Cours de logique

Cassis

Janvier 2023

Table des matières

Introduction

Ce document est un projet que j'écris lors de ma première année de master d'informatique. Son objectif est d'être un cours de logique relativement abordable pour un lecteur avec un bagage moyen de licence de mathématiques, et s'attache à une certaine rigueur pour la majeure partie de ce qui est décrit.

La structure suit trois grandes parties : une partie de logique générale, qui commence par décrire les structures inductives et finit par décrire la théorie des ensembles, en passant par les théorèmes de complétude et d'incomplétude de Gödel. Les derniers chapitres de la première partie sont assez indépendants du reste, mais sont un prolongement naturel de ce qui est donné dans les premiers chapitres (ils sont la mise en œuvre directe des constructions logique du calcul des prédicats, en permettant alors d'avoir des théories logiques concrètes permettant d'étudier réellement les maths et de retomber sur nos pieds avec la pratique habituelle).

La deuxième partie s'attache à décrire l'isomorphisme de Curry-Howard et le lien entre le lambda-calcul et la logique (intuitionniste principalement). Les premiers chapitres s'attachent à une étude relativement générale du lambda-calcul quand les dernier chapitres se concentrent sur l'aspect calculatoire de la logique à travers cet isomorphisme.

La dernière partie est l'introduction du langage catégorique dans l'étude de la logique. Les premiers chapitres constituent une courte introduction aux catégories, pour permettre dans les chapitres suivants de donner des aspects logiques que la théorie des catégorie permet de décrire.

Ce document est avant tout un recueil de cours que j'ai lus (parfois intégrés suffisamment pour ne pas les lire en écrivant, parfois qui sont presque du mot à mot). Pour la partie 1, mes cours de L3 de *Logique*, donnés par Natacha Portier à l'ENS de Lyon, ont constitué l'apport principal. Le chapitre sur les machines de Turing a été, lui, surtout extrait de mes cours de *Fondement de l'Informatique* de L3, donnés par Pascal Koiran. La partie 2 vient, elle, de cours de M1 donnés par Colin Riba, principalement le cours nommé *Proofs and Programs*. La dernière partie vient de lectures diverses, et beaucoup de ce qui y est dit peut se retrouver dans le livre *Category Theory* de Steve Awodey, une bonne partie étant également extraite du cours *Proofs and Programs*.

Première partie

Logique

Chapitre 1

Structure inductive et calcul propositionnel

Nous traiterons dans ce chapitre des propositions en tant qu'objets syntaxiques. Pour cela, nous introduirons dans la première section les outils de base pour raisonner sur des ensembles inductifs et des prédicats inductifs. Nous construirons ensuite le calcul propositionnel du premier ordre sans quantificateur, pour définir la notion de valuation et le calcul en déduction naturel (qui sera notre choix de formalisme de preuve pour ce document). Nous y prouverons alors les théorèmes de compacité, de correction et de complétude. La fin du chapitre consistera en une introduction aux algèbres de Boole puis à une construction de l'algèbre de Lindenbaum-Tarski du calcul propositionnel.

1.1 Construction inductive

En logique mathématique, nous manipulons principalement des objets syntaxiques, signifiant que l'on considère des suites finies de termes définis par une grammaire inductive. On peut en général représenter de tels objets par des arbres : ainsi une proposition peut se dessiner comme un arbre où chaque nœud est un connecteur logique reliant des sous-formules. Nous en donnons un exemple dans la figure 1.1. Une définition par induction se ramènera, d'une façon ou d'une autre, à une structure d'arbre. Ainsi, comme toutes les définitions par inductions partagent des similarités, nous allons en introduire des outils permettant leur étude systématique, comme par exemple le raisonnement par induction structurelle. Nous utiliserons comme exemple dans cette section les entiers naturels et les arbres binaires, le premier par sa facilité (l'ensemble n'est constitué que de deux règles) et le second pour donner un exemple un peu plus général.

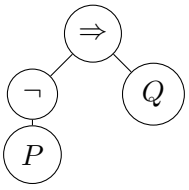


Figure 1.1 – Représentation de la proposition $\neg P \implies Q$.

1.1.1 Forme de Backus-Naur

Nous utiliserons, pour définir un ensemble inductif, une grammaire dite en forme de Backus-Naur (abrégée BNF pour *Backus-Naur Form*), dont le principe est de lister les constructeurs d'un ensemble inductif. Une BNF se présente de la façon suivante :

$$\alpha, \beta ::= \alpha_1 \mid \alpha_2 \mid \alpha_3 \cdots$$

qui se lit :

- Les éléments de l'ensemble que nous définissons seront notés α ou β.

- α_i est une règle permettant de construire un nouvel élément de notre ensemble inductif à partir d'éléments déjà construits. Chaque règle possède une arité associée, qui est le nombre d'arguments nécessaires pour construire un nouvel élément de l'ensemble inductif.

Nous allons maintenant donner deux exemples :

Exemple 1.1.1 (Entiers naturels)**.** Les entiers naturels peuvent être définis par

$$n, m ::= 0 \mid Sn$$

ce qui signifie que $0 \in \mathbb{N}$ et que si $n \in \mathbb{N}$ alors $Sn \in \mathbb{N}$ (où le constructeur S est l'opération successeur, c'est-à-dire la fonction $n \mapsto n + 1$).

Exemple 1.1.2 (Arbres binaires étiquetés par un ensemble)**.** Soit E un ensemble. On définit les arbres binaires étiquetés par E, notés $T(E)$ par la grammaire

$$t, t' ::= \texttt{Nil} \mid \texttt{Node}(e, t, t')$$

où $e \in E$. Un arbre binaire est donc soit vide, soit un nœud relié à un sous-arbre gauche et un sous-arbre droit.

Remarquons que cette définition inductive peut se faire en introduisant des constructeurs : pour chaque règle de construction de notre grammaire, on associe une « fonction » prenant un nombre fixé d'arguments inductifs pour générer un nouvel objet inductif.

Exemple 1.1.3 (Arbres binaires)**.** On peut définir la règle précédente de la façon suivante :

- il existe une constante $\texttt{Nil} \in T(E)$
- pour $e \in E$, il existe une fonction $\texttt{Node}_e : T(E) \times T(E) \to T(E)$

Si l'on s'abstrait des détails de notations, on peut définir une grammaire inductive comme un ensemble C de constructeurs, avec une fonction α qui à un constructeur associe un entier, qui est le nombre d'arguments inductifs de la fonction permettant de construire un nouvel élément.

Définition 1.1.1 (Définition inductive)**.** *Comme dit précédemment, on appelle définition inductive un couple $\langle C, \alpha \rangle$ où $\alpha : C \to \mathbb{N}$. On appelle arité du constructeur c l'entier $\alpha(c)$.*

Dans l'exemple 1.1.3 la définition inductive est donc $\langle \{0\} + E, \alpha \rangle$ où $\alpha(0) = 0$ et $\alpha(e) = 2$ pour $e \in E$.

1.1.2 Ensemble défini par induction

Nous avons vu la définition d'un ensemble par une grammaire, mais il convient encore de donner une construction d'un tel ensemble inductif. Il existe deux constructions dont on peut montrer qu'elles sont équivalentes : la construction « par le bas » et la construction « par le haut ». Nous n'étudierons ici que la construction par le bas, car celle-ci a le mérite d'être plus explicite.

Définition 1.1.2 (Construction par le haut d'un ensemble inductif)**.** *Soit $\langle C, \alpha \rangle$ une définition inductive. A une telle définition on associe :*

- *l'ensemble $I_0 = \{c \in C \mid \alpha(c) = 0\}$*
- *pour un ensemble I_n, l'ensemble $I_{n+1} = \{(c, x) \in C \times I_n^{\alpha(c)} \mid \alpha(c) > 0\} \cup I_n$*

On peut alors définit l'ensemble $I = \bigcup_{n \in \mathbb{N}} I_n$ associé à la définition inductive.

Remarque 1.1.1. Remarquons que, par construction, $I_n \subseteq I_{n+1}$. I est alors l'ensemble des éléments qui appartiennent à un certain I_n. L'ensemble que nous avons construit est donc celui des éléments construits depuis une constante et une application finie de constructeurs.

Ces considérations nous mènent à donner le lemme suivant :

Lemme 1.1.1 (Lecture unique)**.** *Soit un ensemble inductif E et $e \in E$. Alors soit e est une constante (constructeur d'arité 0) soit il existe c, x tels que $e = (c, x)$ où c est un constructeur d'arité k et x est un $k-$uplet d'éléments de E. Ces cas sont tous exclusifs, au sens où les éléments c et x sont uniques.*

Démonstration. Par définition, $e \in E$ signifie qu'il existe un $n \in \mathbb{N}$ tel que $e \in E_n$ (en notant E_n les étapes de notre construction inductive). Dans ce cas, soit $N = \{m \in \mathbb{N} \mid e \in E_m\}$. Comme $n \in N$ et que N est une partie de $\mathbb{N}$, il existe un plus petit élément, notons-le m. Si $m = 0$ alors e est une constante. Si $m \neq 0$, alors $e = (c, x)$ où c est un constructeur d'arité k et x est un k-uplet d'éléments de E_{m-1} (la cas où $e \in E_{m-1}$ est impossible puisque m est le minimum des étapes où e est construit).

Pour l'unicité, on remarque que si $(c, x) = e = (c', x')$ alors $c = c'$ et $x = x'$. □

Remarque 1.1.2. Nous pouvons considérer un constructeur c comme une fonction de son ensemble inductif dans lui-même (de la forme $c : E^{\alpha(c)} \to E$). Nous confondrons en général les deux objets. Remarquons aussi que chaque constructeur est injectif par construction.

Si toute cette construction semble alambiquée pour une notion aussi simple, avoir formulé de façon rigoureuse les définitions inductives permet de faire des preuves rigoureuses sur les structure inductives : c'est ce qu'on appelle une preuve par induction structurelle.

Théorème 1.1.1 (Induction structurelle)**.** *Soit une propriété P dépendant d'un paramètre x, sur un ensemble inductif E généré par $\langle C, \alpha\rangle$. Alors P est vraie sur E tout entier si et seulement si $P(c)$ est vraie pour toute constante $c \in C$ et si pour c d'arité k, pour tout $x_1, \ldots, x_k \in E$, $P(x_1)$ et $P(x_2)$ et ... et $P(x_k)$ implique $P(c, (x_1, \ldots, x_k))$.*

Démonstration. Un sens est évident : si P est vraie sur E, les deux conditions précisées sont vraies. Il suffit donc de prouver le sens réciproque. Procédons par récurrence sur E_n :

- Par hypothèse, P est vraie sur E_0.
- Si P est vraie sur E_n, alors soit $e \in E_{n+1}$. Si $e \in E_n$, $P(e)$ est vraie. Par le lemme précédent, $e = (c, x)$ or chaque composante de x est dans E_n donc par hypothèse de récurrence, $P(x_1), \ldots, P(x_{\alpha(c)})$ sont vraies. On en déduit par hypothèse que $P(c, x)$ est vraie, donc P est vraie sur E_{n+1}.

Donc par récurrence, P est vraie sur E. □

De façon similaire, on peut définir une fonction uniquement grâce aux constructeurs.

Théorème 1.1.2 (Définition d'une fonction par induction)**.** *Soit E un ensemble défini par induction par $\langle C, \alpha\rangle$ et F un ensemble quelconque. Supposons qu'à chaque constante c_0 on associe un élément $f_{c_0} \in F$, et qu'à chaque constructeur c d'arité a on associe une fonction $f_c : F^a \to F$. Alors il existe une unique fonction f telle que $f(c_0) = f_{c_0}$ et pour chaque constructeur c, $f((c, (x_1, \ldots, x_a))) = f_c(f(x_1), \ldots, f(x_a))$.*

Démonstration. La preuve de l'existence et de l'unicité se fait par récurrence sur E_n en prenant la proposition $P(n) =$ pour tout $e \in E_n$, il existe une unique fonction vérifiant ces conditions :

- Sur E_0, la seule fonction correspondante est $f_0 : c_0 \mapsto f_{c_0}$.
- Si on suppose définie une unique fonction $f_n : E_n \to F$, alors comme tout élément de E_{n+1} est soit de la forme (c, x) soit déjà dans E_n, un élément dans E_n sera envoyé dans F par f_n (par unicité de f_n partant de E_n) et un élément de la forme (c, x) sera défini par la relation de récurrence : $f_{n+1}(c, x) = f_c(f(x_1), \ldots, f(x_n))$. Cette relation caractérise donc f_{n+1} de façon unique.

D'où le résultat par récurrence. □

On peut représenter la définition inductive d'une telle fonction en disant que les diagrammes suivants, pour chaque constructeur c, commutent (dire qu'un diagramme commute signifie que deux chemins entre deux mêmes points sont égaux) :

$$\begin{array}{ccc} E \times \ldots \times E & \xrightarrow{f \times \ldots \times f} & F \times \ldots \times F \\ \downarrow{\scriptstyle c} & & \downarrow{\scriptstyle f_c} \\ E & \xrightarrow{\qquad f \qquad} & F \end{array}$$

Exemple 1.1.4 (Liste)**.** Soit A un ensemble, on définit l'ensemble $\texttt{List}(A)$ par la grammaire

$$l ::= \texttt{Nil} \mid \texttt{cons}(a, l)$$

On notera par exemple [ ] pour écrire la liste vide (plutôt que $\texttt{Nil}$) ou $[a, b, c]$ pour la liste construite par $\texttt{cons}(a, \texttt{cons}(b, \texttt{cons}(c, \texttt{Nil})))$, ou encore $a :: [b, c]$.

Définissons alors la fonction $|-|$ qui à une liste associe sa longueur. Par exemple $|[a, b, c]| = 3$. Pour cela, on définit la fonction par induction :

- $|[\,]| = 0$
- pour tout $a \in A$, $|a :: l| = 1 + |l|$.

Ainsi $|[a, b, c]| = 1 + |[b, c]| = 1 + 1 + |[c]| = 1 + 1 + 1 + |[\,]| = 3$.

Exercice 1.1.1 (Les opérations dans $\mathbb{N}$)**.** Définir les opérations add, mult et exp, définies naturellement par $\text{add}(n, m) = n + m$, $\text{mult}(n, m) = n \times m$ et $\exp(a, b) = a^b$.
Indication : on peut définir une fonction à deux variables comme une fonction à une variable qui associe une fonction à une variable, et alors utiliser une définition inductive sur le premier argument.

Exercice 1.1.2 (Une meilleur récurrence)**.** Remarquons qu'avec notre définition, si l'on définit $\mathbb{N}$ par le constructeur 0 d'arité 0 et le constructeur s d'arité 1, alors le principe d'induction pour définir une fonction de la forme $\mathbb{N} \to F$ consiste à donner un élément $c_0 \in F$ et une fonction $F \to F$. Montrer que ce principe de définition par induction permet de construire des fonctions définies par un élément $c_0 \in F$ et une fonction $\mathbb{N} \times F \to F$.
Indication : comme ce principe marche pour tout F, on peut chercher une fonction à valeur dans $\mathbb{N} \times F$ qu'on projettera ensuite sur F.

Exercice 1.1.3. En réutilisant la définition précédente des arbres binaires étiquetés par un ensemble E, construire une fonction donnant le nombre de nœuds de l'arbre (un arbre vide n'est pas un nœud).

Nous décrirons désormais nos ensembles inductifs directement par leur BNF. Nous supposerons donc assimilé le passage d'une grammaire à l'ensemble inductif qu'il décrit.

1.1.3 Règle d'induction

Nos résultats théoriques sont assez puissants pour définir beaucoup de choses, mais on peut remarquer que le formalisme qui leur est associé est assez lourd : nous allons donc définir une façon plus simple d'écrire nos définitions inductives. Pour construire un ensemble inductif, la grammaire en BNF est toute indiquée (elle est lisible et donne toutes les informations souhaitées). Cependant, pour définir une fonction par induction, nous allons donner un formalisme plus efficace, appelé règles d'induction. Comme nous l'avons dit, on peut classer nos constructeurs en deux sortes : ceux qui n'acceptent pas d'argument (les constantes) et les autres, qui acceptent un nombre précis d'arguments. Ainsi, si l'on veut définir une fonction $f : E \to F$ avec E inductif, il suffit d'écrire une « règle » permettant de passer d'un ensemble d'image à leur image par la fonction d'induction associée au constructeur.

Exemple 1.1.5. Supposons que l'on veuille définir $- \times 2 : \mathbb{N} \to \mathbb{N}$, il nous suffit d'écrire que $0 \times 2 = 2$ et $(Sn) \times 2 = S(S(n \times 2))$.

Mais dans le cas de structure inductives plus lourdes, il peut être avantageux d'avoir une présentation moins linéaire, c'est pourquoi l'on introduit la notion de règle d'induction.

Définition 1.1.3 (Règle d'induction)**.** *Pour définir une fonction f_c pour un constructeur c, on va utiliser la présentation suivante :*

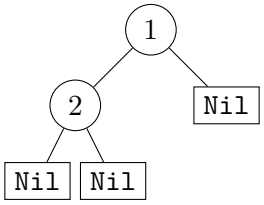
$$\frac{a_1 \quad a_2 \quad \dots \quad a_n}{f_c(a_1, \dots, a_n)}$$

Remarque 1.1.3. Dans le cas d'une constante c_0, la règle s'écrit directement

$$\frac{}{f_{c_0}}$$

On appelle souvent dérivation l'application d'une suite de règles d'inductions pour trouver l'image d'une fonction inductive, formant un arbre dont la racine est le résultat et les feuilles des constantes.

Exemple 1.1.6. Pour l'arbre A suivant :



nous allons calculer son nombre de nœuds, en utilisant la fonction définie dans l'exercice précédent, dont on peut exprimer les règles d'induction de la façon suivante :

$$\frac{}{|\texttt{Nil}| = 0} \qquad \frac{|g| = k \quad |d| = k'}{|\texttt{Node}(e, g, d)| = 1 + k + k'}$$

La dérivation est donc :

$$\frac{\dfrac{\dfrac{}{|\texttt{Nil}| = 0} \quad \dfrac{}{|\texttt{Nil}| = 0}}{|\texttt{Node}(2, \texttt{Nil}, \texttt{Nil})| = 1} \quad \dfrac{}{|\texttt{Nil}| = 0}}{|A| = 2}$$

Exercice 1.1.4. Défnir les règles d'induction de la fonction $|-|$ sur les listes, associant à une liste sa longueur. Écrire l'arbre de dérivation du calcul de $|[a, b, c]|$.

1.1.4 Relation inductive

Intéressons-nous désormais aux relations entre objets inductifs. Comme précédemment, nous utiliserons des règles d'induction pour les définir (puisqu'une relation $\mathcal{R}$ peut être vue comme une fonction dans $\{0, 1\}$), mais nous distinguerons deux types de prémisses : les prémisses du système formel et les prémisses « meta ». Les premières sont celles qui interviennent directement dans les constructeurs de notre ensemble inductif, tandis que les deuxièmes sont des conditions que l'on peut dire extérieures à notre formalisme. Nous écrirons à côté de la règle les prémisses non syntaxiques (celles qui n'appartiennent pas au système formel). Attention cependant : nous nommerons souvent nos règles en donnant leur nom à droite de la barre, il faudra donc comprendre lorsqu'il y a une condition écrite à droite qu'elle est une prémisse non syntaxique, et lorsqu'il y a seulement un nom (par exemple $\to_e$) que c'est le nom de la règle.

Exemple 1.1.7. Nous allons définir le prédicat P_k sur $\texttt{List}(\mathbb{N})$ signifiant qu'une liste d'entiers possède un certain élément k :

$$\frac{}{P_k(\texttt{cons}(x,l))}\ x=k \qquad \frac{P_k(l)}{P_k(\texttt{cons}(x,l))}$$

Exercice 1.1.5. Montrer que $P_2([5,3,4,2,1,9])$ par un arbre de dérivation.

Remarque 1.1.4. On peut aussi définir des relations inductives mutuelles, par exemple un prédicat « être pair » faisant appel à un prédicat « être impair » faisant lui-même appel au premier prédicat (les deux ayant un cas de base), mais nous ne traiterons pas ce genre d'extension ici.

Donnons une définition de relation donnée par induction, car c'est un cas particulier qui nous sera utile.

Définition 1.1.4 (Relation inductive)**.** *Soient E, F deux ensembles. Soit un triplet $\langle C, \alpha, (D_c)_{c\in C}\rangle$ où $\alpha : C \to \mathbb{N}, D_c : (E\times F)^{\alpha(c)} \to E\times F$ et C est un ensemble. On dit que la relation $\mathcal{R}$ est définie par induction par ce triplet si $\mathcal{R}$ est la plus petite relation telle que pour tout $c \in C$, si pour tout $i \in \{0,\dots,\alpha(c)\}, \mathcal{R}\langle x_i, y_i\rangle$ alors $\mathcal{R}(D_c(\langle x_1,y_1\rangle,\dots,\langle x_{\alpha(c)},y_{\alpha(c)}\rangle))$. On note en général le triplet de la façon suivante : pour chaque $c\in C$, on écrit une règle de la forme*

$$\frac{\mathcal{R}(x_1,y_1) \quad \mathcal{R}(x_2,y_2) \quad \dots \quad \mathcal{R}(x_{\alpha(c)},y_{\alpha(c)})}{\mathcal{R}(D_c(\langle x_1,y_1\rangle,\dots,\langle x_{\alpha(c)},y_{\alpha(c)}\rangle))}\ c$$

avec x_i, y_i non spécifiés, qu'il faut alors considérer comme quantifiée sur respectivement tout E et tout F.

Exercice 1.1.6. Montrer que si deux propositions sont stables par les règles définissant une relation inductive, alors l'intersection de ces propositions est aussi stable par ces règles. En déduire que la relation $\mathcal{R}$ définie par induction précédemment est l'intersection des propositions stables par les règles.

Exercice 1.1.7. Montrer qu'une relation définie par induction par un ensemble de règles $\langle C, \alpha, (D_c)\rangle$ est non vide si et seulement s'il existe $c \in C$ tel que $\alpha(c) = 0$.

Remarque 1.1.5. Il arrivera que l'on écrive une règle dont les prémisses contiennent une ou des conditions qui ne sont pas de la forme $\mathcal{R}(x,y)$. Dans ce cas on peut considérer que l'on ajoute une règle de construction inductive pour chaque élément vérifiant cette prémisse et que le reste de la règle est paramétré par cet élément.

Nous pouvons maintenant réécrire notre théorème d'induction structurelle avec le formalisme des règles d'induction :

Corollaire 1.1.1 (Preuve par induction)**.** *Soit une relation $\mathcal{R}$ entre deux ensembles E, F générée par un ensemble de règles d'induction. Alors pour tout prédicat P sur $E\times F$, pour montrer que $\mathcal{R}\subseteq P$ il suffit de montrer, pour chaque règle $c, \alpha(c), D_c$ définissant $\mathcal{R}$ la propriété suivante : pour tout $(x_1,\dots,x_{\alpha(c)}) \in P^{\alpha(c)}, P(D_c(x_1,\dots,x_{\alpha(c)}))$, ce qui revient à montrer :*

$$\frac{P(x_1) \quad P(x_2) \quad \dots \quad P(x_{\alpha(c)})}{P(D_c(x_1,\dots,x_{\alpha(c)}))}\ c$$

Démonstration. C'est une conséquence directe du fait que $\mathcal{R}$ est définie comme la plus petite relation possédant cette propriété. □

1.2 Définition des propositions

Cette section vise à présenter les propositions, que nous introduirons comme des objets syntaxiques. La façon de séparer les propositions de leur interprétation (par exemple de dire que $\forall x.x = x$ n'est pas le même objet que $\top$) est une démarche qui se révélera fructueuse pour pouvoir trouver le bon formalisme pour parler de preuves et de valeur de vérité. On peut voir cela comme un analogue à la distinction entre un polynôme formel et une fonction polynomiale : le premier permet d'étudier de façon plus systématique le deuxième.

La première approche que nous aurons sera celle du calcul des propositions : on peut considérer une proposition (dans son sens intuitif) comme une phrase portant sur des objets mathématiques, par exemple avec la proposition $1 = 1$, auxquelles on associe des mots de liaison, comme $(1 = 1) \vee (0 = 1)$ indiquant que $1 = 1$ ou $0 = 1$. Cependant, on peut s'abstraire des prédicats de la forme $1 = 1$ et remplacer ceux-ci par une variable booléenne : une variable x pouvant valoir 1 (si elle représente une proposition atomique vraie) ou 0 (si elle est fausse). Dans un second temps seulement, nous verrons le calcul des prédicats, permettant de parler plus précisément de ces propositions atomiques.

1.2.1 Calcul des propositions

Définition 1.2.1 (Proposition)**.** *Soit un ensemble infini (dénombrable) de variables booléennes $\mathcal{V}$ que nous noterons $x_1, x_2, \ldots$, nous définissons l'ensemble des propositions $\mathcal{P}$ par la grammaire suivante :*

$$P, Q ::= x_i \mid \top \mid \bot \mid (P \to Q) \mid (P \vee Q) \mid (P \wedge Q) \mid (\neg P)$$

où $\top$ et $\bot$ sont deux propositions atomiques constantes (appelées « vrai » et « absurde »).

Remarque 1.2.1. Un constructeur de cette grammaire est aussi appelé connecteur logique.

Exemple 1.2.1. Voici quelques propositions :

- $\top$
- $((\top \wedge (\neg x_1)) \vee (x_2 \to x_3))$

Remarque 1.2.2. Nous simplifierons l'écriture en rendant $\neg$ le constructeur le plus prioritaire, $\vee$ et $\wedge$ prioritaire sur $\to$ et $\wedge$ prioritaire sur $\vee$ (le tout pour omettre un maximum de parenthèses). Par exemple la dernière proposition donnée en exemple peut se réécrire $\top \wedge \neg x_1 \vee (x_2 \to x_3)$. Enfin, l'opération $\to$ sera associative à droite : $x_0 \to x_1 \to x_2$ se lira $x_0 \to (x_1 \to x_2)$.

Nous avons déjà une intuition assez naturelle de ce que veut dire chaque constructeur : par exemple, $P \wedge Q$ est vraie exactement lorsque P et Q à la fois sont vraies. Cette idée intuitive peut se formaliser par ce que l'on appelle des tables de vérité, représentant pour un connecteur la valeur de la proposition qu'il construit en fonction de la valeur de vérité de ses entrée. Voici les tables de vérité des différents connecteurs logiques :

P	$\neg P$
0	1
1	0

P	Q	$P \to Q$
0	0	1
0	1	1
1	0	0
1	1	1

P	Q	$P \vee Q$
0	0	0
0	1	1
1	0	1
1	1	1

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

Figure 1.2 – Tables de vérité

On peut alors essayer de donner une table de vérité pour n'importe quelle formule propositionnelle, mais les tables de vérité ont plusieurs inconvénients : elles sont longues à faire et elles sont difficiles à étudier mathématiquement. Nous leur préférerons donc la notion de valuation, que nous allons maintenant introduire.

Définition 1.2.2 (Valuation)**.** *Soit une fonction $\nu : \mathcal{V} \to \{0,1\}$ (qui à une variable associe si elle est vraie ou non), nous allons étendre par induction ν en une fonction $\nu^* : \mathcal{P} \to \{0,1\}$ par les règles suivantes :*

$$\frac{}{\nu^*(\top) = 1}\ \top \qquad \frac{}{\nu^*(\bot) = 0}\ \bot \qquad \frac{}{\nu^*(x_i) = \nu(x_i)}\ Ax$$

$$\frac{\nu^*(P) = b}{\nu^*(\neg P) = 1 - b}\ \neg \qquad \frac{\nu^*(P) = b_1 \quad \nu^*(Q) = b_2}{\nu^*(P \to Q) = 1 - b_1 + b_1 \times b_2}\ \to$$

$$\frac{\nu^*(P) = b_1 \quad \nu^*(Q) = b_2}{\nu^*(P \vee Q) = b_1 + b_2 - b_1 \times b_2}\ \vee \qquad \frac{\nu^*(P) = b_1 \quad \nu^*(Q) = b_2}{\nu^*(P \wedge Q) = b_1 \times b_2}\ \wedge$$

Remarque 1.2.3. Les opérations utilisées pour les valuations de $P \to Q$ et $P \vee Q$ sont un peu longues, mais elles correspondent simplement à leur table de vérité.

Exercice 1.2.1. Montrer par une dérivation que chaque table de vérité est vérifiée avec notre définition de ν^*.

Exercice 1.2.2. Montrer que deux valuations ν, ν' donnent la même valeur sur une proposition P si pour toute variable x_i intervenant dans P, $\nu(x_i) = \nu'(x_i)$.

On peut donc se restreindre aux valuations ν partielles, c'est-à-dire n'associant qu'un nombre fini de valeurs à l'ensemble $\mathcal{V}$, dont le domaine contient les variables intervenant dans une proposition P, pour évaluer P.

Exemple 1.2.2. Soit la proposition $P := x_0 \vee x_1 \to x_2$, et la valuation

$$\nu : x_0 \mapsto 0, x_1 \mapsto 1, x_2 \mapsto 0$$

on peut alors calculer $\nu^*(P)$:

$$\dfrac{\dfrac{\dfrac{}{\nu^*(x_0) = 0}\ \text{Ax} \quad \dfrac{}{\nu^*(x_1) = 1}\ \text{Ax}}{\nu^*(x_0 \vee x_1) = 1}\ \vee \quad \dfrac{}{\nu^*(x_2) = 0}\ \text{Ax}}{\nu^*(x_0 \vee x_1 \to x_2) = 0}\ \to$$

Si l'on considère que 1 signifie que la proposition est vraie et que 0 signifie que la proposition est fausse, on peut donc dire qu'une proposition est vraie pour une certaine valuation.

Définition 1.2.3 (Modèle d'une proposition)**.** *Soit ν une valuation. On dit que P est vraie (ou satisfaite) pour la valuation ν, ou encore que ν est modèle de P, et on note $\nu \models P$, si $\nu^*(P) = 1$.*

Définition 1.2.4 (Tautologie)**.** *On dit que P est une tautologie si pour toute valuation ν, $\nu^* \models P$.*

Par la suite, on confondra directement ν et ν^*.

1.2.2 Satisfiabilité et compacité

Étendons maintenant notre notion de vérité pour un ensemble de propositions :

Définition 1.2.5 (Satisfiabilité)**.** *On dit qu'une formule P est satisfiable s'il existe une valuation ν telle que $\nu \models P$.*

Si $\mathbf{P} \subseteq \mathcal{P}$ est un ensemble de proposition, alors on dit que $\mathbf{P}$ est satisfiable s'il existe une valuation ν telle que pour tout $P \in \mathbf{P}$, $\nu \models P$. On note alors $\nu \models \boldsymbol{P}$.

Remarque 1.2.4. Cette digression est hors du cadre de ce document, mais le problème de trouver, étant donnée une proposition, si elle est satisfiable, est un problème dit NP-complet. Il est historiquement le premier problème démontré comme étant NP-complet et revêt en ce sens une importance primordiale en théorie de la complexité algorithmique.

Donnons aussi la définition d'un ensemble contradictoire :

Définition 1.2.6 (Ensemble contradictoire)**.** *Un ensemble $\mathbf{P} \subseteq \mathcal{P}$ (potentiellement infini) est contradictoire s'il n'existe pas de valuation ν telle que $\nu \models \mathbf{P}$.*

Le calcul des propositions n'est malheureusement pas le plus pertinent, car il manque à ce système deux éléments essentiels : pouvoir s'appliquer à des objets mathématiques et pouvoir introduire des quantifications. Nous nous contenterons donc de donner un théorème central de la logique : le théorème de compacité.

Définition 1.2.7 (Ensemble finiment satisfiable, contradictoire)**.** *Soit $\mathbf{P}$ un ensemble (infini) de propositions, on dit que $\mathbf{P}$ est finiment satisfiable si pour toute partie finie $P \subseteq_{\text{fin}} \mathbf{P}$, P est satisfiable.*

De même, $\mathbf{P}$ est dite finiment contradictoire s'il existe une partie finie de $\mathbf{P}$ qui est contradictoire.

Théorème 1.2.1 (Compacité de la logique propositionnelle)**.** *Soit $\mathbf{P}$ un ensemble de propositions. Alors $\mathbf{P}$ est satisfiable si et seulement si $\mathbf{P}$ est finiment satisfiable.*

Démonstration. Il est évident qu'un ensemble satisfiable est finiment satisfiable, le théorème tient bien sûr dans le sens réciproque.

Soit un ensemble de propositions $\mathbf{P}$. Nous allons définir une suite (ε_n) qui définira notre valuation par $\nu(x_n) = \varepsilon_n$. Cette construction se fera par induction :

- Tout d'abord, deux cas sont possibles :
  1. pour tout sous-ensemble fini B, il existe une valuation δ satisfiant B et telle que $\delta(x_0) = 0$: dans ce cas on définit $\varepsilon_0 = 0$.
  2. il existe un sous-ensemble fini B_0 tel que pour toute valuation δ satisfiant B_0, $\delta(x_0) = 1$: dans ce cas on définit $\varepsilon_0 = 1$.
- Montrons l'initialisation de la propriété « il existe une suite $\varepsilon_0, \dots, \varepsilon_n$ telle que pour tout sous-ensemble B il existe un valuation δ satisfaisant B et telle que $\delta(x_i) = \varepsilon_i$ » : dans le cas 1, le résultat est direct. Dans le cas 2, alors pour une partie finie B on considère $B \cup B_0$: si $\delta \models B \cup B_0$ alors $\delta \models B_0$ donc $\delta(x_0) = 1$ et $\delta \models B$, donc notre hypothèse est vérifiée.
- Supposons qu'il existe une suite $\varepsilon_0, \dots, \varepsilon_n$ vérifiant les conditions énoncées, alors on peut encore séparer deux cas :
  1. pour toute partie finie B il existe une valuation $\delta \models B$ telle que $\delta(x_i) = \varepsilon_i$ pour $i \leq n$, on a aussi $\delta(x_{n+1}) = 0$.
  2. il existe une partie finie B_{n+1} telle que pour toute valuation $\delta \models B$ telle que $\delta(x_i) = \varepsilon_i$ pour $i \leq n$, on a $\delta(x_{n+1}) = 1$.

  Dans le premier cas, on définit $\varepsilon_{n+1} = 0$ et l'hérédité est vérifiée, et dans le deuxième cas on définir $\varepsilon_{n+1} = 1$ et alors pour B une partie finie, il existe $\delta \models B \cup B_{n+1}$ avec $\delta(x_i) = \varepsilon_i$ pour $i \leq n$, et comme $\delta \models B_{n+1}$ on en déduit qu'il existe $\delta \models B$ avec $\delta(x_i) = \varepsilon_i$ pour $i \leq n+1$.

Ainsi, par récurrence, nous avons défini la valuation $\nu : x_i \mapsto \varepsilon_i$ qui satisfait toute proposition de $\mathbf{P}$. □

Exercice 1.2.3 (Formulation équivalente du théorème de compacité)**.** Montrer que le théorème de compacité équivaut au théorème suivant : tout ensemble est contradictoire si et seulement s'il est finiment contradictoire.

1.3 Liens entre syntaxe et sémantique

Nous avons défini, dans notre syntaxe, le connecteur $\to$, qui se traduit en général par « $P \to Q$ signifie que Q est une conséquence logique de P ». Cependant, ce connecteur ne possède pas de sens en lui-même : il n'a de sens qu'à travers notre règle d'induction pour la définition de ν^*, car c'est bien le comportement des connecteurs vis à vis des valuations qui détermine le sens de ces connecteurs (plus précisément : le sens est donné par les valuations, c'est pour cela qu'on considère que celles-ci sont un aspect sémantique de notre logique, là où les suites de symboles que sont les propositions sont purement syntaxiques). Cependant, de par la forme des propositions, on peut établir un système qui, sans utiliser de valuations, va pouvoir exprimer des notions de conséquence (entre autres choses).

Définition 1.3.1 (Conséquence sémantique)**.** *Soit $\mathbf{P} \subseteq \mathcal{P}$ un ensemble de proposition et P une proposition, on dit que P est conséquence sémantique de $\mathbf{P}$ si pour toute valuation $\nu \models \mathbf{P}$, on a $\nu \models P$.*

Exemple 1.3.1. Il est évident que si $P \in \mathbf{P}$, alors P est conséquence sémantique de $\mathbf{P}$. On peut aussi montrer que si $P \in \mathbf{P}$ et $Q \in \mathbf{P}$ alors $P \wedge Q$ est conséquence sémantique de $\mathbf{P}$.

Exercice 1.3.1. Montrer qu'une formulation équivalente du théorème de compacité est la suivante : une proposition P est conséquence sémantique d'un ensemble de propositions $\mathbf{P}$ si et seulement si elle est conséquence sémantique d'une partie finie de $\mathbf{P}$.

1.3.1 Déduction naturelle avec présentation en séquents

A partir de nos exemples se dessine un schéma de règles assez naturel : pour un ensemble de propositions, on peut directement dériver des propositions qui en sont des conséquences sémantiques, sans avoir à raisonner réellement sur les valuations. Nous allons donc définir un système syntaxique appelé déduction naturelle permettant de formaliser cette idée.

Définition 1.3.2 (Séquent en déduction naturelle)**.** *Soit Γ un ensemble de propositions (présenté sous forme de liste) et P une propositions, on note $\Gamma \vdash P$, et on lit « gamma thèse P » le séquent exprimant que P peut se déduire des hypothèses Γ. Ceci constitue une relation inductive dont les règles de constructions sont précisées dans la figure 1.3 ci-dessous.*

Remarque 1.3.1. On appelle un arbre de preuve une dérivation, à partir de ces règles, d'un séquent.

Si $\Gamma = \varnothing$, on notera directement $\vdash P$.

Exemple 1.3.2. Voici un exemple de dérivation de $P \to P$:

$$\dfrac{\dfrac{}{P \vdash P}\,\mathrm{Ax}}{\vdash P \to P}\,{\to_{\mathrm{i}}}$$

Exercice 1.3.2. Pour chaque proposition, construire un arbre de preuve :

- $\vdash P \to Q \to P$
- $\vdash (P \to Q \to R) \to (P \to Q) \to P \to R$
- $(P \vee Q) \wedge R \vdash (P \wedge R) \vee (Q \wedge R)$

$$\frac{}{\Gamma \vdash \top}\ \top_{\mathrm{i}} \qquad \frac{}{\Gamma, P \vdash P}\ \mathrm{Ax} \qquad \frac{\Gamma, P \vdash Q}{\Gamma \vdash P \to Q}\ \to_{\mathrm{i}} \qquad \frac{\Gamma \vdash P \to Q \quad \Gamma \vdash P}{\Gamma \vdash Q}\ \to_{\mathrm{e}}$$

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q}\ \wedge_{\mathrm{i}} \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P}\ \wedge_{\mathrm{e}}^{\mathrm{g}} \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q}\ \wedge_{\mathrm{e}}^{\mathrm{d}}$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q}\ \vee_{\mathrm{i}}^{\mathrm{g}} \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q}\ \vee_{\mathrm{i}}^{\mathrm{d}} \qquad \frac{\Gamma \vdash P \vee Q \quad \Gamma, P \vdash C \quad \Gamma, Q \vdash C}{\Gamma \vdash C}\ \vee_{\mathrm{e}}$$

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \neg_{\mathrm{i}} \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash \bot}\ \neg_{\mathrm{e}} \qquad \frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ \bot_{\mathrm{c}}$$

Figure 1.3 – Règles de la déduction naturelle

- $\neg\neg A \to A$
- $\neg A \vee A$

Exercice 1.3.3 (Principe d'explosion)**.** Une règle importante en logique, nommée *ex falso quodlibet*, stipule que si l'on prouve $\bot$ (la formule fausse) alors on peut prouver n'importe quoi, elle s'écrit

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot_{\mathrm{i}}$$

Montrer que $\bot_{\mathrm{i}}$ est dérivable, c'est-à-dire que l'on peut construire un arbre partant de la prémisse de la règle pour arriver à sa conclusion (la lettre i pour désigner cette règle fait référence à l'intuitionnisme, car cette règle remplace $\bot_{\mathrm{c}}$, l'absurde classique, en logique intuitionniste).

Nous pouvons remarquer que chaque règle d'induction correspond à une règle logique évidente :

- La règle $\top_{\mathrm{i}}$ dit simplement que l'on peut toujours déduire la formule vraie.
- La règle Ax dit que l'on peut déduire P directement si l'on a fait l'hypothèse P.
- La règle $\to_{\mathrm{i}}$ dit que pour montrer $P \to Q$ il suffit de supposer P pour montrer Q.
- La règle $\to_{\mathrm{e}}$, aussi appelée *modus ponens*, dit que si l'on a prouvé $P \to Q$ et P, alors on peut en déduire Q.
- Les règles liées à $\vee$, $\wedge$ et $\neg$ sont assez lisibles, à l'exception de $\vee_{\mathrm{e}}$: elle correspond à la disjonction de cas. En effet, si l'on a prouvé $P \vee Q$, alors prouver C revient à le prouver sous l'hypothèse P et sous l'hypothèse Q.
- La règle $\bot_{\mathrm{c}}$, appelée raisonnement par l'absurde, permet de déduire A du fait que $\neg A$ mène à une contradiction.

L'exercice suivant est un résultat de structure de notre système, qui sera utile par la suite :

Exercice 1.3.4 (Affaiblissement)**.** Montrer par induction que s'il existe une dérivation de $\Gamma \vdash P$ alors pour toute proposition A, il existe une dérivation de $\Gamma, A \vdash P$ (ceci signifie que si l'on a prouvé un résultat, on peut encore le prouver en rajoutant des hypothèses).

Exercice 1.3.5 (Codage de la négation)**.** Montrer que le séquent suivant est dérivable :

$$\vdash (\neg P \to (P \to \bot)) \wedge ((P \to \bot) \to \neg P)$$

1.3.2 Correction et complétude du calcul propositionnel

Ces règles paraissent dont très convaincantes pour créer un formalisme robuste de raisonnement. Pourtant, ce ne sont que des règles formelles : des enchaînements de symboles qui n'ont *a priori* pas de lien avec la sémantique de nos propositions. Heureusement, il existe une paire de théorème nous permettant d'établir l'équivalence entre la conséquence sémantique et la conséquence syntaxique (c'est-à-dire le fait de prouver $\Gamma \vdash P$).

Théorème 1.3.1 (Correction du calcul propositionnel)**.** *Si $\Gamma \vdash P$ pour Γ un ensemble de propositions, alors P est conséquence sémantique de Γ, i.e. pour toute valuation ν, si $\nu \models \Gamma$ alors $\nu \models P$.*

Démonstration. La preuve se fait par induction sur $\Gamma \vdash P$, il suffit donc de montrer que pour chaque règle, si le théorème est vrai pour ses prémisses, alors il est vrai pour sa conclusion. Nous laissons en exercice au lecteur le traitement de la plupart des règles, mais donnerons le traitement de deux règles pour l'exemple :

- Cas de $\to_{\mathrm{e}}$: supposons que $\Gamma \vdash P \to Q$ et $\Gamma \vdash P$, et de plus que cela $P \to Q$ et P sont conséquences sémantiques de Γ (par hypothèse d'induction), montrons qu'alors Q est conséquence sémantique de Γ. Soit ν satisfaisant Γ, alors par hypothèse, $\nu(P) = 1$ et $\nu(P \to Q) = 1$, or $\nu(P \to Q) = 1 - \nu(P) + \nu(P)\nu(Q)$ donc en simplifiant, on en déduit que $\nu(Q) = 1$: donc Q est conséquence sémantique de Γ.
- Cas de $\vee_{\mathrm{e}}$: on suppose donc que $P \vee Q$ est conséquence sémantique de Γ, que C est conséquence sémantique de Γ, P et de Γ, Q, et on veut montrer que C est conséquence sémantique de Γ. Soit ν satisfaisant Γ, alors $\nu(P \vee Q) = 1$, c'est-à-dire $\nu(P) + \nu(Q) - \nu(P)\nu(Q) = 1$. Alors soit $\nu(P) = 1$, soit $\nu(Q) = 1$ (possiblement les deux). Supposons sans perte de généralité que $\nu(P) = 1$: cela signifie que ν satisfait Γ, P, donc comme C est conséquence logique de Γ, P, $\nu(C) = 1$. Donc C est conséquence logique de Γ.

□

Exercice 1.3.6. Montrer le théorème pour les autre règles d'induction.

Ce théorème nous dit donc que ce qu'on prouve avec ces règles est correct. La question naturelle qui s'ensuit est « peut-on prouver tout ce qui est vrai ? » Il se trouve que c'est le cas, c'est ce que montre le prochain théorème.

Pour préparer ce résultat, nous aurons besoin d'une suite de lemmes (le premier sera ici donné en exercice). Ces lemmes ont des preuves principalement techniques et il est conseillé pour le lecteur de ne pas s'attarder sur celles-ci.

Exercice 1.3.7. Montrer que P est conséquence sémantique d'un ensemble fini de propositions Γ si et seulement si $\left(\bigwedge_{\gamma \in \Gamma} \gamma\right) \to P$ est une tautologie. Où cette notation signifie que l'on considère la conjonction (connecteur « et ») sur toutes les formules de Γ.

Nous allons construire, pour une valuation ν, une proposition permettant de représenter, en quelque sorte, ν. La preuve se basera sur le fait que $\nu \models P$ pourra s'interpréter comme $\vdash \nu \to P$.

Lemme 1.3.1. *Soit P une proposition et ν une valuation sur les variables de P, on note $V(\nu)$ une proposition correspondant à une conjonction contenant toutes les propositions x_i telles que $\nu(x_i) = 1$ et les négations des propositions x_j telles que $\nu(x_j) = 0$. Par exemple pour $\nu(x_0) = 1, \nu(x_1) = 0$ on a au moins $V(\nu) = x_0 \wedge \neg x_1$. Alors :*

- *soit $\nu(P) = 1$ et alors $V(\nu) \vdash P$*
- *soit $\nu(P) = 0$ et alors $V(\nu) \vdash \neg P$.*

Démonstration. Procédons par induction sur P :

- Les cas $\top$ et $\bot$ sont évidents, puisqu'ils ne font pas intervenir de variables.
- Cas x_i : soit $V(\nu)$ avec ν une valuation sur x_i. Supposons que $\nu(x_i) = 1$. Alors

$$\dfrac{\dfrac{}{V(\nu) \vdash V(\nu)}\,\mathrm{Ax}}{V(\nu) \vdash x_i}\,\wedge_{\mathrm{e}}$$

où l'on note $\wedge_{\mathrm{e}}$ pour choisir le x_i précis dans $V(\nu)$ (on peut se convaincre qu'une suite finie d'applications de $\wedge_{\mathrm{e}}$ permet de faire ceci). La preuve est la même si $\nu(x_i) = 0$ (on obtiendra simplement $\neg x_i$ à la place).

- Cas $\vee$: soit ν une valuation contenant les variables de P et de Q, et $V(\nu)$ une proposition associée à ν. Quatre cas peuvent alors arriver, mais nous ne traiterons que deux d'entre eux (le lectuer assidu vérifiera les autres) : si $\nu(P) = 1$ et $\nu(Q) = 0$ (et donc que $\nu(P \vee Q) = 1$) alors on sait que $V(\nu) \vdash P$ et $V(\nu) \vdash \neg Q$. On construit alors :

$$\dfrac{V(\nu) \vdash P}{V(\nu) \vdash P \vee Q}\,\vee_{\mathrm{i}}$$

Si $\nu(P) = \nu(Q) = 0$ alors $V(\nu) \vdash \neg P$, $V(\nu) \vdash \neg Q$ et on veut montrer que $V(\nu) \vdash \neg(P \vee Q)$:

$$\dfrac{\dfrac{\dfrac{}{V(\nu), P \vee Q \vdash P \vee Q}\,\mathrm{Ax} \quad \dfrac{V(\nu), P \vee Q, P \vdash \neg P \quad \dfrac{}{V(\nu), P \vee Q, P \vdash P}\,\mathrm{Ax}}{V(\nu), P \vee Q, P \vdash \bot}\,\neg_{\mathrm{e}} \quad \dfrac{V(\nu), P \vee Q, Q \vdash \neg Q \quad \dfrac{}{V(\nu), P \vee Q, Q \vdash Q}\,\mathrm{Ax}}{V(\nu), P \vee Q, Q \vdash \bot}\,\neg_{\mathrm{e}}}{V(\nu), P \vee Q \vdash \bot}\,\vee_{\mathrm{e}}}{V(\nu) \vdash \neg(P \vee Q)}\,\neg_{\mathrm{i}}$$

- Cas $\wedge$: soit ν une valuation contenant les variables de P et de Q et $V(\nu)$ une proposition associée à ν. Là encore, quatre cas sont possibles. Si $\nu(P) = \nu(Q) = 0$ alors par hypothèse d'induction $V(\nu) \vdash \neg P$, $V(\nu) \vdash \neg Q$, et alors :

$$\dfrac{\dfrac{V(\nu), P \wedge Q \vdash \neg P \qquad \dfrac{\dfrac{}{V(\nu), P \wedge Q \vdash P \wedge Q}\,\mathrm{Ax}}{V(\nu), P \wedge Q \vdash P}\,\wedge_{\mathrm{e}}^{\mathrm{g}}}{V(\nu), P \wedge Q \vdash \bot}\,\neg_{\mathrm{e}}}{V(\nu) \vdash \neg(P \wedge Q)}\,\neg_{\mathrm{i}}$$

Si $\nu(P) = 0$ et $\nu(Q) = 1$, remarquons que l'arbre plus haut est encore parfaitement utilisable puisque $\nu(P \wedge Q) = 0$. De même en inversant les rôles de P et Q, on peut prouver le cas où $\nu(P) = 1$ et $\nu(Q) = 0$. Enfin, si $\nu(P) = \nu(Q) = 1$:

$$\dfrac{V(\nu) \vdash P \qquad V(\nu) \vdash Q}{V(\nu) \vdash P \wedge Q}\,\wedge_{\mathrm{i}}$$

- Cas $\neg$: soit ν une valuation contenant les variables de P, et $V(\nu)$ une proposition associée. Si $\nu(P) = 1$ alors $\nu(\neg(P)) = 0$, on veut donc montrer qu'à partir de $V(\nu) \vdash P$ on peut déduire $V(\nu) \vdash \neg\neg P$:

$$\dfrac{\dfrac{V(\nu), \neg P \vdash P \qquad \dfrac{}{V(\nu), \neg P \vdash \neg P}\,\mathrm{Ax}}{V(\nu), \neg P \vdash \bot}\,\neg_{\mathrm{e}}}{V(\nu) \vdash \neg\neg P}\,\neg_{\mathrm{i}}$$

Si $\nu(P) = 0$, alors on veut montrer qu'à partir de $V(\nu) \vdash \neg P$ on peut en déduire que $V(\nu) \vdash \neg P$, ce qui est évident.

- Cas $\rightarrow$: soit ν une valuation contenant les variables de P et Q, et $V(\nu)$ une proposition associée. Il reste encore quatre cas à traiter. Si $\nu(P) = \nu(Q) = 0$, alors on veut montrer que $V(\nu) \vdash P \rightarrow Q$ à partir de $V(\nu) \vdash \neg P$ et $V(\nu) \vdash \neg Q$:

$$\dfrac{\dfrac{\dfrac{V(\nu), P \vdash \neg P \qquad \dfrac{}{V(\nu), P \vdash P}\,\mathrm{Ax}}{V(\nu), P \vdash \bot}\,\neg_{\mathrm{e}}}{V(\nu), P \vdash Q}\,\bot_{\mathrm{i}}}{V(\nu) \vdash P \rightarrow Q}\,\rightarrow_{\mathrm{i}}$$

Remarquons que nous n'avons, là encore, pas eu besoin d'utiliser l'hypothèse sur Q. On peut donc traiter avec le même arbre le cas où $\nu(P) = 0$ et $\nu(Q) = 1$. Si $\nu(P) = 1$ et $\nu(Q) = 0$, il nous faut prouver $V(\nu) \vdash \neg(P \to Q)$ avec les hypothèses $V(\nu) \vdash P$ et $V(\nu) \vdash \neg Q$:

$$\dfrac{\dfrac{V(\nu), P \to Q \vdash \neg Q \qquad \dfrac{\dfrac{}{V(\nu), P \to Q \vdash P \to Q}\,\text{Ax} \qquad V(\nu), P \to Q \vdash P}{V(\nu), P \to Q \vdash Q}\,\to_{\text{e}}}{V(\nu), P \to Q \vdash \bot}\,\neg_{\text{e}}}{V(\nu) \vdash \neg(P \to Q)}\,\neg_{\text{i}}$$

Enfin, si $\nu(P) = \nu(Q) = 1$, alors par $\to_{\text{i}}$ on déduit de $V(\nu), P \vdash Q$ que $V(\nu) \vdash P \to Q$.

□

Lemme 1.3.2. *Soit P une proposition. Si P est une tautologie, alors $\vdash P$.*

Démonstration. Procédons par récurrence sur l'indice maximal des variables présentes dans P :

- Si la seule variable apparaissant dans P est x_0, alors comme P est une tautologie et par le lemme précédent, on sait que $x_0 \vdash P$ et $\neg x_0 \vdash P$, ce qui nous permet de déduire que $\vdash P$ (en utilisant un exercice précédent où l'on a montré que $\vdash A \vee \neg A$) :

$$\dfrac{\vdash x_0 \vee \neg x_0 \qquad x_0 \vdash P \qquad \neg x_0 \vdash P}{\vdash P}\,\vee_{\text{e}}$$

- Supposons que pour toute proposition faisant intervenir des variables jusqu'à i, être une tautologie implique d'être dérivable dans notre système. Soit alors une proposition P faisant intervenir des variables jusqu'à $i+1$. Soit ν une valuation sur les i premières variables. Comme P est une tautologie, alors P est satisfaite par ν' valant ν jusqu'à x_i et 1 en x_{i+1}, et par ν'' valant ν jusqu'à x_i et 0 en x_{i+1}. On en déduit donc que $V(\nu') \vdash P$ et $V(\nu'') \vdash P$, soit $V(\nu) \wedge x_{i+1} \vdash P$ et $V(\nu) \wedge \neg x_{i+1} \vdash P$. Par introductions successives de l'implication, on a alors $x_{i+1} \vdash V(\nu) \to P$ et $\neg x_{i+1} \vdash V(\nu) \to P$, nous ramenant (en utilisant le même raisonnement que le cas précédent) à $\vdash V(\nu) \to P$, ce qui revient à $V(\nu) \vdash P$ qui est dérivable par le lemme précédent.

Donc par induction, $\vdash P$. □

Remarque 1.3.2. Nous avons ici utilisé que si $P \wedge Q \vdash R$ alors $Q \vdash P \to R$. Ceci n'est pas une règle de à proprement parler, mais on peut prouver ce résultat par induction sur les arbres de preuve. Cela peut faire un exercice, mais il est surtout fastidieux et l'idée est principalement que l'on peut remplacer $P \wedge Q \vdash R$ par $P, Q \vdash R$ en remplaçant chaque utilisation de l'axiome pour P ou pour Q par un axiome pour $P \wedge Q$ puis une élimination du $\wedge$. Nous pouvons le statuer comme une sorte de meta-règle (qui nous permet de dire qu'un séquent est dérivable à partir d'un autre séquent qui est dérivable), qu'on appelle une règle admissible.

Théorème 1.3.2 (Complétude du calcul propositionnel)**.** *Soit Γ un ensemble de propositions et P une proposition. Si P est conséquence sémantique de Γ, i.e. pour toute valuation ν, si $\nu \models \Gamma$ alors $\nu \models P$, alors $\Gamma \vdash P$.*

Démonstration. Sous nos hypothèses, avec le théorème de compacité, on peut considérer une partie finie $\Gamma_0 \subseteq \Gamma$ telle que P en est conséquence sémantique. On veut donc prouver que $\Gamma_0 \vdash P$, ce qui revient à montrer que $\vdash \left(\bigwedge_{\gamma \in \Gamma_0} \gamma\right) \to P$, et la condition sémantique est équivalente à dire que $\left(\bigwedge_{\gamma \in \Gamma_0} \gamma\right) \to P$ est une tautologie. Nous n'avons donc à montrer que le résultat (plus faible) suivant : si P est une tautologie, alors $\vdash P$ est dérivable. C'est justement le résultat du lemme précédent. □

Exercice 1.3.8 (Énoncé équivalent du théorème de complétude)**.** Montrer que le théorème de complétude revient à l'énoncé suivant : si Γ est contradictoire, alors $\Gamma \vdash \bot$.

Nous savons maintenant qu'avec notre système de calcul, la preuve syntaxique et la correspondance sémantique sont équivalentes. Ainsi, être conséquence sémantique ou syntaxique correspond à la même chose, d'où la définition suivante :

Définition 1.3.3 (Conséquence, équivalence)**.** *On dit qu'une proposition Q est conséquence d'une proposition P si l'un des conditions équivalentes est vérifiées : $P \vdash Q$; Q est conséquence sémantique de P ; $\vdash P \to Q$; $P \to Q$ est une tautologie. On notera parfois cela $P \implies Q$ (mais on évitera en général de le faire pour ne pas confondre avec $\to$, qui est un connecteur logique, là où $\implies$ représente donc une méta-proposition, une propriété sur nos propositions elles-mêmes).*

On dit que deux propositions P et Q sont équivalentes lorsque P est conséquence de Q et Q est conséquence de P. On notera alors $P \iff Q$ ou $P \equiv Q$ (la deuxième notation sera privilégiée).

1.4 Algèbres de Boole

Le calcul propositionnel nous a donné un formalisme pour parler de propositions, de vérité et de preuves. Nous avons vu que les éléments essentiels du calcul propositionnels sont les constructeurs (c'est-à-dire $\vee$,$\wedge$,$\to$,$\neg$) des propositions, puisque ce calcul ne s'intéresse qu'au lien entre les propositions. Ceci motive alors l'étude des algèbres de Boole, qui sont des structures se comportant de la même façon : elles possèdent des constructeurs $\vee$,$\wedge$ et $\neg$ (remarquons que $P \to Q \equiv \neg P \vee Q$ et qu'il suffit donc des trois constructeurs précédents). Le formalisme permettant au mieux de généraliser ces constructeurs se trouve dans la théorie des treillis (une branche de la théorie des ensembles ordonnés). Nous verrons donc d'abord des rappels de théorie des ordres, pour définir ensuite les algèbres de Boole. Nous ferons finalement le lien avec le calcul propositionnel par le biais de l'algèbre de Lindenbaum-Tarski.

1.4.1 Un peu de théorie des ensembles ordonnés

Dans la suite de cette section, nous fixerons $(X, \preceq)$ un ensemble ordonné. Rappelons qu'un ensemble ordonné est un ensemble (ici X) muni d'une relation $\preceq$ vérifiant :

- pour tout élément $x \in X$, $x \preceq x$
- pour tous éléments $x, y \in X$, si $x \preceq y$ et $y \preceq x$ alors $x = y$
- pour tous éléments $x, y, z \in X$, si $x \preceq y$ et $y \preceq z$ alors $x \preceq z$.

On dit que x est inférieur à y si $x \preceq y$.

Exemple 1.4.1. Si les exemples d'ensembles ordonnés ne manquent pas, l'exemple canonique que nous utiliserons ici est, pour un ensemble E fixé, l'ensemble ordonné $(\mathcal{P}(E), \subseteq)$ des parties de E ordonnées par l'inclusion. Remarquons déjà que les opérations ensemblistes $\cup$ et $\cap$ ont une portée logique assez forte : pour une partie A et une partie B, on peut écrire $A \cup B = \{x \in E \mid x \in A \vee x \in B\}$. De même, $E \setminus A$ peut s'écrire $\{x \in E \mid \neg(x \in A)\}$. Nous verrons au long de cette section que les opérations ensemblistes revêtent un aspect logique similaire au calcul propositionnel.

Un point essentiel de l'étude des ensembles ordonnés est la notion de majorant et de minorant :

Définition 1.4.1 (Majorant, minorant)**.** *Soit Y une partie de X, on dit que M est un majorant de Y si tout élément de Y est inférieur à M. On dit que m est un minorant de Y si tout élément de Y est supérieur à m.*

Remarque 1.4.1. De par ces premières définitions, on s'aperçoit déjà d'un phénomène très important en théorie des ensembles ordonnés, qu'on appelle la dualité : chaque notion peut s'énoncer de deux façons différentes suivant si l'on considère la relation $\preceq$ (« inférieur à ») ou $\succeq$ (« supérieur à »). En effet, la relation $\succeq$ définie par $x \succeq$

$y \iff y \preceq x$ inverse les minorants, les majorants, et toutes les notions d'ordre que nous verrons plus tard. C'est pour cela que nous donnerons en général deux définitions mais une seule preuve : l'autre se déduit en reprenant la preuve dans l'ordre opposé.

Exemple 1.4.2. Dans notre cas, une « partie de X » correspond à un ensemble de parties de E, c'est-à-dire une partie de $\mathcal{P}(E)$. Remarquons d'abord que si $F \subseteq \mathcal{P}(E)$ alors $\varnothing$ et E sont respectivement un minorant et un majorant de F. Un autre minorant de F est $\bigcap F$, c'est-à-dire l'intersection de toutes les parties de E appartenant à F.

A partir de ces exemples, on voit qu'il existe de meilleurs minorants que d'autres : si $\varnothing$ est minorant de toute partie de $\mathcal{P}(E)$, l'intersection nous donne un minorant que l'on pourrait dire « taillé sur mesure ». En fait, il est même le meilleur minorant, au sens suivant :

Définition 1.4.2 (Borne supérieure, borne inférieure)**.** *Soit F une partie de X. La borne supérieure de F, notée $\sup_F$ ou $\sup(F)$, si elle existe, est le plus petit majorant de F : c'est le majorant de F tel que pour tout M majorant de F, $\sup(F) \preceq M$.*

La borne inférieure, elle, est le plus grand minorant et se note $\inf_F$ ou $\inf(F)$. C'est donc le minorant de F tel que pour tout autre minorant m de F, on ait $m \preceq \inf(F)$.

Démonstration. Puisque l'on a dit « le », c'est pour indiquer qu'il est unique (sous réserve d'existence) : prouvons-le. Supposons que s, s' soient deux borne supérieures de F. Alors s est un majorant de F, donc par définition du fait que s' est borne supérieure, $s' \preceq s$. De même, s' est majorant de F donc $s \preceq s'$. Il en résulte par double inégalité que $s = s'$: la borne supérieure est unique si elle existe. □

Ce problème de l'existence d'une borne supérieure semble ne pas exister dans notre exemple canonique, d'après l'exercice suivant :

Exercice 1.4.1. Soit E un ensemble, et $(\mathcal{P}(E), \subseteq)$ l'ensemble ordonné de ses parties. Montrer que pour tout ensemble F de parties de E, la borne supérieure de F est $\bigcup F$ et sa borne inférieure est $\bigcap F$.

Pourtant, il existe bien des cas où il n'existe pas de borne supérieure, par exemple en considérant l'ordre donné par le diagramme de la figure 1.4.1 (une flèches $x \to y$ représente le fait que $x \preceq y$) et la partie $\{x_1, x_2\}$.

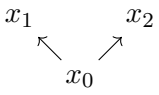


Figure 1.4 – Ordre sans borne supérieure

Comme nous l'avons dit plus tôt, notre candidat pour l'opération $\vee$ par exemple est $\cup$, ce qui signifie que nous voudrions avoir, pour toute partie, une borne supérieure. Cependant, cette exigence est peut-être un peu trop grande, puisque $\vee$ est une opération binaire (que l'on peut étendre facilement en une opération n-aire) : nous souhaitons donc principalement avoir des bornes supérieures et inférieures pour les parties finies.

1.4.2 Vers les algèbres de Boole

Définition 1.4.3 (Treillis)**.** *Un ensemble ordonné est un treillis si pour toute partie finie $F \subseteq_{\text{fin}} X$, il existe une borne supérieure et une borne inférieure à F, que nous noterons respectivement $\bigvee F$ et $\bigwedge F$.*

Remarque 1.4.2. Remarquons que $\varnothing$ est une partie finie de X. Par convention, si $F = \varnothing$, alors $\bigvee F = \bot$ et $\bigwedge F = \top$ où $\top$ et $\bot$ sont respectivement le majorant et le minorant de X. Un treillis est donc un ensemble borné, c'est-à-dire un ensemble avec un majorant et un minorant.

Exercice 1.4.2. Montrer que ces conditions sont équivalentes à l'existence :

- d'un majorat $\top$
- d'un minorant $\bot$
- d'une opération binaire $\vee$ telle que pour tous éléments x, y, $x \preceq x \vee y$, $y \preceq x \vee y$ et pour tout z tel que $x \preceq z$ et $y \preceq z$, $x \vee y \preceq z$
- d'une opération binaire $\wedge$ telle que pour tous élément x, y, $x \wedge y \preceq x$, $x \wedge y \preceq y$ et pour tout z tel que $z \preceq x$ et $z \preceq y$, $z \preceq x \wedge y$

Nous utiliserons en général la version binaire $\vee$ et $\wedge$ plutôt que des bornes sur des parties finies, à la fois par souci de lisibilité et car nous avons vu qu'il suffit de travailler avec ces opérations.

Exercice 1.4.3. Montrer que $\top$ est neutre pour $\wedge$, c'est-à-dire que $x \wedge \top = x$ pour tout x. De même, $\bot$ est neutre pour $\vee$ (par dualité).

Exercice 1.4.4. Montrer que $\wedge$ est associatif, c'est-à-dire que $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ (on écrira alors directement $x \wedge y \wedge z$) et commutatif, c'est-à-dire que $x \wedge y = y \wedge x$. De même $\vee$ est associatif et commutatif par dualité.

Exercice 1.4.5. Montrer que pour tout x, y, les propositions suivantes sont équivalentes :

- $x \preceq y$
- $x \wedge y = x$
- $x \vee y = y$

Exemple 1.4.3. On remarque directement qu'en prenant $\wedge = \cap$ et $\vee = \cup$ dans notre exemple canonique, notre ensemble de parties est bien un treillis. Un fait supplémentaire est qu'il existe une distributivité : $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

En fait, cette distributivité ne peut se déduire directement du fait d'être un treillis, c'est pour cela qu'on définit les treillis distributifs :

Définition 1.4.4 (Treillis distributif)**.** *Un treillis est dit distributif si pour tous éléments x, y, z l'une des deux conditions équivalentes est vérifiées :*

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

Pour prouver notre équivalence, nous allons introduire un lemme :

Lemme 1.4.1. *Pour tous x, y, $x \vee (y \wedge x) = x$ et $x \wedge (x \vee y) = x$.*

Démonstration. Nous ne prouverons que la première égalité, l'autre suivant par dualité :

- d'abord, par définition de $\vee$, $x \preceq x \vee (y \wedge x)$
- par réflexivity, $x \preceq x$, et par définition de $\wedge$, $y \wedge x \preceq x$, il s'ensuit que $x \vee (y \wedge x) \preceq x$

Par double inégalité, notre égalité est vérifiée. □

Nous pouvons maintenant procéder à la preuve de l'équivalence des distributivités :

Démonstration. Là encore, la dualité nous permet de ne montrer que l'une des deux implications, nous supposerons donc la première égalité pour prouver la deuxième :

$$\begin{aligned} x \vee (y \wedge z) &= (x \vee (y \wedge x)) \vee (y \wedge z) \\ &= x \vee [(y \wedge x) \vee (y \wedge z)] \\ &= x \vee [y \wedge (x \vee z)] \\ &= [x \wedge (x \vee z)] \vee [y \wedge (x \vee z)] \\ x \vee (y \wedge z) &= (x \vee y) \wedge (x \vee z) \end{aligned}$$

□

Notre exemple canonique est évidemment un treillis distributif. Il nous reste à définir la négation dans un tel treillis, et nous aurons alors défini la notion d'algèbre de Boole. On base cette définition sur la remarque suivante : dans un ensemble E, pour une partie $A \subseteq E$ on a $B = \complement_E A$ si et seulement si $B \cup A = E$ et $B \cap A = \varnothing$.

Définition 1.4.5 (Complément)**.** *Soit x un élément d'un treillis distributif. On dit que x' est le complément de x si $x \vee x' = \top$ et $x \wedge x' = \bot$.*

Démonstration. Nous allons prouver que le complément est unique, sous réserve d'existence. Soit x un élément, et c, c' deux compléments de x. Alors :

$$\begin{aligned} c &= c \wedge \top \\ &= c \wedge (x \vee c') \\ &= (c \wedge x) \vee (c \wedge c') \\ &= \bot \vee (c \wedge c') \\ c &= c \wedge c' \end{aligned}$$

On en déduit que $c \preceq c'$. Comme c et c' ont des rôles symétriques, la même suite d'égalités peut se faire pour prouver que $c' \preceq c$, donc $c = c'$. □

On notera alors $\neg x$ le complément de x, s'il existe. La définition d'algèbre de Boole peut enfin être donnée :

Définition 1.4.6 (Algèbre de Boole)**.** *Une algèbre de Boole, ou algèbre booléenne, est un treillis distributif complémenté, i.e. tel que tout élément possède un complément.*

Exemple 1.4.4. On retrouve donc que $(\mathcal{P}(E), \subseteq)$ est une algèbre de Boole, avec l'intersection et l'union comme opérations binaires, $\varnothing$ et E comme minorant et majorant, et $E \setminus X$ comme complémentaire d'une partie X.

Remarque 1.4.3. Il existe une algèbre de Boole triviale, qui est l'ensemble $\{\top, \bot\}$, assimilable à $\mathcal{P}(\{\varnothing\})$, ne contenant que deux éléments et dont les définitions des opérations sont évidentes.

Exercice 1.4.6. Montrer que dans une algèbre de Boole, les loi de De Morgan sont vérifiées :

$$\neg(x \vee y) = \neg x \wedge \neg y \qquad \neg(x \wedge y) = \neg x \vee \neg y$$

et que la loi de la double négation est vérifiée aussi :

$$\neg\neg x = x$$

Nous pouvons aussi définir des morphismes d'algèbres de Boole, c'est-à-dire des fonctions entre algèbres de Boole qui transmettent l'information d'une algèbre de Boole :

Définition 1.4.7 (Morphisme d'algèbre de Boole)**.** *Si* $(X, \preceq, \vee, \wedge, \neg)$ *et* $(Y, \leq, \sqcup, \sqcap, \sim)$ *sont deux algèbre de Boole, on appelle morphisme d'algèbre de Boole un morphisme de treillis, c'est-à-dire une fonction* $f : X \to Y$ *telle que pour tout* $x, y \in X$*, on ait* $f(x \vee y) = f(x) \sqcup f(y)$ *et* $f(x \wedge y) = f(x) \sqcap f(y)$ *ainsi que* $f(\top_X) = \top_Y$ *et* $f(\bot_X) = \bot_Y$*.*

Exercice 1.4.7. Montrer que cette définition suffit à montrer que $f(\neg x) = \sim f(x)$.

Exercice 1.4.8 (Les algèbres de Boole comme catégorie)**.** Montrer que l'identité sur une algèbre de Boole est un morphisme d'algèbre de Boole et que la composée de deux morphisme d'algèbres de Boole est encore un morphisme d'algèbre de Boole.

Définition 1.4.8. *On dit que deux algèbre de Boole* $\mathcal{B}$ *et* $\mathcal{B}'$ *sont isomorphes s'il existe deux morphismes d'algèbres de Boole* $f : \mathcal{B} \to \mathcal{B}'$ *et* $g : \mathcal{B}' \to \mathcal{B}$ *telles que* $f \circ g = \mathrm{id}_{\mathcal{B}'}$ *et* $g \circ f = \mathrm{id}_{\mathcal{B}}$

1.4.3 Algèbre de Lindenbaum-Tarski

Comme le comportement d'une algèbre de Boole est similaire à celui du calcul propositionnel (en remplaçant l'équivalence de propositions par l'égalité), nous allons construire une algèbre de Boole associée à notre calcul propositionnel.

Définition 1.4.9 (Algèbre de Lindenbaum-Tarski)**.** *Soit* $\mathcal{P}$ *l'ensemble des propositions, on définit* $\overline{\mathcal{P}} = \mathcal{P}/_{\equiv}$ *c'est-à-dire l'ensemble des propositions quotienté par la relation d'équivalence sémantique (ou syntaxique, les deux coïncidant), cela signifie que si* $P \equiv Q$ *alors on considère que* $P = Q$*. On munit* $\overline{\mathcal{P}}$ *de la relation d'ordre de conséquence sémantique, ou de façon équivalente de la relation* $P \vdash Q$*. On a alors une algèbre de Boole, notée* $\mathcal{L}(\mathcal{P})$*, en prenant comme opérations* $\vee$*,*$\wedge$ *et* $\neg$*.*

Exercice 1.4.9. Vérifier que cette définition donne bien une algèbre de Boole.

Chapitre 2

Calcul des prédicats

Là où le calcul propositionnel étudiait les propositions uniquement par leurs liens logiques, le calcul des prédicats établit une étude plus fine en considérant que les propositions peuvent porter sur des objets mathématiques. C'est à partir de cette étape que notre système propositionnel est suffisamment expressif pour formaliser les mathématiques. Nous allons, comme dans le chapitre précédent, y présenter les règles syntaxiques de la déduction naturelle (il sera nécessaire d'introduire de nouvelles règles puisque nous avons un système étendu) ainsi que la notion de théorie logique, puis nous parlerons de la notion de $\mathcal{L}$-structure, pour pouvoir alors parler de modèle. Nous donnerons ensuite les théorèmes de correction et de complétude de la déduction naturelle, ainsi que celui de compacité. Nous terminerons sur des considérations de théorie des modèles avec le théorème de Lovenheim-Skolem.

2.1 Langage et propositions sur un langage

Comme nous l'avons dit, le calcul des prédicats, plutôt que de simplement porter sur les relations logiques entre des variables, va s'intéresser aux relations entre des objets mathématiques. Les propositions, sont en quelque sorte, l'équivalent mathématiques des phrases pour le langage naturel, et à ce titre, il convient d'abord de se donner un alphabet pour écrire nos phrases. Si nous avons vu les éléments purement connectifs de notre alphabets (les connecteurs logiques), nous allons ici devoir nous munir d'un alphabet permettant de désigner les objets mathématiques eux-mêmes. Cependant, au lieu d'appeler cela un alphabet, on appelle cela une signature. Nous travaillerons dans une logique du premier ordre, ce qui signifie que nos formules auront pour but de se quantifier sur un domaine entier : on pourra formuler une propriété sur tous les objets, mais pas sur une partie seulement des objets (ce serait une quantification au second ordre).

Définition 2.1.1 (Signature)**.** *Une signature Σ est la donnée d'un quadruplet $\langle \mathcal{F}, \mathcal{R}, \alpha_f, \alpha_r \rangle$ où $\mathcal{F}$ est appelé l'ensemble des symboles de fonctions, $\mathcal{R}$ l'ensemble des symboles de relations, $\alpha_f : \mathcal{F} \to \mathbb{N}$ et $\alpha_r : \mathcal{R} \to \mathbb{N}$. On supposera, sauf mention explicite du contraire, qu'il existe une relation $=$ appartenant à $\mathcal{R}$ telle que $\alpha_= = 2$ représentant l'égalité.*

Remarque 2.1.1. On parlera souvent, à la place de signature Σ, de langage $\mathcal{L}$, et il arrivera fréquemment de simplement noter la liste des symboles puis d'expliciter leur arité et s'ils correspondent à des fonctions ou à des relations.

Comme leur nom l'indique, l'ensemble des symboles de fonctions représente des fonctions, et la fonction α_f est donc l'arité associée à chaque symbole de fonction (un symbole d'arité nulle est un symbole de constante) et l'ensemble des symboles de relations représente des relations, de nombre d'arguments donné par α_r. Nous allons alors constituer ce que l'on appelle des termes, qui seront la brique atomique des propositions manipulées dans ce formalisme, et qui permettront donc de définir les propositions.

Remarque 2.1.2. Nous parlons ici de propositions, comme dans le chapitre précédent, mais il n'y a pas d'ambiguïté : nous ignorons simplement la notion précédente de propositions (qui nous aiderons dans notre compréhension des propositions de ce chapitre, car elles en sont un modèle simplifié).

Définition 2.1.2 (Expression, terme, proposition)**.** *On se donne un ensemble $\mathcal{V}$ dénombrable de variables (dont nous noterons $x, y, \ldots$ les éléments), et on définit l'ensemble $\mathcal{E}_\Sigma$ des expressions sur notre signature Σ par la BNF suivante :*

$$e, e', \ldots ::= x \mid f(e, e', \ldots)$$

où $f \in \mathcal{F}$ et où le nombre d'arguments de f est $\alpha_f(f)$ (on se convainc à partir de notre construction inductive depuis une BNF que cette construction est cohérente).

On définit alors l'ensemble $\mathcal{T}_\Sigma$ des termes par

$$\mathcal{T}_\Sigma = \{r(e_1, \ldots, e_{\alpha_r(r)}) \mid r \in \mathcal{R}, (e_1, \ldots, e_{\alpha_r(r)}) \in \mathcal{E}^{\alpha_r(r)}\}$$

Enfin, l'ensemble $\mathcal{P}_\Sigma$ des propositions sur une signature Σ est défini par :

$$P, Q ::= t \mid \top \mid \bot \mid P \to Q \mid P \vee Q \mid P \wedge Q \mid \neg P \mid \exists x.P \mid \forall x.P$$

en notant t un élément de $\mathcal{T}_\Sigma$ (et en gardant la convention que x est un élément quelconque de $\mathcal{V}$).

Nous connaissons déjà le sens intuitif des connecteurs logiques, mais nous allons préciser le sens de $\exists$ et $\forall$, appelés quantificateurs :

- la proposition $\forall x.P$ signifie intuitivement que pour n'importe quel objet a, la proposition $P[x/a]$ obtenue en remplaçant les occurrences de x par a, est vraie : on lit donc « pour tout x, P ».
- la proposition $\exists x.P$ signifie intuitivement qu'un certain objet a existe tel que $P[x/a]$ est vraie : on lit donc « il existe x tel que P ».

Exemple 2.1.1 (Langage des groupes)**.** Le langage des groupes est constitué d'une constante (i.e. une fonction sans argument) e, d'une fonction d'arité 2 noté $\cdot$ et d'une fonction d'arité 1 notée $-^{-1}$ ou encore i. Il contient aussi l'égalité. Voici des exemples de propositions dans le langage des groupes :

- $\forall x.x \cdot e = y^{-1}$
- $e^{-1} \cdot e = e$
- $\neg(\forall x.x = x) \vee (\exists y.x^{-1} = e)$

Remarquons que ces propositions, en elles-mêmes, semblent avoir peu de sens. La deuxième proposition est compréhensible, mais qui est y dans la première proposition, et qui est x dans la deuxième propositions ? est-ce que x de $\forall x.$ plus tôt ? Pourtant, il n'est pas dans le même terme. Nous allons donc introduire, pour mieux analyser nos propositions, la notion de variable libre.

Remarque 2.1.3. Lorsqu'on considère une constante comme une fonction sans argument, on peut se demander à quelle réalité cela correspond quand on interprète notre langage. Une fonction d'arité n sera interprétée par une fonction $X^n \to X$ (avec X à préciser, ce sera le sujet d'une section prochaine) et une fonction d'arité 0 sera alors une fonction $1 \to X$ avec 1 un ensemble à un élément connu (souvent noté $*$). Ainsi au lieu d'écrire $c(*)$ pour parler de la constante associée par c à $*$, on parlera directement de la constante c.

Définition 2.1.3 (Variable libre, formule close)**.** *Une variable libre dans une proposition est une variable dont la présence n'est pas précédée par un quantificateur. Moralement, c'est une variable pour laquelle il est nécessaire d'attribuer une valeur pour que la proposition ait un sens. Dans une proposition, on construit l'ensemble inductif* $\mathrm{vl}(P)$ *des variables libres de P de la façon suivant :*

- *Pour une expression e,* $\mathrm{vl}(e)$ *est l'ensemble des variables apparaissant dans e (si on représente e comme*

un arbre, c'est l'ensemble des feuilles qui ne sont pas des constantes).

- *Pour une proposition, on construit* $\mathrm{vl}(P)$ *par induction :*

$$\frac{}{\mathrm{vl}(r(e_1,\dots,e_p)) = \bigcup_{i=1}^{p} \mathrm{vl}(e_i)} \qquad \frac{}{\mathrm{vl}(\top) = \varnothing} \qquad \frac{}{\mathrm{vl}(\bot) = \varnothing} \qquad \frac{\mathrm{vl}(P) = E}{\mathrm{vl}(\neg P) = E}$$

$$\frac{\mathrm{vl}(P) = E \quad \mathrm{vl}(Q) = F}{\mathrm{vl}(P \vee Q) = E \cup F} \qquad \frac{\mathrm{vl}(P) = E \quad \mathrm{vl}(Q) = F}{\mathrm{vl}(P \wedge Q) = E \cup F} \qquad \frac{\mathrm{vl}(P) = E \quad \mathrm{vl}(Q) = F}{\mathrm{vl}(P \to Q) = E \cup F}$$

$$\frac{\mathrm{vl}(P) = E}{\mathrm{vl}(\forall x.P) = E \setminus \{x\}} \qquad \frac{\mathrm{vl}(P) = E}{\mathrm{vl}(\exists x.P) = E \setminus \{x\}}$$

On dit qu'une variable est liée si elle n'est pas libre. On appelle formule close une proposition P *telle que* $\mathrm{vl}(P) = \varnothing$*, c'est-à-dire une formule dont toutes les variables sont liées.*

Remarque 2.1.4. On retrouve cette notion de variable liée par exemple dans les intégrales :

$$\int_a^b f(t)\mathrm{d}t$$

fait bien intervenir la variable t, mais celle-ci est liée (elle peut donc être vue comme interchangeable, de la même façon que dans notre cas dire « pour tout x, $P(x)$ » et « pour tout y, $P(y)$ » reviennent à la même chose).

Exercice 2.1.1. Pour chaque proposition de l'exemple 2.1.1, donner l'ensemble de leurs variables libres (écrire au moins une dérivation au complet parmi les trois cas). Y a-t-il des formules closes ? Si oui, lesquelles ?

2.2 Déduction naturelle et théorie logique

Cette fois-ci, nous aborderons d'abord l'aspect syntaxique (nous ne donnerons pas de définition au sens intrinsèque d'une proposition avant la sous-section suivante, mais nous comptons sur notre intuition préalable pour faire sens des règles syntaxiques). Les règles sont les mêmes pour les connecteurs logiques : il nous suffit juste de donner des règles d'introduction et d'élimination pour les quantificateurs. Pour cela, nous avons besoin d'une définition rigoureuse de substitution (la notation $P[x/a]$ présentée plus bas).

Définition 2.2.1 (Substitution dans une proposition)**.** *On appelle substitution de* x *par* a *dans l'expression* e*, notée* $e[x/a]$*, la fonction inductive suivante :*

- *Cas* x *:* $x[x/a] = a$
- *Cas* y *(où* $y \neq x$*) :* $y[x/a] = y$
- *Cas* $f(e_1,\dots,e_p)$ *:* $f(e_1,\dots,e_p)[x/a] = f(e_1[x/a],\dots,e_p[x/a])$

Enfin, cette définition s'étend inductivement aux propositions :

$$\frac{}{r(e_1,\dots,e_p)[x/a] = r(e_1[x/a],\dots,e_p[x/a])} \qquad \frac{}{\top[x/a] = \top} \qquad \frac{}{\bot[x/a] = \bot}$$

$$\frac{P[x/a] = P' \quad Q[x/a] = Q'}{(P \vee Q)[x/a] = P' \vee Q'} \qquad \frac{P[x/a] = P' \quad Q[x/a] = Q'}{(P \wedge Q)[x/a] = P' \wedge Q'} \qquad \frac{P[x/a] = P' \quad Q[x/a] = Q'}{(P \to Q)[x/a] = P' \to Q'}$$

$$\frac{P[x/a] = P'}{(\exists y.P)[x/a] = \exists y.P'}\ [x \neq y] \qquad \frac{P[x/a] = P'}{(\forall y.P)[x/a] = \forall y.P'}\ [x \neq y]$$

$$\frac{}{(\exists x.P)[x/a] = \exists x.P} \quad \frac{}{(\forall x.P)[x/a] = \forall x.P}$$

Remarque 2.2.1. Le choix des deux dernières règles est simple : puisque la variable x est liée dans une formule quantifiée sur x, elle ne peut se remplacer par une substitution « extérieure ».

Nous pouvons maintenant définir notre déduction naturelle, qui ne change de celle du calcul propositionnel que par les règles sur les quantifications et la règle d'égalité.

Définition 2.2.2 (Déduction naturelle)**.** *Soit* Γ *un ensemble de formules présenté sous forme de liste, on définit la relation inductive, appelée « jugement », par les règles inductives de la figure 2.1*

$$\frac{}{\Gamma \vdash \top}\ \top_{\mathrm{i}} \quad \frac{}{\Gamma, P \vdash P}\ \mathrm{Ax} \quad \frac{\Gamma, P \vdash Q}{\Gamma \vdash P \to Q}\ \to_{\mathrm{i}} \quad \frac{\Gamma \vdash P \to Q \quad \Gamma \vdash P}{\Gamma \vdash Q}\ \to_{\mathrm{e}}$$

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q}\ \wedge_{\mathrm{i}} \quad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P}\ \wedge_{\mathrm{e}}^{\mathrm{g}} \quad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q}\ \wedge_{\mathrm{e}}^{\mathrm{d}}$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q}\ \vee_{\mathrm{i}}^{\mathrm{g}} \quad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q}\ \vee_{\mathrm{i}}^{\mathrm{d}} \quad \frac{\Gamma \vdash P \vee Q \quad \Gamma, P \vdash C \quad \Gamma, Q \vdash C}{\Gamma \vdash C}\ \vee_{\mathrm{e}}$$

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \neg_{\mathrm{i}} \quad \frac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash \bot}\ \neg_{\mathrm{e}} \quad \frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ \bot_{\mathrm{c}}$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash \forall x.P}\ [x \notin \mathrm{vl}(\Gamma)]\ \forall_{\mathrm{i}} \quad \frac{\Gamma \vdash \forall x.P}{\Gamma \vdash P[x/a]}\ \forall_{\mathrm{e}}$$

$$\frac{\Gamma \vdash P[x/a]}{\Gamma \vdash \exists x.P}\ \exists_{\mathrm{i}} \quad \frac{\Gamma \vdash \exists x.P \quad \Gamma, P \vdash C}{\Gamma \vdash C}\ [x \notin \mathrm{vl}(\Gamma, C)]\ \exists_{\mathrm{e}}$$

$$\frac{}{\Gamma \vdash a = a}\ =_{\mathrm{i}} \quad \frac{\Gamma \vdash a = b \quad \Gamma \vdash P[x/a]}{\Gamma \vdash P[x/b]}\ =_{\mathrm{e}}$$

Figure 2.1 – Règles de déduction de la déduction naturelle du calcul des prédicats

Une façon de visualiser le comportement de $\forall$ et $\exists$ est de les voir respectivement comme une conjonction sur l'univers et une disjonction sur l'univers.

Ceci signifie que prouver $\forall x.P$, il faut prouver pour un x quelconque la proposition P, d'où la condition non syntaxique que $x \notin \mathrm{vl}(\Gamma)$: x doit être choisi quelconque (c'est d'ailleurs bien ainsi qu'on raisonne pour montrer « pour tout x, $P(x)$ » : on considère un x quelconque et on prouve qu'il possède cette propriété). L'élimination est très proche, elle aussi, de l'élimination de $\wedge$: à partir de cette conjonction universelle, on peut extraire un cas particulier, c'est-à-dire déduire $P[x/a]$ pour un élément a.

L'introduction de $\exists$, de même, est simplement un analogue de l'introduction de $\vee$: il suffit de prouver que l'un des $P[x/a]$ est vrai pour prouver qu'il existe bien x tel que P. L'élimination, quant à elle, peut sembler compliquée, mais elle est juste un analogue à la disjonction de cas : le principe est que si l'on a prouvé $\exists x.P$, alors pour prouver une proposition C, il suffit de la prouver sous l'hypothèse P. Cependant, nous devons prendre une variable x non présente dans C et Γ. C'est ainsi que l'on procède en général d'ailleurs : si l'on sait qu'il existe x tel que $P(x)$, alors on introduit un tel x pour prouver notre proposition.

Les deux dernières règles, liées à l'égalité, sont assez naturelles : un élément est égal à lui-même et pour toute propriété, P, si $a = b$ alors on peut remplacer les occurrences de a par b dans P.

Exercice 2.2.1 (L'égalité est une relation d'équivalence)**.** Dériver le fait que l'égalité est une relation d'équivalence, c'est-à-dire :

- $\vdash \forall x.x = x$
- $\vdash \forall x.\forall y.(x = y) \to (y = x)$
- $\vdash \forall x.\forall y.\forall z.(x = y) \to (y = z) \to (x = z)$

Remarque 2.2.2. Pour pouvoir utiliser plus convenablement nos propositions, on considère que les propositions $\forall x.P$ et $\forall z.P[x/z]$ avec $z \notin \mathrm{vl}(P)$ sont équivalentes, et pareil pour $\exists x.P$. Cela nous permet d'introduire de nouvelles variables à chaque fois et donc de gagner en lisibilité. De plus, le quantificateur sera de priorité minimale : $\forall x.x = c \vee d = e$ sera lu comme $\forall x.((x = c) \vee (d = e))$.

Remarque 2.2.3. Encore pour gagner en lisibilité, nous noterons souvent $P(x)$ si l'on considère une proposition où x en est une variable libre. Cela permet par exemple d'écrire $\forall x.P(x)$ devenant $P(a)$ en utilisant $\forall_{\mathrm{e}}$. Nous avons déjà utilisé quelques fois cette convention, mais nous la considérerons comme usuelle maintenant.

Remarque 2.2.4. Les règles $\neg_{\mathrm{i}}$ et $\neg_{\mathrm{e}}$ permettent d'identifier directement $\neg A$ et $A \to \bot$. Dans notre logique actuelle le rôle de la négation est assez fort, aussi nous continuerons de garder ces règles, mais nous verrons plus tard qu'elles peuvent être une mauvaise manière de considérer la négation.

Exemple 2.2.1. Considérons un langage simple : $\mathcal{L} = \{f, r\}$ où f est une fonction d'arité 1 et r est une relation elle aussi d'arité 1. Montrons alors que

$$(\forall x.r(x) \to r(f(x))) \to (\forall x.r(x) \to r(f(f(x))))$$

avec l'arbre de dérivation suivant (pour alléger la lecture de l'arbre, définissons $\Gamma := \forall x\ r(x) \to r(f(x)), r(t))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash r(t)}\,\mathrm{Ax} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall x\ r(x) \to r(f(x))}\,\mathrm{Ax}}{\Gamma \vdash r(t) \to r(f(t))}\,\forall_{\mathrm{e}}}{\Gamma \vdash r(f(t))}\,\to_{\mathrm{e}} \quad \dfrac{\dfrac{}{\Gamma \vdash \forall x\ r(x) \to r(f(x))}\,\mathrm{Ax}}{\Gamma \vdash r(f(t)) \to r(f(f(t)))}\,\forall_{\mathrm{e}}}{\forall x\ r(x) \to r(f(x)), r(t) \vdash r(f(f(t)))}\,\to_{\mathrm{e}}}{\forall x\ r(x) \to r(f(x)) \vdash r(t) \to r(f(f(t)))}\,\to_{\mathrm{i}}}{\forall x\ r(x) \to r(f(x)) \vdash \forall x\ r(x) \to r(f(f(x)))}\,\forall_{\mathrm{i}}}{\vdash (\forall x\ r(x) \to r(f(x))) \to (\forall x\ r(x) \to r(f(f(x))))}\,\to_{\mathrm{i}}$$

Comme avant, utiliser la déduction naturelle est laborieuse et les arbres de preuve sont longs et peu lisibles. Ceci dit, leur correspondance avec la manière naturelle de rédiger des preuves en maths en fait un bon outil pour savoir ce qui fait une bonne rédaction.

2.2.1 Théorie logique

Nous nous trouvons alors devant un blocage assez fort : la déduction naturelle ne nous parle pas de l'intérieur des termes. Par exemple, nous ne savons pas comment se comporte une fonction f donnée dans notre langage. Ainsi si nous pouvons à la limite ajouter des règles sur le prédicat égalitaire, il en reste que nous n'avons pas d'ensemble de règles qui permette de décrire chaque fois le comportement de notre signature. Heureusement, nous pouvons amener un ensemble d'hypothèses dans nos raisonnements en les mettant à gauche du symbole $\vdash$: notre règle axiome correspond exactement à utiliser un axiome que nous avons placé au préalable à gauche du symbole $\vdash$. Nous en venons donc naturellement à la notion de théorie logique.

Définition 2.2.3 (Théorie logique)**.** *Une théorie logique est un ensemble T de formules closes appelées axiomes (le risque de confusion entre cette utilisation du mot axiome et la règle du même nom est peu important, puisque dans la plupart des cas ils signifient la même chose). On dira que P est dérivable dans T lorsque $T \vdash P$. Une théorie logique pourra en particulier comporter une infinité d'axiomes.*

Exemple 2.2.2 (Théorie des groupes)**.** Nous avons vu le langage des groupes, et nous allons maintenant définir la théorie des groupes (qui se construit donc sur ce langage) :

$$\begin{aligned}
\mathrm{Ax}_1 &:= \forall x. e \cdot x = x \\
\mathrm{Ax}_2 &:= \forall x. x \cdot e = x \\
\mathrm{Ax}_3 &:= \forall x. \forall y. \forall z. x \cdot (y \cdot z) = (x \cdot y) \cdot z \\
\mathrm{Ax}_4 &:= \forall x. x \cdot (x^{-1}) = e \\
\mathrm{Ax}_5 &:= \forall x. x^{-1} \cdot x = e
\end{aligned}$$

On définit alors $T_{\mathrm{Grp}} = \{\mathrm{Ax}_1, \mathrm{Ax}_2, \mathrm{Ax}_3, \mathrm{Ax}_4, \mathrm{Ax}_5\}$.

Remarque 2.2.5. Le symbole := sert à donner la définition d'une formule, d'abord pour éviter de confondre cela avec une égalité au sein de la théorie syntaxique, et ensuite pour expliciter l'introduction de notre proposition.

Exercice 2.2.2. Prouver que $T_{\mathrm{Grp}} \vdash (x^{-1})^{-1} = x$

Si nous avions jusque là travaillé avec des logiques « pures » (au sens où elles ne considéraient que des tautologies), l'introduction de théorie logique nous fait considérer des propriétés qui leur sont liées.

Définition 2.2.4 (Théorie cohérente, incohérente, complète)**.** *On dit qu'une théorie T est cohérente si $T \nvdash \bot$, c'est-à-dire s'il n'existe aucune dérivation de $\bot$ avec les axiomes de T.*

Au contraire, si une théorie T est telle que $T \vdash \bot$, on dit qu'elle est incohérente.

Enfin, on dit qu'une théorie T est complète si T est cohérente et si pour toute proposition P soit $T \vdash P$ soit $T \vdash \neg P$ (remarquons que l'hypothèse que T est cohérente force ce choix à être exclusif).

Exercice 2.2.3 (Compacité syntaxique)**.** Montrer que si P est dérivable dans la théorie T, alors il existe un ensemble fini T_0 tel que $T_0 \vdash P$. *Indication : tous les objets impliqués sont finis.*

Montrer (c'est un énoncé équivalent) que si T est finiment cohérente, alors elle est cohérente.

Les théorie intéressantes sont évidemment les théories cohérentes, mais la complétude d'une théorie, au contraire, est trop difficilement atteignable pour être réellement satisfaisante (à cause des célèbres théorèmes d'incomplétude de Gödel, dont nous reparlerons).

2.3 Structures et modèles

Cette section s'attardera à l'aspect sémantique du calcul des prédicats : nous chercherons comment définir le sens d'une proposition. Il y a effectivement un fossé entre la simple valuation étant une fonction $2^{\mathcal{V}}$ et l'interprétation d'une proposition pouvant désigner des objets. Nous verrons donc, dans un premier temps, la notion de $\mathcal{L}$-structure, permettant d'interpréter un langage, puis celle de valuation dans une $\mathcal{L}$-structure, avant de définir ce qu'est un modèle.

2.3.1 Définitions préliminaires

Définition 2.3.1 ($\mathcal{L}$-structure)**.** *Soit $\mathcal{L}$ un langage fixé, on appelle $\mathcal{L}$-structure un triplet $\mathcal{S} = (|\mathcal{S}|, \iota_{\mathcal{F}}, \iota_{\mathcal{R}})$ où :*

- *$|\mathcal{S}|$ est un ensemble qu'on appelle le domaine de $\mathcal{S}$*
- *$\iota_{\mathcal{F}}$ est une fonction qui, à chaque élément $f \in \mathcal{F}$ associe une fonction $f^{\mathcal{S}} : |\mathcal{S}|^{\alpha_f(f)} \to |\mathcal{S}|$*
- *$\iota_{\mathcal{R}}$ est une fonction qui, à chaque élément $r \in \mathcal{R}$ associe une partie $r^{\mathcal{S}} \subseteq |\mathcal{S}|^{\alpha_r(r)}$.*

Remarque 2.3.1. Tout d'abord, on notera en général $\mathcal{S}$ le domaine de $\mathcal{S}$ au lieu de $|\mathcal{S}|$ (pour alléger l'écriture). De plus, on n'utilisera bien sûr pas les fonctions $\iota_{\mathcal{F}}$ et $\iota_{\mathcal{R}}$ explicitement : on parlera directement d'une relation $r^{\mathcal{S}}$ pour parler de l'interprétation du symbole r par exemple.

Un $\mathcal{L}$-structure est donc simplement une interprétation de notre langage dans un ensemble fixé, où l'on interprète nos fonctions et relations de façon naturelle (rappelons que dire que $x_1, \dots, x_n$ sont en relation pour une relation r peut s'écrire $(x_1, \dots, x_n) \in r$ et donc qu'on peut voir une relation comme une partie d'une puissance de notre ensemble).

Exemple 2.3.1 (La structure de groupe $\mathbb{Z}$)**.** On donne un exemple évident de structure pour le langage des groupes :

- Le domaine est $\mathbb{Z}$.
- Le symbole de fonction $\cdot$ est interprété par l'opération $+ : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ usuelle.
- Le symbole de fonction $-^{-1}$ est interprété par la fonction $x \mapsto -x$.

Sur cet exemple, nous voyons bien qu'en plus d'être une structure pour le langage des groupes, les conditions de T_{Grp} sont vérifiées : c'est ce que l'on appelle un modèle. Cependant, cette idée intuitive de condition vérifiée se doit d'être formalisée.

Avant de formaliser la notion de modèle, nous allons définir les morphismes entre structures. Comme en algèbre (ou pour les algèbres de Boole), un morphisme est une fonction qui préserve la structure, mais ce sens peut être fait de façon bien plus générale avec notre formalisme.

Définition 2.3.2 (Morphisme de $\mathcal{L}$-structure)**.** *Soit $\mathcal{L}$ un langage et $\mathcal{S}, \mathcal{M}$ deux $\mathcal{L}$-structures, on appelle morphisme de $\mathcal{L}$-structure (ou $\mathcal{L}$-morphisme, ou simplement morphisme si le contexte évite les ambiguïtés) une fonction $h : |\mathcal{S}| \to |\mathcal{M}|$ telle que :*

- *pour tout symbole de fonction f d'arité p, on a*

$$h(f^{\mathcal{S}}(e_1, \dots, e_p)) = f^{\mathcal{M}}(h(e_1), \dots, h(e_2))$$

- *pour tout symbole de relation r d'arité p, on a*

$$(e_1, \dots, e_p) \in r^{\mathcal{S}} \implies (h(e_1), \dots, h(e_p)) \in r^{\mathcal{M}}$$

Exercice 2.3.1 ($\mathcal{L}$-structures comme catégories)**.** Soit un langage $\mathcal{L}$, $\mathcal{S}$,$\mathcal{M}$ et $\mathcal{N}$ trois $\mathcal{L}$-structures. Soient $f : \mathcal{S} \to \mathcal{M}$ et $g : \mathcal{M} \to \mathcal{N}$ deux morphismes. Montrer que la composée $g \circ f$ est un morphisme de $\mathcal{S}$ vers $\mathcal{N}$.

Montrer que pour une $\mathcal{L}$-structure $\mathcal{S}$, $\text{id}_{\mathcal{S}}$ est un $\mathcal{L}$-morphisme.

Définition 2.3.3 (Isomorphisme)**.** *Soient $\mathcal{S}$ et $\mathcal{M}$ deux $\mathcal{L}$-structures. On dit qu'un morphisme $f : \mathcal{S} \to \mathcal{M}$ est un isomorphisme s'il existe g un $\mathcal{L}$-morphisme tel que $f \circ g = \text{id}_{\mathcal{M}}$ et $g \circ f = \text{id}_{\mathcal{S}}$. On dit alors que $\mathcal{S}$ et $\mathcal{M}$ sont isomorphes.*

Exercice 2.3.2. Soit $f : \mathcal{S} \to \mathcal{M}$ un $\mathcal{L}$-morphisme tel que pour chaque relation la condition de la forme $x \in r^{\mathcal{S}} \implies y \in r^{\mathcal{M}}$ est remplacée par $x \in r^{\mathcal{S}} \iff y \in r^{\mathcal{M}}$. Montrer que f est un isomorphisme si et seulement si f est bijective.

Remarque 2.3.2. Nous verrons tout au long de cette section l'influence des morphismes sur les éléments logiques. En effet, ils permettent en quelque sorte d'interpréter des structures au sein d'autres, et possèdent donc une forme « d'uniformité ».

2.3.2 Évaluation

A la différence du calcul propositionnel, il nous faut ici considérer des éléments d'un ensemble, et donc introduire une notion d'évaluation des expressions pour pouvoir travailler sur nos termes. Pour cela, remarquons que toute variable apparaissant dans une expression est libre (elle ne peut être liée qu'au niveau de la proposition). Nous définissons donc d'abord la notion d'environnement.

Définition 2.3.4 (Environnement)**.** *Soit $\mathcal{L}$ un langage et $\mathcal{S}$ une $\mathcal{L}$-structure, on appelle environnement une fonction partielle $\nu : \mathcal{V} \to \mathcal{S}$ et on note* $\mathrm{dom}(\nu)$ *l'ensemble des variables où l'environnement ν est défini.*

Définition 2.3.5 (Évaluation d'expression)**.** *Soit $\mathcal{L}$ un langage, $\mathcal{S}$ une $\mathcal{L}$-structure, ν un environnement et e une expression telle que* $\mathrm{vl}(e) \subseteq \mathrm{dom}(\nu)$*. Alors on définit l'évaluation* $\mathrm{eval}(\nu, e)$ *par induction :*

- $\mathrm{eval}(\nu, x_i) = \nu(x_i)$
- $\mathrm{eval}(\nu, f(e_1, \dots, e_p)) = f^{\mathcal{S}}(\mathrm{eval}(\nu, e_1), \dots, \mathrm{eval}(\nu, e_p))$

On notera généralement ν la fonction $\mathrm{eval}(\nu, -)$ *qui évalue une expression dans un contexte donné.*

Exercice 2.3.3. Soient e une expression, ν et ν' deux environnement. Montrer que si ν et ν' sont égales sur $\mathrm{vl}(e)$ alors $\nu(e) = \nu'(e)$.

Exercice 2.3.4 (Evaluation par un morphisme)**.** Soit ν un environnement sur une structure $\mathcal{S}$, et $f : \mathcal{S} \to \mathcal{M}$ un $\mathcal{L}$-morphisme. On note $f(\nu)$ l'environnement $x_i \mapsto f(\nu(x_i))$. Montrer alors que

$$f(\mathrm{eval}(\nu, e)) = \mathrm{eval}(f(\nu), e)$$

pour tout expression e.

Ainsi, comme pour le calcul propositionnel, la valeur de ν n'a d'intérêt quant à une expression e que pour les variables apparaissant dans e.

Finalement, avant de définir la notion de valuation, comme les quantificateurs nécessitent de faire des substitutions, nous allons nous en tirer en modifiant directement l'environnement : on va noter $\nu[x \leftarrow a]$ l'environnement ν dans lequel on a remplacé la valeur de $\nu(x)$ par a (si ν n'était pas défini en x, alors on étend ν).

On peut alors définir une valuation par rapport à une environnement :

Définition 2.3.6 (Valuation)**.** *Soit $\mathcal{L}$ un langage, $\mathcal{S}$ une $\mathcal{L}$-structure et ν un environnement, on définit* $\mathrm{Val}_{\mathcal{S},\nu} : \mathcal{P}_{\mathcal{L}} \to \{0, 1\}$ *par induction :*

$$\frac{}{\mathrm{Val}_{\mathcal{S},\nu}(\top) = 1} \quad \frac{}{\mathrm{Val}_{\mathcal{S},\nu}(\bot) = 0} \quad \frac{(\nu(e_1), \dots, \nu(e_{\alpha_r(r)})) \in r^{\mathcal{S}}}{\mathrm{Val}_{\mathcal{S},\nu}(r(e_1, \dots, e_{\alpha_r(r)})) = 1} \quad \frac{(\nu(e_1), \dots, \nu(e_{\alpha_r(r)})) \notin r^{\mathcal{S}}}{\mathrm{Val}_{\mathcal{S},\nu}(r(e_1, \dots, e_{\alpha_r(r)})) = 0}$$

$$\frac{\mathrm{Val}_{\mathcal{S},\nu}(P) = b_1 \quad \mathrm{Val}_{\mathcal{S},\nu}(Q) = b_2}{\mathrm{Val}_{\mathcal{S},\nu}(P \vee Q) = b_1 + b_2 - b_1 \times b_2} \quad \frac{\mathrm{Val}_{\mathcal{S},\nu}(P) = b_1 \quad \mathrm{Val}_{\mathcal{S},\nu}(Q) = b_2}{\mathrm{Val}_{\mathcal{S},\nu}(P \wedge Q) = b_1 \times b_2}$$

$$\frac{\mathrm{Val}_{\mathcal{S},\nu}(P) = b_1 \quad \mathrm{Val}_{\mathcal{S},\nu}(Q) = b_2}{\mathrm{Val}_{\mathcal{S},\nu}(P \to Q) = 1 - b_1 + b_1 \times b_2} \qquad \frac{\mathrm{Val}_{\mathcal{S},\nu}(P) = b_1}{\mathrm{Val}_{\mathcal{S},\nu}(\neg P) = 1 - b_1}$$

$$\frac{}{\mathrm{Val}_{\mathcal{S},\nu}(\forall x.P) = \min_{s\in\mathcal{S}}(b_s)} \; [\mathrm{Val}_{\mathcal{S},\nu[x\leftarrow s]}(P) = b_s] \qquad \frac{}{\mathrm{Val}_{\mathcal{S},\nu}(\exists x.P) = \max_{s\in\mathcal{S}}(b_s)} \; [\mathrm{Val}_{\mathcal{S},\nu[x\leftarrow s]}(P) = b_s]$$

Si cette notion de valuation n'est pas parfaitement satisfaisante (la valuation des quantificateurs demande un nombre potentiellement infini de conditions, et elles ne sont pas à strictement parler syntaxiques à cause de cette infinité), elle permet une définition plutôt lisible : la valuation de $\forall x.P$ est la plus petite valuation possible de P quand x varie dans $\mathcal{S}$ et inversement $\exists x.P$ est la plus grand valuation obtenue en faisant varier x dans $\mathcal{S}$. Nous allons maintenant introduire un lemme important :

Lemme 2.3.1. *Soient ν et ν' identiques partout sauf sur une variable x, alors la valuation de $\forall x.P$ et $\exists x.P$ reste inchangée selon si on se place dans l'environnement ν ou l'environnement ν'.*

Démonstration. Nous considérerons juste $\forall x.P$ puisque le raisonnement est identique pour $\exists x.P$. Par définition de la valuation de $\forall x.P$, nous évaluons $\mathrm{Val}_{\mathcal{S},\nu[x\leftarrow m]}(P)$ et $\mathrm{Val}_{\mathcal{S},\nu'[x\leftarrow m]}(P)$ mais par hypothèse, comme ν et ν' sont identiques ailleurs qu'en x, il en résulte que ces deux valeurs sont égales : donc l'ensemble de valeurs décrites est le même, donc le minimum (et le maximum) est identique : on en déduit que

$$\mathrm{Val}_{\mathcal{S},\nu}(\forall x.P) = \mathrm{Val}_{\mathcal{S},\nu'}(\forall x.P)$$

□

On peut alors prouver par induction que si P est une formule close, alors $\mathrm{Val}_{\mathcal{S},\nu}(P)$ est identique pour tout environnement ν.

Exercice 2.3.5. Prouver ce résultat.

La satisfiabilité dans le calcul des prédicats en découle naturellement :

Définition 2.3.7 (Satisfiabilité)**.** *On dit qu'une formule P sur un langage $\mathcal{L}$ est satisfiable dans un modèle $\mathcal{S}$ et dans un environnement ν si $\mathrm{Val}_{\mathcal{S},\nu}(P) = 1$. On note alors $\mathcal{S}, \nu \models P$.*

Si P est une formule close, alors on dit simplement qu'elle est satisfiable si elle est satisfiable dans tout environnement (de façon équivalente dans un environnement, de façon équivalente dans l'environnement vide), et on note alors $\mathcal{S} \models P$.

Ainsi on peut définir ce qu'est un modèle :

Définition 2.3.8 (Modèle d'une théorie)**.** *Soit T une théorie sur un langage $\mathcal{L}$, on dit qu'une $\mathcal{L}$-structure $\mathcal{M}$ est un modèle de T, et on note $\mathcal{M} \models T$, si pour toute formule $P \in T$, $\mathcal{M} \models P$.*

Définition 2.3.9 (Conséquence sémantique)**.** *Soit T une théorie logique. On dit que P est conséquence sémantique de T, et on note $T \vDash P$ si pour tout modèle $\mathcal{M} \models T$, on a $\mathcal{M} \models P$.*

Exercice 2.3.6. Vérifier que la structure donnée dans l'exemple 2.3.1 est bien un modèle de T_{Grp}.

2.3.3 Sous-modèle et morphisme

Les isomorphismes consistent en des transformations gardant pleinement l'information : on veut que deux modèles isomorphes satisfassent les mêmes propositions.

Lemme 2.3.2 (Invariance des valuations par isomorphisme)**.** *Soient $\mathcal{M}$ et $\mathcal{N}$ deux $\mathcal{L}$-structure, avec $f : \mathcal{M} \to \mathcal{N}$ un $\mathcal{L}$-isomorphisme de réciproque g, et T une théorie sur $\mathcal{L}$. Alors pour tout environnement ν sur $\mathcal{M}$, $\mathrm{Val}_{\mathcal{M},\nu}(P) = \mathrm{Val}_{\mathcal{N},f(\nu)}(P)$.*

Démonstration. Remarquons que l'exercice 2.3.4 va nous servir. Nous allons raisonner par induction sur $\mathrm{Val}_{\mathcal{M},\nu}$:

- Cas $\top$ et $\bot$: le calcul ne fait rien intervenir, le résultat est immédiat.
- Cas d'un terme appartenant à une relation : supposons que $(\nu(e_1), \ldots, \nu(e_p)) \in r^{\mathcal{M}}$, alors on en déduit, puisque f est un morphisme que $(f(\nu(e_1)), \ldots, f(\nu(e_p))) \in r^{\mathcal{N}}$, qui est une écriture non ambiguë grâce à l'exercice 2.3.4 (en effet, l'implication nous donne $f(\mathrm{eval}(\nu, e))$, mais par le résultat démontré en exercice nous pouvons le remplacer par $\mathrm{eval}(f(\nu), e)$). Alors, en appliquant la règle d'induction, on en déduit que

$$\mathrm{Val}_{\mathcal{N},f(\nu)}(r(e_1, \ldots, e_p)) = 1 = \mathrm{Val}_{\mathcal{M},\nu}(r(e_1, \ldots, e_p))$$

- Cas d'un terme n'appartenant pas à une relation : l'autre règle se traite exactement de la même façon mais en utilisant la réciproque g pour avoir l'implication réciproque, nous donnant alors que

$$\mathrm{Val}_{\mathcal{N},f(\nu)}(r(e_1, \ldots, e_p)) = \mathrm{Val}_{\mathcal{M},\nu}(r(e_1, \ldots, e_p))$$

- Les autres cas se traitent de façon directe jusqu'aux quantificateurs.
- Cas des quantificateurs : on montre d'abord que $f(\nu[x \leftarrow m]) = f(\nu)[x \leftarrow f(m)]$ pour tout m dans $|\mathcal{M}|$, nous donnant alors l'égalité des valuations. La démonstration de l'égalité sur l'image d'un environnement par rapport à un remplacement de valeur peut se faire sur la définition inductive de $[x \leftarrow m]$ mais nous ne le ferons pas ici (le lecteur assidu peut le faire). Enfin, la surjectivité de f signifie que

$$\max_{m \in \mathcal{M}} \mathrm{Val}_{\mathcal{N},f(\nu)[x \leftarrow f(m)])}(P) = \max_{n \in \mathcal{N}} \mathrm{Val}_{\mathcal{N},f(\nu[x \leftarrow n])}(P)$$

  et de même pour min.

□

Théorème 2.3.1 (Invariance par isomorphisme)**.** *Soient $\mathcal{M}$ et $\mathcal{N}$ deux $\mathcal{L}$-structures, $f : \mathcal{M} \to \mathcal{N}$ un $\mathcal{L}$-isomorphisme de réciproque g, et T une théorie sur $\mathcal{L}$. $\mathcal{M} \models T$ si et seulement $\mathcal{N} \models T$.*

Démonstration. Le lemme précédent nous permet directement de conclure que pour tout $P \in T$, $\mathrm{Val}_{\mathcal{M}}(P) = \mathrm{Val}_{\mathcal{N}}(P)$ et donc que $\mathcal{M} \models T$ si et seulement si $\mathcal{N} \models T$. □

Nous savons par exemple que dans les groupes, l'image par un morphisme d'un groupe est un sous-groupe du groupe d'arrivée. Cette idée se généralise aux $\mathcal{L}$-structures.

Définition 2.3.10 (Sous-structure)**.** *Soit $\mathcal{L}$ un langage, $\mathcal{S}$ une $\mathcal{L}$-structure. On dit qu'une $\mathcal{L}$-structure $\mathcal{M}$ est une sous-structure de $\mathcal{S}$ si $|\mathcal{M}| \subseteq |\mathcal{S}|$ et :*

- *pour chaque symbole de fonction f, $f^{\mathcal{S}}_{||\mathcal{M}|} = f^{\mathcal{M}}$, c'est-à-dire que $f^{\mathcal{S}}$ et $f^{\mathcal{M}}$ coïncident sur $\mathcal{M}$. En particulier, cela signifie que pour un symbole de constante c, $c^{\mathcal{S}} = c^{\mathcal{M}}$.*
- *pour chaque symbole de relation r d'arité p, on a $r^{\mathcal{M}} = r^{\mathcal{S}} \cap |\mathcal{M}|^p$*

Proposition 2.3.1. *Si $f : \mathcal{M} \to \mathcal{N}$ est un $\mathcal{L}$-morphisme injectif, alors $\mathcal{M}$ est isomorphe à une sous-structure de $\mathcal{N}$.*

Exercice 2.3.7. Démontrer ce résultat.

2.4 Correction et complétude dans le calcul des prédicats du premier ordre

L'objectif de cette section sera de montrer l'équivalence entre $T \vdash P$ et $T \vDash P$. Nous y gagnerons le théorème de compacité quand nous aurons le théorème de complétude.

2.4.1 Correction

Énonçons directement le théorème que nous cherchons à montrer.

Théorème 2.4.1 (Correction)**.** *Si $T \vdash P$ alors $T \vDash P$.*

Pour reformuler ce théorème, cela signifie que si notre système de preuve dérive $T \vdash P$, alors pour toute structure $\mathcal{M}$ telle que $\mathrm{Val}_{\mathcal{M}}(T) = 1$, on a $\mathrm{Val}_{\mathcal{M}}(P) = 1$. Le raisonnement se fera donc évidemment par induction sur $T \vdash P$. Comme les structures sur les connecteurs logiques sont les mêmes, nous n'aborderons pas les cas de $\vee$, $\wedge$, $\rightarrow$ et $\neg$, identiques à ceux du calcul propositionnel. Il nous reste donc à traiter le cas des quantificateurs et de l'égalité. Cependant, nous allons prouver une version plus forte pour pouvoir considérer les environnements (et donc des formules non closes).

Nous allons donc introduire des lemmes liés à la gestion des environnements pour pouvoir calculer efficacement les substitutions dans les termes :

Lemme 2.4.1. *Soient t, u des termes et ν un environnement permettant d'évaluer t et u. Soit $v = t[x/u]$ et $\nu' = \nu[x \leftarrow \mathrm{eval}(u, \nu)]$. Alors $\nu(v) = \nu'(t)$.*

Démonstration. La preuve se fait par induction sur la définition de eval. Nous ne détaillerons pas la preuve car elle est directe, mais l'idée est que dans le cas de base, $\nu(x_i) = \nu'(x_i)$ car soit on a x et les deux valeurs sont $\nu(u)$, soit le fait que ν et ν' coïncident ailleurs nous donne la propriété. Le passage pour une fonction f est direct aussi. □

Lemme 2.4.2. *Soit P une formule, t un terme et ν un environnement. On définit $\nu' = \nu[x \leftarrow \nu(t)]$, alors $\mathrm{Val}_{\mathcal{M},\nu}(P[x/t]) = \mathrm{Val}_{\mathcal{M},\nu'}(P)$.*

Démonstration. L'induction se fait sur Val de façon naturelle. □

Exercice 2.4.1. Écrire l'induction.

Lemme 2.4.3. *Si $T \vdash P$ où P est une proposition avec (possiblement) des variables libres et T est un ensemble de propositions (possiblement) des variables libres, alors pour toute structure $\mathcal{M}$ sur $\mathcal{L}$ et pour tout environnement ν, si $\mathrm{Val}_{\mathcal{M},\nu}(\Gamma) = 1$ alors $\mathrm{Val}_{\mathcal{M},\nu}(P) = 1$.*

Démonstration. Nous allons donc faire l'induction sur les cas considérés :

- Cas $\exists_{\mathrm{i}}$: on suppose que $T \vdash P[x/a]$ et qu'alors, $\mathrm{Val}_{\mathcal{M},\nu}(P[x/a]) = 1$. Par le lemme précédent, $\mathrm{Val}_{\mathcal{M},\nu[x\leftarrow\nu(a)]}(P) = 1$, ce qui prouve que $\max_{m\in\mathcal{M}} \mathrm{Val}_{\mathcal{M},\nu[x\leftarrow m]}(P) = 1$, donc que $\mathrm{Val}_{\mathcal{M},\nu}(\exists x.P) = 1$.
- Cas $\exists_{\mathrm{e}}$: On peut remplacer $\Gamma, P \vdash C$ par $\Gamma \vdash P \rightarrow C$. Dans ce cas, on suppose $T \vdash \exists x.P$ et $T \vdash P \rightarrow C$. On en déduit que
$$\max_{m\in\mathcal{M}} \mathrm{Val}_{\mathcal{M},\nu[x\leftarrow m]}(P) = 1$$
et que $\mathrm{Val}_{\mathcal{M},\nu}(P \rightarrow C) = 1$ pour un certain environnement ν. Comme x n'apparaît ni dans T ni dans C, la valeur de $\mathrm{Val}_{\mathcal{M},\nu}(C)$ ne dépend pas de $\nu(x)$: cela signifie que $\mathrm{Val}_{\mathcal{M},\nu}(C) = \mathrm{Val}_{\mathcal{M},\nu[x\leftarrow m]}(C)$ où m est tel que $\mathrm{Val}_{\mathcal{M},\nu[x\leftarrow m]}(P) = 1$ (qui existe par hypothèse). Par la définition de $\mathrm{Val}_{\mathcal{M},\nu[x\leftarrow m]}(P \rightarrow C) = 1$ on en déduit que $\mathrm{Val}_{\mathcal{M},\nu}(C) = 1$ pour toute valuation ν.

- Cas $\forall_i$: On suppose que $T \vdash P$ et que x n'apparaît pas dans T. Cela signifie que pour chaque $m \in \mathcal{M}$, la valeur $\mathrm{Val}_{\mathcal{M},\nu}(P[x/m])$ (qui est $\mathrm{Val}_{\mathcal{M},\nu[x\leftarrow m]}(P)$) est indépendante de T, et donc constante. Comme ces valeurs valent toutes 1, on en déduit que $\mathrm{Val}_{\mathcal{M},\nu}(\forall x.P) = 1$.
- Cas $\forall_e$: puisque $\mathrm{Val}_{\mathcal{M},\nu}(P)$ est supérieure au minimum des valeurs pour $\nu[x \leftarrow m]$, on en déduit directement que $\mathrm{Val}_{\mathcal{M},\nu}(P) = 1$ par l'hypothèse d'induction.
- Cas $=_i$: Le terme est vrai de façon évidente.
- Cas $=_e$: Pour que $\mathrm{Val}(a = b) = 1$, il faut que a et b soient identiques, donc on en déduit que

$$\mathrm{Val}_{\mathcal{M},\nu}(P[x/a]) = \mathrm{Val}_{\mathcal{M},\nu[x\leftarrow a]}(P) = \mathrm{Val}_{\mathcal{M},\nu[x\leftarrow b]}(P) = \mathrm{Val}_{\mathcal{M},\nu}(P[x/b])$$

Cela termine notre induction. □

On peut alors directement en déduire la démonstration du théorème de correction :

Démonstration. Si $T \vdash P$ avec P close, alors pour tout modèle $\mathcal{M} \models T$, le lemme précédent nous dit que $\mathcal{M} \models P$, donc $T \vDash P$. □

Donnonc maintenant un exercice important pour construire des théories complètes à partir de modèles :

Exercice 2.4.2. Soit $\mathcal{M}$ une $\mathcal{L}$-structure. On note $\mathcal{T}h(\mathcal{M})$ l'ensemble des formules vraies dans $\mathcal{M}$, c'est-à-dire l'ensemble des P tels que $\mathcal{M} \models P$. Monter que $\mathcal{T}h(\mathcal{M})$ est une théorie complète et que si $\mathcal{M} \models T$ alors $t \subseteq \mathcal{T}h(\mathcal{M})$.

Proposition 2.4.1 (Énoncé équivalent de la correction)**.** *Le théorème de correction est équivalent à la propriété suivante : s'il existe un modèle $\mathcal{M} \models T$ alors $T \nvdash \bot$ (T est cohérente).*

Démonstration. Supposons le théorème de correction, supposons qu'il existe un modèle $\mathcal{M} \models T$. Supposons que $T \vdash \bot$. Alors par construction (par analyse sur les règles de $\vdash$) on trouve une proposition P telle que $T \vdash P$ et $T \vdash \neg P$. On trouve alors que $\mathcal{M} \models P$ et $\mathcal{M} \models \neg P$ ce qui, en calculant Val, nous donne que $0 = 1$.

Réciproquement, si l'on suppose que l'existence d'un modèle implique que $T \nvdash \bot$, alors supposons que $T \vdash P$. Dans ce cas, $T, \neg P \vdash \bot$ donc par contraposée de l'hypothèse, il n'existe aucun modèle de $T, \neg P$. Donc si $\mathcal{M} \models T$, alors $\mathrm{Val}_{\mathcal{M}}(\neg P) = 0$, donc $\mathrm{Val}_{\mathcal{M}}(P) = 1$, ce qui signifie que $\mathcal{M} \models P$. □

2.4.2 Théorème de complétude de Gödel

L'autre sens, qui est donc le théorème de complétude, dû historiquement à Gödel (si les démonstrations modernes divergent de sa démonstration historique, on continue de l'appeler théorème de complétude de Gödel), est beaucoup plus dur à démontrer. Nous allons procéder par étapes pour cela, mais commençons par présenter le théorème :

Théorème 2.4.2 (Complétude de Gödel)**.** *Si $T \vDash P$ alors $T \vdash P$.*

Comme pour le théorème de correction, le théorème de complétude a une version équivalente liée à la contradiction (le fait de prouver $\bot$ d'un côté, le fait de ne pas avoir de modèle de l'autre). La version équivalente est évidemment la réciproque de la version équivalente du théorème de correction, mais dans le cas de la complétude c'est bien cette version équivalente que nous allons prouver.

Lemme 2.4.4 (Énoncé équivalent de la complétude)**.** *Le théorème de complétude est équivalent à la propriété suivante : si $T \nvdash \bot$ alors il existe un modèle $\mathcal{M} \models T$.*

Démonstration. Supposons le théorème de complétude. Par contraposée, supposons qu'il n'existe aucun modèle de T, c'est-à-dire que $\mathrm{Val}_{\mathcal{M}}(T) = 0$ pour toute structure $\mathcal{M}$. Alors pour toute proposition P, comme la prémisse $\mathcal{M} \models T$ est fausse, on a $T \vDash P$, donc par complétude $T \vdash P$. On peut donc appliquer cet argument à P et $\neg P$ pour obtenir, par la règle $\neg_{\mathrm{e}}$, que $T \vdash \bot$.

Réciproquement, supposons que si $T \nvdash \bot$ alors il existe un modèle $\mathcal{M} \models T$. Supposons de plus que $T \vDash P$ et montrons que $T \vdash P$. Notre hypothèse nous dit que $T, \neg P$ n'a pas de modèle, donc $T, \neg P \vdash \bot$ par contraposée de l'hypothèse. Par l'utilisation de la règle $\bot_{\mathrm{c}}$, on en déduit que $T \vdash P$. □

Nous voulons donc construire un modèle en supposant que T est une théorie cohérente. Cette construction, complexe, sera détaillée en plusieurs étapes. D'abord, nous allons avoir besoin de résultats sur les extensions d'un langage.

Extension d'un langage

L'idée d'une extension est de rajouter des fonctions ou des relations à notre signature, pour avoir des $\mathcal{L}$-structures plus, justement.. Structurées.

Définition 2.4.1 (Extension d'un langage)**.** *Soit $\mathcal{L} = \langle \mathcal{F}, \mathcal{R}, \alpha_f, \alpha_r \rangle$ une signature, on dit que $\mathcal{L}' = \langle \mathcal{F}', \mathcal{R}', \alpha'_f, \alpha'_f \rangle$ est une extension de $\mathcal{L}$ lorsque :*

- $\mathcal{F} \subseteq \mathcal{F}'$
- $\mathcal{R} \subseteq \mathcal{R}'$
- *pour tout* $f \in \mathcal{F}, \alpha_f(f) = \alpha'_f(f)$
- *pour tout* $r \in \mathcal{R}, \alpha_r(r) = \alpha'_r(r)$

Une extension est donc simplement un langage contenant le langage initial.

Exemple 2.4.1. Il est évident que le langage des anneaux est une extension du langage des groupes.

Avec cette extension syntaxique intervient une extension sémantique :

Définition 2.4.2 (Enrichissement d'une structure)**.** *Soit $\mathcal{M}$ une $\mathcal{L}$-structure et $\mathcal{L}'$ une extension de $\mathcal{L}$. On dit que $\mathcal{M}'$ est un enrichissement de $\mathcal{M}$ (par rapport à $\mathcal{L}'$) si :*

- $|\mathcal{M}| = |\mathcal{M}'|$
- *pour tout symbole de fonction f de $\mathcal{L}$, $f^{\mathcal{M}} = f^{\mathcal{M}'}$*
- *pour tout symbole de relation r de $\mathcal{L}$, $r^{\mathcal{M}} = r^{\mathcal{M}'}$*

C'est donc une structure $\mathcal{M}'$ identique à $\mathcal{M}$ vis-à-vis du langage $\mathcal{L}$.

Exemple 2.4.2. En reprenant la comparaison entre groupe et anneau : tout anneau est l'enrichissement de sa structure de groupe additif sous-jacente. Cependant, pour un anneau $(A, +, \times)$, le groupe $(A^{\times}, \times)$ des éléments inversibles de A n'est

Remarque 2.4.1. Étant donné que $|\mathcal{M}| = |\mathcal{M}'|$, tout environnement sur $\mathcal{M}$ est un environnement sur $\mathcal{M}'$ et inversement (et ce, sans abus de notation).

L'idée intuitive est que la vérité des propositions dans $\mathcal{L}$ ne dépendra pas des éléments de $\mathcal{L}'$ hors de $\mathcal{L}$. Nous formalisons cela par les lemmes suivants :

Lemme 2.4.5. *Si t est un terme de $\mathcal{L}$, ν un environnement de $\mathcal{M}$ et ν' un environnement de $\mathcal{M}'$ avec $\mathcal{M}'$ un enrichissement de $\mathcal{M}$ tels que ν et ν' coïncident, alors $\nu(t) = \nu'(t)$.*

Démonstration. Par induction sur t : si t est une variable, $\nu(t) = \nu'(t)$ par hypothèse. Si t est une fonction, c'est une fonction de $\mathcal{L}$ et par définition de l'enrichissement d'une structure, on en déduit que $\nu(t) = \nu'(t)$. □

Lemme 2.4.6. *Si P est une proposition de $\mathcal{L}$ et ν un environnement, alors*

$$\mathrm{Val}_{\mathcal{M},\nu}(P) = \mathrm{Val}_{\mathcal{M}',\nu}(P)$$

Exercice 2.4.3. Prouver le lemme précédent.

On en déduit :

Proposition 2.4.2. *Soit un langage $\mathcal{L}$ permettant d'écrire les termes de P et $\mathcal{M}$ une $\mathcal{L}$-structure, alors pour toute extension $\mathcal{L}'$ de $\mathcal{L}$ et enrichissement $\mathcal{M}'$ de $\mathcal{M}$ par rapport à $\mathcal{L}'$,*

$$\mathcal{M}' \models P \text{ si et seulement si } \mathcal{M} \models P$$

Plan d'attaque de la preuve Notre preuve vise à constuire un modèle à partir de T, qui soit « canonique » : nous utiliserons comme interprétation des termes les termes directement. Mais plusieurs problèmes vont naitre de cette approche. Nous allons les lister ici, puis successivement les résoudre :

- Le premier problème est que pour un terme $\exists x.P$, nous n'avons pas forcément d'objet a tel que $P[x/a]$ (ce qui est la prémisse de la règle pour dériver cette proposition). Nous allons donc construire une extension de notre langage où nous ajouterons des constantes qui, pour chaque proposition de la forme $\exists x.P$, aura une constante c_P et permettra d'avoir $\exists x.P \to P[x/c_P]$. Pour cela, nous étendront aussi notre théorie, mais il est évident que si l'on a construit un modèle pour une théorie plus large que T, on a aussi construit un modèle pour T.
- Le deuxième problème est que, comme on l'a vu, l'ensemble des formules vraies sur un modèle est une théorie complète. Il nous faut donc chercher à étendre notre théorie en une théorie complète (cette étape peut sembler artificielle mais elle sera ensuite importante pour résoudre le troisième problème).
- Il est possible d'avoir deux termes non égaux syntaxiquement mais qui devraient être égaux : par exemple en arithmétique on ne souhaiterait surtout pas que 2 et $1+1$ soient des termes différents (pourtant, ils sont bien des termes différents pour l'instant). On doit donc quotienter notre ensemble de termes syntaxiques par les égalités que l'on peut prouver, pour nous donner des termes (puis vérifier que les relations et symboles passent bien au quotient).

A la fin, on se retrouve donc avec un modèle sur un langage énorme, d'une théorie bien plus grande que la théorie originelle. Heureusement, on a prouvé qu'être un modèle était invariant par extension du langage et que « qui peut le plus peut le moins » : être modèle d'une surthéorie implique qu'on est modèle de la théorie d'origine.

Témoins de Henkin

Comme nous l'avons dit, nous allons enrichir notre langage, et augmenter notre théorie. Nous allons d'abord définir notre langage puis notre théorie. Cependant, par souci de lisibilité, nous allons désormais noter $P(x)$ une proposition possédant une seule variable libre, étant x (de même pour n'importe quelle variable $\alpha \in \mathcal{V}$) et dans ce cas, nous noterons $P(a)$ pour $P[x/a]$, là encore pour alléger l'écriture.

Définition 2.4.3 (Langage et théorie des témoins)**.** *A partir du langage $\mathcal{L}$ et de la théorie T, nous allons définir $\mathcal{L}'$ et T' par induction. Comme les conditions sont longues à écrire et que l'induction d'un ensemble se fait sur l'autre, nous allons faire la construction inductive directement. De plus, comme $\mathcal{L}_0 = \mathcal{L}$ et que nous n'ajouterons que des constantes, nous travaillerons comme si $(\mathcal{L}_n)$ était une suite d'ensembles (les arités des constantes étant 0, elles s'induisent facilement, et les arités des autres symboles sont donnés par $\mathcal{L} = \mathcal{L}_0$). On va noter $\mathcal{P}_{1,\mathcal{L}}$ l'ensemble des propositions sur le langage $\mathcal{L}$ avec 1 variable libre. La construction est alors :*

- $\mathcal{L}_0 = \mathcal{L}, T_0 = T$
- $\mathcal{L}_{n+1} = \mathcal{L}_n \cup \{c_P \mid P \in \mathcal{P}_{1,\mathcal{L}_n}\}$
- $T_{n+1} = T_n \cup \{(\exists x.P(x) \to P(c_P)) \mid P \in \mathcal{P}_{1,\mathcal{L}_n}\}$

Et on définit alors $\mathcal{L}' = \bigcup_{n\in\mathbb{N}} \mathcal{L}_n$ *et* $T' = \bigcup_{n\in\mathbb{N}} T_n$.

La définition est assez lourde, mais son intuition est claire : on ajoute à chaque formule $\exists x.P$ un témoin c_P.

Il convient alors de prouver que T' est encore cohérente. Là encore, nous allons devoir nous armer de nombreux lemmes de structure pour travailler sur les séquents.

Lemme 2.4.7 (Généralisation d'une constante)**.** *Soient* Γ *un contexte et* A *une formule, et* c *une constante n'apparaissant pas dans* Γ*. Alors si* $\Gamma \vdash P$*, alors* $\Gamma \vdash \forall x.P[c/x]$.

Démonstration. Le but est évidemment d'appliquer $\forall_\text{i}$. On procède par induction sur la preuve de $\Gamma \vdash P$ pour montrer que $\Gamma \vdash P[c/x]$ où x est libre dans Γ (l'idée est que si c n'apparaît pas dans Γ, x n'y apparaît pas non plus au long de l'arbre de dérivation pour $P[c/x]$). La règle s'applique ensuite directement. □

Lemme 2.4.8 (Propriété du témoin)**.** *Pour toute propriété* $P(x)$ *sur* $\mathcal{L}'$*, on a* $T' \vdash \exists x.P(x) \to P(c_P)$.

Démonstration. Il suffit de remarquer que $P(x)$ est un objet fini et utilise donc des symboles de $\mathcal{L}'$ qui ne dépassent pas un certain rang m : on peut donc utiliser la règle Ax dans la théorie T_m puis par affaiblissement en déduire le résultat. □

Lemme 2.4.9. *Les séquents suivants sont dérivables :* $(\forall x.(P(x) \to \bot)) \vdash ((\exists x.P(x)) \to \bot)$ *et* $((\exists x.P(x)) \to \bot) \vdash \forall x.(P(x) \to \bot)$

Exercice 2.4.4. Prouver ce lemme.

Lemme 2.4.10. *Les séquents suivants sont dérivables :* $\exists x.\exists y.(P(x) \wedge Q(y)) \vdash (\exists x.P(x)) \wedge (\exists y.Q(y))$ *et* $(\exists x.P(x)) \wedge (\exists y.Q(y)) \vdash \exists x.\exists y.P(x) \wedge Q(y)$

Exercice 2.4.5. Prouver ce lemme.

Lemme 2.4.11. *Les séquents suivants sont dérivables, si l'on suppose que* x *n'est pas une variable libre de* P *:* $\exists x.(P \to Q) \vdash P \to (\exists x.Q)$ *et* $P \to (\exists x.Q) \vdash \exists x.(P \to Q)$.

Exercice 2.4.6. Prouver ce lemme.

Il nous reste maintenant à prouver que T' est encore cohérente :

Proposition 2.4.3 (Cohérence de T')**.** T' *est cohérente.*

Démonstration. Remarquons d'abord qu'avec l'exercice 2.2.3 le fait d'être finiment cohérent implique d'être cohérent (et toute partie finie de T' se trouve dans un T_n), il nous suffit donc de prouver que T_n est cohérente par récurrence :

- Par hypothèse, $T = T_0$ est cohérente.
- Supposons que T_n est cohérente. Raisonnons par l'absurde et supposons que T_{n+1} est incohérente : $T_{n+1} \vdash \bot$. Par définition de T_{n+1}, et par compacité syntaxique, on trouve $A_1, \dots, A_k$ avec des propositions $P_1, \dots, P_k$ à une variable libre écrites sur $\mathcal{L}_n$ telles que $A_i = \exists x.P_i(x) \to P_i(c_{P_i})$ et $T_n, A_1, \dots, A_k \vdash \bot$ ce que l'on va traduire alors. D'abord, on va transformer cette suite d'implications en une conjonction. En effet comme la règle $\dfrac{A, B \vdash P}{A \wedge B \vdash P}$ est admissible, le fait qu'on ait une dérivation de $T_n, A_1, \dots, A_k \vdash \bot$ nous permet d'en déduire une dérivation de

$$T_n, \bigwedge_{i=1}^{k} A_i \vdash \bot$$

soit avec la règle $\to_{\mathrm{i}}$ que

$$T_n \vdash \left(\bigwedge_{i=1}^{k} A_i \right) \to \bot$$

On peut alors utiliser pour chaque c_{P_i} le lemme de généralisation d'une constante, puisque par définition les c_{P_i} appartiennent à $\mathcal{L}_{n+1}$ et n'apparaissent donc pas dans T_n, ce qui nous donne :

$$T_n \vdash \forall y_1, \dots, \forall y_k, \left(\bigwedge_{i=1}^{k} (\exists x.P_i(x) \to P_i(y_i)) \right) \to \bot$$

En utilisant maintenant le lemme 2.4.9 pour distribuer le $\forall$ à gauche du $\to \bot$:

$$T_n \vdash \left(\exists y_1, \dots, \exists y_k. \bigwedge_{i=1}^{k} (\exists x.P_i(x) \to P_i(y_i)) \right) \to \bot$$

On utilise maintenant le lemme 2.4.10 pour faire passer chaque quantificateur existentiel de y_i à l'intérieur de la conjonction :

$$T_n \vdash \left(\bigwedge_{i=1}^{k} (\exists y_i.\exists x.P_i(x) \to P_i(y_i)) \right) \to \bot$$

Enfin le dernier lemme nous donne :

$$T_n \vdash \left(\bigwedge_{i=1}^{k} (\exists x.P_i(x) \to \exists y_i P(y_i)) \right) \to \bot$$

Il nous reste à prouver cette conjonction, pour laquelle nous n'avons à prouver qu'un cas (chaque clause à la même forme) :

$$\dfrac{\dfrac{\dfrac{}{\exists x.P(x) \vdash \exists x.P(x)}\ \mathrm{Ax} \quad \dfrac{\dfrac{}{\exists x.P(x), P(x) \vdash P(x)}\ \mathrm{Ax}}{\exists x.P(x), P(x) \vdash P(x)}\ \exists_{\mathrm{i}}}{\exists x.P(x) \vdash \exists y.P(y)}\ \exists_{\mathrm{e}}}{\vdash \exists x.P(x) \to \exists y.P(y)}\ \to_{\mathrm{i}}$$

Ce qui, par un *modus ponens*, nous permet de déduire $T_n \vdash \bot$, ce qui est en contradiction avec l'hypothèse de récurrence.

Ainsi, par récurrence, T_n est cohérente et donc T' l'est aussi. □

Compléter une théorie

Pour les besoins de la preuve, nous voulons plonger T' dans une théorie complète, c'est-à-dire une théorie $\mathcal{T}h(T')$ telle que $T' \subseteq \mathcal{T}h(T')$ et pour toute proposition P, $\mathcal{T}h(T') \vdash P$ ou $\mathcal{T}h(T') \vdash \neg P$. Pour cela, nous allons utiliser le lemme de Zorn, que nous allons d'abord devoir énoncer.

Axiome 2.4.1 (Lemme de Zorn)**.** Pour tout ensemble inductif, c'est-à-dire tout ensemble ordonné $(X, \leq)$ non vide tel que pour toute suite croissante, (x_i) possède un majorant, il existe un élément maximal, c'est-à-dire un élément x tel qu'il n'existe pas d'élément y tel que $x \leq y$.

Nous allons utiliser ce lemme pour construire une théorie $\mathcal{T}h(T')$ complète à partir de T'. Tout d'abord, nous allons appeler $\mathcal{E}$ (pour extension) la partie engendrée par T', au sens où $T' \subseteq \mathcal{E}$ et si $T' \vdash P$ alors $P \in \mathcal{E}$. De plus, si $P \in \mathcal{E}$ et $Q \in \mathcal{E}$, alors $P \wedge Q \in \mathcal{E}$. On appelle cette structure le filtre engendré par T'. Remarquons qu'on peut interpréter $\mathcal{E}$ comme l'ensemble des conséquences de la théorie T'.

Définition 2.4.4 (Filtre)**.** *Soit l'ensemble $X = \mathcal{P}_{\mathcal{L}'}$, alors une partie $\mathcal{F} \subseteq X$ est appelé un filtre si :*

- $\top \in \mathcal{F}$
- *si* $A \in \mathcal{F}$ *et* $A \vdash B$ *alors* $B \in \mathcal{F}$
- *si* $A \in \mathcal{F}$ *et* $B \in \mathcal{F}$ *alors* $A \wedge B \in \mathcal{F}$
- $\bot \notin \mathcal{F}$

Remarque 2.4.2. Dans la littérature, la condition de ne pas avoir $\bot$ n'est pas toujours donnée, mais les filtres intéressants sont toujours ceux dits non triviaux, c'est-à-dire différents de $P(X)$, qui est le seul « filtre » contenant la partie vide.

Exercice 2.4.7. Montrer que $\mathrm{Sp}_F(X)$, l'ensemble des filtres de X contenant un filtre fixé F, ordonné par l'inclusion, est un ensemble inductif.

On en déduit le théorème suivant :

Théorème 2.4.3 (Extension de théorie)**.** *Soit T une théorie logique cohérente, alors il existe une théorie complète contenant T.*

Démonstration. On considère donc $\mathcal{E}$ l'extension de la théorie T, et par le résultat de l'exercice précédent et le lemme de Zorn, on déduit l'existence de $\mathcal{T}h(T)$ un filtre maximal pour la conséquence logique qui contient T (de par la définition de l'ensemble inductif). Montrons alors que $\mathcal{T}h(T)$ est complète, pour cela, soit P une proposition :

- Soit $\mathcal{T}h(T) \vdash P$
- Soit $\mathcal{T}h(T) \vdash \neg P$
- Soit $\mathcal{T}h(T)$ ne prouve ni P, ni $\neg P$. Dans ce cas, on peut prendre au choix l'extension de $\mathcal{T}h(T) \cup \{P\}$ ou $\mathcal{T}h(T) \cup \{\neg P\}$: puisque les deux jouent un rôle symétrique, nous ne traiterons que le premier cas. Comme $\mathcal{T}h(T)$ est maximal pour l'inclusion, cela signifie que l'ensemble des propositions découlant de $\mathcal{T}h(T)$ et P est l'ensemble des propositions (le filtre trivial), donc $\mathcal{T}h(T), P \vdash \bot$ ce qui signifie que $\mathcal{T}h(T) \vdash \neg P$.

Dans tous les cas soit $\mathcal{T}h(T) \vdash P$ soit $\mathcal{T}h(T) \vdash \neg P$: nous avons complété notre théorie T. □

Nous pouvons donc appliquer ce théorie pour étendre T' en $\mathcal{T}h(T')$.

Construire le modèle à proprement parler

Nous avons un langage très expressif et une théorie très puissante, il est donc temps de construire notre modèle. L'idée première est de quotienter nos termes par l'égalité, en ce sens que si nos axiomes nous permettent de construire que $1+1=2$ alors $1+1$ et 2 doivent représenter le même terme. Comme nous avons montré plus tôt (cd exo 2.2.1) la relation $=$ est une relation d'équivalence. Nous allons donc définir la relation $\sim$ sur nos termes clos (dont nous noterons l'ensemble $\mathcal{C}$) par

$$t \sim t' \iff \mathcal{T}h(T') \vdash t = t'$$

(remarquons que par exemple le témoin de $\exists x.x = x$ montre que notre ensemble n'est pas vide). Pour un terme t, on note $\overline{t}$ sa classe. On définit alors pour notre modèle :

- le domaine $|\mathcal{M}| = \mathcal{C}/_{\sim}$
- pour interpréter une fonction f d'arité p, on définit $f^{\mathcal{M}}(\overline{t_1}, \dots, \overline{t_p}) = \overline{f(t_1, \dots, t_p)}$. Cela signifie évidemment que pour une constante c, on définit $c^{\mathcal{M}} = \overline{c}$. Bien sûr, il faut vérifier que cette interprétation est bien définie, c'est-à-dire que la valeur de notre fonction ne dépend pas des représentants choisis, nous allons montrer par récurrence sur k que si pour tout i, $\mathcal{T}h(T') \vdash t_i = u_i$, alors

$$\mathcal{T}h(T') \vdash f(t_1, \dots, t_p) = f(u_1, , \dots, u_k, t_{k+1}, \dots, t_p)$$

  — Dans le cas de base, par $=_{\text{i}}$ on a directement $\mathcal{T}h(T') \vdash f(t_1, \dots, t_p) = f(t_1, \dots, t_p)$.
  — Si l'on suppose que $\mathcal{T}h(T') \vdash f(t_1, \dots, t_p) = f(u_1, , \dots, u_k, t_{k+1}, \dots, t_p)$, alors en utilisant $=_{\text{e}}$ et sachant que $\mathcal{T}h(T') \vdash t_{k+1} = u_{k+1}$, on en déduit que $\mathcal{T}h(T') \vdash f(t_1, \dots, t_p) = f(u_1, , \dots, u_k, u_{k+1}, t_{k+2}, \dots, t_p)$

  Ce qui montre notre résultat par récurrence finie. Donc notre interprétation $f^{\mathcal{M}}$ est bien définie.
- Si r est un symbole de relation d'arité p, alors on définit directement $(\overline{t_1}, \dots \overline{t_p}) \in r^{\mathcal{M}} \iff \mathcal{T}h(T') \vdash r(t_1, \dots, t_p)$.

Montrons un dernier lemme avant d'aboutir à la complétude de notre système :

Lemme 2.4.12 (Modèle et théorie)**.** *Soit P une formule à p variable libre et $t_1, \dots, t_p$ des termes clos. Alors $\mathcal{M} \models F(\overline{t_1}, \dots, \overline{t_p})$ si et seulement si $\mathcal{T}h(T') \vdash P(t_1, \dots, t_p)$*

Démonstration. La preuve se faire par induction sur P. Nous ne traiterons pas tous les cas, mais seulement des cas représentatifs que sont $\bot$, les relations, $\neg$, $\vee$ et $\exists$. On pourrait justifier que toute formule est équivalence à une formule constituée seulement de ces constructeurs, mais le lecteur assidu pourra vérifier l'induction dans les autres cas.

- Cas $\bot$: si $\mathcal{M} \models \bot$, on arrive à une contradiction. De même, comme $\mathcal{T}h(T')$ est par hypothèse cohérente, elle ne prouve pas $\bot$, donc l'équivalence est vérifiée dans ce cas.
- Cas $P = r(u_1, \dots, u_k)$ où les u_i sont des termes non nécessairement clos. Alors on va nommer $v_i := u_i[x_1, \dots, x_p/t_1, \dots, t_p]$ qui sont, eux des termes clos. Par définition, $P(t_1, \dots, t_p) = r(v_1, \dots, v_k)$ et par définition de la substitution. Par définition de Val, on sait que

$$\mathcal{M} \models P(\overline{t_1}, \dots, \overline{t_p}) \iff \mathcal{M} \models r(\overline{v_1}, \dots, \overline{v_k})$$

  ce qui est, comme on parle ici de modéliser une relation, équivalent de façon définitionnelle à $\mathcal{T}h(T') \vdash r(v_1, \dots, v_p)$, ce qui est bien équivalent à $\mathcal{T}h(T') \vdash r(u_1, \dots, u_p)[x_1, \dots, x_p/t_1, \dots, t_p]$.
- Cas $P = \neg Q$: on considère $t_1 \dots, t_p$ des termes clos. Alors $\mathcal{M} \models \neg Q(t_1, \dots, t_p)$ est équivalent à $\text{Val}_{\mathcal{M}}(Q(t_1, \dots, t_p)) = 0$, ce qui par hypothèse d'induction est équivalent à $\mathcal{T}h(T') \nvdash Q(t_1, \dots, t_p)$ donc à $\mathcal{T}h(T') \vdash \neg Q(t_1, \dots t_p)$.
- Cas $P = Q \vee R$: d'abord, $Q \vee R(t_1, \dots, t_p) = (Q(t_1, \dots, t_p)) \vee (R(t_1, \dots t_p))$ que nous abrégerons par $Q' \vee R'$. Par disjonction de cas et analyse des règle, on remarque que pour avoir $\text{Val}_{\mathcal{M}}(P(\overline{t_1}, \dots, \overline{t_p}) \vee Q(\overline{t_1}, \dots, \overline{t_p})) = 1$, il soit avoir soit $\mathcal{M} \models P(\overline{t_1}, \dots, \overline{t_p})$ auquel cas l'hypothèse d'induction nous donne que $\mathcal{T}h(T') \vdash P'$ et donc $\mathcal{T}h(T') \vdash P' \vee Q'$, soit l'autre cas avec Q. Pour que la valeur soit 0, il faudrait que $\text{Val}_{\mathcal{M}}(P(\overline{t_1}, \dots, \overline{t_p})) = 0$ et $\text{Val}_{\mathcal{M}}(Q(\overline{t_1}, \dots, \overline{t_p})) = 0$, ce qui signifie par hypothèse d'induction que $\mathcal{T}h(T') \nvdash P'$ et $\mathcal{T}h(T') \nvdash Q'$, ce qui nous donne bien une équivalence entre les deux.
- Cas $P = \exists x.Q$: soient $t_1, \dots, t_p$ des termes clos. $\mathcal{M} \models \exists x.Q(\overline{t_1}, \dots, \overline{t_p})$ est équivalent à dire qu'il existe un terme clos t (par définition du domaine de $\mathcal{M}$) et $\mathcal{M} \models Q(\overline{t_1}, \dots, \overline{t_p}, \overline{t})$ ce qui est équivalent, par hypothèse d'induction, à ce que $\mathcal{T}h(T') \vdash G(t_1, \dots, t_p, t)$, ce qui est équivalent par la propriété du témoin et la règle $\exists_{\text{i}}$ à $\mathcal{T}h(T') \vdash G(t_1, \dots, t_p)$.

Cette induction conclut le lemme. □

Remarque 2.4.3. On voit l'importance des témoins de Henkin en particulier à la fin, où il faut pouvoir avoir une équivalence sur le $\exists$. De plus, le fait d'avoir considéré une théorie complète nous a permis de gérer le cas $\neg$.

Proposition 2.4.4. $\mathcal{M} \models \mathcal{T}h(T')$

Démonstration. Toute formule close P dans $\mathcal{T}h(T')$, par la règle Ax, est telle que $\mathcal{T}h(T') \vdash P$, donc $\mathcal{M} \models P$. □

Théorème 2.4.4. *Si T est telle que $T \nvdash \bot$, alors il existe $\mathcal{M}$ une $\mathcal{L}$-structure telle que $\mathcal{M} \models T$.*

Démonstration. On a donc, à force de constructions, prouvé qu'il existait $\mathcal{M}$ un modèle de $\mathcal{T}h(T')$, or $T \subseteq T' \subseteq \mathcal{T}h(T')$ donc pour toute proposition $P \in T$, $\mathcal{M} \models P$, ce qui signifie que $\mathcal{M} \models T$. □

Théorème 2.4.5 (Unification de la logique du premier ordre)**.** *Ainsi les deux relation $\vdash$ et $\models$ coïncident pour les formules du calcul du précidat du premier ordre.*

2.5 Un peu de théorie des modèles

Maintenant que nous avons la correction et la complétude de notre logique, nous allons étudier les résultats les plus important de la théorie des modèles, ainsi que leur conséquences.

2.5.1 Théorème de compacité

D'abord, le théorème de compacité syntaxique énoncé plus tôt, et l'équivalence entre $\vdash$ et $\models$ nous permet de déduire le théorème important suivant :

Théorème 2.5.1. *Soit Γ un ensemble de formules closes. Alors Γ possède un modèle si et seulement si toute partie finie $\Gamma_0 \subseteq \Gamma$ possède un modèle.*

De façon équivalente, Γ ne possède pas de modèle si et seulement s'il existe une partie finie $\Gamma_0 \subseteq \Gamma$ qui ne possède ne pas de modèle.

Démonstration. Dans un sens, si $\mathcal{M} \models \Gamma$ alors $\mathcal{M} \models \Gamma_0$ pour toute partie finie Γ_0. Dans l'autre sens, comme posséder un modèle est équivalent à être cohérent, supposons que $\Gamma_0 \nvdash \bot$ pour toute partie finie Γ_0 : alors par théorème de complétude syntaxique, si $\Gamma \vdash \bot$ alors on trouverait un Γ_0 tel que $\Gamma_0 \vdash \bot$, donc $\Gamma \nvdash \bot$, donc il existe un modèle de Γ.

La formulation équivalente est juste l'équivalence des négations. □

Ce théorème est essentiel pour construire des modèles. Donnons un exemple avec le résultat suivant :

Proposition 2.5.1. *Si T est une théorème admettant des modèles finis arbitrairement grands, c'est-à-dire que pour tout $n \in \mathbb{N}$ il existe un modèle $\mathcal{M} \models T$ tel que $\#|\mathcal{M}| \geq n$, alors T admet un modèle infini.*

Démonstration. D'abord, considérons la famille de formules

$$\varphi_n := \exists x_1, \ldots, \exists x_n. \bigwedge_{k=1}^{n} \bigwedge_{p=1}^{k-1} \neg(x_k = x_p)$$

qui signifie « il y a au moins n éléments ». Alors on considère $T' = T \cup \left(\bigcup_{n \in \mathbb{N}} \varphi_n \right)$ et on veut montrer qu'il existe un modèle de T', qui serait alors un modèle infini de T.

Soit $\Gamma \subseteq T'$ un ensemble fini de formules. Il contient un nombre fini de formules de la forme φ_n donc on peut trouver $i \in \mathbb{N}$ tel que toutes les formules φ_k pour $k > i$ n'appartiennent pas à Γ. Cela signifie qu'un modèle de T de taille supérieure à i est un modèle de Γ. Donc toute partie finie de T' a un modèle, donc T a un modèle infini. □

Exercice 2.5.1. Montrer qu'il n'existe pas de théorie des groupes finis (du premier ordre).

2.5.2 Théorème de Löwenheim-Skolem

Remarquons aussi un point important : la construction que nous avons faite avec les témoins de Henkin nous fournit un modèle $\mathcal{M}$ pour une théorie T cohérente mais dont le cardinal est le maximum entre celui de $\mathcal{L}$ et $\aleph_0$ (nous étudierons les cardinaux plus en détail plus tard, mais $\aleph_0$ est le cardinal de $\mathbb{N}$, i.e. qu'être de cardinal $\aleph_0$ signifie être dénombrable), c'est-à-dire que si $\mathcal{L}$ est fini ou dénombrable, alors $\mathcal{M}$ est un modèle dénombrable de T. Cela nous pousse à définir le théorème suivant :

Théorème 2.5.2 (Löwenheim-Skolem descendant)**.** *Soit $\mathcal{M}$ un modèle infini sur un langage $\mathcal{L}$. Alors il existe un modèle $\mathcal{N}$ de cardinal $\max(|\mathcal{L}|, \aleph_0)$ de $\mathcal{M}$ qui lui est équivalent au sens où pour toute proposition P dans le langage $\mathcal{L}$, on a $\mathcal{M} \models P$ si et seulement si $\mathcal{N} \models P$.*

Démonstration. Comme nous l'avons dit, nous pouvons effectuer la construction sur n'importe quelle théorie cohérente. De plus, nous avons défini plus tôt $\mathcal{T}h(\mathcal{M})$ l'ensemble des propositions vraies sur $\mathcal{M}$, qui est une théorie complète. Alors en appliquant notre construction sur $\mathcal{T}h(\mathcal{M})$ on a construit un modèle $\mathcal{N}$ de $\mathcal{T}h(\mathcal{M})$ du cardinal voulu. De plus, si $\mathcal{N} \models P$ alors par les résultats précédent, cela signifie que $\mathcal{T}h(\mathcal{M}) \vdash P$ (car P est écrit dans $\mathcal{L}$), c'est-à-dire $\mathcal{M} \models P$. □

Remarque 2.5.1. Nous n'avons pas utilisé l'hypothèse que $\mathcal{M}$ est infini, mais le théorème de Löwenheim-Skolem a pour but de changer le cardinal d'un modèle sans en modifier ses propriétés logiques. Ainsi, nous avons appelé ce résultat-ci le résultat descendant car il nous donne une borne inférieure de la taille d'un modèle logiquement équivalent à notre modèle $\mathcal{M}$. Nous allons ensuite prouver la version ascendante, qui utilise le fait que $\mathcal{M}$ est infini : nous aurons donc ensuite un théorème général qui fonctionne pour tous les cardinaux souhaités.

Théorème 2.5.3 (Théorème de Löwenheim-Skolem)**.** *Soit $\mathcal{M}$ un modèle infini sur un langage $\mathcal{L}$, et κ un cardinal supérieur au cardinal de $\mathcal{L}$. Alors il existe un modèle $\mathcal{N}$ de cardinal κ équivalent à $\mathcal{M}$, au sens où pour tout P écrit dans $\mathcal{L}$, $\mathcal{M} \models P$ si et seulement si $\mathcal{N} \models P$.*

Démonstration. Soit K de cardial κ, alors on construit pour chaque élément $i \in K$ une constante c_i et pour $i, j \in K$ une proposition $\varphi_{i,j} := \neg(c_i = c_j)$. Alors la théorie $\mathcal{T}h(\mathcal{M}) \cup \bigcup_{i,j \in K} \varphi_{i,j}$ a $\mathcal{M}$ induit un modèle pour toute partie finie (comme $\mathcal{M}$ est infinie, on peut toujours trouver assez d'éléments différents à assigner aux c_i pour qu'ils soient différents, étant donné qu'on n'a qu'un nombre fini de c_i à assigner, tous les autres pouvant valoir la même valeur). Il existe donc un modèle $\mathcal{N}$ de cardinal κ tel que $\mathcal{N} \models \mathcal{T}h(\mathcal{M})$, donc si $\mathcal{M} \models P$ alors $\mathcal{N} \models P$. Dans le sens inverse, si $\mathcal{N} \models P$, comme P est écrit dans $\mathcal{L}$, soit $\mathcal{M} \models P$ soit $\mathcal{M} \models \neg P$ mais le second cas est impossible car cela impliquerait que $\mathcal{N} \models \neg P$. □

Remarque 2.5.2. Des conséquences particulièrement contre-intuitives arrivent alors :

- Il existe un modèle de la théorie de $\mathbb{R}$ en tant que corps ordonné (c'est-à-dire muni de ses loi usuelles et de $\leq$ défini comme habituellement) qui est dénombrable.
- Il existe un modèle de ZF, la théorie des ensembles, qui est dénombrable, et dans laquelle on peut démontrer que $\mathbb{R}$ est indénombrable.

On appelle cela le *paradoxe de Skolem*, mais il ne faut pas s'y tromper : il n'y a aucune réelle contradiction ici, car il ne faut pas confondre le cardinal du modèle et la notion de cardinal définie dans la théorie. Moralement, un modèle dénombrable de $\mathbb{R}$ signifie que l'on ne s'intéresse qu'aux nombres que l'on peut nommer, décrire par une proposition : la plupart des nombres de $\mathbb{R}$ ne seront jamais explicités, et nous ne construirons toujours qu'un

nombre fini de nombres explicitement, donc pour notre usage (avec des arbres finis, des propositions finis et qui plus est sur un langage fini), les infinis ne font aucune différence.

Nous allons en voir une application possible dans le théorème de Łos-Vaught.

Définition 2.5.1 (Théorie κ-catégorique)**.** *On dit qu'une théorie T est κ-catégorique pour un cardinal κ donné s'il n'existe qu'un modèle de T de cardinal κ à isomorphisme près.*

Théorème 2.5.4 (Łos-Vaught)**.** *Une théorie T dans un langage $\mathcal{L}$ sans modèle fini, κ-catégorique où $\kappa \geq \max(|\mathcal{L}|, \aleph_0)$ est complète.*

Démonstration. Soit P une proposition écrite dans $\mathcal{L}$. Soit un modèle $\mathcal{M} \models T$, alors par Löwenheim-Skolem on construit un modèle équivalent $\mathcal{M}'$ de cardinal κ. Comme il n'existe qu'un modèle de cardinal κ, alors $\mathcal{M}$ est équivalent à ce modèle (que nous appellerons $\mathcal{U}$). Par définition, $\mathcal{U} \models P$ ou $\mathcal{U} \models \neg P$, donc soit $\mathcal{M} \models P$ soit $\mathcal{M} \models \neg P$, ce qui signifie par complétude que soit $T \vdash P$ soit $T \vdash \neg P$: T est donc complète. □

Exercice 2.5.2. Soit $\mathcal{L} = \{=, \sim\}$ un langage où l'on a seulement les deux symboles de relation binaire $=$ et $\sim$.

- Définir des propositions exprimant que $\sim$ est une relation d'équivalence.
- On va nommer $T_\sim$ la théorie décrivant que $\sim$ est une relation d'équivalence. Trouver une famille de proposition (T_i) telle qu'un modèle de $T_\sim \cup \bigcup_i T_i$ est exactement un ensemble où $\sim$ possède une infinité de classes d'équivalences.
- Trouver de même une théorie T dont les modèles sont les ensembles munis d'une relation d'équivalence $\sim$ telle qu'il y a une infinité de classes d'équivalences, où chacune possède une infinité d'éléments.
- Montrer que T est complète. *Indication : on montrera d'abord que cette théorie est $\aleph_0$-catégorique.*

Chapitre 3

Calculabilité

Une part essentielle de la logique, qui s'est largement développée pendant le XX^{e} siècle, est la théorie de la calculabilité. On pourrait considérer cette théorie comme la fondation de la notion d'algorithme, qui n'avais jusqu'alors pas vraiment de définition rigoureuse (et on continue en général de considérer qu'un algorithme est « une suite de tâches simples, applicables par une machine »). La thèse de Church-Turing dit, en substance, que la bonne notion de ce qu'est un algorithme, est représenté par l'un des trois formalismes équivalents suivant : les machines de Turing, les fonctions récursives et le lambda-calcul (il existe de nombreux autres formalisme équivalents mais ces trois-là sont les fondamentaux). Nous dédierons une partie entière au lambda-calcul pour des raisons liées à la logique, et nous allons donc traiter les deux autres formalismes. Plus précisément, ce chapitre s'organisera tout d'abord en menant aux machines de Turing par le biais des automates finis puis des automates à pile, avant de parler à proprement parler de machine de Turing et des principaux théorèmes dessus. Nous verrons enfin les fonctions récursives et leur équivalence avec les machines de Turing.

La calculabilité s'occupe de deux cas distincts : décider ou calculer. Le premier cas concerne celui où l'on considère une propriété sur des objets, et que l'on veut pouvoir décider de si l'objet en question a cette propriété (par exemple savoir si un entier est pair). Le deuxième cas, plus général, consiste à définir un processus permettant d'effectuer des opérations sur une entrée pour en obtenir une sortie : cela correspond exactement à l'intuition que l'on a d'une fonction, et l'on traitera donc de fonctions. On peut facilement utiliser le second cas pour parler du premier cas en calculant la fonction caractéristique de l'ensemble des éléments vérifiant la propriété, mais la décision sera notre première façon d'aborder la calculabilité, aussi la considérerons-nous pendant plus longtemps.

3.1 Automates finis et langages

3.1.1 Mots et langages

Avant de nous intéresser à proprement parler aux automates ou autres machines, nous allons devoir formaliser ce dont il sera question dans ce chapitre. En reprenant ce qui a été dit plus tôt, il nous faut formaliser la notion d'objet, et celle de propriété sur un objet. En théorie des ensembles, un objet est simplement un ensemble comme un autre, et une propriété sur un ensemble est simplement une propriété du calcul des prédicats comme nous l'avons vu. Mais d'un point de vue d'informaticien, un objet est en général codé, et c'est ce point de vue que nous allons adopter. Nous allons donc nous intéresser, comme objets, à des « codes d'objets », qui seront plus précisément des mots :

Définition 3.1.1 (Mot sur un alphabet)**.** *Soit Σ un ensemble fini non vide que l'on appellera alphabet. On dit que u est un mot sur Σ si u est une suite finie d'éléments de Σ, c'est-à-dire qu'il existe n tel que $u \in \Sigma^n$. L'ensemble des mots sur Σ est noté Σ^*, et est donc défini par*

$$\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$$

en prenant la convention que $\Sigma^0 = \{\varepsilon\}$ où ε est appelé le mot vide. Pour un mot $u \in \Sigma^*$, on appellera sa longueur l'entier n tel que $u \in \Sigma^n$, et on la notera $|u|$. De plus, on notera u_i pour la i^e composante du mot u.

Par convention, les indices vont de 0 *à* $|u| - 1$.

Exemple 3.1.1. Soit l'alphabet $\Sigma = \{0, 1\}$. Un mot sur Σ est par exemple 0010 (on notera ainsi, plutôt que $(0, 0, 1, 0)$, les mots).

Notre formalisme repose donc sur l'étude des mots. Nous allons donc définir l'opération essentielle sur les mots.

Définition 3.1.2 (Concaténation)**.** *On définit l'opération de concaténation, qui à un mot u et un mot v associe le mot $u \cdot v$ défini par*

$$(u \cdot v)_i = \begin{cases} u_i & si\ i < |u| \\ v_{|u|-i} & sinon. \end{cases}$$

Cette opération est définie pour tous $n, m \in \mathbb{N}$ comme $\cdot : \Sigma^n \times \Sigma^m \to \Sigma^{n+m}$ et peut donc directement se prolonger en une loi de composition interne sur Σ^*.

Exemple 3.1.2. Toujours sur l'alphabet $\Sigma = \{0, 1\}$, la concaténation nous donne $0010 \cdot 100 = 0010100$.

Exercice 3.1.1 (Les mots forment un monoïde)**.** Montrer que $\cdot$ est une loi associative, c'est-à-dire que $(u \cdot v) \cdot w = u \cdot (v \cdot w)$ pour tous $u, v, x \in \Sigma^*$. Montrer de plus que ε est neutre pour $\cdot$, c'est-à-dire que $u \cdot \varepsilon = \varepsilon \cdot u = u$ pour tout $u \in \Sigma^*$.

Remarque 3.1.1. On aurait pu définir l'ensemble des mots sur Σ de façon plus algébrique en considérant ce qu'on appelle le monoïde libre sur Σ, mais ça n'est pas le point de vue le plus adapté dans notre étude actuelle car nous ne parlerons pas de morphisme de monoïde.

Nous en venons maintenant à chercher comment définir une propriété sur ces mots : la façon la plus naturelle est de regarder tous les mots vérifiant la propriété, d'où la définition suivante :

Définition 3.1.3 (Langage)**.** *On appelle un langage sur Σ une partie $L \subseteq \Sigma^*$. C'est donc un ensemble de mots, et on peut le voir comme la description d'une propriété : au lieu de dire qu'un mot u a une certaine propriété P, on associera à P un langage L et on dira directement que $u \in L$.*

Nous allons maintenant définir les opérations basiques associées aux langages :

Définition 3.1.4 (Opérations sur les langages)**.** *Soient L et L' deux langage. On définit les opérations suivantes :*

- $L \cdot L' = \{u \cdot v \mid u \in L, v \in L'\}$
- $L \cup L'$ *que l'on notera souvent* $L + L'$.
- L^n *défini par* $L^0 = \{\varepsilon\}$ *et* $L^{n+1} = L \cdot L^n$.
- $L^* = \bigcup_{n \in \mathbb{N}} L^n$
- $L^c = \Sigma^* \setminus L$

Exemple 3.1.3. Soit $L = \{0, 01\}$ et $L' = \{\varepsilon, 100\}$, alors $L \cdot L' = \{0, 01, 0100, 01100\}$, $L + L' = \{0, 01, \varepsilon, 100\}$, L^c est l'ensemble des mots qui ne sont ni 0 ni 01 et L^* est l'ensemble des mots qui s'écrivent $u_1 \cdot u_2 \cdots u_p$ où $u_i \in L$ pour $1 \leq i \leq p$, par exemple $0010001 = 0 \cdot 01 \cdot 0 \cdot 0 \cdot 01 \in L^*$.

3.1.2 Définition d'un automate déterministe et indéterministe

Intuitivement, donnons-nous le langage L sur $\{0,1\}$ des mots qui sont une alternance de 0 et de 1. Quel procédé pourrions-nous trouver, étant donné un mot u, pour savoir si $u \in L$? Par procédé nous cherchons un objet mathématique qui pourrait trier pour nous les mots appartenant à L ou non. Dans cette section, nous nous intéresserons à la construction la plus basique de ces objets, appelés automates. Les automates prennent donc un mot en entrée et, par une successions d'actions (définies purement mathématiquement) répondent si le mot est dans le langage L ou non. La première approche est de considérer le mot lettre par lettre, et de « lire » linéairement le mot, en stockant une information au maximum à la lecture du mot. Cela nous mène à la définition suivante :

Définition 3.1.5 (Automate fini déterministe)**.** *Soit Σ un alphabet, un automate $\mathcal{A}$ est la donnée d'un quintuplet $(\Sigma, Q, q_0, \delta, F)$ où :*

- *Σ est l'alphabet sur lequel on travaille.*
- *Q est un ensemble d'états fini non vide.*
- *$q_0 \in Q$ est l'état initial de l'automate.*
- *$\delta : \Sigma \times Q \to Q$ est la fonction de transition (considéré partielle), disant après la lecture d'une lettre dans quel état doit se rendre l'automate.*
- *$F \subseteq Q$ est l'ensemble des états finaux.*

Si la définition peut faire peur, elle revient généralement à dessiner un graphe où les états sont les sommets et les arcs (orientés) sont étiquetés par des lettres. On note une flèche entrante pour désigner l'état initial, et on entoure en double les états finaux. Nous donnons une présentation d'un automate dans la figure 3.1 suivante :

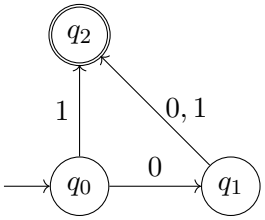


Figure 3.1 – Un automate $\mathcal{A}_0$.

On a donc $\Sigma = \{0,1\}$, $Q = \{q_0, q_1, q_2\}$, $F = \{q_2\}$ et la fonction de transition décrite par

$$\begin{aligned}\delta(0, q_0) &= q_1\\ \delta(1, q_0) &= q_2\\ \delta(0, q_1) &= q_2\\ \delta(1, q_1) &= q_2\\ \delta(\alpha, q_2) &= \bot\end{aligned}$$

où $\bot$ signifie que la fonction n'est pas définie sur ce couple.

On remarque rapidement que si l'on arrive à un état, on peut réappliquer la fonction de transition depuis cet état si l'on a une nouvelle lettre : ce sera le fonctionnement de notre automate.

Définition 3.1.6 (Langage accepté par un automate déterministe)**.** *Soit $\mathcal{A}$ un automate fini déterministe. On étend par récurrence la fonction de transition sur Σ^* :*

- *$\delta^*(\varepsilon, q) = q$ pour tout état $q \in Q$*
- *$\delta^*(\alpha \cdot v, q) = \delta(\alpha, \delta^*(v, q))$ pour α une lettre et v un mot.*

On notera aussi, en général, $q \cdot u$ pour désigne $\delta^*(u, q)$, nous donnant dont $q \cdot (\alpha \cdot v) = (q \cdot \alpha) \cdot v$, ce qui est plus lisible.

On dit que $\mathcal{A}$ accepte un mot u si $q_0 \cdot u \in F$, et on note alors $\mathcal{A}(u) = 1$. Si $q_0 \cdot u$ n'est pas défini (qu'on arrive sur une transition non définie, comme dans l'exemple avec $\perp$) ou que $q_0 \cdot u \notin F$, on dit que $\mathcal{A}$ rejette le mot u et on note alors $\mathcal{A}(u) = 0$.

On définit par $L(\mathcal{A})$ l'ensemble des mots acceptés par $\mathcal{A}$:

$$L(\mathcal{A}) = \{u \in \Sigma^* \mid \mathcal{A}(u) = 1\}$$

De façon équivalente, qu'un mot u soit accepté par $\mathcal{A}$ peut s'écrire comme le fait qu'il existe une suite $q_0, \dots, q_p$ où $q_p \in F$ et $q_{i+1} = \delta(q_i, u_i)$. Ces automates sont dits déterministes car cette suite est unique, lors de la lecture d'un mot. Nous allons donc introduire la notion d'automate non déterministe :

Définition 3.1.7 (Automate fini non déterministe)**.** *Un automate fini non déterministe est un quintuplet $(\Sigma, Q, Q_0, \delta, F)$ où Σ, Q et F sont identiques au cas déterministe, mais $Q_0 \subseteq Q$ et $\delta : \Sigma \times Q \to \mathcal{P}(Q)$. Le changement sur Q_0 signifie simplement que l'on s'autorise plusieurs états initiaux, mais le changement sur delta mérite plus d'explications : l'image d'une transition est un ensemble d'états, on peut voir cela comme « les états qui peuvent arriver après la transition ».*

On peut ensuite définir ce qu'est le langage reconnu par un automate non déterministe :

Définition 3.1.8. *Soit $\mathcal{A}$ un automate non déterministe, on dit que u est reconnu par $\mathcal{A}$ s'il existe une suite $q_0, \dots, q_p$ où $q_0 \in Q_0$, $q_{i+1} \in \delta(u_i, q_i)$ et $q_p \in F$.*

On peut alors définir de façon analogue le langage reconnu par $\mathcal{A}$, que l'on notera $L(\mathcal{A})$.

Les automates non déterministes sont beaucoup plus pratiques à manipuler, mais leur expressivité est la même que les automates déterministes :

Théorème 3.1.1. *Soit L un langage sur un alphabet Σ. L est le langage reconnu par un automate déterministe si et seulement s'il est le langage reconnu par un automate non déterministe.*

Démonstration. Un des sens est évident : on peut facilement transformer un automate déterministe en un automate non déterministe en fixant $Q_0 = q_0$ et $\delta'(\alpha, q) = \{q \cdot \alpha\}$.

Nous allons donc prouver le sens réciproque : soit $\mathcal{A}$ non déterministe dont le langage reconnu est L, construisons $\mathcal{A}'$ déterministe tel que $L(\mathcal{A}) = L(\mathcal{A}')$. En notant $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ on va construire notre automate $\mathcal{A}' = (\Sigma, Q', q_0, \delta', F')$ de la façon suivante :

- $Q' = \mathcal{P}(Q)$: l'ensemble des états de $\mathcal{A}'$ est l'ensemble des parties des états de $\mathcal{A}$.
- $q_0 = Q_0$ puisque $Q_0 \in \mathcal{P}(Q)$.
- $\delta'(\alpha, p) = \bigcup_{q \in p} \delta(\alpha, q)$
- $F' = \{P \subseteq Q \mid P \cap F \neq \varnothing\}$

Montrons qu'alors $\mathcal{A}(u) = \mathcal{A}'(u)$:

- Supposons que $\mathcal{A}$ reconnaisse u, alors il existe une suite $q_0, \dots, q_p$ telle que $q_0 \in Q_0$, pour chaque i $q_{i+1} \in \delta(u_i, q_i)$ et $q_p \in F$. Dans $\mathcal{A}'$, on remarque que chaque q_i appartient à $\delta'^*(u_{<i}, Q_0)$ (en notant $u_{<i}$ le préfixe de u construit en prenant les i premières lettres). En effet par hypothèse $q_0 \in Q_0$, puis si $q_i \in \delta'^*(u_{<i}, Q_0)$, en réécrivant $\delta'^*(u_{<i+1}, Q_0)$, on obtient

$$\delta'^*(u_{<i+1}, Q_0) = \bigcup_{q \in \delta'^*(u_{<i}, Q_0)} \delta(u_i, q)$$

  mais comme $q_{i+1} \in \delta(u_i, q_i)$ on en déduit que $q_{i+1} \in \delta'^*(u_{<i+1}, Q_0)$. Enfin, comme le dernier état q_p est final, cela signifie que le dernier état de la lecture par $\mathcal{A}'$ contient un état final, ce qui signifie que c'est un état final : on en déduit que $\mathcal{A}'$ reconnaît u.

- Réciproquement, si $\mathcal{A}'$ accepte un mot u, montrons qu'il en est de même pour $\mathcal{A}$. Pour cela, on va montrer la réciproque de la propriété précédente : pour toute suite $Q_0, \dots, Q_p$ où $Q_{i+1} = \delta'(u_i, Q_i)$ et tout élément $q_k \in Q_k$ il existe une suite $q_0, \dots, q_k$ telle que $q_0 \in Q_0$ et $q_{i+1} \in \delta(u_i, q_i)$. Le cas de base, avec q_0, est évident, passons donc à l'hérédité : prenons $q_{k+1} \in Q_{k+1}$, alors $Q_{k+1} = \delta'(u_k, Q_k)$ et par hypothèse de récurrence, $q_k \in Q_k$, et on remarque que $\delta'(u_k, Q_k)$ contient $\delta(u_k, q_k)$ (puisque c'est l'un des termes de l'union), donc $q_{k+1} \in Q_{k+1}$. Par hypothèse, $\mathcal{A}'$ accepte u, ce qui signifie qu'il existe $q_f \in Q_p$ tel que $q_f \in F$, donc par le résultat que l'on vient de montrer, il existe un chemin acceptant dans $\mathcal{A}'$, donc $\mathcal{A}'$ accepte u.

On en déduit donc que l'expressivité des automates déterministes et indéterministes est la même. □

Exercice 3.1.2. Construire l'automate des parties de $\mathcal{A}_0$ donné en exemple plus haut.

Exercice 3.1.3. Considérons maintenant des automates déterministes mais où l'on peut ajouter ce que l'on appelle des « epsilon-transitions », qui sont des transitions que l'on peut effectuer pour passer d'un état à un autre sans lire de lettre. Montrer que les langages reconnus avec epsilon-transitions ou sans sont les mêmes. *Indication : on se ramènera à un automate non déterministe.*

Exercice 3.1.4. Montrer qu'un automate est toujours équivalent (au sens où il reconnaît le même langage) qu'un automate total, c'est-à-dire un automate où sa fonction de transition δ est une fonction totale.

3.1.3 Les expressions rationnelles

Nous pouvons nous interroger sur la forme des langages reconnus par des automates finis, que nous appellerons langages reconnaissables. Il se trouve qu'il existe une autre définition, équivalence, ne faisant pas intervenir les automates.

Définition 3.1.9 (Langage rationnel)**.** *Soit Σ un alphabet, on définit sur $\mathcal{P}(\Sigma^*)$ la classe des langages rationnels comme étant la plus petite partie de $\mathcal{P}(\Sigma^*)$ contenant :*

- *Le langage vide $\varnothing$.*
- *Pour chaque $\alpha \in \Sigma$, le langage $\{\alpha\}$.*
- *Le langage $\{\varepsilon\}$*
- *Pour deux langages rationnels L et L', $L + L'$ et $L \cdot L'$*
- *Pour un langage rationnel L, L^*.*

On reconnaît ici une structure inductive évidente, que l'on va expliciter à l'aide des expressions rationnelles :

Définition 3.1.10 (Expression rationnelle)**.** *On définit une expression rationnelle comme un mot sur la grammaire suivante*

$$e, e' ::= \varnothing \mid \alpha \mid \varepsilon \mid e + e' \mid e \cdot e' \mid e^*$$

et on remarque qu'on a une bijection naturelle entre ces expressions et les langages rationnels : le langage correspondant à l'expression α^* est par exemple le langage $\{\varepsilon, \alpha, \alpha\alpha, \alpha\alpha\alpha, \dots\}$. Plus précisément, on définit par induction la bijection qui à un caractère associe le langage de son singleton, qui à $\varnothing$ associe $\varnothing$ et qui applique inductivement les opérations sur les langages. On note $Rat(\Sigma)$ l'ensemble des expressions rationnelles sur Σ et $L(e)$ le langage engendré par l'expression e.

Exercice 3.1.5. Donner une expression rationnelle pour décrire les langages suivants (sur $\Sigma = \{a, b, c\}$) :

- les mots n'utilisant pas de b.
- les mots où chaque a est écrit en double.

- les mots avec un seul b.

Il se trouve qu'un langage est reconnaissable par un automate fini si et seulement s'il est rationnel. Pour prouver le premier sens, nous allons démontrer des lemmes intermédiaires.

Lemme 3.1.1. *Soient L et L' des langages sur Σ reconnus respectivement par $\mathcal{A}$ et $\mathcal{A}'$. Alors il existe un automate $\mathcal{A}''$ reconnaissant $L + L'$.*

Démonstration. Nous allons construire $\mathcal{A}''$, en prenant comme convention que chaque $Q, \delta, \ldots$ est associé à son automate par le nombre de $'$ qu'il a :

- $Q'' = \{q_0''\} \cup Q \sqcup Q'$ ($\sqcup$ représente l'union disjointe, on peut la définir comme $Q \times \{0\} \cup Q' \times \{1\}$, l'important est qu'aucun élément n'est commun aux deux parties)
- q_0'' est tout nouvel état
- $\delta''(\alpha, q) = \begin{cases} \delta(\alpha, q) & \text{si } q \in Q \\ \delta'(\alpha, q) & \text{si } q \in Q' \end{cases}$ pour les états $q \neq q_0''$, et q_0'' possède une epsilon-transition vers q_0 et q_0'
- $F'' = F \sqcup F'$

L'idée de cet automate est de faire un choix non déterministe au début : soit le mot reconnu l'est par $\mathcal{A}$, soit il l'est par $\mathcal{A}'$.

Montrons que $\mathcal{A}''$ reconnaît $L + L'$:

- si $u \in L + L'$, alors soit $u \in L$ soit $u \in L'$. Sans perte de généralité, disons que $u \in L$. Alors il existe un chemin acceptant $q_0, \ldots, q_p$ dans $\mathcal{A}$. Dans ce cas, le chemin $q_0'', q_0, \ldots, q_p$ est acceptant en utilisant une epsilon-transition au début.
- si u est reconnu par $\mathcal{A}''$ alors il existe un chemin acceptant partant de q_0''. Comme q_0'' n'est pas acceptant, il y a au moins une transition, qui est ici une epsilon-transition. Alors la suite d'état restante est soit une suite entièrement dans Q, soit entièrement dans Q', dans le premier cas $u \in L$ et dans le deuxième cas $u \in L'$, donc $u \in L + L'$.

Nous donnons l'idée de la preuve avec la figure 3.2 □

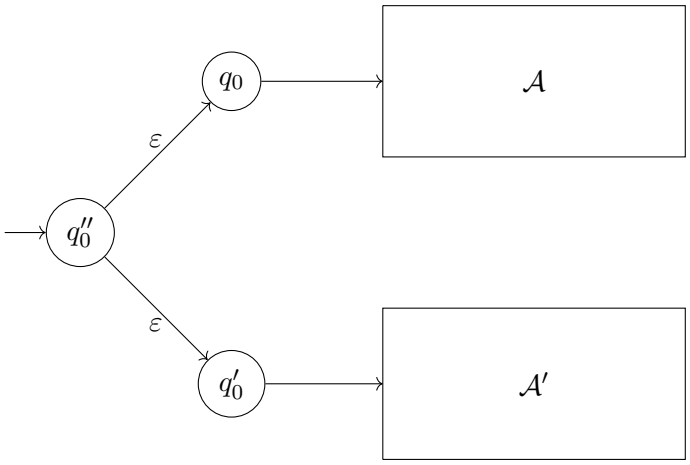


Figure 3.2 – Schéma de l'automate pour une union.

Lemme 3.1.2. *Si L et L' sont des langages sur Σ reconnus respectivement par $\mathcal{A}$ et $\mathcal{A}'$ alors il existe un automate $\mathcal{A}''$ reconnaissant $L \cdot L'$.*

Démonstration. On construit $\mathcal{A}''$ de la façon suivante :

- $Q'' = Q \sqcup Q'$
- $q_0'' = q_0$
- $\delta(\alpha, q) = \begin{cases} \delta(\alpha, q) & \text{si } q \in Q \\ \delta'(\alpha, q) & \text{si } q \in Q' \end{cases}$ avec, pour chaque élément $q_f \in F$, une epsilon transition de q_f vers q_0'.
- $F'' = F'$

L'idée de cet automate est de relier directement la fin d'un mot accepté par $\mathcal{A}$ au début d'un mot accepté par $\mathcal{A}'$.

Montrons que $\mathcal{A}''$ reconnaît bien $L \cdot L'$:

- Si $u \in L \cdot L'$ alors on réécrit $u = v \cdot w$ où $v \in L$ et $w \in L'$. Comme $v \in L$, il existe un chemin acceptant $q_0, \dots, q_p$ dans $\mathcal{A}$ pour v, et de même il existe un chemin $q_0', \dots, q_k'$ acceptant w dans $\mathcal{A}'$. Alors le chemin $q_0, \dots, q_p, q_0', \dots, q_k'$ est un chemin acceptant dans $\mathcal{A}''$, donc u est reconnu par $\mathcal{A}''$.
- Si u est reconnu par $\mathcal{A}''$, alors il existe un chemin acceptant dans $\mathcal{A}''$ $q_0, \dots, q_p$ où $q_p \in F'$. On remarque que par construction, le seul moyen de passer d'un état de Q à un état de Q' est d'utiliser une epsilon-transition, et qu'il ne peut y en avoir qu'au plus une dans la lecture du mot. On peut donc décomposer notre suite d'états en deux suites d'états, et par construction sur les conditions d'où sont placées les epsilon-transitions, la suite d'états partant de q_0 et arrivant à l'epsilon-transition est un chemin acceptant de $\mathcal{A}$, et la suite d'états partant de q_0' vers un état de F' est un chemin acceptant de $\mathcal{A}'$. Cela constitue donc la concaténation d'un moye de L et d'un mot de L', donc $u \in L \cdot L'$.

□

Exercice 3.1.6. Montrer que si L est reconnu par un automate $\mathcal{A}$, alors il existe un automate $\mathcal{A}'$ reconnaissant L^*. *Indication : on pourra placer une epsilon-transition entre les états finaux et l'état initial.*

Proposition 3.1.1. *Si L est un langage rationnel, alors il existe un automate $\mathcal{A}$ qui le reconnaît.*

Démonstration. Pour prouver cela, il nous suffit de montrer qu'il existe des automates reconnaissant les différents constructeurs d'un langage rationnel :

- Un automate avec $F = \varnothing$ n'accepte aucun mot.
- L'automate ne possédant qu'un état initial et un état final, et une transition étiquetée par α entre les deux (cf. figure 3.3) reconnaît exactement $\{\alpha\}$.
- Pour reconnaître uniquement ε, il suffit de faire un automate comme le précédent, en étiquetant toutes les transitions vers q_1, et en définissant $F = \{q_0\}$.
- Le lemme 3.1.1 nous donne le résultat.
- Le lemme 3.1.2 nous donne le résultat.
- L'exercice 3.1.6 nous donne le résultat.

Donc, puisque la classe des langages rationnels est la plus petite stable par ces règles, elle est incluse dans la classe des langage reconnaissables. □

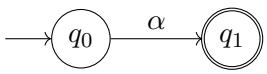


Figure 3.3 – Automate reconnaissant α uniquement.

Nous allons maintenant prouver le sens réciproque. Pour cela, nous allons introduire une nouvelle classe d'automates, que l'on va appeler automates généralisés, dont les transitions seront étiquetées par des expressions rationnelles.

Définition 3.1.11 (Automate généralisé)**.** *Un automate généralisé est un automate dont la fonction de transition est définie par* $\delta : \mathrm{Rat}(\Sigma) \times Q \to Q$ *et qui ne possède qu'un état final. Un mot* u *est accepté s'il peut être décomposé en* $u_0 \cdot u_1 \cdots u_p$ *et qu'il existe une suite* $q_0, \ldots, q_{p+1}$ *avec* q_{p+1} *final et pour tout* $i \leq p$*, il existe* $e \in \mathrm{Rat}(\Sigma)$ *telle que* $\delta(e, q_i) = q_{i+1}$ *et* $u_i \in L(e)$*. De plus, on impose que* δ *n'est pas définie depuis* q_f *l'état final et que* q_0 *n'est pas dans l'image de* δ*.*

Exercice 3.1.7. Montrer que ces automates ont la même expressivité que les autres automates finis.

Remarque 3.1.2. Grâce à cet exercice, on peut supposer que pour tout automate $\mathcal{A}$ il existe un automate généralisé $\mathcal{A}'$ qui reconnaît le même langage.

On va maintenant procéder sur l'automate généralisé en supprimant des transitions et des états jusqu'à n'obtenir qu'une seule transition, dont le label sera l'expression rationnelle associée à l'automate.

Lemme 3.1.3 (Réduction des transitions)**.** *Soit* $\mathcal{A}$ *un automate généralisé, alors s'il existe* e, e' *deux expressions et un état* q *tels que* $\delta(e, q) = \delta(e', q)$ *alors on a un automate équivalent en supprimant ces deux transitions et en ajoutant* $\delta(e + e', q)$*.*

Démonstration. Un mot reconnu par $\mathcal{A}$ dont le chemin de lecture utilise $\delta(e, q)$ utilisera de la même façon $\delta(e+e', q)$, et réciproquement si un chemin de lecture utilise $\delta(e + e', q)$ alors soit le mot u associé à cette transition est dans $L(e)$, auquel cas $\delta(e, q)$ donnera un chemin de lecture valide, soit il est dans $L(e')$ et $\delta(e', q)$ sera utilisé. □

On peut représenter cette réduction de la façon suivante :

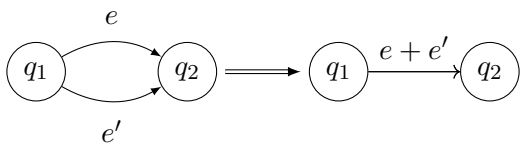


Figure 3.4 – Réduction des transitions

Ainsi, par application successive du lemme, on peut supposer que l'on travail avec des états ne possédant qu'une transition entre deux états donnés. Nous allons maintenant procéder par élimination d'états.

Lemme 3.1.4 (Élimination des états)**.** *Soit un automate généralisé* $\mathcal{A}$ *où l'on a réduit ses transitions, et* q *un état ni initial ni final. Alors on peut construire un automate* $\mathcal{A}'$ *reconnaissant le même langage en supprimant* q *et, pour chaque couple d'états* (q_1, q_2) *tel que* $\delta(e_1, q_1) = q$ *et* $\delta(e_2, q) = q_2$*, rajouter une transition de* q_1 *vers* q_2 *de la façon suivante :*

- *s'il existe* e *telle que* $\delta(e, q) = q$ *alors on définit* $\delta(e_1 \cdot e^* \cdot e_2, q_1) = q_2$
- *sinon, on définit* $\delta(e_1 \cdot e_2, q_1) = q_2$

Démonstration. Dans le cas où le chemin de lecture ne passe pas par q, le lemme est évident. Supposons que dans la suite d'états $q_0, \ldots, q_p$ lisant un mot u dans $\mathcal{A}$ se trouve q, alors soit q_1 le prédécesseur de q et q_2 son successeur (supposons d'abord que $q_2 \neq q$), dans la suite d'états. On sait qu'il existe une expression e_1 et une expression e_2 telles que $\delta(e_1, q_1) = q$ et $\delta(e_2, q) = q_2$ avec u_1, u_2 des parties du mot u tels que $u_i \in L(e_i)$. Par définition, $u_1 \cdot u_2 \in L(e_1 \cdot e_2)$ donc la suite d'états sans q est valide dans $\mathcal{A}$. Si jamais $q_2 = q$, alors on peut remplacer toutes les suites de q contiguës en disant que la transition est étiquetée par e^* avec e l'expression étiquetant la transition de q à q.

La réciproque est laissée en exercice au lecteur. □

On peut voit l'élimination d'états de la façon suivante :

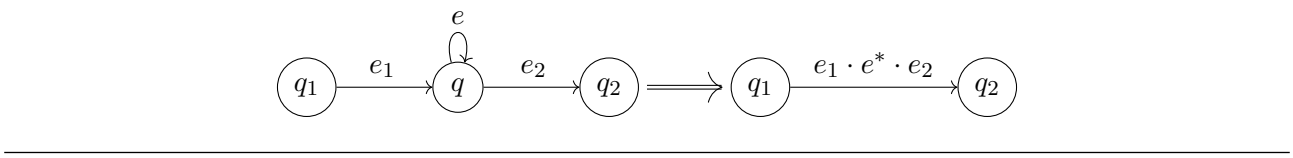


Figure 3.5 – Élimination des états

On en déduit le résultat suivant :

Proposition 3.1.2. *Pour un automate généralisé $\mathcal{A}$ il existe un automate généralisé $\mathcal{A}'$ lisant le même langage et n'ayant qu'un état initial et un état final, et une seule transition entre les deux : le langage lu par $\mathcal{A}'$ est exactement $L(e)$ pour e l'étiquette de sa seule transition.*

Démonstration. Il suffit de raisonner par récurrence sur le nombre d'états en appliquant l'élimination des états, et en réduisant les transitions qui en découlent. La dernière partie est par définition de la reconnaissance d'un mot dans un automate généralisé. □

On en déduit le théorème de Kleene :

Théorème 3.1.2 (Kleene)**.** *La classe des langages rationnels est exactement l'ensemble des langages reconnus par des automates finis.*

Démonstration. C'est une conséquence directe des propositions 3.1.1 et 3.1.2. □

3.1.4 Myhill-Nérode

Cette sous-section va s'attarder à une autre caractérisation des langages rationnels. Celle-ci est plus théorique mais présente plus facilement les limites des langages rationnels, et permet de répondre directement à la question « y a-t-il pour un langage donné, un plus petit automate acceptant ce langage ? »

Tout d'abord, présentons un lemme en général utilisé pour montrer que des langages ne sont pas rationnels :

Lemme 3.1.5 (De l'étoile)**.** *Soit un langage rationnel L, alors il existe un nombre M tel que pour tout mot $u \in L$ de longueur supérieure à M, il existe v, w, x des mots ($w \neq \varepsilon$ mais les deux autres peuvent être vides) tels que $u = v \cdot w \cdot x$ et pour tout $n \in \mathbb{N}$, $v \cdot w^n \cdot x \in L$. L'idée derrière est que pour les mots assez grands, il existe une partie du mot qui peut être répétée en restant dans le langage.*

Démonstration. L'argument est principalement combinatoire : supposons que $u \in L$ avec $|u| > |Q|$ pour Q l'ensemble d'états d'un automate $\mathcal{A}$ reconnaissant L, alors lors de la lecture de u, on a la suite $q_0, \ldots, q_{|u|-1}$ et par principe des tiroirs, on trouve $i < j < |u|$ tels que $q_i = q_j$. On décompose alors le mot u avec la partie préfixe $v := u_{<i}$, la partie infixe $w := u_{i,\ldots,j-1}$ et la partie postfixe $x := u_{j,\ldots,|u|-1}$. Dans ce cas, la lecture du mot $v \cdot w^n \cdot x$ donnera une suite d'états $q_0, \ldots, q_i$ puis n fois la suite d'états de q_i à lui-même décrite par w, puis la suite d'états $q_j, \ldots, q_{|u|-1}$ qui est un état final par hypothèse. □

Remarque 3.1.3. Ce lemme est utile pour un langage infini : pour un langage fini, la constante sera simplement supérieure au cardinal du langage. De plus, la constante M de ce lemme semble dépendre de l'automate reconnaissant le langage L, ce qui n'est pas naturel : le théorème de cette sous-section nous donnera justement une constante M optimale et dépendant directement de L.

Exercice 3.1.8. En utilisant le lemme de l'étoile, montrer que le langage $\{0^n 1^n \mid n \in \mathbb{N}\}$ n'est pas rationnel.

La notion principale pour pouvoir analyser les langages rationnels est celle de séparation des mots. L'idée derrière est simple : prenons deux mots u et v, on peut les étendre naturellement en leur ajoutant un mot z par concaténation, donnant $u \cdot z$ et $v \cdot z$. Dans ce cas, peut-on avoir un mot z qui mène, depuis u, à un mot de L, et depuis v à un mot qui n'est pas dans L ? Cette notion motive notre prochaine définition :

Définition 3.1.12 (Résiduel d'un langage)**.** *Soit x un mot, on dénote par $x^{-1}L$ l'ensemble*

$$x^{-1}L = \{u \in \Sigma^* \mid x \cdot u \in L\}$$

appelé le résiduel de L par x.

De façon équivalente, on peut définir la relation d'équivalence d'inséparabilité : $x \sim y$ lorsque pour tout mot $u \in \Sigma^*$, $x \cdot u \in L \iff y \cdot u \in L$. Les classes d'équivalence pour cette relation sont les résiduels.

Plusieurs observations s'ensuivent :

- $L = \varepsilon^{-1}L$
- pour tout automate sur un langage L fixé on peut associer à chaque état q un résiduel de L.

Ces observations permettent d'imaginer les résiduels de L comme un automate, lorsque le nombre de ses résiduels est fini. Le théorème de Myhill-Nérode stipule justement que ce nombre de résiduels est fini si et seulement si L est rationnel.

Théorème 3.1.3 (Myhill-Nérode)**.** *Soit L un langage, alors L est rationnel si et seulement s'il n'existe qu'un nombre fini de résiduels de L.*

Démonstration. Avec les observations antérieures, en supposant qu'il n'existe qu'un nombre fini de résiduels, on construit l'automate $\mathcal{A}$ suivant :

- $Q = \{x^{-1}L \mid x \in \Sigma^*\}$
- $q_0 = \varepsilon^{-1}L$
- $\delta(\alpha, x^{-1}L) = (x \cdot \alpha)^{-1}L$ (la fonction est bien définie car si $x \sim y$ alors $x \cdot \alpha \sim y\dot{\alpha}$)
- $x^{-1}L$ est final s'il contient ε.

Si $x \in L$, alors $x^{-1}L$ contient ε par construction, et s'il existe une suite d'états $q_0, \ldots, q_p$ menant à un état final dans cet automate, alors le dernier état est de la forme $x^{-1}L$ pour x le mot lu et $\varepsilon \in x^{-1}L$ donc $x \cdot \varepsilon \in L$, c'est-à-dire $x \in L$. L est donc reconnu par cet automate.

Supposons que L est rationnel. Alors soit $\mathcal{A}$ un automate reconnaissant L. Pour chaque état q, on peut définir l'ensemble des mots menant de q_0 à q : notons-le P_q. Alors pour deux mots $u, v \in P_q$, si $u \cdot z \in L$ alors $v \cdot z \in L$ car la lecture en partant de q et en lisant les lettres de z mène à un état final. Donc pour chaque état q, $P_q \subseteq x^{-1}L$ pour un certain x (dépendant de q). On en déduit qu'il y a plus d'états que de résiduels, et comme l'automate est fini, qu'il y a un nombre fini de résiduels. □

Remarque 3.1.4. La construction avec les résiduels nous donne donc aussi un automate minimal au sens du cardinal pour reconnaître un langage donné. Ainsi le lemme de l'étoile peut se reformuler en remplaçant « il existe M » en remplaçant directement M par le nombre de résiduels de L.

Exercice 3.1.9. Redémontrer que le langage $\{0^n 1^n \mid n \in \mathbb{N}\}$ n'est pas rationnel, cette fois-ci en utilisant le théorème de Myhill-Nérode.

Exercice 3.1.10. L'ensemble des langages rationnels est-il stable par complémentaire ? Par intersection ?

Exercice 3.1.11. Le langage des codes binaires de nombres multiples de 6 est-il rationnel ?

Exercice 3.1.12. Si L est un langage rationnel, est-ce que le langage $\overline{L}$ des mots de L à l'envers est rationnel ? Le langage des palindromes (mots qui sont les mêmes à l'envers) est-il rationnel ?

La théorie des automates finis est bien plus vaste que le survol proposé ici : on peut par exemple parler de l'extension de ces automates aux mots infinis avec les automates de Büchi, ou encore de la détermination de l'arithmétique de Presburger avec des automates (ce dernier sujet aurait tout à fait sa place dans cet ouvrage dédié à la logique, mais comme l'objectif de ce chapitre n'est pas de voir en détail les automates mais seulement de mener naturellement aux machines de Turing, nous en ferons l'impasse).

Cependant, le problème avec les automates reste toujours le même : leur absence de mémoire et leur traitement purement linéaire de l'information. Le contre-exemple classique nous le montre bien : on ne peut pas compter de façon arbitrairement grande une valeur.

3.1.5 Automates à pile

Pour palier l'absence de mémoire de nos automates finis, l'idée naturelle est d'ajouter de la mémoire (jusque là, la solution paraît évidente). Il existe de nombreuses structures de données permettant de jouer la mémoire, mais comme l'indique le nom de nos automates, nous allons stocker les informations dans une pile, c'est-à-dire une structure linéaire de type *last in first out*, où l'information extraite est la dernière information qui a été mise dans la pile. Le formalisme que nous utiliserons pour la pile est directement celui des mots : on peut représenter une pile par un mot u sur lequel on ajoute ou enlève une lettre à gauche.

Définition 3.1.13 (Automate à pile)**.** *Un automate à pile $\mathcal{A}$ (non déterministe) est la donnée d'un tuple $(\Sigma, \Gamma, Q, q_0, \delta, F, \bot)$ où :*

- *Σ est l'alphabet de lecture : c'est des mots de Σ que l'on va reconnaître.*
- *Γ est l'alphabet de pile : c'est avec cet alphabet que l'on écrira le mot qui nous servira de mémoire.*
- *Q est l'ensemble des états.*
- *$q_0 \in Q$ est toujours l'état initial.*
- *la fonction de transition est désormais $\delta : \Sigma^+ \times \Gamma \times Q \to \mathcal{P}(\Gamma^* \times Q)$ où $A^+ = A \cup \{\varepsilon\}$*
- *$F \subseteq Q$ est l'ensemble des états finaux.*
- *$\bot \in \Gamma$ est le symbole de fond de pile : il indique que la pile est vide lorsqu'il est lu.*

On appelle configuration interne une paire $(q, \gamma) \in Q \times \Gamma^*$: cela représente l'état de l'automate à un instant donné. Une configuration est définie comme un triplet $(q, \gamma, u) \in Q \times \Gamma^* \times \Sigma^*$. On appelle transition entre deux configurations $(q, \alpha{\cdot}\gamma, \beta{\cdot}u)$ et $(q', \gamma'{\cdot}\gamma, u)$ la relation $(\gamma', q') \in \delta(\beta, \alpha, q)$, et on la note $(q, \alpha{\cdot}\gamma, \beta{\cdot}u) \vdash (q', \gamma'{\cdot}\gamma, u)$.

On dit que le mot u est accepté par $\mathcal{A}$ s'il existe une suite de transitions depuis la configuration $(q_0, \bot, u)$ vers une configuration de la forme $(q_f, \gamma, \varepsilon)$ avec $q_f \in F$.

Cette définition d'automate permet de définir la classe des langages algébriques, qui est une classe strictement plus large que les langages rationnels (elle contient par exemple le langage $\{0^n 1^n \mid n \in \mathbb{N}\}$)

Ceux-ci possèdent des propriétés de clôture différentes : ils sont clos par union, concaténation, et passage à l'étoile, mais aussi par passage au miroir. Enfin, l'intersection d'un langage rationnel et d'un langage algébrique est un langage algébrique.

Il existe une théorie parallèle, celle des grammaires, qui permet de donner une autre description équivalente des langages algébriques, mais nous ne nous attarderons pas dessus, car l'exemple des automates à pile nous sert surtout à introduire la notion de mémoire et de configuration. Remarquons simplement que la pile est un format de mémoire très limité : on peut y stocker des informations mais n'accéder qu'à une information à la fois. Nous allons donc voir un système plus expressif, avec une mémoire plus manipulable : les machines de Turing.

3.2 Les machines de Turing

L'idée des machines de Turing est assez similaire à celle d'automate à pile : simplement, au lieu d'utiliser une pile, nous utiliserons un ruban de mémoire, qui est une succession infinie de case, chacune contenant une lettre. La machine, au lieu de simplement prendre la lettre en haut de la pile, peut se déplacer librement sur le ruban pour lire et écrire sur le ruban de mémoire. Cette section sera dédiée à la présentation des machines de Turing, mais comportera peu de preuves : celles-ci, à cause du formmalisme lourd d'un système si complexe, sont trop techniques pour être présentées dans ce cours. Certaines idées de preuves seront données, tout de même.

3.2.1 Des définitions équivalentes

Définition 3.2.1 (Machine de Turing)**.** *Une machine de Turing sur un alphabet Σ est un tuple $(Q, \Gamma, q_0, \delta, q_a, q_r)$ où :*

- *Q est l'ensemble des états de la machine.*
- *Γ est l'alphabet de travail : c'est l'alphabet dans lequel seront les lettres du ruban de travail. On impose que $\Sigma \subseteq \Gamma$ et que $B \in \Gamma$ où B est un symbole particulier, appelé symbole blanc, représentant l'absence d'écriture sur une case. On ajoute aussi le symbole $\triangleright$ qui sera en début de ruban pour marquer qu'on est au début.*
- *q_0 est l'état initial de la machine.*
- *$\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ est la fonction de transition : elle considère l'état de la machine, la lettre lue et réécrit sur la case une nouvelle lettre, change d'état et fait bouger la tête de lecture vers la gauche ou la droite d'une case.*
- *$q_a \in Q$ est l'état acceptant : si la machine arrive dans cet état, le calcul s'arrête et le mot est accepté.*
- *$q_r \in Q$ est l'état de rejet : si la machine arrive dans cet état, le calcul s'arrête et le mot est rejeté.*

Ces définitions ont peu de sens si l'on ne donne pas la définition de configuration et de ruban :

Définition 3.2.2 (Ruban, configuration, transition)**.** *Le ruban de travail est un mot infini $r \in \Gamma^\omega$ sur l'alphabet de travail. On appelle configuration un tuple (q, i, r) où q est l'état de la machine de Turing, r est le ruban et i est l'indice de la case sur laquelle se trouve la tête de lecture. Plutôt que de noter cela comme un tuple, on note en général une configuration par un mot de la forme $r_{<i} q r_{\geq i}$, c'est-à-dire que l'on écrit les symboles qui sont à gauche de la tête de lecture, puis l'état dans lequel est la machine, puis le reste du mot. La tête de lecture se trouve donc sur la lettre juste à droite de l'état q dans cette représentation.*

On appelle transition la fonction entre configurations définie de façon naturelle par la fonction de transition, en partant de la configuration $u\alpha q\beta r$:

- *si $\delta(q, \beta) = (q', a, \leftarrow)$ alors la transition sera $u\alpha q\beta r \to uq'\alpha ar$*
- *si $\delta(q, \beta) = (q', a, \rightarrow)$ alors la transition sera $u\alpha q\beta r \to u\alpha aqr$*

On appelle calcul la suite de transitions depuis une configuration donnée (comme la machine est déterministe, la suite est unique, potentiellement infinie).

De plus, on note B^∞ la fin du ruban (remplie forcément de B à l'infini).

Définition 3.2.3 (Acceptation, rejet)**.** *Soit x un mot sur un alphabet Σ et $\mathcal{M}$ une machine de Turing sur Σ. On dit que $\mathcal{M}$ accepte x si le calcul sur la configuration $q_0 \triangleright xB^\infty$ finit par $\mathcal{M}$ entrant dans l'état q_a. On dit que $\mathcal{M}$ rejette x si le calcul sur la même configuration finit par $\mathcal{M}$ entrant dans l'état q_r.*

Remarque 3.2.1. Un mot x peut n'être ni accepté, ni rejeté.

Donnons un exemple imagé d'une machine $\mathcal{M}$ agissant sur la configuration $\triangleright 0q1B^\infty$ en supposant que $\delta(q, 1) = (0, q', \leftarrow)$:

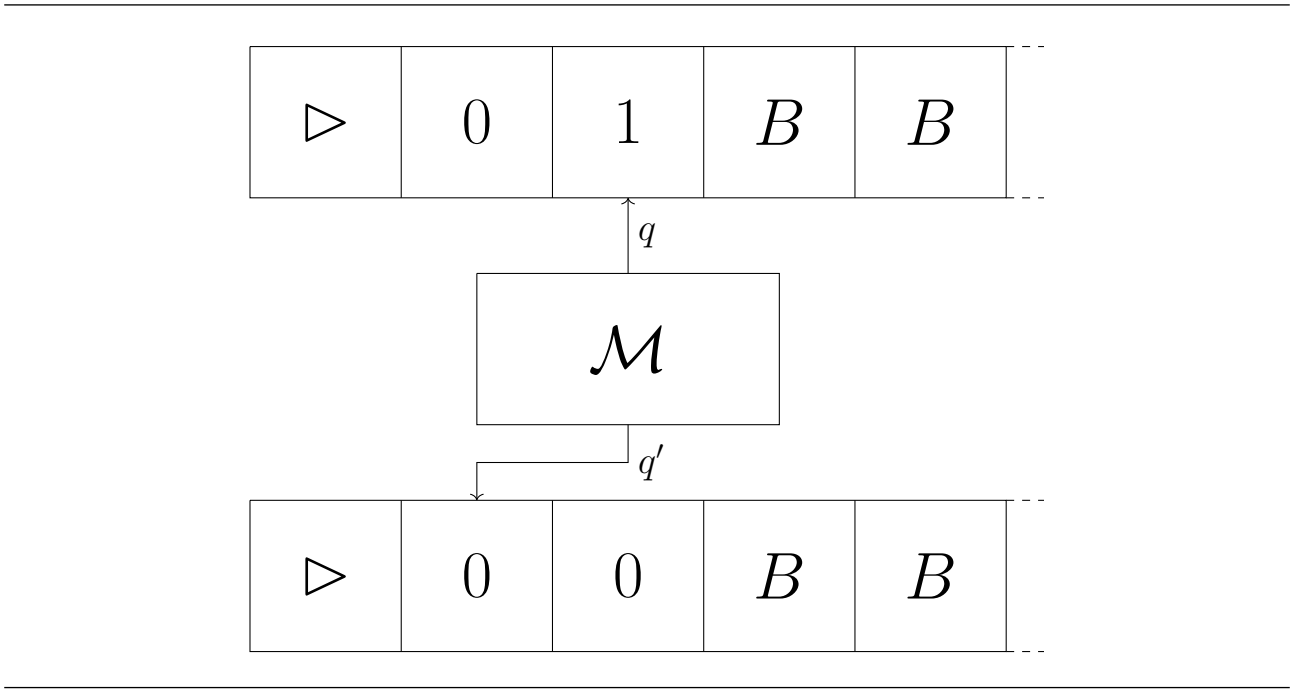


Figure 3.6 – Illustration de l'exécution d'une machine de Turing

Plusieurs versions équivalentes des machines de Turing existent, nous allons donner une liste et une façon de les simuler avec une machine à un ruban telle que décrite ici :

- La machine peut se retrouver sur un ruban infini des deux côtés. Dans ce cas, comme on a une bijection entre $\mathbb{N}$ et $\mathbb{Z}$ qui envoie $-n$ sur $2n+1$ et n sur $2n$, il nous suffit de créer des états qui permettent de gérer des transitions de deux cases à chaque fois qu'il faut faire une transition d'une case, sauf sur la case d'indice 0 où un changement d'état indiquera comment se comporter.
- On peut changer l'alphabet, et coder par exemple n'importe quel alphabet fini Σ avec $\{0,1\}$ en considérant les cases par paquets (suffisamment gros pour pouvoir coder chaque élément de Σ par un nombre binaire), ce qui augmentera largement le nombre d'états mais fonctionne. C'est d'ailleurs ce qu'il se passe dans les ordinateurs, où l'on code les données par paquets de 32 ou 64 bits.
- On peut considérer plusieurs rubans, et chaque tête de lecture se déplacera indépendamment. On peut simuler cela sur un ruban en considérant un alphabet plus gros : chaque case consistera en le n-uplet de cases de la machine à plusieurs rubans. Il faut néanmoins gérer le fait de bouger dans des sens différents : on peut pour cela considérer un alphabet encore plus gros qui serait constitué de parties de l'ancien alphabet et permettrait de faire une machine non déterministe.
- Une machine non déterministe peut se simuler avec une machine déterministe : chaque fois que plusieurs transitions sont possibles, on effectue une transition, puis on « backtrack » (fait la transition inverse) pour faire l'autre transition. On effectue cette démarche pour chaque suite de transitions possibles de longueur k et par k croissant.

En bref, notre modèle est très expressif, mais malheureusement très peu manipulable dans le sens où se donner une machine de Turing concrète est très fastidieux. Nous n'en donnerons d'ailleurs pas ici : nous n'en donnerons que des codes, c'est-à-dire des suites de symboles qui peuvent être traduites en une machine de Turing. Plus précisément, tout algorithme écrit en pseudo-code correspondra à une machine de Turing (on l'admet ici).

Enfin, on peut construire une version équivalente des machine de Turing où, au lieu d'avoir des états acceptant et rejetant, on a un simple état d'arrêt, et la machine retourne le mot écrit sur le ruban lorsqu'elle s'arrête (puisque le mot infini contient une infinité de B, il le mot retourné est bien un mot fini). On appelle ces fonctions $\Sigma^* \to \Sigma^*$ des fonctions partielles calculables. On dit qu'une fonction est calculable quand elle est totale calculable.

3.2.2 Les principaux théorèmes

Un premier théorème fondamental dans la théorie de la calculabilité est celui qu'il existe une machine universelle :

Théorème 3.2.1. *Il existe une machine de Turing dite universelle, $\mathcal{U}$ sur un alphabet Σ telle que, pour toute machine de Turing $\mathcal{M}$ codée dans Σ (le codage $\ulcorner\ \urcorner$ dépend de $\mathcal{U}$) et toute entrée x codée dans Σ, $\mathcal{U}$ accepte $\langle \ulcorner M \urcorner, x\rangle$ si et seulement si M accepte x, et refuse $\langle \ulcorner M \urcorner, x\rangle$ si et seulement si M refuse x.*

Comme dit plus tôt, nous ne donnerons pas de preuves pour ce genre de théorèmes, nous nous convaincrons seulement que l'on arrive en effet à coder un interpréteur pour un langage informatique donné, ce qui est l'analogue informatique d'une machine de Turing universelle.

La notation $\langle x, y\rangle$ sert à parler du codage d'un couple. Un codage simple dans l'alphabet $\{0, 1\}$ est $1^{|x|}0xy$: l'information nous est donnée en comptant le nombre de 1 de quelle taille fait x.

Le théorème suivant dit qu'on peut exécuter une machine sur elle-même :

Théorème 3.2.2 (Récursion de Kleene)**.** *Soit t une fonction calculable, il existe alors une machine de Turing F telle que $t\ulcorner F\urcorner$ est une machine équivalente à F.*

Une version équivalente est qu'une machine $\mathcal{M}$ a accès à son code source $\ulcorner\mathcal{M}\urcorner$.

3.2.3 Langage récursif, récursivement énumérable

Naturellement, nous pouvons associer à une machine $\mathcal{M}$ le langage $L(\mathcal{M})$ des mots reconnus par $\mathcal{M}$. Cependant, on peut remarquer une différence importante : là où dans les anciens automates la lecture était linéaire et donc terminait forcément, ici nous n'avons aucune garantie que l'exécution d'une entrée x sur $\mathcal{M}$ va arriver à un état acceptant ou refusant. Ceci signifie qu'il existe deux classes de langages bien différentes :

- les langages dits décidables, ou récursifs, qui sont ceux de la forme $L(\mathcal{M})$ pour une machine de Turing donnée qui, sur toute entrée, termine (i.e. tout calcul sur une entrée x mène à un des deux états q_a ou q_r).
- les langages dits récursivement énumérables, qui sont ceux de la forme $L(\mathcal{M})$ pour une machine de Turing $\mathcal{M}$ (qui peut donc ne pas terminer).

De même, on différencie comme on l'a dit plus tôt les fonctions calculables des fonctions calculables partielles. Pour montrer l'importance de cette distinction, nous allons montrer qu'il existe des langages récursivement énumérables non récursifs.

Remarque 3.2.2. Le nom de « récursivement énumérable » vient du fait qu'une condition équivalente est qu'il existe une machine $\mathcal{M}$ donnant la liste de tous les éléments du langage : une machine qui les énumère, donc. (Évidemment, ça ne suffit pas à décider si une entrée x est dans le langage, puisqu'il faudrait avoir accèd à l'entièreté de la liste, en général infinie.)

Définition 3.2.4 (Problème de l'arrêt)**.** *Soit* $\text{HALT} = \{\langle \ulcorner\mathcal{M}\urcorner, x\rangle \mid \mathcal{M}$ *s'arrête sur l'entrée* $x\}$*. HALT est récursivement énumérable mais pas récursif.*

Démonstration. L'existence d'une machine de Turing universelle suffit à montrer que le langage est récursivement énumérable : on exécute $\mathcal{M}$ sur x avec une machine universelle, et si elle termine on entre dans l'état acceptant. Maintenant, supposons que HALT soit récursif. On a donc une machine $\mathcal{M}$ qui peut dit si une machine s'arrête sur une entrée donnée. On définit alors le programme f suivant, prenant en entrée x :

- si $\langle \ulcorner x\urcorner, x\rangle \in \text{HALT}$ alors on crée une boucle infinie.
- sinon, on entre dans l'état q_a.

On remarque alors que, d'après le théorème de Kleene, on peut exécuter le programme f en prenant son code en entrée : alors si $\langle \ulcorner f\urcorner, f\rangle \in \text{HALT}$ on en déduit que $f(f)$ entre dans une boucle infinie, donc $\langle \ulcorner f\urcorner, f\rangle \notin \text{HALT}$ et de même si $\langle \ulcorner f\urcorner, f\rangle \notin \text{HALT}$ on en déduit que $\langle \ulcorner f\urcorner, f\rangle \in \text{HALT}$. □

Un problème dont on sait qu'il n'est pas récursif est dit indécidable : c'est le cas du problème de l'arrêt, mais plus généralement d'une très large classe de langages. En effet, le prochain théorème montre que tout langage défini par une propriété non triviale (c'est-à-dire autre que toujours vraie ou toujours fausse) sur un langage récursivement énumérable est indécidable. On appelle une telle propriété une propriété sémantique : elle porte sur un langage récursivement énumérable, par contraste avec une propriété syntaxique qui porte sur les machines elles-mêmes (on peut par exemple compter le nombre de boucle *While* d'un programme : les propriétés syntaxiques décrivent souvent des ensembles récursifs.

Théorème 3.2.3 (Rice)**.** *Soit RE la classe des langages récursivement énumérables sur un alphabet Σ. Soit une partie E de RE (donc une classe de langage), et P l'ensemble des machines qui reconnaissent un ensemble $A \in E$ (donc les machines $\mathcal{M}$ telles que $L(\mathcal{M}) \in E$). P est récursif si et seulement si $E = \varnothing$ ou $E = \mathrm{RE}$.*

Exercice 3.2.1. Montrer ce théorème en montrant que si P est non trivial alors il peut se réduire au problème de l'arrêt, c'est-à-dire construire à partir d'un P fixé une entrée du problème de l'arrêt, que l'on sait indécidable.

3.2.4 Fonctions récursives

Le formalisme des machines de Turing permet d'avoir une approche qui fait un lien fort avec l'informatique, puisqu'il est le formalisme à la base même des ordinateurs et de la notion de programme et de code. Cependant, pour l'étude logique, un autre formalisme équivalent va être privilégié : celui des fonctions récursives. Nous allons donc voir les fonctions récursives, d'abord par les fonctions récursives primitives, et nous verrons ensuite l'équivalence entre fonction récursive et fonction calculable par une machine de Turing.

Fonctions récursives primitives

Définition 3.2.5 (Classe RP)**.** *On définit la classe des fonctions récursives primitives RP comme le plus petit sous-ensemble de fonctions $\mathbb{N}^k \to \mathbb{N}$, k parcourant $\mathbb{N}$, tel que :*

- *pour chaque $n \in \mathbb{N}$ et $k \in \mathbb{N}$, la fonction $n : \mathbb{N}^k \to \mathbb{N}, x \mapsto n$ est dans RP.*
- *$S : \mathbb{N} \to \mathbb{N}, x \mapsto x + 1$ est dans RP.*
- *les fonctions $\pi_i^k : \mathbb{N}^k \to \mathbb{N}, (x_1, \ldots, x_k) \mapsto x_i$ sont dans RP.*
- *si $f_1, \ldots, f_k : \mathbb{N}^n \to \mathbb{N}$ sont des fonctions de RP et $h : \mathbb{N}^k \to \mathbb{N}$, alors la fonction $h\langle f_1, \ldots, f_k\rangle : \mathbb{N}^n \to \mathbb{N}, x \mapsto h(f_1(x), \ldots, f_k(x))$ est aussi dans RP.*
- *si $f : \mathbb{N}^k \to \mathbb{N}$ et $g : \mathbb{N}^{k+2} \to \mathbb{N}$ sont des fonctions de RP, alors la fonction $\texttt{rec}(f, g) : \mathbb{N}^{k+1} \to \mathbb{N}$ est dans RP, où*

$$\texttt{rec}(f, g)(x, 0) = f(x)$$
$$\texttt{rec}(f, g)(x, S(n)) = g(x, n, \texttt{rec}(f, g)(x, n))$$

Le point important pour l'expressivité de cette classe de fonction est évidemment le fait d'autoriser les appels récursifs. On peut construire de nombreuses fonctions usuelles par cette classe.

Exercice 3.2.2. Montrer que les fonctions suivantes sont dans RP :

- La fonction identité $x \mapsto x$
- La fonction somme : $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x + y$
- La fonction produit : $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x \times y$
- La fonction puissance : $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x^y$
- La fonction prédécesseur : $\mathbb{N} \to \mathbb{N}, x \mapsto \max(0, x - 1)$
- La fonction différence : $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto \max(0, x - y)$
- La fonction moitié : $\mathbb{N} \to \mathbb{N}, x \mapsto \lfloor x/2 \rfloor$

- La fonction division : $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto \lfloor x/y \rfloor$ et 0 si $y = 0$
- La fonction reste : $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x\%y$ le reste de x par la division par y, et 0 si $y = 0$
- Pour tout polynôme de $\mathbb{N}[X]$, la fonction $\mathbb{N} \to \mathbb{N}$ associée.
- Pour f, g, h trois fonctions de RP, la fonction $\texttt{if}(f, g, h) : x \mapsto g(x)$ si $f(x) = 0$ et $h(x)$ sinon.

Exercice 3.2.3. Montrer que la fonction $b : \mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto \frac{1}{2}(x + y)(1 + x + y) + y$ établit une bijection entre $\mathbb{N}^2$ et $\mathbb{N}$. Construire aussi les fonction π_1 et π_2 telles que $\pi_1(b(x, y)) = x$ et $\pi_2(b(x, y)) = y$.

On peut aussi analyser des ensembles grâce à leur fonction caractéristique.

Exercice 3.2.4. Construire les fonctions caractéristiques des ensembles suivants :

- $\mathbb{N}_{>k}$ pour $k \in \mathbb{N}$
- $\{0\}$
- $n\mathbb{N}$
- $\{(x, x) \mid x \in \mathbb{N}\}$
- $\{(x, y) \mid x < y\}$
- $\{(x, y) \mid y = P(x)\}$ où $P \in \mathbb{N}[X]$ est fixé.

Opérateur mu

Cependant, toutes les fonctions calculables par des machines de Turing ne sont pas forcément dans RP : certaines croissent trop vite. En effet, une remarque assez directe est que les fonctions récursives primitives ne peuvent pas appliquer une fonction un nombre non borné de fois : il faut calculer au préalable une borne supérieure du nombre d'application de notre fonction. De plus, toutes les fonctions dans RP sont des fonctions totales, alors que nous voulons pouvoir écrire des fonctions partielles. Nous allons donc définir Rec, la classe des fonctions récursives, en ajoutant un constructeur :

Définition 3.2.6 (Classe Rec)**.** *La classe Rec est la classe RP à laquelle on ajoute le constructeur μ : si $f : \mathbb{N}^{k+1} \to \mathbb{N}$ est dans Rec, alors $g(x_1, \ldots, x_k) = \mu(y, f)$ renvoie le plus petit y tel que $f(x_1, \ldots, x_k, y) = 0$ et pour tout $x < y$, $f(x_1, \ldots, x_k, x)$ est définie, et n'est pas définie sinon.*

Il se trouve que Rec est exactement la classe des fonctions calculables partielles.

Théorème 3.2.4 (Simulation des machines de Turing)**.** *Soit $\mathcal{M}$ une machine de Turing sur $\{0, 1\}$ définissant une fonction $f : \mathbb{N} \to \mathbb{N}$ à partir de son comportement sur $\{0, 1\}^* \to \{0, 1\}^*$. Alors $f \in$* Rec.

Démonstration. Nous allons pour cela construire par étape une fonction f récursive qui simule $\mathcal{M}$:

- Nous allons coder les états de $\mathcal{M}$ par des entiers.
- Une configuration peut être vue comme un élément de $\mathbb{N}^3$ avec ce codage : nous allons donc construire une fonction qui effectue une transition, ou plus précisément trois fonctions, l'une retournant le nouvel indice de la tête de lecture, la deuxième retournant le code de l'état et la troisième retournant la forme nouvelle du ruban.
- Cette fonction peut se faire par disjonction de cas : on sait coder la fonction caractéristique de chaque entier, et faire des branchements if/then/else. Il ne reste qu'à effectuer cette disjonction de cas sur chaque état en entrée, puis une disjonction de cas suivant si le symbole lu est 0 ou 1. La fonction d'indice décrémente ou incrémente l'indice directement, ce que l'on peut faire, la fonction d'état associe une constante, et la fonction de ruban n'a qu'à effectuer de l'arithmétique pour ajouter ou soustraire 2^i avec i l'indice de la tête de lecture.

- Si l'on nomme g cette fonction, on peut alors l'appliquer en boucle jusqu'à obtenir un résultat. C'est ici que l'on a besoin de μ, qui s'appliquera à la fonction caractéristique de l'état d'arrêt composée avec la fonction qui pour i effectue i itérations de g.

□

Remarque 3.2.3. La construction d'une fonction récursive simulant une machine de Turing ne nécessite l'opérateur μ que pour pouvoir appliquer l'opération de transition un nombre non borné de fois : cela signifie que pour une machine $\mathcal{M}$ où l'on peut borner son temps de calcul en fonction de son entrée, il suffit d'une fonction primitive récursive pour coder la machine $\mathcal{M}$.

Chapitre 4

Arithmétique de Peano et incomplétude

Nous avons étudié la logique des prédicats, mais ne l'avons pas encore mise en œuvre. Ce chapitre en sera une première application : nous allons introduire l'axiomatisation de l'arithmétique de Peano, c'est-à-dire la théorie axiomatique définissant les entiers naturels usuellement. Nous commencerons donc par énoncer les axiomes de la théorie nommée PA, pour ensuite définir quelques propriétés basiques. Historiquement, l'axiomatisation de l'arithmétique a été la première réussite du logicisme : réussir à formuler de façon purement formelle ce que nous entendons par les entiers et leurs relations usuelles. C'est cette réussite qui a mené David Hilbert à poser la question à toute la communauté scientifique : « arriverons-nous à décider toute proposition arithmétique de façon purement mécanique ? » (Nous dirions maintenant « existe-t-il un algorithme décidant les proposition vraies ? » mais le problème a été posé avant que la théorie de la calculabilité émerge). La réponse, bien connue maintenant, est non : non seulement il n'existe pas d'algorithme permettant de décider les propositions, mais tout système suffisamment expressif pour parler de toute l'arithmétique, mais suffisamment restreint pour être manipulable, possède des propositions qui sont indécidables, c'est le premier théorème d'incomplétude de Gödel. Nous reviendrons à la fin du chapitre sur ce théorème, ainsi que le second, qui sont des passages obligés du traitement moderne de la logique. Bien sûr, nous avons maintenant l'outil de la calculabilité qui rendra beaucoup plus digeste (bien que toujours très dur à assimiler) la preuve que nous ferons du premier théorème, là où historiquement la preuve est plus vieille que la notion de machine de Turing. Le second théorème d'incomplétude sera mentionné mais non développé, pour des raisons que nous développerons dans la dernière section.

4.1 Axiomes de Peano

Nous allons donc expliciter la théorie axiomatique PA. Pour cela, donnons d'abord le langage de l'arithmétique : $\mathcal{L} = \{0, S, +, \times\}$ où, sans surprise, 0 est une constante, S une fonction d'arité 1 représentant la fonction $x \mapsto x+1$, $+$ et $\times$ des fonctions d'arité 2. On pourrait ajouter $\leq$ une relation binaire, mais nous allons voir que nous pouvons la définir de façon peu coûteuse. L'idée de notre définition de $\mathbb{N}$ est que $\mathbb{N}$ est défini comme une structure inductive avec une constante et un constructeur unaire.

Plutôt que d'énoncer tous les axiomes de Peano à la suite, nous allons nous concentrer sur chacun d'eux (ou au moins chaque groupe).

Définition 4.1.1 (Axiomes du successeur)**.** *Le premier axiome est* $A_1 := \forall x.\neg(Sx = 0)$*, nous nous permettrons d'écrire* $\neg(a = b)$ *par* $a \neq b$*.*

Le second axiome est $A_2 := \forall x.\exists y.(x \neq 0 \to Sy = x)$*.*

Le troisième axiome est $A_3 := \forall x.\forall y.((Sx = Sy) \to (x = y))$

Ces axiomes traduisent le comportement de l'opération S : 0 n'est le successeur de personne, mais tout autre élément est le successeur d'au moins un élément, et la fonction S est injective (donc il n'existe finalement qu'un seul prédécesseur à un entier non nul). Les axiomes qui suivent définissent $+$ et $\times$:

Définition 4.1.2 (Axiomes de l'addition)**.** *Les deux axiomes suivants sont :*

$$A_4 := \forall x.(x + 0 = x)$$
$$A_5 := \forall x.\forall y.x + (Sy) = S(x + y)$$

Ces axiomes traduisent directement la définition récursive de l'addition. De même les deux axiomes suivants définissent la multiplication de façon récursive :

Définition 4.1.3 (Axiomes de la multiplication)**.** *Les deux axiomes suivants sont :*

$$A_6 := \forall x.(x \times 0 = 0)$$
$$A_7 := \forall x.\forall y.x \times (Sy) = (x \times y) + x$$

On note alors PA_0 la théorie $\mathrm{PA}_0 = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$. Cette théorie est très faible : elle ne permet aucune preuve par récurrence. Ceci est dû au fait qu'a priori, tous les éléments ne sont pas forcément des successeurs de 0 : simplement qu'un élément est localement un successeur ou 0. Cependant, si l'énoncé commun de la récurrence est « toute partie $F \subseteq \mathbb{N}$ contenant 0 et stable pour S, càs telle que $x \in F \implies Sx \in F$ alors $F = \mathbb{N}$ », cette quantification sur les parties $F \subseteq \mathbb{N}$ n'est pas définissable dans notre théorie du premier ordre (c'est exactement une quantification du second ordre). Cependant, de la façon dont on se sert de la récurrence, on remarque que la notion importante est celle de proposition, et non de sous-ensemble : on veut que toute proposition vraie pour 0 et dont la vérité à un rang implique sa vérité au rang suivant est vraie pour tous les objets. Cette version peut se faire, pour une proposition donnée : il suffit d'écrire pour une proposition P la proposition $P(0) \wedge (\forall x.P(x) \to P(S(x))) \to \forall x.P(x)$ et de l'ajouter en axiome. L'idée est donc, alors, d'ajouter autant d'axiomes que nous avons de propositions : on va définir alors ce qu'on appelle un schéma d'axiomes (c'est-à-dire un ensemble d'axiomes dont on décrit la forme, de façon équivalente dont on donne un algorithme de construction).

Définition 4.1.4 (Schéma d'axiomes de récurrence)**.** *Pour P une proposition à n variables libres $x_0, \ldots, x_n$, la proposition*

$$\forall x_1 \ldots \forall x_n.((P(0, x_1, \ldots, x_n) \wedge \forall x.(P(x, x_1, \ldots, x_n) \to P(S(x), x_1, \ldots, x_n))) \to \forall x.P(x, x_1, \ldots x_n))$$

appartient à l'ensemble $\mathcal{P}_{\mathrm{i}}$.

Lemme 4.1.1 (Récursivité de la récurrence)**.** *L'ensemble $\mathcal{P}_{\mathrm{i}}$ est un langage récursif (et donc décidable) sur $(\mathcal{L} \cup \{(,), \vee, \wedge, \to, \neg, \forall., \exists.\})^*$.*

Démonstration. Nous ne donnerons pas une réelle démonstration ici mais une idée de pourquoi le résultat est vraie (une démonstration rigoureuse nécessiterait d'entrer en détails dans le fonctionnement des machines de Turing, ce que nous nous refusons de faire pour garder une certaine lisibilité). Ce résultat vient directement du fait qu'on a donné une description algorithmique de $\mathcal{P}_{\mathrm{i}}$ (on pourrait penser que cela nous donne uniquement un ensemble récursivement énumérable, mais on peut par exemple ordonner les propositions par longueur et considérer le nombre de symboles de la proposition dont on veut savoir si elle est dans $\mathcal{P}_{\mathrm{i}}$: il n'y aura qu'un nombre fini de propositions plus petites qu'il faudra donc tester). □

On peut alors définir PA comme l'ensemble précédent muni du schéma d'axiomes de récurrence.

Définition 4.1.5 (Théorie PA)**.** *On définit* $\mathrm{PA} := \mathrm{PA}_0 \cup \mathcal{P}_{\mathrm{i}}$.

Lemme 4.1.2 (Récursivité de PA)**.** *PA est un langage récursif.*

Démonstration. Il suffit de tester pour une proposition si elle est l'une des 7 premières puis d'exécuter l'algorithme pour vérifier si la proposition est dans $\mathcal{P}_{\mathrm{i}}$. □

Exercice 4.1.1. Montrer que les axiomes de $\mathcal{P}_\text{i}$ peuvent être remplacés par une nouvelle règle :

$$\frac{\Gamma \vdash P(0) \qquad \Gamma \vdash \forall x.P(x) \to P(S(x))}{\Gamma \vdash \forall x.P(x)} \text{ Rec}$$

et que l'on peut alors définir PA comme la théorie PA_0 enrichie de cette règle logique dans la relation de dérivation de jugement logique. *Remarque : Cette règle n'a de sens que dans le langage de l'arithmétique, et on présuppose largement que* $\Gamma \subseteq \text{PA}_0$ *pour que la règle ait un sens.*

Exercice 4.1.2. Montrer qu'il existe un modèle non standard de PA, c'est-à-dire un modèle qui n'est pas l'ensemble usuel des entiers naturels. *Indication : on introduira une constante dont on montrera qu'elle n'est égale à aucun entier naturel en utilisant le théorème de compacité.*

4.2 L'arithmétique telle que nous la connaissons

On peut alors commencer à prouver des résultats semblant évidents mais qui n'appartiennent pas directement à la théorie. Nous ne donnerons pas d'arbre de preuve mais seulement des arguments. Leur mise en arbre de preuve sera laissée en exercice.

Proposition 4.2.1 (Neutre)**.** 0 *est neutre pour* $+$*, c'est-à-dire que*

$$\forall x.x + 0 = x$$
$$\forall x.0 + x = x$$

1 *est neutre pour* $\times$ *:*

$$\forall x.x \times 1 = x$$
$$\forall x.1 \times x = x$$

Démonstration. La première proposition est purement un axiome de notre théorie, mais la neutralité à gauche de 0 est déjà à prouver (par récurrence évidemment). D'abord $0+0=0$ en appliquant l'axiome A_4, puis si $0+x=x$ alors $0+(Sx)=S(0+x)=Sx$ en remplaçant $0+x$ par x grâce à l'hypothèse. Donc pour tout x, $0+x=x$.

La neutralité à droite de 1 pour $\times$ est un simple calcul :

$$\begin{aligned} x \times 1 &= x \times (S0) \\ &= (x \times 0) + x \\ &= x \end{aligned}$$

pour la neutralité à gauche, nous allons encore raisonner par récurrence : d'abord $1 \times 0 = 0$ grâce à l'axiome A_6, puis si l'on suppose que $1 \times n = n$ alors $1 \times (Sn) = (1 \times n) + 1$ puis on remarque que $n+1 = Sn$ de façon directe, eet que $1 \times n = n$ par hypothèse de récurrence, nous donnant que $1 \times (Sn) = Sn$. □

Proposition 4.2.2 (Associativité et commutativité de l'addition)**.** *L'addition est associative et commutative, ce qui signifie :*

$$\forall x.\forall y.\forall z.x + (y + z) = (x + y) + z$$
$$\forall x.\forall y.x + y = y + x$$

Démonstration. Prouvons le premier point par récurrence sur z (quitte à échanger les variables) : dans le premier cas, on a $x+y=x+y$ après réduction des additions de 0 à droite. Si l'on suppose que $x+(y+z)=(x+y)+z$ alors

$x+(y+Sz) = x+S(y+z) = S(x+(y+z))$ et ici, par hypothèse de récurrence, on peut remplacer $x+(y+z)$ par $(x+y)+z$ et on finit la chaîne d'égalités par $S((x+y)+z) = (x+y)+Sz$, prouvons la propriété par récurrence.

Pour le deuxième point, nous prouverons un lemme intermédiaire : $\forall x.\forall y.S(x+y) = Sx+y$ par récurrence sur y : dans le cas nul, on a $S(x+0) = S(x)+0$ qui est vrai. Si $\forall x.S(x+y) = Sx+y$, alors $Sx+Sy = S(Sx+y)$ et par application de l'hypothèse de récurrence, en remplaçant x par Sx, on en déduit que $S(Sx+y) = SS(x+y)$, et $SS(x+y) = S(x+Sy)$ donc

$$Sx+Sy = S(x+Sy)$$

ce qui complète notre démonstration par récurrence.

Montrons maintenant la commutativité par récurrence sur y : dans le cas nul, on doit montrer que $x+0 = 0+x$ ce qui se fait grâce au fait que 0 est neutre. Supposons que $\forall x.x+y = y+x$, alors en fixant un certain x, $x+Sy = S(x+y)$ puis par hypothèse de récurrence, $S(x+y) = S(y+x)$ et en utilisant le lemme précédent, $S(y+x) = Sy+x$, donc

$$x+Sy = Sy+x$$

concluant notre preuve par récurrence. □

Nous allons maintenant prouver la distributivité.

Proposition 4.2.3 (Distributivité)**.** *La multiplication est distributive sur l'addition, c'est-à-dire que :*

$$\forall x.\forall y.\forall z.x\times(y+z) = x\times y + x\times z$$

en prenant les conventions de priorité habituelles, et

$$\forall x.\forall y.\forall z.(x+y)\times z = x\times z + y\times z$$

Démonstration. Prouvons le premier résultat par récurrence sur z : dans le cas où $z=0$, on trouve $\forall x.\forall y.x\times(y+0) = x\times y + x\times 0$, ce qui se vérifie facilement. Dans le cas où $\forall x.\forall y.x\times(y+z) = x\times y + x\times z$, alors pour x, y fixéx,

$$\begin{aligned} x\times(y+Sz) &= x\times S(y+z) \\ &= (x\times(y+z))+x \\ &= (x\times y + x\times z)+x \\ &= x\times y + (x\times z + z) \\ &= x\times y + x\times(Sz) \end{aligned}$$

ce qui conclut la récurrence.

Le deuxième résultat se prouve aussi par récurrence sur z : pour $z=0$, on trouve $\forall x.\forall y.(x+y)\times 0 = 0 = x\times 0 + y\times 0$. Si $\forall x.\forall y.(x+y)\times z = x\times z + y\times z$ alors

$$\begin{aligned} (x+y)\times Sz &= (x+y)\times z + (x+y) \\ &= (x\times z + y\times z)+(x+y) \\ &= (x\times z + x)+(y\times z + y) \\ &= (x\times Sz)+(y\times Sz) \end{aligned}$$

ce qui conclut la récurrence. □

De même, démontrons ces propriétés pour la multiplication.

Proposition 4.2.4 (Associativité et commutativité de la multiplication)**.** *La multiplication est associative et commutative.*

Démonstration. On raisonne encore par récurrence sur z : dans le premier cas tout se simplifie en $0 = 0$. Si $\forall x.\forall y.x \times (y \times z) = (x \times y) \times z$, alors pour x, y fixés, on a

$$\begin{aligned} x \times (y \times Sz) &= x \times (y \times z + y) \\ &= x \times (y \times z) + x \times y \\ &= (x \times y) \times z + x \times y \\ &= (x \times y) \times Sz \end{aligned}$$

ce qui conclut la récurrence.

Pour la commutativité le cas de $y = 0$ est évident. Si l'on suppose que $\forall x.x \times y = y \times x$, alors pour prouver que $x \times Sy = Sy \times x$, nous allons procéder par récurrence sur x : $0 \times Sy = Sy \times 0$ de façon évidente, et si $x \times Sy = Sy \times x$, alors

$$\begin{aligned} Sy \times Sx &= (Sy \times x) + Sy \\ &= (x \times Sy) + Sy \\ &= ((x \times y) + x) + Sy \\ &= (x \times y) + (x + Sy) \\ &= (x \times y) + S(x + y) \\ &= (x \times y) + S(y + x) \\ &= (x \times y) + (y + Sx) \\ &= ((x \times y) + y) + Sx \\ &= ((y \times x) + y) + Sx \\ &= (y \times Sx) + Sx \\ &= (Sx \times y) + Sx \\ &= Sx \times Sy \end{aligned}$$

donc on en déduit que $\forall x.x \times Sy = Sy \times x$, donc on en déduit que $\forall y.\forall x.x \times y = y \times x$. □

Les opérations sont aussi régulières :

Proposition 4.2.5 (Régularité)**.** *On a*

$$\forall x.\forall y.\forall z.(x + z = y + z) \implies x = y$$

et

$$\forall x.\forall y.\forall z.(z \neq 0) \implies (x \times z = y \times z) \implies x = y$$

Démonstration. Prouvons la proposition par récurrence sur z : le premier cas est $x + 0 = y + 0$ d'où $x = y$. Si $(x + z = y + z) \implies x = y$, alors supposons que $x + Sz = y + Sz$, par axiome on a $S(x + z) = S(y + z)$ puis, encore par axiome, que $x + z = y + z$ donc $x = y$.

Prouvons la deuxième proposition par récurrence sur x et y (d'abord sur x puis sur y). Pour $x = 0$, on suppose que $x \times z = y \times z$ donc que $y \times z = 0$ et que $z \neq 0$, donc $y = 0 = x$. Supposons alors que pour tout y, z, $(z \neq 0) \implies (x \times z = y \times z) \implies x = y$, et prouvons que le résultat $\forall y.\forall z.(z \neq 0) \implies (Sx \times z = y \times z) \implies x = y$ est vrai par récurrence sur y. Dans le cas nul, on retrouve le raisonnement précédent. On suppose donc maintenant que pour tout z, $(z \neq 0) \implies (Sx \times z = y \times z) \implies x = y$ et on veut prouver que pour tout z, $(z \neq 0) \implies (Sx \times z = Sy \times z) \implies x = y$. Supposons donc que $Sx \times z = Sy \times z$, alors $z \times Sx = z \times Sy$ par commutativité, puis par définition $z \times x + z = z \times y + z$, d'où par régularité de l'addition, $z \times x = z \times y$. Alors puisque par hypothèse $z \neq 0$, on peut appliquer l'hypothèse d'induction du début pour déduire que $x = y$. □

Les opérations d'addition et de multiplication se comportent bien comme on l'attend, ce qui est très rassurant quant à notre axiomatisation (avec le principe de récurrence et 7 axiomes, nous avons déjà de nombreuses propriétés qui semblent bien correspondre à celles de $(\mathbb{N}, +, \times)$).

4.3 Extension possible, fonction représentable

Tout d'abord, on peut définir le prédicat $x \leq y$ en disant que ce symbole, à l'origine hors du langage, est juste une façon rapide d'écrire $\exists k.x + k = y$: en effet être inférieur pour les entiers signifie simplement qu'il y a une différence positive entre les deux entiers. On peut alors montrer que $x \leq y$ est bien une relation d'ordre, et même qu'elle est totale.

Exercice 4.3.1. Montrer que $\leq$ est une relation d'ordre totale dans PA. Montrer de plus qu'elle est compatible avec l'addition, c'est-à-dire que

$$\forall x.\forall y.\forall z.(x \leq y) \implies (x + z \leq y + z)$$

et avec la multiplication :

$$\forall x.\forall y.\forall z.(x \leq y) \implies (x \times z \leq y \times z)$$

Nous pouvons donc facilement définir des prédicats en tant que sucre syntaxique : un prédicat pouvant déjà se définir par une proposition, il nous suffit de travailler au sein des propositions. Cependant, si l'on veut définir une fonction f au sein de la théorie, comment faire ? Pour cela, nous allons avoir besoin d'abord de déterminer ce que l'on appelle le modèle standard de PA, qui est $\mathbb{N}$ dans son sens usuel.

Définition 4.3.1 (Modèle standard de PA)**.** *On définit $\mathbb{N}$, le modèle standard de PA, par les définitions suivantes :*

- $|\mathbb{N}|$ *est l'ensemble des entiers naturels tels que l'on les entend en général (à la limite, penser à leur construction dans ZF, si l'on veut être formaliste, mais sinon il suffit de s'imaginer ce qu'on entend intuitivement par $\mathbb{N}$: l'ensemble des nombres finis)*
- 0 *est interprété par l'entier* 0
- S *est interprété par la fonction* $x \mapsto x + 1$
- $+$ *est interprété par l'addition évidente.*
- $\times$ *de même.*

Remarquons que chaque n de $\mathbb{N}$ est exactement représenté par $S^n 0$ où l'on considère l'application en boucle de S, n fois. Nous noterons $\overline{n}$ pour considérer le représentant dans PA du nombre $n \in \mathbb{N}$.

Exercice 4.3.2. Montrer que tout modèle de PA possède un sous-modèle isomorphe à $\mathbb{N}$, et que de plus il en est un segment initial, au sens où si n est l'image d'un entier standard et $m \leq n$ alors m est l'image d'un entier standard.

En réalité, ce qui nous intéresse en premier lieu dans PA est de formaliser $\mathbb{N}$: si l'on peut prouver l'existence de modèles non standards de PA, l'important est que tout modèle non standard possède un représentant de $\mathbb{N}$, et que l'on peut donc y définir des fonctions $\mathbb{N}^k \to \mathbb{N}$ par exemple.

Par exemple ? Non, c'est exactement le cadre des fonctions récursives, et ce que nous allons vouloir faire maintenant est de montrer que l'on peut représenter les fonctions récursives dans PA. Commençons par définir une fonction représentable en général.

Définition 4.3.2 (Fonction représentable)**.** *Soit $f : \mathbb{N}^k \to \mathbb{N}$ une fonction (totale), on dit que f est représentable dans PA lorsqu'il existe une formule F à $k+1$ variables libres telle que pour tous entiers $n_1, \ldots, n_k$, on a*

$$\text{PA} \vdash \forall y.(F(y, \overline{n_1}, \ldots, \overline{n_k}) \leftrightarrow y = \overline{f(n_1, \ldots, n_k)})$$

Une fonction représentable est donc une fonction dont on peut définir le fait d'appartenir à son graphe par une formule F. Remarquons que l'on ne définit pas une formule close ici : on veut pouvoir donner en entrée à notre formule F les paramètres de notre fonction f.

Nous allons de même définir un ensemble représentable comme correspondant à une proposition :

Définition 4.3.3 (Ensemble représentable)**.** *Un ensemble $A \subseteq \mathbb{N}^k$ est dit représentable s'il existe une proposition F à k variables libres $x_1, \dots, x_k$ telle que :*

- *si $(x_1, \dots, x_k) \in A$ alors* $\text{PA} \vdash F(\overline{x_1}, \dots, \overline{x_k})$
- *si $(x_1, \dots, x_k) \notin A$ alors* $\text{PA} \vdash \neg F(\overline{x_1}, \dots, \overline{x_k})$

Exercice 4.3.3. Montrer que A est représentable si et seulement si sa fonction caractéristique 1_A est représentable.

Montrons donc que les fonctions de base de RP sont représentables :

- La fonction successeur est évidemment représentable par S.
- La projection π_i^k est représentée par la formule à $k+1$ variables $x_0 = x_i$ (en reprenant les conventions précédentes).
- La fonction constante est représentée par la formule à $k+1$ variables $x_0 = \overline{k}$.

Nous allons ensuite prouver la stabilité pour les trois schémas : la composition, la récursion et l'opérateur μ. Rappelons d'après notre théorème de simulation des machines de Turing qu'une fonction récursive est une fonction récursive primitive à laquelle on n'a appliqué qu'une seule foie l'opération μ.

Lemme 4.3.1 (Représentabilité de la composition)**.** *Si $f_1, \dots, f_p : \mathbb{N}^k \to \mathbb{N}$ sont représentables et $g : \mathbb{N}^p \to \mathbb{N}$ est représentable, alors la fonction $f : \mathbb{N}^k \to \mathbb{N}, x \mapsto (f_1(x), \dots, f_p(x))$ est représentable.*

Démonstration. Soient $f_1, \dots, f_p$ des fonctions représentables à k entrées et g une fonction représentable à p entrée, représentées respectivement par $F_1, \dots, F_p, G$. On va représenter f par la proposition

$$F(x_0, \dots, x_k) := \exists y_1 \dots \exists y_p . (G(x_0, y_1, \dots, y_p) \wedge \bigwedge_{1 \leq i \leq n} F_i(y_i, x_1, \dots, x_p))$$

qui, moralement, assigne les valeurs $f_i(x_1, \dots, x_p)$ à y_i pour calculer ensuite $x_0 = g(y_1, \dots, y_p)$. Rappelons que nous avons pour chaque i et $n_1, \dots, n_k \in \mathbb{N}$:

$$\text{PA} \vdash \forall y . (F(y, \overline{n_1}, \dots, \overline{n_k}) \leftrightarrow y = \overline{f_i(n_1, \dots, n_k)})$$

et pour $n_1, \dots, n_p$:

$$\text{PA} \vdash \forall y . (G(y, \overline{n_1}, \dots, \overline{n_p}) \leftrightarrow y = \overline{g(n_1, \dots n_p)})$$

Nous allons démontrer notre résultat par complétude : soit $\mathcal{M}$ un modèle de PA (on identifiera directement $\mathbb{N}$ à une partie de $\mathcal{M}$). Alors pour $n_1, \dots, n_k \in \mathbb{N}$ et $y \in \mathcal{M}$ on a :

- $F(y, n_1, \dots, n_k)$ vraie si et seulement si il existe $y_1, \dots, y_p \in |\mathcal{M}|$ tels que pour tout i, la formule $F_i(y_i, n_1, \dots, n_k)$ est vraie et $G(y, y_1, \dots, y_p)$ est vraaie aussi.
- Ce qui équivaut, en utilisant la représentation des f_i par F_i dans le cadre sémantique, à ce qu'il existe $y_1, \dots, y_p \in |\mathcal{M}|$ tels que pour tout i, $y_i = f_i(n_1, \dots, n_k)$ et $G(y, y_1, \dots, y_p)$.
- En utilisant la représentation de g par G, ceci équivaut à ce qu'il existe $y_i, \dots,_p \in |\mathcal{M}|$ tels que pour tout i, $y_i = f_i(n_1, \dots, n_k)$ et $y = g(y_1, \dots, y_p)$.
- Ceci revient à dire que $y = f(n_1, \dots, n_k)$

Donc, comme ceci est vrai dans tout modèle, on en déduit que pour tous $n_1, \dots, n_k \in \mathbb{N}$:

$$\text{PA} \vdash \forall y . (F(y, \overline{n_1}, \dots, \overline{n_k}) \leftrightarrow y = \overline{f(n_1, \dots, n_k)})$$

□

Remarque 4.3.1. On interprète bien f_i par f_i et non par $f_i^{\mathcal{M}}$ car les f_i sont des fonctions déjà définies dans le modèle $\mathcal{M}$, en tant que fonctions sur la sous-interprétation $\mathbb{N}$. Il en va de même pour g.

Plutôt que de nous occuper du schéma de récursion, nous allons d'abord coder l'opérateur μ. Cependant, nous imposons la restriction de ne travailler qu'avec des fonctions totales (donc nous allons utiliser des fonctions récursives mais pas récursives partielles, ce qui peut être engendré par l'ensemble des fonctions récursives primitives avec l'ajout de l'opérateur μ).

Lemme 4.3.2 (Représentabilité de l'opérateur mu)**.** *Si à partir de fonctions représentables on obtient par l'opérateur μ une fonction totale, alors la fonction est représentable.*

Démonstration. Soient $g : \mathbb{N}^{k+1} \to \mathbb{N}$ représentable totale et $f : \mathbb{N}^k \to \mathbb{N}$ définie par

$$f(x_1, \dots, x_k) = \mu(x_0, g(x_0, \dots, x_k))$$

et supposons que f est totale. Montrons que f est représentable :

Soit G représentant g, on définit

$$F(y, x_1, \dots, x_k) := G(0, y, x_1, \dots, x_k) \wedge (\forall z.z < y \to \neg G(0, z, x_1, \dots, x_k))$$

pour représenter f. Soit $\mathcal{M}$ un modèle de PA et $n_1, \dots, n_k$ des entiers, $y \in |\mathcal{M}|$. Alors $F(y, n_1, \dots, n_k)$ est vraie si et seulement si $G(0, y, n_1, \dots, n_k)$ et pour tout $z < y$, $\neg G(0, z, n_1, \dots, n_k)$. Montrons que $b = f(n_1, \dots, n_k)$ est le seul élément à satisfaire $F(y, n_1, \dots, n_k)$:

$F(b, n_1, \dots, n_k)$ est équivalent à $G(0, b, n_1, \dots, n_k)$ et pour tout $z < b$, $\neg G(0, z, n_1, \dots, n_k)$ or $G(0, b, n_1, \dots, n_k)$ est vraie car G représente g et b est définie par μ-opérateur. De plus, comme $\mathbb{N}$ est un segment initial de $\mathcal{M}$, on en déduit que tout élément $z < b$ est dans $\mathbb{N}$, et $\neg G(0, z, n_1, \dots, n_k)$ est donc directement vraie par définition du μ-opérateur. Donc $F(b, n_1, \dots, n_k)$ est vraie. S'il existait un autre y satisfaisant $F(y, n_1, \dots, n_k)$ alors par définition si y est entier standard, on a $y = b$ (par μ-opération) et si y est entier non standard, alors forcément $y > b$ donc la condition $\forall z.z < y \to \neg G(0, y, n_1, \dots, n_k)$ n'est pas respectée pour $z = b$.

Donc l'équivalence est vraie dans tout modèle, donc l'opérateur μ donne une fonction représentable si totale. □

Pour montrer la représentabilité du schéma de récursion primitif, nous aurons besoin d'abord d'un lemme nous permettant de représenter des séquences finies d'entiers. C'est-à-dire une fonction qui, pour un tuple $(k_1, \dots, k_p)$ associe une fonction $i \mapsto k_i$. C'est le lemme que nous allons montrer maintenant.

Lemme 4.3.3 (Fonction β de Gödel)**.** *Il existe une fonction β à trois arguments, représentable et récursive primitive, telle que pour toute suite finie $(k_0, \dots, k_p)$ il existe $b, c \in \mathbb{N}$ tels que $\beta(b, c, i) = k_i$ pour tout $0 \leq i \leq p$.*

Démonstration. Nous allons définir β de la façon suivante : $\beta(x, y, z) = x\%(1 + (z + 1)y)$ où $\%$ représente le reste par la division euclidienne. Sa définition en tant que fonction représentable et récursive primitive est évidente (puisqu'elle utilise l'arithmétique standard) en considérant que l'on peut représenter $a\%b$ par $R(r, a, b) := \exists q.a = b \times q + r \wedge 0 \leq r < b$. L'existence de b et c codant la séquence est une conséquence du théorème des restes chinois : si $x_0, \dots, x_p$ sont premiers entre eux deux à deux, et $y_0, \dots, y_p$ sont des entiers, alors il existe un entier α tel que

$$\begin{aligned} \alpha &\equiv y_0[x_0] \\ &\vdots \\ \alpha &\equiv y_n[x_n] \end{aligned}$$

Le but est alors de créer un système de congruences permettant de coder notre suite : de par la forme de notre système et de notre fonction, on souhaite que $y_i = k_i$ et on va définir $x_i = i \times y + 1$ en prenant y supérieur à tous les k_i tel que pour tout $i < p$, $y \times i + 1$ forme une suite finie de nombres premiers entre eux deux à deux. Le résultat tient alors dans le fait que l'on va prendre $b := \alpha$, $c := y$ et i variant de 0 à p. □

Exercice 4.3.4. Prouver qu'il existe bien un tel y en prenant $p!$: pour $1 \leq i, j \leq p+1$ où $i \neq j$, $i \times p! + 1$ et $j \times p! + 1$ sont premiers entre eux.

On peut désormais prouver la stabilité par récursion primitive :

Lemme 4.3.4 (Stabilité par récursion)**.** *L'ensemble des fonctions (totales) représentables est stable par schéma de récursion primitive.*

Démonstration. Pour construire notre schéma de récursion, prenons g et h deux fonctions représentables, et G, H les propositions les représentant. Soit B la proposition représentant la fonction β de Gödel, en prenant l'indice en premier argument. Soit

$$B'(x_0, x_1, x_2, x_3) := B(x_0, x_1, x_2, x_3) \wedge (\forall z < x_0, \neg B(z, x_1, x_2, x_3))$$

alors par un argument similaire à celui pour définir la représentation de l'opérateur μ, B' a la propriété suivante : si $\mathcal{M} \models \mathrm{PA}$, x est standard dans $\mathcal{M}$ et a, b, c sont des éléments de $|\mathcal{M}|$, alors $\mathcal{M} \models B'(x, a, b, c)$ implique que pour tout autre x' (standard ou non), $\mathcal{M} \models B'(x', a, b, c)$ implique $x = x'$. On définit f comme obtenue par le schéma de récursion primitive sur g et h (g pour le cas nul et h pour la récursion). Remarquons qu'alors, dans $\mathbb{N}$, $y = f(x_0, \ldots, x_k)$ si et seulement s'il existe une suite $z_0, \ldots, z_{x_0}$ telle que

$$z_0 = g(x_1, \ldots, x_k)$$

et

$$z_{i+1} = h(i, z_i, x_1, \ldots, x_k)$$

pour $i < x_0$, et $y = z_{x_0}$, ce qui revient à

$$F(y, x_0, \ldots, x_k) := \exists a. \exists b. \Big(\exists z_0. (B'(z_0, 0, a, b) \wedge G(z_0, x_1, \ldots, x_k)) \\ \wedge \forall i. (i < x_0) \rightarrow \exists z. \exists z'. (B'(z, i, a, b) \wedge B'(z', Si, a, b) \wedge H(z', i, z, x_1, \ldots, x_k)) \wedge B'(y, x_0, a, b) \Big)$$

Prouvons que F représente f, en prenant un modèle $\mathcal{M} \models \mathrm{PA}$, $n_0, \ldots, n_k \in \mathbb{N}$ et $c \in \mathcal{M}$:

- si c interprète $f(n_0, \ldots, n_k)$ alors en choisissant a et b pour générer la bonne suite, comme décrite plus tôt, on a bien $F(c, n_0, \ldots, n_k)$.
- réciproquement, si $\mathcal{M} \models F(c, \overline{n_0}, \ldots, \overline{n_k})$ alors il existe a et b ainsi que z_0 tels que $B'(z_0, 0, a, b)$ et $G(z_0, n_1, \ldots, n_k)$ donc $z_0 = g(n_1, \ldots, n_k)$, ensuite pour tout $i < n_0$ il existe z, z' tels que $B'(z, i, a, b)$ et $B'(z', Si, a, b)$ et $H(z', i, z, n_1, \ldots, n_k)$, ce qui signifie que la $i+1^{\mathrm{e}}$ composante de la suite finie définie par $\beta(a, b)$ est telle que $z_{i+1} = h(z_i, i, n_1, \ldots, n_k)$ donc (on peut le prouver par récurrence finie) on a une suite $z_0, \ldots, z_{n_0}$ respectant les conditions pour que $c = f(n_0, \ldots, n_k)$.

Donc f est représentée par F. □

On en déduit le théorème de représentation :

Théorème 4.3.1 (Représentation)**.** *Si f est une fonction récursive totale, alors elle est représentable dans PA.*

Démonstration. La preuve est une conséquence directe des lemmes précédents. □

Remarque 4.3.2. En réalité, nous ne nous sommes à aucun moment servi d'une récursion : ce théorème est valide directement dans PA_0, qu'on appelle l'arithmétique de Robinson. Remarquons aussi qu'avec le schéma de récursion on peut au contraire définir $+$ et $\times$ en tant que fonctions représentables. Enfin, pointons le fait que la représentabilité de $\times$ est essentielle pour le théorème de représentation à travers la définition de la fonction β de Gödel : il est nécessaire de pouvoir représenter par exemple % qui nécessite d'employer la multiplication.

En effet, l'arithmétique n'utilisant que l'addition et pas la multiplication, appelée arithmétique de Presburger,

est décidable même par automate fini. Nous verrons que ce n'est pas du tout le cas pour PA (ni PA_0).

Remarque 4.3.3 (A propos du choix de PA)**.** Pourquoi avoir décidé de présenter ces théorèmes dans PA plutôt que PA_0 alors que cela ne fait aucune différence ? Simplement d'un point de vue pédagogique : l'arithmétique de Robinson étant plus faible, certaines choses « inattendues » peuvent s'y produire, là où le comportement des modèles de PA, bien que parfois surprenant (les modèles non standards de PA sont durs à conceptualiser mentalement), est très régulier puisqu'on peut y coller son interprétation de $\mathbb{N}$, encore plus en sachant que $\mathbb{N}$ est isomorphe à un segment initial de tout modèle de PA.

Nous avons donc moyen d'enrichir, en utilisant des propositions (tout de même illisibles), notre langage en utilisant des relations fonctionnelles. On pourrait par exemple définir l'exponentiation, et globalement toutes les fonctions auxquelles on pense (oui, la fonction $x \mapsto \lfloor \ln(x) \rfloor$ peut se représenter dans PA).

4.4 Premier théorème d'incomplétude

Maintenant que nous avons montré que toute fonction récursive totale était représentable, nous pouvons nous attaquer au cœur de la démonstration du premier théorème d'incomplétude de Gödel. La preuve se fait en deux parties : on montre que l'on peut représenter la fonction reconnaissant, étant donnée une proposition, si elle est vraie, puis on effectue un argument diagonal (un peu comme pour le problème de l'arrêt) pour construire une proposition dont la preuve implique la preuve de sa propre négation.

La première partie peut se voir comme la réciproque de la fin de la section précédente : après avoir montré que les propositions pouvaient représenter des fonctions, on va montrer que les fonctions peuvent représenter des propositions. Pour cela, la première étape est de coder nos propositions en tant que nombre. Il y a de nombreuses façons de faire, et la façon historique est d'utiliser la décomposition en facteurs premiers d'un nombre, mais nous utiliserons plutôt une autre représentation.

4.4.1 Coder la logique dans PA

Définition 4.4.1. *Il existe une fonction représentable $\langle -, - \rangle : \mathbb{N}^2 \to \mathbb{N}$ bijective, et deux fonctions $\pi_1, \pi_2 : \mathbb{N} \to \mathbb{N}$ toutes trois récursives et représentables, telles que $\pi_1\langle x, y\rangle = x$ et $\pi_2\langle x, y\rangle = y$.*

Démonstration. L'exercice 3.2.3 consistait justement à prouver la bijectivité de $\langle -, - \rangle$ et son caractère récursif primitif (donc récursif). On pourrait s'en contenter, mais nous allons donner une représentation simple de cette fonction :

$$z = \langle x, y\rangle \iff 2 \times z = (x + y) \times (x + y + 1) + y$$

et les représentations de π_1 (et on se doute que π_2 sera construit de façon analogue) :

$$x = \pi_1(z) \iff \exists y.(y \leq z) \wedge (2 \times z = (x + y) \times (x + y + 1) + y)$$

On vérifie facilement que $\pi_1\langle x, y\rangle = x$ puisque cette proposition est équivalente à l'existence de y' tel que $\langle x, y\rangle = \langle x, y'\rangle$, ce qui est vrai pour $y' = y$ de façon évidente (et pas pour un y' inférieur puisque notre bijection est croissante, à x fixé, en y). □

Cette fonction sera notre brique essentielle pour coder les propositions, que nous coderons comme des arbres binaires. Une autre façon de les représenter aurait pu être comme des chaînes de caractères, càd donner un nombre précis à chaque caractère et coder notre formule ainsi, mais le codage que nous allons avoir sera, somme toute, plus élégant.

Remarque 4.4.1. Si à l'époque de Gödel, l'idée de coder uniquement par l'arithmétique des notions aussi complexes que la logique était révolutionnaire, il est beaucoup moins choquant aujourd'hui d'imaginer tout mettre en binaire et faire des opérations infiniment complexes avec de longues suites d'opérations arithmétiques simples sur

ces suites binaires : nos ordinateurs le font en permanence.

Plutôt que de généraliser notre proposition avec une représentation bijective $\langle -, - \rangle : \mathbb{N}^k \to \mathbb{N}$, on se contentera d'écrire des $k-uplets$ comme des couples imbriqués vers la droite : $\langle a, b, c \rangle$ sera codé par $\langle a, \langle b, c \rangle \rangle$.

Coder les termes

Nous pouvons maintenant coder les termes dans le langage de l'arithmétique :

Définition 4.4.2 (Codage d'un terme)**.** *Soit un terme t, on définit la fonction $\ulcorner - \urcorner$ qui associe à un terme un entier, de façon injective et représentable, par induction sur t :*

- *si $t = 0$ alors on associe l'entier $\langle 0, 0 \rangle$*
- *so $t = x_n$ une variable, on associe l'entier $\langle 0, n+1 \rangle$*
- *si $t = St'$ alors on associe l'entier $\langle 1, \ulcorner t' \urcorner \rangle$*
- *si $t = t' + t''$ alors on associe l'entier $\langle 2, \langle \ulcorner t' \urcorner, \ulcorner t'' \urcorner \rangle \rangle$*
- *si $t = t' \times t''$ alors on associe l'entier $\langle 3, \langle \ulcorner t' \urcorner, \ulcorner t'' \urcorner \rangle \rangle$*

Démonstration. La fonction est injective par récursion sur le terme : à chaque constructeur différent d'un terme on associe un premier argument différent dans le couple qu'on code, et le codage du couple est bijectif donc injectif. Sa représentabilité découle purement de celle de $\langle -, - \rangle$. □

Lemme 4.4.1. *L'ensemble* $\text{Term} = \{\ulcorner t \urcorner \mid t$ *est un terme sur le langage de l'arithmétique*$\}$ *est récursif primitif.*

Démonstration. Construisons la fonction caractéristique en donnant un algorithme (qui fait une récursion dont le nombre d'étapes est strictement inférieur au nombre codé, d'où le caractère primitif) calculant $1_{\text{Term}}(t)$:

- si $\pi_1(t) = 0$ alors $1_{\text{Term}}(t) = 1$
- si $\pi_1(t) = 1$ alors $1_{\text{Term}}(t) = 1_{\text{Term}}(\pi_2(t))$
- si $\pi_1(t) = 2$ ou $\pi_1(t) = 3$ alors $1_{\text{Term}}(t) = \min(1_{\text{Term}}(\pi_2(\pi_2(t))), 1_{\text{Term}}(\pi_2(\pi_2(t))))$
- sinon, $1_{\text{Term}}(t) = 0$

□

Coder les propositions et agir dessus

De même que les termes sont des objets inductifs, les propositions le sont, et le codage sera très similaire :

Définition 4.4.3 (Codage des propositions)**.** *On définit la fonction de code $\ulcorner - \urcorner$ sur les propositions de la façon suivante, pour P une proposition :*

- *on remplacera toutes les occurrences de $\top$ par $0 = 0$ et $\bot$ par $0 = 1$ (qui reviennent au même étant donné qu'on peut prouver que $\neg(0 = 1)$ et que $0 = 0$ est vrai aussi)*
- *si $P = (t_1 = t_2)$ alors on associe l'entier $\langle 0, \langle \ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner \rangle \rangle$*
- *si $P = \neg P'$ alors on associe l'entier $\langle 1, \ulcorner P \urcorner \rangle$*
- *si $P = P_1 \vee P_2$ alors on associe l'entier $\langle 2, \langle \ulcorner P_1 \urcorner, \ulcorner P_2 \urcorner \rangle \rangle$*
- *si $P = P_1 \wedge P_2$ alors on associe l'entier $\langle 3, \langle \ulcorner P_1 \urcorner, \ulcorner P_2 \urcorner \rangle \rangle$*
- *si $P = \exists x_i . P'$ alors on associe l'entier $\langle 4, \langle i, \ulcorner P' \urcorner \rangle \rangle$*
- *si $P = \forall x_i . P'$ alors on associe l'entier $\langle 5, \langle i, \ulcorner P' \urcorner \rangle \rangle$*

qui est injective et représentable, par le même argument que précédemment.

Lemme 4.4.2. *L'ensemble* $\text{Prop} = \{\ulcorner P \urcorner \mid P \text{ est une proposition sur le langage de l'arithmétique}\}$ *est primitif récursif.*

Exercice 4.4.1. Prouver ce lemme.

Nous voulons maintenant coder la substitution de propositions, c'est-à-dire une fonction prenant en entrée un entier i et deux codes de termes $\ulcorner t \urcorner$ et $\ulcorner u \urcorner$ et retourne $\ulcorner t[x_i/u] \urcorner$, qui nous servira lorsque l'on voudra travailler sur une règle comme $\forall_\text{e}$. De plus, on aura besoin d'un lemme pour caractériser la liberté d'une variable dans une proposition (par exemple pour la règle $\forall_\text{i}$).

Lemme 4.4.3. *Les ensembles suivants sont récursifs primitifs :*

- $\Theta_0 = \{\langle \ulcorner t \urcorner, n\rangle \mid t \text{ est un terme et } x_n \text{ n'a pas d'occurrence dans } t\}$
- $\Theta_1 = \{\langle \ulcorner t \urcorner, n\rangle \mid t \text{ est un terme et } x_n \text{ au au moins une occurrence dans } t\}$
- $\Phi_0 = \{\langle \ulcorner P \urcorner, n\rangle \mid P \text{ est une proposition et } x_n \text{ n'a pas d'occurrence dans } P\}$
- $\Phi_1 = \{\langle \ulcorner P \urcorner, n\rangle \mid P \text{ est une proposition et } x_n \text{ n'a pas d'occurrence libre dans } P\}$
- $\Phi_2 = \{\langle \ulcorner P \urcorner, n\rangle \mid P \text{ est une proposition et } x_n \text{ n'a pas d'occurrence liée dans } P\}$
- $\Phi_2 = \{\langle \ulcorner P \urcorner, n\rangle \mid P \text{ est une proposition et } x_n \text{ a au moins une occurrence libre dans } P\}$
- $\Phi_2 = \{\langle \ulcorner P \urcorner, n\rangle \mid P \text{ est une proposition et } x_n \text{ a au moins une occurrence liée dans } P\}$
- $\Phi_5 = \{\ulcorner P \urcorner \mid P \text{ est une formule close}\}$

Exercice 4.4.2. Prouver ce lemme (on fera pour chaque cas une récurrence et une disjonction de cas sur la forme du code).

Lemme 4.4.4 (Substitution)**.** *Il existe deux fonctions primitives récursives* Subst*, que l'on notera ici* Subst_t *et* Subst_p *respectivement, qui étant donnés deux terme* t, u *et une proposition* P*, sont telles que :*

$$\text{Subst}_t(n, \ulcorner t \urcorner, \ulcorner u \urcorner) = \ulcorner u[x_n/t] \urcorner \qquad \text{Subst}_p(n, \ulcorner t \urcorner, \ulcorner P \urcorner) = \ulcorner P[x_n/t] \urcorner$$

Démonstration. On définit la fonction Subst_t par récursion et disjonction de cas sur $k := \ulcorner u \urcorner$:

- si $\pi_1(k) = 0$ alors si $\pi_2(k) = n + 1$, alors $\text{Subst}_t(n, \ulcorner t \urcorner, k) = \ulcorner t \urcorner$, si $\pi_2(k) \neq n + 1$ alors $\text{Subst}_t(n, \ulcorner t \urcorner, k) = k$.
- si $\pi_1(k) = 1$ alors $\text{Subst}_t(n, \ulcorner t \urcorner, k) = \langle 1, \text{Subst}_t(n, \ulcorner t \urcorner, \pi_2(k))\rangle$
- si $\pi_2(k) = 2$ ou $\pi_2(k) = 3$ alors
$$\text{Subst}_t(n, \ulcorner t \urcorner, k) = \langle \pi_1(k), \langle \text{Subst}_t(n, \ulcorner t \urcorner, \pi_1(\pi_2(k))), \text{Subst}_t(n, \ulcorner t \urcorner, \pi_2(\pi_2(k)))\rangle\rangle$$
- sinon, on associe 0 (cela n'a pas d'importance)

De même on définit Subst_p par récursion et disjonction de cas sur $k := \ulcorner P \urcorner$:

- si $\pi_1(k) = 0$ alors
$$\text{Subst}_p(n, \ulcorner t \urcorner, k) = \langle 0, \langle \text{Subst}_t(n, \ulcorner t \urcorner, \pi_1(\pi_2(k))), \text{Subst}_t(n, \ulcorner t \urcorner, \pi_2(\pi_2(k)))\rangle\rangle$$
- si $\pi_1(k) = 1$ alors $\text{Subst}_p(n, \ulcorner t \urcorner, k) = \langle 1, \text{Subst}_p(n, \ulcorner t \urcorner, \pi_2(k))\rangle$
- si $\pi_1(k) = 2$ ou $\pi_1(k) = 3$ alors
$$\text{Subst}_p(n, \ulcorner t \urcorner, k) = \langle \pi_1(k), \langle \text{Subst}_p(n, \ulcorner t \urcorner, \pi_1(\pi_2(k))), \text{Subst}_p(n, \ulcorner t \urcorner, \pi_2(\pi_2(k)))\rangle\rangle$$
- si $\pi_1(k) = 4$ ou $\pi_1(k) = 5$, alors on regarde $j = \pi_1(\pi_2(k))$: si $j = n$ alors $\text{Subst}_p(n, \ulcorner t \urcorner, k) = k$ et sinon $\text{Subst}_p(n, \ulcorner t \urcorner, k) = \langle \pi_1(k), \langle j, \text{Subst}_p(n, \ulcorner t \urcorner, \pi_2(\pi_2(k)))\rangle\rangle$
- sinon, on associe 0 (cela n'a pas d'importance)

□

Dans la suite, on dira simplement Subst puisque Subst_t ne sert qu'à définir Subst_p, et c'est donc la dernière fonction qui sera désignée.

Coder un séquent

Nous allons maintenant définir un séquent, de la forme $\Gamma \vdash P$, pour pouvoir ensuite construire nos preuves.

Définition 4.4.4 (Codage d'un contexte)**.** *On définit le codage d'un contexte par* $\ulcorner \varnothing \urcorner = 0$ *et* $\ulcorner \Gamma, P \urcorner = 1 + \langle \ulcorner \Gamma \urcorner, P \rangle$.

Lemme 4.4.5 (Contexte récursif et décodage unique)**.** *La fonction de codage de contextes est injective, et l'ensemble* $\mathrm{Ctx} = \{\ulcorner \Gamma \urcorner \mid \Gamma$ *est un contexte*$\}$ *est primitif récursif.*

Démonstration. L'injectivité peut se montrer par double récurrence sur la longueur de deux contextes Γ, Γ' : seul le contexte vide est envoyé sur 0, et si $\ulcorner \Gamma \urcorner = \ulcorner \Gamma' \urcorner$ alors cette égalité est aussi vraie quand on retire 1, et donc les deux contextes codent la même dernière proposition P (parce que le codage de proposition est injectif et que la fonction donnant une paire est injective), ce qui signifie que les codes de leur contexte sans la dernière proposition sont égaux, ce qui par hypothèse de récurrence signifie que les deux contextes sont égaux. Le fait d'être primitif récursif est juste l'application de la même idée : on applique récursivement le test de supprimer 1, de vérifier si la composante de droite est le code d'une une proposition, puis de vérifier si la composante de gauche est le code d'un contexte. □

Un point important est alors le suivant :

Lemme 4.4.6 (Liberté dans un contexte)**.** *Vérifier si une variable donnée est libre dans un contexte est primitif récursif.*

Démonstration. On applique simplement la fonction de test de liberté de la variable à chaque proposition du contexte, récursivement. □

De plus, pour facilement utiliser la règle d'axiome, nous allons utiliser le lemme suivante :

Lemme 4.4.7. *Vérifier si une proposition appartient à un contexte est primitif récursif.*

Exercice 4.4.3. Prouver le lemme précédent.

Coder les preuves

Nous en venons alors à coder nos arbres de preuves. L'idée première serait d'écrire toutes les informations de l'arbre, ce qui donnerait un nombre énorme de données à coder (un contexte à chaque fois, déjà). Une analyse plus fine nous permet de remarquer qu'il est inutile d'aller jusque là, car si l'on part de la racine où l'on fixe le séquent, l'application d'une règle nous permet toujours de deviner le contexte résultant. Ceci signifie que coder un arbre de preuves peut se faire en ne codant qu'un arbre de couples avec une étiquette (pour le numéro de la règle) et la proposition à prouver, puis la liste des sous-arbres de preuves. Cependant, l'avantage de coder tous les contextes est de faire une définition inductive simple.

Définition 4.4.5 (Codage d'un arbre)**.** *On définit donc la fonction* $\ulcorner - \urcorner$ *sur les arbres de preuve inductivement de la façon suivante :*

- *si la règle est une utilisation d'axiome sur* $\Gamma \vdash P$ *alors le code sera* $\langle 0, \ulcorner P \urcorner \rangle$.
- *si la règle est un* $\to_{\mathrm{i}}$ *appliqué à* $\Gamma \vdash P \to Q$ *alors le code sera* $\langle 1, \langle \ulcorner Q \urcorner, \ulcorner \alpha \urcorner \rangle \rangle$ *où* α *est un arbre de preuve de* $\Gamma, P \vdash Q$.
- *si la règle est un* $\to_{\mathrm{e}}$ *appliqué à* $\Gamma \vdash Q$ *alors le code sera* $\langle 2, \langle \langle \ulcorner P \to Q \urcorner, \ulcorner \alpha \urcorner \rangle, \langle \ulcorner Q \urcorner, \ulcorner \beta \urcorner \rangle \rangle \rangle$ *où* α *est un arbre de preuve de* $\Gamma \vdash P \to Q$ *et* β *est un arbre de preuve de* $\Gamma \vdash Q$.
- *si la règle est un* $\wedge_{\mathrm{i}}$ *appliqué à* $\Gamma \vdash P \wedge Q$ *alors le code sera, pour* α *un arbre de preuve de* $\Gamma \vdash P$ *et* β *un arbre de preuve de* $\Gamma \vdash Q$ *:* $\langle 3, \langle \langle \ulcorner P \urcorner, \ulcorner \alpha \urcorner \rangle, \langle \ulcorner Q \urcorner, \ulcorner \beta \urcorner \rangle \rangle \rangle$

- *si la règle est un* $\wedge_{\mathrm{e}}^{\mathrm{g}}$ *appliqué à* $\Gamma \vdash P$ *alors le code sera, pour* α *un arbre de preuve de* $\Gamma \vdash P \wedge Q$ *:* $\langle 4, \langle \ulcorner P \wedge Q \urcorner, \ulcorner \alpha \urcorner \rangle \rangle$
- *si la règle est un* $\wedge_{\mathrm{e}}^{\mathrm{d}}$ *appliqué à* $\Gamma \vdash Q$ *alors le code sera, pour* α *un arbre de preuve de* $\Gamma \vdash P \wedge Q$ *:* $\langle 5, \langle \ulcorner P \wedge Q \urcorner, \ulcorner \alpha \urcorner \rangle \rangle$
- *si la règle est un* $\vee_{\mathrm{i}}^{\mathrm{g}}$ *appliqué à* $\Gamma \vdash P \vee Q$ *alors le code sera, pour* α *un arbre de preuve de* $\Gamma \vdash P$ *:* $\langle 6, \langle \ulcorner P \urcorner, \ulcorner \alpha \urcorner \rangle \rangle$
- *si la règle est un* $\vee_{\mathrm{i}}^{\mathrm{d}}$ *appliqué à* $\Gamma \vdash P \vee Q$ *alors le code sera, pour* α *un arbre de preuve de* $\Gamma \vdash Q$ *:* $\langle 7, \langle \ulcorner Q \urcorner, \ulcorner \alpha \urcorner \rangle \rangle$
- *si la règle est un* $\vee_{\mathrm{e}}$ *appliqué à* $\Gamma \vdash C$ *alors, pour* α *un arbre de preuve de* $\Gamma \vdash P \vee Q$*,* β *un arbre de preuve de* $\Gamma, P \vdash C$ *et* γ *un arbre de preuve de* $\Gamma, Q \vdash C$*, le code sera*

$$\langle 8, \langle \langle \ulcorner P \vee Q \urcorner, \ulcorner \alpha \urcorner \rangle, \langle \langle \ulcorner C \urcorner, \ulcorner \beta \urcorner \rangle, \langle \ulcorner C \urcorner, \ulcorner \gamma \urcorner \rangle \rangle \rangle$$

- *si la règle est un* $\neg_{\mathrm{i}}$ *appliqué à* $\Gamma \vdash \neg A$ *alors pour* α *un arbre de preuve de* $\Gamma \vdash (0 = 1)$ *le code sera* $\langle 9, \langle \ulcorner 0 = 1 \urcorner, \alpha \rangle \rangle$
- *si la règle est un* $\neg_{\mathrm{e}}$ *appliqué à* $\Gamma \vdash (0 = 1)$ *alors pour* α *un arbre de preuve de* $\Gamma \vdash P$ *et* β *un arbre de preuve de* $\Gamma \vdash \neg P$ *le code sera* $\langle 10, \langle \langle \ulcorner P \urcorner, \ulcorner \alpha \urcorner \rangle, \langle \ulcorner \neg P \urcorner, \ulcorner \beta \urcorner \rangle \rangle \rangle$
- *si la règle est un* $\bot_{\mathrm{c}}$ *appliqué à* $\Gamma \vdash P$ *et que* α *est un arbre de preuve de* $\Gamma, \neg P \vdash (0 = 1)$ *alors le code sera* $\langle 11, \langle \ulcorner 0 = 1 \urcorner, \ulcorner \alpha \urcorner \rangle \rangle$
- *si la règle est* $\forall_{\mathrm{i}}$ *appliqué à* $\Gamma \vdash \forall x_n.P$ *et que* α *est un arbre de preuve de* $\Gamma \vdash P$ *alors le code sera* $\langle 12, \langle \ulcorner P \urcorner, \ulcorner \alpha \urcorner \rangle \rangle$
- *si la règle est* $\forall_{\mathrm{e}}$ *appliqué à* $\Gamma \vdash P[x_n/a]$ *et que* α *est un arbre de preuve de* $\Gamma \vdash \forall x_n.P$ *alors le code sera* $\langle 13, \langle \ulcorner \forall x_n.P \urcorner, \ulcorner \alpha \urcorner \rangle \rangle$
- *si la règle est* $\exists_{\mathrm{i}}$ *appliqué à* $\Gamma \vdash \exists x_n.P$ *et que* α *est un arbre de preuve de* $\Gamma \vdash P[x_n/t]$ *alors le code sera* $\langle 14, \langle \ulcorner P[x_n/t] \urcorner, \ulcorner \alpha \urcorner \rangle \rangle$
- *si la règle est* $\exists_{\mathrm{e}}$ *appliqué à* $\Gamma \vdash C$*, que* α *est un arbre de preuve de* $\Gamma \vdash \exists x_n.P$ *et que* β *est un arbre de preuve de* $\Gamma, P \vdash C$ *alors le code sera* $\langle 15, \langle \langle \ulcorner \exists x_n.P \urcorner \ulcorner \alpha \urcorner \rangle, \langle \ulcorner C \urcorner, \ulcorner \beta \urcorner \rangle \rangle \rangle$
- *si la règle est* $=_{\mathrm{i}}$ *appliqué à* $\Gamma \vdash t = t$*, alors le code sera* $\langle 16, \ulcorner t = t \urcorner \rangle$
- *si la règle est* $=_{\mathrm{e}}$ *appliqué à* $\Gamma \vdash P[x_n/u]$*, que* α *est un arbre de preuve de* $\Gamma \vdash t = u$ *et que* β *est un arbre de preuve de* $\Gamma \vdash P[x_n/t]$ *alors le code sera* $\langle 17, \langle \langle \ulcorner t = u \urcorner, \ulcorner \alpha \urcorner \rangle, \langle \ulcorner P[x_n/t] \urcorner, \ulcorner \beta \urcorner \rangle \rangle \rangle$
- *sinon, on associe n'importe quoi, disons* 0

On a alors la proposition suivante :

Proposition 4.4.1. *Le langage* $\mathrm{Proof}_\Gamma = \{\langle \ulcorner P \urcorner, \ulcorner \alpha \urcorner \rangle \mid \alpha$ *est un arbre de preuve valide de* $\Gamma \vdash P\}$ *est récursif si* Γ *est un contexte récursif.*

Exercice 4.4.4. Prouver cette proposition. *Indication : on raisonnera par récurrence, et on utilisera les lemmes pour les substitutions et la liberté des variables.*

Remarque 4.4.2. Le résultat est récursif primitif à l'exception du fait de vérifier la règle axiome, dont la complexité dépend directement du contexte.

De plus, on a le lemme suivant :

Lemme 4.4.8 (Extraction d'une proposition)**.** *Si* $k \in \mathrm{Proof}_\Gamma$ *alors on peut trouver une proposition* P *telle que* k *code une démonstration de* P*.*

Démonstration. Il suffit d'appliquer des projections. □

4.4.2 Prédicat de démonstration et diagonalisation

On déduit alors le théorème suivant :

Théorème 4.4.1. *Soit T une théorie récursivement axiomatisable, il existe une proposition à une variable libre, notée* Dem_T *telle que* $\mathrm{Dem}_T(\lceil P \rceil)$ *si et seulement si P est démontrable dans la théorie T.*

Démonstration. On utilise la représentation de π_1 qui extrait du code d'un arbre la proposition prouvée, et de Proof_T qui dit qu'un nombre correspond à un arbre valide dans la théorie T, puis on définit

$$\mathrm{Dem}_T(\lceil P \rceil) = \exists z.\mathrm{Proof}_T(z) \wedge \pi(\lceil P \rceil, z)$$

qui est, par définition, équivalente au fait qu'il existe un arbre de preuve valide de racine P, c'est-à-dire une preuve de P, dans la théorie T. □

Remarque 4.4.3. La récursivité nous a permis de justifier la représentabilité de la fonction vérifiant un arbre de preuve, et celle de la théorie axiomatique utilisée nous a permis de justifier, plus précisément, la représentabilité au niveau de la règle axiome.

Nous pouvons alors énoncer la premier théorème d'incomplétude, de Gödel :

Théorème 4.4.2 (Premier théorème d'incomplétude)**.** *Soit T une théorie récursivement axiomatisable, pouvant prouver les axiomes de* PA_0 *et cohérente, alors il existe une proposition G telle que ni G ni $\neg G$ ne sont démontrables dans T.*

Démonstration. Avec la construction précédente, nous savons qu'il existe dans T un codage $\lceil - \rceil$ des propositions et un prédicat Dem_T tel que $\mathrm{Dem}_T(\lceil P \rceil)$ si et seulement si P est démontrable dans T. De plus, on note Subst la fonction représentant la fonction Subst. On construit alors la proposition

$$\Delta(x_0) := \forall z.(\mathrm{Subst}(z, 0, x_0, x_0) \implies \neg \mathrm{Dem}_T(z))$$

qui signifie donc « toute proposition appliquée à son propre code ne peut être démontrée » et on appelle $G := \Delta(\lceil \Delta \rceil)$

Cela signifie que $\mathrm{Subst}(G, 0, \lceil \Delta \rceil, \lceil \Delta \rceil)$ donc si G est démontrable, alors en spécialisant G en Δ, on aurait $\neg \mathrm{Dem}_T(\lceil G \rceil)$. Au contraire, si $\neg G$ était démontrable, alors on aurait un code z tel que $\mathrm{Subst}(z, 0, \lceil \Delta \rceil, \lceil \Delta \rceil)$ et $\mathrm{Dem}_T(z)$, ce qui signifie donc que $\mathrm{Dem}_T(\lceil G \rceil)$, donc que G est démontrable, ce qui est absurde. □

Corollaire 4.4.1. *Les théories* PA_0 *et* PA *ne sont pas complètes.*

Une conséquence assez directe est la suivante :

Théorème 4.4.3 (Church)**.** *Soit $\mathcal{L}$ un langage contenant l'arithmétique, alors l'ensemble des théorèmes logiques sur $\mathcal{L}$ n'est pas récursif : la théorie $\varnothing$ n'est pas décidable.*

Démonstration. On définit $\mathcal{T} = \bigvee_{i=1}^{7} A_i$ la conjonction des axiomes de PA_0, et on va noter T_0 l'ensemble des théorèmes logiques sur $\mathcal{L}$. On peut vérifier facilement que $\mathrm{PA}_0 \vdash P$ si et seulement si $\vdash \mathcal{T} \implies P$, càd $(\mathcal{T} \implies P) \in T_0$, or si T_0 était récursive cela signifierait que PA_0 serait décidable : on sait que ça n'est pas le cas puisqu'il existe des énoncés de PA_0 indécidables. □

Remarque 4.4.4. La théorie des ensembles usuelle, ZF, permet de formaliser l'arithmétique (nous le verrons dans le prochain chapitre), ce qui signifie que c'est une théorie (récursivement axiomatisable aussi) qui peut démontrer les axiomes de PA_0, donc elle n'est elle non plus pas complète.

4.5 Sur le deuxième théorème d'incomplétude

Le second théorème d'incomplétude est considéré comme une conséquence du premier, mais utilisant des hypothèses plus fortes. La première chose à remarquer est que la définition de Dem_T signifie qu'on peut définir une proposition $\mathrm{Coh}(T) := \neg\mathrm{Dem}_T(\bot)$, qui signifie que la théorie T est cohérente (la proposition dit précisément que T ne démontre pas l'absurde). Le second théorème d'incomplétude nous dit alors que $T \nvdash \mathrm{Coh}(T)$. Paul Bernays a précisé trois conditions essentielles sur T pour permettre de démontrer le second théorème d'incomplétude.

Définition 4.5.1 (Conditions de Hilbert-Bernays-Löb)**.** *Les conditions sont les suivantes :*

- *Si P est démontrable dans T, alors $\mathrm{Dem}_T(\ulcorner P \urcorner)$ est démontrable dans T.*
- *On peut démontrer dans T la proposition $\mathrm{Dem}_T(\ulcorner P \urcorner) \implies \mathrm{Dem}_T(\ulcorner \mathrm{Dem}_T(\ulcorner P \urcorner) \urcorner)$*
- *On peut démontrer dans T la proposition*

$$\mathrm{Dem}_T(\ulcorner P \implies Q \urcorner) \implies \mathrm{Dem}_T(\ulcorner P \urcorner) \implies \mathrm{Dem}_T(\ulcorner Q \urcorner)$$

Ces conditions sont vérifiées dans PA, mais pas dans PA_0. En effet, la deuxième condition porte sur le fait de pouvoir encoder notre preuve de « validité » de Dem_T : il faut pour cela utiliser la récurrence (toutes nos preuves sur le codage et sur la vérification qu'un arbre est valide reposent sur des récurrences). Les deux autres conditions, elles, sont principalement techniques.

Si on veut expliciter ce que signifie chacune : la première condition énonce simplement que notre prédicat de prouvabilité est valide. La deuxième condition, elle, dit que l'on peut formaliser la première condition au sein de la théorie T. La troisième condition dit que notre encodage est stable par *modus ponens*, qui tient principalement au fait que nous possédons la règle $\to_{\mathrm{e}}$ dans notre logique (et donc dans notre définition de Dem_T). Remarquons qu'alors, en utilisant le fait que $\neg P \equiv P \implies \bot$, on peut écrire avec la troisième condition que

$$\mathrm{Dem}_T(\neg P) \implies \mathrm{Dem}_T(P) \implies \neg\mathrm{Coh}(T)$$

Nous n'allons pas démontrer que ces trois conditions sont respectées, et nous contenterons des arguments donnés plus haut. Une preuve rigoureuse est possible, mais elle est principalement technique.

Nous avons alors

Théorème 4.5.1 (Second théorème d'incomplétude)**.** *Si une théorie T, récursivement axiomatisable et cohérente, permet de démontrer les axiomes de* PA*, alors $T \nvdash \mathrm{Coh}(T)$, donc on ne peut prouver la cohérence d'une théorie au sein de cette théorie.*

Démonstration. La théorie T vérifie les conditions de Hilbert-Bernays-Löb, puisqu'elle démontre les axiomes de PA et est récursivement axiomatisable. Soit G la formule construire dans le premier théorie d'incomplétude, dont on ne peut prouver ni la démontrabilité ni la non démontrabilité.

Si T est cohérente, cela entraine $T \vdash \neg\mathrm{Dem}_T(\ulcorner G \urcorner)$ par définition de G. La deuxième condition, appliquée au premier théorème d'incomplétude, nous permet de déduire que $T \vdash \mathrm{Dem}_T(\ulcorner G \implies \neg\mathrm{Dem}_T(\ulcorner G \urcorner) \urcorner)$, donc avec la troisième condition

$$T \vdash \mathrm{Dem}_T(\ulcorner G \urcorner) \implies \mathrm{Dem}_T(\ulcorner \neg\mathrm{Dem}_T(\ulcorner G \urcorner) \urcorner)$$

or en utilisant la deuxième condition sur $\mathrm{Dem}_T(\ulcorner G \urcorner)$ (à gauche dans la formule) et la troisième condition dans le cas particulier de la négation, on en déduit que

$$T \vdash \mathrm{Dem}_T(\ulcorner G \urcorner) \implies \neg\mathrm{Coh}(T)$$

ce qui, par définition de G, signifie que $T \vdash \neg G \implies \neg\mathrm{Coh}(T)$ donc, par contraposée

$$T \vdash \mathrm{Coh}(T) \implies G$$

or G n'est pas démontrable dans T, donc $\mathrm{Coh}(T)$ non plus : $T \nvdash \mathrm{Coh}(T)$. □

Remarque 4.5.1. D'après le second théorème d'incomplétude, pour une théorie T (ayons PA à l'esprit) respectant les conditions précédentes, alors $T \cup \{\text{Coh}(T)\}$ est cohérente, et même $T \cup \{\neg\text{Coh}(T)\}$ l'est. Si cela peut sembler perturbant, il faut garder à l'esprit que les deux prédicats $\text{Coh}(T)$ et $\neg\text{Coh}(T)$ n'ont de sens que dans T : dans la nouvelle théorie à laquelle on ajoute l'axiome que T est cohérente, alors $\text{Dem}_{T'}$ est un prédicat qui n'a plus rien à voir, et il ne faut donc certainement pas comprendre que $T \cup \{\neg\text{Coh}(T)\}$ est par essence incohérente : elle l'est autant que T (donc généralement pas).

Remarque 4.5.2. Si l'on ne peut prouver la cohérence d'une théorie au sein même de cette théorie, on peut bien prouver la cohérence d'une théorie si l'on dispose d'une théorie plus forte. Par exemple, Gentzen a démontré dans ZF (la théorie usuelle des ensembles) que l'arithmétique de Peano était cohérente, en effectuant une récursion transfinie jusqu'à ε_0.

Chapitre 5

Théorie des ensembles

En parallèle du développement de la logique, et de sa focalisation sur l'arithmétique, les logiciens, après le travail de Cantor, ont commencé à comprendre qu'il était possible d'exprimer de nombreux phénomènes à travers les ensembles. Si l'arrivée du paradoxe de Russell a d'abord fait croire à la défaite de cette théorie, l'axiomatisation de la théorie des ensembles par Zermelo et Fraenkel a ouvert la voie à l'unification des mathématiques. Nous allons donc étudier d'abord quels sont ces axiomes, puis comment encoder les mathématiques de base dedans. La suite du chapitre sera dédiée à la théorie des ordinaux et des cardinaux, qui sont des outils majeurs de la théorie des ensembles.

5.1 Théorie naïve des ensembles

Avant de voir les axiomes de la théorie ZF, nous allons parler de la théorie naïve des ensembles. L'idée première d'un ensemble est de regrouper des objets mathématiques : on décrit aisément des ensembles par énumération de leurs éléments, comme $\{1, 2, 3\}$, mais le point central est de pouvoir décrire des ensembles par compréhension, par exemple $\{x \mid \exists y.x = 2 \times y\}$ est un ensemble dans lequel on décrit les élément x qui sont le double d'au moins un élément. Remarquons ici que notre quantification du premier ordre fait perdre le sens de ce certains objets : parlons-nous de x dans $\mathbb{N}$, dans $\mathbb{R}$? L'ensemble décrit sera fondamentalement différent. Mais outre ce problème mineur (que l'on peut régler en s'accordant sur le sens qu'on donne aux opérations par exemple), le vrai problème réside dans le paradoxe de Russell.

Rappelons que le langage des ensembles est la signature $\{\in, =\}$ où les deux symboles sont des symboles de relation binaires, le premier exprimant l'appartenance d'un élément à un ensemble, e.g. $2 \in \{1, 2, 3\}$.

Nous allons d'abord donner des définitions :

Définition 5.1.1 (Ensemble, axiome naïf de compréhension)**.** *On appelle ensemble une collection d'objets mathématiques. Ces collections possèdent la propriété suivante : si P est une proposition à une variable libre sur le langage des ensembles, alors il existe un ensemble E_P dont les éléments sont exactement les objets qui vérifient P.*

Le paradoxe de Russell peut alors s'énoncer :

Proposition 5.1.1. *La théorie naïve des ensembles est incohérente.*

Démonstration. Notons X l'ensemble défini par la proposition $x \notin x$. Alors si $X \in X$, on en déduit que $X \notin X$ par définition de X, et si $X \notin X$ alors $X \in X$ par définition de X : on arrive dans tous les cas à une contradiction. □

Remarque 5.1.1. Si on voit $\in$ comme une relation binaire sur l'univers utilisé dans notre théorie des ensemble, alors $x \in x$ exprime exactement la diagonale de cette relation, et $x \notin x$ est donc sa négation : on a là encore un argument diagonal.

La théorie naïve des ensembles possède donc un problème majeur : on ne peut pas utiliser un axiome aussi fort que l'axiome de compréhension sans restriction. La solution proposée par l'axiomatique de ZF est alors la suivante : on va pouvoir construire des ensembles de plus en plus grands en utilisant des opérations telles que l'union ou l'ensemble des parties, et c'est seulement au sein d'un ensemble que l'on pourra utiliser l'axiome de compréhension (en supposant donc que la collection de tous les ensembles n'est pas elle-même un ensemble, pour ne pas retomber dans le paradoxe précédent).

5.2 Axiomatique de ZF

Nous utilisons donc le langage des ensembles. Nous allons présenter les axiomes un à un, sous leur forme formelle et en expliquant leur sens.

5.2.1 A propos de la Skolémisation

Nous utiliserons dans cette partie la skolémisation de nombreuses fois, qui est un principe très naturel : si l'on a une formule de la forme $\forall x_1 \dots \forall x_n . \exists y . P(x_1, \dots, x_n, y)$ alors on peut définir un symbole de fonction f correspondant, tel que $\forall x_1 \dots \forall x_n . P(x_1, \dots, x_n, f(x_1, \dots, x_n))$. C'est-à-dire que le choix de y dépend des quantificateurs universels qui le précèdent, donc on peut remplacer la propriété d'existence de y par une fonction de ces universels qui le précèdent. Un cas particulier est celui d'une proposition de la forme $\exists X . P(X)$: on pourra donner un nom spécial à un tel X (surtout si l'on sait qu'il est unique). Dans les axiomes qui suivront, justement, la propriété associée à l'objet dont on stipule l'existence assure son unicité, ce qui nous conforte dans l'introduction d'un symbole de fonction. Nous considérerons ici, étant donné notre cadre d'étude (la fonction de Skolem associée étant unique), que c'est un simple procédé de réécriture pour rendre lisible la théorie des ensembles.

5.2.2 Axiome d'extensionnalité

Axiome 5.2.1 (Extensionnalité)**.** L'axiome d'extensionnalité est le suivant :

$$\mathrm{ZF}_1 := \forall X . \forall Y . (\forall x . x \in X \iff x \in Y) \implies X = Y$$

et signifie qu'un ensemble est défini exactement par les éléments qui le constituent.

Exercice 5.2.1. Montrer le sens réciproque :

$$\vdash \forall X . \forall Y . X = Y \implies (\forall x . x \in X \iff x \in Y)$$

Le premier axiome sert à définir l'essence même d'un ensemble : c'est ce qui renforce l'égalité pour lui donner la forme qu'on accepte implicitement en théorie des ensembles : pour une collection d'objets donnée, il n'y a qu'un ensemble qui correspond à cette collection. Il nous montre déjà que les répétitions et l'ordre n'ont pas d'importance dans la définition d'un ensemble.

Sur l'inclusion

On peut déjà définir la relation d'inclusion, notée $X \subseteq Y$ par $\forall x . x \in X \implies x \in Y$.

Exercice 5.2.2. Montrer que $\subseteq$ est une relation d'ordre si l'on accepte l'extensionnalité, c'est-à-dire :

- $\mathrm{ZF}_1 \vdash X \subseteq X$
- $\mathrm{ZF}_1 \vdash (X \subseteq Y \wedge Y \subseteq X) \implies X = Y$
- $\mathrm{ZF}_1 \vdash (X \subseteq Y \wedge Y \subseteq Z) \implies X \subseteq Z$

On ajoutera donc à notre langage la relation $\subseteq$ comme façon plus simple d'écrire la proposition qui la définit.

5.2.3 Axiome de la paire

Axiome 5.2.2 (Paire)**.** L'axiome s'énonce comme suit :

$$\mathrm{ZF}_2 := \forall X.\forall Y.\exists Z.\forall x.x \in Z \iff (x = X \vee x = Y)$$

et signifie que si X et Y sont des ensembles, alors l'ensemble $\{X, Y\}$ contenant les éléments X et Y et aucun autre, est aussi un ensemble.

Remarque 5.2.1. En prenant $X = Y$ alors on construit le singleton $\{X\}$.

Exercice 5.2.3 (Sur les couples)**.** Pour x, y deux ensembles, on définit le couples (x, y) par $(x, y) = \{\{x\}, \{x, y\}\}$. Montrer que la propriété des couples est respectée : si $(x, y) = (x', y')$ alors $x = x'$ et $y = y'$, pour tous ensembles x, x', y, y'.

5.2.4 Axiome de l'union

Axiome 5.2.3 (Union)**.** L'axiome de la réunion est :

$$\mathrm{ZF}_3 := \forall X.\exists Z.\forall x.(x \in Z \iff \exists y.(x \in y \wedge y \in X))$$

et signifie que si l'on a un ensemble X, alors on peut construire l'ensemble Z (en général noté $\bigcup X$) qui est l'union des éléments de X, c'est-à-dire que ses éléments sont exactements les éléments d'éléments de X.

Exercice 5.2.4. Montrer que si on a un nombre fini d'ensembles, disons $(X_i)_{1 \leq i \leq n}$ alors $\{X_1, \ldots, X_n\}$ est un ensemble, et qu'il est l'unique ensemble contenant exactement chaque X_i, et rien d'autre. *Remarque : notre (X_i) est une collection au sens intuitif et non formel, nous supposons juste qu'il y a un nombre fini d'ensembles.*

Exercice 5.2.5. Montrer que si on a un nombre fini d'ensembles $(X_i)_{1 \leq i \leq n}$ alors $\bigcup_{i=1}^{n} X_i$ est aussi un ensemble. En particulier, pour deux ensembles, on notera $X \cup Y$ leur union.

5.2.5 Axiome de l'ensemble des parties

Axiome 5.2.4 (Ensemble des parties)**.** L'axiome de l'ensemble des parties est :

$$\mathrm{ZF}_4 := \forall X.\exists Z.\forall x.(x \in Z \iff x \subseteq X)$$

qui signifie que si X est un ensemble, alors la collection des ensembles inclus dans X forme aussi un ensemble. On notera en général $\mathcal{P}(X)$ l'ensemble des parties de X.

Exercice 5.2.6. Soit X, Y des ensembles, montrer qu'un couple (x, y) où $x \in X$ et $y \in Y$ est un élément de $\mathcal{P}(\mathcal{P}(X \cup Y)$.

5.2.6 Axiome de l'infini

L'idée de l'axiome de l'infini est de stipuler qu'il existe un ensemble infini. Pour cela, nous allons définir deux notions qui feront fortement penser à $\mathbb{N}$: un ensemble correspondant à 0 et une « fonction » correspondant au successeur.

Axiome 5.2.5 (Ensemble vide)**.** L'axiome est

$$\mathrm{ZF}_5 := \exists X.\forall x.\neg(x \in X)$$

et signifie qu'il existe un ensemble vide, c'est-à-dire un ensemble ne contenant aucun élément. On le note $\varnothing$.

Exercice 5.2.7 (Opération successeur)**.** Si X est un ensemble, montrer que l'ensemble $SX := X \cup \{X\}$ est aussi un ensemble. On appelle S la fonction successeur, qui à X associe SX (et on prendra la convention que $SSX = S(SX)$).

Calculer $SSS\varnothing$.

On peut alors définir l'axiome de l'infini :

Axiome 5.2.6 (Infini)**.** L'axiome se présente de la façon suivante :

$$\mathrm{ZF}_6 := \exists X.(\varnothing \in X \wedge (\forall x.x \in X \implies Sx \in X))$$

qui signifie qu'il existe un ensemble contenant $\varnothing$ et qui est stable par l'opération successeur.

Remarque 5.2.2. Ce que ne venons de définir n'est pas à proprement parler $\mathbb{N}$, car $\mathbb{N}$ est le plus petit ensemble contenant 0 et stable par successeur. Nous verrons que cet axiome nous permet, avec le suivant, de construire $\mathbb{N}$.

5.2.7 Schéma d'axiomes de remplacement

Plutôt que l'axiome de compréhension, la théorie ZF utilise un axiome plus fort : celui de remplacement. L'idée est qu'au lieu de simplement exprimer des ensembles par compréhension, on se permet de donner une image à ces axiomes. Pour cela, il nous faut d'abord définir la notion de proposition fonctionnelle.

Définition 5.2.1 (Proposition fonctionnelle)**.** *Une proposition P à $k+2$ variables libres est dite fonctionnelle, si, étant donnés $a_1, \ldots, a_k$ des paramètres fixés, on a*

$$\forall x.\forall y.\forall y'.((P(x, y, a_1, \ldots, a_k) \wedge P(x, y', a_1, \ldots, a_k)) \implies y = y')$$

On va noter $\mathrm{fonc}(P)$ *pour signifier que P est fonctionnelle (sous-entendu à $a_1, \ldots, a_k$ fixés).*

On peut donc voir P comme décrivant le graphe d'une fonction partielle : si x et y sont en relation par P alors cela signifie que y est l'image de x. Cela donne l'idée du schéma d'axiomes de remplacement (on a une infinité d'axiomes).

Axiome 5.2.7 (Remplacement)**.** Le schéma d'axiomes de remplacement est, pour P une proposition à $k+2$ variables libres :

$$\mathrm{ZF}_{7,P} := \forall a_1 \ldots \forall a_k.(\mathrm{fonc}(P) \implies \forall X.\exists Z.\forall y.(y \in Z \iff \exists x.(x \in X \wedge P(x, y, a_1, \ldots, a_k))))$$

Exercice 5.2.8. Une version plus faible du schéma d'axiome de remplacement est le schéma d'axiomes de compréhension : pour P une proposition à $k+1$ variables libres, le schéma est

$$\forall a_1 \dots \forall a_k . \forall X . \exists Z . \forall x . (x \in Z \iff (x \in X \wedge P(x, a_1, \dots, a_k))$$

Montrer que l'on peut déduire ce schéma d'axiomes du schéma d'axiomes de remplacement.

Nous utiliserons principalement le schéma d'axiomes de compréhension, et nous utiliserons pour cela la notation suivante :

$$\{x \in X \mid P(x)\}$$

pour signifier que l'on considère l'ensemble des éléments de X vérifiant la proposition $P(x)$. Si nous voulons utiliser le schéma d'axiomes de remplacement, nous utiliserons

$$\{y \mid x \in X, P(x, y)\}$$

voire $\{f(x) \mid x \in X\}$ si la proposition P définit de façon évidente une fonction f. Dans tous les cas, nous verrons que la plupart des fonctions que nous utiliserons pourrons être vues comme des ensembles, et nous aurons donc simplement à considérer le schéma d'axiomes de remplacement.

5.3 De la théorie ZF à notre pratique usuelle

On définit donc $\mathrm{ZF} = \mathrm{ZF}_{1,\dots,6} \cup \bigcup_{P \in \mathrm{Prop}} \mathrm{ZF}_{7,P}$ qui est donc l'ensemble des axiomes que nous avons introduit auparavant.

Exercice 5.3.1. Montrer que l'axiome de la paire est redondant, c'est-à-dire qu'il peut se montrer à l'aide des autres axiomes. De même, montrer que l'axiome de l'ensemble vide n'est pas utile si on considère qu'il existe au moins un ensemble.

5.3.1 A propos de l'intersection

Nous avons vu l'union d'un ensemble, mais nous pouvons aussi définir l'intersection d'un ensemble, pour peu que celui-ci soit non vide.

Définition 5.3.1 (Intersection)**.** *Soit X un ensemble non vide. On définit $\bigcap X$ par compréhension, en choisissant un élément $y \in X$ quelconque :*

$$\bigcap X := \{x \in y \mid \forall z \in X . x \in z\}$$

Exercice 5.3.2. Montrer que cette définition est satisfaisante au sens où l'ensemble défini ne dépend pas de l'élément $y \in X$ choisi.

Exercice 5.3.3. Soit une collection finie $(X_i)_{1 \leq i \leq n}$ d'ensembles, montrer que $\bigcap_{i=1}^{n} X_i$ est un ensemble bien défini. On notera $A \cap B$ pour l'intersection de deux ensembles.

5.3.2 Les entiers naturels

On va désormais construire les entiers naturels. On a vu qu'il existait un ensemble contenant $\varnothing$ (que l'on prendra pour notre 0) et qui était stable par successeur, mais il n'était pas $\mathbb{N}$ au sens où il était possiblement plus grand que $\mathbb{N}$. Nous allons donc définir $\mathbb{N}$ comme étant exactement l'ensemble des éléments qui sont dans toute partie contenant 0 et stable par successeur.

Définition 5.3.2 (Entiers naturels)**.** *On nomme X l'ensemble obtenu par l'axiome de l'infini, et on définit*

$$\mathbb{N} := \{x \in X \mid \forall F.(F \subseteq X \wedge \varnothing \in F \wedge (\forall z.z \in F \implies Sz \in F)) \implies x \in F\}$$

Un construction équivalente est de considérer $\mathcal{F} := \{F \in \mathcal{P}(X) \mid \varnothing \in F \wedge (\forall z.z \in F \implies Sz \in F)\}$ et de définir $\mathbb{N} := \bigcap \mathcal{F}$, qui existe bien puisque $X \in \mathcal{F}$.

Exercice 5.3.4. Montrer que ces deux définitions sont équivalentes, c'est-à-dire que les deux ensembles définis sont égaux.

On peut alors démontrer le théorème de récurrence :

Proposition 5.3.1 (Récurrence)**.** *Si $F \subseteq \mathbb{N}$ est telle que $\varnothing \in F$ et $\forall z.z \in F \implies Sz \in F$, alors $F = \mathbb{N}$.*

Démonstration. Il nous suffit de montrer l'inclusion $\mathbb{N} \subseteq F$, mais comme F est une partie de X (en prenant notre convention précédente) contenant $\varnothing$ et stable par successeur, elle contient tous les éléments de $\mathbb{N}$, donc $\mathbb{N} \subseteq F$, d'où l'égalité. □

Nous reviendrons plus en détail sur la récurrence lors de l'encodage de PA dans ZF.

5.3.3 Produit cartésien et relations

Nous avons vu qu'individuellement, un couple (a, b) appartenait bien à un plus gros ensemble, mais nous voulons que les ensembles de couples forment eux-mêmes un ensemble.

Définition 5.3.3 (Produit cartésien)**.** *Soient X et Y deux ensembles. On définit $X \times Y$, le produit cartésien de X et Y, par*

$$X \times Y := \{z \in \mathcal{P}(\mathcal{P}(X \cup Y)) \mid \exists x.\exists y.(x \in X \wedge y \in Y \wedge (z = \{x\} \vee z = \{x, y\}))\}$$

De par la définition, on notera plus directement $(x, y) \in X \times Y$.

On peut alors définir les relations binaires sur des ensembles :

Définition 5.3.4 (Relation binaire)**.** *On appelle relation binaire entre E et F une partie $\mathcal{R} \subseteq E \times F$. On note en général $x\mathcal{R}y$ pour signifier $(x, y) \in \mathcal{R}$.*

Définition 5.3.5 (Relation transposée, complémentaire, composition)**.** *Si $\mathcal{R}$ est une relation binaire, alors on note $\mathcal{R}^{-1}$ la relation transposée définie par $x\mathcal{R}^{-1}y \iff y\mathcal{R}x$ et on note $\overline{\mathcal{R}}$ la relation complémentaire de $\mathcal{R}$ définie par $x\overline{\mathcal{R}}y \iff \neg(x\mathcal{R}y)$.*

Si $\mathcal{R}$ est un relation binaire entre E et F, et $\mathcal{S}$ est une relation binaire entre F et G (E, F, G des ensembles) on note $\mathcal{S} \circ \mathcal{R}$ la relation définie par

$$x(\mathcal{S} \circ \mathcal{R})z \iff \exists y.y\mathcal{S}z \wedge x\mathcal{R}y$$

que l'on note composée de $\mathcal{R}$ et $\mathcal{S}$.

Exercice 5.3.5 (Associativité de la composition)**.** Montrer que la composition est associative, i.e. que si $\mathcal{R}$,$\mathcal{S}$ et $\mathcal{T}$ sont des relations binaires respectivement entre E et F, entre F et G et entre G et H, alors $(\mathcal{R} \circ \mathcal{S}) \circ \mathcal{T} = \mathcal{R} \circ (\mathcal{S} \circ \mathcal{T})$

Définition 5.3.6. *On note aussi* id_E *ou* Δ_E *la relation binaire entre* E *et* E *(on dit aussi « sur* E *») définie par* $x\Delta_E y \iff x = y$.

Exercice 5.3.6 (Vers la catégorie des relations binaires)**.** Montrer que pour toute relation $\mathcal{R}$ entre E et F, $\mathcal{R} \circ \mathrm{id}_E = \mathcal{R}$ et $\mathrm{id}_F \circ \mathcal{R} = \mathcal{R}$.

Traitons alors des relations fonctionnelles, qui nous permettront de définir ensuite les fonctions en elles-mêmes.

Définition 5.3.7 (Relation fonctionnelle)**.** *Soit* $\mathcal{R}$ *une relation entre* E *et* F*. On dit que* $\mathcal{R}$ *est fonctionnelle si la propriété suivante est vérifiée :*

$$\forall x.\forall y.\forall y'.(x\mathcal{R}y \wedge x\mathcal{R}y') \implies y = y'$$

On dit que $\mathcal{R}$ *est une relation fonctionnelle totale si, de plus, on a la propriété suivante :*

$$\forall x.\exists y.x\mathcal{R}y$$

Exercice 5.3.7. Montrer que la relation entre $X \times Y$ et X définie par $z\pi_1 x \iff \exists y.z = (x, y)$ est une relation fonctionnelle totale. De même on définit π_2 une relation fonctionnelle entre $X \times Y$ et Y.

Exercice 5.3.8 (Vers la catégorie des ensembles)**.** Montrer que si $\mathcal{R}$ et $\mathcal{S}$ sont des relations fonctionnelles, respectivement entre E et F et entre F et G, alors $\mathcal{S} \circ \mathcal{R}$ est aussi une relation fonctionnelle. Montrer que id_E est une relation fonctionnelle.

Une fonction est alors simplement une relation fonctionnelle pour laquelle on précise les deux ensembles entre lesquels est la relation :

Définition 5.3.8 (Fonction)**.** *Une fonction est un triplet* (E, F, Γ) *où* Γ *est une relation fonctionnelle totale entre* E *et* F *appelée le graphe de la fonction. Si* Γ *est simplement une relation fonctionnelle, on parle de fonction partielle. Ce triplet s'écrit en réalité* $((E, F), \Gamma)$ *puisque nous ne pouvons construire que des couples. On appelle* E *le domaine de* f*,* F *son codomaine.*

Remarque 5.3.1. Historiquement, le terme fonction désigne les fonctions partielles, et le terme d'application les fonctions totales. Mais dans le cadre de ce cours, nous utiliserons les noms de fonction et d'application de façon interchangeable.

Exercice 5.3.9. Montrer que l'ensemble F^E, aussi noté $\mathcal{F}(E, F)$, constitué des fonctions de E dans F, est un ensemble.

On notera en général $f : E \to F$ pour dire que $f \in F^E$, et au lieu d'écrire le graphe nous écrirons $x \mapsto y$ pour donner, à un x fixé, quel est l'unique y tel que $x\Gamma y$.

Exemple 5.3.1. Une définition de fonction peut être

$$\begin{array}{rccc} f & : & \mathbb{N} & \longrightarrow & \mathbb{N} \\ & & x & \longmapsto & x + x \end{array}$$

L'intérêt d'une fonction, c'est évident, est d'être appliqué à un argument pour donner une image. Cependant, ce que nous avons actuellement est purement un couple avec d'un côté deux ensembles, et de l'autre une relation fonctionnelle. Nous allons donc convenir maintenant d'une convention dans notre écriture de proposition : si f est une fonction, alors pour une proposition $P(x)$ on écrira $P(f(x))$ à la place de $\forall\Gamma.\forall y.f\pi_2\Gamma \wedge x\Gamma y \implies P(y)$. C'est-à-dire qu'on considérera directement que $f(x)$ est la deuxième projection du couple (x, y) du graphe de f dont la première projection est x : $f(x)$ est l'image de x.

Remarque 5.3.2. Tant qu'à parler de conventions d'usage, en mathématiques usuelles on se permet en général des quantifications typées, c'est-à-dire par exemple $\forall x \in \mathbb{R}$. Nous pouvons parfaitement traduire cela dans notre théorie du premier ordre (uniquement car nous sommes en théorie des ensembles) en remplaçant $\forall x \in E.P(x)$ par $\forall x.(x \in E \implies P(x))$ et $\exists x \in E.P(x)$ par $\exists x.(x \in E \wedge P(x))$. Nous utiliserons donc, dorénavant, des quantifications typées lorsque nous en avons l'envie (remarquons que, comme on peut définir une partie par une propriété grâce à l'axiome de compréhension, notre théorie du premier ordre code en fait une théorie d'ordre bien supérieur : on peut quantifier nos variables sur des propositions exprimées en tant que parties d'un ensemble).

On va rappeler ici les notions d'être injective, surjective et bijective pour une fonction.

Définition 5.3.9 (Injection, surjection, bijection)**.** *Une fonction $f : E \to F$ est*

- *une injection si* $\forall x \in E.\forall y \in E.f(x) = f(y) \implies x = y$
- *une surjection si* $\forall y \in F.\exists x \in E.f(x) = y$
- *une bijection si* $\forall y \in F.\exists x \in E.f(x) = y \wedge (\forall x' \in E.f(x) = y \implies x = x')$

Exercice 5.3.10. Montrer que

- s'il existe une fonction $g : F \to E$ telle que $g \circ f = \mathrm{id}_E$, alors f est injective.
- s'il existe une fonction $g : F \to E$ telle que $f \circ g = \mathrm{id}_F$, alors f est surjective.
- f est bijective si et seulement s'il existe une fonction $g : F \to E$ telle que $f \circ g = \mathrm{id}_F$ et $g \circ f = \mathrm{id}_E$.

5.3.4 Encoder PA

Comme nous avons utilisé la théorie PA dans la section précédente, montrons que l'on peut exhiber un modèle de PA dans ZF en utilisant $\mathbb{N}$. D'abord, l'interprétation de 0 est évidemment $\varnothing$ et l'interprétation de S est S.

Remarque 5.3.3. Dans notre codage des entiers, le nombre n est exactement l'ensemble $\{0, \dots, n-1\}$.

Nous allons maintenant considérer des lemmes nous permettant de construire des fonctions par récurrence :

Lemme 5.3.1 (Définition par récurrence)**.** *Soit X un ensemble, $x_0 \in X$ et $f : X \to X$. Alors il existe une unique fonction $g : \mathbb{N} \to X$ telle que $g(0) = x_0$ et $g(Sx) = f(g(x))$.*

Démonstration. L'important est de montrer que le graphe ainsi défini existe de façon unique et est fonctionnel et total. Montrons-le par récurrence sur la première projection :

- d'abord le graphe de g est défini pour tout couple de la forme $(0, y)$: seul $(0, x_0)$ appartient au graphe de g, c'est donc un graphe fonctionnel total sur $\{0\} \times X$ et il n'existe qu'un seul graphe possible respectant les propriétés énoncées plus haut.
- si le graphe de g (unique) est défini et fonctionnel total sur $\{0, \dots, n\} \times X$, alors (Sn, y) est dans le graphe de g si et seulement si $y = f(g(n))$. Or par hypothèse de récurrence $g(n)$ est bien défini et est unique, et comme f est une fonction, $f(g(n))$ est bien défini et est unique. Donc il existe un seul couple de la forme (Sn, y) dans le graphe de g sur $\{0, \dots, Sn\} \times X$.

Par récurrence, le graphe de $g : \mathbb{N} \to X$ existe, est unique et est fonctionnel et total. □

Un autre lemme qui sera important est appelé la curryfication (du nom de Haskell Curry) :

Lemme 5.3.2 (Curryfication). *Il y a une bijection entre $\mathcal{F}(X \times Y, Z)$ et $\mathcal{F}(X, Z^Y)$.*

Démonstration. Soit $f : X \times Y \to Z$, on définit

$$\begin{array}{rccc} \hat{f} & : & X & \longrightarrow & Z^Y \\ & & x & \longmapsto & (y \mapsto f(x,y)) \end{array}$$

et pour $g : X \to Z^Y$ on définit

$$\begin{array}{rccc} \overline{g} & : & X \times Y & \longrightarrow & Z \\ & & (x,y) & \longmapsto & g(x)(y) \end{array}$$

Soit $(x, y) \in X \times Y$, l'image de ce couple par f est la même que par $\overline{\hat{f}}$ donc f et $\overline{\hat{f}}$ coïncident, donc sont égales (même domaine, même codomaine et même graphe). Soit $x \in X$, son image par g est la fonction $y \mapsto g(x)(y)$ et son image par $\hat{\overline{g}}$ est $y \mapsto g(x)(y)$ donc les deux fonctions sont égales, donc g et $\hat{\overline{g}}$ coïncident sur X, donc sont égales.

On en déduit que nos deux transformations effectuent bien une bijection. □

Exercice 5.3.11. Montrer qu'il existe une bijection entre $X \times Y$ et $Y \times X$. En déduire qu'il y a une bijection entre $\mathcal{F}(X \times Y, Z)$ et $\mathcal{F}(Y, Z^X)$.

On peut alors définir l'addition et la multiplication :

Définition 5.3.10 (Addition). *On définit l'addition dans $\mathbb{N}$ par la fonction $f : \mathbb{N} \to \mathbb{N}^{\mathbb{N}}$ définie par récurrence de la façon suivante :*

- *l'image de 0 est $\mathrm{id}_{\mathbb{N}}$.*
- *la récurrence est définie par la fonction $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}, f \mapsto S \circ f$.*

En utilisant l'exercice précédent, cela nous donne une fonction $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ définie par :

- $(n, 0) \mapsto n$
- $(n, Sm) \mapsto S(n + m)$

Définition 5.3.11 (Multiplication). *En utilisant le même procédé, on définit la multiplication $\times : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ par :*

- $(n, 0) \mapsto 0$
- $(n, Sm) \mapsto (n \times m) + n$

Exercice 5.3.12. Vérifier alors que notre structure $(\mathbb{N}, +, \times)$ est bien un modèle de PA.

On peut alors définir, au-delà des couples, les n-uplets, ou tuples, et le produit d'ensembles indexé par une famille.

Définition 5.3.12 (Tuple). *Soient $E_0, \ldots, E_{n-1}$ des ensembles, l'ensemble $E_0 \times \ldots \times E_{n-1}$ (sans parenthèses) est l'ensemble*

$$\{f \in \mathcal{F}(n, \bigcup_{i=1}^{n} E_i) \mid \forall i \in n. f(i) \in X_i\}$$

Définition 5.3.13 (Produit quelconque d'ensembles)**.** *Soit I un ensemble non vide et $(X_i)_{i\in I}$ une famille d'ensembles indexée par i. On appelle le produit de ces ensembles, et on le note $\prod_{i\in I} X_i$ l'ensemble*

$$\prod_{i\in I} X_i = \{f \in \mathcal{F}(I, \bigcup_{i\in I} X_i) \mid f(i) \in X_i\}$$

On remarque que l'union existe car, par l'axiome de remplacement, l'ensemble des X_i existe.

A partir de maintenant, on supposera que l'on peut traduire (avec plus ou moins de difficulté) nos constructions usuelles en théorie des ensembles.

5.4 Ordinaux

Cette section va développer la théorie des ordinaux. Les ordinaux forment une classe (au sens de collection qui n'est pas un ensemble) centrale dans la théorie des ensembles, car elle constitue en quelque sorte une hiérarchie canonique des ensembles. On peut aussi les voir comme une extension de l'ensemble des entiers, en continuant d'applique l'opération de succession à $\mathbb{N}$, puis à son successeur, etc. Pour pouvoir nous définir les ordinaux, il va nous falloir d'abord définir deux notions : celle de bon ordre et celle d'ensemble transitif.

5.4.1 Définition et propriétés d'un ordinal

Nous avons vu, lors de l'étude des algèbre booléennes, la notion d'ensemble ordonné. Un bon ordre est un cas particulier d'ensemble ordonné.

Définition 5.4.1 (Bon ordre)**.** *Pour un ensemble ordonné $(X, \leq)$, on dit que $\leq$ est un bon ordre si pour toute partie non vide $F \subseteq X$, il existe un plu petit élément f. La version formelle de « $\leq$ est un bon ordre » est donc*

$$\forall F \subseteq X. F \neq \varnothing \implies \exists m \in F. \forall y \in F. m \leq y$$

On peut remarquer une première propriété :

Proposition 5.4.1. *Si $(X, \leq)$ est bien ordonné, alors $\leq$ est un ordre total, ce qui signifie que pour $x, y \in X$, on a $x \leq y \vee y \leq x$.*

Démonstration. Il suffit de considérer la partie $\{x, y\} \subseteq X$ qui possède un plus petit élément. □

Définition 5.4.2 (Ensemble transitif)**.** *Un ensemble X est dit transitif si pour tous x, y tels que $x \in y$ et $y \in X$, on a $x \in X$.*

Exercice 5.4.1. Montrer qu'être un ensemble X transitif est équivalent à la proposition

$$\forall Y. Y \in X \implies Y \subseteq X$$

Cela nous mène à la définition d'un ordinal :

Définition 5.4.3 (Ordinal)**.** *On dit qu'un ensemble X est un ordinal si X est transitif et $(X, \in)$ est un ensemble bien ordonné strict, c'est-à-dire que $\in$ est antiréflexive et transitive (relation d'ordre stricte), et que $\in'$, définie par $x \in y \vee x = y$, donne un bon ordre à X. On va noter le prédicat $\mathrm{On}(X)$ pour désigner le fait*

d'être un ordinal :

$$\mathrm{On}(X) := (\forall x \in X.x \notin x) \wedge (\forall x \in X.\forall y \in X.\forall z \in X.x \in y \wedge y \in z \implies x \in z) \\ \wedge (\forall Y.Y \in X \implies Y \subseteq X) \wedge (\forall Y \subseteq X.Y \neq \varnothing \implies \exists y \in Y.\forall y' \in Y.y = y' \vee y \in y')$$

Exemple 5.4.1. Les entiers comme nous les avons codés (appelés entiers de Von Neumann) sont des ordinaux. $\mathbb{N}$ est aussi un ordinal, que l'on nommera en général ω lorsque nous parlons d'ordinaux. $\omega \cup \{\omega\}$ est lui aussi un ordinal.

Contre-Exemple 5.4.1. Le prédicat On ne définit pas un ensemble : si $\mathrm{On} = \{X \mid \mathrm{On}(X)\}$ était un ensemble, alors $S\mathrm{On} = \mathrm{On} \cup \{\mathrm{On}\}$ serait un ordinal plus grand que On, mais par définition de On, $S\mathrm{On} \in \mathrm{On}$ donc par transitivité, $\mathrm{On} \in \mathrm{On}$, ce qui contredit l'hypothèse que $\in$ est une relation d'ordre stricte dans On. C'est donc une classe propre. Cependant comme le prédicat On est bien défini, on peut manipuler cette classe propre presque comme un ensemble.

On notera par α, β etc. des ordinaux.
Donnons alors plusieurs propriétés importantes des ordinaux :

Définition 5.4.4 (Segment initial)**.** *Soit α un ordinal. On appelle segment initial propre de α un ensemble de la forme*

$$S_\xi(\alpha) = \{\eta \in \alpha \mid \eta < \xi\}$$

Un segment initial de α est une partie S de α telle que si $x \in S$ et $y < x$ alors $y \in S$.

Proposition 5.4.2. *Un segment initial d'un ordinal α est soit α soit un élément de α.*

Démonstration. Soit un segment initial $S_\xi(\alpha)$, alors $S_\xi(\alpha) = \{\eta \in \alpha \mid \eta \in \xi\} = \xi \cap \alpha$ or $\xi \subseteq \alpha$ donc $S_\xi(\alpha) = \xi$. □

Proposition 5.4.3. *Si* $\mathrm{On}(X)$ *et $x \in X$ alors* $\mathrm{On}(x)$*.*

Démonstration. Comme X est transitif, les éléments de x sont aussi des éléments de X, donc $\in$ est un ordre stricte puisqu'induit par l'ordre strict de X. Si $Y \subseteq x$ alors $Y \subseteq X$ comme X est transitif, donc Y a un plus petit élément dans X, qui est aussi un plus petit élément dans x (puisque $x \subseteq X$). En spécialisant la transitivité de l'ordre $\in$ dans X on déduit que si $y \in x$ et $z \in y$ alors $z \in x$, ce qui signifie que x est transitif. □

Proposition 5.4.4. *Pour tout ordinal α, $\alpha \notin \alpha$.*

Démonstration. Cette proposition est vraie par définition d'un ordinal (puisque $\in$ est antiréflexif dans α). □

Proposition 5.4.5. *Si α et β sont des ordinaux, alors soit $\alpha = \beta$, soit $\alpha \in \beta$, soit $\beta \in \alpha$ (et ces cas sont exclusifs deux à deux).*

Démonstration. On considère $\xi = \alpha \cap \beta$. ξ est un segment initial de α puisque si $x \in \xi$ et $y \in x$ alors $y \in \alpha$ et $y \in \beta$ donc $y \in \xi$. De même ξ est un segment initial de β, d'où $(\xi = \alpha \vee \xi \in \alpha) \wedge (\xi = \beta \vee \xi \in \beta)$, ce qui nous donne quatre cas :

— si $\xi = \alpha$ et $\xi = \beta$, alors $\alpha = \beta$.
— si $\xi \in \alpha$ et $\xi = \beta$ alors $\beta \in \alpha$.

— si $\xi = \alpha$ et $\xi \in \beta$ alors $\alpha \in \beta$.
— si $\xi \in \alpha$ et $\xi \in \beta$ alors on en déduit que $\xi \in \xi$ puisque $\xi \in \alpha \cap \beta$, ce qui est une contradiction puisque ξ est un ordinal.

□

On en déduit la proposition suivante :

Proposition 5.4.6. *La relation $\in$ est une relation de bon ordre stricte sur* On *et la relation d'ordre associée est $\subseteq$.*

Exercice 5.4.2. Prouver cette proposition.

Définition 5.4.5 (Successeur)**.** *Si α est ordinal, alors il existe un plus petit ordinal contenant α, appelé successeur de α (et noté $S\alpha$ ou $\alpha + 1$) qui est $\alpha \cup \{\alpha\}$.*

Démonstration. $\alpha + 1$ est bien un ordinal (cela se vérifie facilement) et tout ordinal γ strictement supérieur à α contient les éléments de α, donc $\alpha \subseteq \gamma$ et contient α, donc $\alpha \in \gamma$, donc $\alpha \cup \{\alpha\} \subseteq \gamma$. □

Proposition 5.4.7 (Borne supérieure)**.** *Soit O un ensemble d'ordinaux, alors O a une borne supérieure qui est $\bigcup O$.*

Démonstration. On définit $\beta = \bigcup O$. Si x est une partie non vide de β, alors $x \cap \alpha_0 \neq \varnothing$ pour un certain $\alpha_0 \in O$. Dans ce cas, $x \cap \alpha_0$ a un plus petit élément puisque c'est une partie de α_0, et cet élément est le plus petit élément de x (puisque pour tout autre ordinal $\alpha \in O$, $x \cap \alpha$ contient $x \cap \alpha_0$ ou est contenu dans celui-ci, mais quoi qu'il arrive l'élément minimal de $x \cap \alpha_0$ est bien dans $x \cap \alpha$). Donc β est bien ordonné par $\in$. Si $x \in \beta$ et $y \in x$ alors $x \in \alpha$ pour un certain $\alpha \in O$ donc $y \in \alpha$, donc $y \in \beta$. On en déduit donc que β est bien un ordinal. Le fait que β soit la borne supérieure de O est direct : il contient tout élément de O donc $\forall \alpha \in O, \alpha \leq \beta$ et si un ordinal γ contient tous les éléments de O alors il contient aussi leur union, donc $\beta \leq \gamma$. □

Lien avec les bons ordres

Il se trouve que les ordinaux peuvent être vus aussi comme une forme canonique d'ensemble bien ordonné, au sens où tout ensemble bien ordonné est isomorphe à un unique ordinal (un isomorphisme d'ensembles ordonnés est une bijection croissante de réciproque croissante, c'est-à-dire telle que pour tous éléments a et b, $a \leq b \implies f(a) \leq f(b)$). On peut interpréter ce résultat comme le fait que tout ensemble bien ordonné est ordonné par $\in$ dans un ordinal, en renommant ensuite les éléments. Nous allons prouver ce théorème en plusieurs étapes.

Lemme 5.4.1. *Soient α et β deux ordinaux, et $f : \alpha \to \beta$ une application strictement croissante. Alors $\alpha \leq \beta$ et $\xi \leq f(\xi)$ pour tout $\xi \in \alpha$*

Démonstration. Soit ξ le plus petit élément de α tel que $f(\xi) < \xi$ s'il en existe. Comme f est strictement croissante, on a $f(f(\xi)) < f(\xi)$. Donc, en posant $\eta = f(\xi)$, on a $\eta < \xi$ et $f(\eta) < \eta$, ce qui contredit la définition de ξ. On a donc $\xi \leq f(\xi)$ pour tout $\xi \in \alpha$, donc $\xi \in \beta$ puisque $f(\xi) \in \beta$, donc $\alpha \subseteq \beta$, donc $\alpha \leq \beta$. □

Proposition 5.4.8. *Soient α et β deux ordinaux, et f un isomorphisme d'ensembles ordonnés de α sur β. Alors $\alpha = \beta$ et $f = \mathrm{id}_\alpha$.*

Démonstration. D'après le lemme précédent, $\alpha \leq \beta$ (puisqu'un isomorphisme d'ensembles ordonnés est par extension strictement croissant) et $\xi \leq f(\xi)$ pour tout $\xi \in \alpha$. De plus, f^{-1} est un isomorphisme d'ensembles ordonnés donc $\beta \leq \alpha$, donc $\alpha = \beta$, et pour tout $\xi \in \beta$, $\xi \leq f(\xi)$. On en déduit que $f(\xi) = x$ pour tout $\xi \in \alpha$ et donc que $f = \mathrm{id}_\alpha$. □

Théorème 5.4.1. *Pour chaque ensemble bien ordonné $(E, \leq)$, il existe un isomorphisme et un seul de $(E, \leq)$ vers un ordinal.*

Démonstration. Montrons d'abord l'unicité : si f est un isomorphisme de $(E, \leq)$ vers un ordinal α, et g un isomorphisme de $(E, \leq)$ vers β, alors en composant g^{-1} et f on déduit que $\alpha = \beta$ et que $g = f$.

Montrons maintenant l'existence : on pose

$$B = \{x \in E \mid S_x(E) \text{ est isomorphe à un ordinal}\}$$

et on note $\beta(x)$ l'ordinal isomorphe à x, pour $x \in B$.

Si $y \in B$ et $x < y$ alors $x \in B$ et $\beta(x) < \beta(y)$ car dans l'isomorphisme de $S_y(E)$ sur $\beta(y)$, le segment initial $S_x(E)$ de $S_y(E)$ devient un segment initial strict de $\beta(y)$, c'est-à-dire un ordinal $\beta(x) < \beta(y)$. D'après le schéma de substitution, l'ensemble des $\beta(x)$ pour $x \in B$ est un segment initial de On, c'est donc un ordinal δ et l'application $x \mapsto \beta(x)$ est un isomorphisme entre B et δ.

Si $B \neq E$, alors $B = S_{x_0}(E)$ pour $x_0 \in E$ puisque b est un segment initial de E. Mais cela signifie que $x_0 \in B$ donc $x_0 \in S_{x_0}(E)$ ce qui contredit la définition de $S_{x_0}(E)$. Donc $B = E$ et on a un isomorphisme de E dans β. □

5.4.2 Induction transfinie

L'un des points les plus importants avec les ordinaux est la possibilité de généraliser la récurrence forte à la fois pour prouver des propositions sur tous les ordinaux et pour construire des fonctions dont le domaine est On (remarquons que ces « fonctions » sont plus précisément des relations fonctionnelles, mais qui restreintes à des ensembles donnent des fonctions au sens de notre théorie). Nous verrons donc le théorème de construction par induction transfinie, puis quelques exemples de constructions transfinies (les opérations arithmétiques transfinies, ainsi que la hiérarchie cumulative de Von Neumann).

Tout d'abord, donnons le principe de preuve par induction sur les ordinaux.

Proposition 5.4.9 (Induction sur les ordinaux)**.** *Soit $E(x)$ un prédicat à une variable libre, alors*

$$(\forall x.\mathrm{On}(x) \implies E(x)) \iff (\forall \alpha.\mathrm{On}(\alpha) \implies [(\forall \beta.(\beta < \alpha) \implies E(\beta)) \implies E(\alpha)])$$

Démonstration. Un sens est évident (si E est vrai sur tous les ordinaux, a alors la proposition de droite est vraie). Dans l'autre sens, si E n'est pas vraie pour tous les ordinaux, on considère le plus petit ordinal tel que $E(\alpha)$ est fausse : on en déduit que $E(\alpha)$ est vraie puisque E est vraie sur tous les éléments strictement plus petits que α. □

Remarque 5.4.1. Dans la proposition de droite, on remarque que comme $\beta < \varnothing$ n'est jamais vrai, $E(\varnothing)$ est vrai.

Pour pouvoir définir de façon efficace nos objets, nous allons introduire une nouvelle notation : pour une relation fonctionnelle F, $F \restriction C$ signifie la restriction de F à C, c'est-à-dire la relation $F(x) \wedge x \in C$ pour un objet x. Nous confondrons les relations fonctionnelles et les fonctions induites : la relation fonctionnelle $F \restriction C$ correspond à une fonction de C dans l'union des $F(x)$ pour $x \in C$, si C est un ensemble.

Définition 5.4.6 (Relation H-inductive)**.** *Soit une relation fonctionnelle H. Une fonction f est H-inductive si elle a pour domaine un ordinal α et si pour tout ordinal $\beta \in \alpha$, $f \restriction \beta$ est dans le domaine de H (c'est-à-dire qu'il existe un élément y tel que $H(\beta, y)$ est vraie), $H(f \restriction \beta, f(\beta))$.*

Proposition 5.4.10. *Il existe au plus une relation H-inductive de domaine α.*

Démonstration. Supposons que f et g soient deux fonctions H-inductives de domaine α. On considère alors β le plus petit ordinal tel que $f(\beta) \neq g(\beta)$ (s'il n'existe pas, alors le résultat est prouvé). Alors pour tout $\gamma < \beta$, on en déduit que $f(\gamma) = g(\gamma)$, donc $f \restriction \beta = g \restriction \beta$, donc $f(\beta) = g(\beta)$ puisque $H(f \restriction \beta, f(\beta))$ et $H(g \restriction \beta, g(\beta))$ et H est fonctionnelle, ce qui est une contradiction. □

On en déduit que si f est H-inductive de domaine α et $\beta \in \alpha$, alors $f \restriction \beta$ est H-inductive de domaine β.

Théorème 5.4.2 (Fonction H-inductive)**.** *Soit H une relation fonctionnelle, et α un ordinal tel que toute fonction H-inductive de domaine $\beta < \alpha$ soit dans le domaine de H. Alors il existe une fonction H-inductive et une seule de domaine α.*

Démonstration. L'unicité vient directement de la proposition précédente.

Soit τ l'ensemble des $\beta < \alpha$ tels qu'il existe une fonction H-inductive f_β de domaine β. τ est un segment initial et d'après l'unicité, si $\gamma < \beta$ alors $f_\gamma = f_\beta \restriction \gamma$. τ est donc un ordinal $\tau \leq \alpha$, puisque c'est un segment initial d'ordinaux. On peut donc définir f_τ par $f_\tau(\beta) = H(f_\beta)$ (en notant $H(f_\beta)$ l'unique y tel que $H(f_\beta, y)$) pour $\beta \in \tau$. Si $\gamma < \beta < \tau$, alors $f_\tau(\gamma) = H(f_\gamma) = H(f_\beta \restriction \gamma) = f_\beta(\gamma)$, donc $f_\beta = f_\tau \restriction \beta$ donc $f_\tau(\beta) = H(f_\tau \restriction \beta)$, donc f_τ est une fonction H-inductive de domaine τ. Si $\tau < \alpha$, la définition de τ montre que $\tau \in \tau$ ce qui est impossible, donc $\tau = \alpha$ et f_τ est la fonction recherchée. □

On peut même étendre ce résultat en considérant la version suivante :

Corollaire 5.4.1. *Soient α un ordinal, A une collection, M la collection des applications définies sur $\beta < \alpha$ et à valeurs dans A et H une relation fonctionnelle de domaine M à valeurs dans A. Alors il existe une fonction f et une seule, définie sur α, telle que $f(\beta) = H(f \restriction \beta)$ pour tout $\beta < \alpha$.*

On peut donc définir une fonction (voire une fonction à deux variables avec le corollaire) par induction transfinie, c'est-à-dire par induction sur les ordinaux, jusqu'à un ordinal α donné. Nous allons maintenant généraliser cette construction pour pouvoir générer des relations fonctionnelles dont le domaine est On et telles que leur restriction à un ordinal est une fonction.

Théorème 5.4.3. *Soit H une relation fonctionnelle, telle que toute fonction H-inductive soit dans le domaine de H. On peut alors définir une relation fonctionnelle F, de domaine* On*, telle que $F(\alpha) = H(f \restriction \alpha)$ pour tout ordinal α. C'est la seule relation fonctionnelle ayant ces propriétés et de plus $F \restriction \alpha$ est une fonction H-inductive pour tout ordinal α.*

Démonstration. On définit F simplement par $y = F(\alpha)$ si et seulement si « il existe une fonction H-inductive f_α de domaine α et $y = H(f_\alpha)$.

D'après le théorème précédent, il existe une telle fonction f_α et une seule, donc F est bien une relation fonctionnelle de domaine On. De plus, si $\beta < \alpha$, on a $f_\beta = H(f_\beta) = H(f_\alpha \restriction \beta) = f_\alpha(\beta)$ pour tout $\beta < \alpha$. Donc $f_\alpha = F \restriction \alpha$ et $F \restriction \alpha$ est H-inductive. Comme $F(\alpha) = H(f_\alpha)$, on a bien $F(\alpha) = H(F \restriction \alpha)$.

Pour l'unicité, l'argument est toujours le même : regarder le plus petit ordinal où deux relations fonctionnelles F et G diffèrent et déduire qu'elles sont égales en cet ordinal puisqu'égales sur les ordinaux plus petits (strictement). □

On peut encore donner une version comme celle du corollaire 5.4.1 :

Corollaire 5.4.2. *Soient A une collection, M la collection des applications définies sur les ordinaux, à valeurs dans A, H une relation fonctionnelle de domaine M, à valeurs dans A. On peut alors définir une relation fonctionnelle F de domaine* On *à valeurs dans A telle que $F(\alpha) = H(F \restriction \alpha)$ pour tout ordinal α. C'est la seule telle relation fonctionnelle.*

5.4.3 Opérations ordinale

Nous allons utiliser le théorème de définition des relations fonctionnelles par induction transfinie pour construire les opérations + et × sur les ordinaux. Pour cela, nous allons d'abord avoir besoin du lemme suivant :

Lemme 5.4.2 (Disjonction de cas sur les ordinaux)**.** *Soit α un ordinal. Alors α est de l'une des formes suivantes :*

- $\alpha = 0$
- *il existe β tel que $\alpha = S\beta$*
- *α n'est ni l'un ni l'autre, et est alors appelé ordinal limite, auquel cas $\alpha = \bigcup_{\beta<\alpha} \beta$.*

Démonstration. Le fait que ces cas sont exclusifs est évident. Il nous suffit donc de prouver que si α est non nul et n'est pas un successeur, alors $\alpha = \bigcup_{\beta<\alpha} \beta$.

α contient tous les éléments de $\bigcup_{\beta<\alpha} \beta$, il nous suffit donc de montrer l'autre inégalité. Si $\bigcup_{\beta<\alpha} \in \alpha$, alors α contient tous les éléments de $\bigcup_{\beta<\alpha} \beta$ et $\bigcup_{\beta<\alpha} \beta$, donc $\alpha = \bigcup_{\beta<\alpha} \beta \cup \{\bigcup_{\beta<\alpha} \beta\} = S(\bigcup_{\beta<\alpha} \beta)$ (α ne peut pas être strictement supérieur à ce successeur car sinon l'union ne serait pas la borne supérieure des éléments de α) ce qui contredit l'hypothèse que α n'est pas successeur. □

On peut donc facilement définir une relation fonctionnelle sur les ordinaux par disjonction de cas sur ces ordinaux (donner une définition différente pour 0, les successeurs et les ordinaux limites). On définit alors les opérations de la façon suivante :

Définition 5.4.7 (Addition ordinale)**.** *Soient α et β des ordinaux, on définit de la façon suivante l'ordinal $\alpha + \beta$:*

- $\alpha + 0 = \alpha$
- $\alpha + S\beta = S(\alpha + \beta)$
- $\alpha + \lambda = \bigcup_{\beta<\lambda} (\alpha + \beta)$ *si λ est un ordinal limite.*

On remarque alors que par définition, $S(\alpha) = \alpha + 1$.

Exercice 5.4.3. Soit $(E_i)_{i\in I}$ une famille d'ensembles indicée par I. On définit

$$\sum_{i\in I} E_i$$

comme l'ensemble des couples (i, x) où $x \in E_i$.

Soit α et β deux ordinaux. Montrer qu'en considérant l'ordre lexicographique sur $\sum_{i\in\beta} \alpha$ (donc une somme sur une famille constante d'ensembles) on trouve un ensemble bien ordonné. Montrer que cet ensemble est isomorphe à $\alpha + \beta$.

Remarque 5.4.2. L'addition n'est pas commutative avec des ordinaux infinis : $1 + \omega = \bigcup 1 + n = \omega$ et $\omega + 1 = S(\omega) = \mathbb{N} \cup \{\mathbb{N}\}$.

Définition 5.4.8 (Multiplication ordinale)**.** *On définit la multiplication $\alpha \times \beta$ de la façon suivante :*

- $\alpha \times 0 = 0$
- $\alpha \times S\beta = \alpha \times \beta + \alpha$

- $\alpha \times \lambda = \bigcup_{\beta<\lambda} (\alpha \times \beta)$ *si* λ *est un ordinal limite.*

Exercice 5.4.4. Montrer que l'ordre lexicographique sur $\alpha \times \beta$ est un bon ordre. En déduire que $\alpha \times \beta$ en tant qu'ensemble, avec l'ordre lexicographique, est isomorphe à l'ordinal $\alpha \times \beta$.

Hiérarchie cumulative de Von Neumann

On définit la classe V des ensembles de Von Neumann de la façon suivante :

- $V_0 = \varnothing$
- $V_{S\alpha} = \mathcal{P}(V_\alpha)$
- si λ est un ordinal limite, alors $V_\lambda = \bigcup_{\beta<\lambda} V_\beta$.

La classe V est alors l'union des V_α pour $\alpha \in \mathrm{On}$ (c'est donc une classe propre, ou collection, et pas un ensemble). On pourrait par exemple écrire $V(x)$ par

$$\exists \alpha.\mathrm{On}(\alpha) \wedge x \in V_\alpha$$

Remarque 5.4.3. L'univers de Von Neumann V est ce qu'on appelle un modèle interne de la théorie des ensembles : c'est une classe qui est stable par les axiomes de ZF (et l'axiome du choix). Il se trouve que les mathématiques usuelles peuvent s'exprimer en ne considérant que V et pas d'ensembles extérieurs. De plus, V vérifie aussi l'axiome appelé axiome de fondation, qui signifie entre autre qu'il n'existe pas d'ensemble a tel que $a \in a$. Il se trouve que l'axiome de fondation est équivalent à l'énoncé « tout ensemble est dans la classe V ».

5.5 Axiome du choix

Nous allons maintenant nous intéresser plus précisément à l'axiome du choix, qui est utilisé en mathématique avec parcimonie pour des questions de constructibilité. Nous allons simplement donner les énoncés équivalents de l'axiome du choix, donc le lemme de Zermelo et le lemme de Zorn, et les raisons poussant à ne pas toujours le considérer. L'axiome du choix, comme son nom l'indique, permet d'effectuer des choix. Cela signifie, étant donnée une collection d'ensembles non vides (donc un ensemble arbitrairement grand d'ensembles non vides), de prendre un élément de chacun de ces ensembles non vides. Dans le cas fini, l'axiome du choix est vrai, mais dans le cas infini cet axiome est nécessaire.

Axiome 5.5.1 (Choix)**.** On a trois énoncés équivalents de l'axiome du choix :

- Pour chaque ensemble a dont les éléments sont non vides et disjoints deux à deux, il existe un ensemble dont l'intersection avec chaque élément de a est un ensemble à un seul élément.
- Pour tout ensemble a, il existe une application h de l'ensemble des parties non vides de a dans a telle que $f(x) \in x$ pour toute partie x non vide de a.
- Le produit d'une famille d'ensembles non vides est non vide.

Démonstration. Montrons que la deuxième définition implique la première : on appelle b la réunion des éléments de a. En appliquant l'axiome du choix dans sa deuxième formulation à b, comme chaque élément x de a est une partie non vide de b, l'ensemble $\{h(x) \mid x \in a\}$ a exactement un élément en commun avec chaque élément x de a.

La première définition implique la troisième : on définit $(a_i)_{i\in I}$ une famille d'ensembles non vides. Soit $b_i = \{i\} \times a_i$. Alors $(b_i)_{i\in I}$ est formée d'ensembles non vides et disjoints deux à deux. Soit ξ un ensemble tel que $\xi \cap b_i$ ait un seul élément pour tout $i \in I$, alors $(\xi \cap \bigcup_{i\in I} b_i) \in \prod_{i\in I} a_i$, par définition de $\prod_{i\in I} a_i$.

La troisième définition implique la deuxième : si a est un ensemble quelconque, on forme le produit

$$\prod_{x\subseteq a, x\neq\varnothing} x$$

d'après la troisième formulation de l'axiome du choix cet ensemble est non vide. Pour un élément φ de cet ensemble, $\varphi(x) \in x$ pour toute partie x non vide de a. □

Une formulation assez utile est le théorème de Zermelo, qui dit que tout ensemble peut être muni d'un bon ordre. Ceci signifie en particulier que tout ensemble est isomorphe à un ordinal en prenant ce bon ordre.

Théorème 5.5.1 (Zermelo)**.** *Il existe un bon ordre sur tout ensemble.*

Démonstration. Cet énoncé implique la deuxième formulation de l'axiome du choix : on considère un bon ordre $\leq$ sur l'ensemble E et on définit $h : \mathcal{P}(E) \setminus \{\varnothing\} \to E$ par $h(x) = \min(x)$ où le minimum est défini par le fait que $\leq$ est un bon ordre.

Réciproquement, soit E un ensemble, on suppose que la deuxème formulation de l'axiome du choix est vraie et donc qu'on a $h : \mathcal{P}(E) \setminus \{\varnothing\} \to E$ où $h(x) \in x$. Supposons qu'il n'existe pas de bon ordre sur E. On définit une relation fonctionnelle H, dont le domaine est la collection des fonctions f dont l'image ne contient pas a. Cette relation est définie en posant $H(f) = h(a \setminus \mathrm{Im}(f))$.

Soit f une fonction H-inductive, dont le domaine est un ordinal α. Pour $\beta < \alpha$, on a $f(\beta) = H(f \restriction \beta) \in a \setminus \mathrm{Im}(f \restriction \beta)$, donc $f(\beta) \notin \mathrm{Im}(f \restriction \beta)$. Par suite, si $\gamma < \beta$, on a $f(\beta) \neq f(\gamma)$, ce qui montre que f est une injection de α dans E. Si f n'est pas dans le domaine de H, on a f $\supseteq E$. Par suite, f est une bijection de l'ordinal α vers E. Il en résulte que E peut être bien ordonné, ce qui est faux par hypothèse.

Donc toute fonction H-inductive est dans le domaine de H. On a donc une relation fonctionnelle $y = F(\alpha)$ de domaine On telle que $F(\alpha) = H(F \restriction \alpha)$ et $F \restriction \alpha$ est H-inductive pour tout ordinal α. Par suite, $G \restriction \alpha$ est une injection de α dans E et donc F est une relation fonctionnelle injective de domaine On, à valeurs dans E. Ceci est clairement impossible, donc il existe un bon ordre sur E. □

Remarque 5.5.1. Cette preuve est un peu plus forte que l'équivalence entre l'axiome du choix et le théorème de Zermelo : elle montre que pour un ensemble E donné, avoir un bon ordre est équivalent à avoir une fonction $h : \mathcal{P}(E) \setminus \{\varnothing\} \to E$ telle que $h(x) \in x$.

Le théorème de Zorn découle aussi de l'axiome du choix :

Théorème 5.5.2 (Zorn)**.** *Soit E un ensemble ordonné dont toute partie bien ordonnée est majorée. Alors E possède un élément maximal (i.e. un élément x tel que $\forall y \in E, \neg(x < y)$).*

Démonstration. Soit $(E, \leq)$ un ensemble ordonné, on suppose que l'on dispose d'une fonction $h : \mathcal{P}(E) \setminus \{\varnothing\} \to E$ telle que $h(x) \in x$. Soit c l'ensemble des parties de a qui ont un majorant strict. On définit une application $m : c \to E$ en posant

$$m(x) = h(\text{l'ensemble des majorants stricts de } x)$$

donc pour tout $x \in c$, $m(x)$ est un majorant strict de x.

On définit une relation fonctionnelle H dont le domaine est la collection des fonctions f telles que $\mathrm{Im}(f) \in c$, en posant $H(f) = m(\mathrm{Im}(f))$.

Soit f une fonction H-inductive, dont le domaine est un ordinal α. Pour $\beta < \alpha$, on a $f(\beta) = H(f \restriction \beta)$ qui est un majorant strict de $\mathrm{Im}(f \restriction \beta)$. Si $\gamma < \beta$, on a $f(\gamma) < f(\beta)$, dont f est une fonction strictement croissante de α dans E. L'image de f est donc une partie bien ordonnée de E, et a donc un majorant $\mu \in E$.

Si f n'est pas dans le domaine de H, on a $\mathrm{Im}(f) \notin c$. Alors $\mathrm{Im}(f)$ n'a pas de majorant strict. Il en résulte que μ n'a pas de majorant strict, c'est-à-dire est maximal, ce qui donne le résultat cherché.

Si on suppose que toute fonction H-inductive est dans le domaine de H, alors on a une relation fonctionnelle $y = F(\alpha)$ de domaine On telle que $F(\alpha) = H(F \restriction \alpha)$ et $F \restriction \alpha$ est H-inductive pour tout ordinal α. On a donc une relation fonctionnelle strictement croissante (donc injective) de On dans E, ce qui est impossible. □

Le sens réciproque se démontre à partir d'une version affaiblie du théorème de Zorn.

Proposition 5.5.1. *Si tout ensemble ordonné qui a la propriété que tout sous-ensemble totalement ordonné possède une borne supérieure, a un élément maximal, alors l'axiome du choix est vrai.*

Démonstration. Soit E un ensemble dont tous les éléments sont non vides et disjoints deux à deux. Soit U la réunion des éléments de E et X l'ensemble des parties de U qui rencontrent chaque élément x de E en un élément au plus. X est ordonné par l'inclusion. Soit Y un sous-ensemble totalement ordonné de X, alors $\bigcup_{y\in Y} y \in X$. L'ensemble X a la propriété exigée, et on en déduit donc un élément $y_0 \in X$ maximal. Alors $x \cap y_0$ est un ensemble a un élément (s'il ne possède pas d'élément, on peut ajouter un élément à y_0 ce qui contredit la maximalité) pour tout élément x de E. □

Accepter l'axiome du choix revient donc à accepter (un ou) toutes les propositions ci-dessus. Le problème principal de cet axiome est que les raisonnements qu'il engendre sont dits non constructifs. Cela signifie que l'on a l'existence d'objets qu'on ne peut pas forcément exhiber. Par exemple il devient possible de montrer que l'ensemble $\mathbb{R}/\mathbb{Q}$ possède un système de représentant (ce qui revient à dire que $\mathbb{R}$ en tant que $\mathbb{Q}$-ev a une base), mais il est impossible de construire un tel système de représentants.

5.6 Cardinal

Nous allons finir ce chapitre en mentionnant les cardinaux, qui sont une façon de mesurer la taille d'ensembles. Nous utiliserons l'axiome du choix dans cette partie.

L'idée première pour définir un cardinal est de regarder les ensembles modulo la relation d'équipotence. Ceci signifie qu'on considère une relation $E \equiv F$ si et seulement s'il existe une bijection entre E et F. Le problème de cette construction est qu'il est impossible de construire un ensemble quotient ainsi, puisqu'il n'y a pas d'ensemble de tous les ensembles. Cependant, on peut souhaiter chercher un représentant de chaque classe d'équivalence, en considérant pour un ensemble E un ensemble canonique qui lui serait associé. C'est avec cette stratégie que l'on définit un cardinal, en prenant comme famille d'ensembles canoniques les ordinaux, où l'on considère le plus petit ordinal possible pour une classe d'équipotence donnée.

Définition 5.6.1 (Cardinal)**.** *On appelle cardinal un ordinal κ tel que pour tout ordinal $\alpha < \kappa$, α n'est pas en bijection avec κ (c'est donc le plus petit ordinal de se classe d'équipotence).*

On dit qu'un ensemble E est de cardinal κ s'il est en bijection avec κ. Par le théorème de Zermelo pour tout ensemble E il existe un cardinal, noté $\mathrm{card}(E)$*, associé à l'ensemble E.*

Remarque 5.6.1. A priori, le théorème de Zermelo n'empêche pas plusieurs bons ordres d'être donnés à un ensemble E donné, et alors E pourrait alors être isomorphe à deux ordinaux différent suivant l'ordre associé. Cependant, ceux deux ordinaux sont en bijection entre eux, et sont donc associés au même cardinal.

On note Cn la classe (propre) des cardinaux.

Proposition 5.6.1. *Soient E et F deux ensembles non vides. Les conditions suivantes sont équivalentes :*

- *il existe une injection $i : E \to F$.*
- *il existe une surjection $s : F \to E$.*
- $\mathrm{card}(E) \leq \mathrm{card}(F)$.

Démonstration. S'il existe une injection $i : E \to F$ alors on construit une surjection $s : F \to E$ en prenant un élément $x_0 \in E$ et en envoyant chaque antécédent d'un élément x par i sur x, et chaque élément de F sans antécédent sur x_0.

S'il existe une surjection $s : F \to E$, alors on construit une injection $i : E \to F$ en utilisant l'axiome du choix pour prendre un représentant de chaque ensemble $s^{-1}(\{y\})$ et en envoyant y dessus.

Si $\text{card}(E) \leq \text{card}(F)$ alors on a deux bijections $f : E \to \text{card}(E)$ et $g : F \to \text{card}(F)$ ainsi qu'une injection $i : \text{card}(E) \to \text{card}(F)$ définie comme l'identité sur $\text{card}(E)$. En composant ces applications, on en déduit une injection de E dans F.

Soit j une injection de E dans F, alors $g \circ j$ est une injection de E dans $\text{card}(E)$. L'image de cette fonction est alors un ensemble bien ordonné $u \subseteq \text{card}(E)$: on a donc un isomorphisme $\phi : \beta \to u$. Ceci signifie que $\beta \leq \text{card}(E)$. Comme E est équipotent à u et donc à β, on en déduit que $\text{card}(E) \leq \beta$, donc $\text{card}(E) \leq \text{card}(E)$. □

Le théorème suivant est un théorème classique de théorie des ensembles :

Théorème 5.6.1 (Cantor-Bernstein)**.** *Soient E et F deux ensemble tels qu'il existe une injection de E dans F et une injection de F dans E. Alors il existe une bijection entre E et F.*

Démonstration. Dans le cas actuel, la preuve découle directement du résultat précédent : on déduit par double inégalité que $\text{card}(E) = \text{card}(E)$ donc E et F sont équipotents à un même cardinal : ils sont donc équipotents. □

L'inégalité définie par « il existe une injection d'un ensemble dans l'autre » définie donc un ordre total sur les cardinaux.

On va pour finir définir les deux familles $\aleph$ et $\beth$ de cardinaux. Pour la première nous allons d'abord voir le résultat suivant :

Proposition 5.6.2. *Soit κ, alors il existe un plus petit cardinal plus grand que κ, que l'on va noter κ^+.*

Démonstration. Le résultat suivant nous montre que pour tout cardinal κ il existe un cardinal $\kappa' > \kappa$. Ainsi les cardinaux supérieurs strictement à κ forment une classe d'ordinaux, et possède donc un plus petit élément : cet élément est κ^+. □

On définit alors la classe $\aleph$ comme la hiérarchie des cardinaux infinis :

Définition 5.6.2 (Aleph)**.** *Les nombres aleph sont définis par induction transfinie :*

- $\aleph_0 = \omega$
- $\aleph_{S\alpha} = \aleph_\alpha^+$
- *si λ est un ordinal limite, alors $\aleph_\lambda = \bigcup_{\beta<\lambda} \aleph_\beta$.*

Le théorème de Cantor nous donne l'existence pour chaque ensemble d'un ensemble strictement plus grand :

Théorème 5.6.2 (Cantor)**.** *Soit E un ensemble, alors $\mathcal{P}(E)$ est de cardinal strictement supérieur à E.*

Démonstration. On a une injection évidente de E dans $\mathcal{P}(E)$ qui à x associe $\{x\}$. Supposons qu'il existe une bijection entre E et $\mathcal{P}(E)$. Notons cette bijection y, on définit alors $F = \{x \in E \mid x \notin f(x)\}$. Puisque f est une bijection, on trouve y tel que $F = f(y)$. Si $y \in F$, alors par définition de F on en déduit que $y \notin f(y) = F$, ce qui est une contradiction. Si $y \notin F$, alors par définition de F on en déduit que $y \in F$ puisque $F \notin f(y)$: on a encore une contradiction. Il n'y a donc pas de bijection entre E et $\mathcal{P}(E)$. □

On définit alors la hiérarchie des nombres $\beth$:

Définition 5.6.3 (Beth)**.** *Les nombres beth sont définis par induction transfinie :*

- $\beth_0 = \aleph_0$
- $\beth_{S\alpha} = \mathcal{P}(\beth_\alpha)$
- *si λ est un ordinal limite, $\beth_\lambda = \bigcup_{\beta<\lambda} \beth_\beta$.*

L'hypothèse du continu se formule alors de la façon suivante :

Axiome 5.6.1 (Hypothèse du continu)**.** $\beth_1 = \aleph_1$

Et l'hypothèse du continu généralisée est :

Axiome 5.6.2 (Hypothèse du continu généralisée)**.** Pour tout ordinal α, $\aleph_\alpha = \beth_\alpha$.

Remarque 5.6.2. Ces deux résultats sont donnés en tant qu'axiomes car il a été démontré par Cohen, avec la méthode du forcing, que l'hypothèse du continu était indépendante des axiomes de ZF(C) et donc indémontrable dans ces théories. Ces résultats sont donc ni vrais ni faux, ils dépendent des modèles de ZF(C) dans lesquels on se place. La démonstration de Cohen, se basant sur la technique du forcing, dépasse largement le cadre de ce cours.

Deuxième partie

Isomorphisme de Curry-Howard

Chapitre 6

Lambda-calcul non typé

Ce chapitre porte sur la version la plus basique du lambda-calcul, qui est le lambda-calcul non typé.

Le lambda-calcul a d'abord été développé par Church pour rendre compte de la notion de calculabilité, même si ce formalisme s'est porté bien plus fécond qu'attendu. Il est à la base du paradigme fonctionnel en programmation et les langages tels que Haskell et OCaml l'utilisent.

Nous commencerons par introduire la syntaxe du lambda-calcul, avec la forme des termes et les conversions α et β. Nous allons ensuite montrer que ce formalisme est aussi expressif que les machines de Turing en montrant son équivalence avec les fonctions récursives, puis clore le chapitre sur les théorèmes principaux du lambda-calcul.

6.1 Définition du lambda-calcul

L'idée première du lambda-calcul est d'étudier des fonctions. Le système consiste donc en un ensemble d'expressions dont on peut considérer qu'elles sont toutes des fonctions, mais nous commencerons pour donner l'intuition en ajoutant des constantes permettant de donner un sens plus évident à certaines notions. Une expression sera donc une fonction, de la forme $x \mapsto \Phi$ où Φ est une expression contenant x, ou bien l'application d'une fonction (i.e. d'une expression) à une autre expression.

Définition 6.1.1 (Lambda-terme)**.** *On se donne un ensemble $\mathcal{V}$ infini de variables, que l'on notera $x, y, \ldots$ et on définit l'ensemble des lambda-termes (ou expressions) Λ_0 par la grammaire suivante :*

$$M, N ::= x \mid \lambda x.M \mid (M\ N)$$

où $x \in \mathcal{V}$ et $M, N \in \Lambda_0$. On dit que M est une λ-abstraction si c'est un terme de la forme $\lambda x.M$ et que c'est une application si c'est un terme de la forme $(M\ N)$.

Moralement, le terme $\lambda x.M$ va représenter la fonction $x \mapsto M$, et le terme $(M\ N)$ va représenter l'application de M à l'argument N. Quand cela n'est pas nécessaire, on peut oublier les parenthèses extérieures et l'application est prioritaire sur la λ-abstraction (par exemple le terme $\lambda x.x\ x$ doit se lire $\lambda x.(x\ x)$ et est a priori différent de $(\lambda x.x)x$). Enfin on considère l'asssociativité de l'application à droite, ce qui signifie que le terme $M\ N\ P$ doit se lire $(M\ N)\ P$.

Un point important est la notion de curryfication : pour écrire une fonction de la forme $(x, y) \mapsto \Phi$, on utilise le fait que cette fonction à deux variables est équivalente à une fonction à une variable renvoyant une fonction à une variable, c'est-à-dire qu'on va considérer à la place la fonction $x \mapsto (y \mapsto \Phi)$. Cela signifie que si nous utilisons des fonctions à plusieurs variables, nous aurons à introduire autant de lambda-abstractions. Pour alléger les notations, on écrire $\lambda x\ y\ z.M$ au lieu de $\lambda x.\lambda y.\lambda z.M$.

On introduit les notions de variables libres et de variables libres dans les lambda-termes de façon analogue aux variables libres dans les propositions :

Définition 6.1.2 (Variable libre)**.** *Soit $M \in \Lambda_0$, on définit par induction l'ensemble* $\mathrm{vl}(M)$ *:*

- *si $M = x \in \mathcal{V}$, alors* $\mathrm{vl}(M) = \{x\}$.

- *si* $M = \lambda x.M'$ *alors* $\mathrm{vl}(M) = \mathrm{vl}(M') \setminus \{x\}$.
- *si* $M = (M'\ N')$ *alors* $\mathrm{vl}(M) = \mathrm{vl}(M') \cup \mathrm{vl}(N')$.

On dit que M *est un terme clos si* $\mathrm{vl}(M) = \varnothing$.

Remarque 6.1.1. Dans un lambda-terme comme $M := (\lambda x.x)x$, $x \in \mathrm{vl}(M)$ var malgré le λ qui supprime la liberté de x à gauche, le terme de droite x est bien constitué d'une variable libre.

Pour pouvoir définir les manipulations qui suivront, nous avons besoin de la notion de substitution, que nous allons définir rigoureusement par induction sur la structure de Λ_0.

Définition 6.1.3 (Substitution)**.** *Soient* $M, N \in \Lambda_0$ *et* $x \in \mathcal{V}$*. On définit la substitution de* x *par* N *dans* M*, que l'on note* $M[N/x]$*, par induction sur* M *:*

- *si* $M = x$ *alors* $M[N/x] = N$.
- *si* $M = y \in \mathcal{V}$*,* $y \neq x$*, alors* $M[N/x] = y$.
- *si* $M = \lambda x.M'$ *alors* $M[N/x] = M$.
- *si* $M = \lambda y.M'$*,* $y \neq x$*, alors on trouve* $z \notin \mathrm{vl}(N)$ *et* $M[N/x] = \lambda z.M'[y/z][N/x]$.
- *si* $M = M'\ N'$ *alors* $M[N/x] = (M[N/x])\ (N'[N/x])$.

Remarque 6.1.2. Le fait de ne pas reprendre y dans la substitution de $\lambda y.M'$ permet d'éviter la capture de variable dans le cas où $y \in \mathrm{vl}(N)$. Par exemple en considérant $M = \lambda y.x$ et $N = y\ y$, la substitution $M[N/x]$ vaudrait $\lambda y.y\ y$ qui sans la modification, là où avec notre ajustement on a $M[N/x] = \lambda z.y\ y$ qui fait plus sens, car z comme y sont des variables muettes et changer y en z ne change pas le sens de l'expression (nous allons détailler cette notion juste après avec l'α-conversion).

Cependant, pour que notre définition fonctionne, il faut que le choix de $z \notin \mathrm{vl}(N)$ dans la définition ne soit pas ambigu. Un moyen de régler ce problème est de considérer que chaque variable est numérotée, c'est-à-dire qu'on pourrait écrire $\mathcal{V} = \{v_0, v_1, \ldots\}$ et considérer $z = v_i$ où $i = \min\{j \mid j \in \mathbb{N}, v_j \notin \mathrm{vl}(N)\}$ par exemple. Ces considérations sont assez peu profondes et concernent surtout les ajustements à effectuer pour que les définitions fonctionnent de façon non approximative.

Exercice 6.1.1. Montrer que la substitution est bien définie. *Indication : on associera à chaque terme une quantité dans* $\mathbb{N}$ *qui décroît strictement à chaque étape de la substitution, en particulier pour passer de* M *à* $M'[y/z][N/x]$ *dans le quatrième cas.*

6.1.1 Alpha-conversion et indices de De Bruijn

Nous allons commencer par rendre rigoureuse la phrase « changer y en z ne change pas le sens de l'expression ». Le sens de cette phrase étant que l'on veut qu'une variable liée puisse être renommée sans changement sémantique. Renommer une variable liée signifie identifier, pour un terme $M \in \Lambda_0$, les deux termes $\lambda x.M$ et $\lambda y.M[y/x]$: nous allons effectuer cette identification en quotientant Λ_0 à partir de la clôture d'équivalence de toutes les telles identifications.

Définition 6.1.4 (α-équivalence)**.** *On définit la relation* $=_\alpha$ *sur* Λ_0 *par induction :*

$$\frac{}{M =_\alpha M} \qquad \frac{M =_\alpha N}{N =_\alpha M} \qquad \frac{M =_\alpha N \quad N =_\alpha P}{M =_\alpha P}$$

$$\frac{M =_\alpha M' \quad N =_\alpha N'}{M\ N =_\alpha M'\ N'} \qquad \frac{M =_\alpha N}{\lambda x.M =_\alpha \lambda x.N}\ x \in \mathcal{V} \qquad \frac{}{\lambda x.M =_\alpha \lambda y.M[y/x]}\ y \notin \mathrm{vl}(M)$$

Définition 6.1.5. *On définit maintenant l'ensemble des lambda-termes* Λ *comme* $\Lambda = {}^{\Lambda_0}/_{=_\alpha}$.

Exercice 6.1.2. Soient $M, N \in \Lambda_0$ tels que $M =_\alpha N$ et $P, Q \in \Lambda_0$ tels que $P =_\alpha Q$. Montrer que pour tout $x \in \mathcal{V}, M[P/x] =_\alpha N[Q/x]$.

On peut donc en déduire que la substitution passe au quotient.

Exercice 6.1.3. Soit $M \in \Lambda$ et $x \in \mathcal{V}$, montrer que $M[x/x] = M$.

Remarque 6.1.3. Les mots « lambda-termes » ou « expressions » deviennent alors ambigus, mais dans la pratique on considère uniquement l'ensemble Λ, l'ensemble Λ_0 ne servant qu'en première étape de construction, donc nous ne travaillerons plus avec des lambdas-termes sans identifier par α-équivalence. De même on confond par abus de notation M et $[M]$, le terme syntaxique et sa classe d'équivalence et $M[N/x]$ représentera en général $[M[N/x]]$.

Pour l'implémentation du lambda-calcul en tant que langage de programmation, la gestion des variables et de l'alpha-conversion est principalement une perte de temps. Pour pallier cela, et puisque la lisibilité n'est pas un critère important pour la machine, on considère la notation des indices de De Bruijn, qui remplace les variables par des nombres correspondant à la profondeur pour laquelle il faut remonter pour trouver une abstraction.

Définition 6.1.6 (Lambda-terme en indices de De Bruijn)**.** *On définit l'ensemble des lambda-termes en indices de De Bruijn* Λ_B *par induction :*

$$M, N ::= n \mid \lambda M \mid (M\ N)$$

où $n \in \mathbb{N}$ *et* $M, N \in \Lambda_B$.

On peut déjà faire correspondre une représentation en indices de De Bruijn à $\lambda x.(\lambda y.x\ y)x$ en considérant $(\lambda((\lambda(1\ 0))0)$. On doonne dans la figure 6.1.1 la façon de comprendre les notations en indices de De Bruijn.

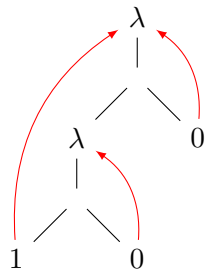


Figure 6.1 – Représentation de $\lambda((\lambda(1\ 0))0)$

Pour pouvoir définir convenablement une traduction $\Lambda \cong \Lambda_B$ en indices de De Bruijn, nous procédons par induction forte. Pour pouvoir faire cela, commençons par définir la notion de sous-terme.

Définition 6.1.7 (Sous-terme)**.** *Soit* $M \in \Lambda$*, on définit un sous-terme de* M *par un mot* u *dans* $\{0, 1, 2\}^*$ *de la façon suivante :*

- *si* $u = \varepsilon$ *alors le sous-terme correspondant à* u *est* M.
- *si* $M = \lambda x.M'$ *et* $u = 0 \cdot u'$ *alors le sous-terme correspondant à* u *est le sous-terme correspondant* u' *dans* M'.

- *si $M = M'\ N'$, alors soit $u = 1 \cdot u'$ et le sous-terme correspondant à u est le sous-terme correspondant à u' dans M', soit $u = 2 \cdot u'$ et le sous-terme correspondant à u est le sous-terme correspondant à u' dans N'.*

Pour un terme $M \in \Lambda$ fixé, on notera M_u le sous-terme de M correspondant à u. On dit qu'un sous-terme u de M est atomique s'il n'existe pas de mot v dont u est préfixe tel que v est un sous-terme de M. On note $\mathrm{Sub}(M)$ *l'ensemble des sous-termes de M.*

Exemple 6.1.1. Pour le terme $Y := \lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$, le sous-terme Y_{01022} est x et le sous-terme Y_{02} est $\lambda x.f(x\ x)$.

Exercice 6.1.4. Montrer que la définition de sous-terme ne dépend pas du représentant choisi pour M.

Exercice 6.1.5. Montrer que pour toute variable x apparaissant dans M, il existe un ensemble M_x de sous-termes tels que $\forall u \in M_x, M_u = x$.

Montrons qu'on peut définir une fonction sur un lambda-terme en définissant la fonction sur les sous-termes atomiques et son comportement sur les constructeurs. Ce résultat sera évident une fois qu'on aura montré que les sous-termes atomiques correspondent exactement aux variables d'un lambda-terme.

Proposition 6.1.1. *Soit $M \in \Lambda$, alors un sous-terme u de M est atomique si et seulement s'il est une variable.*

Démonstration. Supposons que u est atomique. Par définition, M est construit de l'une des façons suivantes : $(M = x \in \mathcal{V}) \vee (M = \lambda x.M') \vee (M = M'\ N')$ mais on remarque immédiatement que dans les deux cas suivants, on peut considérer respectivement $u \cdot 0$ et $u \cdot 1$ pour avoir un sous-terme dont u est préfixe, donc M_u est une variable.

Réciproquement, supposons que u est une variable. Par définition d'un sous-terme, il n'existe pas de cas inductif sur M_u permettant de définir un sous-terme dont u est préfixe, donc u est atomique. □

Proposition 6.1.2. *Soit $M \in \Lambda$ et E un ensemble, on suppose que :*

- *pour chaque sous-terme atomique u on associe un certain $f(u) \in E$*
- *on dispose d'une fonction $g : E \to E$*
- *on dispose d'une fonction $h : E \times E \to E$*

Alors il existe une unique fonction $f^* : \mathrm{Sub}(M) \to E$ telle que si u est atomique, $f^*(M_u) = f(u)$, si $M_u = \lambda x.M'$ alors $f^*(M_u) = g(f^*(M_{0\cdot u}))$ et si $M_u = M'\ N'$ alors $f^*(M_u) = h(f^*(M_{1\cdot u}), f^*(M_{2\cdot u}))$.

Démonstration. Le résultat est analogue à la définition d'une fonction par induction, mais nous allons prouver ce résultat en raisonnant sur les sous-termes de M directement. Tout d'abord, la taille d'un sous-terme (en nombre de lettres) est majorée, puisque M est défini par induction (c'est un objet d'une profondeur finie). Comme cette profondeur est finie, soit m le maximum des longueurs des sous-termes de M. A un sous-terme u, on associe $\overline{u} = m - |u|$ et on construit alors notre fonction f^* par étages :

- f_0 est définie sur l'ensemble des sous-termes u tels que $\overline{u} = 0$, et $f_0(u) = f(u)$. La fonction est bien définie car par définition, si $\overline{u} = 0$ alors u est atomique. Elle est la seule fonction à vérifier la proposition sur les mots tels que $\overline{u} = 0$.
- si f_n est définie pour $n \in \mathbb{N}$ sur les sous-termes tels que $\overline{u} \leq n$ de façon unique vérifiant la proposition initiale, alors on définit f_{n+1} sur les mots tels que $\overline{u} \leq n+1$ en posant $f_{n+1} = f_n$ sur l'ensemble de définition de f_n, et en traitant le reste par disjonction de cas :
  - si u est atomique, alors $f_{n+1}(M_u) = f(u)$.

— si u n'est pas atomique et que M_u est de la forme $\lambda x.M'$, alors $v := u \cdot 0$ est tel que $\overline{v} = n$, donc $f_n(M_v)$ est défini. On définit alors $f_{n+1}(M_u) = g(f_n(M_v))$

— si u n'est pas atomique et que M_u est de la forme $M'\ N'$, alors on pose $v := u \cdot 1$ et $w := u \cdot 2$, et comme $\overline{v} = \overline{w} = n$ on peut définir l'image de u, en posant $f_{n+1}(u) = h(f_n(v), f_n(w))$.

Il n'y a alors qu'une seule fonction f_{n+1} définie pour $\overline{u} \leq n+1$ qui vérifie la proposition initiale.

Ainsi par récurrence finie pour $n \leq m$ on définit une fonction f^* sur les sous-termes u tels que $\overline{u} \leq 0$, i.e. sur tous les sous-termes u, qui vérifie la proposition. Cette fonction est de plus unique. □

Remarque 6.1.4. Comme $M_\varepsilon = M$, cette fonction permet d'associer un élément de E à M.

On peut maintenant définir la traduction en indices de De Bruijn :

Définition 6.1.8. *Soit $M \in \Lambda$ un terme clos. On définit la traduction $\overline{M} \in \Lambda_B$ par induction sur les sous-termes de M :*

- *Pour u un sous-terme atomique, en notant $x = M_u$, on considère v le plus grand préfixe de u tel que $M_v = \lambda x.M'$ (ce préfixe existe car M_u n'est pas libre dans M) et on associe à u le nombre $|u|_0 - |v|_0 - 1$ où $|-|_0$ associe à un mot le nombre de 0 qui apparaissent dedans.*
- *On considère la fonction* $\lambda : \begin{array}{ccc} \Lambda_B & \longrightarrow & \Lambda_B \\ M & \longmapsto & \lambda M \end{array}$.
- *On considère la fonction* $\mathrm{app} : \begin{array}{ccc} \Lambda_B \times \Lambda_B & \longrightarrow & \Lambda_B \\ (M,N) & \longmapsto & M\ N \end{array}$.

Si M n'est pas clos, on fixe un environnement $\rho : \mathcal{V} \to \mathbb{N}$ qui est une fonction partielle contenant toutes les variables libres de M, et on définit la traduction $\overline{M}^\rho$ comme $\overline{M}$ mais en associant à une occurrence libre x de M le nombre $\rho(x) + |u|_0$ où u est le sous-terme associé.

Remarque 6.1.5. La traduction de $(\lambda x.x)x$ sera bien $(\lambda 0)\rho(x)$ et non $(\lambda\rho(x))\rho(x)$. Il faut donc définir $f(u)$ dans la fonction sur les sous-termes par disjonction de cas suivant s'il existe un préfixe de u tel que $M_u = \lambda x.M'$ et $M_u = x$ ou non.

Exercice 6.1.6. Montrer que si ρ et ρ' sont deux environnements qui coïncident sur les variables libres de M alors $\overline{M}^\rho = \overline{M}^{\rho'}$.

Proposition 6.1.3. *Soit $M \in \Lambda$ et $x \in \mathcal{V}$, alors*

$$\overline{\lambda x.M}^\rho = \lambda \overline{M}^{\rho_x}$$

où $\rho_x : y \mapsto \rho(y) + 1$ si $y \neq x$ et $\rho(x) = 0$.

Démonstration. Remarquons d'abord qu'un sous-terme u de M correspond exactement au sous-terme $0 \cdot u$ de $\lambda x.M$. Étant donnée la définition de $\overline{-}^\rho$, il suffit de montrer que $\overline{\lambda x.M}^\rho$ et $\lambda\overline{M}^{\rho_x}$ coïncident sur les sous-termes atomiques, i.e. sur les variables. Si $y \in \mathcal{V}$ est une occurrence de variable liée de M et que u est le sous-terme associé, alors en notant v le plus grand préfixe de u tel que $f(u) = |u|_0 - |v|_0 - 1$, on remarque que comme chaque sous-terme dans $\lambda x.M$ est un sous-terme de M avec 0 avant, on a $f(0 \cdot u) = 1 + |u|_0 - 1 - |v|_0 - 1 = f(u)$. Si $y \in \mathcal{V}$ est une occurrence libre dans M associée à u, alors y est remplacé dans M par $\rho_x(y) = 1 + \rho(y) + |u|_0$, et y est remplacé dans $\lambda x.M$ par $\rho(y) + |0 \cdot u|_0 = \rho(y) + |u|_0 + 1$. Enfin, pour une occurrence libre de x associée à u, la valeur associée dans M est $\rho_x(x) = |u|_0$ et celle dans $\lambda x.M$ est $|0 \cdot u|_0 - 1 = |u|_0$ car le préfixe de u le plus grand de la forme $\lambda x.M$ dans $\lambda x.M$ est ε, sinon x ne serait pas libre dans M.

Ainsi les deux termes coïncident sur les sous-termes atomiques, ce qui implique qu'ils sont identiques. □

Exercice 6.1.7. Montrer que la traduction en indices de De Bruijn est bijective entre les termes clos de Λ et ceux de Λ_B, en appelant terme clos de Λ_B un terme M tel que pour toute variable n apparaissant dans M, chaque sous-terme de M valant n possède au moins $n+1$ fois la lettre 0.

6.1.2 Beta-réduction

Nous allons nous intéresser à la notion de calcul en introduisant la β-réduction. L'idée derrière cette réduction est élémentaire : si l'on considère une fonction f et un argument a, alors $f(a)$ est équivalent à l'expression dans f où l'on remplace l'argument par a. On peut définir la β-réduction pour ne considérer que des remplacements les plus externes possibles, mais nous allons directement traiter la réduction à l'intérieur des sous-termes en donnant une relation qui commute avec les constructeurs.

Définition 6.1.9 (β-réduction)**.** *On définit la relation $\triangleright \subseteq \Lambda \times \Lambda$ par les règles suivantes :*

$$\frac{M \triangleright M'}{\lambda x.M \triangleright \lambda x.M'} \qquad \frac{M \triangleright M'}{M\ N \triangleright M'\ N} \qquad \frac{N \triangleright N'}{M\ N \triangleright M\ N'}$$

$$\frac{}{(\lambda x.M)N \triangleright M[N/x]}$$

On dit que M se réduit en un pas vers M' lorsque $M \triangleright M'$. On dit que M se réduit en M' lorsque $M \triangleright^* M'$, où $\triangleright^*$ est la clôture réflexive et transitive de $\triangleright$. Une définition possible de $M \triangleright^* M'$ est qu'il existe une suite finie $(M_i)_{i=1,\dots,n}$ avec $M_1 = M, M_n = M'$ et $\forall i \in \{1,\dots,n-1\}, M_i \triangleright M_{i+1}$.

Exemple 6.1.2. On définit le lambda-terme $I := \lambda x.x$. Pour tout lambda-terme $M \in \Lambda$, on a $I\ M \triangleright M$

$$\begin{aligned} I\ M &= (\lambda x.x)M \\ &\triangleright x[M/x] = M \end{aligned}$$

Exercice 6.1.8. Montrer que la définition de $\triangleright^*$ donne bien la plus petite relation réflexive et transitive contenant $\triangleright$.

Exercice 6.1.9. Soit le lambda-terme $\omega := \lambda x.x\ x$. On définit $\Omega := \omega\omega$. Chercher vers quel terme se réduit Ω.

Exercice 6.1.10. Réduire le plus possible le lambda-terme suivant :

$$(\lambda f.\lambda x.f\ (f\ (f\ x)))(\lambda f.\lambda x.f\ (f\ x))$$

On peut considérer la β-réduction comme l'exécution d'un calcul, et la notion de résultat correspond à ce qu'on appelle une forme normale.

Définition 6.1.10 (Forme normale, terme normalisable)**.** *On dit que $M \in \Lambda$ est une forme normale si $\forall N \in \Lambda. M \not\triangleright N$.*

Un terme $M \in \Lambda$ est dit faiblement normalisable s'il existe une forme normale N telle que $M \triangleright^* N$.

Un terme $M \in \Lambda$ est dit fortement normalisable si toute suite (M_n) telle que $\forall i \in \mathbb{N}, M_i \triangleright M_{i+1}$ et $M_0 = M$ est finie.

Comme le nom l'indique, un terme fortement normalisable est aussi faiblement normalisable.

On définit de plus la clôture d'équivalence de $\triangleright$ par $=_\beta$, qui sera la notion que nous utiliserons pour dire que deux termes ont le même sens.

Définition 6.1.11 (*beta*-équivalence)**.** *On définit $=_\beta$ comme la plus petite relation d'équivalence contenant $\triangleright$ (de façon équivalente, contenant $\triangleright^*$). Une définition de $M =_\beta M'$ et qu'il existe une suite finie $(M_i)_{i=1,\dots,n}$ telle que $M_1 = M$, $M_n = M'$ et $\forall i \in \{1,\dots,n\}, (M_i \triangleright M_{i+1}) \vee (M_{i+1} \triangleright M_i)$.*

Exercice 6.1.11. Montrer que cette définition donne bien la plus petite relation d'équivalence contenant $\triangleright$.

On définit enfin un opérateur important pour traiter les fonctions : la composition $\circ$.

Définition 6.1.12. *On définit le lambda-terme* rond *par* $\text{rond} := \lambda f.\lambda g.\lambda x.f\ (g\ x)$ *et on note* $g \circ f := \text{rond}\ g\ f$.

Nous allons donner plusieurs lemmes utiles sur la substitution et $\triangleright$.

Proposition 6.1.4. *Soit $M \in \Lambda$ et M' tel que $M \triangleright M'$ alors* $\text{vl}(M') \subseteq \text{vl}(M)$.

Démonstration. On montre d'abord, par induction sur N, que $\text{vl}(N[P/x]) \subseteq (\text{vl}(N) \setminus \{x\}) \cup \text{vl}(P)$:

- Si $N = x$ alors $N[P/x] = P$ et le résultat est direct.
- Si $N = y, y \neq x$ alors $N[P/x] = y$ et le résultat est direct.
- Si $N = K\ Q$ avec $\text{vl}(K[P/x]) \subseteq (\text{vl}(K) \setminus \{x\}) \cup \text{vl}(P)$ et $\text{vl}(Q[P/x]) \subseteq (\text{vl}(Q) \setminus \{x\}) \cup \text{vl}(P)$ alors $N[P/x] = K[p/x]\ Q[P/x]$ donc

$$\begin{aligned}\text{vl}(N[P/x]) &\subseteq (\text{vl}(K) \setminus \{x\}) \cup \text{vl}(P) \cup (\text{vl}(Q) \setminus \{x\}) \cup \text{vl}(P) \\ &= ((\text{vl}(K) \cup \text{vl}(Q)) \setminus \{x\}) \cup \text{vl}(P) \\ &= (\text{vl}(K\ Q) \setminus \{x\}) \cup \text{vl}(P)\end{aligned}$$

- Si $N = \lambda y.Q$ avec $\text{vl}(Q[P/x]) \subseteq (\text{vl}(Q) \setminus \{x\}) \cup \text{vl}(P)$ alors l'inclusion est encore valide en retirant $\{y\}$ de chaque côté.

D'où le résultat par induction.

On raisonne ensuite par induction sur $M \triangleright M'$:

- Si $M = (\lambda x.N)\ P$ alors $\text{vl}(M) = (\text{vl}(N) \setminus \{x\}) \cup \text{vl}(P)$ et par le lemme précédent, on en déduit que $\text{vl}(N[P/x]) \subseteq \text{vl}(M)$.
- Pour les autres règles de $M \triangleright M'$ on remarque qu'on applique des fonctions ensemblistes croissantes, donc l'inclusion est préservée.

Ce qui montre le résultat. □

Proposition 6.1.5. *Pour tous $M, N, P \in \Lambda$ et $x, y \in \mathcal{V}, x \neq y$ où $x \notin \text{vl}(P)$ on a*

$$M[N/x][P/y] = M[P/y][N[P/y]/x]$$

.

Démonstration. On procède par induction sur M :

- si $M = x$ alors $M[N/x] = N$ donc $M[N/x][P/y] = N[P/y] = M[P/y][N[P/y]/x]$.
- si $M = y$ alors $M[N/x] = y$ donc $M[N/x][P/y] = P = M[P/y][N[P/y]/x]$ car $x \notin \text{vl}(P)$ donc $P[Q/x] = P$ pour n'importe quel $Q \in \Lambda$.
- si $M = \lambda z.M'$ avec $M'[N/x][P/y] = M'[P/y][N[P/y]/x]$ alors par définition de la substitution d'une abstraction $(\lambda z.M'[N/x][P/y]) = M[P/y][N[P/y]/x]$.
- si $M = M'\ M''$ où l'hypothèse d'induction est vérifiée pour M' et M'', alors $M[N/x][P/y] = M[P/y][N[P/y]/x]$.

□

Proposition 6.1.6. *Si $M \rhd M'$ alors pour tout $y \notin \mathrm{vl}(M)$, $M[y/x] \rhd M'[y/x]$.*

Démonstration. On raisonne par induction sur $M \rhd M'$:

- si $M = \lambda z.P$ et $M' = \lambda z.P'$ avec $P[y/x] \rhd P'[y/x]$, alors $(\lambda z.P[y/x]) \rhd (\lambda z.P'[y/x])$
- si $M = P\ Q$ et $M' = P'\ Q$ avec $P[y/x] \rhd P'[y/x]$ alors $P[y/x]\ Q[y/x] \rhd P'[y/x]\ Q[y/x]$ donc $M[y/x] \rhd M'[y/x]$.
- si $M = P\ Q$ et $M' = P\ Q'$ avec $Q[y/x] \rhd Q'[y/x]$ alors $M[y/x] \rhd M'[y/x]$.
- si $M = (\lambda z.P)\ Q$ et $M' = P[Q/z]$ alors comme $z \notin \mathrm{vl}(y)$ (parce que $y \notin \mathrm{vl}(M)$), $P[Q/z][y/x] = P[y/x][Q[y/x]/z]$ donc $M[y/x] \rhd M'[y/x]$.

□

Proposition 6.1.7. *Si $M \rhd^* M'$ alors $\forall y \notin \mathrm{vl}(M), M[y/x] \rhd^* M'[y/x]$.*

Démonstration. Soit une suite $(M_i)_{i=1,\dots,n}$ témoignant que $M \rhd^* M'$, on va montrer par récurrence sur n que $M[y/x] \rhd M'[y/x]$ et que $y \notin \mathrm{vl}(M')$:

- Si $n = 1$ alors on a de façon évidente $M[y/x] \rhd^* M[y/x]$ et par hypothèse $x \in \mathrm{vl}(M)$.
- Si la proposition est vraie pour un certain n, alors soit une suite de taille $n+1$: on considère $N = M_n$. Par hypothèse de récurrence, $M[y/x] \rhd N[y/x]$ et $y \notin \mathrm{vl}(N)$. On peut donc appliquer la proposition précédente à la réduction $N \rhd M'$ pour en déduire que $N[y/x] \rhd M'[y/x]$ et comme $\mathrm{vl}(M') \subseteq \mathrm{vl}(N)$, cela signifie que $y \notin \mathrm{vl}(M')$.

Ainsi par récurrence $M[y/x] \rhd^* M'[y/x]$. □

6.1.3 Réduction avec les indices de De Bruijn

On peut se demander comment se comporte la réduction sur les termes dans Λ_B. La première étape est de considérer l'action de la substitution. Dans un cas simple, si l'on considère par exemple $M = \lambda\lambda 1$ et N un terme clos, alors remplacer le 1 dans M (et supprimer le λ extérieur) va simplement nous donner λN car l'ensemble des variables utilisées dans N ne « sort » pas du terme. Cependant, cela ne suffit pas dans le cas où N n'est pas libre.

Exemple 6.1.3. Si l'on prend $M = \lambda x.\lambda y.x\ y$ et $N = z$, alors $P := \lambda z.(M\ N) \rhd \lambda z.\lambda y.z\ y$ mais on remarque que la traduction de N n'est pas définie car z est libre dans N. Cependant, cette variable libre dans N mais pas dans le terme total P peut être récupérée en indice de De Bruijn.

On définit donc la fonction de lifting, qui va actualiser dans une situation comme celle ci-dessus, actualiser les variables libres de N pour changer leur référence. Cela sera utile dans la définition de substitution sur les indices de De Bruijn.

Définition 6.1.13 (Lifting)**.** *Soit $M \in \Lambda_B$, on définit l'opération de lifting, notée $\uparrow_i^j$ par induction sur M :*

- *si $M = n \in \mathbb{N}$ et $n < i$ alors $M\uparrow_i^j = M$.*
- *si $M = n \in \mathbb{N}$ et $n \geq i$ alors $M\uparrow_i^j = n + j$.*
- *si $M = \lambda M'$ alors $M\uparrow_i^j = \lambda(M'\uparrow_{i+1}^j)$.*
- *si $M = M'\ N'$ alors $M\uparrow_i^j = (M'\uparrow_i^j)\ (N'\uparrow_i^j)$.*

Exercice 6.1.12. Montrer que $M\uparrow_i^j\uparrow_i^k = M\uparrow_i^{j+k}$.

Cela nous permet de définir la substitution : le lifting $\uparrow_0^j$ va augmenter la référence de toutes les variables libres de N de j et il suffit donc d'ajuster j pour être le nombre de λ rencontrés dans M.

Définition 6.1.14 (Substitution). *Soit $M \in \Lambda_B$, $N \in \Lambda_B$ et $i \in \mathbb{N}$. On définit $M[N/i]$ par induction sur M :*

- *si $M = i$ alors $M[N/i] = N$.*
- *si $M = n$ et $n \neq i$ alors $M[N/i] = n$.*
- *si $M = \lambda M'$ alors $M[N/i] = \lambda(M'[N\uparrow_0^1 /i+1])$.*
- *si $M = M'\ N'$ alors $M[N/i] = M'[N/i]\ N'[N/i]$.*

Théorème 6.1.1. *Soient $M, N \in \Lambda$, ρ un environnement contenant les variables libres de M et N et $x \in \mathrm{vl}(M)$, alors*

$$\overline{M[N/x]}^{\rho} = \overline{M}^{\rho}[\overline{N}^{\rho}/\rho(x)]$$

Remarque 6.1.6. Si x n'est pas une variable libre de M alors la substitution donne simplement M ce qui n'est pas intéressant.

Démonstration. On va montrer ce résultat par induction sur M, pour tout $N \in \Lambda$, tout environnement ρ contenant les variables libres de M et N et tout $x \in \mathrm{vl}(M)$:

- si $M = x$ alors soient $N \in \Lambda$, ρ environnement convenable et $x \in \mathrm{vl}(M)$. $M[N/x] = N$ et $\overline{M}^{\rho} = \rho(x)$ donc $\overline{M}^{\rho}[\overline{N}^{\rho}/\rho(x)] = \overline{N}^{\rho} = \overline{M[N/x]}^{\rho}$.
- si $M = y, y \neq x$ alors soient $M \in \Lambda, \rho$ et x convenables. Alors $M[N/x] = y$ et $\overline{M}^{\rho} = \rho(y)$ donc $\overline{M}^{\rho}[\overline{N}^{\rho}/\rho(x)] = \rho(y) = \overline{M[N/x]}^{\rho}$.
- si $M = \lambda y.M', y \neq x$ alors soient N, ρ, x vérifiant les hypothèses. Quitte à effectuer une α-conversion sur M, y est libre dans M' et n'apparaît pas dans N, et $M[N/x] = \lambda y.M'[N/x]$. Par hypothèse d'induction on sait que $\overline{M'[N/x]}^{\rho} = \overline{M'}[\overline{N}^{\rho}/\rho(x)]$. Par définition,

  $$\overline{M[N/x]}^{\rho} = \lambda(\overline{M'[N/x]}^{\rho_y}) = \lambda(\overline{M'}^{\rho_y}[\overline{N}^{\rho_y}/\rho(x)+1])$$

  en utilisant l'hypothèse d'induction et le fait que $\rho_y(x) = \rho(x) + 1$. De plus,

  $$\overline{M}^{\rho}[\overline{N}^{\rho}/\rho(x)] = \lambda(\overline{M'}^{\rho_y}[(\overline{N}^{\rho})\uparrow_0^1 /\rho(x)+1])$$

  Il nous suffit donc de montrer que $(\overline{N}^{\rho})\uparrow_0^1 = \overline{N}^{\rho_y}$. Pour cela, on raisonne par induction sur N en généralisant la situation à $\uparrow_k^1$ pour $k \in \mathbb{N}$ et $\rho_{y,k}$ qui est défini comme $\rho_{y,k}(z) = \rho(z) + 1$ si $\rho(z) \geq k$ et $\rho(z)$ sinon, et $\rho_{y,k}(y) = 0$, c'est-à-dire qu'on montre que $(\overline{N}^{\rho})\uparrow_k^1 = \overline{N}^{\rho_{y,k}}$ pour tout ρ :
  - si $N = z \in \mathcal{V}$, sachant que $z \neq y$ par hypothèse, alors $\overline{N}^{\rho_{y,k}} = \rho_{y,k}(z)$ et $(\overline{N}^{\rho})\uparrow_k^1 = (\rho(z))\uparrow_k^1$. Si $\rho(z) \geq k$ alors les deux valeurs valent $\rho(z)+1$, et sinon elles valent $\rho(z)$.
  - si $N = \lambda z.N'$ alors $(\overline{\lambda z.N'}^{\rho})\uparrow_k^1 = (\lambda\overline{N'}^{\rho_{z,0}})\uparrow_k^1 = \lambda(\overline{N'}^{\rho_{z,0}}\uparrow_{k+1}^1)$ et $\overline{\lambda z.N'}^{\rho_{y,k}} = \lambda(\overline{N'}^{(\rho_{y,k})_{z,0}})$. Par hypothèse d'induction, on peut écrire $\lambda(\overline{N'}^{\rho_{z,0}}\uparrow_{k+1}^1) = \lambda(\overline{N'}^{(\rho_{z,0})_{y,k+1}})$. On veut donc montrer que $(\rho_{z,0})_{y,k+1} = (\rho_{y,k})_{z,0}$. Soit $\alpha \in \mathcal{V}$ (on ignore le cas $\alpha = y$ car y n'apparaît pas dans nos termes), si $\rho(\alpha) = 0$ ou $\alpha = z$ alors son image par les deux fonctions est 2 si $k = 0$ et 1 sinon. Si $\rho(\alpha) < k$ alors les deux fonctions renvoient $\rho(\alpha)+1$. Si $\rho(\alpha) \geq k$ alors les deux fonctions renvoient $\rho(\alpha)+2$. Les deux fonctions coïncident donc : elles sont égales.
  - si $N = M'\ N'$ alors $(\overline{N}^{\rho})\uparrow_0^1 = (\overline{M'}^{\rho})\uparrow_0^1\ \ (\overline{N'}^{\rho})\uparrow_0^1$ puis par hypothèse d'induction on en déduit $(\overline{N}^{\rho})\uparrow_0^1 = \overline{M'}^{\rho_y}\ \overline{N'}^{\rho_y} = \overline{N}^{\rho_y}$.

  Donc en particulier $(\overline{N}^{\rho})\uparrow_0^1 = \overline{N}^{\rho_y}$, d'où l'égalité.
- si $M = \lambda x.M'$ alors par α-conversion on peut poser $M = \lambda y.M'[y/x]$ avec $y \neq x$, permettant de se ramener au cas précédent.
- si $M = M'\ N'$ alors soient N, ρ, x vérifiant les hypothèses. Par définition de la substitution, $M[N/x] = M'[N/x]\ N'[N/x]$ donc $\overline{M[N/x]}^{\rho} = \overline{M'[N/x]}^{\rho}\ \overline{N'[N/x]}^{\rho}$ par définition de la traduction, ce qui, en utilisant l'hypothèse d'induction, donne

  $$\overline{M[N/x]}^{\rho} = (\overline{M'}^{\rho}[\overline{N}^{\rho}/\rho(x)])\ (\overline{N'}^{\rho}[\overline{N}^{\rho}/\rho(x)])$$

  qui est par définition $\overline{M'\ N'}^{\rho}[\overline{N}^{\rho}/\rho(x)] = \overline{M}^{\rho}[\overline{N}^{\rho}/x]$.

Ainsi, par induction, l'égalité est vérifiée. □

On peut donc définir une β-réduction sur Λ_B par transport de structure, mais nous donnerons d'abord la définition purement axiomatique.

Définition 6.1.15 (β-réduction dans Λ_B)**.** *On définit sur Λ_B la relation $\triangleright$ par les règles suivantes :*

$$\frac{M \triangleright M'}{\lambda M \triangleright \lambda M'} \qquad \frac{M \triangleright M'}{M\ N \triangleright M'\ N} \qquad \frac{N \triangleright N'}{M\ N \triangleright M\ N'}$$

$$\frac{}{(\lambda M)N \triangleright M[N/0]}$$

Proposition 6.1.8. *Soient $M, N \in \Lambda$ des termes et ρ un environnement pour les traduire. Si $M \triangleright N$ alors $\overline{M}^\rho \triangleright \overline{N}^\rho$.*

Démonstration. On prouve le résultat par induction :

- si $\overline{M}^\rho \triangleright \overline{M'}^\rho$, alors $\overline{\lambda x.M}^\rho = \lambda \overline{M}^{\rho[x\mapsto 0]} \triangleright \lambda \overline{M'}^{\rho[x\mapsto 0]} = \overline{\lambda x.M'}^\rho$.
- si $\overline{M}^\rho \triangleright \overline{M'}^\rho$ alors $\overline{M\ N}^\rho = \overline{M}^\rho\ \overline{N}^\rho \triangleright \overline{M'}^\rho\ \overline{N}^\rho = \overline{M'\ N}^\rho$.
- si $\overline{N}^\rho \triangleright \overline{N'}^\rho$ alors $\overline{M\ N}^\rho = \overline{M}^\rho\ \overline{N}^\rho \triangleright \overline{M}^\rho\ \overline{N'}^\rho = \overline{M\ N'}^\rho$.
- $\overline{(\lambda x.M)N}^\rho \triangleright \overline{M[N/x]}^\rho$ car $\overline{(\lambda x.M)N}^\rho = (\lambda \overline{M}^{\rho[x\mapsto 0]})\overline{N}^\rho$ et $\overline{M[N/x]}^{\rho[x\mapsto 0]} = \overline{M}^\rho[\overline{N}^\rho/0]$ d'après le théorème précédent.

□

Corollaire 6.1.1. *Si $M \triangleright N$ alors $\overline{M} \triangleright \overline{N}$ pour tous $M, N \in \Lambda$ clos.*

Corollaire 6.1.2. *Si $M \triangleright^* N$ alors $\overline{M}^\rho \triangleright^* \overline{N}^\rho$.*

6.2 Le lambda-calcul est Turing-complet

Le but de cette section est de montrer que l'on peut simuler les fonctions récursives avec des lambda-termes. Nous commencerons par montrer comment coder différentes structures de données élémentaires, puis nous donnerons le théorème de Turing-complétude du λ-calcul.

6.2.1 Entiers de Church

La première étape est de définir un codage pour les entiers, c'est-à-dire que l'on va associer à chaque entier $n \in \mathbb{N}$ un lambda-terme $\underline{n}$ correspondant. Ces lambda-termes s'appellent les entiers de Church.

Définition 6.2.1 (Entiers de Church)**.** *Soit $n \in \mathbb{N}$, on définit le codage $\underline{n}$ de n par*

$$\underline{n} := \lambda f.\lambda x.f^n\ x$$

où $f^0\ x = x$ et $f^{n+1}\ x = f\ (f^n\ x)$.

On sait que les données importantes sont en fait les constructeurs 0 et S, pour définir $\mathbb{N}$, et nous allons donc donner une présentation des entiers de Church n'utilisant que ces deux constructeurs.

Proposition 6.2.1. *Soient* $\underline{0} := \lambda f.\lambda x.x$ *et* $\underline{S} := \lambda n.\lambda f.\lambda x.f\ (n\ f\ x)$ *alors* $\forall n \in \mathbb{N}, \underline{n} =_\beta \underline{S}^n(\underline{0})$

Démonstration. On raisonne par récurrence :

- $\underline{0} = \underline{S}^0(\underline{0})$ par définition de $\underline{S}^0$.
- On suppose que $\underline{n} =_\beta \underline{S}^n(\underline{0})$, alors :

$$\begin{aligned}
\underline{n+1} &=_\beta \lambda f.\lambda x.f^{n+1}\ x \\
&=_\beta \lambda f.\lambda x.f\ (f^n\ x) \\
\underline{S}^{n+1}(\underline{0}) &= \underline{S}(\underline{S}^n(\underline{0})) \\
&=_\beta (\lambda n.\lambda f.\lambda x.f\ (n\ f\ x))(\underline{n}) \\
&\rhd \lambda f.\lambda x.f\ (\underline{n}\ f\ x) \\
&\rhd \lambda f.\lambda x.f\ (f^n\ x)
\end{aligned}$$

D'où le résultat par récurrence. □

On en déduit le résultat suivant :

Corollaire 6.2.1. *Soit* P *un prédicat sur* Λ *stable par* $=_\beta$*, c'est-à-dire tel que* $\forall M \in \Lambda, (P(M) \wedge (M =_\beta N)) \implies P(N)$*. Si* $P(\underline{0})$ *et* $\forall n \in \mathbb{N}, P(\underline{n}) \implies P(\underline{S}\ \underline{n})$ *alors* $\forall n \in \mathbb{N}, P(\underline{n})$*.*

Démonstration. On montre que l'ensemble $\{M \mid P(M)\}$ contient $\underline{\mathbb{N}} = \{\underline{n} \mid n \in \mathbb{N}\}$. Supposons qu'il existe un $n \in \mathbb{N}$ tel que $\neg P(\underline{n})$, alors on peut considérer un n minimal respectant cette condition : comme $n \neq 0$ on en déduit que $P(\underline{n-1})$ donc $P(\underline{S}\ \underline{n-1})$ mais $\underline{S}\ \underline{n-1} =_\beta \underline{S}^{1+(n-1)}\ \underline{0} =_\beta \underline{n}$ donc $P(\underline{n})$, ce qui est absurde. □

Remarque 6.2.1. On ne vérifiera pas, en général, que P est stable par $=_\beta$ car ce sera évident (les propositions dans cette partie utiliseront comme relation atomique $=_\beta$). Dans ce document, on fera référence à ce raisonnement comme la λ-récurrence.

L'intuition derrière le codage de n est de considérer la fonction d'ordre supérieure qui, étant donnée une fonction f, retourne la fonction qui itère n fois f. Cette intuition va nous guider pour définir facilement les opérations $+$ et $\times$.

Définition 6.2.2 (Addition)**.** *On définit le terme* add *par*

$$\text{add} := \lambda n.\lambda m.n\ \underline{S}\ m$$

et ce terme vérifie

$$\forall n \in \mathbb{N}, \forall m \in \mathbb{N}, \text{add}\ \underline{n}\ \underline{m} =_\beta \underline{n+m}$$

Démonstration. Il nous suffit de prouver cette propriété par λ-récurrence sur n avec m quelconque fixé avant :

- Tout d'abord :

$$\begin{aligned}
\text{add}\ \underline{0}\ \underline{n} &=_\beta \underline{0}\ \underline{S}\ \underline{n} \\
&=_\beta \underline{S}^0\ \underline{n} \\
&=_\beta \underline{n} = \underline{0+n}
\end{aligned}$$

- Si l'on suppose que add $\underline{n}\ \underline{m} =_\beta \underline{n+m}$ alors :

$$\begin{aligned}\text{add } \underline{n+1}\ \underline{m} &=_\beta \underline{n+1}\ \underline{S}\ \underline{m}\\ &=_\beta \underline{S}\ (\underline{S}^n\ (\underline{m})\\ &=_\beta \underline{S}\ (\underline{n}\ \underline{S}\ \underline{m})\\ &=_\beta \underline{S}\ (\text{add } \underline{n}\ \underline{m})\\ &=_\beta \underline{S}(\underline{n+m})\\ &=_\beta \underline{n+m+1}\\ &=_\beta \underline{(n+1)+m}\end{aligned}$$

D'où le résultat par récurrence. □

Exercice 6.2.1. Montrer que le terme add $:= \lambda n.\lambda m.\lambda f.\lambda x.n\ f\ (m\ f\ x)$ vérifie la même propriété et qu'il est donc un choix possible pour l'addition.

Pour alléger les notations, on notera maintenant $n + m$ pour add $n\ m$. De façon similaire on peut définir la multiplication.

Définition 6.2.3 (Multiplication)**.** *On définit le terme* mult *par*

$$\text{mult} := \lambda n.\lambda m.n\ (\text{add}\ m)\ \underline{0}$$

et ce terme vérifie

$$\forall n \in \mathbb{N}, \forall m \in \mathbb{N}, \text{mult}\ \underline{n}\ \underline{m} =_\beta \underline{n \times m}$$

Démonstration. Par λ-récurrence sur n :

- Si $n = 0$:

$$\begin{aligned}\text{mult } \underline{0}\ \underline{m} &=_\beta \underline{0}\ (\text{add } \underline{m})\ \underline{0}\\ &=_\beta \underline{0} = \underline{0 \times m}\end{aligned}$$

- Si l'on suppose que mult $\underline{n}\ \underline{m} =_\beta \underline{n \times m}$ alors :

$$\begin{aligned}\text{mult } \underline{n+1}\ \underline{m} &=_\beta \underline{n+1}\ (\text{add } \underline{m})\ \underline{0}\\ &=_\beta \text{add } \underline{m}\ (\underline{n}\ (\text{add } \underline{m})\ \underline{0}\\ &=_\beta \text{add } \underline{m}\ (\text{mult } \underline{n}\ \underline{m})\\ &=_\beta \text{add } \underline{m}\ \underline{n \times m}\\ &=_\beta \underline{n \times m + m}\\ &=_\beta \underline{(n+1) \times m}\end{aligned}$$

D'où le résultat par récurrence. □

On notera $n \times m$ pour mult $n\ m$.

Exercice 6.2.2. Montrer que

$$\forall n \in \mathbb{N}, \forall m \in \mathbb{N}, \underline{n}\ \underline{m} =_\beta \underline{m^n}$$

et en déduire une traduction de la fonction $\begin{array}{ccc}\exp : & \mathbb{N} \times \mathbb{N} & \longrightarrow & \mathbb{N} \\ & (n, m) & \longmapsto & n^m\end{array}$ en lambda-terme.

6.2.2 Structure de données

Nous allons nous intéresser maintenant au codage des tuples. Remarquons d'abord qu'il est suffisant de coder des paires : si l'on sait coder $\langle x, y\rangle$ alors on peut coder $\langle x_1, \dots, x_n\rangle$ par $\langle \cdots \langle x_1, x_2\rangle, x_3\rangle, \dots, x_n\rangle$ voire par $\langle \cdot \langle \bullet, x_1\rangle, x_2\rangle, \dots, x_n\rangle$, où $\bullet$ est un lambda-terme quelconque, qui donne une présentation plus systématique des projections. Commençons donc par définir le codage des paires et des projections.

Définition 6.2.4 (Paires)**.** *Soient $M, N \in \Lambda$, on définit le lambda-terme*

$$\langle M, N\rangle := \lambda p.p\ M\ N$$

et les lambda-termes

$$\pi_1 := \lambda p.(p\ (\lambda x.\lambda y.x)) \qquad \pi_2 := \lambda p.(p\ (\lambda x.\lambda y.y))$$

Proposition 6.2.2. *Pour tous termes $M, N \in \Lambda$, on a $\pi_1\ \langle M, N\rangle =_\beta M$ et $\pi_2\ \langle M, N\rangle =_\beta N$*

Démonstration. Prouvons l'identité pour π_1, l'autre cas étant identique :

$$\begin{aligned}\pi_1\ \langle M, N\rangle &=_\beta (\lambda p.p\ (\lambda x.\lambda y.x))(\lambda p.p\ M\ N)\\ &=_\beta (\lambda p.p\ M\ N)(\lambda x.\lambda y.x)\\ &=_\beta (\lambda x.\lambda y.x)\ M\ N\\ &=_\beta M\end{aligned}$$

□

On peut alors définir des tuples, et leurs projections associées.

Définition 6.2.5 (Tuple)**.** *Soient $M_1, \dots, M_n$ des lambda-termes, on définit $(M_1, \dots, M_n)$ par induction sur $n \in \mathbb{N}^*$:*

- $(M_1) := \langle \lambda x.x, M_1\rangle$
- $(M_1, \dots, M_{n+1}) := \langle (M_1, \dots, M_n), M_{n+1}\rangle$

On définit de plus les projections $\pi_{n,i}, i \leq n$ par induction sur $n - i$:

- $\pi_{n,n} := \pi_2$
- $\pi_{n,i} := \pi_{n-1,i} \circ \pi_1$

Proposition 6.2.3. *Soient $M_1, \dots, M_n$ des lambda-termes, alors*

$$\forall i \in \{1, \dots, n\}, \pi_{n,i}\ (M_1, \dots, M_n) =_\beta M_i$$

Démonstration. On prouve ce résultat pour tout $i \leq n$ et $M_1, \dots, M_n \in \Lambda$ par récurrence sur $n \in \mathbb{N}^*$:

- Si $n = 1$ alors $(M_1) = \langle I, M_1\rangle$ et $\pi_{1,1} = \pi_2$, ce qui avec la proposition précédente nous donne bien le résultat.
- Si pour tout $i \leq n$, l'équation est vraie, alors soit $M_1, \dots, M_{n+1} \in \Lambda$. Soit $i \leq n + 1$. Si $i = n + 1$ alors

$$\pi_{n+1,i}\ (M_1, \dots, M_{n+1}) = \pi_2\ \langle (M_1, \dots, M_n), M_{n+1}\rangle =_\beta M_{n+1}$$

Si $i \leq n$ alors

$$\begin{aligned}\pi_{n+1,i}\ (M_1, \dots, M_{n+1}) &= (\pi_{n,i} \circ \pi_1)\ (M_1, \dots, M_{n+1})\\ &=_\beta \pi_{n,i}\ (\pi_1\ \langle (M_1, \dots, M_n), M_{n+1}\rangle)\\ &=_\beta \pi_{n,i}\ (M_1, \dots, M_n)\\ &=_\beta M_i\end{aligned}$$

D'où le résultat. □

La structure de tuple sera utile pour le résultat théorique d'équivalence du lambda-calcul avec les fonctions récursives, mais elle donne aussi une idée simple pour définir les booléens et les conditions : dire « si b alors M sinon N » revient à considérer $\langle M, N\rangle$ et identifier les projections avec les valeurs de vérité.

Définition 6.2.6 (Booléen)**.** *On définit deux lambda-termes*

$$\top := \lambda x.\lambda y.x \qquad \bot := \lambda x.\lambda y.y$$

Exercice 6.2.3. Montrer que la lambda-terme $\neg := \lambda b.b \bot \top$ vérifie $\neg \top =_\beta \bot$ et $\neg \bot =_\beta \top$.

Exercice 6.2.4. Construire deux lambda-termes, respectivement $\vee$ et $\wedge$ représentant les opérations booléennes de disjonction et de conjonction. Montrer qu'elles respectent les tables de vérité attendues.

Définition 6.2.7 (Condition)**.** *Soient $M, N, b \in \Lambda$, on définit*

$$\texttt{if } b \texttt{ then } M \texttt{ else } N := b\ M\ N$$

On peut vérifier de façon évidente que les conditions se comportent comme attendu : on a bien `if` $\top$ `then` M `else` $N =_\beta M$ et `if` $\bot$ `then` M `else` $N =_\beta N$. Enfin, pour que les booléens puissent exprimer une condition pertinente, il faut pouvoir trouver au moins une fonction qui associe un booléen à un entier.

Définition 6.2.8 (Égalité à 0)**.** *On définit le lambda-terme suivant :*

$$\texttt{eq0} := \lambda n.n\ (\lambda x.\bot)\ \top$$

Ce terme vérifie $\texttt{eq0}\ \underline{0} =_\beta \top$ *et* $\texttt{eq0}\ (\underline{S}\ \underline{n}) =_\beta \bot$.

Démonstration. Il suffit de faire le calcul :

$$\begin{aligned}
\texttt{eq0}\ \underline{0} &=_\beta \underline{0}\ (\lambda x.\bot)\ \top \\
&=_\beta \top \\
\texttt{eq0}\ (\underline{S}\ \underline{n}) &=_\beta (\underline{S}\ \underline{n})\ (\lambda x.\bot)\ \top \\
&=_\beta (\lambda x.\bot)(\underline{n}\ (\lambda x.\bot)\ \top) \\
&=_\beta \bot
\end{aligned}$$

□

Exercice 6.2.5 (L'opération prédécesseur)**.** L'objectif de cet exercice est de trouver un terme pred tel que $\text{pred}(\underline{0}) =_\beta \underline{0}$ et $\text{pred}(\underline{S}\ \underline{n}) =_\beta \underline{n}$.

1. Construire un lambda-terme f tel que $f\ \langle M, \underline{n}\rangle =_\beta \langle \underline{n}, \underline{n+1}\rangle$
2. Montrer que $\underline{n}\ f\ \langle \underline{0}, \underline{0}\rangle =_\beta \langle \underline{n-1}, \underline{n}\rangle$
3. En déduire un lambda-terme pred qui vérifie les conditions de l'énoncé et montrer que ces conditions sont bien vérifies.

Définition 6.2.9 (Soustraction)**.** *On définit le lambda-terme*

$$\text{min} := \lambda n.\lambda m.m \text{ pred } n$$

et ce lambda-terme vérifie, pour tous $n, m \in \mathbb{N}$*, si* $n \leq m$ *alors* min $\underline{n}\ \underline{m} =_\beta \underline{0}$ *et sinon* min $\underline{n}\ \underline{m} = \underline{n-m}$*.*

Démonstration. Montrons d'abord, par λ-récurrence, que $\forall n, m \in \mathbb{N}, \text{min } \underline{m+n}\ \underline{n} =_\beta \underline{m}$:

- Par définition, $\underline{0}$ pred $=_\beta I$ donc min $\underline{m+0}\ \underline{0} =_\beta \underline{m}$.
- Si min $\underline{m+n}\ \underline{n} =_\beta \underline{m}$ alors

$$\begin{aligned}
\text{min } \underline{m+n+1}\ \underline{n+1} &=_\beta \text{min } (\underline{S}\ \underline{n})\ (\underline{S}^{m+n+1}\ \underline{0}) \\
&=_\beta (\underline{S}^{m+n+1} \text{ pred})\ (\underline{S}\ \underline{n}) \\
&=_\beta (\underline{S}^{m+n} \text{ pred})\ (\text{pred } (\underline{S}\ \underline{n})) \\
&=_\beta \underline{n} \text{ pred } \underline{m+n} \\
&=_\beta \text{min } \underline{m+n}\ \underline{n} \\
&=_\beta \underline{m}
\end{aligned}$$

De plus, montrons que min $\underline{n}\ \underline{m+p} =_\beta \text{min}(\text{min } \underline{n}\ \underline{m})\ \underline{p}$ par récurrence sur p :

- D'abord $\underline{m+0} = \underline{m}$ et pour tout $M \in \Lambda, \text{min } M\ \underline{0} =_\beta M$ donc le résultat est vrai pour $p = 0$.
- Si pour tout $n, m \in \mathbb{N}, \text{min } \underline{n}\ \underline{m+p} =_\beta \text{min}(\text{min } \underline{n}\ \underline{m})\ \underline{p}$ alors pour $p+1$ on a :

$$\begin{aligned}
\text{min } \underline{n}\ \underline{m+p+1} &=_\beta \text{pred}^{m+p+1}\ \underline{n} \\
&=_\beta \text{pred } (\text{min } (\text{min } \underline{n}\ \underline{m})\ \underline{p}) \\
&=_\beta \text{min } (\text{min } \underline{n}\ \underline{m})\ \underline{p+1}
\end{aligned}$$

En combinant les deux résultats, si $n \leq m$, en écrivant $m = n + (m-n)$, on en déduit que

$$\text{min } \underline{n}\ \underline{m} =_\beta \text{min } (\text{min } \underline{n}\ \underline{n})\ \underline{m-n} =_\beta \text{min } \underline{0}\ \underline{m-n}$$

et min $\underline{0}\ \underline{n} =_\beta \underline{0}$ pour tout $n \in \mathbb{N}$ puisque c'est un point fixe de la fonction pred.

Enfin, si $m \leq n$ alors on peut écrire $n = m + (n-m)$ donc

$$\text{min } \underline{n}\ \underline{m} = \text{min } \underline{(n-m)+m}\ \underline{m} =_\beta \underline{n-m}$$

□

On notera par souci de lisibilité $n - m$ pour le terme min $n\ m$ mais il faudra faire attention à bien considérer que cette soustraction est dans $\mathbb{N}$. De plus on notera $n - \underline{1}$ plutôt que pred n en général.

Exercice 6.2.6 (Inégalité)**.** Construire un lambda-terme ineq tel que pour tous $n, m \in \mathbb{N}, \text{ineq } \underline{n}\ \underline{m} =_\beta \top$ si $n \leq m$ et ineq $\underline{n}\ \underline{m} =_\beta \bot$ sinon.

Exercice 6.2.7 (Égalité)**.** Constuire un lambda-terme eq tel que eq $\underline{n}\ \underline{n} =_\beta \top$ et eq $\underline{n}\ \underline{m} =_\beta \bot$ pour $n \neq m$ deux entiers quelconques, et montrer que ce lambda-terme convient bien.

On notera $n == m$ pour eq $n\ m$.

6.2.3 Point fixe et récursion

L'élément qui nous manque maintenant est de pouvoir faire des appels récursifs. Nous avons vu pour les fonctions récursives la récursion bornée, qui s'obtient en appliquant une fonction un certain nombre de fois sur un argument donné, moralement en construisant une fonction de la forme $n \mapsto f^n(x)$ pour une certaine fonction f et un certain x. En lambda-calcul, la récursion sera non bornée : on définit simplement une fonction f dans laquelle on peut appliquer f directement. Utilisons un exemple classique de fonction récursive avec la fonction factorielle :

$$\text{fact} := \lambda n.\texttt{if } n == \underline{0} \texttt{ then } \underline{1} \texttt{ else } n \times \text{fact } (n - \underline{1})$$

Cette définition n'est valide qu'en considérant que fact est une variable libre dans ce terme, ce qui n'est pas notre objectif. Pour régler le souci et permettre de faire une définition inductive, il va nous falloir monter d'un cran en abstraction, en considérant au lieu de la fonction factorielle, une fonction qui va « factorialiser » une fonction quelconque : prenant une fonction f, elle définira une fonction qui fera une étape de calcul de fact avant d'appliquer potentiellement f. On va la définir par

$$g := \lambda f.\lambda n.\texttt{if } n == \underline{0} \texttt{ then } \underline{1} \texttt{ else } n \times f\ (n - \underline{1})$$

Le point essentiel est alors le suivant : la fonction fact que l'on veut est exactement un point fixe de g. En effet, c'est une fonction telle que la factorialiser ne changera pas son comportement, car elle est déjà la factorielle. Imaginons qu'on trouve f telle que $g\ f =_\beta f$, alors f se comportera bien comme la factorielle :

$$\begin{aligned} f\ \underline{n} &=_\beta g\ f\ \underline{n} \\ &=_\beta \texttt{if } \underline{n} == \underline{0} \texttt{ then } \underline{1} \texttt{ else } \underline{n} \times f\ (\underline{n} - \underline{1}) \end{aligned}$$

Ce qui nous donne bien le comportement attendu. Justement, il se trouve que tout lambda-terme possède un point fixe, et c'est ce que nous allons prouver.

Définition 6.2.10 (Combinateur de point fixe)**.** *On définit le lambda-terme*

$$Y := \lambda f.(\lambda x.f\ (x\ x))(\lambda x.f\ (x\ x))$$

et le lambda-terme

$$\Theta := (\lambda y.\lambda x.y\ (x\ x\ y))(\lambda y.\lambda x.y\ (x\ x\ y))$$

Ces termes sont des combinateurs de points fixes, ce qui signifie que pour tout lambda-terme M, on a $Y\ M =_\beta M\ (Y\ M)$ et $\Theta\ M =_\beta M\ (\Theta\ M)$

Démonstration. Il nous suffit de faire le calcul :

$$\begin{aligned} Y\ M &\rhd (\lambda x.M\ (x\ x))(\lambda x.M\ (x\ x)) \\ &\rhd M\ ((\lambda x.M\ (x\ x))(\lambda x.M\ (x\ x))) \\ &=_\beta M\ (Y\ M) \\ \Theta\ M &\rhd (\lambda x.M\ (x\ x\ M))(\lambda y.\lambda x.y\ (x\ x\ y)) \\ &\rhd M\ ((\lambda y.\lambda x.y\ (x\ x\ y))\ (\lambda y.\lambda x.y\ (x\ x\ y))\ M) \\ &= M\ (\Theta\ M) \end{aligned}$$

□

Remarque 6.2.2. Le combinateur Θ donne un résultat plus fort : le point fixe $\Theta\ M$ l'est pour la réduction elle-même et non simplement la β-équivalence.

On peut donc définir notre fonction fact :

$$\text{fact} := Y(\lambda f.\lambda n.\texttt{if } n == \underline{0} \texttt{ then } \underline{1} \texttt{ else } n \times f\ (n - \underline{1}))$$

Exercice 6.2.8. Montrer que

$$\forall n \in \mathbb{N}, \text{fact } \underline{n} =_\beta \underline{n!}$$

6.2.4 Fonction récursive en lambda-terme

Nous pouvons maintenant nous atteler à montrer que toute fonction récursive peut être simulée par un lambda-terme.

Définition 6.2.11 (Simulation)**.** *Soit $f : \mathbb{N}^k \to \mathbb{N}$, on dit que f est simulée par le lambda-terme $\underline{f}$ si la propriété suivante est vérifiée :*

$$\forall n_1, \ldots, n_k \in \mathbb{N}, \underline{f}\ (\underline{n_1}, \ldots, \underline{n_k}) =_\beta \underline{f(n_1, \ldots, n_k)}$$

Pour montrer que les fonctions récursives sont simulables, il suffit de montrer les résultats de stabilité qu'on a déjà pu voir par exemple dans le chapitre sur l'arithmétique de Peano.

Proposition 6.2.4. *Les fonctions constantes, les projections et la fonction successeur sont simulables, respectivement par $\lambda x.\underline{n}$ pour la fonction constante n, $\pi_{k,i}$ pour la $i$ème projection de $\mathbb{N}^k \to \mathbb{N}$, et par $\underline{S}$ pour la fonction successeur.*

Démonstration. On a déjà prouvé ces propriétés en amont. □

Proposition 6.2.5. *Si $f_1, \ldots, f_k : \mathbb{N}^m \to \mathbb{N}$ sont des fonctions simulées respectivement par $\underline{f_1}, \ldots, \underline{f_k}$ et $h : \mathbb{N}^k \to \mathbb{N}$ est simulée par $\underline{h}$, alors la fonction $\underline{k} := \lambda x.\underline{h}\ (\underline{f_1}\ x, \ldots, \underline{f_k}\ x)$ simule la fonction*

$$\begin{array}{rccc} k : & \mathbb{N}^m & \longrightarrow & \mathbb{N} \\ & n_1, \ldots, n_m & \longmapsto & h(f_1(n_1, \ldots, n_m), \ldots, f_k(n_1, \ldots, n_m)) \end{array}$$

Démonstration. Faisons le calcul :

$$\begin{aligned} \underline{k}\ (\underline{n_1}, \ldots, \underline{n_m}) &=_\beta \underline{h}\ (\underline{f_1}\ (\underline{n_1}, \ldots, \underline{n_m}), \ldots, \underline{f_k}\ (\underline{n_1}, \ldots, \underline{n_m})) \\ &=_\beta \underline{h}\ (\underline{f_1(n_1, \ldots, n_m)}, \ldots, \underline{f_k(n_1, \ldots, n_m)}) \\ &=_\beta \underline{h(f_1(n_1, \ldots, n_m), \ldots, f_k(n_1, \ldots, n_m))} \\ &=_\beta \underline{k(n_1, \ldots, n_m)} \end{aligned}$$

Respectivement car $\underline{f_i}$ simule f_i et car $\underline{h}$ simule h. □

Proposition 6.2.6 (Récursion primitive)**.** *Si $f : \mathbb{N}^k \to \mathbb{N}$ est simulée par $\underline{f}$, $g : \mathbb{N}^{k+2} \to \mathbb{N}$ est simulée par $\underline{g}$ alors la fonction $\texttt{rec}(f, g) : \mathbb{N}^{k+1} \to \mathbb{N}$ telle que définie dans le chapitre des fonctions récursives est simulée par la fonction*

$$\underline{\texttt{rec}(f, g)} := \Theta(\lambda h.\lambda x.\texttt{if}\ \pi_{k+1,k+1} == \underline{0}\ \texttt{then}\ \underline{f}\ x\ \texttt{else}\ \underline{g}\ \langle\langle \pi_1\ x, \text{pred}\ (\pi_2\ x)\rangle, h\ \langle \pi_1\ x, \text{pred}\ (\pi_2\ x)\rangle\rangle)$$

Démonstration. On vérifie que la fonction ainsi définie représente bien $\texttt{rec}(f, g)$ par λ-récurrence sur son dernier

argument :

$$\begin{aligned}
\underline{\texttt{rec}(f,g)}(\underline{n_1},\dots,\underline{n_k},\underline{0}) &= \texttt{if } \pi_{k+1,k+1}\ (\underline{n_1},\dots,\underline{n_k},\underline{0}) == \underline{0} \texttt{ then } \underline{f}\ (\underline{n_1},\dots,\underline{n_k}) \texttt{ else } [\dots] \\
&=_\beta \texttt{if } \underline{0} == \underline{0} \texttt{ then } \underline{f}\ (\underline{n_1},\dots,\underline{n_k}) \texttt{ else } [\dots] \\
&=_\beta \underline{f}\ (\underline{n_1},\dots,\underline{n_k}) \\
&=_\beta \underline{f(n_1,\dots,n_k)} \\
\underline{\texttt{rec}(f,g)}(\underline{n_1},\dots,\underline{n_k},\underline{S}\ \underline{n}) &=_\beta \underline{g}\ \langle(\underline{n_1},\dots,\underline{n_k}),\text{pred}\ (\underline{S}\ \underline{n})\rangle, \underline{\texttt{rec}(f,g)}\ \langle(\underline{n_1},\dots,\underline{n_k}),\text{pred}\ (\underline{S}\ \underline{n})\rangle \\
&=_\beta \underline{g(\underline{n_1},\dots,\underline{n_k},\underline{n},\underline{\texttt{rec}(f,g)}\ (\underline{n_1},\dots,\underline{n_k},\underline{n}))} \\
&=_\beta \underline{g(\underline{n_1},\dots,\underline{n_k},\underline{n},\underline{\texttt{rec}(f,g)}\ (n_1,\dots,n_k,n))} \\
&=_\beta \underline{g(n_1,\dots,n_k,n,\texttt{rec}(f,g)(n_1,\dots,n_k,n))}
\end{aligned}$$

□

Proposition 6.2.7 (Schéma μ)**.** *Soit $f : \mathbb{N}^{k+1} \to \mathbb{N}$ simulée par $\underline{f}$, alors la fonction $g : x \mapsto \mu(x,f)$ associant à un tuple x le plus petit y tel que $f(x,y) = 0$ s'il existe, est simulée par la fonction $\underline{g} := \lambda x.\Theta(\lambda h.\lambda n.\texttt{if } f\ \langle x,n\rangle == \underline{0} \texttt{ then } n \texttt{ else } h\ \langle x,\underline{S}\ \underline{n}\rangle)\ \underline{0}$*

Démonstration. Soient $n_1,\dots,n_k \in \mathbb{N}$, on suppose qu'il existe y (que l'on prend minimal) tel que $f(n_1,\dots,n_k,y) = 0$. On remarque que pour tout $n < y$, on a pour une fonction h quelconque

$$(\lambda n.\texttt{if } \underline{f}\ \langle(\underline{n_1},\dots,\underline{n_k}),n\rangle == \underline{0} \texttt{ then } n \texttt{ else } h\ \langle(\underline{n_1},\dots,\underline{n_k}),\underline{S}\ \underline{n}\rangle)\ \underline{n} =_\beta h\ (\underline{n_1},\dots,\underline{n_k},\underline{n+1})$$

car par hypothèse de minimalité de y, $f(n_1,\dots,n_k,n) \neq 0$ et $\underline{f}$ simule f, forçant la réduction à entrer dans le cas *else*. De plus, $\underline{y}$ est un point fixe de cette fonction. Notons

$$h := \Theta(\lambda h.\lambda n.\texttt{if } f\ \langle x,n\rangle == \underline{0} \texttt{ then } n \texttt{ else } h\ \langle x,\underline{S}\ n\rangle)$$

pour $x = (\underline{n_1},\dots,\underline{n_k})$. On montre par récurrence finie sur $y - n$ que $h\ \underline{i} =_\beta \underline{y}$:

- Par l'argument précédent, $h\ y =_\beta y$
- Supposons que $h\ i =_\beta y$, alors

$$\begin{aligned}
h\ (\underline{i-1}) &=_\beta \texttt{if } f\ \langle x,\underline{i-1}\rangle == \underline{0} \texttt{ then } n \texttt{ else } h\ \langle x,\underline{S}\ \underline{i-1}\rangle \\
&=_\beta h\ \langle x,\underline{S}\ \underline{i-1}\rangle \\
&=_\beta h\ (x,\underline{i}) \\
&=_\beta \underline{y}
\end{aligned}$$

Donc pour $i = 0$ cela nous donne $h\ \underline{0} =_\beta \underline{y}$, soit

$$\underline{g}\ (\underline{n_1},\dots,\underline{n_k}) =_\beta \underline{g(n_1,\dots,n_k)}$$

□

Théorème 6.2.1 (Simulation)**.** *Pour toute fonction récursive $f : \mathbb{N}^k \to \mathbb{N}$ il existe un lambda-terme $\underline{f}$ tel que*

$$\forall n_1 \in \mathbb{N},\dots,\forall n_k \in \mathbb{N}, \underline{f}\ (\underline{n_1},\dots,\underline{n_k}) =_\beta \underline{f(n_1,\dots,n_k)}$$

Démonstration. En utilisant les propositions précédentes, on sait que les fonctions récursives atomiques sont simulables, et que la récursion, la composée et le schéma μ de fonctions simulables sont simulables, donc toutes les fonctions récursives sont simulables. □

Pour le sens réciproque, comme nous avons montré que l'on peut simuler les machines de Turing par des fonctions récursives, il suffit de montrer que l'on peut simuler le lambda-calcul par une machine de Turing. Nous ne le ferons pas, mais les constructions sur les indices de De Bruijn permettent par exemple de définir de façon purement algorithmique un interpréteur de lambda-calcul. Nous proposons un codage possible $\overline{M}$ du lambda-terme $M \in \Lambda_B$ sur l'alphabet $\Gamma := \{0, 1, 2, 3\}$:

- $\overline{n} = 3^{n+1}$
- $\overline{\lambda M} = 0 \cdot \overline{M}$
- $\overline{(M\ N)} = 1 \cdot \overline{M} \cdot 2 \cdot \overline{N}$

6.3 Propriétés du lambda-calcul

Dans cette section, nous allons nous concentrer sur les propriétés du lambda-calcul en tant que système de réécriture. Nous chercherons à décrire les classes d'égalité $\Lambda_\beta := {}^{\Lambda}\!/_{=_\beta}$ dans un premier temps, en parlant du théorème de Church-Rosser, puis nous parlerons de la règle η et de l'extensionalité.

6.3.1 Propriété de Chuch-Rosser

Le résultat essentiel pour décrire Λ_β est le théorème de Church-Rosser, mais nous allons commencer par donner le formalisme nécessaire à bien appréhender cette notion.

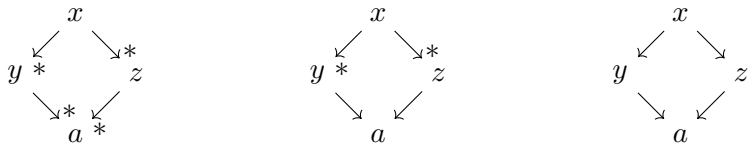
Définition 6.3.1 (Système de réécriture)**.** *On appelle ici système de réécriture un couple $(E, \to)$ où $\to \subseteq E \times E$.*

Le but ici sera évidemment de considérer le système de réécriture $(\Lambda, \rhd)$. Plus précisément, on veut montrer la propriété de confluence de ce système de réécriture :

Définition 6.3.2 (Confluence, confluence locale)**.** *On dit qu'un système de réécriture $(E, \to)$ est confluent si pour tous $x, y, z \in E$, tels que $y \,{}^*\!\leftarrow x \to^* z$ il existe $a \in E$ tel que $y \to^* a$ et $z \to^* a$.*

On dit qu'il est localement si pour tous $x, y, z \in E$ tels que $y \leftarrow x \to z$, il existe $a \in E$ tel que $y \to^* a$ et $z \to^* a$.

On dit qu'il a la propriété du diamant si pour tous $x, y, z \in E$ tels que $y \leftarrow x \to z$ il existe $a \in E$ tel que $y \to a \leftarrow z$.



Confluence Confluence locale Propriété du diamant

Figure 6.2 – Illustration des différentes propriétés

Une propriété équivalente à la confluence, dont la formulation nous intéresse plus, est celle de Church-Rosser :

Définition 6.3.3 (Church-Rosser)**.** *On dit qu'un système de réécriture $(E, \to)$ a la propriété de Church-Rosser si pour tous $x, y \in E$ tels que $x \leftrightarrow^* y$ il existe $z \in E$ tel que $x \to^* z$ et $y \to^* z$ où $\leftrightarrow^*$ est la plus petite relation d'équivalence contenant $\to$.*

Proposition 6.3.1. *$(E, \to)$ a la propriété de Church-Rosser si et seulement si c'est un système confluent.*

Démonstration. Si $(E, \to)$ a la propriété de Church-Rosser, alors pour $y \,{}^*\!\leftarrow x \to^* z$ on peut directement déduire que $y \leftrightarrow^* z$ donc on trouve $a \in E$ tel que $y \to^* a$ et $z \to^* a$.

Réciproquement, si $(E, \to)$ est confluent, soient $x, y \in E$ tels que $x \leftrightarrow^* y$. Par définition de $\leftrightarrow^*$, on trouve une suite $(x_i)_{i=1,\dots,n}$ finie telle que $x_0 = x, x_n = y$ et pour tout $i = 1, \dots, n-1$, $x_i \to^* x_{i+1}$ ou $x_{i+1} \to^* x_i$. On va montrer par récurrence sur ce n (car toute telle suite est finie) qu'il existe $a \in E$ tel que $x \to^* a$ et $y \to^* a$:

- si la suite est (x), alors on a $x \,{}^*\!\leftarrow x \to^* x$ d'où l'existence de a par confluence.
- supposons que pour toute suite de taille n telle que décrite plus haut, il existe un élément a tel que $x \to^* a$ et $y \to^* a$ pour x et y les deux extrémités de la suite. Soit z tel que $y \to^* z$, alors cela signifie que $a \,{}^*\!\leftarrow y \to^* z$ donc on trouve par confluence $b \in E$ tel que $a \to^* b$ et $z \to^* b$, et comme $x \to^* a$ la transitivité nous donne $x \to^* b$. Soit z tel que $z \to^* y$, on peut donc écrire $a \,{}^*\!\leftarrow z \to^* z$ par transitivite de $z \to^* y$ et $y \to^* a$, donc on trouve par confluence b tel que $z \to^* b$ et $a \to^* b$, ce qui comme précédemment permet de conclure. Donc pour toute suite $(x_i)_{i=1,\dots,n+1}$ il existe un terme $a \in E$ tel que $x_1 \to^* a$ et $x_{n+1} \to^* a$.

On a donc trouvé pour $x \leftrightarrow^* y$ un $a \in E$ tel que $x \to^* a$ et $y \to^* a$. □

Enfin, la propriété du diamant est plus forte que la confluence :

Proposition 6.3.2. *Si $(E, \to)$ a la propriété du diamant, alors $(E, \to)$ est confluente.*

Démonstration. Étant donnés $x, y, z \in E$ tels que $x \to^* y$ et $x \to^* z$, on peut trouver deux chemins $(y_i)_{i=0,\dots,n}$ et $(z_i)_{i=0,\dots,m}$ respectivement de x à y et de x à z avec $y_i \to y_{i+1}$ et $z_i \to z_{i+1}$. On montre par récurrence sur $\max(n, m)$ qu'il existe $a \in E$ tel que $y \to^m a$ et $z \to^n a$:

- Si $\max(n, m) = 0$ alors $y = x = z$. En prenant $a = x$ on a le résultat.
- Si $n = m = 1$ alors la propriété du diamant nous donne le résultat.
- Si $n = 0$ ou $m = 0$, en prenant respectivement $a = z$ et $a = y$ la propriété est vérifiée.
- Supposons que pour tous x, y, z tels que $x \to^n y$ et $x \to^m z$, si $\max(n, m) \leq k$ alors il existe $a \in E$ tel que $y \to^m a$ et $z \to^n a$. Soient x, y, z tels que $x \to^n y$ et $x \to^m z$ avec $\max(n, m) = k + 1$ et $\min(n, m) \geq 1$. Soient $y' = y_1$ et $z' = z_1$, par propriété du diamant on trouve $b \in E$ tel que $y' \to b$ et $z' \to b$. On sait donc que $y' \to^{m-1} b$ et $z' \to^{n-1} b$, donc on peut appliquer l'hypothèse d'induction à (y', b, y) et (z', b, z) pour trouver respectivement $c \in E$ et $d \in E$ tels que $b \to^{n-1} c, y \to c$ et $b \to^{m-1} d, z \to d$. On applique encore une fois l'hypothèse d'induction mais à (b, c, d) pour trouver $a \in E$ tel que $c \to^{m-1} a, d \to^{n-1} a$. On a alors

$$y \to c \to^{m-1} a \qquad z \to d \to^{n-1} a$$

  d'où $y \to^m a$ et $z \to^n a$

Ainsi par récurrence on a trouvé un résultat qui donne l'existence de $a \in E$ tel que $y \to^* a, z \to^* a$. □

Proposition 6.3.3. *$(E, \to)$ est confluent si et seulement si $(E, \to^*)$ a la propriété du diamant.*

Démonstration. Dire que $(E, \to^*)$ a la propriété du diamant revient à dire que pour $x \to^* y$ et $x \to^* z$, il existe c tel que $y \to^* c$ et $z \to^* c$, ce qui est l'énoncé de la confluence de $\to$. □

Proposition 6.3.4. *Si $(E, \to^*)$ est confluent, alors $(E, \to)$ est confluent aussi.*

Démonstration. Comme $\to^*$ est la plus petite relation réflexive et transitive contenant $\to$, la plus petite relation réflexive et transitive la contenant est elle-même, donc $\to^*$. Comme $\to^* = (\to^*)^*$ est confluent, on en déduit que $\to^*$ a la propriété du diamant, donc que $\to$ est confluent. □

Pour montrer que $(\Lambda, \rhd)$ est confluent, on va d'abord définir une relation intermédiaire entre $\rhd$ et $\rhd^*$ qui va, pour plusieurs réductions parallèles possibles, effectuer un nombre arbitraire de ces réductions.

Définition 6.3.4 (Réduction parallèle). *On définit la relation $\rhd_\parallel$ par les règles suivantes :*

$$\frac{}{x \rhd_\parallel x}\ x \in \mathcal{V} \qquad \frac{M \rhd_\parallel M' \quad N \rhd_\parallel N'}{M\ N \rhd_\parallel M'\ N'} \qquad \frac{M \rhd_\parallel M'}{\lambda x.M \rhd_\parallel \lambda x.M'}$$

$$\frac{M \rhd_\parallel M' \quad N \rhd_\parallel N'}{(\lambda x.M)\ N \rhd_\parallel M'[N'/x]}$$

Donnons les propriétés essentielles de $\rhd_\parallel$:

Proposition 6.3.5. *La relation $\rhd_\parallel$ vérifie les propriétés suivantes :*

- $\rhd_\parallel$ *est réflexive.*
- *Si* $M \rhd N$ *alors* $M \rhd_\parallel N$.
- *Si* $M \rhd_\parallel N$ *alors* $M \rhd^* N$.
- $\rhd_\parallel^* = \rhd^*$.
- *Si* $M \rhd_\parallel M'$ *et* $N \rhd_\parallel N'$ *alors* $M[N/x] \rhd_\parallel M'[N'/x]$.

Démonstration. On montre ces propriétés :

- La relation est réflexive par induction sur un terme M :
  - $x \rhd_\parallel x$
  - si $M \rhd_\parallel M$ et $M \rhd_\parallel N$ alors $M\ N \rhd_\parallel M\ N$
  - si $M \rhd_\parallel M$ alors $\lambda x.M \rhd_\parallel \lambda x.M$

  Donc par induction $M \rhd_\parallel M$ pour tout $M \in \Lambda$.
- Si $M \rhd N$ alors $M \rhd_\parallel N$. En effet, comme $M \rhd_\parallel M$ et $N \rhd_\parallel N$ on en déduit que $(\lambda x.M)\ N \rhd_\parallel M[N/x]$, et les autres règles définissant $\rhd$ sont aussi respectées.
- Si $M \rhd_\parallel N$ alors $M \rhd^* N$. On procède par induction sur $\rhd_\parallel$:
  - comme $\rhd^*$ est transitive, la première règle fonctionne.
  - si $M \rhd^* M'$ et $N \rhd^* N'$, alors dans un premier temps $M\ N \rhd^* M\ N'$ puis $M\ N' \rhd^* M'\ N'$ donc $M\ N \rhd^* M'\ N'$.
  - si $M \rhd^* M'$ alors $\lambda x.M \rhd^* \lambda x.M'$
  - si $M \rhd^* M'$ et $N \rhd^* N'$ alors $(\lambda x.M)\ N \rhd^* (\lambda x.M')N$ puis $(\lambda x.M')\ N \rhd^* (\lambda x.M')\ N'$ et enfin $(\lambda x.M')\ N' \rhd M'[N'/x]$, d'où $(\lambda x.M)\ N \rhd^* M'[N'/x]$

  Donc par induction si $M \rhd_\parallel N$ alors $M \rhd^* N$.
- $\rhd_\parallel^* = \rhd^*$ puisque $\rhd \subseteq \rhd_\parallel \subseteq \rhd^*$.
- Si $M \rhd_\parallel M'$ et $N \rhd_\parallel N'$ alors $M[N/x] \rhd_\parallel M'[N'/x]$. On raisonne par induction sur $M \rhd_\parallel M'$:
  - si $M = M' = x$ alors $M[N/x] = N \rhd_\parallel N' = M'[N'/x]$. Si $M = M' = y$ où $y \neq x$ alors $M[N/x] = y \rhd_\parallel y = M'[N/x]$.
  - si $M = P\ Q$ et $M = P'\ Q'$ avec $P[N/x] \rhd_\parallel P'[N'/x], Q[N/x] \rhd_\parallel Q'[N'/x]$ alors $M[N/x] = P[N/x]\ Q[N/x] \rhd_\parallel P'[N'/x]\ Q'[N'/x] = M'[N'/x]$.
  - si $M = \lambda y.P$ et $M' = \lambda y.P'$ avec $P[N/x] \rhd_\parallel P'[N'/x]$, alors $M[N/x] = \lambda y.P[N/x] \rhd_\parallel \lambda y.P'[N'/x] = M'[N'/x]$.
  - si $M = (\lambda y.P)\ Q$ et $M' = P'[Q'/x]$ où $P[N/x] \rhd_\parallel P'[N'/x]$ et $Q[N/x] \rhd_\parallel Q'[N'/x]$, alors $M[N/x] = (\lambda y.P[N/x])\ Q[N/x] \rhd_\parallel P'[N'/x][Q'[N'/x]/y] = P'[Q'/y][N'/x] = M'[N'/x]$.

  Donc par induction, $M[N/x] \rhd_\parallel M'[N'/x]$.

□

Définissons enfin la réduction maximale en un pas, pour un terme donné, qui nous donnera un terme vers lequel réduire deux termes issus d'un même terme.

Définition 6.3.5 (Réduction maximal en un pas)**.** *On définit $M^\bullet$ par induction :*

- *si $M = x$ alors $M^\bullet = x$*
- *si $M = \lambda x.M'$ alors $M^\bullet = \lambda x.M'^\bullet$*
- *si $M = (\lambda x.M')\ N'$ alors $M^\bullet = M'^\bullet[N'^\bullet/x]$*
- *si $M = M'\ N'$ où M' n'est pas une λ-abstraction, alors $M^\bullet = M'^\bullet\ N'^\bullet$.*

On peut voir $M^\bullet$ comme le terme M auquel on a appliqué le plus de réductions parallèles possibles. Il est donc naturel de s'attendre à ce que si $M \rhd_\parallel N$, donc en n'effectuant qu'une partie de ces réductions, on puisse poursuivre la réduction parallèle et avoir donc $N \rhd_\parallel M^\bullet$.

Proposition 6.3.6. *Si $M \rhd_\parallel M'$ alors $M' \rhd_\parallel M^\bullet$.*

Démonstration. On va prouver ce résultat par induction sur $M \rhd_\parallel M'$:

- Si $M = M' = x$ alors $M^\bullet = x$ et $M' \rhd_\parallel M^\bullet$.
- Si $M = N\ P$ et $M' = N'\ P'$ où $N' \rhd_\parallel N^\bullet$ et $P' \rhd_\parallel P^\bullet$, alors dans le cas où N n'est pas une λ-abstraction, cela nous donne directement $N'\ P' \rhd_\parallel (N\ P)^\bullet$. Dans le cas où N est une λ-abstraction, on pose $N = \lambda x.Q$, et $(N\ P)^\bullet = Q^\bullet[P^\bullet/x]$, et comme $N = \lambda x.Q$, par inversion (en considérant toutes les règles d'induction on ne trouve qu'une règle permettant $N \rhd_\parallel N'$ dans ces conditions) on déduit que $Q' \rhd_\parallel Q^\bullet$ pour $N' = \lambda x.Q'$. Cela signifie donc $Q' \rhd_\parallel Q^\bullet$ et $P' \rhd_\parallel P^\bullet$, donc on en déduit que $(\lambda x.Q')\ P' \rhd_\parallel Q^\bullet[P^\bullet/x]$, c'est-à-dire $N'\ P' \rhd_\parallel (N\ P)^\bullet$.
- si $M = \lambda x.N$ et $M' = \lambda x.N'$ avec $N' \rhd_\parallel N^\bullet$, alors $M^\bullet = \lambda x.N^\bullet$ donc $M' \rhd_\parallel M^\bullet$.
- si $M = (\lambda x.N)\ P$ et $M' = N'[P'/x]$ avec $N' \rhd_\parallel N^\bullet$ et $P' \rhd_\parallel P^\bullet$, alors $M^\bullet = N^\bullet[P^\bullet/x]$ et grâce à une propriété précédemment montrée, on en déduit que $N'[P'/x] \rhd_\parallel N^\bullet[P^\bullet/x]$ donc que $M' \rhd_\parallel M$.

Le résultat découle donc par induction. □

On peut alors prouver que $\rhd_\parallel$ a la propriété du diamant :

Proposition 6.3.7. *$\rhd_\parallel$ a la propriété du diamant.*

Démonstration. Si l'on considère $M \rhd_\parallel N$ et $M \rhd_\parallel P$ alors par la proposition précédente, $N \rhd_\parallel M^\bullet$ et $P \rhd_\parallel M^\bullet$. □

On peut enfin montrer le résultat essentiel :

Théorème 6.3.1 (Church-Rosser)**.** *Si $M =_\beta N$ alors il existe P tel que $M \rhd^* P$ et $N \rhd^* P$.*

Démonstration. Il nous suffit de montrer que $\rhd$ est confluent. Pour cela, on sait que $\rhd_\parallel$ a la propriété du diamant, donc $\rhd_\parallel^* = \rhd^*$ est confluent : on en déduit donc que $\rhd$ est confluent. □

Les conséquences principales de ce résultat sont les suivantes :

Proposition 6.3.8. *Soit $M \in \Lambda$. Si M est faiblement normalisable, alors il ne possède qu'une forme normale.*

Démonstration. Soient M', M'' deux formes normales de M. Par définition, $M \rhd^* M'$ et $M \rhd^* M''$ donc par confluence $M' \rhd^* N$ et $M'' \rhd^* N$ pour un certain N. De plus comme M' et M'' sont des formes normales, M' et M'' ne peuvent se réduire que vers elles-mêmes, donc $M' = N = M''$. □

Proposition 6.3.9. *Soient M, N deux lambda-termes (faiblement) normalisables. Alors $M =_\beta N$ si et seulement si M et N ont la même forme normale.*

Démonstration. Soit M' la forme normale de M et N' celle de N. Par la propriété de Church-Rosser, on trouve P tel que $M \rhd^* P$ et $N \rhd^* P$, par confluence on trouve Q_1, Q_2 tels que $P \rhd^* Q_1, P \rhd^* Q_2, M' \rhd^* Q_1, N' \rhd^* Q_2$ et par une dernière confluence, on trouve Q tel que $Q_1 \rhd^* Q$ et $Q_2 \rhd^* Q$, donc M' et N' se réduisent vers le même terme, mais cela signifie que $Q = M' = N'$, donc M et N ont la même forme normale.

Réciproquement, si $M \rhd^* P$ et $N \rhd^* P$ alors par définition $M =_\beta N$. □

Proposition 6.3.10. *Si $M =_\beta N$ alors M est faiblement normalisable si et seulement si N l'est.*

Démonstration. Si M est normalisable alors on pose M' sa forme normale. Comme $M =_\beta M'$ et $M =_\beta N$ on peut appliquer la propriété de Church-Rosser à M' et N : on trouve P tel que $M' \rhd^* P$ et $N \rhd^* P$, mais comme M' et une forme normale, $P = M'$ donc $N \rhd^* M'$: N est faiblement normalisable. La réciproque se traite de la même manière. □

On peut donc décrire Λ_β pour les termes faiblement normalisables : les formes normales forment un système de représentants de $\mathcal{N}_\beta = \mathcal{N}/_{=\beta}$ où $\mathcal{N}$ est l'ensemble des lambda-termes faiblement normalisables.

6.3.2 Eta-réduction

Enfin, nous allons considérer l'ajout d'une réduction appelée l'eta-réduction, et sa conséquence principale : l'exentionsalité. L'idée de l'extensionalité est que si nous interprétons un lambda-terme comme une fonction, alors deux lambda-termes qui associent des valeurs identiques devraient pouvoir être identifiés. L'eta-réduction est la version syntaxique de cette définition plus sémantique.

Exemple 6.3.1. Soit $M := \lambda x.y\ x$, on remarque que si l'on prend N quelconque, $M\ N =_\beta y\ N$. Ainsi M et y représentent la même fonction, mais pourtant M est une forme normale, et y aussi, donc M et y sont des éléments différents de Λ_β. La règle η va permettre de définir une relation rendant égaux les termes M et y.

Définition 6.3.6 (Eta-réduction)**.** *On définit l'eta-réduction par la règle suivante :*

$$\frac{}{\lambda x.M\ x \rhd_\eta M}\ x \notin \mathrm{vl}(M)$$

Nous allons introduire un peu de vocabulaire pour décrire les relations sur les lamda-termes.

Définition 6.3.7 (Relation compatible, congruence)**.** *On dit qu'une relation $\mathcal{R} \subseteq \Lambda \times \Lambda$ est compatible si elle vérifie les règles suivantes :*

$$\frac{M \mathcal{R}\ M'}{M\ N \mathcal{R}\ M'\ N} \qquad \frac{N \mathcal{R}\ N'}{M\ N \mathcal{R}\ M\ N'} \qquad \frac{M \mathcal{R}\ M'}{\lambda x.M \mathcal{R}\ \lambda x.M'}$$

On dit que $\mathcal{R}$ est une congruence si c'est une relation d'équivalence compatible.

Exemple 6.3.2. Les relations $=$, $=_\beta$, $\rhd$, $\rhd_{\|}$ et $\rhd^*$ sont compatibles. La relation $=_\beta$ et la relation $=$ sont des congruences.

On définit alors la $\beta - \eta$-réduction de la façon suivante :

Définition 6.3.8 ($\beta\eta$-réduction)**.** *La relation $\rhd_{\beta\eta}$ est la plus petite relation compatible qui contient $\rhd$ et $\rhd_\eta$.*

Définition 6.3.9 ($\beta\eta$-équivalence)**.** *La relation $=_{\beta\eta}$ est la plus petite congruence contenant $\rhd_{\beta\eta}$.*

En parallèle, donnons la notion d'avoir le même comportement :

Définition 6.3.10. *On définit la relation $\cong$, qu'on appellera isomorphisme, comme la plus petite relation d'équivalence vérifiant les règles suivantes :*

$$\frac{M =_\beta N}{M \cong N} \qquad \frac{M\ x \cong N\ x}{M \cong N}\ x \notin \mathrm{vl}(M,N) \qquad \frac{M \cong M' \qquad N \cong N'}{M\ N \cong M'\ N'}$$

Exercice 6.3.1. Montrer que si $M \cong N$ alors $\lambda x.M \cong \lambda x.N$. Montrer que $\cong$ est une congruence.

Donnons un résultat important pour comprendre la motivation de la règle avec $M\ x \cong N\ x$.

Proposition 6.3.11. *$\forall P \in \Lambda, M\ P =_\beta N\ P$ si et seulement si $M\ x =_\beta N\ x$ pour $x \notin \mathrm{vl}(M,N)$, si et seulement si $\forall x \notin \mathrm{vl}(M,N), M\ x =_\beta N\ x$.*

Démonstration. Si $\forall P \in \Lambda, M =_\beta N$ alors en considérant simplement $P = x$ on en déduit le résultat pour tout $x \notin \mathrm{vl}(M)$ et donc en particulier pour un $x \notin \mathrm{vl}(M)$.

Supposons que $\exists x \notin \mathrm{vl}(M,N), M\ x =_\beta N\ x$. Soit $P \in \Lambda$ et $x \notin \mathrm{vl}(M,N)$ comme défini plus tôt. Par hypothèse $M\ x =_\beta N\ x$ donc par compatibilité de $=_\beta$, $(\lambda x.M\ x)\ P =_\beta (\lambda x.N\ x)\ P$ et comme $(\lambda x.M\ x)\ P \rhd M\ P$ et $(\lambda x.N\ x)\ P \rhd N\ P$ on en déduit que $M\ P =_\beta N\ P$. □

Enfin, le résultat essentiel donnant l'intuition de l'extensionalité est le suivant :

Théorème 6.3.2. *Les relations $\cong$ et $=_{\beta\eta}$ sont égales.*

Démonstration. Montrons par induction sur $\cong$ que $\cong \subseteq =_{\beta\eta}$:

- Si $M =_\beta N$ alors $M =_{\beta\eta} N$.
- Si $M\ x =_{\beta\eta} N\ x$ pour $x \notin \mathrm{vl}(M,N)$, alors par compatibilité $\lambda x.M\ x =_{\beta\eta} \lambda x.N\ x$, or $\lambda x.M\ x \rhd_\eta M$ et $\lambda x.N\ x \rhd_\eta N$, donc $M =_{\beta\eta} N$.
- Si $M =_{\beta\eta} M'$ et $N =_{\beta\eta} N'$ alors par compatibilité $M\ N =_{\beta\eta} M\ N'$ puis $M\ N' =_{\beta\eta} M'\ N'$ donc $M\ N =_{\beta\eta} M'\ N'$

Donc $\cong \subseteq =_{\beta\eta}$.

Réciproquement, en faisant une induction sur $M =_{\beta\eta} N$:

- Si $M \rhd N$ alors $M \cong N$.
- Si $M \rhd_\eta N$ alors on a $M = \lambda x.N\ x$ pour $x \notin \mathrm{vl}(M,N)$ et on remarque que pour $y \notin \mathrm{vl}(M,N)$ on a bien $(\lambda x.N\ x)\ y =_\beta N\ y$ donc $(\lambda x.N\ x)\ y \cong N\ y$.
- Par définition $\cong$ est une relation d'équivalence.
- On a montré dans un exercice précédent que $\cong$ est une relation d'équivalence.

Donc $=_{\beta\eta} \subseteq \cong$. □

Exercice 6.3.2. Adapter la preuve du théorème de Church-Rosser au cas de $=_{\beta\eta}$, en adaptant $\rhd_\parallel$ et en montrant que cette relation vérifie les mêmes propriétés mais vis à vis de la $\beta - \eta$ réduction plutôt que la β-réduction.

Chapitre 7

Lambda-calcul simplement typé

Maintenant que nous avons la théorie du lambda-calcul, nous allons introduire la notion de typage. Ce chapitre s'attachera au typage en lui-même et donnera des résultats théoriques motivant le typage (principalement le théorème de normalisation forte) ainsi que les résultats pour utiliser un système de type.

Motivons d'abord l'introduction des types. Nous avons parlé plus tôt de la normalisation, et avons vu qu'un terme tel que $\Omega := (\lambda x.x\ x)(\lambda x.x\ x)$ n'est pas normalisable. Cependant, on peut se demander si un tel terme correspond vraiment à une fonction comme on l'imagine. En effet, le terme $\lambda x.x\ x$ prend un terme et l'applique à lui-même : dans la conception habituelle de fonction, c'est un processus impossible puisque $E \neq F^E$ pour quelque F que ce soit, donc si $f : E \to F$ alors on ne peut pas donner f en argument à f. Il y a donc naturellement une notion d'homogénéité dans notre appréciation des fonctions, et cette notion que vont traduire les types. De plus, une conséquence particulièrement appréciable du fait de forcer l'homogénéité est que tous les termes homogènes en ce sens, sont fortement normalisables.

La théorie des types a d'ailleurs été initialement introduite par Russell pour éviter le paradoxe d'avoir un ensemble de tous les ensembles, qui d'une certaine façon peut se rapprocher de l'élimination des lambda-termes s'appliquant à eux-mêmes. Un type sera donc une étiquette ajoutée à un terme pour indiquer sa nature. On pourra par exemple trouver le type `int` qui étiquette les termes qui représenteront les entiers.

7.1 Types des structures habituelles

Cette section sera séparée en plusieurs parties : nous allons chaque fois renforcer notre grammaire de typage et les termes que l'on peut construire, en partant de la notion la plus simple, celle-ci étant de simplement ajouter des types aux lambda-termes habituels.

7.1.1 Types des fonctions

Commençons par introduire la notion de types. On définit pour cela un ensemble de types de base $\iota \in \mathcal{B}$ qui pourra moralement représenter tous les types habituels concrets, leur donnée exacte n'étant pas pertinente dans l'étude théorique du lambda-calcul simplement typé. On ajoute le constructeur de type $\to$ qui dénote, étant donnés deux types σ et τ, le type des fonctions qui à un argument de type σ associe une sortie de type τ.

Définition 7.1.1 (Grammaire des types)**.** *On définit inductivement l'ensemble $T_\to$ des types par la grammaire suivante :*

$$\sigma, \tau ::= \iota \mid \sigma \to \tau$$

où $\sigma, \tau \in T_\to$ et $\iota \in \mathcal{B}$.

Remarque 7.1.1. On utilisera une associativité à droite de la flèche : $\tau \to \tau' \to \sigma$ se lira $(\tau \to \tau') \to \sigma$. Cela se justifie de la même manière que notre convention d'associativité à gauche de l'application, car une fonction de la forme $f : (x, y) \mapsto f(x, y)$, une fois curryfiée, donnera une fonction de la forme $f : x \mapsto (y \mapsto f(x, y))$ qui,

pour x de type τ, y de type τ' et $f(x,y)$ de type σ, a le type $\tau \to (\tau' \to \sigma)$. Comme cette situation arrive très fréquemment, on simplifie l'écriture pour n'avoir qu'à écrire $\tau \to \tau' \to \sigma$.

Deux choix sont possibles pour définir un lambda-calcul typé : considérer les termes comme des lambda-termes classiques et y ajouter des annotations de type, ou bien construire des termes déjà typés en partie. Nous opterons pour le second, car il nous semble plus proche de la philosophie de la théorie des types, où les lambda-termes dépendent des types.

Définition 7.1.2 (Lambda-terme typé)**.** *On définit l'ensemble des pré-lambda-termes typés $\Lambda_0^{\to}$ par la grammaire suivante :*

$$M, N ::= x^\tau \mid \lambda x^\tau.M \mid (M\ N)$$

où $x \in \mathcal{V}, \tau \in T, M, N \in \Lambda_0^{\to}$.

Le typage est alors l'action d'associer à un lambda-terme donné, un type. Remarquons que le processus se fait en deux temps : on commence par définir le lambda-terme, avant de définir son type. Cependant, comme nous le prouverons plus tard, la procédure qui à un lambda-terme associe son type est univoque. Nous utiliserons une notation déjà vue en logique, qui est celle des séquents. Pour pouvoir l'utiliser, nous allons définir la notion d'environnement de typage avant de définir les jugements de typage.

Dans $\Lambda_0^{\to}$, nous avons ajouté des annotations de types aux variables et aux abstractions, mais nous noterons rarement les annotations sur les variables. Pour les abstractions, on pourra aussi noter $\lambda(x : \tau).$ à la place de $\lambda x^\tau.$ mais le deuxième étant plus court, il sera privilégié ici.

Définition 7.1.3 (Environnement de typage)**.** *Un environnement de typage est une liste $\Gamma := (\mathcal{V} \times T)^*$. On notera en général la liste dans le sens inverse du sens habituel : l'élément de tête sera à droite. L'intuition d'un environnement Γ est d'associer à un nombre fini de variables un certain type. Plutôt que $[(x_1, \sigma_1), \dots, (x_n, \sigma_n)]$ nous écrirons $x_1 : \sigma_1, \dots, x_n : \sigma : n$ pour écrire la liste, et $\Gamma, x : \sigma$ pour signifie la liste à laquelle on ajoute (x, σ) en tête de liste.*

Définition 7.1.4 (Jugement de typage)**.** *On définit par induction une relation $\vdash \subseteq (\mathcal{V} \times T)^* \times (\Lambda_0^{\to} \times T)$, nommée relation de typage, qu'on notera $\Gamma \vdash M : \sigma$ pour $\vdash (\Gamma, (M, \sigma))$, par la relation suivante :*

$$\frac{}{\Gamma, x : \tau \vdash x^\tau : \tau} \qquad \frac{\Gamma \vdash M : \tau}{\Gamma, x : \sigma \vdash M : \tau}\ x \notin \mathrm{vl}(M)$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x^\sigma.M : \sigma \to \tau} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash (M\ N) : \tau}$$

On appelle jugement de typage une instance de cette relation. Si Γ est la liste vide, on notera $\vdash M : \tau$ pour $\varnothing \vdash M : \tau$.

Exemple 7.1.1. Nous allons montrer que l'on peut typer le terme $\lambda f^{\iota\to\iota}.\lambda x^\iota.f\ (f\ x)$ dans l'environnement vide :

$$\dfrac{\dfrac{\dfrac{\dfrac{f : \iota \to \iota \vdash f : \iota \to \iota}{f : \iota \to \iota, x : \iota \vdash f : \iota \to \iota} \quad \dfrac{\dfrac{f : \iota \to \iota \vdash f : \iota \to \iota}{f : \iota \to \iota, x : \iota \vdash f : \iota \to \iota} \quad f : \iota \to \iota, x : \iota \vdash x^\iota : \iota}{f : \iota \to \iota, x : \iota \vdash f\ x^\iota : \iota}}{f : \iota \to \iota, x : \iota \vdash f\ (f\ x^\iota) : \iota}}{f : \iota \to \iota \vdash \lambda x^\iota.f\ (f\ x^\iota) : \iota \to \iota}}{\vdash \lambda f^{\iota\to\iota}.\lambda x^\iota.f\ (f\ x^\iota) : (\iota \to \iota) \to \iota \to \iota}$$

Nous allons maintenant montrer des résultats de structure sur la relation de typage. Ceux-ci seront essentiels pour pouvoir montrer des résultats de façon rapide.

Proposition 7.1.1 (Unicité du typage)**.** *Soient $M \in \Lambda_0^{\to}$ et Γ, Γ' deux environnements tels que $\Gamma \vdash M : \tau$ et $\Gamma' \vdash M : \sigma$, alors $\tau = \sigma$.*

Démonstration. On procède par induction sur $\Gamma \vdash M : \tau$:

- Si $\Gamma \vdash x^{\tau'} : \tau$ où $\Gamma = \Delta, x : \tau$, alors on raisonne par induction sur $\Gamma' \vdash x^{\tau'} : \sigma$:
  - Si $\Gamma' \vdash x^{\tau'} : \sigma$ où $\Gamma' = \Delta', x : \sigma$ alors par construction $\sigma = \tau'$ et de même $\tau = \tau'$ donc au total $\tau = \sigma$.
  - Si $\Delta' \vdash x^{\tau'} : \sigma'$ où $\Delta = \Delta', y : \tau'', y \notin \mathrm{vl}(M)$ et $\sigma' = \tau$ alors $\Delta \vdash x^{\tau'} : \sigma'$ et $\sigma = \sigma'$ par hypothèse d'induction donc $\sigma = \tau = \sigma'$.
  - Dans les deux autres cas M ne peut pas être mis sous la forme voulue, donc la prémisse est fausse, menant à une conclusion vraie peu importe la conclusion.
- Si $\Delta \vdash M : \tau'$ où $\Gamma = \Delta, x : \tau'$ avec $x \notin \mathrm{vl}(M)$ et $\tau' = \sigma$ alors $\Gamma \vdash M : \tau$ et $\tau = \tau' = \sigma$.
- Si $M = \lambda x^{\tau'}.N$ et que $\Gamma, x : \tau' \vdash N : \tau''$ où $\tau = \tau' \to \tau''$, alors on raisonne par induction sur $\Gamma' \vdash M$ en éliminant les deux cas non pertinents :
  - Si $\Delta' \vdash M : \sigma$ avec $\Delta', y : \sigma' = \Delta, y \notin \mathrm{vl}(N)$ et $\sigma = \tau$ alors on en déduit que $\tau = \sigma$ et $\Delta \vdash M : \sigma$.
  - Si $\Gamma', x : \tau' \vdash N : \sigma'$ alors par hypothèse d'induction $\sigma' = \tau$ donc $\Gamma' \vdash \lambda x^{\tau'}.N : \tau' \to \sigma'$ et $\Gamma \vdash \lambda x^{\tau'}.N : \tau' \to \sigma'$ donc $\tau = \sigma$.
- Si $M = P\ Q$ alors on raisonne de façon analogue au cas précédent pour montrer que $\tau = \sigma$.

Donc $\tau = \sigma$. □

Exercice 7.1.1. Montrer le lemme de structure suivant, pour $x \neq y$: si $\Gamma, x : \tau, y : \tau', \Gamma' \vdash M : \sigma$ alors $\Gamma, y : \tau', x : \tau, \Gamma' \vdash M : \sigma$.

Exercice 7.1.2. Montrer le lemme de structure suivant : si $x \notin \mathrm{vl}(M)$ et que $\Gamma \vdash M : \tau$ alors $\Gamma' \vdash M : \tau$ où Γ' est l'environnement Γ où l'on a retiré la dernière occurrence de x.

Définition 7.1.5 (Terme typable)**.** *On dit qu'un terme M est typable s'il existe un environnement Γ et type τ tels que $\Gamma \vdash M : \tau$. On dit que M est typable dans l'environnement Γ s'il existe un type τ tel que $\Gamma \vdash M : \tau$. On dit que τ est habité s'il existe un lambda-terme M tel que $\vdash M : \tau$.*

Remarque 7.1.2. Si M est typable alors le type associé est unique d'après une propriété précédente.

Nous voulons alors quotienter les lambda-termes par α-équivalence pour définir $\Lambda^{\to} = \Lambda_0^{\to}/_{=_\alpha}$, mais il faut pour cela définir la substitution sur les termes typés, qui prend en compte la cohérence des types.

Proposition 7.1.2 (Substitution typée)**.** *Soient $M, N \in \Lambda_0^{\to}$, $x \in \mathcal{V}$ et Γ un environnement tel que*

$$\Gamma \vdash M : \tau \qquad \Gamma \vdash N : \sigma \qquad \Gamma \vdash x : \sigma$$

alors $\Gamma \vdash M[N/x] : \tau$ où $M[N/x]$ est défini comme pour une substitution non typée.

Démonstration. On procède par induction sur $M[N/x]$:

- Si $M = x$, alors $\Gamma \vdash x : \tau$ donc par l'exercice précédent $\tau = \sigma$, d'où $\Gamma \vdash N : \tau$.
- Si $M = y$, alors $\Gamma \vdash y : \tau$.
- Si $M = \lambda y^{\tau'}.M'$ pour $y \notin \mathrm{vl}(N)$, et par hypothèse d'induction $\Gamma, y : \tau' \vdash M'[N/x] : \tau''$ et $\tau = \tau' \to \tau''$, alors en appliquant la règle de typage correspondante on en déduit que $\Gamma \vdash \lambda y.M' : \tau' \to \tau''$ d'où $\Gamma \vdash M[N/x] : \tau$.
- Si $M = (P\ Q)$, $\Gamma \vdash P[N/x] : \tau' \to \tau$ et $\Gamma \vdash Q[N/x] : \tau'$ alors on en déduit que $\Gamma \vdash (P[N/x]\ Q[N/x]) : \tau$ donc par définition de $[N/x]$ cela signifie $\Gamma \vdash M[N/x] : \tau$.

Donc par induction $\Gamma \vdash M[N/x] : \tau$. □

Exercice 7.1.3. Montrer que si $\Gamma \vdash (\lambda x.M) : \tau$ alors $\Gamma' \vdash (\lambda y.M[y/x]) : \tau$ où $y \notin \mathrm{vl}(M) \cup \Gamma$ et Γ' est l'environnement Γ dans lequel on a remplacé chaque couple de la forme (x, σ) par (y, σ). En déduire que $\lambda x.M$ est typable si et seulement si $\lambda y.M[y/x]$ l'est aussi.

Définition 7.1.6 (α-équivalence)**.** *On définit notre α-équivalence $=_\alpha$ comme la plus petite congruence vérifiant la règle suivante :*

$$\frac{}{\lambda x.M =_\alpha \lambda y.M[y/x]}\ y \notin \mathrm{vl}(M)$$

On pose alors $\Lambda^{\to} = {\Lambda_0^{\to}}/_{=_\alpha}$ l'ensemble des termes typés.

Exercice 7.1.4. Montrer que si $M =_\alpha N$ et $\Gamma \vdash M : \tau$ alors $\Gamma \vdash N : \tau$ et donc que le typage est bien défini sur $\Lambda^{\to}$.

Théorème 7.1.1 (Préservation du typage)**.** *En adaptant la réduction $\rhd$ aux lambda-termes typés de façon évidente, si $\Gamma \vdash M : \tau$ et $M \rhd N$ alors $\Gamma \vdash N : \tau$.*

Démonstration. On raisonne par induction sur $M \rhd N$:

- Si $\Gamma \vdash M : \tau$ et que $M = (\lambda x^\sigma.P)Q$ avec $N = Q[P/x]$ alors on peut prendre $x \notin \mathrm{vl}(Q)$ et faire une inversion sur le typage pour trouver que $\Gamma \vdash Q : \sigma$ et $\Gamma, x : \sigma \vdash P : \tau$, mais cela signifie aussi que $\Gamma, x : \sigma \vdash Q : \sigma$ et $\Gamma, x : \sigma \vdash x : \sigma$ donc $\Gamma, x : \sigma \vdash P[Q/x] : \tau$. De plus comme $x \notin \mathrm{vl}(P)[Q/x]$, on en déduit que $\Gamma \vdash P[Q/x] : \tau$.
- Les autres cas se déroulent directement en appliquant les hypothèses d'induction.

□

On en déduit que $\Lambda_\beta^{\to} = {\Lambda^{\to}}/_{=_\beta}$ est compatible avec la relation de typage.

Exercice 7.1.5. Vérifier que le théorème de Church-Rosser s'applique encore pour $\Lambda^{\to}$.

Exercice 7.1.6. Vérifier que l'ajout de la règle η a les mêmes propriétés que pour Λ.

7.1.2 Type des paires

On définit ici un nouveau lambda-calcul, nommé $\Lambda^{\to\times 1}$ qui permet de considérer des paires. En effet, contrairement au lambda-calcul non typé, il n'est pas possible en lambda-calcul de coder directement les paires et les projections. L'argument intuitif est que le codage $\langle M, N\rangle := \lambda p.p\ M\ N$ quantifie p sur tous les types possibles et il faudrait donc pouvoir définir une fonction prenant un type quelconque. Nous verrons que cela est possible si on autorise des quantifications de second ordre, mais le typage simple est juste un typage qui n'utilise pas ces quantifications. On ajoute donc en parallèle les types produits, que l'on peut voir comme des analogues des produits cartésiens, le type unit qui est un type contenant un unique élément, et les fonctions permettant à ces types d'être construits et utilisés.

Définition 7.1.7 (Types produit)**.** *On définit un nouvel ensemble de types, que l'on notera $T_{\to\times 1}$, par la grammaire suivante :*

$$\sigma, \tau ::= \iota \mid \texttt{unit} \mid \sigma \to \tau \mid \sigma \times \tau$$

où $\iota \in \mathcal{B}$, `unit` est une constante de type, et $\sigma, \tau \in T_{\to\times 1}$.

Remarque 7.1.3. On note $\times$ prioritaire sur $\to$, donc $\sigma \times \tau \to \kappa$ se lit $(\sigma \times \tau) \to \kappa$.

Définition 7.1.8 ($\Lambda_0^{\to\times 1}$)**.** *On définit l'ensemble $\Lambda_0^{\to\times 1}$ par la grammaire enrichie sur celle de $\Lambda^\to$ suivante :*

$$M, N ::= \ldots \mid \langle M, N\rangle \mid \pi_1\ M \mid \pi_2\ M \mid \langle\rangle$$

On définit $=_\alpha$ de la même façon que pour $\Lambda_0^\to$ et on définit alors $\Lambda^{\to\times 1} = \Lambda_0^{\to\times 1}/_{=_\alpha}$.

Définition 7.1.9 (Typage dans $\Lambda^{\to\times 1}$)**.** *On définit la relation de typage $\vdash$ en ajoutant des règles à la relation de typage $\vdash$ dans $\Lambda^\to$:*

$$\frac{\Gamma \vdash M : \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash \langle M, N\rangle : \tau \times \sigma} \qquad \frac{\Gamma \vdash M : \tau \times \sigma}{\Gamma \vdash \pi_1\ M : \tau} \qquad \frac{\Gamma \vdash M : \tau \times \sigma}{\Gamma \vdash \pi_2\ M : \sigma}$$

$$\frac{}{\Gamma \vdash \langle\rangle : \mathbf{unit}}$$

Exercice 7.1.7. Montrer que la relation de typage est bien définie sur $\Lambda^{\to\times 1}$ i.e. que si $M =_\alpha N$ et $\Gamma \vdash M : \tau$ alors $\Gamma \vdash N : \tau$. *Indication : on généralisera d'abord les propriétés de structure nécessaires de $\Lambda_0^\to$.*

Définition 7.1.10 (β-réduction)**.** *On définit aussi la relation $\rhd \subseteq \Lambda^{\to\times 1} \times \Lambda^{\to\times 1}$ comme la plus petite relation compatible contenant les règles suivantes :*

$$\frac{}{(\lambda x^\tau.M)\ N \rhd M[N/x]} \qquad \frac{}{\pi_1\ \langle M, N\rangle \rhd M} \qquad \frac{}{\pi_2\ \langle M, N\rangle \rhd N}$$

Et $=_\beta$ comme la congruence associée.

Remarque 7.1.4. Comme nous avons augmenté le nombre de constructeurs, les relations que vérifie une congruence ou une relation compatible sont aussi plus nombreuses :

$$\frac{M\ \mathcal{R}\ M'}{\langle M, N\rangle\ \mathcal{R}\ \langle M', N\rangle} \qquad \frac{N\ \mathcal{R}\ N'}{\langle M, N\rangle\ \mathcal{R}\ \langle M, N'\rangle}$$

$$\frac{M\ \mathcal{R}\ M'}{\pi_1\ M\ \mathcal{R}\ \pi_1\ M'} \qquad \frac{M\ \mathcal{R}\ M'}{\pi_2\ M\ \mathcal{R}\ \pi_2\ M'}$$

Exercice 7.1.8. Vérifier que la préservation du typage est encore vraie.

Exercice 7.1.9. Vérifier que $(\Lambda^{\to\times 1}, \rhd)$ vérifie bien la propriété de Church-Rosser.

Définition 7.1.11 (η-réduction)**.** *On définit la relation $\rhd_{\beta\eta} \subseteq \Lambda^{\to\times 1} \times \Lambda^{\to\times 1}$ comme la plus petite relation compatible contenant $\rhd$ et les règles suivantes :*

$$\frac{}{\lambda x^\tau.M\ x \rhd_{\beta\eta} M}\ x \notin \mathrm{vl}(M) \qquad \frac{}{\langle \pi_1\ M, \pi_2\ M\rangle \rhd_{\beta\eta} M} \qquad \frac{}{M \rhd_{\beta\eta} \langle\rangle}\ M : \mathbf{unit}$$

Et $=_{\beta\eta}$ comme la congruence associée.

Remarque 7.1.5. La propriété de Church-Rosser ne tient pas pour $(\Lambda^{\to\times 1}, \rhd_{\beta\eta})$: en effet, si on prend $x : \tau \times \texttt{unit}$ et qu'on considère $M := \lambda\pi_1\ x, \pi_2\ x\rangle$ alors on peut au choix réduire M en x ou bien en $\langle\pi_1\ x, \langle\rangle\rangle$ qui sont deux formes normales.

Exercice 7.1.10. Définir la relation $\cong$ sur $\Lambda^{\to\times 1}$ et montrer qu'elle est égale à $=_{\beta\eta}$.

7.1.3 Types des unions disjointes

Nous allons enfin ajouter un constructeur de type analogue aux sommes d'ensembles, c'est-à-dire aux unions disjointes. En théorie des ensembles, $E + F$, que l'on note aussi $E \sqcup F$, est défini par $E + F := \{(x, 0) \mid x \in E\} \cup \{(y, 1) \mid y \in F\}$, et on possède alors les fonctions

$$\begin{array}{rccc} \kappa_1 : & E & \longrightarrow & E + F \\ & x & \longmapsto & (x, 0) \end{array} \qquad \begin{array}{rccc} \kappa_2 : & F & \longrightarrow & E + F \\ & y & \longmapsto & (y, 1) \end{array}$$

qui sont universelle en ce sens que s'il existe deux fonctions $f : E \to G$ et $g : F \to G$ pour un ensemble G quelconque, alors il existe une unique fonction $[f, g] : E + F \to G$ telle que $[f, g](x, 0) = f(x)$ et $[f, g](y, 1) = g(y)$ pour $x \in E, y \in F$. Cet ensemble de fonctions et cette propriété universelle caractérisent presque l'ensemble $E + F$ (en fait à bijection près) mais cela suffit largement pour décrire le comportement de l'objet $E + F$. D'un point de vue du lambda-calcul, cette définition utilisant principalement des fonctions est privilégiée à l'approche ensembliste de base car elle permet de définir facilement les constructeurs dont nous avons besoin : un constructeur qui se comporte comme κ_1, un autre comme κ_2 et un constructeur de la forme $[-, -]$ qui permet de construire des fonctions $E + F \to G$ à partir d'une fonction $E \to G$ et une fonction $F \to G$. On ajoute enfin un type vide, correspondant à l'ensemble $\varnothing$. En théorie des ensembles on possède une fonction $\varnothing \to E$ pour tout ensemble E, donc il faut introduire aussi une telle fonction.

Définition 7.1.12 (Types somme)**.** *On définit $T_{\to\times 1+0}$ par la grammaire suivante :*

$$\sigma, \tau ::= \iota \mid \texttt{unit} \mid \texttt{void} \mid \sigma \to \tau \mid \sigma \times \tau \mid \sigma + \tau$$

où $\iota \in \mathcal{B}$, $\texttt{unit}$ et $\texttt{void}$ sont des constantes de types et $\sigma, \tau \in T_{\to\times 1+0}$.

Remarque 7.1.6. On donne une priorité à $+$ intermédiaire entre $\times$ et $\to$: $\tau \times \sigma + \kappa$ se lit $(\tau \times \sigma) + \kappa$ et $\tau + \sigma \to \kappa$ se lit $(\tau + \sigma) \to \kappa$.

Définition 7.1.13 ($\Lambda^{\to\times 1+0}$)**.** *En quotientant par α-équivalence, on définit $\Lambda^{\to\times 1+0}$ en enrichissant la grammaire de $\Lambda^{\to\times 1}$:*

$$M, N, P ::= \ldots \Big| \kappa_1\ M \Big| \kappa_2\ N \Big| \delta\ (x \mapsto M \mid y \mapsto N)\ P \Big| \delta_\bot\ M$$

Remarque 7.1.7. La gestion des variables libre doit être actualisée par rapport au fait que x est liée dans $\delta\ (x \mapsto M \mid y \mapsto N)\ P$, mais cette adaptation est un processus administratif. De plus le constructeur peut prendre des variables différentes, par exemple $\delta\ (a \mapsto M \mid b \mapsto N)\ P$ est aussi valide, et l'identification dans $=_\alpha$ a donc une règle en plus.

Définition 7.1.14 (Typage dans $\Lambda^{\to\times 1+0}$)**.** *On définit $\vdash$ sur $\Lambda^{\to\times 1+0}$ en enrichissant les règles précédentes :*

$$\frac{\Gamma \vdash M : \tau}{\Gamma \vdash \kappa_1\ M : \tau + \sigma} \qquad \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \kappa_1\ M : \tau + \sigma} \qquad \frac{\Gamma \vdash P : \sigma + \tau \quad \Gamma, x : \sigma \vdash M : \kappa \quad \Gamma, y : \tau \vdash N : \kappa}{\Gamma \vdash \delta\ (x \mapsto M \mid y \mapsto N)\ P : \kappa}$$

$$\frac{\Gamma \vdash M : \texttt{void}}{\Gamma \vdash \delta_\bot\ M : \tau}$$

Remarque 7.1.8. Il n'y a pas de constructeur de type `void`, ce qui est normal pour un type vide. De plus au lieu de considérer une fonction $M : \sigma \to \kappa$ on considère un terme $M : \kappa$ en ajoutant $x : \sigma$ à l'environnement ; s'il est évident que les deux situations sont équivalentes, la deuxième nous permettra une meilleure visualisation de l'isomorphisme de Curry-Howard développé dans un chapitre ultérieur.

Remarque 7.1.9. La règle de typage pour $\delta_\bot$ fait qu'il n'y a plus unicité du type d'une expression. On peut contourner ce problème en définissant plutôt $\delta^\tau_\bot$ où on indique le type d'arrivée de $\delta_\bot$, et en considérant qu'on ne l'écrira jamais, de la même façon qu'on note x une variable et non x^τ.

On ajoute ensuite les réductions.

Définition 7.1.15 (β-relations)**.** *On définit la relation $\rhd$ comme la plus petite relation compatible contenant les règles suivantes :*

$$\frac{}{(\lambda x^\tau.M)\ N \rhd M[N/x]} \qquad \frac{}{\pi_1\ \langle M, N\rangle \rhd M} \qquad \frac{}{\pi_2\ \langle M, N\rangle \rhd N}$$

$$\frac{}{\delta\ (x \mapsto M \mid y \mapsto N)\ (\kappa_1\ P) \rhd M[P/x]} \qquad \frac{}{\delta\ (x \mapsto M \mid y \mapsto N)\ (\kappa_2\ P) \rhd N[P/y]}$$

Et $=_\beta$ est alors la congruence associée.

Remarque 7.1.10. On ajoute encore des relations pour la compatibilité :

$$\frac{M\ \mathcal{R}\ M'}{\kappa_1\ M\ \mathcal{R}\ \kappa_1\ M'} \qquad \frac{M\ \mathcal{R}\ M'}{\kappa_2\ M\ \mathcal{R}\ \kappa_2\ M'} \qquad \frac{M\ \mathcal{R}\ M'}{\delta\ (x \mapsto M \mid y \mapsto N)\ P\ \mathcal{R}\ \delta\ (x \mapsto M' \mid y \mapsto N)\ P}$$

$$\frac{N\ \mathcal{R}\ N'}{\delta\ (x \mapsto M \mid y \mapsto N)\ P\ \mathcal{R}\ \delta\ (x \mapsto M \mid y \mapsto N')\ P} \qquad \frac{P\ \mathcal{R}\ P'}{\delta\ (x \mapsto M \mid y \mapsto N)\ P\ \mathcal{R}\ \delta\ (x \mapsto M \mid y \mapsto N)\ P'}$$

$$\frac{M\ \mathcal{R}\ M'}{\delta_\bot\ M\ \mathcal{R}\ \delta_\bot\ M'}$$

On notera aussi $\rhd_0$ la réduction en surface, qui est la version de $\rhd$ sans la compatibilité, c'est-à-dire n'effectuant une réduction que si celle-ci apparait directement dans le terme entier.

Définition 7.1.16 (β-η-relations)**.** *On définit la relation $\rhd_{\beta\eta}$ comme la plus petite relation compatible contenant $\rhd$ et les règles suivantes :*

$$\frac{}{\lambda x^\tau.M\ x \rhd_{\beta\eta} M}\ x \notin \mathrm{vl}(M) \qquad \frac{}{\langle \pi_1\ M, \pi_2\ M\rangle \rhd_{\beta\eta} M} \qquad \frac{}{M \rhd_{\beta\eta} \langle\rangle}\ M : \texttt{unit}$$

$$\frac{}{\delta\ (x \mapsto M\ (\kappa_1\ x) \mid y \mapsto M\ (\kappa_2\ y))\ P \rhd_{\beta\eta} M\ P}$$

Et $=_{\beta\eta}$ comme la congruence associée.

Exercice 7.1.11 (Booléens)**.** On définit le type $\texttt{bool} := \texttt{unit} + \texttt{unit}$ et $\top_\texttt{bool} := \kappa_1\ \langle\rangle, \bot_\texttt{bool} := \kappa_2\ \langle\rangle$. Définir les opérations booléennes usuelles $\neg, \wedge, \vee$ et vérifier qu'elles se comportent comme attendu. Soit $\tau \in T_{\to\times 1+0}$, définir

une fonction $\texttt{if}\ -\ \texttt{then}\ -\ \texttt{else}\ - : \texttt{bool} \to \tau \to \tau \to \tau$ telle que

$$\texttt{if}\ \top_{\texttt{bool}}\ \texttt{then}\ M\ \texttt{else}\ N =_\beta M$$
$$\texttt{if}\ \bot_{\texttt{bool}}\ \texttt{then}\ M\ \texttt{else}\ N =_\beta N$$

7.2 Normalisation des termes typables

Nous allons voir la propriété essentielle du lambda-calcul simplement typé : tout lambda-terme bien typé est normalisable, faiblement mais aussi fortement. Nous allons dans un premier temps montrer la propriété de la disjonction pour illustrer la technique de preuve qui nous servira ensuite pour la normalisation faible, puis la normalisation forte. Nous nous plaçons dans $\Lambda^{\to\times 1+0}$ mais les résultats fonctionnent aussi en se restreignant respectivement à $\Lambda^{\to\times 1}$ et $\Lambda^{\to}$. Pour la normalisation faible, nous allons introduire la notion de réduction de tête, qui est une réduction plus faible que $\rhd$ mais nous détaillons dans le passage à la normalisation forte comment adapter la normalisation faible pour la réduction de tête à celle pour la β-réduction classique.

7.2.1 Préliminaires

Commençons par donner quelques définitions préliminaires pour faciliter la lecture de nos résultats.

Définition 7.2.1 (Substitution simultanée)**.** *Soit $\sigma : \mathcal{V} \to \Lambda^{\to\times 1+0}$ une fonction partielle. Pour un terme M on donne la définition de la substitution simultanée $M\ \sigma$ par induction sur M, où nous noterons* $\mathrm{vl}(\sigma) = \bigcup_{x\in\mathrm{dom}(\sigma)} \mathrm{vl}(\sigma(x))$ *:*

- *Si $M = x \in \mathrm{dom}(\sigma)$ alors $M\ \sigma = \sigma(x)$.*
- *Si $M = x \notin \mathrm{dom}(\sigma)$ alors $M\ \sigma = M$.*
- *Si $M = \lambda x.N$ où $x \notin \mathrm{dom}(\sigma) \cup \mathrm{vl}(\sigma)$ alors $M\ \sigma = \lambda x.(N\ \sigma)$.*
- *Si $M = N\ P$ alors $M\ \sigma = (N\ \sigma)\ (P\ \sigma)$.*
- *Si $M = \langle\rangle$ alors $M\ \sigma = \langle\rangle$.*
- *Si $M = \langle N, P\rangle$ alors $M\ \sigma = \langle N\ \sigma, P\ \sigma\rangle$.*
- *Si $M = \pi_i\ N$ où $i \in \{1,2\}$ alors $M\ \sigma = \pi_i\ (N\ \sigma)$.*
- *Si $M = \kappa_i\ N$ où $i \in \{1,2\}$ alors $M\ \sigma = \kappa_i\ (N\ \sigma)$.*
- *Si $M = \delta\ (x \mapsto N \mid y \mapsto P)\ Q$ où $x, y \notin \mathrm{dom}(\sigma) \cup \mathrm{vl}(\sigma)$ alors*

$$M\ \sigma = \delta\ (x \mapsto N\ \sigma \mid y \mapsto P\ \sigma)\ (Q\ \sigma)$$

On appelle par abus de langage substitution une fonction partielle $\sigma : \mathcal{V} \to \Lambda^{\to\times 1+0}$.

On notera aussi, par convention, $\sigma[N/x]$ pour désigner la substitution $y \mapsto \sigma(y)$ si $y \neq x$ et $x \mapsto u$, avec $x \notin \mathrm{dom}(\sigma)$. On définit aussi pour des variables $x_1, \ldots, x_n$ et des termes $M, M_1, \ldots, M_n$, la substitution $M[M_1/x_1, \ldots, M_n/x_n]$ associée à $\sigma : x_i \mapsto M_i$.

Enfin on note, pour $\mathcal{B}$ un ensemble de types de base fixé, `Type` l'ensemble $T_{\to\times 1+0}$ associé.

Réduction de tête

L'idée de la réduction de tête est de ne considérer que le sous-terme le plus à gauche du lambda-terme et de le réduire autant que possible, puis s'arrêter. Supposons par exemple que l'on ait un lambda-terme $M = P\ Q$ où P est une fonction, le but sera principalement de réduire P jusqu'à avoir une abstraction $\lambda x.P'$ pour pouvoir ensuite réduire $(\lambda x.P')\ Q \rhd_0 P'[Q/x]$. Notre réduction de tête va suivre cette idée.

Définition 7.2.2 (Réduction de tête)**.** *On définit la relation $\rhd_{\mathrm{wh}}$ sur $\Lambda^{\to\times 1+0}$ par les règles suivantes :*

$$\frac{}{(\lambda x^\tau.M)\ N \rhd_{\text{wh}} M[N/x]} \qquad \frac{}{\pi_1\ \langle M, N\rangle \rhd_{\text{wh}} M} \qquad \frac{}{\pi_2\ \langle M, N\rangle \rhd_{\text{wh}} N}$$

$$\frac{}{\delta\ (x \mapsto M \mid y \mapsto N)\ (\kappa_1\ P) \rhd_{\text{wh}} M[P/x]} \qquad \frac{}{\delta\ (x \mapsto M \mid y \mapsto N)\ (\kappa_2\ P) \rhd_{\text{wh}} N[P/y]}$$

$$\frac{M \rhd_{\text{wh}} M'}{M\ N \rhd_{\text{wh}} M'\ N} \qquad \frac{M \rhd_{\text{wh}} M'}{\pi_i\ M \rhd_{\text{wh}} \pi_i\ M'}\ i \in \{1,2\} \qquad \frac{M \rhd_{\text{wh}} M'}{\delta_\perp M \rhd_{\text{wh}} \delta_\perp M'}$$

$$\frac{P \rhd_{\text{wh}} P'}{\delta\ (x \mapsto M \mid y \mapsto N)\ P \rhd_{\text{wh}} \delta\ (x \mapsto M \mid y \mapsto N)\ P'}$$

Nous utiliserons cependant une définition équivalente utilisant la notion de contexte d'élimination.

Définition 7.2.3 (Contexte d'élimination)**.** *On définit l'ensemble* Elim *des contextes d'élimination de la façon suivante :*

$$E[\,] ::= [\,] \;\Big|\; E[\,]\ M \;\Big|\; \pi_1\ E[\,] \;\Big|\; \pi_2\ E[\,] \;\Big|\; \delta\ (x \mapsto M \mid y \mapsto N)\ E[\,] \;\Big|\; \delta_\perp\ E[\,]$$

où [ ], appelé un trou, est un caractère spécial. On définit $E[M]$ de façon naturelle en remplaçant dans l'expression de $E[\,]$ le caractère [ ] par M.

Remarque 7.2.1. Si $E[\,] \in$ Elim et $F[\,] \in$ Elim alors $E[F[\,]] \in$ Elim, ce qui nous donne une composition des contextes.

Proposition 7.2.1. *Pour M, N deux lambda-termes, on a $M \rhd_{\text{wh}} N$ si et seulement s'il existe un contexte $E[\,] \in$ Elim tel que $M = E[P], N = E[P']$ et $P \rhd_0 P'$.*

Démonstration. Montrons la première implication par induction sur $M \rhd_{\text{wh}} N$:

- Si $M \rhd_{\text{wh}} N$ est obtenu par l'une des cinq première règles alors $M \rhd_0 N$ donc avec le contexte $E[\,] = [\,]$ le résultat est direct.
- Si $M = E[M']\ Q$ et $N = E[N']\ Q$ avec $E[\,] \in$ Elim et $M' \rhd_0 N'$ alors $M = E'[M']$, $N = E'[N']$ avec $E' = E[\,]\ Q$ d'où le résultat.
- Si $M = \pi_i\ E[M']$, $N = \pi_i\ E[N']$ où $i \in \{1,2\}$, $E[\,] \in$ Elim et $M' \rhd_0 N'$ alors $M = E'[M']$ et $N = E'[N']$ avec $E' = \pi_i\ E[\,]$ d'où le résultat.
- Si $M = \delta_\perp E[M']$, $N = \delta_\perp E[N']$ avec $E[\,] \in$ Elim et $M' \rhd_0 N'$ alors $M = E'[M']$ et $N = E'[N']$ pour $E' = \delta_\perp\ E[\,]$.
- Si $M = \delta\ (x \mapsto P \mid y \mapsto Q)\ E[M']$ et $N = \delta\ (x \mapsto P \mid y \mapsto Q)\ E[N']$ avec $M' \rhd_0 N'$ alors en posant $E' = \delta\ (x \mapsto P \mid y \mapsto Q)\ E[\,]$ on a le résultat.

Réciproquement, on montre par induction sur $E[\,]$ le résultat :

- Pour $E[\,] = [\,]$ cela signifie que $M \rhd_0 N$ auquel cas $M \rhd_{\text{wh}} N$.
- Pour $E[\,] = E[\,]\ P$ on utilise l'hypothèse d'induction et la 6$^{\text{e}}$ règle d'induction de $\rhd_{\text{wh}}$.
- De même pour chaque règle i de construction de $E[\,]$ de notre grammaire il y a une règle correspondante dans $\rhd_{\text{wh}}$ qui permet de passer directement de l'hypothèse d'induction au résultat.

□

Proposition 7.2.2. *Soit M une forme normale pour $\rhd_{\text{wh}}$, typable et close, alors M est de l'une des formes suivantes :*

$$\lambda x.N \qquad \langle\rangle \qquad \langle N, P\rangle \qquad \kappa_1\ N \qquad \kappa_2\ N$$

Démonstration. On procède par induction sur M :

- M ne peut être une variable car c'est une formule close.
- Si $M = N\ P$ alors comme M est une forme normale, N doit aussi être une forme normale (ou on pourrait réduire à l'intérieur et donc réduire M) mais par hypothèse d'induction, N est de l'une des formes citées. Cependant comme M est typable N est de type $\sigma \to \tau$ donc est forcément de la forme $\lambda x.N'$ mais dans ce cas $M \rhd_0 N'[P/x]$ donc M ne peut être de la forme $N\ P$.
- Si $M = \lambda x.N$ alors M est de l'une des formes voulues, donc le résultat est vérifié.
- Si $M = \langle N, P\rangle$ alors M est de l'une des formes voulues.
- Si $M = \pi_i\ N$ pour $i \in \{1, 2\}$ alors N est une forme normale donc par hypothèse d'induction $N = \langle P_1, P_2\rangle$ pour des raisons de typages, mais alors $M \rhd_0 P_i$ donc M ne peut être de la forme $\pi_i\ N$.
- Si $M = \langle\rangle$ alors M est de l'une des formes voulues.
- Si $M = \kappa_i\ N$ pour $i \in \{1, 2\}$ alors M est de l'une des formes voulues.
- Si $M = \delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ P$ alors par hypothèse P est une forme normale de type $\sigma + \tau$ donc de la forme $\kappa_i\ P'$ donc $M \rhd_0 N_i[P'/x_i]$ donc M ne peut être de cette forme.
- Si $M = \delta_\bot\ M'$ alors M' est une forme normale close de type `void` ce qui est impossible par hypothèse d'induction.

□

Exercice 7.2.1. Montrer que si M est une forme normale pour $\rhd_{\text{wh}}$ typable, alors M est de l'une des formes suivantes :

$$E[x] \qquad \lambda x.N \qquad \langle\rangle \qquad \langle N, P\rangle \qquad \kappa_1\ N \quad \kappa_2\ N$$

où $E[\ ] \in \text{Elim}$ et $x \in \mathcal{V}$.

7.2.2 Interprétation adéquate

Avec ces outils en tête, nous pouvons démontrer le théorème de faible normalisation. Cette preuve n'est pas directe car les outils d'induction sur les lambda-termes ou le typage sont limités. A la place, on va chercher une façon d'associer à un type $\tau \in \texttt{Type}$ un ensemble de lambda-termes $[\![\tau]\!]$ de telle sorte que $[\![\tau]\!] \subseteq P$ avec P la propriété que l'on veut montrer sur les lambda-termes typés. En prouvant alors que $\vdash M : \tau \implies M \in [\![\tau]\!]$ on prouve notre propriété.

Définition 7.2.4 (Interprétation adéquate)**.** *Soit $[\![-]\!] : \texttt{Type} \to \mathcal{P}(\Lambda^{\to\times 1+0})$ une interprétation.*

Étant donnés une substitution σ et un contexte de typage Γ on note $\sigma \models_{[\![-]\!]} \Gamma$ si $\text{dom}(\Gamma) \subseteq \text{dom}(\sigma)$ et si pour tout $(x : \tau) \in \Gamma$, $\sigma(x) \in [\![\tau]\!]$.

On dit que $[\![-]\!]$ est acceptable si pour tout M et $\sigma \models_{[\![-]\!]} \Gamma$ tels que $\Gamma \vdash M : \tau$ on a $M\ \sigma \in [\![\tau]\!]$.

L'objectif est donc de construire une interprétation adéquate. Pour cela on raisonnera généralement par induction sur la relation de typage. Une première remarque est que si $\sigma \models_{[\![-]\!]} \Gamma$ alors pour $M = x$, si $\Gamma \vdash M : \tau$ cela signifie que $M\ \sigma = \sigma(x) \in [\![\tau]\!]$. Nous nous concentrons donc sur la définition de $[\![-]\!]$ pour les autres constructeurs de type. On ajoutera en plus les interprétations des constantes de types, càd des $\iota \in \mathcal{B}$, `unit` et `void`.

Définition 7.2.5. *Soient $A, B \in \mathcal{P}(\Lambda^{\to\times 1+0})$, on définit*

$$A \multimap B := \{M \mid \forall N \in A, M\ N \in B\}$$

et

$$A \otimes B := \{M \mid (\pi_1\ M \in A) \wedge (\pi_2\ M \in B)\}$$

L'objectif de cette définition est bien sûr de considérer $[\![A \to B]\!] = [\![A]\!] \multimap [\![B]\!]$ et $[\![A \times B]\!] = [\![A]\!] \otimes [\![B]\!]$, et on peut donc déjà essayer de démontrer que si l'interprétation est adéquate à une étape, elle l'est en considérant $\multimap$ et $\otimes$ pour construire l'étape suivante. On peut vérifier que le comportement vis à vis des règles d'élimination est le bon, mais pour la règle d'introduction de λ, on trouve un problème.

On considère

$$\frac{\Gamma, x : \kappa \vdash M : \tau}{\Gamma \vdash \lambda x.M : \kappa \to \tau}$$

Alors, pour $\sigma \models_{[\![-]\!]} \Gamma$ et $x \notin \mathrm{vl}(\sigma)$ (quitte à renommer x), comme $(\lambda x.M)\ \sigma = \lambda x.M\ \sigma$, on doit montrer que si $N \in [\![\kappa]\!]$ alors $(\lambda x.M\ \sigma)\ N \in [\![\tau]\!]$. Par hypothèse d'induction, on sait que $(M\ \sigma)[N/x] = M\ \sigma[N/x]$ pour $\sigma \models_{[\![-]\!]} \Gamma$, et on voudrait donc en déduire que $(\lambda x.M\ \sigma)\ N$: ce passage étant une application de réduction $\rhd_0$, on va demander une propriété de stabilité (remarquons qu'ici la stabilité est dans l'autre sens). Cependant il serait vain de ne demander que la stabilité par $\rhd_0$ car cela marchera pour une étape, mais ne nous permettra pas de fonctionner lors d'une induction sur les constructeurs de type. On va donc demander une stabilité pour $\rhd_{\mathrm{wh}}$:

Définition 7.2.6 (Stabilité par expansion de tête)**.** *On dit qu'un ensemble $A \subseteq \Lambda^{\to\times 1+0}$ est stable par expansion de tête si pour tout $N \in A$ et M tel que $M \rhd_{\mathrm{wh}} N$, on a $M \in A$. On va noter $(\mathcal{WHE})$ cette propriété.*

Exercice 7.2.2. Montrer que si A vérifie $(\mathcal{WHE})$ alors pour tout M et $N \in A$ tels que $M \rhd^*_{\mathrm{wh}} N$, on a $M \in A$.

Une première interprétation

Nous allons définir une première interprétation permettant de donner un théorème important : la propriété de la disjonction, qui dit que si $M : \tau + \kappa$ alors $M \rhd_{\mathrm{wh}} \kappa_i\ N$ pour un certain N et un certain $i \in \{1, 2\}$.

Définition 7.2.7 (Interprétation positive)**.** *On définit $\boxplus$, $\mathbf{1}$ et $\mathbf{0}$ de la façon suivante :*

$$A \boxplus B := \{M \mid (\exists N, (M \rhd^*_{\mathrm{wh}} \kappa_1\ N) \wedge (N \in A)) \vee (\exists N, (M \rhd^*_{\mathrm{wh}} \kappa_2\ N) \wedge (N \in B))\}$$

$$\mathbf{1} := \{M \mid M \rhd^*_{\mathrm{wh}} \langle\rangle\}$$

$$\mathbf{0} := \varnothing$$

Prouvons un lemme avant de prouver le théorème important de cette sous-partie :

Lemme 7.2.1. *Soient $A, B, C \in \mathcal{P}(\Lambda^{\to\times 1+0})$ où C vérifie $(\mathcal{WHE})$. Soient M, N_1, N_2 tels que $M \in A \boxplus B, N_1[Q/x_1] \in C$ et $N_2[Q/x_2] \in C$ pour tout $Q \in A \boxplus B$. Alors*

$$\delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ M \in C$$

Démonstration. Par définition de $M \in A \boxplus B$, on trouve M' tel que $M \rhd^*_{\mathrm{wh}} \kappa_i\ M'$ avec $i \in \{1, 2\}$. Cela signifie de plus que $\delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ M \rhd^*_{\mathrm{wh}} \delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ (\kappa_i\ M')$ donc que

$$\delta\ (x \mapsto N \mid y \mapsto P)\ M \rhd^*_{\mathrm{wh}} N_i[M'/x_i]$$

or $N_i[M'/x_i] \in C$ donc comme C vérifie $(\mathcal{WHE})$,

$$\delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ M \in C$$

□

On peut alors donner le résultat menant à la propriété de la disjonction :

Théorème 7.2.1. *Soit $[\![-]\!]$ une interprétation. Si :*

- $[\![\iota]\!]$ *vérifie* $(\mathcal{WHE})$ *pour tout* $\iota \in \mathcal{B}$.
- $[\![\texttt{void}]\!] = \mathbf{0}$ *et* $[\![\texttt{unit}]\!] = \mathbf{1}$.
- $[\![\sigma \to \kappa]\!] = [\![\sigma]\!] \multimap [\![\kappa]\!]$.
- $[\![\sigma \times \kappa]\!] = [\![\sigma]\!] \otimes [\![\kappa]\!]$.
- $[\![\sigma + \kappa]\!] = [\![\sigma]\!] \boxplus [\![\kappa]\!]$.

Alors :

- *Pour tout type* $\tau \in \texttt{Type}$, $[\![\tau]\!]$ *vérifie* $(\mathcal{WHE})$.
- $[\![-]\!]$ *est une interprétation adéquate.*

Démonstration. On montre par induction sur $\tau \in \texttt{Type}$ que $[\![\tau]\!]$ vérifie $(\mathcal{WHE})$:

- Par hypothèse $[\![\iota]\!]$ vérifie $(\mathcal{WHE})$ pour $\iota \in \mathcal{B}$.
- On vérifie que $\mathbf{0}$ et $\mathbf{1}$ vérifient $(\mathcal{WHE})$: pour $\mathbf{0}$ on quantifie sur l'ensemble vide, et pour $\mathbf{1}$, soit N tel que $N \rhd_{\text{wh}}^* \langle\rangle$, alors pour M tel que $M \rhd_{\text{wh}}^* N$, on a $M \rhd_{\text{wh}}^* \langle\rangle$ d'où le résultat.
- Si $[\![\tau]\!]$ et $[\![\kappa]\!]$ vérifient $(\mathcal{WHE})$, alors $[\![\tau]\!] \multimap [\![\kappa]\!]$ vérifie aussi $(\mathcal{WHE})$. En effet, si $N \in [\![\tau]\!] \multimap [\![\kappa]\!]$ et $M \rhd_{\text{wh}} N$ alors pour $P \in [\![\tau]\!]$, on a $N\ P \in [\![\kappa]\!]$ donc, comme $[\![\kappa]\!]$ vérifie $(\mathcal{WHE})$ et $M\ P \rhd_{\text{wh}} N\ P$, on en déduit que $M\ P \in [\![\kappa]\!]$, donc $M \in [\![\tau]\!] \multimap [\![\kappa]\!]$.
- Si $[\![\tau_1]\!]$ et $[\![\tau_2]\!]$ vérifient $(\mathcal{WHE})$, soit $N \in [\![\tau_1]\!] \otimes [\![\tau_2]\!]$ et $M \rhd_{\text{wh}} N$. Alors $\pi_i\ N \in [\![\tau_i]\!]$ et comme $\pi_i\ M \rhd_{\text{wh}} \pi_i\ N$ on en déduit que $\pi_i\ M \in [\![\tau_i]\!]$ donc $M \in [\![\tau_1]\!] \otimes [\![\tau_2]\!]$.
- Si $[\![\tau_1]\!]$ et $[\![\tau_2]\!]$ vérifient $(\mathcal{WHE})$, soit $N \in [\![\tau_1]\!] \boxplus [\![\tau_2]\!]$ et $M \rhd_{\text{wh}} N$. On trouve $i \in \{1, 2\}$ tel que $N \rhd_{\text{wh}}^* \kappa_i\ P$ avec $P \in [\![\tau_i]\!]$ mais alors $M \rhd_{\text{wh}}^* \kappa_i\ P$ donc $M \in [\![\tau_1]\!] \boxplus [\![\tau_2]\!]$.

On en déduit donc que toute interprétation vérifie $(\mathcal{WHE})$.

Montrons maintenant que $[\![-]\!]$ est adéquate. On considère $\sigma \models_{[\![-]\!]} \Gamma$ et on raisonne par induction sur $\Gamma \vdash M : \tau$ pour montrer que $M\ \sigma \in [\![\tau]\!]$:

- Comme on l'a dit, par hypothèse, la règle pour les variables est vérifiée.
- Si $M\ \sigma \in [\![\tau]\!]$ et $y \notin \text{vl}(\sigma)$ alors la conclusion tient toujours en prenant $\Gamma, y : \kappa \vdash M$.
- Si $\Gamma, x : \kappa \vdash M : \tau$ et $M\ \sigma \in [\![\tau]\!]$, alors pour tout $N \in [\![\kappa]\!]$ on sait que $M\ \sigma[N/x] \in [\![\tau]\!]$ donc $M \in [\![\kappa \to \tau]\!]$.
- Si $M = N\ P$, avec $\Gamma \vdash N : \kappa \to \tau$ et $\Gamma \vdash P : \kappa$, où $N\ \sigma \in [\![\kappa \to \tau]\!]$ et $P\ \sigma \in [\![\kappa]\!]$, alors par hypothèse $M\ \sigma = (N\ \sigma)\ (P\ \sigma) \in [\![\tau]\!]$.
- Si $M = \langle N_1, N_2 \rangle$ avec $\Gamma \vdash N_1 : \kappa_1$, $\Gamma \vdash N_2 : \kappa_2$ et $\tau = \kappa_1 \times \kappa_2$, alors $(\pi_i\ M\ )\sigma \rhd_{\text{wh}} N_i\ \sigma \in \kappa_i$ donc $M\ \sigma \in [\![\kappa_1 \times \kappa_2]\!]$.
- Si $M = \pi_i\ N$ avec $i \in \{1, 2\}$ et $\Gamma \vdash N : \langle \kappa_1, \kappa_2 \rangle$ alors par définition de $N \in [\![\kappa_1 \times \kappa_2]\!]$ on en déduit que $M \in [\![\tau]\!]$ et $\tau = \kappa_i$ pour un certain i.
- Si $M = \langle\rangle$ alors $M\ \sigma = \langle\rangle \in [\![\texttt{unit}]\!]$.
- Si $M = \kappa_i\ N$, $i \in \{1, 2\}$ avec $\Gamma \vdash N : \kappa$ et $\tau = \kappa + \kappa'$ alors par définition $M \in [\![\kappa + \kappa']\!]$.
- Si $M = \delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ P$ alors par le lemme précédent cela signifie que $M \in [\![\tau]\!]$ pour τ le type de $N_i[P/x_i]$.
- Si $M = \delta_\perp N$ alors $N \in \varnothing$ donc tout est vrai, la propriété en particulier.

Donc $[\![-]\!]$ est adéquate. □

Corollaire 7.2.1 (Propriété de la disjonction)**.** *On en déduit le corollaire suivant : si M est un terme typable dans l'ensemble vide alors*

- *si* $\vdash M : \tau_1 + \tau_2$ *alors il existe N tel que* $M \rhd_{\text{wh}} \kappa_i\ N$ *et* $\vdash N : \tau_i$ *pour un certain* $i \in \{1, 2\}$.
- $\vdash M : \texttt{void}$ *est impossible.*

Démonstration. On construit l'interprétation comme proposé dans le théorème précédent, en associant à ι l'ensemble $[\![\iota]\!] = \{M \mid \vdash M : \iota\}$. Par préservation du typage, si $N \rhd_{\text{wh}} M$ et $M \in [\![\iota]\!]$ on en déduit que $N \in [\![\iota]\!]$, donc l'interprétation est adéquate. On en déduit directement que $\vdash M : \texttt{void}$ est impossible puisque cela signifie $M \in \varnothing$. Si $\vdash M : \tau + \tau_2$ alors $M \in [\![\tau_1]\!] \boxplus [\![\tau_2]\!]$ donc on trouve par définition de $\boxplus$ un $i \in \{1, 2\}$ et un N tels que $M \rhd_{\text{wh}}^* \kappa_i\ N$ et $N \in [\![\tau_i]\!]$. De plus $\vdash \kappa_i\ N : \tau_1 + \tau_2$ donc $\vdash N : \tau_i$ par inversion sur le typage. □

Remarque 7.2.2. On n'a pas utilisé de σ dans ce cas car $\sigma = \varnothing \models_{[\![-]\!]} \varnothing$ et on considère uniquement le typage dans $\varnothing$.

7.2.3 Normalisation faible

On va considérer la normalisation faible pour $\rhd_{\text{wh}}$ dans toute cette partie. On note $\mathcal{WN}$ l'ensemble des termes faiblement normalisables pour $\rhd_{\text{wh}}$. Un premier résultat est que si $N \in \mathcal{WN}$ et $M \rhd_{\text{wh}} N$ alors $M \in \mathcal{WN}$. Donnons un autre résultat :

Lemme 7.2.2. *Soit $A \subseteq \mathcal{WN}$, l'ensemble $\{M \mid \exists N \in A, M \rhd_{\text{wh}}^* N\}$ est le plus petit ensemble contenant A et vérifiant $(\mathcal{WHE})$.*

Démonstration. Par définition si $M \in A$ alors $M \rhd_{\text{wh}}^* M$ donc M l'ensemble décrit contient A. Si pour un certain N il existe $P \in A$ tel que $N \rhd_{\text{wh}}^* P$ et que $M \rhd_{\text{wh}} N$ alors $M \rhd_{\text{wh}}^* P$ donc l'ensemble vérifie $(\mathcal{WHE})$. De plus un ensemble contenant A et vérifiant $(\mathcal{WHE})$ contient tous les M tels que $M \rhd_{\text{wh}}^* N$ pour $N \in A$ d'après un résultat précédent. □

Lemme 7.2.3. *Soient $A, B \subseteq \mathcal{WN}$, alors $A \otimes B \subseteq \mathcal{WN}$.*

Démonstration. On considère $M \in A \otimes B$. Si $M = \langle N, P\rangle$ alors M est une forme normale et donc $M \in \mathcal{WN}$. Sinon, alors $\pi_1\ M \in A \subseteq \mathcal{WN}$ donc $\pi_1\ M$ est sous forme normale (et donc M aussi) ou on trouve $N \in \mathcal{WN}$ tel que $\pi_1\ M \rhd_{\text{wh}} N$ mais par inversion sur $\rhd_{\text{wh}}$ on en déduit que $N = \pi_1\ N'$ et $M \rhd_{\text{wh}} N'$, donc $N' \in \mathcal{WN}$, donc $M \in \mathcal{WN}$. □

Cependant un nouveau problème se présente. Supposons qu'on veuille définir $[\![\texttt{void}]\!] = \mathbf{0}$ comme précédemment, alors $\mathbf{0} \multimap A = \{M \mid \forall N \in \varnothing, M\ N \in A\}$ qui correspond à l'ensemble $\Lambda^{\to\times 1+0}$ tout entier, et est donc trop large pour considérer que $[\![\texttt{void}]\!] \subseteq \mathcal{WN}$. On veut donc imposer que tous les ensembles $[\![\tau]\!]$ soient non vides. Évidemment, cela signifie qu'il nous faut modifier notre traduction de `void`, mais plus généralement cette condition est compatible avec les traductions négatives (qui se basent sur les éliminateurs comme les projections ou les applications) mais pas avec les traductions positives (qui se basent sur les constructeurs comme les injections). On veut donc une traduction de `unit`, `void` et $\tau + \tau'$ qui se basent sur les éliminateurs.

Un dernier obstacle à contourner est que si l'on regarde l'élimination de `void` par exemple, on a un type dont on ne connaît rien qui est mentionné, et de même pour l'élimination de $+$. On voudrait donc quantifier sur tous les $[\![\tau]\!]$ mais comme nous cherchons à définir $[\![\tau]\!]$, la définition serait circulaire. De plus, considérer juste l'ensemble des parties $A \subseteq \Lambda^{\to\times 1+0}$ est truc large, donc nous allons définir des parties plus restreintes vérifiant nos conditions dans lesquelles on veut travailler, que l'on appelle des ensembles saturés.

Définition 7.2.8 (Ensemble saturé)**.** *On définit $\mathcal{SAT}_{\mathcal{WN}} \subseteq \mathcal{P}(\mathcal{WN})$ l'ensemble des parties dites $\mathcal{WN}$-saturées, qui est tel que $A \in \mathcal{SAT}_{\mathcal{WN}}$ si et seulement*

- *si $E[x] \in A$ pour tout $E[\,] \in \text{Elim}$ et $x \in \mathcal{V}$*
- *$M \in A$ si $M \rhd_{\text{wh}} N$ et $N \in A$*

Remarque 7.2.3. La deuxième condition est exactement que A vérifie $(\mathcal{WHE})$.

Proposition 7.2.3. *L'ensemble $\mathcal{SAT}_{\mathcal{WN}}$ est un treillis complet pour l'inclusion dont le majorant est $\top := \mathcal{WN}$ et le minorant est $\bot := \{M \mid \exists E[\,] \in \text{Elim}, \exists x \in \mathcal{V}, M \rhd_{\text{wh}}^* E[x]\}$.*

Démonstration. Il est évident que $\mathcal{WN}$ est le majorant pour l'inclusion de $\mathcal{SAT}_{\mathcal{WN}}$. On sait de plus que l'ensemble $\{E[x] \mid E[\,] \in \text{Elim}, x \in \mathcal{V}\}$ est contenu dans tous les éléments de $\mathcal{SAT}_{\mathcal{WN}}$, donc le plus petit ensemble contenant cet ensemble et vérifiant $(\mathcal{WHE})$ est inclus dans tous les éléments de $\mathcal{SAT}_{\mathcal{WN}}$. Cet ensemble est $\{M \mid \exists E[\,] \in \text{Elim}, x \in \mathcal{V}, M \rhd_{\text{wh}}^* E[x]\}$ donc $\bot$ est effectivement le minorant de tous les éléments de $\mathcal{SAT}_{\mathcal{WN}}$ pour l'inclusion. Soit $\mathcal{A} \subseteq \mathcal{SAT}_{\mathcal{WN}}$ un ensemble de parties $\mathcal{WN}$-saturées, montrons que $\bigcap \mathcal{A} \in \mathcal{SAT}_{\mathcal{WN}}$. On sait que pour tout $E[\,] \in \text{Elim}, x \in \mathcal{V}$, $E[x] \in A$ pour $A \in \mathcal{A}$ et de plus si $N \in \bigcap \mathcal{A}$ alors pour chaque $A \in \mathcal{A}$, si $M \rhd_{\text{wh}} N$ alors $M \in A$, donc au total $M \in \bigcap \mathcal{A}$, donc $\bigcap \mathcal{A} \in \mathcal{SAT}_{\mathcal{WN}}$. Pour $\bigcup \mathcal{A}$, le fait de contenir les $E[x]$ découle déjà du résultat pour l'intersection, et si $N \in \mathcal{A}$ et $M \rhd_{\text{wh}} N$ alors on trouve $A \in \mathcal{A}$ tel que $N \in A$ et comme A vérifie $(\mathcal{WHE})$, $M \in A$ donc $M \in \bigcup \mathcal{A}$. □

On définit alors les interprétations négatives sur $\mathcal{SAT}_{\mathcal{WN}}$.

Proposition 7.2.4. *Une autre écriture de $\bot$ est*

$$\bot = \{M \mid \forall A \in \mathcal{SAT}_{\mathcal{WN}}, \delta_\bot\ M \in A\}$$

Démonstration. Si $M \in \bot$ alors on trouve $E[x]$ tel que $M \rhd_{\text{wh}}^* E[x]$ et $E[x] \in A$ pour tout $A \in \mathcal{SAT}_{\mathcal{WN}}$ donc $M \in A$ et $\delta_\bot\ M \in A$ en considérant $E'[x] = \delta_\bot\ E[x]$. Réciproquement, si $\delta_\bot\ M \in A$ pour tout $A \in \mathcal{SAT}_{\mathcal{WN}}$, alors en particulier $\delta_\bot\ M \in \bot$ donc on trouve $E[x]$ tel que $\delta_\bot\ M \rhd_{\text{wh}}^* E[x]$ et en prenant $E'[x] = \delta_\bot\ E[x]$ on trouve donc $E'[x]$ tel que $M\ \rhd_{\text{wh}}^* E'[x]$ par inversion sur $\rhd_{\text{wh}}$, donc $M \in \bot$. □

Définition 7.2.9. *Soient $A, B \in \mathcal{SAT}_{\mathcal{WN}}$. On définit pour $C \in \mathcal{SAT}_{\mathcal{WN}}$ l'ensemble intermédiaire $\Lambda_{A,C,x} := \{N \mid \forall P \in A, N[P/x] \in C\}$ puis*

$$A \oplus_{\mathcal{WN}} B := \{M \mid \forall C \in \mathcal{SAT}_{\mathcal{WN}}, \forall N_1 \in \Lambda_{A,C,x_1}, \forall N_2 \in \Lambda_{B,C,x_2}, \delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ M \in C\}$$

On vérifie alors que $\mathcal{SAT}_{\mathcal{WN}}$ est bien stable par les relations attendues.

Lemme 7.2.4. *Soient $A, B \in \mathcal{SAT}_{\mathcal{WN}}$, alors les ensembles suivants sont dans $\mathcal{SAT}_{\mathcal{WN}}$:*

$$A \multimap B \qquad A \otimes B \qquad A \oplus_{\mathcal{WN}} B$$

Démonstration. Pour $E[x]$ avec $E[\,] \in \text{Elim}, x \in \mathcal{V}$, si l'on prend $M \in A$ alors $E[x]\ M = E'[x]$ avec $E'[\,] = E[\,]\ M$ donc comme $B \in \mathcal{SAT}_{\mathcal{WN}}$, $E[x]\ M \in B$, d'où $E[x] \in A \multimap B$. Si $N \in A \multimap B$ et $M \rhd_{\text{wh}} N$ alors pour $P \in A$, $M\ P \rhd_{\text{wh}} N\ P$ et $N\ P \in B$ donc par $(\mathcal{WHE})$ on en déduit que $M\ P \in B$, donc $M \in A \multimap B$, donc $A \multimap B$ vérifie $(\mathcal{WHE})$.

Soit $E[\,] \in \text{Elim}, x \in \mathcal{V}$, on définit $E' := \pi_i\ E[\,]$ et comme A et B contiennent tous les $E'[x]$, cela signifie que $E[x] \in A \otimes B$. Si $N \in A \otimes B$ et $M \rhd_{\text{wh}} N$ alors comme $\pi_1\ N \in A$ et $\pi_2\ N \in B$, cela signifie que $\pi_1\ M \in A$ et $\pi_2\ M \in B$ par $(\mathcal{WHE})$, donc $M \in A \otimes B$. Donc $A \otimes B$ vérifie $(\mathcal{WHE})$.

Pour $E[\,] \in \text{Elim}, x \in \mathcal{V}$, $C \in \mathcal{SAT}_{\mathcal{WN}}$ et $N_1 \in \Lambda_{A,C,x_1}, N_2 \in \Lambda_{B,C,x_2}$, en posant comme nouveau contexte $E'[\,] = \delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ E[\,]$ on trouve que $E'[x] \in C$ car $C \in \mathcal{SAT}_{\mathcal{WN}}$. Si $N \in A \oplus_{\mathcal{WN}} B$ et $M \rhd_{\text{wh}} N$ alors soient C, N_1, N_2 définis comme précédemment. On sait de plus que $\delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ N \in C$ donc par $(\mathcal{WHE})$ on en déduit que $\delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ M \in C$, donc $M \in A \oplus_{\mathcal{WN}} B$. Donc $A \oplus_{\mathcal{WN}} B$ vérifie $(\mathcal{WHE})$. □

On en vient maintenant au théorème analogue à celui de la section précédente.

Théorème 7.2.2. *Soit $[\![-]\!]$ une interprétation telle que*

- *Pour tout $\iota \in \mathcal{B}$, $[\![\iota]\!] \in \mathcal{SAT}_{\mathcal{WN}}$.*
- $[\![\texttt{void}]\!] = \bot$ *et* $[\![\texttt{unit}]\!] = \top$.

- *Pour tout type τ, τ',*

$$[\![\tau \to \tau']\!] = [\![\tau]\!] \multimap [\![\tau']\!] \qquad [\![\tau \times \tau']\!] = [\![\tau]\!] \otimes [\![\tau']\!] \qquad [\![\tau + \tau']\!] = [\![\tau]\!] \oplus_{\mathcal{WN}} [\![\tau']\!]$$

Alors

- *Pour tout $\tau \in$ `Type`, $[\![\tau]\!] \in \mathcal{SAT}_{\mathcal{WN}}$.*
- *$[\![-]\!]$ est adéquate.*

Démonstration. Le fait que $[\![\tau]\!] \in \mathcal{SAT}_{\mathcal{WN}}$ se déduit directement des lemmes précédents en effectuant une induction sur `Type`.

Soit $\sigma \models_{[\![-]\!]} \Gamma$ pour une substitution σ et un environnement Γ, on va montrer par induction sur $\Gamma \vdash M : \tau$ que pour tout $M \in \Lambda^{\to\times 1+0}, \tau \in$ `Type` on a $\Gamma \vdash M : \tau \implies M \in [\![\tau]\!]$:

- Les 6 premières règles d'induction se traitent strictement de la même façon puisque les définitions sont identiques.
- Si $M = \langle\rangle$ alors $M\ \sigma = \langle\rangle \in \mathcal{WN} = [\![\texttt{unit}]\!]$.
- Si $M = \kappa_i\ N, i \in \{1, 2\}$ et $\Gamma \vdash M : \tau_1 + \tau_2$ alors pour $C \in \mathcal{SAT}_{\mathcal{WN}}$, $N_1 \in \Lambda_{[\![\tau_1]\!],C,x}$ et $N_2 \in \Lambda_{[\![\tau_2]\!],C,x}$ on a
$$\delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ M \rhd_{\text{wh}} N_i[N/x_i] \in C$$
donc par $(\mathcal{WHE})$ on en déduit que $\delta\ (x_1 \mapsto N_1 \mid x_2 \mapsto N_2)\ M \in C$, d'où $M \in [\![\tau_1]\!] \oplus_{\mathcal{WN}} [\![\tau_2]\!]$.
- Si $M = \delta_\perp N$, $\Gamma \vdash M : \tau$ avec $N \in [\![\texttt{void}]\!]$ alors par notre définition équivalente de $\perp = [\![\texttt{void}]\!]$ on en déduit que $M \in [\![\tau]\!]$.

Donc $[\![-]\!]$ est adéquate. □

Corollaire 7.2.2. *Si on trouve Γ tel que $\Gamma \vdash M : \tau$ pour un lambda-terme M et un type τ, alors $M \in \mathcal{WN}$.*

Démonstration. Comme $[\![\tau]\!] \subseteq \mathcal{WN}$ et que $[\![-]\!]$ est adéquate, cela signifie que $M\ \sigma \in [\![\tau]\!]$ pour $\sigma \models_{[\![-]\!]} \Gamma$, donc $M\ \sigma$ est normalisable. En prenant la fonction σ qui à x associe x^τ avec τ tel que $(x : \tau) \in \Gamma$, on en déduit que $M\ \sigma = M$ et donc que M est normalisable. □

7.2.4 Normalisation forte

Notion de terme fortement normalisant

Commençons par nous placer dans le cas d'un système de réécriture $(E, \to)$. On va commencer par montrer la validité d'une définition inductive pour traduire ce qu'est un terme fortement normalisable.

Définition 7.2.10 (Ensemble $\mathcal{SN}$)**.** *Soit $(E, \to)$, on définit $\mathcal{SN}$ par la partie de E constituée des éléments $x \in E$ tels qu'il n'existe pas de suite infinie $(x_i)_{i\in\mathbb{N}}$ telle que $x_0 = x$ et $x_i \rhd x_{i+1}$ pour tout $i \in \mathbb{N}$.*

Proposition 7.2.5. *L'ensemble $\mathcal{SN}'$ défini par induction comme la plus petite partie de E stable par la règle suivante :*

$$\frac{\forall y \in E, x \to y \implies y \in \mathcal{SN}'}{x \in \mathcal{SN}'}$$

Et $\mathcal{SN}' = \mathcal{SN}$.

Démonstration. On raisonne par induction sur $\mathcal{SN}'$. Soit $x \in E$, on suppose que pour tout $y \in E$ tel que $x \to y$, il n'existe pas de $(y_i)_{i\in\mathbb{N}}$ avec $y_0 = y$ et $\forall i \in \mathbb{N}, y_i \to y_{i+1}$. Supposons qu'il existe une suite $(x_i)_{i\in\mathbb{N}}$ telle que $x_0 = x$ et $\forall i \in \mathbb{N}, x_i \to x_{i+1}$. En définissant $y_i = x_{i+1}$ on trouve une contradiction, donc une telle suite (x_i) n'existe pas. Donc $\mathcal{SN}' \subseteq \mathcal{SN}$.

Par contraposée, si $x \notin \mathcal{SN}'$ alors on peut trouver un élément $y \notin \mathcal{SN}'$. Dans ce cas on peut trouver par l'axiome du choix dépendant une suite (x_i) telle que $\forall i \in \mathbb{N}, x_i \notin \mathcal{SN}'$. Donc $\mathcal{SN} \subseteq \mathcal{SN}'$. D'où le résultat. □

L'ensemble $\mathcal{SN}$ désignera maintenant l'ensemble $\mathcal{SN}$ correspondant au système de réécriture $(\Lambda^{\to\times 1+0}, \rhd)$.

Passer de la réduction de tête à la β-réduction

Encore une fois, nous allons adapter notre interprétation adéquate en changeant l'invariant qui définira nos ensembles saturés. Pour adapter le fait de contenir $E[x]$, il nous suffit de considérer $E[\,] \in \text{Elim} \cap \mathcal{SN}$. Cependant, pour l'expansion de tête, s'assurer d'avoir $(\lambda x.M\ \sigma)\ N \in A$ si $M\ \sigma[N/x] \in A$, est moins direct. En effet, en imaginant par exemple $(\lambda x.\lambda y.y)\ \Omega$ on voit qu'un terme non normalisable peut disparaître lors de la réduction. Cependant, si N est lui-même dans $\mathcal{SN}$, alors $(\lambda x.M\ \sigma)\ N$ sera bien dans $\mathcal{SN}$.

Pour faciliter les notations, nous allons définir la notion de réduction de tuples.

Définition 7.2.11 (Réduction de tuples)**.** *Soient $M_1, \dots, M_n$ des termes. On note $(M_1, \dots, M_n) \triangleright (N_1, \dots, N_n)$ s'il existe $i \in \{1, \dots, n\}$ tel que $M_i \triangleright N_i$ et pour $j \neq i$ on a $M_j = N_j$.*

Nous allons maintenant montrer un résultat intermédiaire pour montrer ensuite la propriété $(\mathcal{WHE})$ dans les cas qui nous intéressent.

Lemme 7.2.5 (Standardisation faible)**.** *Soit $E[\,] \in \text{Elim}$. Alors :*

- *Si $E[(\lambda x.M)\ N] \triangleright P$, alors soit $P = E[M[N/x]]$, soit $P = E'[(\lambda x.M')\ N']$ où $(E[\,], M, N) \triangleright (E'[\,], M', N')$.*
- *Pour $i \in \{1, 2\}$, si $E[\pi_i\ \langle M_1, M_2\rangle] \triangleright N$, alors soit $N = E[M_i]$ soit $N = E'[\pi_i\ \langle M_1', M_2'\rangle]$ où $(E[\,], M_1, M_2) \triangleright (E'[\,], M_1', M_2)$.*
- *Soit $i \in \{1, 2\}$, si $E[\delta\ (x_1 \mapsto M_1 \mid x_2 \mapsto M_2)\ (\kappa_i\ N)] \triangleright P$ alors on a deux possibilités : $P = E[M_i[N/x_i]]$ ou $P = E'[\delta\ (x_1 \mapsto M_1' \mid x_2 \mapsto M_2')\ (\kappa_i\ N')]$ avec $(E[\,], M_1, M_2, N) \triangleright (E'[\,], M_1', M_2', N')$.*

Démonstration. On montre d'abord le cas sans $E[\,]$:

- Si $(\lambda x.M)\ N \triangleright P$ alors par inversion soit $P = M[N/x]$, soit $P = (\lambda x.M')\ N'$ avec $(M, N) \triangleright (M', N')$.
- Si $\pi_i\ \langle M_1, M_2\rangle \triangleright N$ alors par inversion soit $N = M_i$ soit $N = \pi_i\ \langle M_1', M_2'\rangle$ avec $(M_1, M_2) \triangleright (M_1', M_2')$.
- Si $\delta\ (x_1 \mapsto M_1 \mid x_2 \mapsto M_2)\ (\kappa_i\ N) \triangleright P$ alors par inversion, soit $P = M_i[P/x_i]$ soit $P = \delta\ (x_1 \mapsto M_1' \mid x_2 \mapsto M_2')\ (\kappa_i\ N')$ avec $(M_1, M_2, N) \triangleright (M_1', M_2', N')$.

On raisonne maintenant par induction sur $E[\,]$:

- Si $E = [\,]$ alors le résultat découle de ce qu'on a prouvé juste avant.
- Supposons que la propriété soit vraie pour $E[\,]$. On considère alors $E'[\,] = E[\,]\ M$ et un N entrant dans l'un des cas précédemment traités. Dans chaque cas, si $E'[N] \triangleright P$, par inversion sur $\triangleright$, on trouve au choix que $E[N] \triangleright N'$ et auquel cas par hypothèse d'induction on en déduit le résultat, ou que $M \triangleright M'$ ce qui nous donne que $(E'[\,], N) \triangleright (E''[\,], N)$ avec $E'' = E[\,]\ M'$.

□

Exercice 7.2.3. Traiter les autres cas d'induction.

On en déduit le résultat de stabilité :

Proposition 7.2.6 (Stabilité par réduction de tête)**.** *Soit $E[\,] \in \text{Elim}$. Alors*

- *Si $E[M[N/x]] \in \mathcal{SN}$ et $N \in \mathcal{SN}$ alors $E[(\lambda x.M)\ N] \in \mathcal{SN}$.*
- *Soit $i \in \{1, 2\}$. Si $E[M_i] \in \mathcal{SN}$ et $M_{3-i} \in \mathcal{SN}$ alors $E[\pi_i\ \langle M_1, M_2\rangle] \in \mathcal{SN}$.*
- *Soit $i \in \{1, 2\}$. Si $E[M_i[N/x_i]]$ et $N, M_{3-i} \in \mathcal{SN}$ alors $E[\delta\ (x_1 \mapsto M_1 \mid x_2 \mapsto M_2)\ (\kappa_i\ N)] \in \mathcal{SN}$.*

Démonstration. On va montrer par induction sur $(E[\,], M, N)\mathcal{SN}$ que si $E[M[N/x]] \in \mathcal{SN}$ et $N \in \mathcal{SN}$ alors $E[(\lambda x.M)\ N] \in \mathcal{SN}$. Supposons que pour tout $(E'[\,], M', N')$ tel que $(E[\,], M, N) \triangleright (E'[\,], M', N')$ alors si $E'[M'[N'/x]] \in \mathcal{SN}$ et $N' \in \mathcal{SN}$ alors $E[(\lambda x.M')\ N'] \in \mathcal{SN}$. On suppose de plus que $E[M[N/x]] \in \mathcal{SN}$ et $N \in \mathcal{SN}$. Alors, si $E[(\lambda x.M)\ N] \triangleright P$ il y a deux cas possibles : soit $P = M[N/x]$ et dans ce cas $P \in \mathcal{SN}$, soit $(E[\,], M, N) \triangleright (E'[\,], M', N')$ et $P = E'[(\lambda x.M')\ N']$. Dans ce cas, on sait que $E'[M'[N/x]] \in \mathcal{SN}$ car

$E[M[N/x]] \rhd^* E'[M'[N'/x]]$ et $E[M[N/x]] \in \mathcal{SN}$. De plus $N \rhd^* N'$ donc $N' \in \mathcal{SN}$ d'où $P = E[(\lambda x.M')\ N'] \in \mathcal{SN}$. Ainsi par induction pour tout $(E[\,], M, N) \in \mathcal{SN}$, la propriété est vraie. Enfin, on remarque que si $E[M[N/x]] \in \mathcal{SN}$ alors en particulier $(E[\,], M, N) \in \mathcal{SN}$ car une réduction infinie depuis $(E[\,], M, N)$ serait infinie depuis $E[\,]$ ou depuis M (car $N \in \mathcal{SN}$) et pourrait donc se simuler par compatibilité de $\rhd$ depuis $E[M[N/x]]$.

On traite les deux autres cas de façon analogue. □

Exercice 7.2.4. Rédiger les deux autres cas de la preuve.

Interprétation adéquate

On peut maintenant adapter notre interprétation. Nous allons donner la notion d'ensemble $\mathcal{SN}$-saturé, qui est une adaptation de $\mathcal{SAT}_{\mathcal{WN}}$ comme attendu.

Définition 7.2.12. *On définit $\mathcal{SAT}_{\mathcal{SN}}$ comme la partie de $\mathcal{P}(\mathcal{SN})$ contenant tous et uniquement les ensembles A tels que :*

- *Pour tout $x \in \mathcal{V}$ et $E[\,] \in \mathrm{Elim} \cap \mathcal{SN}$, $E[x] \in A$.*
- *Si $N \in \mathcal{SN}$ et $E[M[N/x]] \in A$ alors $E[(\lambda x.M)\ N] \in A$.*
- *Pour tous $i \in \{1,2\}$, si $E[M_i] \in A$ et $M_{3-i} \in \mathcal{SN}$ alors $E[\pi_i\ \langle M_1, M_2\rangle] \in A$.*
- *Pour tous $i \in \{1,2\}$, si $E[M_i[N/x_i]] \in A$ et $N, M_{3-i} \in \mathcal{SN}$ alors*

$$E[\delta\ (x_1 \mapsto M_1 \mid x_2 \mapsto M_2)\ (\kappa_i\ N)] \in A$$

Exercice 7.2.5. Montrer que $\mathcal{SAT}_{\mathcal{SN}}$ est un treillis complet pour l'inclusion avec pour bornes supérieure et inférieure respectivement

$$\top := \mathcal{SN}$$

et

$$\bot := \{M \mid \exists E[\,] \in \mathrm{Elim}, \exists x \in \mathcal{V}, M \rhd^* E[x]\}$$

On adapte enfin $\oplus$.

Définition 7.2.13 (Somme dans $\mathcal{SAT}_{\mathcal{SN}}$)**.** *On définit l'opération $\oplus_{\mathcal{SN}}$, pour $A \in \mathcal{SAT}_{\mathcal{SN}}$ et $B \in \mathcal{SAT}_{\mathcal{SN}}$, par :*

$$A \oplus_{\mathcal{SN}} B := \{M \mid \forall C \in \mathcal{SAT}_{\mathcal{SN}}, \forall N_1 \in \Lambda_{A,C,x_1}, \forall N_2 \in \Lambda_{B,C,x_2}, \delta\ (x_1 \mapsto M_1 \mid x_2 \mapsto M_2)\ M \in C\}$$

Lemme 7.2.6. *Soient $A, B \in \mathcal{SAT}_{\mathcal{SN}}$, alors les ensembles suivants sont dans $\mathcal{SAT}_{\mathcal{SN}}$:*

$$A \multimap B \qquad A \otimes B \qquad A \oplus_{\mathcal{SN}} B$$

Démonstration. On remarque que si $E[\,] \in \mathrm{Elim} \cap \mathcal{SN}$ et $x \in \mathcal{V}$, alors pour $N \in A$ on a $E[\,]\ N \in B$, et $\pi_1\ E[x] \in A, \pi_2\ E[x] \in B$ et si $C \in \mathcal{SAT}_{\mathcal{SN}}$ et N_1, N_2 définis comme dans la définition de $\oplus_{\mathcal{SN}}$, alors $\delta\ (x_1 \mapsto M_1 \mid x_2 \mapsto M_2)\ E[x] \in C$ car tous les ensembles dans $\mathcal{SAT}_{\mathcal{SN}}$ contiennent $E[x]$ pour $E[\,] \in \mathrm{Elim} \cap \mathcal{SN}$, et nous n'ajoutons que des termes dans $\mathcal{SN}$ donc on obtient bien un $E'[x]$ pour $E'[\,] \in \mathrm{Elim} \cap \mathcal{SN}$.

Supposons que $N \in \mathcal{SN}$ et $E[M[N/x]] \in A \multimap B$, alors soit $P \in A$. D'après ce qu'on a dit, $E[M[N/x]]\ P \in B$, or $B \in \mathcal{SAT}_{\mathcal{SN}}$ et $N \in \mathcal{SN}$ donc $E[(\lambda x.M)\ N]\ P \in B$. De plus par le lemme précédent, comme $N \in \mathcal{SN}$ et $E[M[N/x]]\ P \in \mathcal{SN}$ on en déduit que $E[M[N/x]] \in \mathcal{SN}$. Donc $E[(\lambda x.M)\ N] \in B$.

On traite les autres cas de façon similaire. □

Exercice 7.2.6. Démontrer les cas restants.

On peut alors donner le théorème d'adéquation.

Théorème 7.2.3. *Soit $[\![-]\!]$ une interprétation telle que :*

- *Pour tout $\iota \in \mathcal{B}$, $[\![\iota]\!] \in \mathcal{SAT}_{\mathcal{SN}}$.*
- $[\![\texttt{unit}]\!] = \top$ *et* $[\![\texttt{void}]\!] = \bot$.
- *Pour tout $\tau, \tau' \in \texttt{Type}$, $[\![\tau \to \tau']\!] = [\![\tau]\!] \multimap [\![\tau']\!]$.*
- *Pour tout $\tau, \tau' \in \texttt{Type}$, $[\![\tau \times \tau']\!] = [\![\tau]\!] \otimes [\![\tau']\!]$.*
- *Pour tout $\tau, \tau' \in \texttt{Type}$, $[\![\tau + \tau']\!] = [\![\tau]\!] \oplus_{\mathcal{SN}} [\![\tau']\!]$.*

Alors $[\![-]\!]$ est adéquat.

Exercice 7.2.7. Montrer le théorème précédent.

Corollaire 7.2.3 (Forte normalisation)**.** *Soit $M \in \Lambda^{\to\times 1+0}$ tel qu'il existe Γ, τ tels que $\Gamma \vdash M : \tau$, alors $M \in \mathcal{SN}$.*

Démonstration. On construit l'interprétation adéquate respectant les prémisses du théorème précédent. Pour cela, pour $\iota \in \mathcal{B}$, on associe $[\![\iota]\!] = \{M \mid \exists \Gamma, \Gamma \vdash M : \iota\}$. On vérifie que cet ensemble est bien dans $\mathcal{SAT}_{\mathcal{SN}}$:

- Pour un contexte $E[\,] \in \mathrm{Elim} \cap \mathcal{SN}$ typé dans Γ, on prend $E[x]$ avec $x \notin \mathrm{vl}(E[\,])$ et $(\Gamma, x : \iota)$, donc $E[x] \in [\![\iota]\!]$.
- Si $N \in \mathcal{SN}$ et que $\Gamma \vdash E[M[N/x]] : \iota$ alors comme $E[(\lambda x.M)\ N] \rhd E[M[N/x]]$, par préservation du typage, on déduit que $\Gamma \vdash E[(\lambda x.M)\ N] : \iota$.
- De même pour les autres cas, par préservation du typage.

On en déduit que si $\sigma \models_{[\![-]\!]} \Gamma$ et $\Gamma \vdash M : \tau$ alors $M\ \sigma \in [\![\tau]\!]$ donc en particulier pour σ la substitution triviale, on a $M \in [\![\tau]\!]$ donc $M \in \mathcal{SN}$. □