Logique du premier ordre

Sébastien Hemon
EPITA

December 6, 2015

1 Introduction

1.1 Une formalisation tardive

Si l'histoire nous conte que la logique fut mère de réflexion philosophique et mathématique depuis la Grèce antique, ce n'est que depuis à peine deux siècle que l'on commençât à la formaliser.

Il eut été difficile d'extraire des raisonnements menés, non plus la substance ou le sens, mais bien la structure véhiculée par les pensées de toutes les sciences, qu'elles soient humaines, dures ou techniques. En ce sens, Frege, De Morgan ont été des précurseurs. Aristote et Platon, déjà, avaient envisagé de concevoir des systèmes articulés sur lesquels l'argumentation serait basée. Mais, heurtés par la quête de vérité, ils n'avaient pu convenablement distinguer la structure syntaxique de la logique de la sémantique portée. La notion philosophique de *vérité* s'était dressée contre eux telle un mur infranchissable.

Au XXeme siècle, les travaux de Russel, Gödel, Tarski ou encore Gentzel vont renforcer considérablement l'étude de la logique en tant que tel. Se crée alors une distinction qui va conférer à la logique son autonomie : la dualité syntaxe / sémantique. Syntaxiquement, on prouve, on formule : la vérité échappe à ce concept. Sémantiquement, on interprète, on valide : la vérité est une valeur reflétant ce qui est exprimé. Les travaux de Russel et Tarski portent davantage sur l'aspect sémantique et la recherche de vérité, tandis que ceux de Gödel et Gentzel ont ouvert la voie des études syntaxiques pures.

En informatique, il serait peu pertinent de déclarer que tel algorithme ou procédure est *vraie*. En revanche, nous l'attendrions de l'interprétation que l'on fait du résultat fourni. L'informaticien doit donc travailler différemment sur les aspects syntaxiques et sémantiques, tout en sachant convenablement les entremêler afin de ne pas perdre de vue la réalité que l'on cherche à étudier, prédire ou décrire à l'aide de techniques de simulation virtuelles.

Le point de vue adopté dans ce cours pourra donc être qualifié de moderne, en ce sens que nous considérerons acquises les découvertes fondamentales des grands logiciens des XIXeme et

XXeme siècle. Nous n'oublierons cependant pas les enseignements philosophiques des penseurs de la Grèce Antique : ils sauvegarderont nos esprits de dérivés logiciennes absconces et retiendront cet adage Pascalien : "Science sans conscience n'est que ruine de l'âme".

1.2 Définitions inductives

Nous introduisons un nouveau mode de définition, dit *par induction*.

Definition 1.2.1 (Par induction)**.** *On considère un type d'objets $\mathcal{T}$ à définir. Pour cela, on donne :*

i) Un ensemble d'objets atomiques $\mathcal{A} = \{a_k \ ; \ k \in I\}$[1] *considérés de façon* ad hoc *comme du type $\mathcal{T}$.*

ii) Un ensemble d'opérateurs $\mathcal{C} = \{op_j \ ; \ j \in J\}$ appelés constructeurs, adjoints de leurs arités respectives $ar(op_k) = p_k \in \mathbb{N}$.

iii) Une profondeur $d \in \mathbb{N} \cup \{\omega$ où ω désigne la capacité pour d de prendre toute valeur possible de $\mathbb{N}$. (une construction formelle et rigoureuse de ω pourra être env,isagée durant le module).

Ainsi, on note $\mathcal{T}_n$ les objets de type $\mathcal{T}$ d'ordre $n \in \mathbb{N}$. Ils sont définis par :

$$\begin{aligned} \mathcal{T}_0 &:= \mathcal{A} \\ \mathcal{T}_{n+1} &:= \{op(x_1; x_2; \ldots; x_{ar(op)}) \ ; \ op \in \mathcal{C} \ , (x_1; \ldots x_{ar(op)}) \in \mathcal{T}_n^{ar(op)}\} \end{aligned}$$

Enfin, le type $\mathcal{T}$ est obtenu par :

$$\mathcal{T} = \bigcup_{e \leq d} \mathcal{T}_e$$

Il nous serait possible d'approfondir la notion de profondeur et de définir des processus itératifs excédant les nombres finis sans majorant. Dans un premier temps, ceci n'a pas d'utilité. En revanche, une fois les ordinaux transfinis définis, cela pourrait s'avérer utiles dans certains champs spécifiques d'étude. Voir la section **Prolongements** pour de plus amples informations.

En guise d'application, nous proposons une définition d'une expression algébrique :

Definition 1.2.2. *Les objets du type* expression algébrique *sont définis par :*

- *Les nombres et symboles lettrés de variables sont les expressions algébriques atomiques.*
- *Les opérateurs usuels $+$, $\times$, $-$, $\div$, $\sqrt{\ }$, ... munis de leurs arités respectives*
- *ω est la profondeur maximale.*

Ainsi, si $A_1, \ldots A_n$ sont des expressions algébriques et op un opérateur algébrique d'arité n, alors $op(A,1, \ldots, A_n)$ est aussi une expression algébrique.

[1] I étant un ensemble d'indices permettant l'énumération des objets de $\mathcal{A}$

Notons que cette définition impose une écriture des calculs en notation polonaise. De plus, les notions de *nombres*, *opérateur usuel* ou même symbole de variable pourraient être remis en question selon les consensus ou le contexte.

Nous verrons donc ultérieurement comment remédier à ce problème en définissant des langages algébriques. La notion d'expression algbrique dépendra donc du langage choisi. En contre-partie, il sera possible de travailler avec ses propres expressions, bien qu'elles puissent ne pas être reconnues comme telles par d'autres environnements.

2 Calcul Propositionnel

2.1 Aspect Syntaxique

On se donne un ensemble de symboles appelés *connecteurs logiques* :

$$\mathcal{C}_l = \{\bot\ ;\ \top\ ;\ \neg\ ;\ \wedge\ ;\ \vee\ ;\ \Rightarrow\ ;\ \Leftrightarrow\}$$

D'un point de vue syntaxique, ces symboles sont considérés comme :

- des constantes pour $\bot$ et $\top$
- un opérateur fonctionnel à un argument pour $\neg$
- des opérateurs fonctionnels à deux arguments pour $\wedge\ ;\ \vee\ ;\ \Rightarrow\ ;\ \Leftrightarrow$

NB : Il serait possible d'inclure le symbole $\oplus$ désignant le $X - OR$ et de décliner en conséquence l'exposé. *Remarque :* On pourra garder à l'esprit que ces connecteurs renvoient aux opérateurs logiques booléens classiques. Mais l'intérêt du calcul propositionnel et de logique réside également dans la possibilité d'interpréter différement ces symboles tout en conservant les conclusions générales établies syntaxiquement.

A propos de l'implication

Certains ouvrages proposent le choix de la syntaxe $\rightarrow$ pour désigner l'opérateur flêche, interprétable comme étant l'implication $\Rightarrow$. Ceci réside dans une volonté avérée d'amener le lecteur à distinguer la notion d'implication (sémantique) de celle de déduction (syntaxique). Nous remarquerons en quoi cela peut s'avérer pertinent chaque fois que nécessaire.

On considère à présent l'ensemble Λ constitué de l'alphabet latin majuscule ainsi que, pour chacune de ces lettres, d'une occurrence indexée par chaque entier naturel. On y trouvera donc A, B, X mais encore A_0, A_3 ou encore Z_{127}.

Les éléments de Λ désigneront des *variables propositionnelles*, c'est-à-dire des individus constituant les arguments des connecteurs logiques fonctionnels.

Definition 2.1.1. *On définit inductivement l'ensemble des formules propositionnelles $\mathcal{F}_0$ par :*

- *Les formules propositionnelles atomiques sont les éléments de $\Lambda \cup \{\bot\ ;\ \top\}$*
- *Les constructeurs sont les éléments de $\mathcal{C}_l$*
- *La profondeur est ω.*

Ainsi, si φ et ψ sont des formules propositionnelles, alors :

$$\neg\varphi\ ;\ \varphi \wedge \psi\ ;\ \varphi \vee \psi\ ;\ \varphi \Rightarrow \psi\ ;\ \varphi \Leftrightarrow \psi$$

sont aussi des formules propositionnelles.

Il convient de remarquer que, par souci de clarté, la notation polonaise a été abandonnée. Nous pourrons en revanche nous y ramener à loisir, chaque fois que le formalisme ou la rigueur nous l'imposeront. Il est également proposé au lecteur, en guise d'exercice, de réécrire l'exposé en notation polonaise chaque fois que cette convention a été omise.

Exemple 2.1.2. *Les objets suivants représentent des formules :*

$(A \wedge (\neg B)) \Rightarrow C$

$(\neg Z_4) \Leftrightarrow (\top \vee B_7)$

$((((A_0 \Rightarrow A_1) \Rightarrow A_2) \Rightarrow A_3) \Rightarrow A_4)$

En revanche, $BONJOUR$ ou $A \ \Rightarrow$ ne constituent pas des formules, pas plus que $FORMULE$. On veillera donc bien à dinstinguer Λ^* ou encore $(\mathcal{C} \cup \Lambda)^*$ de l'ensemble, noté $\mathcal{F}_0$, des formules propositionnelles.

Remarque : L'usage des parenthèses peut se révéler fastidieux. Certaines conventions usuelles permettent de s'en passer dans le discours habituel. Nous emploierons ces conventions dès lors que nous aurons pu établir la vallidité de leur usage.

Procédure de vérification

On propose de décrire ici une procédure de vérification de la construction d'une formule. Si $\mathcal{S}$ désigne l'ensemble de tous les symboles utilisés pour construire une formule de $\mathcal{F}_0$, on peut assimiler une entrée de notre procédure à un élément de $\mathcal{S}^*$. Retenons enfin la convention suivante : les éléments atomiques, traités comme des constantes, pourront être considérés d'arité nulle.

Entrée : $(s_i)_{i\leq n} \in \mathcal{S}^*$

Affectations : $\forall i \leq n \;\; val_i := ar(s_i) - 1$

$\forall i \leq n \;\; \sum_{j\leq i} val_j$

Arrêt : Si $\exists i \leq n \;\; \sum_{j\leq i} val_j = -1$

OU fin de séquence

Proposition 2.1.3. *L'algorithme précédant s'arrête en remplissant les deux conditions d'arrêt simultanément si, et seulement si, la séquence d'entrée est un élément de $\mathcal{F}_0$ écrit en notation polonaise (prénexe).*

Démonstration : S'agissant d'un cas d'équivalence, nous prouverons les deux sens.

Supposons $\sigma = (s_i)_{i\leq n} \in \mathcal{S}^*$, pour un certain $n \in \mathbb{N}$, soit un élément de $\mathcal{F}_0$. Prouvons que la procédure s'arrête en remplissant simultanément les deux conditions.

Considérons d'abord le cas où σ est atomique. Il vient que $\sigma = s_1$ d'arité 0 donc $val(s_1) = -1$. L'assertion est vérifiée.

Remarque : Notons que, si la formule est rallongée, elle est de fait incorrecte et l'algorithme termine sans avoir lu la séquence intégralement.

Supposons à présent que l'assertion soit valide pour φ et ψ, deux mots de $\mathcal{S}^*$ qui sont bien des éléments de $\mathcal{F}_0$ (formules). Nous allons montrer que l'assertion reste valide pour les nouvelles formules construites à partir de φ et ψ. Le mot $\neg\varphi$ est bien un élément de $\mathcal{F}_0$ et, de plus, $ar(\neg) = 1$ donc $val(\neg) = 0$. L'algorithme effectue donc le même parcours par la suite que pour la seule entrée φ.

Considérons $*$ un connecteur d'arité 2, donc de valuation 1. L'algorithme parcourt $*\varphi\psi$ en commençant par une valuation de 1, puis en parcourant φ avec 1 point de plus que pour φ seul. A la sortie de la lecture de φ, l'algorithme est à 0 et n'a pas rencontré -1 (sinon, il serait passé par -2 e lisant φ seul). Le parcourt de ψ s'effectue alors comme s'il avait été seul.

Remarque : Notons que, si la formule est rallongée, elle est de fait incorrecte et l'algorithme termine sans avoir lu la séquence intégralement.

La réciproque s'obtient par considération des remarques et en notant qu'une formule est correcte seulement si chaque connecteur employé doit voir apparaître un symbole de valuation -1 par point d'arité. La notation prénexe assure enfin que les connecteurs précédent, d'où le fait que -1 ne doit pas être rencontré avant la fin de séquence.

2.2 Règles de déduction

Nous introduisons ici les règles de déduction formant un système de preuves primitif, noté LK_0. Le choix de ces règles réside dans la volonté double de *construire* un système logique initial ainsi que *respecter* les réflexions historiques ayant initié l'étude de la logique et des démonstrations comme objet à part entière.

Nous introduisons pour cela la notation préséquentielle (avant les séquents) suivante :

$$\frac{\varphi_1 \ \ \varphi_2 \ \ \cdots \ \ \varphi_n}{\psi}[r]$$

traduisant que la formule ψ peut être construite (déduite) à partir de $\varphi_1 \dots \varphi_n$ selon la règle $[r]$.

Vocabulaire : Les formules $\varphi_1, \dots \varphi_n$ sont nommées prémisses et la formule ψ conclusion du préséquent.

On énonce ensuite les règles formant LK_0 :

$$\frac{\varphi \ \Rightarrow \ \psi \ \ \varphi}{\psi}[\text{Modus ponens}]$$

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi}[\wedge - \text{intro}]$$

$$\frac{\varphi}{\varphi \vee \psi}[\vee - \text{intro}]$$

$$\frac{\varphi \ \Rightarrow \ \bot}{\neg\varphi}[\neg - \text{intro}]$$

$$\frac{\psi}{\varphi \Rightarrow \psi}[\Rightarrow - \text{intro}]$$

$$\frac{\varphi}{\top}[\top - \text{intro}]$$

$$\frac{\varphi \ \Rightarrow \ \psi \quad \psi \ \Rightarrow \ \varphi}{\varphi \ \Leftrightarrow \ \psi}[\Leftrightarrow - \text{intro}]$$

Ce système primitif formalise les méthodes de déduction les plus fréquemment employées dans les démonstrations. D'autres peuvent venir et certaines peuvent se confondre ($\neg$-intro et raisonnement par l'absurde par exemple). Nous allons définir la notion de séquent afin de compléter, plus tard, ce système de preuves (voir **logique des prédicats**).

Definition 2.2.1. *On appelle séquent noté $\varphi \vdash \psi$, où φ et ψ sont des formules, la déduction de ψ à partir de φ selon un nombre fini d'applications de règles de déduction préséquentielles.*

Nous verrons plus tard le caractère récursif des séquents en logique des prédicats. Pour le moment, donnons du sens à nos propos.

2.3 Aspect Sémantique

Definition 2.3.1. *On définit une valuation des variables propositionnelles comme une application* $\nu : \Lambda \longrightarrow \{vrai\ ;\ faux\}$.

On définit, de façon associée, une fonction de sémantique :

Definition 2.3.2. *Etant donnée* ν *une valuation des variables propositionnelles, la fonction* $||\ \ ||_\nu : \mathcal{F}_0 \longrightarrow \{vrai; faux\}$ *est définie par induction sur les formules :*

- $||\top||_\nu = vrai$ et $||\bot||_\nu = faux$. De plus, pour tout symbole s de Λ, on a $||s||_\nu = \nu(s)$.
- Si φ désigne un élément de $\mathcal{F}_0$ pour lequel $||\varphi||_\nu$ est connu, alors $||\neg\varphi||_\nu = vrai$ si, et seulement si, $||\varphi||_\nu = faux$.

  Si ψ désigne un autre élément de $\mathcal{F}_0$ pour lequel $||\psi||_\nu$ est connu, alors on a :

  - $||\varphi \wedge \psi||_\nu = vrai$ si, et seulement si, $||\varphi||_\nu = vrai$ et $||\psi||_\nu = vrai$
  - $||\varphi \vee \psi||_\nu = vrai$ si, et seulement si, $||\varphi||_\nu = vrai$ ou $||\psi||_\nu = vrai$
  - $||\varphi \Rightarrow \psi||_\nu = vrai$ si, et seulement si, $||\varphi||_\nu = vrai$ entraîne $||\psi||_\nu = vrai$
  - $||\varphi \Leftrightarrow \psi||_\nu = vrai$ si, et seulement si, $||\varphi||_\nu = vrai$ équivaut à $||\psi||_\nu = vrai$

Ce mode de définition de la sémantique (vérité) est attribué au mathématicien et logicien Tarski. Bien qu'ayant soulevé de nombreuses moqueries, ce mode de définition s'est révélé assez performant dans le traitement des tâches. On retiendra alors la classification proposée par J.Y.Girard de ce type de sémantique en calcul logique, appelée *Broccoli logic* et définie comme toute logique dans laquelle :

$$||\varphi \clubsuit \psi||_\nu = \text{vrai, si et seulement si , } ||\varphi||_\nu = \text{vrai broccoli } ||\psi||_\nu = \text{vrai}$$

Definition 2.3.3. *On dit que* $\varphi \in \mathcal{F}_0$ *est une tautologie lorsque, pour toute* ν*, valuation de variables propositionnelles, on a :*

$$||\varphi||_\nu = vrai$$

On peut de même évoquer la notion d'antilogie :

Definition 2.3.4. *On dit que* $\varphi \in \mathcal{F}_0$ *est une antilogie lorsque, pour toute* ν*, valuation de variables propositionnelles, on a :*

$$||\varphi||_\nu = faux$$

Enfin, on donne une sorte de *troisième cas* qui, formellement, incorporera le premier :

Definition 2.3.5. *On dit que* $\varphi \in \mathcal{F}_0$ *est satisfiable lorsque, il existe* ν*, valuation de variables propositionnelles, telle que :*

$$||\varphi||_\nu = vrai$$

La sémantique des formules permet donc de classifier les formules de la logique propositionnelle en trois catégories disjointes : les antilogies, les tautologies et les formules satisfiables non tautologiques. Nous chercherons à construire une structure calculatoire (algébrique) performante pour déterminer la classe d'appartenance d'une formule et, le cas échéant, déterminer une valuation ν adéquate pour rendre φ satisfiable vraie. Les algèbres de Boole ont été introduites en ce sens (voir la section 'La logique c'est algébrique).

Tables de vérité

Nous introduisons ici la notion de table de vérité en posant *arbitrairement* $0 := vrai$ et $1 := faux$. Nous ne discuterons pas ces notations qui n'ont comme seul but que de nous simplifier la tâche descriptive des tables. Nous adoptons ici le format suivant :

s_1	s_2	...	s_n	φ
0	0	...	0	b_1
⋮	⋮		⋮	
1	1	...	1	b_{2^n}

chaque ligne correspondant à une valuation de valeurs aux variables désignées par les symboles $s_1, s_2, \ldots, s_n$ et où b_i désigne soit 0, soit 1 (en référence aux notations précédantes) pour $1 \leq i \leq 2^n$. Ainsi, on parcourira toutes les combinaisons possibles pour les symboles s_i employés et on omettra les symboles non employés (on gardera cependant à l'esprit qu'ils seront quand même valués, sans que l'on sache comment).

Avec ces notations, on redonne les tables des connecteurs usuels :

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

A	B	$A \Leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

et on se permet de rappeler que, pour le connecteur $\neg$, on obtient tout simplement

A	$\neg A$
0	1
1	0

. Les cas de $\top$ et $\bot$ ne présentent pas d'intérêt.

Remarque : Ces tables de vérités sont génériques. Ainsi, si la table de vérité de $C \Rightarrow D$ est foncièrement identique à celle de $A \Rightarrow B$, on gardera à l'esprit que ce ne sont pas nécessairement les mêmes valuations ν qui génèrent ces tables.

Proposition 2.3.6. *La formule $p \Rightarrow (q \Rightarrow p)$ est toujours valide (tautologie) quelle que soit les symboles p et q de Λ choisis.*

Démonstration : Nous choisissons d'avoir recours à une table de vérité :

p	q	$q \Rightarrow p$	$p \Rightarrow (q \Rightarrow p)$
0	0	1	1
0	1	0	1
1	0	1	1
1	1	1	1

Cette preuve est alors dite *sémantique*

Definition 2.3.7. *Deux forumles φ et ψ sont dites sémantiquement équivalentes lorsqu'elles ont mêmes tables de vérité. On notera $\varphi \equiv \psi$.*

En d'autres termes, les formules φ et ψ sont satisfaites en de mêmes valuations. On retiendra lors que malgré lesapparences, les formules $A \Rightarrow B$ et $C \Rightarrow D$ ne sont pas sémantiquement équivalentes car les tables de vérité ne sont pas construites avec les mêmes valuations.

Proposition 2.3.8. *On a que $\varphi \equiv \psi$ si, et seulement si, $\varphi \Leftrightarrow \psi$ est une tautologie.*

Ceci signifie que l'équivalence sémantique est méta-équivalente à la validité de l'équivalence syntaxique.

Démonstration : Si φ et ψ sont sémantiquement équivalentes, alors $||\varphi||_\nu = ||\psi||_\nu$ pour toute ν, valuation. Ainsi, on a $||\varphi||_\nu = vrai$ équivalent à $||\psi||_\nu = vrai$ par binarité des valeurs de vérité ce qui, par définition de la sémantique (à la Tarski) nous amène à conclure que $||\varphi \Leftrightarrow \psi||_\nu = vrai$ (pour toute ν). La fomule $\varphi \Leftrightarrow \psi$ remplit donc la définition d'une tautologie.

Réciproquement, supposons que $\varphi \Leftrightarrow \psi$ soit une tautologie. Alors, pour toute ν valuation, on a $||\varphi||_\nu = vrai$ équivalent à $||\psi||_\nu = vrai$ (définition à la Tarski). On note $s_1, \dots s_n$ les symboles de Λ présents dans φ ou dans ψ. Soit ν une valuation telle que $\nu(s_i) = b_i$, pour $i \leq n$. Dans la ligne correspondant à $(b_1\ b_2\ \dots\ b_n)$ on lit, dans la table de vérité de φ une valeur de vérité t identique à celle de ψ d'après ce qui précède. Ceci est alors valable dans toute ligne. Si un symbole s n'apparaît ni dans φ, ni dans ψ, alors sa valuation ne change pas la valeur de vérité de l'une ou l'autre des formules. En conclusion, φ et ψ ont mêmes tables de vérité. Les formules sont donc bien sémantiquement équivalentes.

Exemple : On a en particulier que $\varphi \equiv \top$ si, et seulement si, φ est une tautologie.

La nature de la relation $\equiv$ et les règles de calcul induites seront décrites dans la section **La logique c'est algébrique**.

Proposition 2.3.9 (Lois de De Morgan)**.** *Pour tous symboles p et q de Λ on a :*

$$\begin{aligned} \neg(p \wedge q) &\equiv \neg p \vee \neg q \\ \neg(p \vee q) &\equiv \neg p \wedge \neg q \end{aligned}$$

Proposition 2.3.10 (Contrapposition)**.** *Pour tous symboles p et q de Λ on a :*

$$p \Rightarrow q \equiv \neg q \Rightarrow \neg p$$

Proposition 2.3.11 (Material implication)**.** *Pour tous symboles p et q de Λ on a :*

$$p \Rightarrow q \equiv \neg p \vee q$$

La vérification de ces trois propriétés classiques est laissée en guise d'exercice.

3 La logique c'est algébrique

Dans cette section, nous cherchons à décrire algébrique le calcul propositionnel, ce qui justifiera l'emploi du mot *calcul*. Dans un premier temps, nous abordons les relations d'équivalences, qui induisent des congruences, c'est-à-dire une égalité dans une structure où l'on ignore certains aspects d'objets distincts de sorte à les confondre. On réfléchira au fait que, fondamentalement, tout calcul repose sur cet appauvrissement de langage : $2+2=4$ dans la mesure où l'on ignore la possibilité de dinstinguer $2+2$ de 4 au sens des expressions alégébriques. L'arithmétique est donc une théorie algébrique dans laquelle l'égalité est induite par une congruence d'expressions algébriques. Les procédures de vérification syntaxiques ne travaillent donc pas dans ce type de structures, c'est pourquoi il faut garder à l'esprit l'information "ignorée" lors d'un traitement calculatoire.

Que le lecteur se rassure, la clarté viendra en même temps que les notions évoquées ci-avant seront abordées et définies.

3.1 Relations d'équivalence

On rappelle qu'une relation binaire R est un objet reliant des couples $(x;y)$ d'individus généralement pris dans un même ensemble E. Il existent quatre grandes façons de représenter R :

a) Par formule logique du 1er ordre (voir la section correspondante).

   On définit alors R par : xRy SSI $\varphi(x;y)$ est vraie, où la formule φ admet deux (symboles de) variables libres.

b) Par graphe.

   On définit alors R sur E, ensemble de sommets, comme l'ensemble des arêtes reliant deux sommets s_1 et s_2 si, et seulement si, les éléments s_1 et s_2 vérifient s_1Rs_2.

c) Par fonction booléenne.

   On a alors $\mathbb{1}_R(x;y)=1$ si, seulement si, xRy où $\mathbb{1}_R : E^2 \longrightarrow \{0;1\}$.

d) Par sous-ensemble de carré cartésien.

   On a alors $E_R \subset E^2$ avec $(x;y) \in E_R$ si, et seulement si, xRy.

On définit alors :

Definition 3.1.1. *Une relation binaire R sur E est dite* relation d'équivalence *lorsqu'elle satisfait les conditions suivantes :*

i) (Réflexivité) $\forall x \in E \;\; xRx$

ii) (Symétrie) $\forall x \in E \forall y \in E \;\; xRy \;\Rightarrow\; yRx$

iii) (Transitivité) $\forall x \in E \forall y \in E \forall z \in E \;\; (xRy \wedge yRz) \;\; \Rightarrow xRz$

Nous donnons les exemples les plus classiques :

Exemple 3.1.2. *Sur $E = \mathbb{Z}$, on donne $R = \equiv_n$ avec $n \in \mathbb{N}^*$ définie par la formule logique du premier ordre :*

$$a \equiv_n b \quad \Leftrightarrow \quad \exists k \in \mathbb{Z} \ \ a - b = kn$$

Cette relation d'équivalence définit la congruence modulo n.

Exemple 3.1.3. *On donne $f : E \longrightarrow F$ une application surjective. On définit R_f sur l'ensemble E par le sous-ensemble de E^2 :*

$$R_f = \{(x; y) \in E^2 \mid f(x) = f(y)\}$$

Cette relation d'équivalence regroupe ensemble les antécédants d'une même image.

Exemple 3.1.4. *Dans le cadre du calcul propositionnel, $\varphi \equiv \psi$ pour φ et ψ des formules de la logique propositionnelle, désigne une relation d'équivalence sur l'ensemble $\mathcal{F}_0$.*

Exercice : Démontrer que les exemples qui précèdent satisfont bien aux conditions des relations d'équivalence.

Proposition 3.1.5. *Il existe une correspondance bijective entre les partitions d'un ensemble et les relations d'équivalence de cet ensemble.*

On rappelle alors qu'une partition d'un ensemble E est une famille $(A_i)_{i \in I}$ de sous-ensembles de E dont l'union est E et dont les intersections, prises deux à deux, sont vides. Il est alors possible d'écrire $\biguplus_{i \in} A_i = E$ pour désigner ce fait.

Démonstration : Donnons-nous une partition $(A_i)_{i \in I}$ de E. Définissons alors une relation R par :

$$R = \{(x, y) \in E^2 \mid \exists i \in \ \ x \in A_i \ \wedge \ y \in A_i\}$$

La réflexivité et la symétrie sont triviales. Montrons la transitivité : prenons $(x; y; z) \in E^3$ tels que xRy et yRz, c'est-à-dire $(x; y) \in R$ et $(y; z) \in R$. Nous avons alors que, pour un certain $i \in I$ on a $x \in A_i$ et $y \in A_i$ ainsi que, pour un certain $j \in I$, on a $y \in A_j$ et $z \in A_j$. Ainsi, $y \in A_i \cap A_j$. Mais $(A_i)_{i \in I}$ étant une partition de E, ceci implique que $A_i = A_j$ ou que l'un d'eux est vide (ce dernier point n'étant pas admissible : nous prenons des éléments dedans !) Il vient alors que $z \in A_i$ donc $(x; z) \in R$ par construction.

Réciproquement, si R désigne une relation d'équivalence sur E, nous construisons les ensembles :

$$A_x = \{y \in E \mid xRy\}$$

pour chaque $x \in E$. Nous prétendons que $y \in A_x \ \Rightarrow \ A_y = A_x$ par les propriétés de la relation d'équivalence R. Enfin, ne conservons qu'une seule copie de chaque ensemble ainsi construit (élimination des doublons). L'ensemble obtenu induit une partition. En effet, de ce qui précède, on a $A_x \cap A_t \neq \emptyset \ \Rightarrow \ A_x = A_t$ (en considérant tout α dans l'intersection) ce qui nous ramène à un doublon. De plus, par réflexivité, tout $x \in E$ valide xRx donc appartient bien à un A_{x_0} tel que

$A_{x_0} = A_x$ (celui dont on a conservé la copie).

Vocabulaire : On appelle *classe d'équivalence* de x pour R l'ensemble A_x définit ci-dessus. On écrit parfois aussi $\bar{x}$.

Definition 3.1.6. *L'ensemble formé des seules classes d'équivalence est appelé* ensemble quotient *de E par R, souvent noté E/R.*

Il devient égalemment possible de remplacer R par la partition $\mathcal{P}$ à laquelle R est associée.

Exemple 3.1.7. *L'ensemble $\mathbb{Z}/\equiv_n$ représente l'ensemble modulaire des classes $\bar{0}, \bar{1}, \ldots, \bar{n-1}$ souvent assimilé à $\{0, 1, \ldots, n-1\}$.*

Ce ensemble est souvent noté $\mathbb{Z}n\mathbb{Z}$ car $\bar{a} = a + n\mathbb{Z}$ pour tout $a \in \mathbb{Z}$.

En reprenant l'exemple 3.1.3. on peut constater que $f : E/R_f \longrightarrow F$ est toujours une bijection.

Notons que les fonctions peuvent être naturellement employées avec des parties : ainsi $f(A_x)$ avec $A_x \subset E$ est bien définie même si f est définie depuis E. Du point de vue du typage, en revanche, ce n'est pas valide. On pourra donc imposer denoter distinctement f lorsqu'employé avec des parties.

Definition 3.1.8. *L'ensemble $\mathcal{F}_0/\equiv$ quotient des formules propositionnelles par la relation d'équivalence sémantique est appelé algèbre de Boole primitive. On le notera $\mathbb{A}$.*

Remarque : Attention, Ce n'est ni une algèbre, ni même un anneau au sens moderne de ces termes.

Les opérations sur $\mathbb{A}$ sont définies par transmissions des opérateurs logiques à leurs classes d'équivalence (comme$+$ et $\times$ le sont à la structure quotient $\mathbb{Z}/n\mathbb{Z}$). On renotera en revanche ces opérateurs afin de prendre conscience de l'algébrisation ainsi obtenue :

a) Le connecteur $\wedge$ sera noté $\cdot$ lorsqu'il agit sur les classes d'équivalence de $\mathcal{F}_0$, c'est-à-dire les éléments de $\mathbb{A}$.

b) Le connecteur $\vee$ sera noté $+$ lorsqu'il agit sur $\mathbb{A}$.

c) Le connecteur $\neg$ sera désigné par un surlignage lorsqu'il agit sur $\mathbb{A}$.

d) La classe d'équivalence de $\top$ sera notée 1 dans $\mathbb{A}$

e) La classe d'équivalence de $\bot$ sera notée 0 dans $\mathbb{B}$

On désignera par des lettres minuscules des éléments de $\mathbb{A}$.

Proposition 3.1.9. *Si x et y désignent les classes de φet ψ respectivement dans $\mathbb{A}$, alors $\varphi \Rightarrow \psi$ admet $\bar{x} + y$ comme représentant.*

Démonstration En considérant la *material implication*

Cette structure autorise maintenant le traitement calculatoire des opérations logiques. Il faudra pour cela déterminer les propriétés qui s'en dégagent. La section suivante propose de rappeler la classification usuelle des structures algébriques de référence.

3.2 Monoïdes et Groupes

On se donne un ensemble E et un opérateur $* : E \times E \longrightarrow E$ dit *loi de composition interne*. On écrira $x * y$ au lieu de $*(x; y)$ comme l'usage le veut avec les opérateurs.

Definition 3.2.1. *On dit que $(E; *)$ est un monoïde lorsque les conditions suivantes sont satisfaites :*

i) (Neutre) $\exists e \in E \ \ \forall x \in E \ \ e * x = x * e = x$.

ii) (Associativité) $\forall x \in E \, \forall y \in E \, \forall z \in E \ \ x * (y * z) = (x * y) * z$

Nous noterons e_G l'élement neutre de $(E; *)$. La propostion suivante justifie cette désignation.

Proposition 3.2.2. *Il existe un unique élément de E satisfaisant la condition $i)$ si $(E; *)$ est un monoïde*

Démonstration: L'existence de e est assurée. Supposons qu'il existe $f \in$ satisfaisant $i)$ pareillement. Nous avons $e * f = e$ en utilisant $i)$ pour f et aussi $e * f = f$ en utilisant $i)$ pour e. D'où $f = e$.

Definition 3.2.3. *On dit que $(E; *)$ est un groupe lorsque c'est un monoïde de neutre e_G satisfaisant :*

(Inversibilité) $\forall x \in E \, \exists y \in E \ \ x * y = y * x = e$

On dira que y, défini à partir de x est l'inverse de x et on le notera x^{-1}. Souvent, si $*$ est une addition $(+)$ on écrit $-x$ au lieu de x^{-1}. La proposition suivante justifie ces désignations :

Proposition 3.2.4. *Pour chaque $x \in E$, l'élément y décrit par la condition d'inversibilité est unique.*

Démonstration: Un exercice à effectuer.

Exemple 3.2.5. *Si Σ désigne unensemble de caractères (alphabet), alors $(\Sigma^*; @)$ est un monoïde, $@$ désignant la concaténation. Ce n'est en revanche pas un groupe.*

Exemple 3.2.6. *Soit E un ensemble fini à n éléments. On note $\mathbb{S}_E$ l'ensemble des bijections de E dans E, appelées permutations de E. Alors $(\mathbb{S}_E; \circ)$ est un groupe, $\circ$ désignant la loi de composition des applications.*

Ce dernier exemple constitue le premier exemple histotique de groupe répertorié, c'est même l'exemple qui justifia l'introduction de la notion par Evariste Galois. Les monoïdes n'ont été définis que bien plus tard (un bon siècle après).

Definition 3.2.7. *Soit $G = \{g_1, \ \ldots, g_n\}$ un ensemble fini et $*$ un opérateur algébrique à deux arguments défini sur G (techniquement, G^2.) On appelle* table de $*$ *la matrice $M_* = (g_i * g_j)_{i \leq n \ , \ j \leq n}$.*

Il est commode de travailler avec la matrice M_* du point de vue informatique pour tester les propriétés algébriques de la structure $(G, *)$, le premier des tests étant celui du caractère interne de la loi.

Théorème 3.2.8. *Il existe un algorithme en temps $\mathcal{O}(n^3)$ qui teste si $(G, *)$ est un groupe.*

Ainsi, *" Est un groupe fini ?"* est un problème P au sens de la complexité.

Démonstration : On décrit un tel algorithme :

- [Interne] Tester si $\forall i \leq n \forall j \leq n \quad g_i * g_j \in G$.
- [Neutre] Tester si $\exists i \leq n \quad \text{ligne}_i(M_*) = (g_1 \dots g_n) \ \wedge \ \text{colonne}_i(M_*) = {}^t(g_1 \dots g_n)$.
- [Inversibilité] Tester si $\forall i \leq n \ \exists (\sigma; \tau) \in \mathbb{S}_n^2 \ \text{ligne}_i(M_*) = \sigma.(g_1 \dots g_n) \ \wedge \ \text{colonne}_i(M_*) = \tau.\, {}^t(g_1 \dots g_n)$
- [Associativité] On pose $t : (i; j) \mapsto k$ indice tel que $g_k = g_i * g_j$. On construit $\varphi : (i; j; k) \mapsto (M_*)_{t(i;j),k}$ et $\psi : (i; j; k) \mapsto (M_*)_{i,t(j;k)}$.
  Tester si $\forall i \leq n \ \forall j \leq n \ \forall k \leq n \quad \varphi(i; j; k) = \psi(i; j; k)$.

Il est laissé au lecteur le soin d'exhiber une séquence d'instruction prouvant le caractère cubique du test d'inversibilité.

En guise d'exemple, on propose la table de $Z/5\mathbb{Z}$. Le lecteur pourra vérifier qu'il s'agit d'un groupe, l'énumération étant $g_k = k - 1$, pour $1 \leq k \leq 5$.

$$M_+ = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \end{pmatrix}$$

Anneaux

La structure anneau a été introduite par Dedekind. Elle est très utilisée en arithmétique. Elle généralise les travaux algébriques menés dans le cadre des polynômes. En effet, elle requiert deux opérations, souvent assimilées à une addition et une multiplication. Elle se fonde sur les groupes comme on pourra le constater :

Definition 3.2.9. *Un ensemble A muni de deux opérateurs algébriques $+$ et $\times$ forment une structure $(A; +; \times)$ appelée* anneau *lorsque :*

- *$(A, +)$ est un groupe abélien*
- *l'opération $\times$ est associative*

- *l'opération $\times$ est distributive sur $+$:*

$$\forall x \in A \forall y \in A \forall z \in A \quad x \times (y+z) = x \times y + x \times z \quad \wedge \quad (y+z) \times x = y \times x + z \times x$$

On pourra utiliser la notation usuelle xy au lieu de $x \times y$ lorsque le contexte le permet. Ainsi, toute expression (polynomiale) :

$$a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_0$$

prend sens dans un anneau. On notera $A[X]$ l'ensemble des expressions polynômiales d'indéterminée X à coefficients dans l'anneau A. Attention cependant : la nature de la structure $(A[X], +, \times)$ dépend des propriétés éventuellement supplémentaires de A ainsi que de l'interprétation que l'on ferait de X.

Exercice : La structure $(A[X], +, \times)$ est-elle elle-même un anneau ?

Proposition 3.2.10. *L'élément neutre e_A du groupe $(A, +)$ d'un anneau est absorbant pour $\times$ c'est-à-dire :*

$$\forall a \in A \quad a \times e_A = e_A \times a = e_A$$

On notera alors 0_A voire 0 si le contexte est clair l'élément e_A.

Démonstration : Prenons a et $-a$ l'inverse de a pour $+$, unique. On a $a + (-a) = 0_A$ d'après nos notations. D'où :

$$0_A \times a = (a + (-a)) \times a = a^2 + (-a^2) = 0_A$$

même procédure à droite en développant $a \times 0_A$.

On pourra noter $a - b$ au lieu de $a + (-b)$. On peut montrer que $a \times (-b) = (-a) \times b = -(a \times b)$ pour former dans tout anneau une règle des signes.

Les anneaux peuvent disposer de propriétés supplémentaires :

- Un anneau est dit *unitaire* lorsqu'il existe un élément de A noté 1_A vérifiant : $\forall a \in A \quad 1_A \times a = 1_A \times a = a$
- Un anneau est dit *idempotent* lorsqu'il vérifie : $\forall a \in A \quad a^2 = a$
- Un anneau est dit *intègre* lorsqu'il vérifie : $\forall a \in A \forall b \in A \quad a \times b = 0 \quad \Rightarrow \quad a = 0 \vee \ b = 0$
- Un anneau est dit *commutatif* lorsque la loi $\times$ est elle-même commutative.

On retiendra enfin qu'un élément a est dit *inversible* lorsqu'il l'est pour la loi $\times$. En effet, tout élément l'étant pour $+$, la confusion est évitée. La notation a^{-1} est réservée à l'inverse de a pour $\times$ (si existence). On utilise généralement *opposé* pour désigner $-a$ l'inverse de a au sens de $+$.

Exemple 3.2.11. *Les structures usuelles* $(\mathbb{Z}; +; \times)$, $(\mathbb{R}; +; \times)$, $(\mathbb{C}; +; \times)$ *sont des anneaux. Pour* $n \in \mathbb{N}$*, la structure* $(\mathbb{Z}/n\mathbb{Z}; +; \times)$ *est aussi un anneau (dit quotient) formant un arithmétique dite modulaire.*

On accepte que ces structures sont des anneaux. Le lecteur pourra vérifier que chacune est unitaire et commutative. La structure modulaire est intègre si, et seulement si, n est un nombre premier (par le théorème de Bézout). Les autres sont intègres.

On note A^* l'ensemble des inversibles d'un anneau. On prouve que 0_A n'est jamais dans A^*. Il est clair que, si A est unitaire, $(A^*; \times)$ est un groupe nommé *groupe des inversibles*.

Exemple 3.2.12. *Les ensembles* $\mathcal{M}_n(\mathbb{R})$ *et* $\mathcal{M}_n(\mathbb{C})$ *sont des anneaux pour les opérations matricielles* $+$ *et* $\times$ *usuelles.*

Exercice : Vérifiez que ces anneaux sont unitaires, mais ni commutatifs ni intègres.
On a en particulier que $\mathcal{M}_n(\mathbb{R})^* = GL_n(\mathbb{R})$ (groupe linéaire) et analogue avec $\mathbb{C}$.

Proposition 3.2.13. *L'ensemble quotient* $\mathbb{A}$ *muni des opérations* $\oplus$ *(ou-exclusif) et* $\cdot$ *(conjonction) forme un anneau unitaire commutatif et idempotent.*

La démonstration est laissée en guise d'exercice. Chaque point à démontrer provient d'un élément du calcul propositionnel. On peut vérifier que l'anneau n'est pas intègre. En effet, $a \cdot \bar{a} = 0$ sans que cela n'implique que l'un ou l'autre de a ou $\bar{a}$ ne soit nul (comprendre "toujours faux").
Nous rappellons que $A \wedge \neg A$ a pour classe d'équuivalence $a \cdot \bar{a}$ dans $\mathbb{A}$.

Definition 3.2.14. *On appelle* anneau booléen *(ou de Boole) tout anneau unitaire commutatif idempotent.*

Ce type d'anneaux généralise donc le calcul(quotient) propositionnel. Il constitue le modèle mathématique de référence des circuits logiques (classiques).

Exemple 3.2.15. *L'anneau* $(\mathbb{A}; \oplus; \cdot)$ *est évidemment un anneau de Boole.*

Exemple 3.2.16. *L'anneau* $(\mathbb{Z}/2\mathbb{Z}; +; \times)$ *est aussi un anneau de Boole. Il est identifiable à* $(\{vrai; faux\}; X - or; et)$.

Exemple 3.2.17. *Tout ensemble* E *engendre un anneau de Boole au moyen de l'ensemble* $\mathcal{P}(E)$ *de ses parties muni des opérations* Δ *(différence symétrique) et* $\cap$ *(intersection).*

Lorsque *tout va bien*, on peut aller jusqu'à définir un corps :

Definition 3.2.18. *On appelle coprs* $(\mathbb{K}; +; \times)$ *un anneau (unitaire) tel que :*

- $\mathbb{K}^* = \mathbb{K} \backslash \{0\}$

- $(\mathbb{K}^*;\times)$ *est un groupe (abélien)*[2]

Un corps est une structure de calcul permettant d'exécuter des opérations *sans problème*, comme en fin d'école élémentaire. On peut résoudre les équations linéaires par manipulations algébriques classiques, comme au collège.

Exemple 3.2.19. *Les structures* $(\mathbb{R};+;\times)$, $(\mathbb{C};+;\times)$ *sont des corps. De même pour* $(\mathbb{Q};+;\times)$

En revanche, $(\mathbb{Z};+;\times)$ ni même $(\mathbb{D};+;\times)$ n'en sont.

Proposition 3.2.20. *L'anneau* $(\mathbb{Z}/n\mathbb{Z};+;\times)$ *est un corps si, et seulement si,* n *est un nombre premier.*

*Démonstration :*Il suffit de vérifier que $A^* = \{\bar{1};\bar{2};\ldots;\overline{n-1}\}$ si, et seulement si, n est premier par le théorème de Bézout.

Relation d'ordre

Les relations binaires vues précedemment ne servent pas qu'à fabriquer des graphes ou décrire des relations d'équivalence. Elles sont aussi employées pour comparer des objets, c'est-à-dire, déterminer le plus grand ou le plus petit. Selon un critère défini, l'impossibilité de comparer doit également pouvoir être établi.

Definition 3.2.21. *Une relation binaire* $\leq$ *est un ordre (large) sur* E *lorsque les axiomes de réflexivité, de transitivité sont satisfaits, ainsi que l'axiome suivant :*

- *(Anti-symétrie) :* $\forall x \in E \forall y \in E \quad x \leq y \wedge y \leq x \;\Rightarrow\; x = y$

De façon alternative, on peut associer une relation $\equiv$ d'équivalence à un ordre large et remplacer l'égalité $=$ par une congruence $\equiv$. On note alors que, de toute façon, en se plaçant dans l'espace quotient, on obtient la même chose. De plus, tout ensemble $(E;=)$ égalitaire est en fait le quotient d'un autre $(\mathcal{E};\equiv)$ par sa relation d'équivalence, cette dernière pouvant être choisie non triviale (chaque classe n'est pas un singleton).

Exemple 3.2.22. *L'ordre naturel sur les nombres de* $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$.

Exemple 3.2.23. *La relation* $a|b$ *de divisibilité est un ordre sur* $\mathbb{N}$.

Exemple 3.2.24. *La relation* $a \subset b$ *d'inclusion est un ordre sur*

$$\mathcal{P}(E)$$

.

[2]ce dernier point se discute: s'il est fréquemment mentionné dans la littérature, la description des quaternions $\mathbb{H}$ fournit un exemple de corps non commutatif. Le statut de la commutativité de $\times$ d'un corps n'est donc pas totalement clair dans le consensus.

Les ordres ont aussi leurs propriétés :

- Un ordre est *total* lorsque $\forall x \in E \forall y \in E \;\; x \leq y \;\vee\; y \leq x$
- Un ordre est *dense* lorsque $\forall x \in E \forall y \in E \exists z \in E \;\; x \leq z \;\wedge\; z \leq y$

Un ordre non total est dit partiel. C'est le cas de la divisibilité ou de l'inclusion. L'ordre sur $\mathbb{Q}$ ou $\mathbb{R}$ est dense. L'ordre sur $\mathbb{Z}$ est total mais non dense.

On appelle *ordre dual* la relation $\geq$ vérifiant $x \geq y \;\; \Leftrightarrow \;\; y \leq x$ (définie sur le même ensemble que $\leq$). Il est aisé de vérifier que l'ordre dual est un ordre.

Exemple 3.2.25. *Classer une liste de candidats par rang, c'est ordonner duallement selon les scores obtenus.*

Nous rappelons un vocabulaire qui sera utilisé pour définir nos structures :

Vocabulaire : On considère un ensemble E ordonné par $\leq$. On désigne par F une partie non vide de E.

- On appelle *majorant* de F un élément $M \in E$ tel que $\forall x \in F \;\; x \leq M$.
- On appelle *borne supérieure* de F un majorant S de F tel que :

$$M \text{ : majorant de } F \;\; \Rightarrow \;\; S \leq M$$

- On appelle *maximum* de F une borne supérieure $S \in F$.

On démontre que les bornes supérieures et maximaux sont uniques *s'ils existent* (ce qui n'est pas certain). Les majorants sont, en revanche, rarement uniques.

On définit les minorants, bornes inférieures et minima de façon analogue à ci-dessus en appliquant le vocable qui précède à l'ordre dual. Ainsi, par exemple, un minorant de F est un majorant de F pour l'ordre dual $\geq$.

On rappelle que $\mathbb{R}$ possède le propriété de la borne supérieure (et inférieure) : toute partie non vide majorée (resp. minorée) possède une borne supérieure (resp. inférieure).

Definition 3.2.26. *Un treillis est une structure $(E; \leq)$ ordonnée telle que toute paire $\{a; b\} \subset E$ possède une borne supérieure et une borne inférieure.*

Un treillis est dit borné *lorsqu'il admet un élément minimal et un élément maximal.*

Un treillis est dit complet *lorsque tout sous-ensemble admet une borne supérieure et une borne inférieure.*

Exemple 3.2.27. *Par les définitions, il est clair que $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty;\; -\infty\}$ est un treillis borné et complet, là où $\mathbb{R}$ est un simple treillis.*

Exemple 3.2.28. *La structure $(\mathcal{P}(E); \subset)$ est un treillis borné complet.*

Les opérations de bornes supérieures et inférieures dans ce treillis sont données respectivement par $\sup(A_i) = \cup A_i$ et $\inf(A_i) = \cap A_i$.

Proposition 3.2.29. *La structure quotient $\mathcal{F}_0/\equiv$ muni de la relation $\vdash$ de déduction est un treillis borné mais non complet*

On propose la démonstration en exercice.

On notera que la non-complétude de ce treillis provient de l'impossibilité d'écrire une "formule" du type $A_1 \wedge A_2 \wedge \dots$ à l'infini, contrairement à l'homologue ensembliste $A_1 \cup A_2 \cup \dots = \bigcup A_i$.

Notation : On écrira $\bigwedge_{i=1}^{n} A_i$ pour désigner la formule $A_1 \wedge A_2 \wedge \dots \wedge A_n$. Cette désignation est licite dans la mesure où une telle conjonction est bien une formule telle que définie, dès lors que les A_i sont elles-mêmes des variables propositionnelles (ou des formules).

Logique des prédicats

Nous allons définir une logique plus fine en introduisant les quantificateurs. On constatera que $\forall x \;\; A(x)$ est une façon de dire $A(x_1)$ et $A(x_2)$ etc...

On assimilera de même le quantificateur "il existe" à une disjonction (potentiellement infinie). Il est intéressant d'étudier la structure obtenue afin de savoir s'il s'agit bien de la même chose *par extension à l'infini*.

Symboles

On considère l'ensemble $\mathcal{C}$ des connecteurs de la logique propositionnelle. L'ensemble $\mathcal{V}$ désignera l'ensemble des littéraux, c'est-à-dire des symboles de variables. Dans cette section, nous adoptons la convention, cette fois-ci,d'emploi des symboles minuscules afin de distinguer l'usage des logiques ainsi définies (en méta-contexte donc). Enfin, on adjoint un nouvel ensemble commun aux logiques des prédicats : l'ensemble $\mathcal{Q} = \{\forall\,;\,\exists\}$ des quantificateurs (resp. universel et existentiel).

On se donne un ensemble de symboles $\mathcal{L}$ spécifique : il contient des symboles différents selon le langage qui sera considéré. Ainsi, à la différence de la logique propositionnelle, la logique des prédicats n'est pas universelle : un choix de langage est opéré par l'utilisateur. Ce fragment spécifique sera appelé *langage relationnel*. L'ensemble $\mathcal{L}$ est contitué de symboles de trois types :

- Des symboles du type *constante*
- Des symboles du type *fonction*, chacun étant adjoint d'un nombre (entier fini) d'arguments
- Des symboles du type *relation*, chacun étant adjoint d'une arité : un nombre (entier fini)

Un langage relationnel $\mathcal{L}$ est constitué d'autant de symboles de chaque type qu'indiqué(y compris zéro ou une infinité).

Termes

Les termes d'un langage relationnel $\mathcal{L}$ sont définis inductivement :

- Atomiques : Tout symbole de variable ou de constante est un terme de $\mathcal{L}$.
- Constructeurs : Si $\propto$ est un symbole de fonction à n arguments et $\flat_1 \ ; \ \ldots \ ; \flat_n$ sont destermes de $\mathcal{L}$ alors $\propto \flat_1 \ldots \flat_n$ est un terme de $\mathcal{L}$ (en polonais)
- Arrêt : On peut appliquer un nombre de fois entier fini les règles de construction.

On dit qu'un terme est *clos* lorsqu'il ne comprend aucun symbole de variable.

Exemple 3.2.30. *Dans le langage $\mathcal{L} = \{0; 1; +; \times; \equiv\}$ des anneaux, les symboles 0 et 1 sont des symboles de constantes; les symboles $+$ et $\times$ sont des symboles de fonction à deux arguments ; le symbole $\equiv$ est un symbole de relation d'arité deux.*

Les mots (au sens de Kleen) $\varpi_1 = ++\times+01001$ et $\varpi_2 = \times+000$ sont des termes clos (écrits en polonais). Le mot $x + y$ est un terme non clos (écrit en notation classique).
En revanche, $2 + x$ n'est pas un terme de langage, pas plus que ${}^tA \cdot A = 1$.

Exercice : On attribue une valeur de $k - 1$ aux symboles de fonctions à k arguments. Les symboles de constante et de variables prennent alors une valeur de -1. On considère un terme écrit sans abus de notation (c'est à dire que même pour les fonctions à deux arguments $\odot$, on écrit $\odot\flat_1\flat_2$).
Montrer qu'une concaténation de caractères pris dans $\mathcal{V}$ et $\mathcal{L}$ relationnel sans symbole de relation est un terme si, et seulement si, la somme des valeurs des symboles du terme est -1 et, si l'on effectue l'opération d'addition dans le sens de lecture, le sous-total est toujours positif sauf pour le résultat final.

Il vient de cette caractérisation une manière intéressante de vérifier algorithmiquement qu'un terme est bien écrit. On fera le parallèle avec la procédure définie pour la vérification de la construction d'un formule de la logique propositonnelle.

Formules

On définit les *formules* de $\mathcal{L}$ de la même manière :

- Atomiques : Si $\flat_1, \flat_2, \cdots, \flat_n$ sont des termes de $\mathcal{L}$, et que $\diamond$ est un symbole relationnel d'arité n de $\mathcal{L}$, alors $\diamond\flat_1\flat_2\cdots\flat_n$ est une formule.
- Constructeurs : Si ϕ et ψ sont des formules de $\mathcal{L}$, alors $\neg\phi,\ \phi \wedge \psi,\ \phi \vee \psi,\ \phi \Rightarrow \psi,\ \phi \Leftrightarrow \psi$ le sont aussi.

  Par ailleurs, si x est un symbole de variable et que ϕ est une formule de $\mathcal{L}$, alors $\forall x\phi$ et $\exists x\phi$ en sont aussi.

On demandera cependant que ϕ ne contiennent pas déjà un quantificateur suivi du même symbole de variable que celui que l'on introduit. Ainsi,$\exists y \forall x\ x \odot y \equiv y \odot x$ est une formule tandis que $\exists x \forall x\ x \odot y \equiv y \odot x$ ne sera pas considérée comme telle.

- Arrêt : On peut répéter un nombre entier fini de fois l'utilisation des règles de construction.

De même que pour l'écriture des termes fonctionnels, on pourra se permettre d'écrire $\diamond(\flat_1, \flat_2, \cdots, \flat_n)$ au lieu de $\diamond \flat_1 \flat_2 \cdots \flat_n$. Dans le cas où $\diamond$ est à deux arguments, on pourra éventuellement écrire $\flat_1 \diamond \flat_2$ (comme dans $f(x) < y$).

On aura l'habitude de noter $\mathcal{F}(\mathcal{L})$ l'ensemble des formules de $\mathcal{L}$. C'est un sous-ensemble de l'ensemble des mots sur $\mathcal{L} \cup \mathcal{L}_0 \cup \mathcal{V}$. En ce sens, le nombre d'application de ces règles de construction est *fini*.

3.3 Occurrences, variables libres et liées

On dit que x a une *occurrence* dans ϕ si ϕ est une formule où le symbole de variable x apparaît au moins une fois. Si l'une de ces apparitions est précédée d'un symbole de quantificateur logique, on dira aussi qu'elle est *quantifiée*. On appelle alors *champ de quantification* toutes les positions situées à droite du quantificateur. Un symbole de variable est *lié* dans ϕ s'il est quantifié et que toutes ses occurrences sont situées dans un champ de quantification. Dans le cas contraire, la variable admet des occurrences *libres*; on dit de plus qu'elle est *libre* si aucune de ses occurrences n'est quantifiée (dans un champ de quantification). ϕ est dite libre si au moins l'une des occurrences d'une de ses variables est libre.

Exemples : La formule $Px \wedge \forall x\ x \equiv x$ est libre, x est quantifié et possède une occurrence libre. x est non lié et non libre.

La formule $Py \wedge \exists x\ x \equiv y$ est libre, x est quantifiée liée et non libre tandis que y est libre non quantifiée.

La formule $\forall x \exists y\ x \equiv y$ est non libre, x et y sont quantifiées et liées non libres.

On notera $\phi[u_1, \cdots, u_n]$ pour dire que les symboles de variables $u_1, \cdots, u_n$ admettent une occurence dans la formule ϕ. Cela ne voudra en revanche pas dire que ce sont les seuls.

On notera $\phi(x_1, \cdots, x_n)$ pour dire que les symboles de variables $x_1, \cdots, x_n$ admettent tous une occurrence libre dans la formule ϕ. Cela ne voudra pas dire que ce sont les seuls.

Si l'on écrit $\phi[t_1/u_1, \cdots, t_n/u_n, v_1, \cdots, v_k]$, on sous-entendra que l'on considère la formule obtenue à partir de ϕ en remplaçant toute apparition des symboles de variables $u_1, \cdots, u_n$ (alors supposés libres) par, respectivement, les termes $t_1, \cdots, t_n$. Les symboles de variables $v_1, \cdots, v_k$, quel que soit leur statut, seront laissés tels quels.

Une formule dont toutes les occurrences de tous les symboles de variables sont liées est dite *close*. La formule $x \equiv ©\wedge \forall x\ x \equiv x$ n'est pas close, bien que x y soit liée car toute occurrence de x n'est pas dans le champ d'un quantificateur. En revanche, $\forall x\ [x \equiv ©\wedge x \equiv x]$ en est une.

4 Règles de déduction, systèmes de preuves formels

Sans rentrer dans le détail profond de ce que l'on appelle théorie de la démonstration, nous avons besoin de jeter les bases de l'édifice des preuves afin de comprendre les liens entre preuves syntaxiques et sémantiques. Bien que pouvant rester très généralistes, nous nous concentrerons ici sur le cas de la logique classique, comme annoncé, que l'on nomme système déductif LK.

4.1 Règles de déduction de LK

Dans toute la suite, les lettres capitales désignent des formules d'un langage quelconque. On note $A \vdash B$ pour dire que la prémisse A permet d'obtenir B comme conclusion. Nous donnons à présent les règles de construction de preuves :

1. $(Axiome)$ On peut toujours écrire $A \vdash A$.
2. *(Modus Ponens)* Si $F \vdash A$ et que $F \vdash A \Rightarrow B$ alors $F \vdash B$.
3. *(Modus Tolens)* Si $F \vdash \neg B$ et que $F \vdash A \Rightarrow B$ alors $F \vdash \neg A$.
4. *(Affaiblissement prémisses)* Si $F \vdash G$ alors $F \wedge A \vdash G$.
   On peut également conclure que $A \wedge F \vdash G$.
5. *(Intro $\wedge$)* Si $F \vdash A$ et que $F \vdash B$ alors $F \vdash A \wedge B$.
6. *(Affaiblissement conclusions)* Si $F \vdash G$ alors $F \vdash A \vee G$.
   On peut également conclure que $F \vdash G \vee A$.
7. *(Intro $\vee$)* Si $F \vdash A$ et $G \vdash A$ alors $F \vee G \vdash A$.
8. *(Intro $\Rightarrow$)* Si $F \wedge A \vdash B$ alors $F \vdash A \Rightarrow B$.
9. *(Intro $\neg$)* Si $F \vdash A \vee G$ alors $F \wedge \neg A \vdash G$.
10. *(Par l'absurde)* Si $F \wedge \neg A \vdash \bot$ alors $F \vdash A$
    On pourrait conclure de même avec $\neg A \wedge F \vdash \bot$.
11. *(Tiers Exclu)* On a toujours $\top \vdash A \vee \neg A$.
12. *(Coupure)* Si $F \vdash G$ et que $G \vdash H$ alors $F \vdash H$.
13. *($\forall$ Gauche)* Si $F[t/x] \vdash G$ alors $\forall x F \vdash G$.

14. *(Généralisation)* Si x n'est pas libre dans F et que $F \vdash A$ alors $F \vdash \forall x A$

15. *($\exists$ gauche)* Si x n'est libre ni dans F ni dans G et que $F \wedge A \Vdash G$ alors $F \wedge \exists x A \vdash G$.

16. *(Spécification)* Si $F \vdash A[t/x]$ alors $F \vdash \exists x A$.

Une *démonstration* de A à partir de F résulte de l'obtention de $F \vdash A$ à partir d'un nombre fini d'applications de ces règles.

Remarques : Nous avons énoncé bien plus de règles que réellement nécessaire et cela afin de simplifier la tâche à suivre et dans un souci de compréhension intuitive. Comme déjà dit plus haut, nous ne rentrerons pas dans le cadre de la théorie de la démonstration plus que nécessaire.

En temps normal, nous devrions noter $F \Vdash_S A$ où S est le système déductif considéré. Ici, comme nous serions sans cesse en train d'écrire $\Vdash_{LK}$, nous considérons notre cadre bien établi et nous affranchissons de cet indexation superflue.

On notera $\phi \dashv\Vdash \psi$ si $\phi \vdash \psi$ et que $\psi \vdash \phi$. Cela définit une relation d'équivalence non sans rappeler celle du calcul propsitionnelle.