il est possible que toute boite ne contienne pas deux objets (formule ψ). Bien que le principe des tiroirs soit élémentaire, le résultat remarquable suivant montre que toute réfutation de $\neg\mathbf{PHP}_n$ est forcément longue.

Théorème 1.7.5 (Haken) *Pour tout entier n suffisamment grand, toute réfutation de $\neg\mathbf{PHP}_n$ par résolution est de longueur au moins $2^{n/32}$.*

Exercice 4 Donner une réfutation de $\neg\mathbf{PHP}_3$ par résolution.

Applications à certains types de formules

Si pour certaines formules la méthode de résolution produit de façon intrinsèque des preuves de taille exponentielle (en la taille de la formule), il existe un certain nombre de cas particuliers pour lesquels elle s'avère très efficace.

Soit $\mathscr{P}$ un ensemble de variables propositionnelles. Une clause C avec variables dans $\mathscr{P}$ est appelée clause de *Horn* si elle contient au plus un littéral négatif, en d'autres termes, si elle est de la forme :

$$\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_k \vee q$$

où $p_1, ..., p_n, q \in \mathscr{P}$. En utilisant le connecteur "$\rightarrow$", on peut reécrire une telle clause sous la forme :

$$p_1 \wedge p_2 \wedge \cdots \wedge p_k \rightarrow q.$$

Noter que toute clause de longueur un (on appellera *unitaire* ce genre de clause) sont des clauses de Horn. Une formule propositionnelle φ en FNC est appelée formule propositionnelle de Horn si toutes ses clauses sont des clauses de Horn.

On peut exploiter la structure particulière des clauses pour décider rapidement (i.e. par des preuves courtes) de la satisfaisabilité d'une formule de Horn par résolution. Pour cela, on introduit une variante de la résolution appelée *résolution unitaire* (parfois aussi *propagation unitaire*), dont les règles sont les suivantes.

$$\frac{\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_k \vee q \qquad \neg q}{\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_k}$$

$$\frac{\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_k \vee q \qquad p_1}{\neg p_2 \vee \cdots \vee \neg p_k \vee q}$$

ou, en notation ensembliste, en prenant $\{q\} \in \mathsf{unit}(\varphi)$, où $\mathsf{unit}(\varphi)$ est l'ensemble des clauses unitaires de φ.

$$\frac{C_1 \cup \{\neg q\} \qquad \{q\}}{C_1}$$

On remarquera que la clause produite par résolution unitaire est une sous-clause de la clause non unitaire. La résolution unitaire n'est pas une procédure complète en général mais elle l'est pour les formules de Horn en entrée.

Théorème 1.7.6 *La résolution unitaire est cohérente et complète comme méthode de réfutation pour les formules de Horn. On peut décider de la satisfaisabilité d'une formule de Horn φ par la méthode de résolution unitaire en un nombre d'étapes linéaire en la longueur de φ.*

Démonstration. On va raisonner directement sur l'algorithme de satisfaisabilité. On obtient le résultat par modification de l'algorithme de test de satisfisabilité par résolution et par une analyse tenant compte de la structure des clauses.

```
1:  function HORN-SAT-RESOLUTION(φ)
2:      S ← φ
3:      ν(q) ← 0, ∀q ∈ var(S)
4:      while unit(S) ≠ ∅ et □ ∉ S do
5:          choose {p} ∈ unit(S)
6:          ν(p) ← 1
7:          S = S\{C ∈ S : p ∈ C et ¬p ∈ C}
8:          Res ← {C : C ∪ {¬p} ∈ S}
9:          S ← Res ∪ {C ∈ S : p ∉ C et ¬p ∉ C}
10:     if □ ∈ S then return "φ n'est pas satisfaisable"
11:     else return ν    # φ est satisfaisable
```

FIGURE 1.2 – Résolution pour la satisfaisabilité des formules de Horn

Si la clause vide $\square$ peut être dérivée, alors la formule φ n'est pas satisfaisable, par définition de la résolution.

Réciproquement si l'algorithme s'arrête sans générer la clause vide et donc en renvoyant une valuation ν, on va montrer par que φ est satisfaite par ν. On remarque tout d'abord que si un ensemble de clauses de Horn S ne contient pas de clauses unitaires alors il est satisfaisable par la valuation ν telle que $\nu(p) = 0$ pour tout $p \in \mathsf{var}(S)$: en effet, toutes les clauses sont alors de la forme $\neg p_1 \vee \ldots \neg p_k \vee q$ ou $\neg p_1 \vee \ldots \neg p_k$ avec $k \geqslant 1$, chaque clause contenant au moins un littéral négatif, il est satisfait par ν. L'algorithme est initié avec la valuation $\nu(p) = 0$ pour tout $p \in \mathsf{var}(S)$. Comme chaque itération n'affecte qu'une variable présente dans une clause unitaire, à l'issue de cette itération ν satisfait les clauses non unitaires de S et satisfait donc S si celui-ci ne contient plus de clauses unitaires.

Supposons qu'au début d'une itération dans la boucle, S contient une clause unitaire $\{p\}$ (p est un littéral positif ou négatif). Notons ν_1 la valuation ν obtenue après mise à joru de ν : $\nu_1(q) = \nu(q)$ pour $q \neq p$, $\nu_1(p) = 1$. Les clauses C de S se divisent en quatre catégories :

— Soit $p \in C$ et $\neg p \in C$. Dans ce cas C est trivialement vraie.
— Soit $p \in C$ et $\neg p \notin C$. Dans ce cas, $\nu_1(C) = 1$.
— Soit $\neg p \in C$ et $p \notin C$. Dans ce cas, $\nu(C) = 1$ ssi $\nu_1(C_1) = 1$ où $C = C_1 \cup \{p\}$.
— Soit $p \notin C$ et $\neg p \notin C$. Dans ce cas, $\nu(C) = 1$ ssi $\nu_1(C) = 1$

Au final, v satisfait S ssi v_1 satisfait $S_1 = \mathsf{Res} \cup \{C \in S : p \notin C \text{ et } \neg p \notin C\}$ et S_1 correspond à la valeur de S après cette itération dans la boucle. La satisfaisabilité est donc préservée par assignation des variables des clauses unitaires.

En ce qui concerne le nombre d'étape, à chaque passage dans la boucle la taille de l'ensemble de clause à considérer diminue [7]. Une variable p est assignée et toutes les clauses contenant p sont éliminées et seule des sous clauses C de clauses de la forme $C \cup \{\neg p\}$ sont obtenues par résolution unitaire. Le nombre de résolutions unitaires pour $\{p\} \in \mathsf{unit}(S)$ est borné par $occ(p)$, le nombre d'occurences de p ou $\neg p$ dans φ. Dans le pire des cas, chaque variable peut se retrouver prise dans une clause unitaire donc le nombre total de résolutions est bornée par :

$$\sum_{p \in \mathsf{var}(S)} occ(p) \leqslant \sum_{C \in \mathsf{cl}(\varphi)} |C| = |\varphi|.$$

■

Exercice 5 Donner des contre-exemples montrant que la résolution unitaire n'est pas complète pour le calcul propositionnel en général.

Exercice 6 Une formule propositionnelle 2-FNC a toutes ses clauses de longueur au plus 2.

1. Soit p une constante propositionnelle et C et D des 2-clauses, telles que $p \in C$ et $\neg p \in D$. Que peut-on dire du résolvant de C et D ?
2. Montrer que toute réfutation d'une formule 2-FNC φ par la méthode de résolution est de longueur au plus $|\mathsf{var}(\varphi)|^2$
3. En déduire un algorithme de test de la satisfaisabilité d'une formule 2-FNC utilisant au plus $|\mathsf{var}(\varphi)|^2$ étapes de résolutions.

Exercice 7 Une formule propositionnelle en FNC est dite *affine* si toutes ses clauses sont de la forme :

$$l_1 \oplus l_2 \oplus \ldots \oplus l_k$$

noindent où l_i est un littéral pris sur un ensemble de variable $\mathscr{P}$ et $\oplus$ est le connecteur *ou exclusif* défini par : $0 \oplus 0 = 1 \oplus 1 = 0$, $0 \oplus 1 = 1 \oplus 0 = 1$.

1. Montrer que toute formule affine φ sur un ensemble de variables $\mathscr{P} = \{x_1, ..., x_n\}$ est équivalente à un système d'équations linéaires S_φ sur les mêmes variables au sens suivant : pour toute valuation $v : \mathscr{P} \to \{0,1\}$, $v(\varphi) = 1$ ssi $(v(x_1), \ldots, v(x_n))$ est une solution de S_φ.
2. En déduire un algorithme "rapide" de test de satisfaisabilité pour φ.

7. On se rappelle que dans le cas de la résolution "classique" ce nombre de clauses pouvait augmenter de façon quadratique d'une étape à l'autre

Chapitre 2

Calcul des prédicats

2.1 Une première approche très informelle.

2.1.1 Les objets, les énoncés, les preuves.

En mathématiques on traite d'*objets* : les nombres, les points, les droites, les ensembles, les fonctions etc. On énonce des propriétés de ces objets de façon organisée. Certains *énoncés* initiaux, les *axiomes* sont admis, ils décrivent les propriétés de base des objet de la théorie. Par exemple la récurrence qui est une propriété fondamentale des entiers, se décrit par des axiomes. On en déduit d'autres énoncés, les *théorèmes* en utilisant certaines règles de raisonnement, souvent implicites, mais que toute personne pratiquant les mathématiques admet. Les théorèmes décrivent des propriétés de moins en moins évidentes des objets considérés. Une *démonstration* d'un énoncé est une construction qui permet de convaincre que l'on a bien utilisé pour déduire l'énoncé les règles de raisonnement communément admises. On dit alors que l'énoncé est *conséquence* des axiomes utilisés pour la démonstration.

Cet exposé est trop court pour présenter la notion formelle de démonstration, mais on peut formaliser celle-ci, et de plus on considère que d'une certaine façon cette formalisation est achevée, au sens où l'on connaît toutes les règles de raisonnement de nature logique : celles qui ont une portée tout à fait générale et ne sont pas particulières à un domaine mathématique donné. Par exemple le raisonnement par Modus Ponens, dit que de $A \Rightarrow B$ et de A on déduit B : c'est une règle logique primitive. On utilisera, dans un cadre restreint, la règle de coupure qui en est très proche. Le raisonnement par récurrence fait référence aux entiers est n'est pas une règle logique. Si la formalisation est achevée pour la logique pure [1], ce n'est pas le cas pour les entiers : d'une certaine façon elle ne peut l'être de façon satisfaisante [2].

Structures et théories.

Une structure est un ensemble muni de plusieurs fonctions ou opérations, comme l'addition, et de prédicats ou relations, comme l'ordre. C'est celle qui est commune en algèbre, elle généralise les groupes, les anneaux, les corps, les ensembles ordonnés etc., mais aussi les entiers ou les réels munis de leurs opérations habituelles (addition, multiplication etc.).

Un énoncé mathématique est écrit dans un certain langage, et on peut dire qu'il est vrai ou faux dans une structure qui interprète tous les éléments du langage. Par exemple l'énoncé $1+1=0$ est faux pour $(\mathbb{N},+)$, et vrai pour $(\mathbb{Z}/2\mathbb{Z},+)$, de même que l'énoncé « il existe un x différent de 0 tel que $x+x=0$ ».

Un exemple très simple de langage et celui de la théorie des ordres, qui n'utilise qu'un seul prédicat, $\leqslant$, et les symboles logiques, que l'on retrouve dans tous les langages, variables, implication (« si ..., alors ...»), quantificateur universel (pour tout x ...) etc. On peut dire par exemple que la structure $(\mathbb{N},\leqslant)$, possède un plus petit élément, c'est à dire que l'énoncé « il existe x tel que pour tout y, $x \leqslant y$ » est vrai dans les entiers naturels. Ce même énoncé est faux dans $(\mathbb{Z},\leqslant)$.

Une théorie est décrite par un ensemble d'énoncés, ses axiomes, écrits dans un certain langage. Par exemple la théorie des ordres totaux est constituée des quatre axiomes de réflexivité, transitivité, antisymétrie et totalité. D'après ce qui précède l'énoncé « il existe x tel que pour tout y, $x \leqslant y$ » n'est pas déterminé dans la théorie des ordres totaux : il peut être vrai, par exemple dans la structure $(\mathbb{N},\leqslant)$ ou faux, par exemple dans la structure $(\mathbb{Z},\leqslant)$, ces deux structures vérifiant les axiomes d'ordre total.

Cette remarque évidente a pour but de faire le rapport entre démonstration et validité dans une structure. Comme l'énoncé « il existe x tel que pour tout y, $x \leqslant y$ » est faux dans $(\mathbb{Z},\leqslant)$, on ne peut pas le démontrer dans la théorie des ordres totaux. Comme l'énoncé « il existe x tel que pour tout y, $x \leqslant y$ » est vrai dans $(\mathbb{N},\leqslant)$, on ne peut pas non plus démontrer sa négation dans la théorie des ordres totaux. On s'appuie pour cela sur le fait très intuitif que la validité dans une structure est conservée par une déduction correcte, même si on n'a pas examiné les règles du raisonnement.

La notion de conséquence logique, peut donc aussi se décrire par la validité dans les structures : un énoncé est conséquence d'un système d'axiomes s'il est vrai dans toute structure qui valide ces axiomes.

Le théorème de complétude de Gödel énonce que les deux façons d'aborder la conséquence logique, celle par la donnée de règles de raisonnement que nous n'avons même pas essayé de décrire, et celle par la validité dans les structures, coïncident. Plus précisément, le théorème de complétude dit que si un énoncé A n'est pas conséquence formelle des axiomes d'une théorie, on pourra toujours construire une structure dans laquelle tous les axiomes de la théorie sont vrais, mais dans laquelle l'énoncé A est faux.

1. c'est une version très informelle du théorème de complétude de Gödel
2. c'est une version très informelle du théorème d'incomplétude de Gödel

Premier et second ordre.

Pour pouvoir définir de façon précise la notion de validité dans une structure, il est nécessaire de décrire de façon précise les énoncés que l'on va interpréter, le langage dans lequel on va exprimer ceux-ci. Ce seront les langages du calcul des prédicats du premier ordre. Cela signifie essentiellement que les seules variables autorisées sont celles qui sont astreintes aux éléments de l'ensemble de base d'une structure, mais pas aux sous-ensembles de cet ensemble de base, non plus aux fonctions définies sur celui-ci, etc. En fait le plus important et que l'on ne peut pas écrire de quantificateurs autres que portant sur ces variables.

Par exemple l'axiome de bon ordre n'est pas un énoncé du premier ordre (notez bien que l'on utilise ici le mot « ordre » en deux sens qui n'ont rien à voir). En effet, si $(E, \leqslant)$ vérifie déjà les axiomes d'ordre, on dit qu'il est bien ordonné si :

$$\forall A \subset E \; (A \neq \emptyset \Rightarrow \exists x \in A \; \forall y \in A \; x \leqslant y)$$

La quantification « $\forall A \subset E \ldots$ » n'est pas du premier ordre. On dit que c'est un *énoncé du second ordre* de la théorie des ensembles ordonnés, car elle porte sur des sous-ensembles de l'ensemble de base de la structure.

Cela signifie bien sûr pas que l'on refuse en général d'écrire un tel énoncé, mais que celui-ci est considéré comme un énoncé d'une théorie plus large, par exemple la théorie des ordres plus la théorie des ensembles. On n'est pas obligé en fait de faire appel forcément à toute la théorie des ensembles, mais il y aura toujours une forme de l'axiome dit de *compréhension* qui permet de relier à une propriété des objets du langage l'ensemble des objet qui ont cette propriété, cet ensemble ayant alors lui même le statut d'objet. Les démonstrations dans ces théories ne font donc plus seulement appel aux axiomes de la théorie des ordres et aux règles de raisonnement purement logiques. On peut toujours se ramener à une théorie du premier ordre (la théorie des ensembles de Zermelo que l'on étudiera ensuite est elle-même une théorie du premier ordre) mais, d'un point de vue logique, il faut alors étudier la théorie avec les axiomes ensemblistes supplémentaires utiles.

Parmi les théories qui ne sont pas du premier ordre, nous avons vu une axiomatisation de l'arithmétique par les axiomes de Peano, qui utilise la notion d'ensemble. On peut la voir comme une théorie « modulo » la théorie des ensembles, ou comme une théorie du second ordre (les ensembles en jeu sont des sous-ensembles de $\mathbb{N}$ ou des $\mathbb{N}^k$). Il existe une version premier ordre (plus faible), l'arithmétique de Peano du premier ordre, qui permet de développer l'arithmétique élémentaire mais pas l'analyse réelle.

L'axiomatisation des réels n'est pas non plus une théorie du premier ordre. L'axiome d'Archimède fait appel aux entiers :

$$\forall x, y \in \mathbb{R} (x > 0 \Rightarrow \exists n \in \mathbb{N} \; n \cdot x > y)$$

L'axiome de la borne supérieure demande de quantifier sur les sous-ensembles de A (si on utilise la complétude, il faut quantifier sur les suites).

Par contre la théorie axiomatique des groupes, celle des anneaux, celle des corps sont des théories du premier ordre, si on examine les axiomes de base. Cependant on utilise, par exemple en théorie des groupes, la notion de groupe simple (un groupe qui ne possède pas de sous-groupe distingué propre), qui n'est pas du premier ordre.

2.2 Langages du premier ordre.

2.2.1 Introduction.

On commence par définir la *syntaxe* d'un langage du premier ordre, c'est à dire les constructions correctes pour les *termes* qui désignent des objets mathématiques, et les *formules* qui désignent des propriétés ou des assertions sur ces objets. On utilise les définitions inductives.

2.2.2 Signature.

On rappelle qu'une *signature* est la donnée d'une suite de symboles de constantes, d'une suite de symboles de fonctions chacun muni d'un entier appelé "arité" du symbole, et d'une suite de symboles de prédicats, chacun également muni d'une arité. Chacune de ces suites peut-être vide.

Sauf précision l'égalité fait toujours partie du langage (on précise parfois langage égalitaire du premier ordre), et le signe de l'égalité n'apparaît donc pas dans la signature.

On va définir dans la suite le langage égalitaire du premier ordre de signature $\mathscr{S}$, on dira plus rapidement le langage de signature $\mathscr{S}$, voire le langage $\mathscr{S}$.

Pour définir la syntaxe on utilise des définitions inductives, tout comme on l'a fait pour le calcul propositionnel, voir 1.1.1.

2.2.3 Les termes.

On définit tout d'abord les *termes* d'un langage donné, qui désignent des objets. Voyons tout d'abord un exemple.

Termes de l'arithmétique.

Le langage est celui de l'arithmétique de Peano, de signature $\mathscr{P} = (0, s, +, \cdot, \leqslant)$ avec les arités usuelles. On suppose donnée un ensemble dénombrable de variables $\mathscr{V}$. L'ensemble $\mathscr{T}_{\mathscr{P}}$ des termes de signature $\mathscr{P}$ est défini inductivement :

variable toute variable x de $\mathscr{V}$ est un terme.

constante 0 est un terme.

successeur si t est un terme $s\,t$ est un terme.

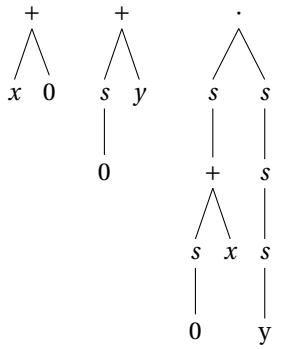
addition si t et t' sont des termes $(t + t')$ est un terme

multiplication si t et t' sont des termes $(t \cdot t')$ est un terme.

L'ensemble $\mathscr{T}$ est de plus engendré librement, c'est-à-dire que l'on a une propriété de lecture unique analogue à celle vue en calcul propositionnel au paragraphe 1.1.1, un terme possède un seul arbre de décomposition (défini de façon analogue). Voici des exemples de termes :

$$(x+0), \quad (s0+y), \quad (s(s0+x)\cdot s\,s\,s\,y),$$

avec leurs arbres de dérivations qui sont respectivement :



(bien entendu, le symbole de prédicat $\leqslant$ n'apparaît pas dans les termes).

Si $\mathscr{T}$ est vu comme un ensemble de mots sur l'alphabet défini par les variables et la signature, l'ensemble $\mathscr{T}$ est réalisé comme le plus petit ensemble de mots qui vérifie la définition inductive, et propriété de lecture unique se démontre pour la notation choisie. Rien n'empêcherait d'ajouter des parenthèses et éventuellement des "séparateurs" (",", espacements etc.).

Cas général.

On va définir les termes d'un langage de signature $\mathscr{L}$ en considérant que tous les symboles de fonction sont en notation préfixe — le symbole de fonction avant les arguments, ce que nous n'avons pas fait dans le cas de + et · ci-dessus, qui sont en notation infixe (le symbole de fonction binaire entre ses deux arguments).

L'ensemble $\mathscr{T}_{\mathscr{L}}$ des termes de signature $\mathscr{L}$ est défini inductivement par :

variable toute variable x de $\mathscr{V}$ est un terme.

constante toute constante est un terme.

fonction pour chaque symbole de fonction n-aire f de $\mathscr{L}$, si $t_1, \ldots, t_n$ sont des termes alors $f\ t_1 \ldots t_n$ est un terme.

Là aussi on suppose que l'ensemble des termes est engendré librement, c'est-à-dire que chaque terme a un seul arbre de dérivation, ou encore qu'il y a une propriété de lecture unique.

Si les termes sont vus comme des mots, avec la seule notation préfixe comme ci-dessus, on a besoin ni de parenthèses ni de séparateur, virgule ou espace (à condition de prendre comme alphabet primitif les variables, les symboles de constante et de fonction) pour obtenir la propriété de lecture unique pour cette représentation des termes. La notation infixe comme ci-dessus pour l'addition et la multiplication demande des parenthèses.

Mais Le choix des notations n'a pas grande importance. Le principal est d'assurer la propriété de lecture unique : un terme de $\mathscr{T}_{\mathscr{L}}$ possède un seul arbre de dérivation, ce qui permet d'utiliser des définitions par induction sur les termes.

Nous utiliserons des langages sans symboles de fonction ni de constante, comme le langage des ordre ($<$) ou celui de la théorie des ensembles ($\in$), qui ont donc pour seuls termes les variables.

Quelques définitions.

On peut maintenant définir par induction sur cette définition quelques notions, par exemple, la notion de *terme clos*, un terme sans variables, se définit par :

variable Une variable x de $\mathscr{V}$ n'est pas un terme clos.

constante Toute constante est un terme clos.

fonction Pour chaque symbole de fonction n-aire f de $\mathscr{L}$, $f\ t_1 \ldots t_n$ est un terme clos si $t_1, \ldots, t_n$ sont des termes clos.

Remarquons qu'un langage sans constantes n'a pas de termes clos.

On peut définir la substitution sur les termes. Étant donné un terme u et une variable x, on définit $t[u/x]$ (t dans lequel u remplace x) pour tout terme t de $\mathscr{L}$ par induction sur la définition des termes :

variable Soit y une variable dans $\mathscr{V}$, si $y \neq x$ $y[u/x] = y$, sinon $x[u/x] = u$.

constante Soit c une constantes de $\mathscr{L}$, $c[u/x] = c$.

fonction pour chaque symbole de fonction n-aire f de $\mathscr{L}$, pour $t_1, \ldots, t_n$ des termes, $f\ t_1 \ldots t_n[u/x] = f\ t_1[u/x] \ldots t_n[u/x]$.

2.2.4 Formules atomiques.

L'arithmétique.

On reprend le langage de l'arithmétique, de signature $\mathscr{P}$. Les *formules atomiques* sont formées à partir de l'égalité et des symboles de prédicat du langage, ici $\leqslant$.

égalité Si t_1 et t_2 sont des termes de $\mathscr{P}$, $t_1 = t_2$ est une formule atomique de $\mathscr{P}$.

ordre Si t_1 et t_2 sont des termes de $\mathscr{P}$, $t_1 \leqslant t_2$ est une formule atomique de $\mathscr{P}$.

Par exemple $(s0 + x) = y$, $(s0 + x) \cdot (s0 + x) = 0$, $0 \leqslant 0$ sont des formules atomiques de l'arithmétique. Remarquez qu'intuitivement les formules atomiques sont essentiellement les égalités polynomiales et les inégalités polynomiales sur les entiers.

Une formule atomique close est définie comme une formule qui n'est construite qu'avec des termes clos, ainsi parmi les formules ci-dessus la seule formule close est $0 \leqslant 0$. La formule $x = x$ n'est pas une formule close, même si intuitivement elle ne dépend pas de x.

Cas général.

On considère que le langage est égalitaire. Exceptionnellement il nous arrivera de considérer des langages non égalitaires, auquel cas on supprime bien sûr la première des clauses de la définition.

égalité Si t_1 et t_2 sont des termes de $\mathscr{L}$, $t_1 = t_2$ est une formule atomique de $\mathscr{L}$.

prédicat Pour chaque symbole de prédicat n-aire P de $\mathscr{L}$, si $t_1, \dots, t_n$ sont des termes de $\mathscr{L}$, alors $P\,t_1 \dots t_n$ est une formule atomique de $\mathscr{L}$.

On définit les formules atomiques closes (aucune variable n'apparaît dans la formule). La substitution sur les formules atomiques est la substitution sur chacun des termes : $P\,t_1 \dots t_n[t/x] = P\,t_1[t/x] \dots t_n[t/x]$.

Pour la plupart des langages étudiés, les symboles de prédicats du langage seront des symboles de prédicat binaire (l'ordre, l'appartenance) que l'on notera comme d'habitude de façon infixe (comme $\leqslant$ pour l'arithmétique ci-dessus).

2.2.5 Formules.

Définition.

On construit de nouvelles formules à l'aide de *connecteurs* et de *quantificateurs*. Prenons comme connecteurs $(\bot, \neg, \to, \vee, \wedge)$. La constante propositionnelle $\bot$ (pour l'absurde) est considérée comme un connecteur à 0 arguments. Le choix est assez arbitraire, on pourrait ajouter $\top$ (pour le vrai) et $\leftrightarrow$ pour l'équivalence, on verra que ça n'a pas grande importance. Les quantificateurs sont $\forall$ et $\exists$.

L'ensemble des *formules* du langage $\mathscr{L}$ est défini inductivement par les clauses suivantes

Formules atomiques. Les formules atomiques de $\mathscr{L}$ sont des formules.

absurde $\bot$ est une formule.

négation Si F est une formule $\neg F$ est une formule.

implication Si F et G sont des formules $(F \to G)$ est une formule.

conjonction Si F et G sont des formules $(F \wedge G)$ est une formule.

disjonction Si F et G sont des formules $(F \vee G)$ est une formule.

quantification universelle Si F est une formule et x une variable, alors $\forall x\, F$ est une formule.

quantification existentielle Si F est une formule et x une variable, alors $\exists x\, F$ est une formule.

Quand une formule n'utilise pas de quantificateurs on dit que c'est une *formule propositionnelle*.

Là encore on admettra le théorème de lecture unique, et donc on utilisera les définitions par induction.

On pourrait se contenter d'un système complet de connecteurs plus réduit, d'un seul connecteur ... A contrario l'équivalence aurait pu être ajoutée au langage, elle est vue ici comme une abréviation :

$$F \leftrightarrow G \text{ est une abréviation pour } (F \to G) \wedge (G \to F),$$

Voici deux formules du langage de l'arithmétique $\mathscr{P}$:

$$\forall x\, (x \leqslant s0 \to (x = 0 \vee x = s0)),\ \ \exists y (x = 2 \cdot y \vee x = 2 \cdot y + s0)$$

Nous avons abordé informellement la notion de variable libre et liée : la première de cette formule est close (sans variables libres), la variable x n'a que des occurrences liées dans cette formule. Dans la seconde de ces formules, les occurrences de x sont libres et celles de y sont liées. On va préciser ci-dessous ces définitions.

Occurrences libres et liées d'une variable

La notion de formule close se définit sans difficulté pour les formules propositionnelles. En présence de quantificateurs, c'est un peu plus compliqué, on peut maintenant le faire proprement grâce aux définitions par induction.

L'*occurrence* d'un symbole dans une formule désigne la place où ce symbole apparaît (cette notion peut se définir formellement, on ne le fera pas). On parle alors d'occurrence libre ou liée d'une variable dans une formule On se contente ci-dessous d'une définition où la notion d'occurrence reste implicite. Tout d'abord l'on peut définir par induction *x apparaît dans une formule F* que l'on dit également x a une occurrence dans F (définition laissée au lecteur).

On définit ensuite *x apparaît libre dans une formule F* (ou x a une *occurrence libre* dans F) également par induction sur les formules :

Formules atomiques. Si x apparaît dans une formule atomique A, x apparaît libre dans A.

absurde x n'apparaît pas libre dans $\bot$.

négation x apparaît libre dans $\neg F$ ssi x apparaît libre dans F.

implication... x apparaît libre dans $(F \to G)$ ssi x apparaît libre dans F ou x apparaît libre dans G. De même pour la conjonction et la disjonction.

quantifications Si $y \neq x$, x apparaît libre dans $\forall y\ F$ ssi x apparaît libre dans F, sinon x n'apparaît pas libre dans $\forall x\ F$. De même pour la quantification existentielle.

On peut maintenant définir une *formule close* : c'est une formule F telle qu'aucune variable de $\mathcal{V}$ n'apparaît libre dans F. On peut également définir *x apparaît liée dans F*, ou x a une *occurrence liée* dans F : cela signifie que x apparaît dans F et que x n'apparaît pas libre dans F.

On considérera que *deux formules sont égales* si elles sont identiques modulo un renommage cohérent des variables liées. Il s'agit d'une notion d'égalité purement syntaxique, que l'on va définir formellement par induction.

Substitution

La substitution, est utile en particulier pour les règles de preuves avons besoin de la substitution pour par exemple les règles de preuve de la déduction naturelle (quantificateurs, égalité).

La notion de substitution que nous allons définir et utiliser est une *substitution logique*, qui évite la *capture de variable* (voir chapitre précédent). Elle diffère de la *substitution simple* qui est la substitution sur les mots (on remplace toutes les occurrences d'un lettre par un mot). Quand on parlera de substitution sans préciser, c'est de la substitution logique qu'il s'agira. La substitution logique et la substitution simple ne diffèrent pas sur les termes. La substitution simple d'un terme à une variable se définit sur les formules par induction comme pour les termes.

On donne maintenant une définition par induction de la substitution logique. On la notera $F[t/x]$ que l'on lit "F dans laquelle t remplace x". Intuitivement la formule substituée est définie à renommage cohérent des variables liées près. Plutôt que de travailler dans le quotient des formules par renommage des variables liées, on va définir un représentant de ce quotient par induction, représentant qui n'a malheureusement rien de canonique. On va considérer que l'ensemble $\mathcal{V}$ des variables du langage est énuméré soit $\mathcal{V} = \{x_i \ / \ i \in \mathbb{N}\}$, l'énumération fournit un bon ordre sur les variables.

Étant donné un terme t, une variable x, on définit pour toute formule F la formule $F[t/x]$ par induction :

Formules atomiques. Si A est une formule atomique $A[t/x]$ est déjà défini.

absurde $\bot[t/x] := \bot$.

négation $\neg F[t/x] := \neg F[t/x]$.

implication... $(F \to G)[t/x] := (F[t/x] \to G[t/x])$ De même pour la conjonction et la disjonction.

quantifications Si $y = x$, $\forall x\ F[t/x] := \forall x F$.

Si $y \neq x$, et y n'apparaît pas dans t $\forall y\ F[t/x] := \forall y F[t/x]$.

si $y \neq x$ et y apparaît dans t, $\forall y\ F[t/x] := \forall z F[z/y][t/x]$, où z est une variable qui n'apparaît pas dans t ni dans F, et, pour être déterministe, on choisit pour z la "plus petite" variable (la première dans l'énumération) vérifiant ceci. Exceptionnellement la notation $F[z/y]$ indique la substitution simple. C'est possible car z n'apparaît pas dans F (libre ou liée).

De même pour la quantification existentielle.

Remarquons qu'à la dernière clause nous n'avons pas tout à fait respecté le schéma de définition par induction tel que nous l'avons énoncé : on a $F[z/y]$ à la place de F. C'est possible, car par exemple ces deux formules ont la même longueur.

On pourrait étendre facilement cette définition à la *substitution simultanée* des termes $u_1, \ldots, u_n$ aux variables $x_1, \ldots, x_n$, notée $F[u_1/x_1, \ldots, u_n/x_n]$. Remarquez que ce n'est en général pas la même chose que la composée des substitutions $F[u_1/x_1]\ldots[u_n/x_n]$ (par exemple si x_2 apparaît dans u_1).

On peut définir également l'*égalité des formules à renommage cohérent des variables liées près*, que l'on va noter temporairement $\approx$, afin de réserver le signe $=$ pour l'identité des formules en tant que mots. Il ne faut pas confondre ce signe "$=$" qui est un signe du métalangage, avec le signe "$=$" qui peut apparaître dans F et qui lui est un signe du langage étudié. Pratiquement cela ne pose pas réellement de problèmes, le contexte permettant de trancher. On définit par induction sur F, $F \approx E$ pour une formule E quelconque.

Formules atomiques, absurde. Si A est une formule atomique ou l'absurde $A \approx E$ ssi $A = E$.

négation $\neg F \approx E$ ssi E s'écrit $\neg E'$ et $F \approx E'$.

implication... $(F \to G) \approx E$ ssi E s'écrit $(E' \to E'')$ et $F \approx E'$ et $G \approx E''$.

De même pour la conjonction et la disjonction.

quantifications $\forall x\, F \approx E$ ssi E s'écrit $\forall y\, E'$ et $F \approx E'[x/y]$.

De même pour la quantification existentielle.

La présence du paramètre E peut rendre la définition un peu moins évidente que les précédentes. Formellement on peut considérer que l'on a définit l'ensemble des formules E telles que $F \approx E$ par induction sur F.

Dorénavant $F \approx G$ se notera $F = G$, et l'on considérera comme identiques des formules identiques à renommage des variables liées près.

Le fait de noter $F = G$ pour $F \approx G$ n'est pas innocent, on sous-entend que l'on peut travailler dans un quotient et que donc $\approx$ a les propriétés de l'égalité, symétrie, transitivité, stabilité par constructions (connecteurs, quantificateurs), par substitution etc. Ces propriétés se démontrent par induction sans grande difficulté autre que d'écriture. On les admettra.

Nous allons nous arrêter là pour la syntaxe. Même si nous n'avons pas terminé la formalisation, le lecteur doit être convaincu que l'on peut définir formellement les notions de variable libre, de formule close, de substitution etc. Nous considérerons dorénavant qu'elles sont suffisamment intuitives pour être maniées sans faire appel à ces définitions, et le plus souvent nous admettrons les propriétés « évidentes » dont nous avons besoin.

Notations des formules avec variables libres

Comme la notation des substitutions est assez lourde, on s'autorise parfois en abréviation une notation fonctionnelle (même s'il ne s'agit pas de fonction) avec les arguments entre [ ]. Par exemple on notera $F[x, y]$ une formule et $F[u, v]$ la formule obtenue à partir de la précédente en remplaçant simultanément les occurrences libres de x par le terme u et les occurrences libres de y par le terme v.

2.3 Interprétation

2.3.1 Introduction

On va maintenant définir formellement *la sémantique* d'un langage du premier ordre. Il s'agit de définir l'interprétation dans une structure donnée des termes et des formules du langage définis à la section précédente.

On va tout simplement formaliser la définition intuitive d'interprétation pratiquée par exemple en algèbre. Il est essentiel dans ce qui suit qu'il n'y ait que deux valeurs de vérité. C'est à dire que dans une structure, un énoncé clos sera vrai ou faux. Peu importe qu'éventuellement nous ne sachions pas quelle alternative est la bonne.

Pour définir la sémantique, on se sert librement des notions logique usuelles, de l'égalité (des éléments d'un structure), des quantificateurs et connecteurs logiques usuels. On parle de langage objet et métalangage : le langage objet est celui dont on a défini la syntaxe et dont on va définir la sémantique. Le métalangage est celui que l'on utilise pour décrire ceci.

2.3.2 Signature et structure

Étant donné une signature $\mathscr{S}$, une *$\mathscr{S}$-structure* $\mathscr{M}$ est la donnée :

— d'un ensemble *non vide*, soit M, appelé *ensemble de base* de la structure $\mathscr{M}$;

— d'un élément $\overline{c}^{\mathscr{M}}$ de M pour chaque symbole de constante c de $\mathscr{S}$;

— d'une fonction [3] $\overline{f}^{\mathscr{M}}$ de M^n dans M pour chaque symbole de fonction f d'arité n dans $\mathscr{S}$;

— d'un sous-ensemble $\overline{R}^{\mathscr{M}}$ de M^n pour chaque symbole de prédicat R d'arité n de $\mathscr{S}$.

Ainsi, si nous prenons le langage de l'arithmétique de signature $\mathscr{P} = (0, s, +, \cdot, \leqslant)$ défini au paragraphe 2.2.3, on peut définir $\mathscr{N} = (\mathbb{N}, \overline{0}^{\mathscr{N}}, \overline{s}^{\mathscr{N}}, \overline{+}^{\mathscr{N}}, \overline{\cdot}^{\mathscr{N}}, \overline{\leqslant}^{\mathscr{N}})$ muni des constantes, opérations et relations, usuelles pour interpréter chacun des symboles de $\mathscr{P}$. Ainsi s est interprété par la fonction de $\mathbb{N} \to \mathbb{N}$ qui ajoute 1 à un entier, + par la fonction addition usuelle (à deux arguments) etc.

Des notations comme $\overline{s}^{\mathbb{N}}$, $\overline{+}^{\mathbb{N}}$ sont assez lourdes. S'il n'y a pas d'ambiguïté sur la structure concernée, on oubliera l'indication de celle-ci, par exemple on notera $\overline{s}$, $\overline{+}$. On peut même de noter de la même façon symbole interprétation, s'il est clair d'après le contexte que l'on parle de l'interprétation.

Une autre $\mathscr{P}$-structure est $\mathbb{Z}$ muni des opérations et prédicats usuels, $\mathbb{Z}/2\mathbb{Z}$ en interprétant 0 par le 0 de $\mathbb{Z}/2\mathbb{Z}$, les opérations avec leur interprétation usuelle, et l'ordre par exemple par $\{(0,0),(0,1),(1,1)\}$ (ce qui signifie que dans cette structure, on a $0 \leqslant 0$, $0 \leqslant 1$ et $1 \leqslant 1$, mais pas $1 \leqslant 0$) [4]. On pourrait tout aussi bien construire une structure $\mathscr{N}'$ d'ensemble de base $\mathbb{N}$, où l'ordre est usuel, mais 0 est interprété par 1, s par la fonction constante nulle etc. Ce sont les axiomes que doit satisfaire la structure qui imposeront des contraintes sur l'interprétation.

L'ensemble de base d'une structure est toujours non vide [5]. Par contre il peut contenir un seul élément, mais c'est un cas assez dégénéré puisque l'interprétation des symboles de fonctions et de constantes est imposée, et que les symboles de prédicats n'ont que deux interprétations possibles ($\varnothing$ ou M^n).

2.3.3 Environnements

On veut définir l'interprétation des termes et des formules closes. Mais il n'y a pas de définition par induction sur les seules formules closes à cause des quantificateurs : si la formule $\forall x\ F$ est close, on s'attend en général à ce que la formule F dépende de x. On va donc interpréter un terme ou une formule avec des variables libres, ce qui est possible si l'on a choisit d'attribuer une certaine valeur aux variables libres en question : c'est ce que l'on appelle un *environnement*.

Plus formellement, on définira un environnement dans une structure $\mathscr{M}$ comme la donnée d'une application d'un ensemble fini de variables dans l'ensemble de base M de $\mathscr{M}$. Si $x_1, \ldots, x_p$ sont les seules variables affectées par l'environnement, et ont pour images $m_1, \ldots, m_p$ (des éléments de M) on note celui ci :

$$[x_1 := m_1, \ldots, x_p := m_p]$$

et parfois on note en abrégé : $[\vec{x} := \vec{m}]$.

3. Les fonctions sont toujours supposées *partout définies*.

4. Cet ordre n'est pas compatible avec l'addition

5. Outre que le cas où l'ensemble de base est vide a assez peu d'intérêt, cela compliquerait les systèmes de démonstrations si l'on voulait que celles-ci restent correctes dans de telles structures.

Remarquez que pour interpréter les formules propositionnelles (sans quantificateurs) et closes (donc sans variables) on peut se passer de la notion d'environnement pour définir l'interprétation.

2.3.4 Interprétation

Soit $\mathcal{M}$ une $\mathcal{S}$-structure d'ensemble de base M. On va définir successivement l'interprétation des termes (par induction), des formules atomiques, et des formules (par induction).

Interprétation des termes

L'interprétation d'un terme t du langage de signature S dans $\mathcal{M}$ pour l'environnement $[x_1 := m_1, \ldots, x_p := m_p]$ est notée $\overline{t}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p]$. C'est un élément de M qui est défini par induction sur t, pour tout environnement qui attribue une valeur à toutes les variables libres de t.

variable $\overline{x_i}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p] := m_i$ (si la variable n'est pas affectée par l'environnement l'interprétation n'est pas définie) ;

constante $\overline{c}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p] := \overline{c}^{\mathcal{M}}$ pour c une constante ;

fonction $\overline{f\, t_1 \ldots t_n}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p] :=$
$\overline{f}^{\mathcal{M}}(\overline{t_1}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p], \ldots, \overline{t_n}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p])$
pour f un symbole de fonction de $\mathcal{S}$ d'arité n.

Interprétation des formules : notations

Une formule est interprétée par "vrai" ou "faux". Pour dire qu'une formule F est *vraie dans la structure* $\mathcal{M}$ relativement à l'environnement $[x_1 := m_1, \ldots, x_p := m_p]$, on note :

$$\mathcal{M} \models F[x_1 := m_1, \ldots, x_p := m_p]$$

et on dira également que la formule F est *satisfaite* dans la structure $\mathcal{M}$, pour l'environnement $[x_1 := m_1, \ldots, x_p := m_p]$. Dans le cas contraire on notera

$$\mathcal{M} \not\models F[x_1 := m_1, \ldots, x_p := m_p]$$

.

Interprétation des formules atomiques

prédicat $\mathcal{M} \models R\, t_1 \ldots, t_n[x_1 := m_1, \ldots, x_p := m_p]$ ssi
$(\overline{t_1}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p], \ldots, \overline{t_n}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p]) \in \overline{R}^{\mathcal{M}}$, pour R un prédicat de S d'arité n, toutes les variables libres de $R\, t_1 \ldots t_n$ étant parmi $x_1, \ldots, x_p$.

égalité $\mathcal{M} \models t = t'[x_1 := m_1, \ldots, x_p := m_p]$ ssi
$\overline{t}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p] = \overline{t'}^{\mathcal{M}}[x_1 := m_1, \ldots, x_p := m_p])$, toutes les variables libres de t et t' étant parmi $x_1, \ldots, x_p$.

Remarquez que dans la dernière assertion, le signe "=" utilisé dans deux sens différents : comme symbole du langage (première occurrence), et dans son sens usuel, celui de l'identité du métalangage, pour la deuxième occurrence.

La différence entre l'égalité et les autres symboles de prédicat est que pour l'égalité, l'interprétation est imposée. La notation choisie n'a pas grande importance, on aurait tout aussi bien pu écrire la clause pour l'égalité ainsi (notons $[\vec{x} := \vec{m}]$ pour $[x_1 := m_1, \ldots, x_p := m_p]$) :

$$\mathcal{M} \models t = t'[\vec{x} := \vec{m}] \text{ ssi } (\overline{t}^{\mathcal{M}}[\vec{x} := \vec{m}], \overline{t'}^{\mathcal{M}}[\vec{x} := \vec{m}]) \in \{(x, x) \,/\, x \in M\}$$

et inversement pour la relation d'ordre $\leqslant$ sur les entiers, on pourrait écrire $\overline{u}^{\mathcal{N}} \overline{\leqslant}^{\mathcal{N}} \overline{v}^{\mathcal{N}}$ plutôt que $(\overline{u}^{\mathcal{N}}, \overline{v}^{\mathcal{N}}) \in \overline{\leqslant}^{\mathcal{N}}$.

Interprétation des formules

On peut maintenant définir l'interprétation des formules par induction.

Notez bien que pour les connecteurs binaires, définis sur F et G, il n'y a que 4 cas possibles suivant que $\mathcal{M} \models F[\overrightarrow{x} := \overrightarrow{m}]$ ou $\mathcal{M} \not\models F[\overrightarrow{x} := \overrightarrow{m}]$ et $\mathcal{M} \models G[\overrightarrow{x} := \overrightarrow{m}]$ ou $\mathcal{M} \models G[\overrightarrow{x} := \overrightarrow{m}]$.

Formules atomiques Voir le paragraphe précédent.

absurde $\mathcal{M} \not\models \bot[x_1 := m_1, \ldots, x_p := m_p]$.

négation $\mathcal{M} \models \neg F[x_1 := m_1, \ldots, x_p := m_p]$ ssi $\mathcal{M} \not\models F[x_1 := m_1, \ldots, x_p := m_p]$.

implication $\mathcal{M} \models (F \to G)[x_1 := m_1, \ldots, x_p := m_p]$ ssi

$\mathcal{M} \models F[x_1 := m_1, \ldots, x_p := m_p] \Rightarrow \mathcal{M} \models G[x_1 := m_1, \ldots, x_p := m_p]$.

Le signe $\Rightarrow$ désigne l'implication dans le métalangage. Cette clause peut paraître plus claire énoncée sous forme contraposée :

$\mathcal{M} \not\models (F \to G)[x_1 := m_1, \ldots, x_p := m_p]$ ssi

$\mathcal{M} \models F[x_1 := m_1, \ldots, x_p := m_p]$ et $\mathcal{M} \not\models G[x_1 := m_1, \ldots, x_p := m_p]$.

conjonction $\mathcal{M} \models (F \wedge G)[x_1 := m_1, \ldots, x_p := m_p]$ ssi

$\mathcal{M} \models F[x_1 := m_1, \ldots, x_p := m_p]$ et $\mathcal{M} \models G[x_1 := m_1, \ldots, x_p := m_p]x$.

disjonction $\mathcal{M} \models (F \vee G)[x_1 := m_1, \ldots, x_p := m_p]$ ssi

$\mathcal{M} \models F[x_1 := m_1, \ldots, x_p := m_p]$ ou $\mathcal{M} \models G[x_1 := m_1, \ldots, x_p := m_p]$.

Le "ou" est le ou inclusif du métalangage, usuel en mathématique. Cette clause peut s'exprimer peut-être plus clairement par contraposée :

$\mathcal{M} \not\models (F \vee G)[x_1 := m_1, \ldots, x_p := m_p]$ ssi $\mathcal{M} \not\models F[x_1 := m_1, \ldots, x_p := m_p]$ et $\mathcal{M} \not\models G[x_1 := m_1, \ldots, x_p := m_p]$.

quantification universelle $\mathcal{M} \models \forall x\, F[x_1 := m_1, \ldots, x_p := m_p]$ ssi

$\mathcal{M} \models F[x_1 := m_1, \ldots, x_p := m_p, x := m]$ pour tout élément m de M, ceci si $x \notin \{x_1, \ldots, x_p\}$.

Si $x \in \{x_1, \ldots, x_n\}$ (x est déjà affectée par l'environnement) on choisit y la "première" (dans l'ordre d'énumération des variables) non affectée dans l'environnement et on donne la définition ci-dessus pour $\forall y\, F[y/x]$.

quantification existentielle $\mathcal{M} \models \exists x\, F[x_1 := m_1, \ldots, x_p := m_p]$ ssi

$\mathcal{M} \models F[x_1 := m_1, \ldots, x_p := m_p, x := m]$ pour au moins un élément m de $\mathcal{M}$, ceci si $x \notin \{x_1, \ldots, x_p\}$.

Si $x \in \{x_1, \ldots, x_n\}$, on procède comme pour $\forall$.

Dans le cas d'une formule close F on dira qu'elle est vraie dans $\mathcal{M}$, ou satisfaite par $\mathcal{M}$, ou que $\mathcal{M}$ *est modèle* de F quand $\mathcal{M}$ est satisfaite par $\mathcal{M}$ pour l'environnement vide. On note alors

$$\mathcal{M} \models F\,.$$

Remarquez que dès qu'une formule close contient des quantificateurs, on a effectivement besoin de la notion d'environnement (non vide!) pour définir l'interprétation.

La *clôture universelle* d'une formule F est la formule close obtenue en quantifiant universellement F pour toutes les variables qui apparaissent libres dans F. Par exemple la clôture universelle de la formule $(x = y \wedge y = z) \Rightarrow x = z$ est $\forall x \forall y \forall z \left[(x = y \wedge y = z) \Rightarrow x = z\right]$. Par définition, la clôture universelle d'une formule est satisfaite dans un modèle quand la formule est satisfaite pour tous les environnement possibles affectant ses variables libres.

Une formule close de $\mathcal{S}$ est dite *universellement valide* quand elle est satisfaite dans toutes les $\mathcal{S}$-structures, et par extension une formule universellement valide est une formule dont la clôture universelle est universellement valide.

Un exemple de formule universellement valide pour n'importe quelle signature $\mathcal{S}$ est $x = x$. En effet pour toute structure $\mathcal{M}$, pour tout élément m de l'ensemble de base de $\mathcal{M}$, $\mathcal{M} \models m = m$, et donc pour toute structure $\mathcal{M}$, $\mathcal{M} \models \forall x\, x = x$.

Voyons un autre exemple, prenons une signature $\mathcal{S}$ quelconque, et soit $F(x)$ une formule de $\mathcal{S}$. alors $\forall x\, F(x) \to \exists x F(x)$ est universellement valide. En effet soit $\mathcal{M}$ une $\mathcal{S}$-structure et $[\vec{y} := \vec{m}]$ un environnement pour toutes les variables libres qui apparaissent dans F en plus de x. Supposons que $\mathcal{M} \models \forall x\, F[\vec{y} := \vec{m}]$. Cela signifie par définition que pour tout élément n de l'ensemble de base de $\mathcal{M}$ $\mathcal{M} \models F(n)[\vec{y} := \vec{m}]$. Comme cet ensemble de base est non vide, il contient au moins un élément n_0 et comme $\mathcal{M} \models F(n_0)[\vec{y} := \vec{m}]$, $\mathcal{M} \models \exists x F[\vec{y} := \vec{m}]$. On a bien montré que $\mathcal{M} \models \forall x\, F(x) \to \exists x F(x)[\vec{y} := \vec{m}]$ pour tout environnement $[\vec{y} := \vec{m}]$.

Dans l'exemple ci-dessus, n'importe quelle formule $F(x)$ convient. On parle alors de *schéma de formules*.

On a vérifié sur ces exemples que la définition de la validité dans une structure fonctionnait correctement. C'est bien de ceci qu'il s'agit, on apprend rien de neuf sur la validité des énoncés eux-mêmes.

On arrête là pour le moment au sujet des formules universellement valides. On verra dans les chapitres suivant des schémas de formules universellement valides purement propositionnels, et d'autres utilisant les quantificateurs comme le précédent.

Remarques

Cette formalisation de la sémantique des formules du premier ordre ci-dessus est due à Tarski. Plutôt que de définir l'interprétation d'une formule, il s'agit de formaliser celle-ci. Cette définition ne peut pas se comprendre sans avoir déjà pratiqué les mathématiques, en particulier la négation, la conjonction et les quantificateurs! Ainsi les connecteurs et les quantificateurs sont tout simplement définis ... en utilisant ces mêmes connecteurs dans le métalangage. Ceci peut paraître tautologique, mais ne l'est pas car cette définition formelle permet de faire proprement des démonstrations de résultats non triviaux utilisant la sémantique, comme le théorème de complétude déjà mentionné.

2.3.5 Satisfaisabilité, axiomatisation

Formules satisfaisables

Une formule close F du langage de signature $\mathscr{S}$ est dite *satisfaisable* si elle possède au moins un modèle, c'est à dire s'il existe une $\mathscr{S}$-structure $\mathscr{M}$ telle que $\mathscr{M} \models F$.

La notion de formule satisfaisable est en quelque sorte duale de la notion de formule universellement valide :

Proposition 2.3.1 *Une formule close F est satisfaisable ssi $\neg F$ n'est pas universellement valide.*

Démonstration. La formule F est satisfaisable ssi existe un modèle $\mathscr{M}$ de F. Par définition de l'interprétation $\mathscr{M} \models F$ ssi $\mathscr{M} \not\models \neg F$, donc F est satisfaisable ssi $\neg F$ n'est pas universellement valide. ■

Une formule satisfaisable définit donc une classe de structures : celles qui la satisfont. Par exemple la formule du langage de l'égalité (signature vide) :

$$\exists x \exists y \; \neg x = y$$

est satisfaite dans toutes les structures égalitaires – c'est-à-dire simplement les ensembles – ayant au moins deux éléments. On dit qu'elle *axiomatise* cette notion.

De même la formule

$$\exists x \exists y \forall z (z = x \vee z = y)$$

est satisfaite dans les ensembles ayant au plus deux éléments, et la conjonction des deux formules précédentes :

$$\exists x \exists y \; \neg x = y \wedge \exists x \exists y \forall z (z = x \vee z = y)$$

dans les ensembles ayant exactement deux éléments.

Conséquence et équivalence sémantiques

Soit Γ un ensemble de formules, et G une formule d'un langage de signature $\mathscr{S}$, alors Γ *a pour conséquence* G quand dans toute $\mathscr{S}$-structure $\mathscr{M}$, et dans tout environnement suffisant pour évaluer Γ et G, si $\mathscr{M}$ satisfait chaque formule F de Γ, alors $\mathscr{M}$ satisfait G, et on écrit $\Gamma \vdash G$. Par exemple il est facile de vérifier (on rappelle que les structures sont d'ensemble de base non vide) que :

$$\exists x \exists y \; \neg x = y \vdash \forall x \exists y \; \neg x = y \, .$$

Par définition $\vdash G$ signifie que G est universellement valide. Il est facile de vérifier que

$$F_1, \ldots, F_n \vdash C \text{ ssi } \vdash (F_1 \wedge \cdots \wedge F_n) \rightarrow C \, .$$

On dit que F et G sont équivalentes quand F a pour conséquence sémantique G et réciproquement. On note :

$$F \equiv G \text{ ssi } F \vdash G \text{ et } G \vdash F \, .$$

On vérifie facilement que :

$$F \equiv G \text{ ssi } \vdash F \leftrightarrow G\,.$$

Par exemple on a :

$$\exists x \exists y\, \neg x = y \equiv \forall x \exists y\, \neg x = y\,.$$

Quelques notations

Tout d'abord on peut s'autoriser des abréviations usuelles. Par exemple on écrira $x \neq y$ pour $\neg x = y$, « $\neq$ » n'est pas un symbole du langage, $x \neq y$ désigne la formule $\neg x = y$, et il est clair qu'à une formule de signature $\mathscr{S} \cup \{\neq\}$ apparaît, on fait correspondre par une substitution « simple » une formule du langage $\mathscr{S}$.

En généralisant les exemples précédents, on s'aperçoit facilement que l'on peut axiomatiser les ensembles à au moins (resp. au plus, resp. exactement) trois éléments, puis généraliser à un entier n quelconque. Pour axiomatiser la notion d'ensemble à au moins n éléments on écrira :

$$\exists x_1 \ldots \exists x_n \bigwedge_{0 \leqslant i < j \leqslant n} x_i \neq x_j. \tag{2.1}$$

Pour pouvoir exprimer ceci on a introduit des notations *qui ne font pas partie du langage étudié* mais qui là encore renvoient *pour chaque entier n* à une formule du premier ordre. En ce qui concerne la suite de quantificateurs $\exists x_1 \ldots \exists x_n\ F$, la signification pour chaque entier n est claire. La conjonction $\bigwedge_{0 \leqslant i < j \leqslant n} x_i \neq x_j$, même si elle pourrait se définir syntaxiquement, utilise implicitement les propriétés dite d'associativité et de commutativité de la conjonction qui sont sémantiques (l'interprétation d'une suite de conjonctions ne dépend ni de l'ordre ni du parenthésage). Tout ce qui nous intéresse est de savoir qu'il existe pour chaque entier n une formule du langage notée $\bigwedge_{0 \leqslant i < j \leqslant n} x_i \neq x_j$ vérifiant pour toute structure $\mathscr{M}$:

$$\mathscr{M} \models \bigwedge_{0 \leqslant i < j \leqslant n} x_i \neq x_j \text{ ssi pour tout couple } i, j \text{ tel que } 0 \leqslant i < j \leqslant n\ \mathscr{M} \models x_i \neq x_j\,.$$

Il faut bien prendre garde à ce que, pour prendre un exemple, on ne peut parler que de *conjonctions finies*, les conjonctions infinies, qui intuitivement ont un sens n'existent pas en langage du premier ordre. Dans le même ordre d'idée, l'entier n dans la formule (2.1) *ne fait pas partie du langage*. En particulier on ne peut quantifier sur n dans le langage. Par exemple on dirait volontiers qu'un ensemble est infini à la condition :

$$\text{Pour tout entier } n\ \ \exists x_1 \ldots \exists x_n \bigwedge_{0 \leqslant i < j \leqslant n} x_i \neq x_j.$$

Mais ceci n'est pas une formule du langage du premier ordre égalitaire (pour éviter les confusions, on n'a pas utilisé le signe $\forall$ pour la quantification universelle sur n). On peut montrer qu'il n'est pas possible d'axiomatiser la notion d'ensemble infini par une seule formule du premier ordre.

Théories

Un moyen de remédier partiellement à certains manques d'expressivité de la logique du premier ordre est d'utiliser des ensembles infinis de formules. Évidemment il faudra bien trouver un moyen « fini » de décrire cette infinité de formules.

Une *théorie du premier ordre* dans un langage de signature $\mathscr{S}$ est un ensemble de formules *closes* du langage (comme on se situe en logique du premier ordre on parlera simplement de théorie).

Une $\mathscr{S}$-structure est $\mathscr{M}$ *modèle d'une théorie* T quand elle est modèle de chacune des formules de la théorie. On note comme pour les formules $\mathscr{M} \models T$.

Une théorie est dite *satisfaisable* si elle a un modèle, c'est à dire s'il existe une structure $\mathscr{M}$ qui est modèle de la théorie. Une théorie est dite *incohérente*, inconsistante, ou contradictoire dans le cas contraire, c'est à dire si elle n'a pas de modèle.

Une propriété des $\mathscr{S}$-structures est dite *axiomatisée* par une théorie T quand les structures ont la propriété en question si et seulement si elles sont modèles de T.

Ainsi la notion d'ensemble infini est axiomatisée par la théorie (infinie) :

$$\{\exists x_1 \ldots \exists x_n \bigwedge_{0 \leqslant i < j \leqslant n} x_i \neq x_j \ /\ n \in \mathbb{N}\}. \tag{2.2}$$

Une propriété *axiomatisable* est une propriété qui est axiomatisée par une certaine théorie. Quand de plus cette propriété est axiomatisée par une théorie finie on dit qu'elle est *finiment axiomatisable*. Il

est facile de voir qu'alors la propriété est axiomatisée par une seule formule : la conjonction des axiomes de la théorie.

Tout ne peut pas se résoudre en considérant une infinité de formules. Ainsi on peut exprimer dans le langage de l'égalité pour chaque entier n qu'un ensemble possède au plus n éléments, comme devrait vous en convaincre l'écriture suivante :

$$\exists x_1 \ldots \exists x_n \; \forall z (z = x_1 \vee \cdots \vee z = x_n)$$

que l'on peut aussi écrire :

$$\exists x_1 \ldots \exists x_n \; \forall z \bigvee_{i=1}^{n} z = x_i \; .$$

Un ensemble est fini s'il vérifie :

$$\text{il existe un entier } n \text{ tel que } \exists x_1 \ldots \exists x_n \; \forall z \bigvee_{i=1}^{n} z = x_i \; .$$

mais il n'y a aucun moyen apparent d'exprimer ceci dans le langage égalitaire du premier ordre, même avec une infinité de formules. On peut montrer qu'en fait la notion d'ensemble fini ne s'axiomatise pas au premier ordre.

Exemple : l'arithmétique de Peano

Voici comment on axiomatise l'arithmétique au premier ordre.

successeur non nul $\forall x \in \mathbb{N} \; s(x) \neq 0$;

injectivité du successeur $\forall x, y \in \mathbb{N} \; (s(x) = s(y) \Rightarrow x = y)$;

récurrence Pour tout prédicat $P(x, x_1, \ldots, x_p)$ du langage de l'arithmétique (on note $\overline{a} = a_1, \ldots, a_p$) :

$$\forall \overline{a} \big(\, P[0, \overline{a}], \forall y \big(P[y, \overline{a}] \Rightarrow P[s\, y, \overline{a}] \big) \Rightarrow \forall x \; P[x, \overline{a}] \, \big) \; ;$$

addition $\forall x \; x + 0 = x$; $\forall x, y \; x + s\, y = s(x + y)$;

multiplication $\forall x \; x \cdot 0 = 0$; $\forall x, y \; x \cdot s\, y = x \cdot y + x$;

ordre $\forall x \; (x \leqslant 0 \Leftrightarrow x = 0)$; $\forall x, y \; [x \leqslant s\, y \Leftrightarrow (x \leqslant y \vee x = s\, y)]$.

Cette axiomatisation n'est pas une formalisation de celle vue en début de cours. Le choix d'un langage du premier ordre de la signature indiquée, entraîne que l'on ne pourra exprimer (par des formules du premier ordre) que des propriétés polynomiales et leurs combinaisons par connecteurs et quantifications. En particulier, la récurrence ne porte *que* sur ces propriétés. Il n'y a plus de théorème général de définition par récurrence (on a pris pour axiomes les définitions par récurrence de l'addition, de la multiplication et de l'ordre). Il s'avère que l'on peut en fait parler indirectement de certaines fonctions, comme l'exponentielle, en représentant leur graphe par une formule [6].

Cette restriction n'est cependant pas sans conséquences : la théorie est suffisante pour développer l'arithmétique élémentaire mais pas l'analyse réelle.

Théorèmes d'une théorie

Étant donné une théorie T, et un ensemble de formules Γ, on écrira parfois $\Gamma \vdash_T G$ pour $T \cup \Gamma \vdash G$, soit *L'ensemble de formule* Γ *a pour conséquence la formule* G *dans la théorie* T.

Quand on explicite la définition, cela signifie que pour toute $\mathscr{S}$-structure $\mathscr{M}$ modèle de T, pour tout environnement suffisant pour évaluer Γ et G, si $\mathscr{M}$ satisfait chaque formule F de Γ, alors $\mathscr{M}$ satisfait G.

De la même façon on note $F \equiv_T G$ pour $F \vdash_T G$ et $G \vdash_T G$.

Une formule close G est un *théorème* de la théorie T quand tout modèle de la théorie T est modèle de G, ce qui équivaut à $\vdash_T G$.

Par définition aucune structure ne satisfait l'absurde $\bot$, donc :

$$\vdash_T \bot \text{ signifie que la théorie } T \text{ est incohérente.}$$

On peut généraliser également ces définitions aux théories : une théorie T' est conséquence d'une théorie T si toute formule de T' est conséquence de T, deux théories T et T' sont équivalentes si T' est conséquence de T et T conséquence de T'.

6. Le résultat pour l'exponentielle n'est pas évident a priori ; Gödel en a eu besoin pour la démonstration de son théorème d'incomplétude en 1931. Il utilise une astuce où intervient le théorème des restes chinois, pour coder la suite des calculs successifs de l'exponentielle, par itération de multiplications.

2.3.6 Morphismes

On généralise les notions de morphisme déjà abordées en algèbre à une signature quelconque.

Étant donné une signature $\mathscr{S}$ on appelle *$\mathscr{S}$-homomorphisme* de $\mathscr{S}$-structures, une application ϕ d'une $\mathscr{S}$-structure $\mathscr{M}$ dans une $\mathscr{S}$-structure $\mathscr{N}$ qui vérifie :

constante $\phi\left(\overline{c}^{\mathscr{M}}\right) = \overline{c}^{\mathscr{N}}$ pour tout symbole de constante c de $\mathscr{S}$.

fonction $\phi\left(\overline{f}^{\mathscr{M}}(m_1,\ldots,m_n)\right) = \overline{f}^{\mathscr{N}}\left(\phi(m_1),\ldots,\phi(m_n)\right)$ pour tout symbole de fonction f d'arité n, pour tous $m_1,\ldots,m_n \in \mathscr{M}$.

prédicat Si $(m_1,\ldots,m_n) \in \overline{R}^{\mathscr{M}}$ alors $(\phi(m_1),\ldots,\phi(m_n)) \in \overline{R}^{\mathscr{N}}$, pour tout symbole de prédicat R d'arité n, pour tous $m_1,\ldots,m_n \in \mathscr{M}$.

Cette définition ne fait intervenir que la signature, et aucune notion d'axiomatique. Par exemple l'identité est un morphisme $(\mathbb{N},0,+)$ dans $(\mathbb{Z},0,+)$ (signature $(0,+)$), qui est un morphisme de monoïde mais pas de groupe. Si l'on veut parler de morphisme de groupe, on ajoutera au langage un symbole de fonction unaire "$-$" pour l'opposé.

Une conséquence immédiate de cette définition est la suivante :

Lemme 2.3.2 *Soit $\mathscr{S}$ une signature, deux $\mathscr{S}$-structures $\mathscr{M}$ et $\mathscr{N}$, ϕ un homomorphisme de $\mathscr{M}$ dans $\mathscr{N}$. Soit t un terme du langage de $\mathscr{S}$ dont les variables libres sont parmi $x_1,\ldots,x_p$. Soit $m_1,\ldots,m_p$ des éléments de $\mathscr{M}$. On a :*

$$\phi\left(\overline{t(m_1,\ldots,m_p)}^{\mathscr{M}}\right) = \overline{t(\phi(m_1),\ldots,\phi(m_p))}^{\mathscr{N}}$$

Démonstration. La preuve est immédiate par induction sur la définition des termes.

Un *monomorphisme* ou *plongement* de $\mathscr{S}$-structures de $\mathscr{M}$ dans $\mathscr{N}$ est un homomorphisme injectif ϕ de $\mathscr{M}$ dans $\mathscr{N}$ qui vérifie de plus que

prédicat $(m_1,\ldots,m_n) \in \overline{R}^{\mathscr{M}}$ **ssi** $(\phi(m_1),\ldots,\phi(m_n)) \in \overline{R}^{\mathscr{N}}$, pour tout symbole de prédicat R d'arité n, pour tous $m_1,\ldots,m_n \in \mathscr{M}$.

Un *isomorphisme* de $\mathscr{S}$-structures de $\mathscr{M}$ dans $\mathscr{N}$ est un homomorphisme bijectif ϕ de $\mathscr{M}$ dans $\mathscr{N}$ dont la réciproque est un homomorphisme de $\mathscr{S}$-structure, ou encore un plongement bijectif.

Intuitivement il devrait être clair que deux structures isomorphes vérifient les mêmes formules closes. Ce résultat se démontre par induction. Comme la définition de l'interprétation des formules closes passe par celle des formules en général, on est amené à utiliser des formules étendues soient pas des éléments de $\mathscr{M}$, soient pas des éléments de $\mathscr{N}$, pour démontrer ce lemme :

Lemme 2.3.3 *Soit $\mathscr{S}$ une signature, soit $\mathscr{M}$ et $\mathscr{N}$ et deux $\mathscr{S}$ structures isomorphes par ϕ. Pour toute formule $F(x_1,\ldots,x_p)$ du langage du premier ordre sur $\mathscr{S}$ dont les variables libres sont parmi $x_1,\ldots,x_p$, pour tout $m_1,\ldots,m_p \in \mathscr{M}$, on a :*

$$\mathscr{M} \models F(m_1,\ldots,m_p) \;\; ssi \;\; \mathscr{N} \models F(\phi(m_1),\ldots,\phi(m_p)) \; .$$

On en déduit, dans le cas des formules closes, le résultat attendu :

Proposition 2.3.4 *Soit $\mathscr{S}$ une signature, soit $\mathscr{M}$ et $\mathscr{N}$ et deux $\mathscr{S}$ structures isomorphes par ϕ. Pour toute formule close F du langage du premier ordre sur $\mathscr{S}$*

$$\mathscr{M} \models F \;\; ssi \;\; \mathscr{N} \models F \; .$$

Démonstration (lemme). On montre par induction sur la structure des formules le résultat *pour toute suite d'éléments de $\mathscr{M}$* de longueur le nombre de variables libres de la formule. Le résultat est intuitivement évident, et de fait aucun cas ne présente de difficulté. Passons en quelques uns en revue.

formules atomiques On utilise le lemme 2.3.2. Si la formule atomique est une égalité c'est une conséquence immédiate de l'injectivité de ϕ. Si la formule atomique est un prédicat, c'est une conséquence de la définition de morphisme pour ϕ et ϕ^{-1} (c'est à dire que ϕ est un plongement).

négation C'est une conséquence immédiate de l'hypothèse d'induction. On utilise la définition de la satisfaction, et le fait que l'on démontre bien une équivalence par induction.

conjonction On utilise directement la définition de la satisfaction, et l'hypothèse d'induction.

quantification universelle C'est une conséquence de l'hypothèse d'induction et de la surjectivité de ϕ. On a $F = \forall x\, F'(x_1,\dots,x_p,x)$. Soient $m_1,\dots,m_p \in \mathscr{M}$. Par définition de la satisfaction :
$\mathscr{M} \models \forall x\, F'(m_1,\dots,m_p)$ si et seulement si

$$\text{pour tout } m \text{ de } \mathscr{M},\ \mathscr{M} \models F'(m_1,\dots,m_p,m) \tag{1}$$

$\mathscr{N} \models \forall x\, F'(\phi(m_1),\dots,\phi(m_p))$ si et seulement si

$$\text{pour tout } n \text{ de } \mathscr{N},\ \mathscr{N} \models F'(\phi(m_1),\dots,\phi(m_p),n] \tag{2}$$

L'hypothèse d'induction donne pour toute suite $(m_1,\dots,m_p,m)$:

$$\begin{gathered}\mathscr{M} \models F'(m_1,\dots,m_p,m)\\ \text{ssi}\\ \mathscr{N} \models F(\phi(m_1),\dots,\phi(m_p),\phi(m))\end{gathered}$$

On doit démontrer que (1) $\Leftrightarrow$ (2).

(1) $\Rightarrow$ (2) Soit n un élément quelconque de $\mathscr{N}$, comme ϕ est surjective, il existe m dans $\mathscr{M}$ tel que $\phi(m) = n$. On utilise l'hypothèse d'induction pour $(m_1,\dots,m_p,m)$.

(2) $\Rightarrow$ (1) Soit m un élément quelconque de $\mathscr{M}$, on utilise l'hypothèse d'induction pour $(m_1,\dots,m_p,m)$.

Les autre cas se démontrent essentiellement de la même façon, par exemple la clause pour l'existentielle utilise également la surjectivité de ϕ. ■

Deux structures peuvent vérifier les mêmes formules closes d'un langage donné sans être isomorphes. C'est le cas par exemple (voir **??**) de $(\mathbb{Q},\leqslant)$ et $(\mathbb{R},\leqslant)$ (il est essentiel que ce que le langage soit restreint à la relation $\leqslant$ et du premier ordre).

2.3.7 Sous-structure

Soient deux structures $\mathscr{M}$ d'ensemble de base M et $\mathscr{N}$ d'ensemble de base N pour la même signature $\mathscr{S}$. On dit que *$\mathscr{N}$ est un sous-structure de $\mathscr{M}$* quand :

— $N \subset M$;

— L'identité sur N est un plongement de $\mathscr{N}$ dans $\mathscr{M}$.

Un sous-ensemble N de M définit une sous-structure de $\mathscr{M}$ pour les restrictions des interprétations des symboles de la signature à N si et seulement

— $N \neq \emptyset$;

— Si c est un symbole de constante de $\mathscr{S}$, $\overline{c}^{\mathscr{M}} \in \mathscr{N}$;

— L'ensemble N est clos pour les $\overline{f}^{\mathscr{M}}$, f un symbole de fonction de $\mathscr{S}$.

On retrouve des notions bien connues en algèbre (sous-groupe, sous-anneau, etc.).

On appelle *formule universelle* une formule dans laquelle les seuls quantificateurs qui apparaissent sont universels et tous en tête de la formule, *formule existentielle* une formule dans laquelle les seuls quantificateurs qui apparaissent sont existentiels et tous en tête de la formule.

Par exemple dans le langage de la théorie des groupes de signature $\left(e,\cdot,(.)^{-1}\right)$, tous les axiomes de groupes sont des formules universelles :

— $\forall x,y,z\ (x\cdot y)\cdot z = x\cdot(y\cdot z)$;

— $\forall x\ x\cdot e = x$, $\forall x\ e\cdot x = x$;

— $\forall x\ x\cdot x^{-1} = e$, $\forall x\ (x)^{-1}\cdot x = e$.

On peut remarquer que :

Proposition 2.3.5 *Si F est une formule close universelle, si $\mathscr{M} \models F$, et si $\mathscr{N}$ est une sous-structure de $\mathscr{M}$, alors $\mathscr{N} \models F$.*

Démonstration. Immédiat par induction sur l'ensemble des formules de $\mathscr{S}$ étendues aux éléments de $\mathscr{N}$. ■

Dans le cas des groupes, dont les axiomes sont universels, on retrouve le résultat bien connu que H est un sous-groupe de G si et seulement si H est non vide, contient l'élément neutre, et qu'il est stable par multiplication et passage à l'inverse.

Exercice 8 Énoncer et justifier le résultat symétrique pour les formules closes existentielles.

2.4 Formules universellement valides, équivalences

2.4.1 Tautologies

Les tautologies du calcul des prédicats fournissent immédiatement des formules universellement valides du calcul des prédicats.

Proposition 2.4.1 (propriété de substitution) *Soit une tautologie F du calcul propositionnel qui utilise les variables propositionnelles $P_1, \ldots, P_k$. Soient $G_1, \ldots, G_k$ des formules du calcul des prédicats d'un langage de signature donnée. Alors la formule obtenue à partir de F en substituant $G_1, \ldots, G_k$ à $P_1, \ldots, P_k$ est une formule universellement valide.*

La démonstration est immédiate, du fait que dans la définition sémantique de la satisfaction dans un modèle en calcul des prédicats, la définition dans le cas des connecteurs coïncide avec celle de la validation par une valuation en calcul propositionnel.

Les formules universellement valides obtenues par la propriété précédente, sont appelées *tautologies du calcul des prédicats*. Ce sont les formules valides pour des raisons purement propositionnelles, comme $\forall x\, A \Rightarrow \forall x\, A$.

Toutes les équivalences et tautologies du calcul propositionnel vu en 1.4 se généralisent donc immédiatement au calcul des prédicats.

2.4.2 Équivalences utilisant les quantificateurs

Les équivalences qui suivent ne sont plus purement propositionnelles, on les vérifie en reprenant la définition sémantique.

La signature du langage est quelconque. On désigne par F et G des formules quelconques.

Quantifications "inutiles" :

Si x n'apparaît pas libre dans F :

$$\forall x\, F \equiv F \quad \exists x\, F \equiv F \tag{1}$$

Commutation des quantificateurs :

$$\forall x \forall y\, F \equiv \forall y \forall x\, F \quad \exists x \exists y\, F \equiv \exists y \exists x\, F$$

$$\exists x \forall y\, F \vdash \forall y \exists x\, F$$

mais en général (cela dépend de F et de la structure dans laquelle on interprète la formule) la réciproque est fausse :

$$\forall y \exists x\, F \nvdash \exists x \forall y\, F$$

Négation des quantificateurs :

$$\neg \forall x\, F \equiv \exists x \neg F \quad \neg \exists x\, F \equiv \forall x \neg F \tag{2}$$

Conjonction et disjonction :

$$\forall x (F \wedge G) \equiv \forall x\, F \wedge \forall x\, G \quad \exists x (F \vee G) \equiv \exists x\, F \vee \exists x\, G \tag{3}$$

$$\exists x (F \wedge G) \vdash \exists x\, F \wedge \exists x\, G \quad \forall x\, F \vee \forall x\, G \vdash \forall x (F \vee G)$$

Mais « en général » la réciproque est fausse :

$$\exists x\, F \wedge \exists x\, G \nvdash \exists x (F \wedge G) \quad \forall x (F \vee G) \nvdash \forall x\, F \vee \forall x\, G$$

par contre si x n'apparaît pas dans G :

$$\exists x (F \wedge G) \equiv \exists x\, F \wedge G \quad \forall x (F \vee G) \equiv \forall x\, F \vee G \tag{4}$$

Remarquez que l'on pouvait déjà déduire des équivalences précédentes que si x n'apparaît pas libre dans G :

$$\exists x (F \vee G) \equiv \exists x\, F \vee G \quad \forall x (F \wedge G) \equiv \forall x\, F \wedge G \tag{4'}$$

Implication :

$$\exists x(F \to G) \equiv \forall x\, F \to \exists x\, G$$

$$\exists x\, F \to \forall x\, G \vdash \forall x(F \to G)$$

mais « en général » la réciproque est fausse :

$$\forall x(F \to G) \nvdash \exists x\, F \to \forall x\, G$$

par contre si x n'apparaît pas libre dans F :

$$\forall x(F \to G) \equiv F \to \forall x\, G \tag{5}$$

et si x n'apparaît pas libre dans G :

$$\forall x(F \to G) \equiv \exists x\, F \to G\,. \tag{6}$$

On pouvait déjà déduire des équivalences précédentes que si x n'apparaît pas libre dans F :

$$\exists x(F \to G) \equiv F \to \exists x\, G \tag{5$'$}$$

et si x n'apparaît pas libre dans G :

$$\forall x(F \to G) \equiv \exists x\, F \to G\,. \tag{6$'$}$$

À l'aide des équivalences (2) qui précèdent, et des équivalences propositionnelles, on montre facilement que toute formule de la logique du premier ordre est équivalente à une formule qui n'utilise que $\exists, \wedge, \neg$, ou encore $\forall, \to, \bot$. On peut donc se restreindre à l'un des ces ensembles de quantificateurs et connecteurs pour démontrer des propriétés sémantiques des formules.

Exercice 9 En calcul des prédicats pour une signature quelconque $\mathscr{S}$, $B[x]$ est une formule à une seule variable libre x et $R[x,y]$ une formule avec deux variables libres x et y. Montrer que les deux énoncés suivant sont des tautologies :

$$\exists x \forall y \big(B[x] \to B[y]\big) \qquad \text{(formule du buveur)}$$

$$\neg\exists x \forall y \big(R[x,y] \leftrightarrow \neg R[y,y]\big) \qquad \text{(paradoxe du barbier)}$$

Propriétés de l'égalité

En calcul des prédicats égalitaire, l'égalité n'est pas une relation ordinaire : elle est toujours interprétée par l'identité. On en déduit immédiatement que l'identité entraîne l'égalité, et que deux objets égaux ont les mêmes propriétés, soit :

Réflexivité de l'égalité $\forall x\ x = x$;

Propriété fondamentale de l'égalité Pour toute formule F du langage dont les variables libres sont parmi $z, z_1, \ldots, z_n$

$$\forall z_1 \ldots \forall z_n \forall x \forall y \big[\big(x = y \wedge F[x/z]\big) \Rightarrow F[y/z]\big]\,;$$

La seconde propriété a pour conséquence que l'on peut remplacer dans une formule certaines occurrences d'un terme par un terme qui lui est égal.

La transitivité de l'égalité est essentiellement un cas particulier de la propriété fondamentale. La symétrie s'en déduit par la réflexivité, en interprétant $x = x$ comme $z = x[x/z]$.

transitivité de l'égalité $\forall x \forall y \forall z \big[\big(x = y \wedge y = z\big) \to x = z\big]$;

Symétrie de l'égalité $\forall x \forall y \big(x = y \to y = x\big)$.

Il est possible, et parfois utile de traiter l'égalité comme une relation ordinaire en calcul des prédicats non égalitaire. La signature est alors étendue par un signe de relation binaire que l'on note encore « = ». La réflexivité et la propriété fondamentale sont pris comme axiomes pour l'égalité, et ajoutés systématiquement aux théories considérées. Les modèles sont différents : rien n'empêche maintenant que deux éléments distincts d'un modèle de la théorie de l'égalité soit égaux (au sens en relation par l'égalité) dès qu'ils ont exactement les mêmes propriétés exprimables dans le langage. On retrouve cependant un modèle du calcul des prédicats égalitaire par passage au quotient par la relation d'équivalence qui interprète l'égalité dans le modèle initial. La propriété fondamentale assure que l'on obtient un modèle des mêmes formules.

Proposition 2.4.1 *Soit $\mathscr{S}$ une signature, T une théorie dans le langage de signature $\mathscr{S}$ avec égalité; Alors T possède un modèle en calcul des prédicats égalitaires (égalité interprétée par l'identité) si et seulement la théorie T augmentée des axiomes de l'égalité (réflexivité, propriété fondamentale) possède un modèle en calcul des prédicats non égalitaire pour la signature $\mathscr{S} \cup \{=\}$.*

Démonstration. Le sens direct est évident, il suffit d'interpréter = par l'identité (la relation diagonale). Pour la réciproque, soit $\mathscr{M}$ un modèle non égalitaire de T et des axiomes de l'égalité (signature $\mathscr{S} \cup \{=\}$) d'ensemble de base M. les propriétés de réflexivité de symétrie et de transitivité assurent que l'égalité est interprété par une relation d'équivalence $=^{\mathscr{M}}$. Si f est un symbole de fonction k-aire de la signature, $f^{\mathscr{M}}$ son interprétation dans $\mathscr{M}$ est compatible avec cette relation, car l'on déduit de la propriété fondamentale

$$\forall x_1 \dots x_k \forall y_1 \dots y_k \left[\left(x_1 = y_1 \wedge \dots x_k = y_k \right) \rightarrow f x_1 \dots x_k = f y_1 \dots y_k \right] . \qquad (C_1)$$

De même pour les éventuels symboles de prédicats k-aires de la signature

$$\forall x_1 \dots x_k \forall y_1 \dots y_k \left[\left(x_1 = y_1 \wedge \dots x_k = y_k \right) \rightarrow P x_1 \dots x_k \rightarrow P y_1 \dots y_k \right] . \qquad (C_2)$$

Ceci assure que l'on peut définir de façon cohérente fonctions et prédicats par passage au quotient sur $M/=^{\mathscr{M}}$, définissant ainsi une structure $\mathscr{N}$. On note $\dot{m}$ la classe d'équivalence pour $=^{\mathscr{M}}$ d'un élément m de M. Alors pour toute formule F du langage, et tout environnement $[\vec{x} := \vec{m}]$ suffisant pour l'interpréter :

$$\mathscr{M} \models F[\vec{x} := \vec{m}] \text{ si et seulement si } \mathscr{N} \models F[\vec{\dot{m}}] .$$

On démontrer d'abord la propriété pour les formules atomiques $x = t$ par induction sur la structure du terme t (x est une variable) , puis pour les formules atomiques égalitaires quelconques, puis pour une formule quelconque par induction sur la structure de F. ■

En calcul des prédicats du premier ordre, la propriété fondamentale de l'égalité demande une infinité d'axiomes : un pour chaque formule. Quand on explicite la démonstration de la proposition précédente, on se rend compte que la propriété fondamentale de l'égalité restreinte aux formules atomiques suffit. On en déduit la proposition suivante.

Proposition 2.4.2 *Dans l'énoncé de la proposition fondamentale, on peut remplacer la propriété fondamentale de l'égalité par la symétrie, la transitivité de l'égalité, et les axiomes de compatibilité avec l'égalité pour chaque symbole de fonction et chaque symbole de prédicat de la signature (du type (C_1) et (C_2) ci-dessus).*

Les axiomes donnés par la proposition sont universels, ce qui peut être exploité en raisonnement automatique.

Cependant utiliser ces axiomes au lieu de la propriété fondamentale pour une formule quelconque revient, à chaque fois que l'on a besoin de remplacer un terme par un terme qui lui est égal, à déconstruire puis reconstruire la formule.

D'autres quantificateurs.

On ne peut pas vraiment parler de système complet pour la logique du premier ordre. Par exemple des quantifications comme «Il existe une infinité de x tels que F » ne s'expriment pas dans le langage du premier ordre.

Par contre il est bien connu que l'on peut exprimer les quantificateurs « il existe au plus un x tel que F » et « il existe un unique x tel que F » en calcul des prédicats avec égalité :

$$!x\, F \equiv_d \forall y \forall y' (F[y/x] \rightarrow F[y'/x] \rightarrow y = y') .$$

$$\exists !x\, F \equiv_d \exists x\, F \wedge !\, x\, F .$$

On montre facilement que :

$$\exists !x\, F \equiv \exists x (F \wedge \forall y (F[y/x] \rightarrow y = x)) .$$
$$\neg !x\, F \equiv \exists y \exists y' (F[y/x] \wedge F[y'/x] \wedge y \neq y') .$$
$$\neg \exists !x\, F \equiv \forall x \neg F \vee \exists y \exists y' (F[y/x] \wedge F[y'/x] \wedge y \neq y') .$$

Exercice 10 Exprimer en calcul des prédicats égalitaire « il existe au plus un couple (x, y) tel que F », « il existe un unique couple (x, y) tel que F ». Généraliser aux n-uplets.

2.5 Formes prénexes et formes de Skolem

2.5.1 Formes prénexes

Une formule F est dite *sous forme prénexe* quand les quantificateurs n'apparaissent dans F qu'en tête de la formule, c'est à dire que F s'écrit :

$$Q_1 x_1 \dots Q_n x_n \; F_0$$

où F_0 est purement propositionnelle et les Q_i sont des quantificateurs $\forall$ ou $\exists$.

Par exemple les formules suivantes sont sous forme prénexe :

$$\forall x \forall y (f\, x = f\, y \to x = y) \; ; \;\; \forall x \exists y \; f\, x = y \; ;$$

la formule suivante n'est pas sous forme prénexe :

$$\forall x \forall y (x \geqslant y \to \exists z \; x + z = y)$$

mais on montre facilement qu'elle est équivalente à la formule sous forme prénexe suivante :

$$\forall x \forall y \exists z (x \geqslant y \to x + z = y)$$

Plus généralement En utilisant les équivalences du paragraphe 2.4.2, on montre que :

Proposition 2.5.1 *Toute formule F d'un langage $\mathscr{L}$ est équivalente à une formule du même langage sous forme prénexe.*

Démonstration. On prouve le résultat par induction sur la structure des formules.

Formules atomiques, $\bot$. Les formules atomiques de $\mathscr{L}$ sont sous forme prénexe, de même $\bot$.

négation Si F est une formule équivalente à la forme prénexe :

$$F \equiv Q_1 x_1 \dots Q_n x_n \; F_0$$

alors en utilisant les équivalences (2) du paragraphe 2.4.2

$$\neg F \equiv Q_1^{\bot} x_1 \dots Q_n^{\bot} x_n \; \neg F_0$$

où $Q_i^{\bot}$ est le connecteur dual de Q_i ($Q_i^{\bot}$ est $\forall$, si Q_i est $\exists$, $Q_i^{\bot}$ est $\exists$, si Q_i est $\forall$).

conjonction Si F et G sont des formules chacune équivalente à une forme prénexe :

$$F \equiv Q_1 x_1 \dots Q_n x_n \; F_0 \quad G \equiv Q'_1 x'_1 \dots Q'_m x'_m \; G_0$$

on suppose de plus, modulo renommage des variables liées, que les ensembles de variables $x_1, \dots, x_n$ et $x'_1, \dots, x'_m$ sont disjoints. On obtient alors, en utilisant les équivalences (1) et parmi (3) et (4) celles qui concernent la conjonction :

$$F \wedge G \equiv Q_1 x_1 \dots Q_n x_n Q'_1 x'_1 \dots Q'_m x'_m (F_0 \wedge G_0) \; .$$

On remarque d'ailleurs que l'ordre entre les quantificateurs Q_i et Q'_i n'a pas d'importance : on peut tout aussi bien placer les Q'_i en premier, les alterner, le tout et de conserver l'ordre entre les Q_i et l'ordre entre les Q'_i. Il pourrait y avoir par ailleurs des mises sous forme prénexe plus économique, par exemple utilisant (3) directement.

disjonction analogue au cas précédent.

implication analogue aux deux cas précédents, il faut utiliser les équivalences (5) et (6).

quantifications Si F est une formule équivalente à une forme prénexe et x une variable, alors $\forall x \; F$ et $\exists x \; F$ sont évidemment équivalentes à des formules sous forme prénexe. ■

Il est clair qu'il n'y a pas unicité de la forme prénexe, ne serait-ce que parce qu'il est toujours possible d'ajouter des quantificateurs « inutiles » (équivalences (1) du paragraphe 2.4.2). On peut essayer de minimiser le nombre de quantificateurs dans la mise sous forme normale, et également de minimiser le nombre d'alternances de groupe de quantificateurs de même nature (nous dirons alternances de quantificateurs) ce qui n'assure pas plus l'unicité. Intuitivement, le langage étant fixé, plus une formule

prénexe a d'alternances de quantificateurs, plus elle est « compliquée » à comprendre et à démontrer. Par exemple l'énoncé de l'injectivité de f définie de A dans B, $\forall x \in A \forall y \in A(f(x) = f(y) \Rightarrow x = y)$ n'a pas d'alternances de quantificateurs, et semble plus « simple » que l'énoncé de la surjectivité de f, $\forall y \in B \exists x \in A(f(x) = y)$ qui contient une alternance de quantificateur. Les énoncés exprimant l'existence d'une limite en un point ont plus d'alternances et semblent plus « compliqués ». Les alternances de quantificateurs fournissent un cadre par exemple pour analyser la complexité des sous-ensembles de $\mathbb{N}$ qui sont définissables par une formule de l'arithmétique, la complexité de certains problèmes algorithmiques etc.

En composant avec les résultats de mise sous forme normale en calcul propositionnel, on obtient que toute formule est équivalente à une formule sous forme prénexe dont la partie propositionnelle est sous forme normale disjonctive, ou conjonctive.

2.5.2 Skolemisation

On ne peut faire disparaître par équivalence les alternances de quantificateurs. Il est par contre possible d'obtenir de telles formules par *équisatisfaisabilité,* en étendant le langage (et donc cela ne contredit pas ce qui précède). Deux formules F et G sont *équisatisfaisables* quand F a un modèle ssi G a un modèle.

Étant donnée une formule F du langage $\mathscr{L}$, il est possible de donner une formule équisatisfaisable F_s dans un langage $\mathscr{L}'$ étendant $\mathscr{L}$, telle que F_s est sous forme prénexe et ne contient que des quantificateurs universels. On construit F_s à partir d'une forme prénexe de F en ajoutant un nouveau symbole de fonction f_i pour chaque quantificateur existentiel $\exists x_i$ qui a autant d'arguments que de quantificateurs universels antérieurs dans la forme prénexe, soient $x_{j_1}, \ldots, x_{j_k}$, et en remplaçant dans la partie propositionnelle de la forme prénexe de F chaque occurrence de variable x_i par le terme $f_i x_{j_1} \ldots x_{j_k}$. Quand les quantificateurs existentiels ne sont pas précédés de quantificateurs universels, les fonctions à 0 arguments introduites sont assimilées à des constantes. On appelle une formule ainsi construite *forme de Skolem de F*, et les nouveaux symboles de fonctions ou de constantes ainsi introduits des fonctions ou constantes de Skolem.

Les symboles de fonctions et constantes de Skolem sont nécessairement distincts entre eux, et distincts des symboles de constantes et fonctions de la signature originale.

Par exemple, prenons pour langage (f), une forme de Skolem de $\forall y \exists x\, fx = y$ est $\forall y\, fgy = y$ formule du langage (f, g). Intuitivement g est une fonction réciproque à droite de f. Un modèle de cette dernière formule fournit évidemment un modèle de la première formule, en « oubliant » l'interprétation de g. Un modèle $\mathscr{M} = (M, \overline{f})$ de la première formule fournit un modèle de sa forme de Skolem : on construit l'interprétation de g en choisissant pour chaque élément m de M un élément m' de M qui vérifie $\mathscr{M} \models fx = y[x := m, y := m']$ (cette démonstration utilise l'axiome du choix si M est infini).

Il s'agit bien seulement d'équisatisfaisabilité et non d'équivalence. Sur l'exemple précédent la formule $\forall y \exists x\, fx = y$ qui exprime que f est surjective, ne peut être équivalente à la formule $\forall y\, fgy = y$ qui exprime que g est la réciproque à droite de F, puisque si c'était le cas elles le seraient pour n'importe quelle fonction g, ce qui est manifestement faux. On sait en mathématiques (et en théorie des ensembles avec axiome du choix) que la surjectivité de f équivaut à l'*existence* d'une réciproque à droite g. Mais la quantification sur une fonction ne s'exprime pas directement au premier ordre.

On a exposé sur un cas particulier simple la démonstration de la proposition suivante.

Proposition 2.5.2 *Soit F une formule prénexe d'un langage de signature $\mathscr{S}$, et soit F_s sa forme de Skolem de F, une formule universelle dans un langage dont la signature $\mathscr{S}'$ est étendue aux fonctions et constantes de Skolem introduits, alors F et F_s sont équisatisfaisables, c'est-à-dire que*

Il existe une $\mathscr{S}$-structure $\mathscr{M}$ telle que $\mathscr{M} \models F$ si et seulement s'il existe une $\mathscr{S}'$-structure $\mathscr{M}'$ telle que $\mathscr{M}' \models F_s$.

Démonstration. Il s'agit essentiellement d'itérer la démonstration développée sur un cas particulier ci-dessus. On introduit pour les besoins de la démonstration une notion de skolemisation partielle : $F_{s,k}$, est la formule du langage étendu aux symboles de Skolem, définie comme la forme de Skolem, mais en substituant seulement les k premiers quantificateurs existentiels de la formule F. Si F possède n quantificateurs existentiels et $k \geqslant n$, alors $F_{s,k} \equiv F_{s,n} \equiv F$.

On montre alors par récurrence sur k, que F et $F_{k,s}$ sont équisatisfaisables. Pour $F_{s,0} \equiv F$ c'est évident.

On suppose le résultat pour $F_{k,s}$. Supposons $k < n$. Supposons que $F_{k,s}$ s'écrive :

$$F_{k,s} \equiv \forall x_1 \ldots \forall x_p \exists y G$$

alors, f étant le symbole de fonction de Skolem associé à y (de constante si $p = 0$), $F_{k+1,s}$ s'écrit :

$$F_{k+1,s} \equiv \forall x_1 \dots \forall x_p G[f x_1 \dots x_p / y] .$$

Dans le langage étendu à tous les symboles de Skolem de F, $F_{k,s}$ et $F_{k+1,s}$ sont équisatisfaisables. En effet,soit $\mathscr{M}'$ une structure pour la signature étendue d'ensemble de base M si $\mathscr{M}' \models \forall x_1 \dots \forall x_p G[f x_1 \dots x_p / y]$ alors, $\mathscr{M}' \models \forall x_1 \dots \forall x_p \exists y G$ (il suffit pour chaque p-uplet $(m_1, \dots, m_p)$ d'éléments de M d'interpréter y par $\overline{f}^{\mathscr{M}'}(m_1, \dots, m_p)$). Réciproquement soit une structure $\mathscr{M}$ d'ensemble de base M telle que $\mathscr{M} \models \forall x_1 \dots \forall x_p \exists y G$, alors on obtient par l'axiome du choix une fonction $\phi : M^p \to M$ en choisissant pour tout p-uplet $(m_1, \dots, m_p)$ d'éléments de M un m telle que $\mathscr{M} \models G[x_1 := m_1, \dots, x_p : m_p, y := m]$ (il en existe au moins un par définition de la satisfaction). En modifiant $\mathscr{M}$ en interprétant f par $\overline{f}^{\mathscr{M}'} = \phi$ on obtient une nouvelle structure $\mathscr{M}'$ telle que $\mathscr{M}' \models \forall x_1 \dots \forall x_p G[f x_1 \dots x_p / y]$.

On en conclut que F et F_s sont équisatisfaisables dans le langage étendu aux symboles de Skolem. Comme F ne contient aucune symbole de Skolem, F est satisfaisable dans le langage initial si et seulement si F est satisfaisable dans le langage étendu (il suffit d'interpréter les nouveaux symboles de façon arbitraire dans le sens direct, d'oublier leur interprétation pour la réciproque), d'où le résultat. ■

Exemples (skolemisation). Soient $F := \exists y \forall x \; y \leqslant x$ et $G := \forall x \exists y \; y \leqslant x$,

- pour skolémiser F, on introduit un nouveau symbole de constante c, $F_s := \forall x \; c \leqslant y$;
- pour skolémiser G on introduit un nouveau symbole de fonction f d'arité 1, $G_s := \forall x \; f(x) \leqslant x$.

2.6 Méthode de Herbrand

Jacques Herbrand est un mathématicien et logicien français qui a disparu dans un accident en montage à 23 ans en 1931. Il a développé dans sa thèse (soutenue en 1930) une méthode qui permet de déterminer si une formule du calcul des prédicats non égalitaire est universellement valide en réduisant la question au calcul propositionnel.

Sa méthode ne donne un résultat que si F est universellement valide. Dans le cas opposé, c'est-à-dire si $\neg F$ est satisfaisable, elle ne permet généralement pas de conclure : on dit que c'est une méthode de *semi-décision*. Il faut imaginer un algorithme, prenant en entrée une formule, dont l'exécution peut ne pas terminer. Quand elle termine alors la formule est universellement valide. Tant qu'elle n'a pas terminé on ne sait pas si c'est parce que l'exécution va terminer plus tard, et alors la formule sera universellement valide, ou bien si c'est parce que l'exécution ne terminera jamais parce que la la formule n'est pas universellement valide. Tel qu'il est décrit, l'algorithme ne donne aucun moyen de trancher tant que l'exécution n'a pas terminé, et donc ne peut jamais permettre de conclure que la formule n'est pas universellement valide.

On ne peut pas faire mieux, du moins en général. Alonzo Church puis Alan Turing ont montré en 1936, qu'il n'existait pas de méthode de décision (une méthode algorithmique, que l'on puisse programmer), pour déterminer si une formule du langage de l'arithmétique est ou non universellement valide. C'est encore le cas par exemple dès que la signature contient un symbole de relation binaire.

Dans cette section nous nous intéresserons au problème dual, celui de la satisfaisabilité d'une formule F ou de sa *réfutabilité*. On va donner une méthode de semi-décision qui permet de conclure quand la formule est réfutable ou contradictoire, c'est-à-dire non satisfaisable.

Une formule n'est pas satisfaisable si et seulement si sa négation est universellement valide. On retrouve donc le résultat de Herbrand en s'intéressant à la négation de la formule considérée : F est universellement valide si et seulement si $\neg F$ est contradictoire. Herbrand travaillait cependant dans un cadre plus général que celui qui va être abordé ici (nous allons utiliser la mise sous forme prénexe des formules, Herbrand procédait de façon plus fine).

Sauf précision, dans cette section le calcul des prédicats est *non égalitaire*.

2.6.1 Domaine et structure de Herbrand

L'ensemble des termes clos d'une certaine signature $\mathscr{S}$ est appelé *domaine de Herbrand* pour cette signature. On le note dans la suite $D_{\mathscr{S}}$.

Les éléments du domaine de Herbrand sont de simples objets syntaxiques, on peut les voir comme des mots sur l'alphabet constitué des symboles de constante et de fonction de la signature. Le domaine de Herbrand est aussi l'algèbre librement engendrée par les symboles de fonction à partir des constantes de la signature.

Ce domaine n'est non vide que si la signature contient au moins un symbole de constante. On va supposer dans la suite que c'est le cas. Comme l'ensemble de base est supposé non vide peut ajouter un nouveau symbole de constante c à une signature $\mathscr{S}$ sans perte de généralité : une $\mathscr{S}$-structure s'enrichissent en une $\mathscr{S}'$-structure (en interprétant arbitrairement c), qui satisfait les mêmes formules du langage $\mathscr{S}$.

Exemples.

- Si la signature ne contient pas de symbole de fonction, le domaine de Herbrand est constitué des seules constantes.
- Soit la signature $\mathscr{S} = (0, s, \leqslant)$, 0 symbole de constante, s de fonction unaire, $\leqslant$ de prédicat binaire. Alors le domaine de Herbrand $D_{\mathscr{S}}$ est infini, il a pour éléments les termes (le symbole de prédicat n'intervient pas) :
$$0,\ s0,\ ss0, sss0, \ldots, s^n0, \ldots$$
où s^n0 est le terme $\underbrace{s \ldots s}_{n} 0$.
- Si on ajoute à cette signature un symbole de fonction binaire + (avec la notation infixe usuelle), alors tous les termes précédent appartiennent au domaine de Herbrand $D_{\mathscr{S}}$ mais aussi par exemple
$$0+0,\ 0+s0,\ 0+(0+0),\ s0+0,\ s0+s0,\ s0+(0+0),\ (0+0)+0,\ \ (0+0)+s0,\ (0+0)+(0+0), \ldots$$
Tous ces termes sont bien des objets distincts du domaine de Herbrand.

Il est clair que, dès que la signature $\mathscr{S}$ possède un symbole de fonction (d'arité au moins un) et un symbole de constante, le domaine de Herbrand $D_{\mathscr{S}}$ est infini.

Une *$\mathscr{S}$-structure de Herbrand* $\mathscr{H}_{\mathscr{S}}$ est une $\mathscr{S}$-structure dont l'ensemble de base est le domaine de Herbrand $D_{\mathscr{S}}$, et dont les constantes et fonctions s'interprètent « syntaxiquement » sur le domaine de Herbrand, c'est-à-dire que si c est une constante et f est un symbole de fonction à k arguments de $\mathscr{S}$:
$$\overline{c}^{\mathscr{H}_{\mathscr{S}}} = c\ ;\ \overline{f}^{\mathscr{H}_{\mathscr{S}}}(t_1, \ldots, t_k) = f\, t_1 \ldots, t_k\ .$$

Par exemple pour la signature $\mathscr{A} = (0, s, \leqslant)$:
$$\overline{0}^{\mathscr{H}_{\mathscr{A}}} = 0\ ;\ \overline{s}^{\mathscr{H}_{\mathscr{A}}}(s^n 0) = s^{n+1} 0$$

c'est-à-dire que sur ce domaine de Herbrand $D_{(0,s)}$, qui est en bijection avec l'ensemble des entiers « ordinaires » (les entiers du métalangage), s doit être interprété pas le successeur dans une structure de Herbrand. Un modèle d'une formule qui est une structure de Herbrand est appelé *modèle de Herbrand* de cette formule. On voit facilement (par induction sur la structure du terme) que pour tout terme clos t du langage de signature $\mathscr{S}$:
$$\overline{t}^{\mathscr{H}_S} = t\ .$$

plus généralement avec un environnement suffisant pour un terme quelconque t de ce langage :
$$\overline{t}^{\mathscr{H}_S}[x_1 := t_1, \ldots, x_k := t_k] = t[t_1/x_1, \ldots, t_k/x_k]$$

et pour une formule du langage F :
$$\overline{F[t_1/x_1, \ldots, t_k/x_k]}^{\mathscr{H}_S} = \overline{F}^{\mathscr{H}_S}[x_1 := t_1, \ldots, x_k := t_k]\ .$$

Deux $\mathscr{S}$-structures de Herbrand ne diffèrent que par l'interprétation des prédicats. Dans le cas de l'exemple de la signature $\mathscr{A} = (0, s, \leqslant)$, ils ne diffèrent que par l'interprétation de $\leqslant$.

Une structure de Herbrand est entièrement définie par la valeur de vérité des formules atomiques closes du langage, les $P\, t_1 \ldots t_p$, pour P un symbole de prédicat de la signature et $t_1, \ldots, t_p$ des termes clos du langage, puisque cela revient à interpréter $P x_1 \ldots, x_p$ dans tous les environnements possibles pour P (rappelons que nous sommes en calcul des prédicats non égalitaire).

Dit autrement une valuation v sur l'ensemble des formules atomiques closes du langage détermine une structure de Herbrand $\mathscr{H}_v$. On appelle *base de Herbrand* pour la signature $\mathscr{S}$, l'ensemble des formules atomiques closes dans ce langage, et on le notera dans la suite $\mathscr{B}_{\mathscr{S}}$.

Ainsi dans l'exemple de la signature $\mathscr{A} = (0, s \leqslant)$, la base de Herbrand est
$$\mathscr{B}_{\mathscr{A}} = \{s^m 0 \leqslant s^n 0 \ /\ m, n \in \mathbb{N}\}$$

et une valuation v sur $\mathscr{B}_{\mathscr{A}}$ détermine la structure de Herbrand $\mathscr{H}_{\mathscr{A},v}$ d'ensemble de base $\{s^n 0 \ /\ n \in \mathbb{N}\}$.

2.6.2 Formules universelles et structures de Herbrand

Une formule close sans quantificateurs peut-être vue comme une formule du calcul propositionnel construit sur des variables propositionnelles qui sont les formules atomiques closes du langage, la base de Herbrand $\mathscr{B}_{\mathscr{S}}$.

Elle est satisfaite dans une structure de Herbrand $\mathscr{H}_{\mathscr{S},v}$ si et seulement si la valuation v correspondante définie sur $\mathscr{B}_{\mathscr{S}}$ valide cette formule en tant que formule du calcul propositionnel :

$$\mathscr{H}_{\mathscr{S},v} \models F \text{ ssi } v(F) = 1 \ (F \text{ sans quantificateurs}).$$

Maintenant, par définition de la satisfaction adaptée au cas des structures de Herbrand, une formule close universelle du langage de signature $\mathscr{S}$, soit $\forall x_1 \dots \forall x_n \, F(x_1, \dots, x_n)$, où $F(x_1, \dots, x_n)$ est sans quantificateurs et avec $x_1, \dots, x_n$ pour seules variables libres, F est satisfaite dans la structure de Herbrand $\mathscr{H}_{\mathscr{S},v}$ si et seulement si toutes les formules possibles obtenues en substituant des termes clos (de $D_{\mathscr{S}}$) aux variables libres de F sont satisfaites par la même structure, c'est-à-dire que :

$$\mathscr{H}_{\mathscr{S},v} \models \forall x_1 \dots \forall x_n \, F(x_1, \dots, x_n) \ \text{ ssi } \ \text{pour tout } (t_1, \dots, t_n) \in D_{\mathscr{S}}^n \ \ \mathscr{H}_{\mathscr{S},v} \models F(t_1, \dots, t_n) \, .$$

La satisfaction d'une formule universelle est donc ramenée à celle d'un ensemble de formules du calcul propositionnel sur l'ensemble des formules atomiques closes $\mathscr{B}_{\mathscr{S}}$. Cet ensemble est infini dès que le domaine de Herbrand est infini.

Or justement, dans le cas des formules universelles, il suffit de tester la satisfaisabilité sur des structures de Herbrand, comme le montre le lemme suivant. C'est donc ce résultat qui va permettre de se ramener au calcul propositionnel.

Lemme 2.6.1 (Formules universelles et structures de Herbrand) *Soit F une formule universelle, alors F est satisfaisable si et seulement si elle est satisfaisable dans une structure de Herbrand.*

Démonstration. Si F est satisfaisable dans une structure de Herbrand, alors F est satisfaisable. Pour la réciproque on suppose que F est satisfaite dans une structure $\mathscr{M}$. On définit (cf. paragraphe précédent) une structure de Herbrand $\mathscr{H}_{\mathscr{M}}$ par

$$\text{pour toute formule atomique close, } P t_1 \dots t_p \ \ \mathscr{H}_{\mathscr{M}} \models P t_1 \dots t_p \text{ ssi } \mathscr{M} \models P t_1 \dots t_p \, .$$

Alors $\mathscr{M}$ et $\mathscr{H}_{\mathscr{M}}$ satisfont les mêmes formules closes sans quantificateur, qui sont construites propositionnellement à partir des formules atomiques closes.

Soit maintenant une formule close universelle F qui s'écrit $F = \forall x_1 \dots x_n \, G(x_1, \dots, x_n)$, où $G(x_1, \dots, x_n)$ est une formule sans quantificateur. Supposons que cette formule F soit satisfaite dans $\mathscr{M}$. Alors la formule $G(x_1, \dots, x_n)$ est satisfaite dans tous les environnements, donc en particulier dans ceux qui affectent aux variables les interprétations des termes clos du langage, et donc $\mathscr{M} \models G(t_1, \dots, t_n)$ pour tous les n-uplets $(t_1, \dots, t_n)$ de termes clos. Il en est donc de même pour $\mathscr{H}_{\mathscr{M}}$, d'où $\mathscr{H}_{\mathscr{M}} \models F$. ■

Vu la discussion précédente, Le lemme permet donc de ramener la satisfaisabilité d'une formule universelle à celle d'un ensemble de formules propositionnelles en général infini dénombrable. Le lemme s'étend par ailleurs sans difficultés à un ensemble quelconque (éventuellement infini) de formules universelles.

Appelons *particularisation* d'une formule universelle $\forall x_1 \dots \forall x_n \, G$, où G est sans quantificateurs, une formule obtenue en remplaçant toutes les variables libres de G par des termes *clos*.

Proposition 2.6.2 *Une formule universelle close du langage de signature $\mathscr{S}$ est satisfaisable si et seulement si toutes ses particularisations sont satisfaisables, vues comme formules du calcul propositionnel sur l'ensemble $\mathscr{B}_{\mathscr{S}}$ des formules closes atomiques du langage.*

De même un ensemble de formules universelles closes du même langage est satisfaisable si et seulement si l'ensemble de toutes les particularisations de toutes les formules est satisfaisable.

Démonstration. Conséquence de la discussion et du lemme qui précèdent. ■

Ce résultat ne donne pas de moyen algorithmique pour déterminer qu'une formule universelle est satisfaisable, puisque l'on s'est ramené à un ensemble en général infini de formules du calcul propositionnel : on a une procédure de décision pour chacune d'entre elle mais pas pour une infinité d'entre elles, puisqu'il faudrait une infinité de vérifications.

Par contre on peut montrer en utilisant la mise sous forme normale conjonctive et la règle de coupure, que si un ensemble infini de formules du calcul propositionnel est non satisfaisable, alors il est réfutable par coupures, ce qui signifie qu'un sous-ensemble fini de cet ensemble infini l'était (celles qui ont pour conséquence les clauses dont on a besoin pour la réfutation par coupures). C'est la *compacité* du calcul propositionnel, dont on déduit le théorème de Herbrand.

Théorème 2.6.3 (Théorème de Herbrand) *Le langage est supposé de signature finie ou dénombrable.*

Si une formule universelle close est non satisfaisable, alors il existe un sous-ensemble fini de l'ensemble des particularisations de cette formule qui est non satisfaisable (en calcul propositionnel).

Plus généralement si un ensemble fini ou dénombrable de formules universelles closes est non satisfaisable, alors il existe un sous-ensemble fini de l'ensemble des particularisations de ces formules qui est non satisfaisable (en calcul propositionnel).

On aurait pu donner ces deux énoncés sous forme d'équivalence, la réciproque étant évidente.

Démonstration. L'ensemble des particularisations d'une formule universelle close est fini ou dénombrable, car l'ensemble des termes clos sur une signature finie ou dénombrable est fini ou dénombrable (infini dès que la signature possède un symbole de fonction d'arité $\geqslant 1$).

L'ensemble des particularisations d'un ensemble fini ou dénombrable de formules universelles closes est dénombrable comme réunion finie ou dénombrable d'ensembles dénombrables.

Le théorème de Herbrand est alors conséquence de la proposition 2.6.2 précédente et du théorème 2.6.4 de compacité du calcul propositionnel qui suit. ■

2.6.3 Compacité du calcul propositionnel

Il s'agit de démontrer dans ce paragraphe les résultats de calcul propositionnel qui manquent pour démontrer le théorème de Herbrand, en particulier :

Théorème 2.6.4 (compacité du calcul propositionnel)
Si un ensemble $\mathscr{F}$ (dénombrable) de formules du calcul propositionnel n'est pas satisfaisable, alors il existe un sous-ensemble fini de $\mathscr{F}$ qui n'est pas satisfaisable,

ou par contraposée
si tous les sous-ensembles finis d'un ensemble $\mathscr{F}$ (dénombrable) de formules du calcul propositionnel sont satisfaisables, alors $\mathscr{F}$ est satisfaisable.

On aurait pu énoncer le théorème comme une équivalence, car la réciproque est évidente par définition de la satisfaisabilité d'un ensemble de formules.

Le théorème est trivial si l'ensemble de formules est fini. Il est vrai pour un ensemble infini quelconque de formules propositionnelles (construites sur un ensemble infini pas forcément dénombrable de formules propositionnelles). La démonstration peut se faire par le lemme de Zorn, elle requiert de toute façon une forme forte de l'axiome du choix. On se contentera du cas où l'ensemble de formules est dénombrable, car construit sur un ensemble dénombrable de variables propositionnelles $\{p_i \mid i \in \mathbb{N}\}$.

Chaque formule propositionnelle est équivalente, par mise sous forme normale conjonctive, à un ensemble fini de *clauses* (disjonctions d'atomes, variables propositionnelle ou négation d'une telle variable). On peut donc déduire le théorème de compacité du cas particulier où les formules sont des clauses.

Un ensemble de clauses, éventuellement infini est réfutable par coupure (ou résolution) [7] si la clause vide est dérivable à partir de ces clauses. Par définition, une dérivation, n'utilise qu'un sous-ensemble fini de ces clauses. Une conséquence de la validité de la règle de coupure et que si un ensemble de clauses est réfutable, alors il n'est pas satisfaisable, et donc par contraposée, si un ensemble de clauses est satisfaisable alors il n'est pas réfutable. On va déduire la compacité du résultat réciproque, qui est la complétude.

Théorème 2.6.5 (complétude pour la résolution en calcul propositionnel) *Soit $\{C_k \mid k \in \mathbb{N}\}$ un ensemble infini dénombrable de clauses qui n'est pas réfutable par coupure, alors il est satisfaisable.*

7. La règle de résolution du calcul propositionnel est appelée également règle de coupure en théorie de la démonstration, dans un contexte plus général. Pour distinguer la règle de résolution du calcul des prédicats, qui va être introduite ensuite, de la règle de résolution du calcul propositionnel, on appelle cette dernière règle de coupure dans la suite de ce chapitre.

Démonstration. On rappelle que si la clause vide est dérivée en utilisant la règle d'affaiblissement, qui consiste à ajouter des atomes à une clause, et la règle de coupure, elle est dérivable par coupure, puisqu'en supprimant les utilisations de l'affaiblissement, on ne peut qu'arriver plus rapidement à la clause vide.

On suppose donc donné un ensemble $\{C_k \ / \ k \in \mathbb{N}\}$ de clauses qui n'est pas réfutable par coupures, et on va en déduire une valuation qui satisfait toutes ces clauses. Une clause est considérée de façon ensembliste : il n'y a jamais répétition du même atome. On la note en séparant les atomes par des virgules (ce sont des disjonctions d'atomes).

La règle de coupure est notée :

$$\frac{\Gamma, A \qquad \Delta, \neg A}{\Gamma, \Delta}$$

où Γ, Δ désigne l'union ensembliste des ensembles d'atomes Γ et Δ.

On construit par récurrence sur n une suite d'arbres de réfutation $\mathscr{A}_n$ (réfutation d'un ensemble de clauses qui n'est pas déterminé a priori) « par le bas » :

— $\mathscr{A}_0$ est réduit à la clause vide (c'est l'arbre de réfutation de la clause vide !) ;

— $\mathscr{A}_{k+1}$ est l'arbre obtenu à partir de $\mathscr{A}_k$ en prolongeant $\mathscr{A}_k$ en toutes les feuilles qui ne sont pas des clauses subsumées par une clause de $\mathscr{C}$ par coupure sur la variable p_{k+1}.

Par exemple, si $\mathscr{C}$ ne contient pas la clause vide, l'arbre $\mathscr{A}_1$ est :

$$\frac{\neg p_1 \quad p_1}{\square}$$

et si $\mathscr{C}$ ne contient de plus ni la clause $\neg p_1$ ni la clause p_1, l'arbre $\mathscr{A}_2$ est :

$$\frac{\dfrac{\neg p_1, \neg p_2 \quad \neg p_1, p_2}{\neg p_1} \qquad \dfrac{p_1, \neg p_2 \quad p_1, p_2}{p_1}}{\square}$$

si $\mathscr{C}$ contient la clause $\neg p_1$ mais pas la clause p_1 ni la clause vide, $\mathscr{A}_2$ est :

$$\frac{\neg p_1 \qquad \dfrac{p_1, \neg p_2 \quad p_1, p_2}{p_1}}{\square}$$

L'arbre $\mathscr{A}_{n+1}$ prolonge l'arbre $\mathscr{A}_n$ et donc la suite d'arbres $(\mathscr{A}_n)$ peut être vue comme un seul arbre, potentiellement de taille infinie dont les $\mathscr{A}_n$ sont les sous-arbres partant de la racine et tronqués à la hauteur n.

Un arbre $\mathscr{A}_n$ possède au moins une clause feuille qui n'est subsumée par aucune des clauses C_i. En effet sinon, cela signifierait que chaque clause feuille de $\mathscr{A}_n$ s'obtient par affaiblissement à partir d'une clause C_i. On a donc un arbre de réfutation par affaiblissement et coupure, donc un arbre de réfutation par coupure de $\{C_i \ / \ i \in \mathbb{N}\}$ (d'un sous-ensemble fini de cet ensemble), ce qui contredit l'hypothèse.

Ceci signifie donc que chaque arbre $\mathscr{A}_n$ est de hauteur n, et que l'arbre $\mathscr{A}_\omega$, qui est formé par la suite $\mathscr{A}_n$, est de hauteur infinie.

On définit maintenant par récurrence une suite d'atomes $(a_n)_{n\geqslant 1}$, avec $a_n := p_n$ ou $a_n := \neg p_n$, qui correspond à une branche infinie de cet arbre, et vérifie que $a_1, \ldots, a_n$ est racine d'un sous-arbre de $\mathscr{A}_\omega$ de hauteur infinie (en particulier $a_1 \vee \cdots \vee a_n$ n'est subsumée par aucune clause de $\mathscr{S}$) :

— au moins l'une des deux clauses p_1 et $\neg p_1$ est racine d'un sous-arbre de racine cette clause et de hauteur infinie, sinon $\mathscr{S}$ est réfutable. On en choisit une, et on prend pour a_1 l'atome correspondant (p_1 ou $\neg p_1$) ;

— par construction, $a_1, \ldots, a_n$ est racine d'un sous-arbre de $\mathscr{A}_\omega$ de hauteur infinie. Au moins l'une des deux clauses $a_1, \ldots, a_n, p_{n+1}$ et $a_1, \ldots, a_n, \neg p_{n+1}$ est racine d'un sous-arbre de $\mathscr{A}_\omega$ de hauteur infinie. on en choisit une et soit a_{n+1} l'atome correspondant (p_{n+1} ou $\neg p_{n+1}$).

La suite (a_n) permet de définir une valuation ν de la façon suivante :

— si $a_n = p_n$, alors $\nu(p_n) = 0$;

— si $a_n = \neg p_n$, alors $\nu(p_n) = 1$.

Cette valuation ν satisfait toutes les clauses C_i. En effet, supposons, pour arriver à une contradiction, que $\nu(C_i) = 0$, comme C_i est une disjonction d'atomes, cela signifierait que chaque atome α de C_i vérifie $\nu(\alpha) = 0$. Soit p_k telle que $\alpha = p_k$ ou $\alpha = \neg p_k$, alors par définition de ν, $\alpha = a_k$. La clause $a_1, \ldots, a_{n_i}$ serait alors subsumée par la clause C_i, où n_i est l'indice le plus élevé d'une variable propositionnelle dans C_i. Ceci contredirait la définition de la suite (a_n).

On a donc bien montré que $\{C_i \ / \ i \in \mathbb{N}\}$, sous l'hypothèse donnée, est satisfaisable. ∎

Pour définir la suite (a_n), il y a potentiellement une infinité de choix, un à chaque étape, car il se peut très bien que les deux atomes p_n et $\neg p_n$ conviennent tous les deux à l'étape n. On peut décider arbitrairement que, dans ce cas, c'est toujours l'atome positif qui est choisi : pas besoin d'invoquer l'axiome du choix. Cependant il n'y a pas en général de moyen effectif de déterminer si un sous-arbre va être infini, c'est-à-dire que la démonstration est non constructive.

Démonstration (compacité). Passons maintenant à la démonstration du théorème de compacité propositionnel. On prend un ensemble $\mathscr{F}$ dénombrable de formules propositionnelles qui n'est pas satisfaisable. L'ensemble dénombrable des clauses obtenues en prenant pour chacune d'entre elle un ensemble de clauses équivalent n'est pas satisfaisable non plus. D'après le théorème de complétude (contraposée), il est réfutable par coupure, ce qui signifie par définition d'une réfutation qu'un sous-ensemble fini de cet ensemble de clauses est réfutable par coupure. Ce sous-ensemble fini n'est pas satisfaisable (validité de la règle de coupure). Étant fini, il est issu d'un nombre fini de formules de $\mathscr{F}$, et donc conséquence d'un sous-ensemble fini de $\mathscr{F}$, ce sous-ensemble fini n'est donc pas non plus satisfaisable. ■

On a ainsi terminé la démonstration du théorème de Herbrand. On va voir que de plus, la résolution propositionnelle utilisée pour démontrer la compacité, peut être généralisée au calcul des prédicats (formules closes universelles) pour un algorithme de semi-décision.

2.6.4 Réduction à la résolution propositionnelle

On appelle *clause* du calcul des prédicats une disjonction de formules atomiques et de négations de formules atomiques. Du fait que toute forme normale propositionnelle peut être mise sous forme normale conjonctive, et que le quantificateur universel commute avec la conjonction on déduit immédiatement la proposition suivante.

Proposition 2.6.6 *Toute formule universelle du calcul des prédicat est équivalent à une conjonction de clôtures universelles de clauses (des formules universelles dont la partie propositionnelle est une disjonction de formules atomiques et négation de formules atomiques).*

Pour simplifier on suppose la signature $\mathscr{S}$ finie. On déduit du théorème de Herbrand une méthode de semi-décision pour la réfutation d'un ensemble fini $\mathscr{C}$ de formules du calcul des prédicats qui sont des clôtures universelles de clauses. Soit $(C_n)_{n\in\mathbb{N}}$ une énumération de toutes les particularisations des formules de $\mathscr{C}$ (on suppose cet ensemble infini, la méthode décrite ensuite s'adapte au cas fini). On a besoin pour que l'argument qui suit soit correct de pouvoir produire effectivement une énumération de ces particularisations, sachant que le domaine de Herbrand est infini mais énumérable (la signature étant supposée finie). Les particularisations sont des clauses propositionnelles sur $\mathscr{B}_{\mathscr{S}}$, c'est-à-dire des disjonctions de formules atomiques closes et de négations de formules atomiques clauses du langage de signature $\mathscr{S}$.

Comme d'après le théorème de Herbrand, un sous-ensemble fini de $\{C_n \ / \ n \in \mathbb{N}\}$ est contradictoire, cela signifie qu'il existe un entier N tel que $\{C_1, \ldots, C_N\}$ est contradictoire, donc réfutable par coupures. Il suffit de prendre pour N le plus grand indice des clauses de l'ensemble fini en question.

Pour un ensemble fini de clauses propositionnelles, on peut par la méthode de résolution en calcul des propositions, déterminer si elle est ou non réfutable. On suppose donc cette procédure connue et s'appliquant à une liste de clauses.

On va donner maintenant un algorithme de semi-décision (qui peut donc ne pas terminer). qui prend en entrée une énumération $(C_n)_{n\in\mathbb{N}}$ des particularisations de $\mathscr{C}$, plus précisément une fonction fonction C qui prend un entier n en entrée et produit en sortie la clause propositionnelle C_n.

Entrée
Une fonction C qui énumère les particularisations des clauses de $\mathscr{C}$.
Variables
L est une liste de clauses, n est un entier
L := []
n := 0
Tant que L n'est pas réfutable
    ajouter C(n) à L
Fin Tant que
retourner $\mathscr{C}$ est réfutable

Si $\mathscr{C}$ est réfutable l'algorithme termine au bout de N étapes pour un certain entier N (cf. ci-dessus). Sinon l'algorithme ne termine pas.

Comme $\mathscr{C}$ est fini, si jamais l'ensemble des particularisations est fini, c'est que la signature n'a que des symboles de constantes. On obtient alors une procédure de décision en prenant toutes les particularisations (en nombre fini).

Dans les autres cas on n'obtient qu'une méthode de semi-décision pour la réfutation d'un ensemble fini de clôtures universelles de clauses, donc pour un ensemble fini de formules universelles, et finalement pour un ensemble fini de formules quelconques par mise sous forme de Skolem.

Cependant cet algorithme particulièrement inefficace n'a qu'un intérêt purement théorique. On peut l'améliorer notablement, à partir d'idées qui étaient déjà présentes dans la thèse de Herbrand, mais qui ont été reprises ou redécouvertes et développées par John Alan Robinson dans les années 1960.

— La première idée est de choisir uniquement les particularisations qui donneront des formules propositionnelles sur laquelle il est possible d'appliquer la règle de coupure, celle qui est utilisée pour la réfutation. Ce sont par exemple nécessairement des formules atomiques formées sur le même symbole de prédicat et apparaissant positivement et négativement dans deux formules différentes de $\mathscr{C}$.

— La seconde idée est de ne pas instancier tout de suite par des termes clos, mais par des termes avec variables libres que l'on peut voir comme regroupant plusieurs termes clos. Cela permet donc de regrouper plusieurs déductions.

C'est la *méthode de résolution du calcul des prédicats*, qui s'appuie sur l'*algorithme d'unification*, un algorithme qui détermine étant donné deux termes t et t' du langage une substitution σ des variables libres de t et t' telle que $\sigma(t) = \sigma(t')$.

2.6.5 Méthode de résolution en calcul des prédicats : une première approche

On peut tout d'abord remarquer que, au vu de la méthode décrite précédemment, on peut utiliser deux règles pour réfuter un ensemble de clôtures universelles de clauses du calcul des prédicats

— La règle de coupure déjà vue en calcul propositionnel

$$\frac{C \vee At_1 \dots t_n \qquad D \vee \neg At_1 \dots t_n}{C \vee D} \; At_1 \dots t_n \text{ formule atomique close}$$

— une règle d'instanciation du quantificateur universel

$$\frac{\forall x\, C}{C[t/x]} \; C \text{ clause universellement quantifiée, } t \text{ est un terme clos}$$

De la même façon qu'en calcul propositionnel, les clauses sont vues comme des ensembles de formules. Par exemple, la dernière règle, comprend comme cas particulier où $C[t/x]$ est déjà présent dans Γ :

$$\frac{\forall x\,(D \vee C)}{D} \; t \text{ est un terme clos et , } C[t/x] \text{ apparaît dans la clause } D$$

Il est très facile de vérifier que ces règles sont valides, c'est-à-dire que pour toute $\mathscr{S}$-structure $\mathscr{M}$, si $\mathscr{M}$ est un modèle de la prémisse ou des prémisses de la règle (notées au dessus de la barre), alors $\mathscr{M}$ est modèle de la formule conclusion (notée sous la barre). On en déduit que si la clause vide se déduit par ces deux règles d'un ensemble de formules, alors cet ensemble de formules n'est pas satisfaisable.

Au paragraphe précédent on a montré la réciproque, à savoir que si un ensemble de clôtures universelles de clauses du calcul des prédicats n'est pas satisfaisable, alors on peut déduire de ces formules la clause vide $\square$ par coupure et instanciation, soit la complétude de la méthode de réfutation utilisant uniquement ces deux règles.

Proposition 2.6.7 (complétude de la réfutation par coupure et instantiation) *Si un ensemble fini ou dénombrable de formules qui sont des clôtures universelles de clauses du calcul des prédicats est non satisfaisable, alors il est réfutable en utilisant la règle de coupure et la règle d'instanciation du quantificateur universelle par un terme clos.*

Démonstration. C'est une conséquence du théorème de Herbrand et du théorème de complétude de la réfutation par coupure en calcul propositionnel (2.6.5 page 49). ■

La règle de coupure reste valide pour des formules atomiques quelconques (non forcément closes), et la règle d'instanciation reste valide si on prend un terme t quelconque. Il faut simplement veiller dans ce dernier cas à ce que les variables libres du terme t ne soient pas « capturées » par des quantificateurs de la formule C.

Autoriser l'instanciation par des termes avec variables libres n'a d'intérêt que si on peut plus tard instancier ces variables libres. Pour pouvoir à nouveau applique la règle d'instanciation on a donc besoin d'abord de la règle suivante, dite *règle de généralisation*.

$$\frac{C}{\forall x\, C}$$

La règle de généralisation est évidemment valide, par définition de la satisfaction dans le cas du quantificateur universel.

Le théorème de complétude est a fortiori valide pour ces trois règles, coupure sur des formules atomiques quelconques, particularisation par un terme quelconque (sans capture) et généralisation.

La méthode de résolution va d'une certaine façon regrouper ces trois règles en une seule. On va représenter maintenant la clôture universelle d'une clause par la clause elle même, c'est-à-dire que les quantificateurs universels sont implicites. Deux clauses doivent avoir des ensembles de variables libres disjoints : si deux variables apparaissent dans deux clauses distinctes, elles correspondent à deux quantificateurs universels distincts.

On n'utilise plus qu'une seule règle, la règle de résolution, dont la règle de coupure est un cas particulier, qui est la suivante :

$$\frac{C \vee A_1 \vee \cdots \vee A_k \qquad D \vee \neg B_1 \vee \ldots \neg B_l}{\sigma(C) \vee \sigma'(D)}\ \sigma \text{ et } \sigma' \text{ telles que } \sigma(A_1) = \cdots = \sigma(A_k) = \sigma'(B_1) = \cdots = \sigma'(A'_l)$$

Une substitution correspond à plusieurs instanciations successives. Celles-ci ont pu identifier des formules d'où la nécessité de prévoir d'éliminer plusieurs formules identifiées par la substitution : une fois celles-ci identifiées de chaque côté, la règle utilisée est bien la règle de coupure sur une seule formule vue en calcul propositionnel.

Les atomes identifiés par la substitution ont forcément même symbole de prédicat. Par exemple, dans le cas particulier où un seul atome est éliminé de chaque côté, la règle peut se détailler (P symbole de prédicat) :

$$\frac{C \vee P t_1 \ldots t_n \qquad D \vee \neg P t'_1 \ldots t'_n}{\sigma(C) \vee \sigma'(D)}\ \sigma \text{ et } \sigma' \text{ telles que } \sigma(t_1) = \sigma'(t'_1), \ldots, \sigma(t_n) = \sigma'(t'_n)$$

La substitution σ agit sur les variables libres de la première prémisse, la substitution σ' sur celles de la seconde prémisse. Ces ensembles étant disjoints, on peut supposer que les deux substitutions agissent chacune sur ces ensembles disjoints (en dehors de ces ensembles leur valeur n'a pas d'importance).

On peut donc regrouper les deux substitutions en une seule définie sur l'ensemble des variables libres des deux prémisses. La règle de résolution s'écrit alors plus simplement

$$\frac{C \vee A_1 \vee \cdots \vee A_k \qquad D \vee \neg B_1 \vee \cdots \vee \neg B_l}{\sigma(C) \vee \sigma(D)}\ \sigma \text{ telle que } \sigma(A_1) = \cdots = \sigma(A_k) = \sigma(B_1) = \cdots = \sigma(B_l)$$

Rappelons que chaque clause est à interpréter comme sa clôture universelle. Cette règle se décompose bien alors en une suite d'instanciations des variables libres de chacune des prémisses jusqu'à obtenir la substitution σ, la règle de coupure, et la généralisation de toutes les variables libres de la conclusion. cette règle est donc valide comme composée des trois précédentes.

Étant donné un ensemble de couples de termes $\{(t_1, t'_1), \ldots, (t_n, t'_n)\}$, une substitution définie sur les variables libres de ces termes telle que

$$\sigma(t_1) = \sigma(t'_1), \ldots, \sigma(t_n) = \sigma(t'_n)$$

est appelé *unificateur* de l'ensemble $\{(t_1, t'_1), \ldots, (t_n, t'_n)\}$.

Un unificateur σ des deux formules atomiques $P t_1 \ldots t_n$ et $P t'_1 \ldots t'_n$ (forcément même symbole de prédicat) et un unificateur de $\{(t_1, t'_1), \ldots, (t_n, t'_n)\}$.

Plus généralement on étend la définition d'unificateur à un ensemble fini de formules atomiques, en se ramenant toujours à un ensemble de couples de termes.

Proposition 2.6.8 (complétude de la résolution, unificateur quelconque) *Si un ensemble fini ou dénombrable de clôtures universelles de clauses du calcul des prédicats est non satisfaisable, alors il est réfutable en utilisant la seule règle de résolution sous la forme ci-dessus.*

On a bien entendu également la réciproque de cet énoncé de complétude, la règle de résolution étant valide.

Démonstration. C'est une conséquence de la proposition 2.6.7 de complétude, en utilisant un unificateur qui substitue des termes clos. ■

Pour mettre en pratique la méthode de résolution, on va justement autoriser d'autres substitutions que celles par des termes clos.

Exemple. On prend pour signature $(s, \leqslant)$. On veut montrer que

$$\forall x\, x \leqslant s\, x,\ \forall x \forall y \forall z((x \leqslant y \wedge y \leqslant z) \rightarrow x \leqslant z) \vdash \forall x\, x \leqslant s\, s\, x$$

cela revient à montrer que l'ensemble de formules suivant est contradictoire :

$$\forall x\, x \leqslant s\, x,\ \forall x \forall y \forall z(\neg x \leqslant y \vee \neg y \leqslant z \vee x \leqslant z), \exists x \neg x \leqslant s\, s\, x$$

seule la dernière formule a besoin d'être skolemisée, soit a une constante de Skolem, on est donc ramené à montrer que les clôtures universelles des trois clauses suivantes sont contradictoires

$$(C_1)\ x \leqslant s\, x,\ (C2)\ \neg x' \leqslant y \vee \neg y \leqslant z \vee x' \leqslant z,\ (C_3)\ \neg a \leqslant s\, s\, a\,.$$

Si on veut commencer par la clause négative (C_3), la seule règle de résolution possible est sur (C_2), il n'est pas possible d'appliquer la règle de résolution à (C_1) et (C_3), car une substitution σ ne peut vérifier à la fois $\sigma(x) = a$ et $\sigma(s\, x) = s\, s\, a$.

1.	$x \leqslant s\, x$	
2.	$\neg x' \leqslant y \vee \neg y \leqslant z \vee x' \leqslant z$	
3.	$\neg a \leqslant s\, s\, a$	
4.	$\neg a \leqslant y \vee \neg y \leqslant s\, s\, a$	résol. 3 et 2 pour α définie par $\alpha(x') = a,\ \alpha(z) = s\, s\, a$
5.	$\neg s\, a \leqslant s\, s\, a$	résol. 4 et 1 pour β définie par $\beta(x) = a,\ \beta(y) = s\, a$
6.	$\square$	résol. 5 et 1 pour $\gamma(x) = s\, a$.

Il se trouve que cette réfutation n'utilise que des substitutions par des termes clos qui sont à chaque fois le seul unificateur possible. Ce n'est pas le cas pour la réfutation qui suit.

1.	$x \leqslant s\, x$	
2.	$\neg x' \leqslant y \vee \neg y \leqslant z \vee x' \leqslant z$	
3.	$\neg a \leqslant s\, s\, a$	
4.	$\neg s\, x'' \leqslant z \vee x'' \leqslant z$	résol. 1 et 2 pour α définie par $\alpha(x) = x'',\ \alpha(x') = x'',\ \alpha(y) = s\, x'$
5.	$\neg s\, x''' \leqslant s\, s\, x'''$	résol. 4 et 1 pour β définie par $\beta(x) = x''', \beta(x'') = x''',\ \beta(z) = s\, s\, x'''$
6.	$\square$	résol. 5 et 3 pour $\gamma(x''') = a$.

Procéder de cette façon pourrait avoir un intérêt pour réfuter l'ensemble de clauses obtenu en remplaçant la clause (C_3) par $\neg a \leqslant s\, s\, a \vee \neg b \leqslant s\, s\, s\, b$.

Exemple (paradoxe du barbier). Cet exemple suivant met en évidence la nécessité d'identifier les formules après substitution dans la règle de coupure. Le langage a pour signature $\{R\}$ constituée d'un seul symbole de prédicat binaire (notation préfixe). Il s'agit de démontrer :

$$\neg \exists x \forall y \big(R\, x\, y \leftrightarrow \neg R\, y\, y\big)$$

(il n'existe pas de barbier qui rase tout ceux qui ne se rasent pas eux-mêmes).

Il s'agit donc de réfuter la formule $\exists x \forall y \big(R\, x\, y \leftrightarrow \neg R\, y\, y\big)$, qui donne par skolemisation, avec a un symbole de constante pour la variable quantifiée x, les deux clauses :

$$\neg R\, a\, y \vee \neg R\, y\, y \ \text{ et } \ R\, y'\, y' \vee R\, a\, y'\,.$$

un unificateur σ des couples de termes $\{(a, y'), (y, y')\}$ (identification des premiers atomes de chaque formules) est défini par $\sigma(y') = a$, $\sigma(y) = a$ (l'identité ailleurs). Il identifie les deux atomes de chacune des deux clauses, d'où par coupure la clause vide.

1.	$\neg R\, a\, y \vee \neg R\, y\, y$	
2.	$R\, y'\, y' \vee R\, a\, y'$	
3.	$\square$	résol. 1 et 2 pour σ définie par $\sigma(y') = a$, $\sigma(y) = a$.

2.6.6 Unification

L'étude de l'unification va permettre d'encore simplifier la méthode de résolution. Rappelons qu'un unificateur d'un ensemble fini de couples de termes $\mathscr{E} = \{(t_1, t'_1), \ldots, (t_n, t'_n)\}$ d'une certaine signature, que l'on va appeler *système à unifier*, est une substitution α de termes aux variables libres vérifiant

$$\alpha(t_1) = t'_1, \ldots, \alpha(t_n) = t'_n \,.$$

Si une telle substitution existe, le système $\mathscr{E}$ est dit *unifiable*

On va montrer que si $\mathscr{E}$ est unifiable, alors il existe une substitution σ de $\mathscr{E}$ telle que

— σ est un unificateur de $\mathscr{E}$;

— α est un unificateur de $\mathscr{E}$ si et seulement s'il existe une substitution β telle que $\alpha = \beta \circ \sigma$.

Une telle substitution σ est appelée *unificateur principal* ou *unificateur le plus général* de $\mathscr{E}$, en anglais *mgu* pour *most general unifier*. Dans l'exemple ci-dessus, tous les unificateurs utilisés ont été choisis principaux (à chaque fois pour un ensemble de deux couples de termes). Un unificateur principal de $\mathscr{E}$ décrit tous les unificateurs possibles de $\mathscr{E}$.

Une notion utile est celle de *support d'une substitution* σ : c'est l'ensemble des variables x telles que $s(x) \neq x$. Comme les formules et les réfutations sont des objets finis, on peut se retreindre à des substitutions ne substituent effectivement qu'un nombre fini de variables, c'est-à-dire à des substitutions à support fini. En particulier les unificateurs principaux d'un ensemble fini de couples de termes (seuls cas envisagés), seront à support fini.

Il n'y a pas unicité de l'unificateur principal, mais presque.

Proposition 2.6.9 *Deux unificateurs principaux σ et σ' d'un même système à unifier se déduisent l'un de l'autre par renommage de variable, c'est à dire par composition par une substitution de variables à des variables.*

Démonstration. Comme σ et σ' sont des unificateurs principaux, on a des substitutions α et β, dont on peut supposer que les supports sont restreint aux variables apparaissant dans les images par σ pour α, dans les images par β pour σ', vérifiant :

$$\sigma' = \alpha \circ \sigma \;\text{ et }\; \sigma = \alpha' \circ \sigma'$$

On obtient $\alpha \circ \alpha' \circ \sigma' = \sigma'$, $\alpha' \circ \alpha \circ \sigma = \sigma$, donc les composées $\alpha \circ \alpha'$ et $\alpha' \circ \alpha$ sont l'identité, notée id, sur sur les variables apparaissant dans les images par σ et σ', donc finalement $\alpha \circ \alpha' = \alpha' \circ \alpha = \text{id}$. Ceci n'est possible que si α et α' substituent des variables à des variables. ■

Exemple. Le système $\{(x,z),\ (y,z)\}$ a pour unificateur principal σ défini par $\sigma(x) = z, \sigma(y) = z$, mais aussi σ' défini par $\sigma'(z) = x, \sigma'(y) = x$ qui se déduit de σ par composition avec le renommage de variables α défini par $\alpha(z) = x$.

On va donner un algorithme pour calculer un unificateur dont on montrera qu'il est principal. L'algorithme consiste à calculer à partir d'un *système à unifier*, un système équivalent qui fournit de façon évidente un unificateur principal, que l'on va appeler *système résolu*. La terminaison et la correction de l'algorithme montrent l'existence d'un unificateur principal.

Système résolu

Un système à unifier sera dit résolu (sous-entendu pour l'algorithme de Robinson) s'il a la forme suivante :

$$\{(x_1, t_1), \ldots, (x_n, t_n)\} \;\text{ où }\; i \neq j \Rightarrow x_i \neq x_j \;\text{ et }\; \text{pour tout } 1 \leqslant i, j \leqslant n,\ x_i \text{ n'apparaît pas dans } t_j$$

Le lemme suivant justifie l'appellation « résolu ». Il suit directement par les définitions.

Lemme 2.6.10 *Un système résolu $\{(x_1, t_1), \ldots, (x_n, t_n)\}$ est unifiable et possède un unificateur principal σ défini par $\sigma(x_1) = t_1, \ \ldots\ , \sigma(x_n) = t_n$.*

Démonstration. Vu les conditions, pour tout $1 \leqslant i \leqslant n$ on a $\sigma(t_i) = t_i$, et σ est un unificateur du système résolu.

Si s est un unificateur d'un tel système, pour tout $1 \leqslant i \leqslant n$ on a $s(x_i) = s(t_i)$. Soit α la substitution identique à s sauf en les variables $x_1, \ldots, x_n$ où elle vaut l'identité :

$$\text{pour tout } 1 \leqslant i \leqslant n,\ \alpha(x_i) = x_i \,;\ \text{ pour toute variable } y \notin \{x_1, \ldots, x_n\},\ \alpha(y) = s(y) \,.$$

Alors par hypothèse sur les t_i, $\alpha(t_i) = s(t_i)$, donc $s = \alpha \circ \sigma$. ■

Systèmes à unifier équivalents

Deux systèmes à unifier seront dit *équivalents* quand ils possède le même ensemble d'unificateurs.

On va donner un un algorithme qui va tester si un système est unifiable, et qui, quand le système est unifiable, va donner un système résolu équivalent.

Il y a essentiellement deux écueils à l'unification. On les décrit ci-dessous sur des exemples pour la signature $\{a, s, p, f\}$, a constante, s unaire, p unaire, f binaire :

— deux termes dont les constructeurs principaux sont différents (*Clash*), par exemple aucun des systèmes qui suivent ne sont unifiables :

$$\{(a, sx)\},\ \{(sx, py)\}\ ,\{(spx, fysz)\} \qquad \text{échec par } \textit{Clash}.$$

— une variable devrait être substituée par un terme où elle apparaît, par exemple aucun des systèmes qui suivent ne sont unifiables :

$$\{(x, sx)\},\ \{(x, fxa)\},\ \{(x, fafxsy)\} \qquad \text{échec au } \textit{test d'occurrence} \text{ (}\textit{occur check}\text{)}.$$

L'idée de l'algorithme est d'engendrer des couples de termes à unifier par « superposition » (penser aux arbres syntaxiques des termes), puis d'appliquer une substitution élémentaire donnée par cette superposition (un terme à une variable) tant que le système n'est pas résolu. Il est décrit par les règles de réécriture suivantes (il s'agit de réécrire suivant les règles de la figure 2.1 tant que l'une d'entre elles peut s'appliquer, l'ordre est indifférent, mais contraint par les conditions d'application de chaque règle).

Superposition
$\{(fu_1 \dots u_n, fv_1 \dots v_n)\} \cup \mathscr{E} \longrightarrow \{(u_1, v_1),\ \dots,\ (u_n, v_n)\} \cup \mathscr{E}$
Clash (échec de la superposition)
$\{(fu_1 \dots u_n, gv_1 \dots v_k)\} \cup \mathscr{E} \longrightarrow$ Echec — si $f \neq g$, f et g peuvent être 0-aires (constantes)
Identité
$\{(x, x)\} \cup \mathscr{E} \longrightarrow \mathscr{E}$ — x variable
Variable à gauche
$\{(t, x)\} \cup \mathscr{E} \longrightarrow \{(x, t)\} \cup \mathscr{E}$ — si x variable et t n'est pas une variable
test d'occurrence
$\{(x, t)\} \cup \mathscr{E} \longrightarrow$ Echec — si la variable x apparaît dans t, $t \neq x$
Substitution
$\{(x, t),\ (u_1, v_1), \dots, (u_p, v_p)\} \longrightarrow \{(x, t),\ (u_1[t/x], v_1[t/x]), \dots, (u_p[t/x], v_p[t/x])\}$
si la variable x n'apparaît pas dans t et apparaît dans au moins un des u_i ou v_i.

FIGURE 2.1 – Algorithme d'unification de Robinson

Proposition 2.6.11

i. *Chacune des règles de l'algorithme d'unification de Robinson (figure 2.1) transforme un système à unifier en un système équivalent, et ne renvoie « Echec » que s'il n'est pas unifiable ;*

ii. *aucune règle ne s'applique si et seulement si le système est résolu ;*

iii. *toute suite de règles est finie (l'algorithme termine).*

Démonstration.

i. La seule règle pour laquelle le résultat n'est pas évident où conséquence de remarques déjà faites (cas d'échec) est la substitution : un unificateur σ de chacun des deux systèmes vérifie forcément $\sigma(x) = \sigma(t)$, donc, si x n'apparaît pas dans t, pour tout terme u, $\sigma(u) = \sigma(u[t/x])$. Ceci assure que les système de départ et d'arrivée ont même ensemble d'unificateurs.

ii. Si le système est résolu aucune règle ne peut s'appliquer. Réciproquement si aucune règle ne s'applique, c'est que tous les membres gauches sont des variables (par règles de superposition, clash et variable à gauche) et que ces variables sont distinctes et n'apparaissent pas à droite (par règles de substitution, identité et test d'occurrence).

iii. On remarque que :

— toutes les règles sauf la substitution font décroître le nombre de signes du système à unifier dans les membres gauches des couples ;
— la substitution $[t/x]$ n'est effectuée que si x a plus d'une occurrence dans le système à unifier et après cette règle la variable x n'a plus comme occurrence dans le système à unifier que celle dans le couple (x,t) ;
— la substitution peut faire augmenter le nombre de signes dans les membres gauches.

Pour les besoins de la démonstration, on appelle variable substituable dans un système $\mathscr{E}$, une variable x qui n'apparaît pas dans $\mathscr{E}$ sous la forme obtenue après substitution, c'est-à-dire que si x apparaît dans un couple de $\mathscr{E}$ de la forme (x,t) avec x n'apparaissant pas dans t, alors x apparaît également dans un des autres couples du système. Par convention, après échec aucune variable n'est substituable.

— si la règle de substitution a été utilisée pour x substituable, alors x n'est plus substituable dans le système qui en résulte ;
— si x n'est pas substituable dans $\mathscr{E}$, soit (x,t) le couple de $\mathscr{E}$,où elle apparaît, alors aucune règle ne rend x substituable :
  — aucune nouvelle occurrence de x ne peut apparaître par règle de substitution, car ce serait forcément par substitution d'une autre variable y, et on aurait un autre couple du système (y,v) tel que x apparaît dans v, et donc x ne serait pas substituable ;
  — aucune des règles autres de substitution, ne peut non plus rendre x substituable : aucune ne s'applique au couple (x,t) et aucune ne peut créeer de nouvelles occurrences de x.

En conclusion le nombre de variables substituables n'augmente jamais en appliquant une règle, et décroit strictement par règle de substitution.

On associe à un système $\mathscr{E}$ à unifier un couple de deux entiers $m(\mathscr{E})$ constitué du nombre de variables substituables du système, et du nombre de lettres dans tous les membres gauches du système, $m(E) \in \mathbb{N}\times\mathbb{N}$ (par convention on prend $m(\text{Echec}) = (0,0)$).

L'ensemble de couples $\mathbb{N}\times\mathbb{N}$ peut être ordonné lexicographiquement, en comparant d'abord le premier membre, puis le second, et l'ordre obtenu, que l'on note $<$, est un bon ordre. Des remarques ci-dessus on déduit que chaque règle fait strictement décroître $m(\mathscr{E})$, plus précisément si $\mathscr{E} \longrightarrow \mathscr{E}'$, alors $m(\mathscr{E}) < m(\mathscr{E}')$:

— toute règle autre que la substitution ne change pas ou fait décroître le nombre de variables substituables, et fait décroître le nombre de signes à gauche, d'où $m(\mathscr{E}) < m(\mathscr{E}')$;
— la règle de substitution fait décroître strictement le nombre de variables substituables : $m(\mathscr{E}) < m(\mathscr{E}')$.

Comme l'ordre lexicographique sur les couples d'entiers est un bon ordre sur $\mathbb{N}\times\mathbb{N}$: toute suite des règles de l'algorithme d'unification de Robinson est finie. ■

Corollaire 2.6.12 *Tout système à unifier est soit non unifiable, soit possède un unificateur principal.*

Démonstration. Corollaire immédiat de la proposition précédente 2.6.11 et du même résultat pour les systèmes résolus (lemme 2.6.10). ■

Autres algorithmes

L'algorithme décrit ci-dessus est de complexité exponentielle dans de mauvais cas, quand il faut réaliser une chaîne de substitutions avec à chaque fois plusieurs occurrences de la variable à substituer, par exemple (f binaire) :

$$\{(x_0, f\,x_1\,x_1),\ (x_1, f\,x_2\,x_2),\ \ldots\ ,\ (x_{n-1}, f\,x_n\,x_n)\}$$

Il est possible de donner un algorithme d'unification linéaire. C'est une question de représentation des termes et des substitutions. On peut en particulier donner une notion de système réduit qui permet par composition de décrire un unificateur principal. Par exemple le système à unifier de l'exemple ci-dessus, bien que non résolu, décrit une substitution par composition de substitutions élémentaires (sur une seule variable). L'absence de cycle, qu'il faut alors tester, se fait en temps linéaire et assure que le système décrit bien une substitution.

2.6.7 Règle de résolution

En restreignant la règle de résolution vue précédemment à utiliser un unificateur principal des couples de termes en présence, on va montrer que l'on obtient encore une méthode de réfutation qui est complète. C'est cette règle restreinte aux unificateur principaux que l'on appelle habituellement règle de résolution, et c'est ce que nous ferons dorénavant.

$$\frac{C \vee A_1 \vee \cdots \vee A_k \qquad D \vee \neg B_1 \vee \cdots \vee \neg B_l}{\sigma(C) \vee \sigma(D)}\ \sigma \text{ unificateur } \textit{principal} \text{ de } A_1, \ldots, A_k, B_1, \ldots, B_l$$

Pour la complétude, il suffit pour cela de montrer que l'on peut « relever » une réfutation par résolution utilisant des unificateurs quelconques, en particulier ceux instanciant par des termes clos, en une réfutation n'utilisant que des unificateurs principaux.

Lemme 2.6.13 (relèvement) *S'il existe une déduction par résolution utilisant des unificateurs quelconques de la clause C à partir d'un ensemble de clauses $\mathcal{T}$, alors il existe*

- *une clause C' et une substitution α tel que $\alpha(C') = C$;*
- *et une réfutation par résolution à partir de $\mathcal{T}$ n'utilisant que des unificateurs principaux, de la clause C'.*

Démonstration. Par induction sur la longueur de la déduction.

- Pour une déduction de longueur 0, la clause C est une clause de $\mathcal{T}$, $C_0 = C$, $s = \mathrm{id}$.
- Supposons la clause C obtenue par résolution à partir des deux clauses $D \vee A_1 \vee \cdots \vee A_k$ et $E \vee \neg B_1 \vee \cdots \vee \neg B_l$ pour σ un unificateur de $\{A_1, \ldots, A_k, B_1, \ldots, B_l\}$. Alors, par hypothèse d'induction, il existe deux substitutions α et β et deux clauses déduites par résolution avec unificateur principal, $D' \vee A'_1 \vee \cdots \vee A'_{k'}$ et $E \vee \neg B'_1 \vee \cdots \vee \neg B'_l$, avec
  - $\alpha(D') = D$, $\alpha(A'_1) = A_1, \ldots, \alpha(A'_{k'}) = A_k$, $k' \leqslant k$ (deux formules peuvent être rendues identiques par la substitution, il est possible que $k' < k$ et que $A'_i \neq A'_j$ mais $A_i = A_j$) ;
  - $\beta(E') = E$, $\beta(B'_1) = B_1, \ldots, \beta(B'_{l'}) = B_l$, $l' \leqslant l$ (même remarque).

  Les variables des clauses correspondant à des quantificateurs différents sont supposées distinctes, et donc on peut supposer les supports des substitutions α et β disjoints (en ne gardant que les variables effectivement substituées dans les formules). On peut donc les regrouper en une seule substitution $\alpha \cup \beta$ qui a même image que α sur le support de α et même image que β sur le support de β.

  La substitution $\sigma \circ (\alpha \cup \beta)$ est un unificateur de $\{A'_1, \ldots, A'_{k'}, B'_1, \ldots, B'_{l'}\}$, qui possède donc un unificateur principal σ', et il existe une substitution γ telle que $\sigma \circ (\alpha \cup \beta) = \gamma \circ \sigma'$.

  Finalement on obtient par résolution avec l'unificateur principal σ' une clause C' qui vérifie $\gamma(C') = C$. ■

Théorème 2.6.14 (fidélité et complétude de la résolution) *Un ensemble fini ou dénombrable de clauses du calcul des prédicats est non satisfaisable, si et seulement s'il est réfutable par résolution, la règle de résolution étant restreinte à utiliser un unificateur principal.*

Démonstration. Le sens direct est la complétude de la résolution, conséquence de la proposition 2.6.8 et du lemme de relèvement qui précède, puisqu'une clause qui donne la clause vide par substitution est nécessairement la clause vide.

La réciproque (fidélité) est conséquence de la validité de la règle de résolution : si un ensemble de clauses du calcul des prédicats est réfutable, c'est-à-dire que la clause vide se déduit par résolution de cet ensemble de clauses, il ne peut être satisfaisable car un modèle de cet ensemble de clauses serait un modèle de toutes les clauses déduites par résolution de celui-ci, en particulier de la clause vide, ce qui est impossible. ■

Comme en calcul propositionnel, il existe plusieurs stratégies de résolution, qui peuvent correspondre à plusieurs façons d'ordonner les règles dans les preuves de résolutions obtenues. Une règle de résolution est possible dès que deux clauses contiennent l'une un atome $Pu_1 \ldots u_n$, l'autre un atome $\neg P v_1 \ldots v_n$ et que $\{(u_1, v_1), \ldots, (u_n, v_n)\}$ est unifiable, sachant que l'unificateur (principal) peut faire apparaître d'autres identifications et mener à éliminer d'autres atomes (comme dans l'exemple du paradoxe du barbier, page 54).

2.6.8 Traitement de l'égalité

On a vu au paragraphe 2.4.2 que l'égalité pouvait être axiomatisée par des formules universelles dépendant de la signature, et qu'un ensemble de formules du calcul des prédicats égalitaire était satisfaisable si et seulement si l'ensemble obtenu de formules obtenu en ajoutant les axiomes de l'égalité était satisfaisable en calcul des prédicats non égalitaire, l'égalité étant vue comme une relation binaire ordinaire. On peut donc étendre la méthode de Herbrand au calcul des prédicats égalitaire. Les axiomes de l'égalité, sont la réflexivité, la transitivité, la symétrie, et la compatibilité avec les fonctions et les prédicats de la signature.

Exercice 11 Écrire une preuve en résolution dans le langage $(0, s, +, =)$ augmenté des symboles de Skolem utiles de $x + s0 = s\,x$ à partir des axiomes de l'addition et de l'égalité.

On peut obtenir certains résultats élémentaires sur l'addition en utilisant la règle de récurrence restreinte aux formules atomiques du langage (il y en a quand même une infinité), voire aux formules atomiques et à leurs négations.

Exercice 12 La signature est supposée contenir des symboles 0 et s (constante et fonction d'arité 1). On note $P(z, x_1, \ldots, x_k)$ une formule atomique de ce langage ayant pour variables libres $z, x_1, \ldots, x_k$. Écrire l'axiome de récurrence pour $P(z, x_1, \ldots, x_k)$ (vue comme dépendant de z), et montrer qu'il équivaut à

$$\forall x_1 \ldots \forall x_k \left[\neg P(0, x_1, \ldots, x_k) \vee \exists y (P(y, x_1, \ldots, x_k) \wedge \neg P(s\,y, x_1, \ldots, x_k)) \vee \forall z\, P(z, x_1, \ldots, x_k)\right]$$

1. Soit f un nouveau (c'est-à-dire qu'il n'est pas dans la signature) symbole de fonction d'arité f, montrer que l'énoncé précédent et l'ensemble des deux clôtures universelles des deux clauses du langage étendu suivant sont équisatisfaisables

$$\neg P(0, x_1, \ldots, x_k) \vee P(f\,x_1 \ldots x_k, x_1 \ldots, x_k) \vee P(z, x_1, \ldots, x_k),$$
$$\neg P(0, x_1, \ldots, x_k) \vee \neg P(s\,f\,x_1 \ldots x_k, x_1, \ldots, x_k) \vee P(z, x_1, \ldots, x_k)\ .$$

   (*Indication :* s'inspirer de la justification de la skolemisation dans le cas des formules prénexes ; ici la mise sous forme prénexe conduirait à faire dépendre inutilement f d'un paramètre supplémentaire).
2. Écrire les clauses dans le cas particulier où la formule $P(z, x, y)$ est $x + (y + z) = (x + y) + z$.

Exercice 13 Le langage est de signature $(0, s, +)$ (arités usuelles). On prend comme axiomes les clauses correspondant à la définition de l'addition et aux axiomes de l'égalité

1. $x + 0 = x$
2. $x' + s\,y = s(x' + y)$
3. $z = z$
4. $\neg x'' = y' \vee \neg y' = z' \vee x'' = z'$
5. $\neg x''' = y'' \vee y'' = x'''$
6. $\neg u = u' \vee \neg v = v' \vee u + v = u' + v'$
7. $\neg w = w' \vee s\,w = s\,w'$

Montrer par réfutation et règle de résolution que

1. $\forall x \forall y\, x + (y + 0) = (x + y) + 0$ est conséquence des axiomes (ajoutez les constantes de Skolem nécessaires).
2. $\forall x \forall y \forall z \left[x + (y + z) = (x + y) + z \rightarrow x + (y + s\,z) = (x + y) + s\,z\right]$ est conséquence des axiomes (ajoutez les constantes de Skolem nécessaires).
3. Montrer que l'associativité de l'addition est conséquence des énoncés démontrées par réfutation aux deux questions précédentes et des clauses obtenues comme indiquées précédemment à partir de l'instance utile du schéma de récurrence.
4. Soit $\mathscr{C}$ un ensemble de clauses, et $P x_1, \ldots x_k$ un littéral. Montrer que si $\mathscr{C} \cup \{P a_1 \ldots a_k\}$ est réfutable, où $a_1, \ldots, a_k$ sont des constantes nouvelles (qui n'apparaissent pas dans $\mathscr{C}$), alors soit $\mathscr{C}$ est réfutable par résolution, soit le littéral opposé $P^{\perp} a_1 \ldots a_k$ se déduit par résolution de $\mathscr{C}$ (*indication : procéder par récurrence sur la longueur de la preuve par résolution*).
5. En déduire qu'il existe une preuve par résolution de l'associativité de l'addition à partir des axiomes de l'égalité, de l'addition, et de clauses correspondant à l'instance utile du schéma de récurrence.

2.7 Compacité du calcul des prédicats

2.7.1 Compacité

La proposition 2.6.2 est valide pour un ensemble infini dénombrable de formules universelles du calcul des prédicats. L'ensemble des particularisations d'un ensemble dénombrable reste dénombrable

(c'est une réunion dénombrable d'ensembles dénombrables). En utilisant le théorème de compacité du calcul propositionnel (2.6.4 page 49), on peut donc généraliser le théorème de Herbrand a un ensemble $\mathscr{E}$ dénombrable de formules universelles avec la même démonstration : si $\mathscr{E}$ est non satisfaisable, un sous-ensemble fini de l'ensemble de toutes les particularisations de ces formules est non satisfaisable. Ces particularisations en nombre fini sont issues d'un sous-ensemble fini de $\mathscr{E}$, qui est donc non satisfaisable (par validité de la règle de particularisation). C'est le théorème de compacité pour les formules universelles du calcul des prédicats pur. Par skolemisation ce théorème est encore valide pour des formules quelconques, en calcul des prédicats pur.

Le théorème se généralise au calcul des prédicats égalitaire de la façon suivante. Si un ensemble $\mathscr{T}$ de formules du calcul des prédicats égalitaire n'est pas satisfaisable, l'ensemble $\mathscr{T}'$ obtenu en ajoutant les axiomes de l'égalité (qui sont dénombrables), ne l'est pas non plus en calcul des prédicats pur, par la proposition 2.4.1 page 43. Un sous-ensemble fini $\mathscr{E}$ de $\mathscr{T}'$ n'est pas satisfaisable en calcul des prédicats pur. L'ensemble des axiomes de $\mathscr{E} \cap \mathscr{T}$ complétés par tous les axiomes de l'égalité contient $\mathscr{E}$ et n'est donc pas satisfaisable. À nouveau en vertu de la proposition 2.4.1, $\mathscr{E} \cap \mathscr{T}$, qui est une partie finie de $\mathscr{T}$, n'est pas satisfaisable. On a donc démontré :

Théorème 2.7.1 (compacité du calcul des prédicats égalitaire) *Si un ensemble $\mathscr{E}$ (dénombrable) de formules du calcul des prédicats n'est pas satisfaisable, alors il existe un sous-ensemble fini de $\mathscr{E}$ qui n'est pas satisfaisable,*

ou par contraposée

si tous les sous-ensembles finis d'un ensemble $\mathscr{E}$ (dénombrable) de formules du calcul des prédicats sont satisfaisables, alors $\mathscr{E}$ est satisfaisable.

Là encore le théorème reste valide si $\mathscr{E}$ est infini quelconque, mais sa démonstration requiert une forme de l'axiome du choix.

Ce théorème est un outil fondamental d'une branche de la logique, la théorie des modèles, pour « fabriquer » de nouveaux modèles, en utilisant la forme contraposée ci-dessus.

2.7.2 Exemple : un modèle non standard de l'arithmétique de Peano

Un exemple simple est le suivant : on prend l'arithmétique de Peano définie en section 2.3.5 page 38, qui permet de développer l'arithmétique élémentaire.

On ajoute un symbole de constante c au langage. Aux axiomes de Peano on ajoute l'infinité dénombrable d'axiomes $\{s^n 0 \leqslant c \ / \ n \in \mathbb{N}\}$ ($s^n 0 := \underbrace{s \ldots s}_{n} 0$) pour obtenir une théorie T. Toute partie finie A de cette théorie T a pour modèle $\mathbb{N}$, avec l'interprétation usuelle pour la signature de l'arithmétique de Peano, et c interprété par n_0, le plus grand entier n tel que $s^n 0 \leqslant c$ apparaît dans F. En effet F contient soit des axiomes de l'arithmétique de Peano, valides dans $\mathbb{N}$, soit des nouveaux axiomes $s^n 0 \leqslant c$, pour $n \leqslant n_0$. On en déduit par le théorème de compacité du calcul des prédicats que la théorie T elle même a un modèle. Ce modèle ne peut être obtenu en enrichissant l'interprétation usuelle sur $\mathbb{N}$, car il n'est pas possible d'interpréter c par un entier de $\mathbb{N}$. Ce modèle contient forcément un entier dit *non standard*, qui est supérieur à tous les entiers dits standards, qui sont les interprétations des termes $s^n 0$. Il n'existe pas de terme pour désigner cet entier non standard dans le seul langage de l'arithmétique de Peano. Tous les axiomes de l'arithmétique de Peano, en particulier le schéma d'axiomes de récurrence (pour les formules de l'arithmétique de Peano, où c n'apparaît pas !), restent pourtant bien valides dans ce nouveau modèle.

Le modèle $\mathscr{N}$ de l'arithmétique de Peano ainsi obtenu, dit non standard, ne peut être isomorphe au modèle standard $\mathbb{N}$. On montre facilement que l'application de $\mathbb{N}$ dans $\mathscr{N}$ qui à n associe $\overline{s^n 0}^{\mathscr{N}}$ est un morphisme injectif, mais non bijectif.

En fait cette démonstration fonctionne pour n'importe quelle théorie arithmétique du premier ordre, en particulier la théorie de tous les énoncés (du premier ordre) vrais dans le modèle standard $\mathbb{N}$. Le modèle non standard $\mathscr{N}$ satisfait alors les mêmes énoncés que le modèle standard $\mathbb{N}$ dans le langage d'origine, mais ne peut être isomorphe à $\mathbb{N}$.

Exercice 14 On rappelle que la classe des ensembles infinis est axiomatisée par une infinité de formules données à la page 37, (2.2), soit I cette théorie.

1. En utilisant le théorème de compacité du calcul des prédicats égalitaire, montrer que l'on ne peut pas axiomatiser au premier ordre la classe de tous les ensembles finis en logique du premier ordre (montrer que s'il existait une telle théorie T, alors $T \cup I$ serait satisfaisable, ce qui est absurde).

2. En déduire que la classe des ensembles infinis n'est pas finiment axiomatisable au premier ordre (on ne peut pas trouver une théorie finie du premier ordre dont les modèles sont seulement les ensembles infinis).

Index

Table des matières