BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA
FACULTE DES SCIENCES DE L'INGENIEUR
DEPARTEMENT D'INFORMATIQUE

جامعة باجي مختار – عنابـــــــــة
كلية علـــــــوم الهندســـــة
قسم الإعـــــلام الآلي

Annaba, le 2 Juin 2021

Procès Verbal de Réunion du Comité Scientifique du Département d'Informatique

En date du 02 Juin 2021, s'est tenue une réunion du Comité scientifique du département d'informatique.

Les membres du CSD ont examiné les rapports d'expertise **Favorables** du cours de Dr Mendjel Mohamed Said Mehdi intitulé 'Logique Mathématique '.

Décision de CSD : **Avis Favorable**

Annaba le 2/06/2021

Le CSD

الأستاذة الدكتورة: سريدي حسينة
رئيسة اللجنة العلمية
لقسم الإعلام الآلي

UNIVERSITE BADJI MOKHTAR-
ANNABA
FACULTE DE TECHNOLOGIE
DEPARTEMENT D'INFORMATIQUE

جامعة باجي مختار – عنابـــــــــة
كلية التكنلوجيا
قسم الإعلام الآلي

Attestation

Je soussigné, M BOULEMDEN Ahmed, chef de département d'informatique de la faculté de technologie de l'université Badji Mokhtar – Annaba, atteste par la présente que le support de cours du module « Logique Mathématique » de **M. MENDJEL Mohamed Said Mehdi** a fait l'objet d'une expertise approfondie menée par deux experts reconnus dans le domaine et qui ont été désignés par les membres du conseil scientifique du département d'informatique.

Les experts en question, Pr Djeddi Chawki (université de Tebessa) et Pr Khadir Mohamed Tarek (université Badji Mokhtar annaba), ont minutieusement examiné le contenu pédagogique du support de cours et ont émis un avis favorable quant à sa qualité, sa pertinence et son adéquation aux objectifs pédagogiques du module.

Les membres du CSD ont aussi examiné les rapports d'expertise le 02/06/2021 et ont donné l'avis favorable à sa validation.

Cette attestation est délivrée à la demande de l'intéressé pour servir et valoir ce que de droit.

Fait à Annaba, le 16/11/2023

Le chef de département

رئيس قسم الإعلام الآلي
كلية التكنولوجيا
الدكتور: احمد بولمدن

رئيس القسم

Université Badji Mokhtar Annaba
Faculté des sciences de l'ingéniorat
Département d'informatique

Support de cours

Logique mathématique

Préparé par : Dr. Mendjel Mohamed Said Mehdi
E-mail : mendjel@labged.net / mehdi.mendjel@yahoo.fr
Validé en : 2020 -2021
URL du cours : https://elearning-facsci.univ-annaba.dz/course/view.php?id=134

Table des matières

1 Chapitre 1 : Introduction à la décidabilité **3**
1.1 Introduction 3
1.2 Petit historique 4
1.3 Quelques problèmes décidables, semi-décidables et indécidables 5

2 Chapitre 2 : Une brève introduction à la calculabilité avec la machine de Turing et les fonctions primitives récursives **7**
2.1 Introduction 7
2.2 La définition formelle d'une machine de Turing (MT) 8
2.3 Les fonctions primitives récursives 10
2.3.1 Définition des fonctions primitives récursives de base . 11
2.3.2 Définition de la composition de fonctions 11
2.3.3 Définition de la récursion primitive 11
2.3.4 Définition des fonctions primitives récursives 12
2.4 Remarques pour les TD 13
2.5 Exercices 14

3 Chapitre 3 : Introduction aux systèmes formels **16**
3.1 Définition 16
3.2 Composants 17
3.3 Les propriétés d'un système formel 19
3.4 Exercices 20

4 Chapitre 4 : La logique propositionnelle **22**
4.1 Introduction 22
4.2 Langage 22
4.2.1 Alphabet 22
4.2.2 Les formules bien formées 23
4.3 Théorie de la preuve 24
4.3.1 Les axiomes 24
4.3.2 Règle d'inférence 25

4.3.3 Notion de démonstration 25
4.3.4 Théorème de déduction 26
4.4 Théorie des modèles 26
4.4.1 Notion d'interprétation et notion de modèle 27
4.4.2 Notion de validité 28
4.5 Équivalence de deux formules et formes normales 28
4.6 Équivalence entre les deux approches 31
4.7 Exercices 33

5 Chapitre 5 : La logique des prédicats du premier ordre 37
5.1 Introduction 37
5.2 Langage 37
5.2.1 L'alphabet 37
5.2.2 Définitions 38
5.2.3 La portée d'un quantificateur 39
5.2.4 Les formules congrues 40
5.2.5 La substitution et l'instantiation 40
5.3 La méthode déductive : Théorie de la preuve 41
5.3.1 Les axiomes 41
5.3.2 Les règles d'inférence 41
5.4 L'approche sémantique 42
5.4.1 Notion d'interprétation 42
5.4.2 Notion de validité et notion de modèle 44
5.5 L'équivalence entre les deux approches 45
5.6 Exercices 47
5.7 Note bibliographique 48

Chapitre 1

Chapitre 1 : Introduction à la décidabilité

1.1 Introduction

Il y a maintenant des dizaines de siècles que les mathématiciens décrivent et utilisent des méthodes de calcul permettant de résoudre leurs problèmes. Mais jusqu'à une date relativement récente (1934) ils ne savaient pas de façon précise ce qu'est une méthode de calcul.

La faculté étonnante des mathématiques à transformer en objets ce qui constitue leurs méthodes et leurs techniques, les rapproche de la philosophie. Cette faculté d'autoréflexivité permet par exemple de prendre les théories mathématiques pour objets d'études (et donc de théorèmes) comme les nombres entiers et les équations aux dérivées partielles sont objets d'études et de théorèmes.

Et effectivement, à propos du problème des méthodes de calcul, l'autoréflexivité des mathématiques a fonctionné, donnant naissance à une série de concepts et de résultats qui sont parmi les plus profonds et les plus féconds du vingtième siècle.

Exemples :
- Algorithmes,
- Fonctions calculables,
- Problèmes décidables et indécidables.

Aujourd'hui les mathématiciens possèdent une définition précise de ce qu'on doit appeler méthode de calcul, ils savent exactement quels sont les

problèmes qu'elles peuvent traiter, et mieux encore, ils savent que certains problèmes ne sont pas "traitables" En effet, et c'est là l'un des aspects les plus extraordinaires de ces travaux, ils permettent d'établir des résultats négatifs, c'est-à-dire de la forme : pour tel problème non seulement aucune méthode n'est actuellement connue, mais aucune ne le sera jamais.

Ces problèmes pour lesquels on démontre qu'il n'existe aucune méthode de calcul adaptée s'appellent des problèmes indécidables.

Les premiers résultats d'indécidabilité des années 1930 semblaient artificiels, très vite on s'est aperçu que des problèmes assez simples entraient dans la classe des problèmes indécidables. En particulier en informatique, de nombreuses questions naturelles qui se posent aux programmeurs se révèlent après étude correspondre à des problèmes indécidables. Lorsqu'on démontre qu'un problème P est indécidable, on en déduit : - qu'on doit renoncer à le résoudre tel quel, et donc que, - qu'il faut trouver une version simplifiée de P, puis, soit réussir à la résoudre, soit à nouveau établir qu'elle est encore indécidable et donc la simplifier encore, etc (les fonctions récursives).

1.2 Petit historique

La notion informelle d'algorithme est extrêmement ancienne et les procédés que nous avons appris à l'école primaire pour additionner deux nombres écrits en base dix ou pour les multiplier sont des algorithmes.

En fait, au début du siècle les mathématiciens ne soupçonnaient pas qu'on pourrait arriver à préciser vraiment cette notion, ni encore moins, qu'on pourrait démontrer pour certains problèmes qu'aucun algorithme n'existe.

Le travail d'identification et de formulation de la notion d'algorithme fut effectué en plusieurs étapes entre 1931 et 1936 par les mathématiciens Church, Kleene, Turing et Gödel.

Ils introduisirent plusieurs classes différentes de fonctions dont ils montrèrent ensuite qu'elles coïncidaient, et qu'ils reconnurent alors comme la classe des fonctions calculables. Une fonction est calculable s'il existe une façon finie de la décrire qui permette effectivement d'en calculer toutes les valeurs. La définition précise de la notion de fonction calculable fixe en même temps celle d'algorithme, et on peut donc dire qu'en 1936 la formulation exacte de la notion d'algorithme était acquise. De toutes les définitions fi-

nalement équivalentes de la notion d'algorithme formulées dans les années 1930 et depuis, celle donnée par Turing en 1936 est la plus pratique pour les théoriciens, et on l'utilise encore aujourd'hui. C'est sur elle que nous allons nous appuyer plus loin pour définir la notion d'algorithme.

Chaque année des dizaines de résultats de décidabilité ou d'indécidabilité sont établis dans de nombreux domaines des mathématiques et on peut être certain, tant les questions ouvertes restent nombreuses, que ce mouvement n'est pas prêt de cesser. Ici nous nous intéresserons uniquement au problème de la possibilité d'un algorithme ou de son impossibilité, et non pas au problème de l'efficacité qui lui aussi est à l'origine d'un domaine de travail et de réflexion des plus vivants aujourd'hui.

1.3 Quelques problèmes décidables, semi-décidables et indécidables

Problème A.
Soient m et n deux entiers donnés > 1. m est-il un multiple de n ? On sait qu'il est vrai que : 12 est un multiple de 2, et qu'il est faux que : 16 est un multiple de 5. On sait même bien plus que cela, on sait comment s'y prendre pour déterminer pour tout n et tout m quand est vrai que "m est un multiple de n" et quand est faux que : "m est un multiple de n".

Il suffit en effet de :
- Faire la division de m par n,
- Regarder le reste obtenu, r,
- Si $r = 0$ alors il est VRAI que "m est un multiple de n",
- Sinon il est FAUX que "n est un multiple de m".

Ce procédé général et systématique de calcul, constitue un algorithme (informel). C'est un procédé absolument sûr, qui fonctionne en un temps fini pour tous les jeux de données possibles et conduit toujours à la bonne réponse : il montre que le problème A est décidable.

Remarquons bien que pour être précis quand on parle de problème décidable ou indécidable il faut indiquer ce que sont les paramètres du problème. Dans notre exemple n et m sont deux entiers > 1. En fait on ne cherche pas à résoudre un problème unique, on cherche à résoudre une classe infinie de problèmes, ici tous les problèmes :

2 est-il un multiple de 2 ?
3 est-il un multiple de 2 ?
.....
10 est-il un multiple de 3 ?
.....

Problème B.
Problème des nombres premiers. Soit n un entier donné > 1
n est-il un nombre premier ?

Problème C.
Il existe aussi des problèmes qui sont semi décidables.
Exemple : Soit le programme P suivant :
"3x + 1" Est-ce que le programme ci-dessous s'arrête sur un x donné ?

while x > 1 do
{ if (x mod 2 = 0) then x := x/2 ;
else x := 3x + 1 ; }

Réponse. Pour ce problème, on peut utiliser la procédure suivante : 'a partir de x donné exécuter ce programme, et s'il s'arrête retourner "OUI". Par contre, si pour un certain x le programme "3x + 1" boucle, cette procédure ne pourra jamais donner la réponse "NON". Une telle procédure s'appelle un semi-algorithme on peut dire aussi que c'est un programme semi décidable.

Il est inconnu s'il existe un algorithme de décision pour ce problème, qui pour chaque x donné renvoie une réponse correcte "OUI" ou "NON".

Problème D.
Il y a aussi des problèmes indécidables.

Exemple :
Soit l'algorithme A/P(x) { tant que AP(x) (si x est pair) alors x := x*2 }.
Il est évident que si on lui passe comme paramètre un x pair, cet algorithme ne se terminera jamais donc il est indécidable.

Chapitre 2

Chapitre 2 : Une brève introduction à la calculabilité avec la machine de Turing et les fonctions primitives récursives

2.1 Introduction

Un programme peut être considéré comme la décomposition de la tâche à réaliser en une séquence d'instructions élémentaires (manipulant des données élémentaires) compréhensibles par l'automate programmable que l'on désire utiliser. Cependant, chaque automate programmable (en pratique chaque processeur) se caractérise par un jeu d'instructions élémentaires qui lui est propre. Dans la pratique, cette diversité est résolue par l'existence de compilateurs ou d'interpréteurs qui traduisent les instructions du langage (évolué) mis à disposition des programmeurs en plusieurs instructions élémentaires (langage machine) du processeur utilisé. Cependant, cette solution n'est pas suffisante pour les besoins de l'informatique théorique, qui requière une représentation unifiée de la notion d'automate programmable, permettant de démontrer des résultats généraux (décidabilité, complexité, ...) vrais pour l'ensemble des automates programmables concrets que l'on peut envisager. A cette fin a été développée la notion de machine de Turing, qui constitue une abstraction (et une formalisation) de la notion d'automate programmable.

la machine de Turing est un modèle de machine théorique fondamental pour la théorie de la calculabilité et de la complexité. Malgré sa simplicité extrême, on peut démontrer qu'elle est capable de simuler toute opération

réalisable par n'importe quel processeur, aussi puissant soit-il. Elle résume de manière saisissante le concept d'ordinateur et constitue un support idéal pour raisonner autour de la notion d'algorithmique.

Il existe de nombreuses formulation de cette machine, et si celle qui va suivre ne correspond pas exactement à celle que vous connaissez, c'est sans grande importance, car il est très facile de montrer qu'elles sont équivalentes.

2.2 La définition formelle d'une machine de Turing (MT)

Une machine de Turing est présentée comme un quintuplet $\langle Q, A, q_0, b, d \rangle$ où :

- Q est un ensemble fini d'états,
- A est un ensemble fini d'alphabet,
- $q_0 \in Q$ représente l'état initial,
- b est un symbole n'appartenant pas à A appelé symbole blanc,
- d est une fonction de transition de Q x B $\longrightarrow$ (B $\cup\{<,>\}$) x Q, avec B = A $\cup\{b\}$.

Nous allons donner une description plus physique du fonctionnement d'une machine de Turing. Pour reprendre l'idée même de Alan Turing (le créateur de la machine qui porte aujourd'hui son nom). Une machine de Turing n'est rien d'autre qu'une machine à écrire modifiée. C'est-à-dire au lieu de travailler sur une feuille de papier, la machine de Turing opère sur une bande infinie à gauche et à droite (indexée par l'ensemble **Z** des entiers relatifs) à l'aide d'une tête de Lecture/ Écriture. La bande est constituée de plusieurs cellules, chaque cellule contenant un symbole de l'alphabet A ou un symbole blanc. On dit qu'une cellule est vierge si le symbole qu'elle contient est le symbole blanc. La tête de Lecture/Écriture est capable :

- D'écrire un symbole dans la cellule courante (celle sur laquelle la tête se pointe),
- D'effacer le contenu de la cellule courante (équivaut à écrire le symbole blanc),
- De se déplacer d'une cellule vers la gauche ou vers la droite ($< et >$ dans la définition),
- De lire le symbole contenu dans la cellule courante.

Exemple. La machine vue dans la figure suivante est une machine qui lit un « b » dans l'état p, passe à l'état q, écrit un « a » à la place du « b » et déplace sa tête de lecture à gauche.

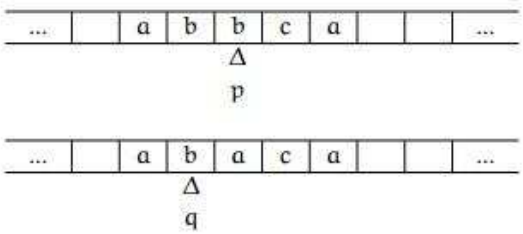


FIGURE 2.1 – Exemple d'une machine de Turing

Du point de vue du modèle, les trois premières opération sont représentées par l'ensemble B $\cup\{<,>\}$, c'est-à-dire on écrit un symbole de B (si ce symbole est b, on considère qu'on efface la cellule), ou on déplace la tête de Lecture/ Écriture d'une cellule vers la gauche ou la droite. Notons que dans la majorité des ouvrages, ces opérations sont représentées par l'ensemble B x $\{<,>\}$,i.e. on écrit un symbole de B et on déplace la tête de Lecture/ Écriture vers la gauche ou la droite.

La fonction de transition d et le contenu initial de la bande qui vont déterminer la séquence des opérations exécutées par la machine ; Au début de l'exécution, la machine est dans l'état initial q_0 et la tête de lecture/ Écriture est par convention placée sur la cellule indexée par 0 ; L'égalité d(q, s) = (o, q′) se lit : si la machine de Turing est dans l'état q et que la cellule contient le symbole s (on dit que la tête de Lecture/ Écriture lit le symbole s), alors la machine de Turing réalise l'opération o et passe à l'état q′.

On appelle instruction d'une machine de Turing , tout quadruplet (q, s, o, q′) tel que d(q, s) = (o, q′). Se donner un ensemble d'instructions, donc de quadruplets de Q x B x (B $\cup\{<,>\}$) x Q revient à définir le graphe d'une fonction de transition, les deux approches sont donc équivalentes. L'ensemble des instructions est appelé *programme* de la machine de Turing. Quand la machine est dans un état q, qu'elle lit un symbole s et que la fonction d n'est

pas définie pour le couple (q, s), on dit que la machine s'arrête et que l'état q est un état d'arrêt.

Exemple applicatif. Écrire la machine de Turing qui reconnait les mots de longueurs paires sachant que l'alphabet = $\{a, \#\}$, si la longueur du mot est paire la machine écrit T sur la bande sinon elle écrit F.

Solution

q_0 a D q_1
q_1 a D q_2
q_2 # T q_3
q_3 T arrêt
q_2 a D q_4
q_4 a D q_2
q_4 # F q_4
q_1 # F q_4
q_4 F arrêt

2.3 Les fonctions primitives récursives

Les fonctions primitives récursives ont été introduites par Godel en 1934 dans son travail sur l'incomplétude. Elles permettent de décrire des fonctions dont il est clair que le calcul termine toujours. Elles correspondent ainsi aux fonctions qui peuvent être calculées sans l'instruction "While" dans un langage typé comme Pascal, c'est à dire avec des " if-then-else" et des "for". L'objectif de cette petite classe est de se convaincre de cette expressivité et d'observer une partie de ses limitations.

Informellement, ces fonctions calculables sont celles que l'on peut définir sur l'ensemble des entiers naturels $\mathbb{N}$ par récurrence :
$\bigcup_{k\in\mathbb{N}} F_k$ avec $F_k = \mathbb{N}^k \longrightarrow \mathbb{N}$

Elles sont comme suit :
- Les fonctions de base,
- La composition de fonctions,
- le mécanisme de récursion.

2.3.1 Définition des fonctions primitives récursives de base

- La fonction zéro() ou $\boldsymbol{O}() \in F_0$ tel que $\boldsymbol{O}() = 0$,
- La fonction successeur succ(x) ou $\sigma(n) \in F_1$ tel que $\sigma(n) =$ n+1,
- La fonction de projection $proj_i^k$ ou $\pi_i^k \in F_k$, $k \geqslant 1, 1 \leqslant i \leqslant k$ tel que $\pi_i^k(n_1, ..., n_k) = n_i$.
  Exemple : $\pi_2^3(5, 6, 8) \longrightarrow 6$.

2.3.2 Définition de la composition de fonctions

Soient $g \in F_l$ et $h_1, ..., h_l \in F_k$. La composition de g et des fonctions $h_1, ..., h_l$ est définie par $f \in F_k$ tel que :
f(n) = $g(h_1(n), ..., h_l(n))$ avec n = $(n_1, ..., n_k)$
Exemple : dans cet exemple, on représente la composition par $\circ$, on peut aussi la représenter par Comp.
succ $\circ$ Zéro $\longrightarrow$ 1,
succ $\circ$ succ $\circ$ Zéro $\longrightarrow$ 2,
succ $\circ$ π_3^3(0, 2, 5)$\longrightarrow$ 6.

2.3.3 Définition de la récursion primitive

Soient $g \in F_k$ et $h \in F_{k+2}$, la fonction $f \in F_{k+1}$ tel que :
- f(n, 0) = g(n),
- f(n, m+1)= h(n,m,f(n,m))

Est la fonction définie à partir de g et h par récursion primitive.
Remarque : La fonction f est calculable si g et h sont calculables.

On peut aussi représenter la récursion par le schéma récursif suivant :

Fonction f(x, k)
Si : k=0
F(x)
Sinon :
G(x, k-1, f(x, k-1))
Fin
Fin

On a remplacé les fonctions g par F et h par G pour dire qu'il y a plusieurs représentations formelles. Si F et G sont primitives récursives alors f

est primitives récursives, f fait la récursivité sur l'argument k, tant dis que l'argument x est une donnée supplémentaire.

2.3.4 Définition des fonctions primitives récursives

D'une manière générale, les fonctions primitives récursives sont toutes les fonctions que l'on peut construire à partir des fonctions de base par composition et récursion primitive.

Exemple : Montrez que la fonction +(x, k) est primitive récursive.

En se basant sur le schéma récursif vu précédemment, on peut avoir :

Fonction +(x, k)
Si : k=0
F(x)
Sinon :
G(x, k-1, f(x, k-1))
Fin
Fin

Donc pour F on prend la fonction identité $\pi_1^1(x)$ et pour la fonction G on prend succ $\circ\pi_3^3$ c'est à dire :

Fonction +(x, k)
Si : k=0
$\pi_1^1(x)$
Sinon :
succ $\circ\pi_3^3$(x, k-1, f(x, k-1))
Fin
Fin

Exemple applicatif : +(5,3)
+(5,3) = succ $\circ\pi_3^3$(5,2,+(5,2)) =
succ $\circ\pi_3^3$(5,2,succ $\circ\pi_3^3$(5,1,+(5,1)) =
succ $\circ\pi_3^3$(5,2,succ $\circ\pi_3^3$(5,1,succ $\circ\pi_3^3$(5,0,+(5,0))=
succ $\circ\pi_3^3$(5,2,succ $\circ\pi_3^3$(5,1,succ $\circ\pi_3^3$(5,0,π_1^1(5))))
Comme : π_1^1(5)=5 alors :
succ $\circ\pi_3^3$(5,0,+(5,0)) = 6 et
succ $\circ\pi_3^3$(5,1,+(5,1)) = 7 et enfin

succ $\circ\pi_3^3$(5,2,+(5,2)) = 8

On remarque que nous avons réussi à construire la fonction + par la règle de récursion à partir des deux fonctions primitives récursives :

- La fonction F = π_1^1 qui est une fonction de base (la projection)
- La fonction G = $succ \circ \pi_3^3$ qui est une composition de deux fonctions de base la projection et la fonction successeur.

Remarque 1 : Dans la récursion, l'ordre des arguments de la fonction G(x, k-1, f(x, k-1)) n'est pas important, on peut trouver dans d'autres documents G(x,f(x, k-1), k-1).

Remarque 2 : On peut aussi résoudre le problème en utilisant la règle de récursion comme suit :
+(x,0) = x + 0 = x = π_1^1
+(x, y+1) = +(x,y)+1 = succ(+(x,y)) = $succ(\pi_2^3(x, +(x, y), y)) = succ \circ \pi_2^3(x, +(x, y), y)$.

Remarque 3 : L'objectif de la partie "Fonction primitives récursives" est d'abord de voir un autre modèle de calcul que la machine de Turing avec ses différentes formes et de montrer que les fonctions primitives récursives et la machine de Turing ont le même pouvoir expressif avec bien sur le λ-calcul (qui n'est pas dans le programme).

2.4 Remarques pour les TD

Afin d'avoir une représentation de la MT beaucoup plus algorithmique, nous écrirons les instructions de la manière suivante :

« état courant » « symbole » « opération » « état suivant ».

« opération » peut être un déplacement vers la droite « D » ou vers la gauche « G » ou bien le remplacement du symbole lu par un autre. Exemple : q_0 s D q_1 qui représente un déplacement vers la droite après la lecture du symbole s, q_0 s s$'$ q_1 dans ce cas il n' y a pas un déplacement par contre il y a un remplacement du symbole s par le symbole s$'$.

Au départ, la tête de L E est toujours placée sur un symbole différent du symbole blanc et on ne boucle pas sur l'état initial.

2.5 Exercices

Exercice 1 soient les instructions suivantes d'une MT quelconque :

q_0 s_0 D q_1
q_0 s_1 s_2 q_2
q_1 s_0 D q_1
q_1 s_2 D q_1
q_1 s_1 D q_2
q_1#
q_2 s_2 G q_3
q_3 (s_0/s_1/s_2) G q_3
q_3#

Dérouler cette MT sur la séquence suivante : #$s_0s_0s_2s_1s_2s_1s_2s_0s_2$# sachant que le # est le symbole blanc.

Exercice 2 Écrire la MT qui étant donné un mot sur le ruban composé des symboles a et b, détermine si le mot se termine par un b ou non. Elle écrit à la fin du mot un T si vrai et un F sinon. Le blanc = $.

Exercice 3 Modifier la MT précédente pour qu'elle vérifie si le mot en entrée se termine par le même symbole de départ (qu'il soit a ou b).

Exercice 4 Les valeurs en base 10 correspondants aux codes ASCII des lettres sont :

- A : 65 ; B : 66 ; C : 67 ; etc,
- a : 97 ; b : 98 ; c : 99 ; etc.

Pour passer du code ASCII d'une lettre minuscule à celui de la majuscule correspondante, il suffit de transformer le 3 ème bit en partant de la gauche de 1 à 0. Par ailleurs, les deux premiers bits sont toujours égaux à « 01 » et les 5 derniers bits ne sont pas modifiés.

A : 65 = 01000001 et a : 97 = 01100001
C : 67 = 01000011 et c : 99 = 01100011
Écrire la MT qui transforme une lettre minuscule en lettre majuscule.

Exercice 5 Écrire la MT qui reconnait la séquence 0001 dans un mot sachant que le ruban contient plusieurs mots et l'alphabet $\Sigma = \{0, 1, \#\}$. Il y a plusieurs mots sur le ruban séparés par un seul #. Deux # successifs désignent la fin de la séquence.

Exercice 6 Écrire la MT qui vérifie si un mot donné sur le ruban contient la séquence de caractères suivants : « aab ». Le blanc = # et il y a plusieurs mots sur le ruban séparés par un seul #. Deux # successifs désignent la fin. A = $\{a, b, \#\}$. On écrit un T ou un N.

Exercice 7 Écrire la MT qui remplace le "0" qui vient après deux "1" par un "1". A = $\{0, 1, \#\}$, q_0 est l'état initial et le ruban contient une seule séquence.

Exercice 8 Écrire la MT qui transforme le mot sur le ruban écrit sur $\{a, b, \#\}$ de telle sorte que tous les "a" soient au début. Exemple : aabbaba devient aaaabbb.

Exercice 9 Montrez que les fonctions suivantes sont primitives récursives :

1. La fonction plus = x + y,
2. La fonction Sigma = $\Sigma_{i=0}^{x} i$,
3. La fonction prédécesseur (pred(x)),
4. La fonction différence (diff(x,y)) telle que diff(x,y) = $\{^{x-y\ si\ x>y}_{0\ si\ x\leq y}$,
5. La fonction différence absolue $|x - y| = \{^{x-y\ si\ x\geq y}_{y-x\ si\ x<y}$,
6. La fonction alpha tel que $\alpha(x) = \{^{1\ si\ x=0}_{0\ si\ x\neq 0}$,
7. La fonction multiplication mult = x * y,
8. La fonction factorielle Fact(x) = $x!$.

Chapitre 3

Chapitre 3 : Introduction aux systèmes formels

L'expression " système formel " se compose de deux mots :

- **Système :** qui est défini comme étant un ensemble d'éléments matériels ou non dépendant les uns des autres de manière à former un tout cohérent et organisé,
- **Formel :** c'est-à-dire qui porte principalement sur la forme et ne tient pas compte du contenu.

3.1 Définition

Les systèmes Axiomatico-Déductifs ou systèmes formels (SF) fournissent un cadre général pour exprimer et étudier de façon rigoureuse et mathématique les notions d'axiomatique et de mécanismes déductifs.

Un système formel est donc un procédé mécanique de construction des phrases d'un langage, une entité idéale qui engendre sous forme de *théorèmes* toutes les conséquences qui découlent selon des critères déterminés (règles) d'un ensemble de propositions initiales considérées comme vérités premières (axiomes). Les expressions qui figurent dans un système formel n'ont aucun sens et résultent des possibilités opératoires précisées dans les règles de maniement du système. Lorsqu'on construit un système formel c'est en général avec l'intention de représenter dans ce système une théorie non formalisée. le but de la formalisation est alors de permettre une étude précise et systématique des aspects structuraux des théories scientifiques.

3.2 Composants

Un système formel S est composé d'un quadruplet :
- Un ensemble fini et dénombrable de symboles appelé alphabet,
- Un sous ensemble récursif W de l'ensemble de suites finies de S appelé " Ensemble de formules bien formées " construites à partir de l'alphabet,
- Un sous ensemble A de W appelé ensemble d'axiomes,
- Un ensemble fini R de règles (dites encore règles d'inférence, de déduction ou de dérivation) telle que R = $\{R_1, R_2, ..., R_n\}$

Notion de démonstration : Établir une démonstration dans S revient à trouver une suite finie de fbf, f_1, f_2,...., f_k telle que f_i (sachant que $1 \leqslant i \leqslant k$) est soit un axiome de S, soit une fbf obtenue d'une autre fbf par application d'une règle d'inférence R.

Notion de théorème : Une formule d'un système formel est dite un *théorème* si c'est un axiome, ou bien si la formule est obtenue par application d'une règle d'inférence à un théorème.

Une preuve d'un système formel est une suite de formules qui sont soit des axiomes, soit des formules déduites des formules précédentes.

Soient f_1, f_2,...., f_k et g des fbf, on note f_1, f_2,...., $f_k \vdash g$ et on lira qu'à partir des formules f_1, f_2,...., f_k on peut déduire la formule g.

Exemple : Soit le système PEU avec :
- L'alphabet S = $\{p, e, u\}$,
- W est l'ensemble des fbf formée à partir de l'alphabet (suite de p,e,u),
- L'ensemble des axiomes A = $\{upueuu\}$,
- Les règles d'inférence sont définies comme suit :
  - R_1 : si une expression de la forme AeB est un théorème $\Longrightarrow$ uAeBu est un théorème
  - R_2 : si une expression de la forme AeB est un théorème $\Longrightarrow$ AueuB est un théorème.

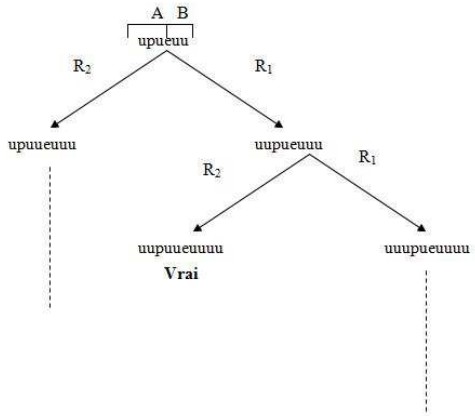
Questions

Q1 : Est ce que uupuueuuuu est un théorème ?

Q2 : Est ce que upupueuuu est un théorème ?

Solutions

Q1 : On peut démontrer que uupuueuuuu est un théorème en construisant l'arbre de dérivation.



Q2 : upupueuuu n'est pas un théorème, car un théorème ne peut pas contenir plus d'un symbole p et on peut démontrer ça par récurrence.

Le raisonnement par récurrence : Les raisonnements en mathématique se font généralement par dérivation ou par déduction comme par exemple dans la Q1. Mais il existe aussi un autre type de raisonnement, que l'on appelle raisonnement par récurrence, particulièrement adapté lorsqu'il est demandé de prouver des propriétés portant sur n paramètres. Par exemple, essayant de démontrer l'égalité suivante :

$1+2+3+4+....+n = \frac{(n)(n+1)}{2}$ quelque soit le nombre n, bien sur Gauss était astucieux car il a pu démontrer cette formule en se basant sur la somme des n premiers nombres entiers positifs.

$S_n = 1 + 2 + 3 + 4 + ...+(n-1) + n$

$S_n = n + (n-1) + (n-2)+ (n-3) +....+ 2 + 1$

Si on fait la somme colonne par colonne, on trouve :

$2S_n$ = (n+1)+(n+1)+(n+1)+(n+1)+(n+1) = n(n+1) d'où la formule $S_n = \frac{(n)(n+1)}{2}$

On peut aussi se rabattre sur le raisonnement par récurrence pour prouver cette formule et il y a deux démonstrations à faire :

- Montrer que la formule est vraie pour n = 1, en effet 1 = (1.2)/2
- On suppose que la formule est vraie à un certain rang n puis on montre qu'elle reste vraie au rang suivant.

Si on prend par exemple S_{n+1} = 1+2+3+4+....+n+(n+1)= S_n + (n+1) $= \frac{(n)(n+1)}{2}$ +(n+1), la formule étant supposée vraie au rang n. Il reste à factoriser et à réduire au même dénominateur cela donne à la fin $\frac{(n+1)(n+2)}{2}$ qui est la formule au rang n+1.

Ces deux démonstrations étant faites, nous pouvons affirmer que la propriété est vraie pour tout rang n.
Revenant maintenant à la question 2 (Q2) pour démontrer que upupueuuu n'est pas un théorème, on prend l'axiome du système qui est le upueuu on remarque qu'il existe uniquement un seul p alors que le mot proposé "upupueuuu" contient deux p. Passant maintenant au rang 1 selon l'arbre de dérivation on constate qu'il y a toujours un seul p dans les deux mots générés et ainsi de suite sachant que le passage d'un rang à un autre se fait à base des deux règles R_1 et R_2.

3.3 Les propriétés d'un système formel

La propriété de consistance Un système formel est consistant (c'est-à-dire non contradictoire) si on ne peut pas prouver à la fois que A est un théorème et son contraire aussi ($\vdash$ A et $\vdash \neg$ A) ou bien $\vdash$ A et $\neg \vdash$ A. Un système non consistant est dit inconsistant.

La propriété de complétude Un système formel est dit complet s'il fournit une stratégie permettant d'atteindre la solution si celle-ci existe. C'est-à-dire , il a la capacité de pouvoir prouver tous les théorèmes valides.

La propriété de décidabilité On dit qu'un système formel est décidable s'il existe une procédure mécanique permettant d'établir en un temps fini si un mot du langage est ou n'est pas un théorème.

La saturation d'un système formel Un SF est saturé si et seulement si en ajoutant une formule f qui n'est pas un théorème, le SF devient inconsistant.

3.4 Exercices

Exercice 1
1. Définissez un système formel de telle sorte qu'on puisse produire les théorèmes kst, kstst, kststst,..... à partir d'un axiome k.

2. Définissez un système formel de telle sorte que l'on puisse produire les théorèmes ca, caba, cababa, cababababa, cababababa,..., etc.L'axiome est c.

3. Définissez un systeme formel de telle sorte que l'on puisse produire les théorèmes b, ba, baa, baaa, baaaa,...,etc. L'axiome est b.

Exercice 2 Soit le système MIU qui comprend :
- L'alphabet S = $\{M, I, U\}$
- L'axiome A = $\{MI\}$
- les règles :
  - R_1 : si une chaine se termine par un I on peut ajouter un U à la fin,
  - R_2 : si on a une chaine Mx on peut former Mxx (où x est une chaine quelconque),
  - R_3 : on peut remplacer III par un U dans une chaine,
  - R_4 : on peut supprimer toute paire UU.

1. Prouver que MUIUI est un théorème.
2. UM est-il un théorème ?
3. MU est-il un théorème ?

Exercice 3 Soit le système formel p-q
S= $\{p, /, q\}$ A= $\{pq\}$ R =
- a- $x \rightarrow /x/$
- b- $xpy \rightarrow xp/y/$ (x et y sont des mots du système)

Peut-on dériver les chaines suivante : $//p/q///$;$/p//q/$;$//////p///q/////////$?

Exercice 4 Soit un système formel composé :
d'un alphabet $\{A, B, C, D\}$,
Des axiomes D, DD,
des règles de production :

- a- ajouter C à la fin d'une chaîne quelconque.
- b- ajouter un A au début et à la fin d'une chaîne quelconque.
- c- remplacer un C par un B dans une chaîne.

Parmi les chaînes suivantes, lesquelles sont des théorèmes ? Donner les preuves
DC, DCCC, DCCA, AAADAAA, AAADAAAA, AADCCCABBA.

Exercice 5 Soit le système formel S (Σ, A, W, R) tel que :

- Σ : c'est l'ensemble de l'alphabet tel que $\Sigma = \{a, b, c\}$,
- A : c'est l'ensemble des axiomes qui ont la forme suivantes A $= \{a^{2i+1}bc^{2i-1} | i \geq 1\}$,
- W : représente l'ensemble des fbfs générées à partir des axiomes et des fbfs déjà générées,
- R : c'est l'ensemble des règles tel que R $= \{r_1 : (a^k bc^m, a^p bc^n) \longrightarrow a^{k+n}bc^{m+p}\}$

Q1 : Est ce que les formules suivantes sont des théorèmes $a^4bc^4, a^5bc^5, a^6bc^6$?

Q2 : Donner les différentes formes possibles de théorèmes.

Chapitre 4

Chapitre 4 : La logique propositionnelle

4.1 Introduction

Dans le cadre de la logique classique, une proposition est un énoncé déclaratif (une assertion) auquel nous pouvons assigner la valeur "vrai" ou "faux", mais pas les deux : on appelle ça la logique propositionnelle.

Dans le cadre de la logique propositionnelle, deux approches de résolution sont possibles : la première est une approche déductive, parfois appelée théorie de la preuve, qui permet de décider si oui ou non une formule est démontrable, la seconde est une approche sémantique, parfois appelée théorie des modèles, elle fait appel à la notion d'interprétation d'une formule.

Dans ce chapitre, nous allons présenter ces deux approches et nous clôturons le chapitre par la notion de complétude qui établit l'équivalence entre les deux approches.

4.2 Langage

4.2.1 Alphabet

Définition 1 : L'alphabet de la logique propositionnelle est constitué de :

- Un ensemble dénombrable de variables propositionnelles (ou formules atomiques , ou encore atomes).

On entend par proposition atomique (formule atomique), un énoncé indécomposable, par exemple : il pleut, la route est mouillée, ce chat est blanc, ..., plus abstraitement : P, Q,..... . Ainsi, nous utilisons P, Q, R, P_1, P_2,....., pour les variables propositionnelles.

– les connecteurs $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ et les parenthèses.

La négation $\neg A$: "non A" (c'est à dire A n'est pas le cas), le connecteur de la négation est un connecteur unaire. Si on prend la proposition A = "ce chat est blanc" , la négation de A n'est pas "ce chat est noir", mais tout simplement "ce chat n'est pas blanc".

La conjonction A $\wedge$ B : "A et B", ce connecteur est binaire.

La disjonction A $\vee$ B : "A ou B", ce connecteur est binaire.

L'implication A $\rightarrow$ B : "Si A alors B", ce connecteur est binaire

L'équivalence A $\leftrightarrow$ B : "A ssi B", ce connecteur est binaire.

– Les séparateurs ou bien les parenthèses (et).

4.2.2 Les formules bien formées

Définition 2 : l'ensemble des formules ou formules bien formées (fbf) de la logique propositionnelle est le plus petit ensemble de mots construits sur l'alphabet tel que :

- Si A est une formule atomique alors A est une formule bien formée,
- $\neg A$ est une formule bien formée si A est une formule bien formée,
- (A $\wedge$ B) est une formule bien formée si A et B sont des formules bien formées,
- (A $\vee$ B) est une formule bien formée si A et B sont des formules bien formées,
- (A $\rightarrow$ B) est une formule bien formée si A et B sont des formules bien formées,
- (A $\leftrightarrow$ B) est une formule bien formée si A et B sont des formules bien formées.

Pour éviter les ambiguïtés et minimiser l'emploi des parenthèses, on introduit une priorité entre les connecteurs. Après les parenthèses le connecteur de la négation a la plus forte priorité, la conjonction, la disjonction, l'implication et enfin l'équivalence.

Exemple : En mettant les parenthèses, déterminer l'ordre de priorité des connecteurs de la formule suivante :
$P \wedge \neg Q \vee R \rightarrow S \leftrightarrow X \vee Y$

Solution : $P \wedge \neg Q \vee R \rightarrow S \leftrightarrow X \vee Y = (((P \wedge \neg(Q)) \vee R) \rightarrow S) \leftrightarrow (X \vee Y)$

4.3 Théorie de la preuve

Nous allons maintenant nous intéresser à la méthode déductive qui permet de décider si une fbf est un théorème ou non. Notons, qu'un axiome est une fbf posée comme étant un théorème sans démonstration, et qu'un théorème est une fbf démontrable à partir des axiomes en utilisant les règles d'inférence ; tout le problème de la déduction consiste à engendrer des théorèmes à partir d'autres théorèmes (dont les axiomes) ; ceci se fait par des règles d'inférence.

4.3.1 Les axiomes

Ils sont obtenus à partir des schémas d'axiomes suivants, en remplaçant A,B et C par n'importe quelle fbf.

Voici les schémas d'axiomes de la logique propositionnelle :
- **1a.** $(A \rightarrow (B \rightarrow A))$
- **1b.** $(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C))$
- **1c.** $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
- **1d.** $(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$
- **2.** $A \rightarrow (B \rightarrow A \wedge B)$
- **3a.** $A \wedge B \rightarrow A$
- **3b.** $A \wedge B \rightarrow B$
- **4a.** $A \rightarrow A \vee B$
- **4b.** $B \rightarrow A \vee B$
- **5.** $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (A \vee B \rightarrow C))$
- **6.** $B \rightarrow ((B \rightarrow C) \rightarrow C)$
- **7.** $A \rightarrow A$
- **8.** $(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$
- **9.** $(A \rightarrow B) \rightarrow ((B \rightarrow A) \rightarrow (A \leftrightarrow B))$

4.3.2 Règle d'inférence

Dans la logique propositionnelle, il y a une seule règle d'inférence appelée **modus-ponens**.

Si P,
et P $\rightarrow$ Q,
On conclut Q

Si on note $\vdash Q$ cela indique que la formule Q est un théorème.

4.3.3 Notion de démonstration

Une fbf Q est un théorème, si et seulement si, il existe une démonstration dont la dernière formule est Q.

La notion de démonstration (ou dérivation, ou preuve) peut être étendue à une " démonstration à partir des hypothèses ". Ainsi prouver qu'une proposition P est un théorème, revient à chercher une démonstration dont la dernière fbf est P.

Exemple 1 : Soit à démontrer que $A \rightarrow A$ est un théorème.
Solution :
1. $\vdash A \rightarrow (A \rightarrow A)$ sh 1a
2. $\vdash (A \rightarrow (A \rightarrow A)) \rightarrow ((A \rightarrow ((A \rightarrow A) \rightarrow A)) \rightarrow (A \rightarrow A))$ sh 1b
remplacer le B par $A \rightarrow A$ et le C par A
3. $\vdash ((A \rightarrow ((A \rightarrow A) \rightarrow A)) \rightarrow (A \rightarrow A)$ m.p 1,2
4. $\vdash A \rightarrow ((A \rightarrow A) \rightarrow A)$ sh 1a remplacer le B par $A \rightarrow A$
5. $\vdash A \rightarrow A$ m.p 3,4

Exemple 2 : Établir la déduction suivante :

$A \rightarrow (B \rightarrow C), B \vdash A \rightarrow C$
Solution :

1. $\vdash$ B 2 ème Hyp
2. $\vdash B \rightarrow (A \rightarrow B)$ sh.1a
3. $\vdash A \rightarrow B$ m.p 1,2
4. $\vdash (A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C))$ sh.1b

5. $\vdash (A \to (B \to C)) \to (A \to C)$ m.p 3,4
6. $\vdash A \to (B \to C)$ 1 ère Hyp
7. $\vdash A \to C$ m.p 5,6

4.3.4 Théorème de déduction

Définition 3 : Si P est une fbf, et E un ensemble de fbf, on dit que P est une conséquence de E ou P est déductible à partir de E tel que les éléments de E sont appelés les hypothèses ou prémisses de la démonstration. Nous noterons "P est une conséquence de E" par $E \vdash P$.
Si E = $\{E_1, E_2, ..., E_n\}$, alors, la notation est : $E_1, E_2, ..., E_n \vdash P$. Cela permet d'énoncer le théorème de déduction.

Théorème de la déduction : Si E est un ensemble de fbf et P et Q sont des fbf :
Si $E, P \vdash Q$ alors $E \vdash P \to Q$ (E,P désigne E $\cup\{P\}$ ($\cup$ c'est l'union ensembliste)).

Dans le cas où E = $\emptyset$, cela nous donnerait : Si $P \vdash Q$ alors $\vdash P \to Q$.
Un cas particulier intéressant du théorème de la déduction est le suivant :
Si $E_1, E_2, ..., E_n \vdash P$ alors $E_1, E_2, ..., E_{n-1} \vdash (E_n \to P)$.

4.4 Théorie des modèles

Dans l'approche déductive du calcul propositionnel on s'est intéressé à la notion de théorème démontrable, en utilisant des axiomes et des règles d'inférence. La deuxième approche historiquement postérieure est basée sur la notion de validité et d'invérifiabilité d'une fbf. L'objectif est de pouvoir assigner à une fbf des valeurs de vérité.

La théorie des modèles ou l'aspect sémantique fait appel à la notion d'interprétation. L'interprétation d'une fbf consiste à donner des règles permettant de lui affecter une valeur de vérité vrai ou faux (que nous noterons V ou F).

pour une formule de la logique propositionnelle, cela consistera à affecter des valeurs de vérité vrai ou faux à chacun des atomes de la formule, et à évaluer l'ensemble de la formule en fonction des tables de vérité, nous disposons ainsi d'une procédure mécanique pour calculer la valeur de vérité

de n'importe quelle fbf. Une telle méthode suppose que l'on connaisse la façon dont les connectives transmettent les valeurs de vérité.

4.4.1 Notion d'interprétation et notion de modèle

Définition 4 : Une interprétation I (ou une valuation) est une application de l'ensemble des variables propositionnelles dans l'ensemble des valeurs de vérité $\{V, F\}$.

Une formule contenant n atomes aura 2^n interprétations. Dans le calcul propositionnel, le nombre d'interprétations est donc toujours fini.

on peut noter une interprétation I d'une fbf A ayant n composants par $I = \{V_1, V_2, ..., V_n\}$ où V_i tel que i $\in \{1, .., n\}$ représente un atome ou sa négation.

Définition 5 : Une interprétation donnée I peut être étendue à l'ensemble des formules par :

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	F	V	F	F	V	V

Exemple : Donnez la table de vérité de la formule suivante : $P \vee (Q \rightarrow R)$

P	Q	R	$Q \rightarrow R$	$P \vee (Q \rightarrow R)$
V	V	V	V	V
V	V	F	F	V
V	F	V	V	V
V	F	F	V	V
F	V	V	V	V
F	V	F	F	F
F	F	V	V	V
F	F	F	V	V

Définition 6 : Quand une interprétation I vérifie une fbf F, on dit que I est un modèle de la formule F.

Définition 7 : Si E est un ensemble de fbf $\{F_1, F_2, ..., F_n\}$,I est un *modèle* de E, si et seulement si I est un modèle pour toutes les formules de E. Donc : I est un modèle de E $\leftrightarrow$ I est un modèle pour $F_1 \wedge F_2 \wedge ... \wedge F_n$.

Définition 8 (conséquence logique) : Une formule A est une *conséquence logique* de $A_1, A_2, ..., A_n$ noté $A_1, A_2, ..., A_n \models A$ ssi tout modèle de $A_1, A_2, ..., A_n$ est un modèle de A.

4.4.2 Notion de validité

Définition 9 : Une fbf est dite *valide* (ou tautologique) ssi elle est vraie pour toute ses interprétations , indépendamment de la valeur de vérité des atomes qui la composent ; Autrement dit, c'est la combinaison des connectives qui en fait une tautologie.

Définition 10 : Une fbf est dite *vérifiable* s'il existe au moins une interprétation I pour laquelle la formule est vraie. De même, nous dirons qu'un ensemble E de fbf est vérifiable, si et seulement si, il existe une interprétation I qui est un modèle de E.

Définition 11 : Une fbf est dite *invérifiable* si et seulement si elle est fausse pour toutes ses interprétations.

Théorème 1 : Une fbf Q est une conséquence valide de P (ou découle logiquement de P), si et seulement si toute interprétation vérifiant P, vérifie aussi Q.
$P \models Q$ sera la notation pour Q est une conséquence valide de P.

4.5 Équivalence de deux formules et formes normales

Théorème 2 : Deux fbf P et Q sont dites équivalente (ou P est équivalent à Q) si et seulement si, les valeurs de vérité de P et Q sont les mêmes pour

toute interprétation de P et Q.

Exemple : Les formules P et Q sont équivalentes
P = $A \vee B$ et Q = $B \vee A$
En outre, si l'ensemble des symboles propositionnels de P et Q, ne sont pas les mêmes, il parait inapproprié d'utiliser cette formule, nous dirons alors pour établir l'équivalence :
deux fbf P et Q sont équivalentes si et seulement si on a à la fois $P \models Q$, et $Q \models P$. Cela revient à dire $P \equiv Q$ est valide ($P \equiv Q$ est une abréviation de P est équivalent à Q).

Théorème 3 : Pour toutes formules P, Q, R les paires de formules suivantes sont équivalentes :

- **Idempotence :** $(P \vee P) \equiv P$ et $(P \wedge P) \equiv P$.
- **Commutativité :** $(P \vee Q) \equiv (Q \vee P)$ et $(P \wedge Q) \equiv (Q \wedge P)$.
- **Associativité :** $((P \vee Q) \vee R) \equiv (P \vee (Q \vee R))$ et $((P \wedge Q) \wedge R) \equiv (P \wedge (Q \wedge R))$.
- **Absorption :** $(P \vee (P \wedge R)) \equiv P$ et $(P \wedge (P \vee R)) \equiv P$.
- **Distributivité :** $(P \wedge (Q \vee R)) \equiv ((P \wedge Q) \vee (P \wedge R))$ et $(P \vee (Q \wedge R)) \equiv ((P \vee Q) \wedge (P \vee R))$.
- **Complémentarité :** $\neg\neg P \equiv P$.
- **Lois de Morgan :** $\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$ et $\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$.
- **Tautologie :** $(F \vee G) \equiv F$ si F est une tautologie
  $(F \wedge G) \equiv G$ si F est une tautologie.
- **insatisfiabilité :** $(F \vee G) \equiv G$ si F est invérifiable
  $(F \wedge G) \equiv F$ si F est invérifiable.

Ce théorème utilisé en conjonction avec le théorème 2 , nous permet de simplifier les formules.

Exemple : Soit F = $((A \vee \neg(B \wedge A)) \wedge (C \vee (D \vee C)))$, simplifiez la formule F.

1. $((A \vee \neg(B \wedge A)) \wedge (C \vee (D \vee C))$
2. $\equiv ((A \vee (\neg B \vee \neg A)) \wedge (C \vee (D \vee C))$ 2 ème loi de Morgan
3. $\equiv ((A \vee (\neg A \vee \neg B)) \wedge (C \vee (D \vee C))$ commutativité du $\vee$
4. $\equiv ((A \vee \neg A) \vee \neg B) \wedge (C \vee (D \vee C))$ associativité du $\vee$
5. $\equiv (A \vee \neg A) \wedge (C \vee (D \vee C))$ 1 ère tautologie.
6. $\equiv (C \vee (D \vee C))$ 2 ème tautologie.
7. $\equiv (C \vee (C \vee D))$ commutativité du $\vee$
8. $\equiv (C \vee C) \vee D)$ associativité du $\vee$
9. $\equiv (C \vee D)$ idempotence.

Définition 12 : Un littéral est un atome ou la négation d'un atome.

Définition 13 : Une conjonction de littéraux est une formule composée de littéraux reliés par l'opérateur $\wedge$.

Définition 14 : Une disjonction de littéraux est une formule composée de littéraux reliés par la connective $\vee$.

Définition 15 : Une formule est dite sous **forme normale conjonctive** si et seulement si, c'est une conjonction de disjonction de littéraux.

Définition 16 : Une formule est dite sous **forme normale disjonctive** si et seulement si, c'est une disjonction de conjonction de littéraux.

Théorème 4 : Pour toute formule du calcul propositionnel, on peut trouver au moins une formule équivalente sous forme normale conjonctive, et une autre sous forme normale disjonctive. Il existe une procédure de transformation qui pour toute formule f, transforme cette formule sous une forme

normale conjonctive ou disjonctive.

Cette procédure consiste à :

- Éliminer respectivement les connectives logiques $\leftrightarrow$ puis $\rightarrow$ en utilisant les lois :
  $P \leftrightarrow Q = (P \rightarrow Q) \wedge (Q \rightarrow P)$
  $P \rightarrow Q = \neg P \vee Q$
- Réduire la portée de la négation par l'application répétitives des lois de complémentarité et de Morgan.
- Utiliser les lois de distributivité, et les équivalences entre formules.

Exemple : Soit à trouver la FNC (forme normale conjonctive) de la formule suivante : $\neg((A \wedge B) \vee (C \wedge \neg D))$.

Solution :

1. $\neg((A \wedge B) \vee (C \wedge \neg D))$
2. $\equiv (\neg(A \wedge B) \wedge \neg(C \wedge \neg D))$ 1 ère loi de Morgan
3. $\equiv ((\neg A \vee \neg B) \wedge (\neg C \vee D))$ elle est sous forme normale conjonctive.

4.6 Équivalence entre les deux approches

En conclusion, nous donnons quelques résultats qui ont été établis pour la logique propositionnelle, notamment les équivalences entre l'approche déductive et l'approche sémantique.

- **La sureté de la logique propositionnelle :** la logique propositionnelle est sure on dit aussi correcte, dans le sens où, si une formule P et démontrable (approche déductive) alors elle est valide (approche sémantique).
  Si $\vdash P$ alors $\models P$.
- **La complétude de la logique propositionnelle :** la logique propositionnelle est complète, c'est-à-dire si une fbf est valide, alors c'est un théorème.
  Si $\models P$ alors $\vdash P$.
- **La décidabilité de la logique propositionnelle :** la logique propositionnelle et décidable. Il existe un algorithme permettant de décider en un nombre fini de pas, pour toute formule d'entrée, si cette formule

est ou n'est pas un théorème : un tel algorithme est une procédure de décision.

- **La cohérence de la logique propositionnelle :** la logique propositionnelle est cohérente ; Pour toute fbf P, on ne peut pas déduire à la fois P et $\neg P$.

4.7 Exercices

Exercice 1

1. Soit A la proposition suivante : « Tous les hommes sont barbus ». Cochez les formulations correctes de la proposition $\neg A$.

☐ « Tous les hommes ne sont pas barbus.»
☐ « Aucun homme n'est barbu.»
☐ « Il existe un homme qui n'est pas barbu.»
☐ « Il existe au moins un homme qui n'est pas barbu.»
☐ « Il n'existe qu'un seul homme qui n'est pas barbu.»

2. Voici une liste de propositions A et B simples, dont vous connaissez la valeur de vérité. Dans chaque cas, exprimez la valeur de vérité de la proposition $A \wedge B$.

V F
☐ ☐ A : « Paris est la capitale de la France.» et B : « 1+1 = 2».
☐ ☐ A : « Un chat a cinq pattes.» et B : « Un carré a quatre cotés.».
☐ ☐ A : « Un triangle rectangle a un angle droit.» et B : « Deux droites parallèles se coupe en un point.».
☐ ☐ A : « 3 * 8 = 32» et B : « Paris est la capitale de la France.».
☐ ☐ A : « Berlin est la capitale de l'Espagne.» et B : « Un triangle rectangle a trois cotés égaux.».
☐ ☐ A : « Une mouche sait voler.» et B : « Le canada est un pays du continent américain.».

3. Voici une liste de propositions A et B simples, dont vous connaissez la valeur de vérité. dans chaque cas, exprimez la valeur de vérité de la proposition $A \vee B$.

V F
☐ ☐ A : « Paris est la capitale de la France.» et B : « 1+1 = 2».
☐ ☐ A : « Un chat a cinq pattes.» et B : « Un carré a quatre cotés.».
☐ ☐ A : « Un triangle rectangle a un angle droit.» et B : « Deux droites parallèles se coupe en un point.».
☐ ☐ A : « 3 * 8 = 32» et B : « Paris est la capitale de la France.».
☐ ☐ A : « Berlin est la capitale de l'Espagne.» et B : « Un triangle rectangle a trois cotés égaux.».
☐ ☐ A : « Une mouche sait voler.» et B : « Le canada est un pays du continent américain.».

Exercice 2 Les expressions suivantes sont elles des formules bien formées ?
1. $p \wedge \neg q$, 2. $p \vee \vee r$, 3. $(p \vee (\neg p))$, 4. $(p \vee \neg p)$.

Exercice 3 Donnez les tables de vérités des formules suivantes. puis indiquez les équivalences entre ces formules.
1. $\neg(p \wedge q)$, 2. $\neg p \vee \neg q$, 3. $\neg(p \vee q)$, 4. $\neg p \wedge \neg q$, 5. $p \vee (p \wedge q)$, 6. $p \wedge (p \vee q)$, 7. p

Exercice 4 Soient p et q deux variables propositionnelles signifiant respectivement « il fait froid » et « il pleut ». Écrire une phrase simple correspondant à chacun des énoncés suivants :
1. $\neg p$, 2. $p \wedge q$, 3. $p \vee q$, 4. $q \vee \neg p$, 5. $\neg p \wedge \neg q$, 6. $\neg\neg q$

Exercice 5 Soient p : « Eric lit Match » , q : « Eric lit l'Express » et r : « Eric lit les Echos ».
Donnez une formule logique pour chacune des phrases suivantes :
1. Eric lit Match ou l'Express, mais ne lit pas les Echos ».
2. Eric lit Match et l'Express, ou il ne lit ni Match ni les Echos.
3. Ce n'est pas vrai qu'eric lit Match mais pas les Echos.
4. Ce n'est pas vrai qu'Eric lit les Echos ou l'Express mais pas Match.

Exercice 6 Énigme.Trois collègues Ahmed, Ali et mostafa déjeunent ensemble chaque jour ouvrable. Les affirmations suivantes sont vraies :

1. Si Ahmed commande un dessert, Ali en commande un aussi,
2. Chaque jour, soit Mostafa, soit Ali, mais pas les deux, commandent un dessert,
3. Ahmed ou Mostafa, ou les deux, commandent chaque jour un dessert,
4. Si Mostafa commande un dessert, Ahmed fait de même.

Questions :

1. Exprimer les données du problème comme des formules propositionnelles.
2. Que peut on déduire sur qui commande un dessert ?

3. Est ce qu'on peut arriver à la même conclusion en supprimant l'une des quatre affirmations ?

Exercice 7 Connecteur de Sheffer. On définit le connecteur de Sheffeur noté | (barre de Sheffer) qui est le NAND par p| q $\equiv \neg(p \wedge q)$.

1. Donner la table de vérité de la formule (p | q).
2. Donner la table de vérité de la formule ((p | q)|(p | q)).
3. Exprimer les connecteurs $\neg$, $\vee$ et $\rightarrow$ en utilisant la barre de Sheffer.

Exercice 8 Établir les tables de vérité des formules suivantes et dites si elles sont valides, vérifiables ou invérifiables :

a. $(\neg P \wedge \neg Q) \rightarrow (\neg P \vee R)$
b. $P \wedge (Q \rightarrow P) \rightarrow P$
c. $(P \vee Q) \wedge \neg P \wedge \neg Q$
d. $(P \rightarrow Q) \wedge (Q \vee R) \wedge P$
e. $((P \vee Q) \rightarrow R) \leftrightarrow P$

Exercice 9 Trouvez les formes normales disjonctives :
a. $(A \vee B \vee C) \wedge (C \vee \neg A)$
b. $(A \vee B) \wedge (C \vee D)$
c. $\neg((A \vee B) \rightarrow C)$

Exercice 10 Trouvez les formes normales conjonctives :
a. $(A \vee B) \rightarrow (C \wedge D)$
b. $(A \vee (\neg B \wedge (C \vee (\neg D \wedge E))))$
c. $A \leftrightarrow (B \wedge \neg C)$

Exercice 11 Démontrez que les formules suivantes sont des théorèmes :
a. $\vdash A \leftrightarrow A$, sachant qu'il ne faut pas prendre $A \rightarrow A$ comme axiome.
b. $\vdash \neg B \rightarrow (B \rightarrow A)$

Exercice 12 Établir les déductions suivantes :
a. $A \rightarrow (B \rightarrow C), A \wedge B \vdash C$
b. $A \rightarrow (B \rightarrow C), B \vdash A \rightarrow C$

c. $A, B \wedge C, A \wedge C \to E \vdash E$

d. $E, E \to (A \wedge D), D \vee F \to G \vdash G$

Les axiomes de la logique propositionnelle sont :

- **1a.** $(A \to (B \to A))$
- **1b.** $(A \to B) \to ((A \to (B \to C)) \to (A \to C))$
- **1c.** $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
- **1d.** $(A \to B) \to ((B \to C) \to (A \to C))$
- **2.** $A \to (B \to A \wedge B)$
- **3a.** $A \wedge B \to A$
- **3b.** $A \wedge B \to B$
- **4a.** $A \to A \vee B$
- **4b.** $B \to A \vee B$
- **5.** $(A \to C) \to ((B \to C) \to (A \vee B \to C))$
- **6.** $B \to ((B \to C) \to C)$
- **7.** $A \to A$
- **8.** $(\neg A \to \neg B) \to (B \to A)$
- **9.** $(A \to B) \to ((B \to A) \to (A \leftrightarrow B))$

Chapitre 5

Chapitre 5 : La logique des prédicats du premier ordre

5.1 Introduction

La logique des prédicats a pour but de généraliser la logique des proposition. On peut considérer un prédicat comme un énoncé général où apparaissent des variables. Par exemple : "L'amphi X est grand" "Si X est père de Y et de Z alors Y et Z sont frères".

Si l'on remplace toutes les variables d'un prédicat par des valeurs définies on obtient une proposition à laquelle on pourra associer une interprétation (vrai, faux). Ainsi, X = A1, dans le premier exemple donne "L'amphi A1 est grand". Un prédicat représente donc une classe de propositions.

Par l'introduction de quantificateurs on peut représenter le fait qu'un énoncé est vrai pour toutes les valeurs possibles des variables ou qu'il existe au moins une valeur des variables qui rend l'énoncé vrai. Par exemple : "quelque soit X si X est un homme alors X est mortel". Les variables pouvant prendre leur valeur dans des ensembles infinis, les quantificateurs permettent donc d'énoncer des faits correspondant à une infinité de propositions.

5.2 Langage

5.2.1 L'alphabet

L'alphabet de la logique des prédicats est constitué de :

- Un ensemble dénombrable de symboles de prédicats à 0, 1, ou plusieurs arguments, notés p, q, r, ..., homme, mortel, père,....
- Un ensemble dénombrable de variables d'objets (ou variables d'individu), notées x, y, z, x_1, x_2,.....
- Un ensemble dénombrable de fonctions à 0, 1, ou plusieurs arguments, notées f, g,...., père-de,....
- Les quantificateurs $\forall$ et $\exists$.
- Les connecteurs $\neg, \wedge, \vee, \rightarrow$ ainsi que les parenthèses de la logique propositionnelle.

Notation.
Les fonctions à 0 arguments sont appelées constantes (souvent notées a, b, ..., Socrate,...).
Les prédicats à 0 arguments ne sont rien d'autre que des variables propositionnelles.

Exemple : Écrire en langage formel de la logique du premier ordre les assertions suivantes :

Tous les chats, sont noirs : $\forall x(Chat(x) \rightarrow Noir(x))$
Certains chats sont noirs : $\exists x(Chat(x) \wedge Noir(x))$
Aucun chat, n'est noir : $\forall x \neg(Chat(x) \wedge Noir(x))$
Dans le domaine des entiers naturels, pour tout entier naturel, il existe un nombre premier qui lui est supérieur : $\forall x \exists y(Pre(y) \wedge Sup(y, x))$

5.2.2 Définitions

Un terme : l'ensemble des termes est le plus petit ensemble de mots construits sur l'alphabet de la logique des prédicats tel que :

- Toute variable est un terme,
- f($t_1, ..., t_n$) est un terme si f est une fonction à n arguments et $t_1, ..., t_n$ sont des termes.

Formule atomique : Si P est un prédicat à n arguments et $t_1, ..., t_n$ sont des termes alors P($t_1, ..., t_n$) est une formule atomique. Quand n = 0, la formule atomique est une proposition.

Formule bien formées : Une formule atomique est une fbf.
Si P et Q sont des fbfs, et x une variable, alors : $\neg P$, $P \wedge Q, P \vee Q, P \rightarrow Q, P \leftrightarrow Q, (\forall x)P, et(\exists x)P$ sont des fbfs.
La priorité entre les connectives est la même que dans la logique propositionnelle ; Quand aux quantificateurs "$\forall$" et "$\exists$", ils ont la même priorité que "$\neg$".

Littéral : Si L est une formule atomique, (L) et $(\neg L)$ sont appelés des littéraux. L et $\neg L$ sont dits littéraux complémentaires.

5.2.3 La portée d'un quantificateur

Une variable est **liée** dans une formule si et seulement si elle est dans la portée d'un quantificateur. Dans les fbfs $(\forall x)$P, et $(\exists x)$P, P est la **portée** des quantificateurs.

Une variable qui n'est pas liée dans une formule est dite **libre** . Une variable peut donc être libre et liée dans la même formule.

Dans la fbf : $(P(x) \vee \forall x Q(x))$, x est libre dans P(x), mais elle est liée dans Q(x).

Une fbf **fermée** est une fbf qui ne contient pas de variables libres (elle contient seulement des constantes ou des variables quantifiées universellement ou existentiellement), sinon elle est **ouverte**.

Si P est une fbf et t un terme, t est dit libre pour x dans P, si et seulement si lorsqu'on remplace une occurrence libre de x par t aucune variable de t ne devient liée. Dans la suite, on ne considère que les formules sans variable libre.

La fermeture d'une fbf P, est définie comme la fbf fermée obtenue à partir de P, en la préfixant avec des quantificateurs universels portant sur les variables libres dans P.

Exemple : La fermeture de : $P(x, y, z) \rightarrow (\exists y)Q(x, y, z)$ est
$\forall x \forall y \forall z(P(x, y, z) \rightarrow (\exists y)Q(x, y, z))$

5.2.4 Les formules congrues

Considérons la formule : $\forall x(P(x) \wedge \exists x Q(x,z) \rightarrow \exists y R(x,y)) \vee Q(z,x)$ (1)

Dans la partie $\exists x Q(x,z)$, chaque x est lié par $\exists x$, ce qu'on peut indiquer en attachant un indice 1 à chacun des x pour montrer qu'ils vont ensemble. On fera de même pour les autres variables. En mettant les indices on doit toujours partir de l'intérieur, en procédant selon l'ordre suivi dans la construction de la formule. Par souci de normalité, on choisira de commencer la numérotation à chaque étape par le quantificateur le plus à gauche. Soit la formule (1) :

1- $\forall x(P(x) \wedge \exists x Q(x,z) \rightarrow \exists y R(x,y)) \vee Q(z,x)$
2- $\forall x_3(P(x_3) \wedge \exists x_1 Q(x_1,z) \rightarrow \exists y_2 R(x_3,y_2)) \vee Q(z,x)$

les occurrences demeurées sans indices sont libres. En effaçant les variables liées, on obtient :

3-$\forall 3(P(3) \wedge \exists 1 Q(1,z) \rightarrow \exists 2 R(3,2)) \vee Q(z,x)$

Soit la formule (2) :

1- $\forall y(P(y) \wedge \exists x Q(x,z) \rightarrow \exists z R(y,z)) \vee Q(z,x)$
2- $\forall y_3(P(y_3) \wedge \exists x_1 Q(x_1,z) \rightarrow \exists z_2 R(y_3,z_2)) \vee Q(z,x)$
3- $\forall 3(P(3) \wedge \exists 1 Q(1,z) \rightarrow \exists 2 R(3,2)) \vee Q(z,x)$

Ceci montre que les formules (1) et (2) sont **congrues**.

5.2.5 La substitution et l'instantiation

L'opération de substitution consiste à remplacer certaines variables libres d'une formule libre F par des termes.

Si F est une formule où $x_1, x_2, ..., x_n$ sont des variables libres, et σ la substitution $(t_1/x_1, ..., t_n/x_n)$, Fσ dénote la formule obtenue en remplaçant toutes les occurrences de x_i dans F par t_i tel que i = 1 à n.

Exemple : F = $Q(x,y_1) \leftrightarrow \forall x(R(x,z_1) \vee S(x,y_1))$
$\sigma = (z_8/x, g(a,b)/y_1)$

F$\sigma = Q(z_8, g(a,b)) \leftrightarrow \forall x(R(x,z_1) \vee S(x, g(a,b)))$

On dira qu'une substitution instancie x si elle remplace x par un terme où n'apparait aucune variable. Ainsi, la substitution $(z_8/x, g(a,b)/y_1)$ instancie y_1 mais pas x.

5.3 La méthode déductive : Théorie de la preuve

De la même manière que pour la logique propositionnelle, nous donnons les axiomes et les règles d'inférence de la logique du premier ordre.

5.3.1 Les axiomes

Nous gardons les schémas d'axiomes de la logique propositionnelle, auxquels on a rajouté :

- Le schéma universel : $\forall x P(x) \to P(a)$
- Le schéma existentiel : $P(a) \to \exists x P(x)$

5.3.2 Les règles d'inférence

Modus Ponens : P,$P \to Q \vdash Q$

La règle universelle : Elle permet de passer de la forme $C \to A(x)$ à la forme $C \to \forall x A(x)$

La règle existentielle : Elle permet de passer de la forme $A(x) \to C$ à la forme $\exists x A(x) \to C$ C ne contenant pas d'occurrence libre de x.

Théorème de déduction : Si E est un ensemble de fbf fermées et P et Q sont des fbf fermées : Si E, P $\vdash$ Q, alors E $\vdash P \to Q$ (E, P désigne E $\cup\{P\}$)

Exemple : Établir la déduction suivante : $\forall y(R \to P(y)) \vdash R \to \forall x P(x)$

1- $\forall y(R \to P(y))$	Hyp
2- $\forall y(R \to P(y)) \to (R \to P(y))$	sh $\forall$
3-$R \to P(y)$	m.p 1,2

4- $R \rightarrow \forall y P(y)$ règle $\forall$
5- $(R \rightarrow \forall y P(y)) \rightarrow ((R \rightarrow (\forall y P(y) \rightarrow \forall x P(x))) \rightarrow (R \rightarrow \forall x P(x)))$ sh.1b
6- $(R \rightarrow (\forall y P(y) \rightarrow \forall x P(x))) \rightarrow (R \rightarrow \forall x P(x))$ m.p 4,5
7- $\forall y P(y) \rightarrow P(x)$ sh.$\forall$
8- $\forall y P(y) \rightarrow \forall x P(x)$ règle$\forall$
9- $(\forall y P(y) \rightarrow \forall x P(x)) \rightarrow (R \rightarrow (\forall y P(y) \rightarrow \forall x P(x)))$ sh.1a
10- $R \rightarrow (\forall y P(y) \rightarrow \forall x P(x))$ m.p 8,9
11- $R \rightarrow \forall x P(x)$ m.p 6,10

5.4 L'approche sémantique

Nous gardons la même démarche que celle adoptée dans la présentation de la logique propositionnelle. Après avoir répondu à la question si une formule F est démontrable ou pas (en utilisant l'approche déductive), on se pose maintenant la question de savoir si une fbf F est valide ou non. On ne pourra pas procéder comme pour la logique propositionnelle, et l'un des problèmes de la logique du premier ordre va consister à trouver les procédures permettant d'établir qu'une formule est invérifiable.

5.4.1 Notion d'interprétation

Dans la logique propositionnelle, on a noté que l'interprétation d'une formule consiste à affecter des valeurs de vérité vrai ou faux à chacun des atomes et à évaluer l'ensemble en fonction des tables de vérité des différents connecteurs. Dans la logique du premier ordre, une interprétation I d'une formule P est définie par :

- Un ensemble non vide D appelé le domaine de l'interprétation,
- Une correspondance entre les constantes et les éléments de D,
- Une correspondance entre chaque fonction n-aire et une application de $D^n \longrightarrow D$,
- Une correspondance entre chaque prédicat n-aire et une application de $D^n \longrightarrow \{V, F\}$.

Interprétation des variables : A chaque variable, on fait correspondre un élément de D.

Interprétation des termes :

- Les variables sont interprétées comme ci-dessus,
- A chaque constante, on fait correspondre un élément de D,
- Si $t'_1, t'_2, ..., t'_n$ sont les interprétations des termes $t_1, t_2, ..., t_n$ et f' l'interprétation de f, alors l'élément $f'(t'_1, t'_2, ..., t'_n)$ de D est l'interprétation du terme $f(t_1, t_2, ..., t_n)$.

Interprétation des formules dans $\{Vrai, Faux\}$:

- Si la formule $P(t_1, t_2, ..., t_n)$ est un atome, sa valeur de vérité est celle de $P'(t'_1, t'_2, ..., t'_n)$ où P' est l'application associée à P et t'_i est l'interprétation du terme t_i,

- Si la formule est de la forme $\neg P, P \wedge Q, P \vee Q, P \rightarrow Q, P \leftrightarrow Q$, alors la valeur de vérité de la formule est donnée par la table de vérité des différentes connectives,

- Si la formule est de la forme $\exists x P$, la valeur de vérité de cette formule est "vrai" s'il existe un élément d de D tel que P est vrai relativement aux interprétations des variables, des termes et formules, à condition de remplacer la variable x par d. Dans les autres cas, la valeur de vérité est "faux",

- Si la formule est de la forme $\forall x P$, alors la valeur de vérité de cette formule est "vrai", si pour tout élément d de D, P a la valeur "vrai" par rapport à I, à condition de remplacer x par d (notation P(x/d)), sinon la valeur de vérité est "faux".

La valeur de vérité d'une formule fermée ne dépend pas de l'interprétation des variables. On peut donc parler de la valeur d'une formule fermée, relativement à une interprétation.

Contrairement à ce qui se passe pour la logique propositionnelle, il y a en général un nombre infini d'interprétations pour une formule de la logique des prédicats du premier ordre ; Il n'y a donc pas d'équivalent à la notion de table de vérité.

Exemple : Soit la formule $F = \forall x(\exists x P(x) \rightarrow P(x)) \wedge P(y)$, à interpréter sur le domaine D = $\{1, 2\}$.

On cherche d'abord à connaitre les variables libres et les variables liées. Ainsi, on a :
$\forall x_2(\exists x_1 P(x_1) \rightarrow P(x_2)) \wedge P(y)$, la variable y est libre dans la formule, on donne donc les interprétations relatives à x :

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
1	V	V	F	F
2	V	F	V	F

La table de vérité est :

P(x)	y	$\forall x(\exists x P(x) \rightarrow P(x)) \wedge P(y)$
f_1	1	V
f_1	2	
f_2	1	
f_2	2	
f_3	1	F
f_3	2	
f_4	1	
f_4	2	

Le calcul de la valeur de vérité de la première ligne est le suivant :

$$\forall x \quad v \left\{ \begin{array}{l} x = 1 \quad v \quad \exists x \quad v \left\{ \begin{array}{ll} f_1(1) & v \\ ou & \\ f_1(2) & v \end{array} \right. \qquad \rightarrow f_1(1) \quad v \\ et \\ x = 2 \quad v \quad \exists x \quad v \left\{ \begin{array}{ll} f_1(1) & v \\ ou & \\ f_1(2) & v \end{array} \right. \qquad \rightarrow f_1(2) \quad v \end{array} \right\} \wedge f_1(1) \quad v$$

5.4.2 Notion de validité et notion de modèle

Soient F une fbf fermée, et $S = \{P_1, P_2, ..., P_n\}$ un ensemble fini de fbf fermées. F et S appartenant au calcul des prédicats du premier ordre.

Définition 1 : F est **vérifiable** si et seulement si, il existe une interprétation I telle que la valeur de vérité de F relativement à I est "vrai".

Si F est vrai pour l'interprétation I, on dit que I est un modèle pour F, et que I vérifie F. S est **vérifiable** s'il existe une interprétation I qui est un modèle pour chaque formule de S. Donc : I est modèle pour S, si et seulement si, I est un modèle pour $P_1 \wedge P_2 \wedge .... \wedge P_n$ sinon S est **invérifiable**.

Définition 2 : F est **valide** , si et seulement si, chaque interprétation I pour F vérifie F. S est valide si chacune de ses interprétations est un modèle pour S.

Définition 3 : F est dite **conséquence valide** de S, si pour toute interprétation I , I est un modèle de S implique que I est un modèle pour F.

F est une **conséquence logique** de S, si et seulement si, $P_1 \wedge P_2 \wedge .... \wedge P_n \rightarrow F$ est valide.

Théorème 2 : F est une conséquence valide de S, si et seulement si , $S \cup \{\neg F\}$ est invérifiable.

5.5 L'équivalence entre les deux approches

A l'instar de la logique propositionnelle, nous concluons le chapitre par l'équivalence entre l'approche déductive et l'approche sémantique.

La sureté et la complétude : La logique du premier ordre est sure, si une fbf est démontrable, alors elle est valide

$$\vdash P \rightarrow \models P$$

La logique du premier ordre est complète. Une fbf P est un théorème si et seulement si, elle est valide (théorème de complétude établit par Godel)

$$\models P \leftrightarrow \vdash P$$

La complétude de la logique des prédicats est intéressante, puisqu'on peut montrer la validité d'une fbf en utilisant la théorie de la preuve.

La décidabilité : La logique des prédicats est **semi-décidable**, car il existe une procédure de preuve qui permet d'établir qu'une fbf est un théorème (en appliquant des règles d'inférences aux axiomes et aux théorèmes). Pour les fbf qui ne sont pas des théorèmes ces procédures ne donnent pas de réponses en un nombre fini de pas.

5.6 Exercices

Exercice 1 Modélisez les phrases suivantes en logique des prédicats, précisez le vocabulaire utilisé.

- Tous les étudiants aiment la logique.
- Tous les étudiants n'aiment pas une matière.
- Les étudiants qui ont une bonne note en logique sont les meilleurs.

Exercice 2 Soient les phrases suivantes :
a- Tous les hommes sont mortels.
b- Socrate est un mortel.
c- Socrate est un homme.

- Identifier les prédicats.
- Exprimer dans la logique des prédicats a, b, c.
- peut on déduire l'énoncé b à partir de a et c ? justifier.

Exercice 3 Soient les énoncés suivants :

1. Les personnes qui ont la grippe A doivent prendre du Tamiflu.
2. Les personnes qui ont de la fièvre et qui toussent ont la grippe A.
3. Ceux qui ont une température supérieure à 38 ont de la fièvre.
4. Mohamed tousse et a une température supérieure à 38.
5. Mohamed doit prendre du Tamiflu.

Modélisez en logique du premier ordre les énoncés ci-dessus en utilisant les prédicats suivants :

- grippe (x) : x a la grippe A.
- prendre (x,y) : x doit prendre y.
- fievre (x) : x a de la fièvre.
- tousse (x) : x tousse.
- temp (x, t) : x a la température t.
- sup (x, y) : x est supérieur à y.

utilisez aussi les constantes suivantes : 38, Mohamed, Tamiflu.

Exercice 4 Établir la table de vérité des formules suivantes : (sachant que le domaine d'interprétation est D = $\{1, 2\}$)
a- $\forall x(P(x) \rightarrow \exists x Q(x))$.
b- $\forall x(P(x, y) \wedge \exists x P(x))$.

Exercice 5
a- Établir les déductions suivantes :
- $\forall x \forall y A(x,y) \vdash A(x,y)$.
- $\forall x(P(x) \to Q(x)), \forall x P(x) \vdash \forall x Q(x)$.
- $P(a), \forall x(P(x) \to Q(x)) \vdash Q(a)$.

b- Démontrer : $\forall x P(x) \to \exists x P(x)$.

5.7 Note bibliographique

Les définitions du chapitre 2 ont été extraites du livre de Jean Yves Girard traduit de l'anglais par Julien Basch et Patrice Blanchard [Gir99]. Le chapitre 3 reprend quelques notes du cours "logique formelle" fait par Mme Kempf [Kem07], du livre de salem "introduction à la logique formelle" [Sal91] et du livre de Gérard Chazal [Cha96] avec quelques exercice de la série de TD de Mme Bahi [Bah06].

Les définitions des chapitres 4 et 5 ont été extraites des livres " Mathematical logic and formalized theories" [L.R74], "logique mathématique" [KCD03] et "Éléments de logique mathématique théorie des modèles " [KK67].

Bibliographie

[Bah06] H. Bahi. *systeme formel serie de TD.* Université Badji Mokhtar annaba, 2006.

[Cha96] Gérard Chazal. *Elements de logique formelle, édition Hermes.* Hermès Lavoisier, 1996.

[Gir99] Jean Ives Girard. *La machine de turing.* Seuil, 1999.

[KCD03] J.L. Krivine, R. Cori, and D.Lascar. *Logique mathématique 1- Calcul propositionel, Algebre de Boole, Calcul des prédicats.* DUNOD Paris, 2003.

[Kem07] Kempf. *Logique formelle.* 2007.

[KK67] G. Kreisel and J.L. Krivine. *Elements de la logique mathématique théorie des modèles.* DUNOD Paris, 1967.

[L.R74] Robert L.Rogers. *Mathematical logic and formalized theorie.* Elsevier, 1974.

[Sal91] Jean Salem. *Introduction à la logique formelle et symbolique.* Nathan université, 1991.