
LINGI1101

Logique et Structures Discrètes

Titulaire : Peter VAN ROY

Table des matières

Remerciements	6
1 Introduction au cours LINGI1101	7
1.1 Déroulement du cours	7
1.2 Plan du cours	8
1.2.1 Logique des propositions	8
1.2.2 Logique des prédicats	8
1.2.3 Interprétations et modèles	8
1.2.4 Théorie de preuve	8
1.2.5 Algorithme de preuve	8
1.2.6 Théorie logique	9
1.2.7 Programmation logique	9
2 Contexte : la méthode scientifique	10
2.1 Formalisation d'un système	10
2.2 Boucle de raisonnement	10
2.2.1 Dédution	10
2.2.2 Induction	11
2.2.3 Abduction	11
2.3 Exemples	11
2.3.1 Loi de Maxwell	11
2.3.2 Sac de billes	12
3 La logique propositionnelle	14
3.1 La grammaire	15
3.2 Les tables de vérité	16
3.3 Les interprétations	17
3.4 Les modèles logiques	19
3.4.1 Conséquence logique	20
3.4.2 Équivalence logique	20
4 Les preuves	22

4.1	Preuve avec table de vérité	22
4.2	Preuve transformationnelle	22
4.3	Preuve déductive	23
4.4	Exemple de preuve propositionnelle	25
5	Deux règles plus sophistiquées	27
5.1	Théorème de déduction	27
5.1.1	Prémisse :	27
5.1.2	Conclusion :	27
5.2	Preuve par contradiction (ou preuve indirecte)	27
5.2.1	Prémisse :	27
5.2.2	Conclusion :	27
6	Exemples de Preuves	27
6.1	Prémisse :	28
6.2	Conclusion :	28
6.3	Exemple sans preuve conditionnelle	28
6.4	Exemple avec preuve conditionnelle	28
6.5	Exemple de Preuve par contradiction	28
7	Quelques concepts	29
7.1	Principe de dualité	29
7.1.1	Dans les formules sans \rightarrow :	29
7.1.2	Formule quelconque :	29
7.2	Forme Normale	29
7.2.1	terminologie	29
7.2.2	Forme normale conjonctive FNC	29
7.3	Algorithme de normalisation	29
7.3.1	Exemple	30
8	Algorithme de preuve	31
8.1	La résolution	31
8.1.1	Exemple de résolution	31
8.1.2	Principe de résolution	32

8.1.3	La résolution préserve les modèles	32
8.2	Algorithme	33
8.2.1	Pseudocode	33
8.3	Exemples	34
8.3.1	Exemple 1	34
8.3.2	Exemple 2	34
8.4	Conclusion	34
9	La logique des prédicats	36
9.1	Introduction	36
9.2	Quantificateurs	37
9.3	Syntaxe	39
9.4	Grammaire	39
9.4.1	Règles de formation	39
9.5	Sémantique	40
10	Logique des prédicats	42
10.1	Détails techniques	42
10.2	Sémantique	42
10.3	Différence avec la logique des propositions	43
10.4	Preuves avec règles	43
11	Preuves en logique des prédicats	45
11.1	Exemple	45
12	Règles en logique des prédicats	47
12.1	La substitution	47
13	Elimination de \forall	49
14	Elimination de \exists	49
15	Introduction de \exists	50
16	Introduction de \forall	50

17 Partie 9	52
18 Algorithme de preuve pour la logique des prédicats	53
18.1 3 transformations :	53
18.2 Résolution	54
18.3 Propriétés de cet algorithme	55
18.4 Transformation de la formule de base vers la forme prénexe .	55
19 Introduction	57
19.1 3 transformations	57
19.2 Résolution	57
19.3 Propriétés de l'algorithme	57
20 Transformation en forme prénexe	57
21 Transformation en forme Skolen	57
21.1 Intuition	57
21.2 Règle	57
22 Transformation en forme normale conjonctive	58
23 La règle de résolution	58
24 Algorithme	59
25 Exemple	60
25.1 Initialisation de S	60
25.2 Itérations	60
25.3 Non-déterminisme	60
26 Stratégies	61
27 Extension d'une théorie	66
28 Liens entre théories	67
29 Théorie des ordres partiels stricts	68

29.1 Exemples de programmes Prolog	70
29.1.1 Exemple 2	71
Références	72

Remerciements

Je tiens à remercier les étudiants de LINGI1101 pour avoir pris des notes pendant mon cours, ce qui faisait la base de ce syllabus. Les contributeurs sont : Antoine Walsdorff, Goeric Huybrechts, Romane Schelkens, Nicolas Van Wallendael, Kilian Verhetsel, Cyril de Vogelaere, Jonathan Legat.

1 Introduction au cours LINGI1101

Le cours *Logique et structures discrètes* a deux buts importants :

- Donner la motivation et l'intuition de la logique, pour que cette matière devienne véritablement utile pour les étudiants.
- Donner les concepts et les formalismes mathématiques nécessaires pour utiliser la logique à bon escient.

L'intuition est donc importante pour ce cours, mais néanmoins la connaissance des formalismes mathématiques reste essentielle. Le cours sera coté sur les deux : intuitions (un tiers) et formalismes (deux tiers).

1.1 Dérroulement du cours

Le cours est composé de deux parties. La première partie, *logique formelle*, représentera deux tiers du cours. La seconde partie, *structures discrètes sur Internet*, comptera quant à elle pour un tiers du cours.

L'évaluation de ce cours se compose de trois parties. Il y aura tout d'abord une interrogation au milieu du quadrimestre portant sur 5 points. Il vous sera également demandé de prendre note pendant une heure de cours par groupe de trois, ceci afin de contribuer au syllabus. Ces notes prises au cours rapporteront au maximum 2 points de la note finale à chacun des participants. L'examen sera divisé en deux parties. La première partie sur 5 points portera sur la matière de l'interrogation. La note retenue sera le maximum entre la note de l'interrogation et celle obtenue à la question de l'examen. La seconde partie de l'examen sera donc cotée sur 13 points et portera sur le reste de la matière.

Afin de suivre ce cours, nous nous baserons sur deux livres de références correspondants chacun à une partie du cours :

- Introductory Logic and Sets for Computer Scientists, by *Nimal Nis-sanke*.
- Networks, Crowds, and Markets : Reasoning About a Highly Connected World, by *David Easley and Jon Kleinberg*.¹

La première partie sera complétée par des sujets et exercices plus avancés qui approfondissent le traitement du livre.

1. Quelques chapitres.

1.2 Plan du cours

Cette partie va parler du rôle des raisonnements et des différentes formes de raisonnement. Nous prendrons en exemple la méthode scientifique.

1.2.1 Logique des propositions

La logique des propositions est un langage formel constitué d'une syntaxe et d'une sémantique. La syntaxe décrit l'ensemble des formules qui appartiennent au langage. La sémantique permet de donner un sens aux formules de langage. C'est une logique très ancienne qui vient de l'antiquité.

1.2.2 Logique des prédicats

C'est la logique la plus expressive et la plupart des travaux mathématiques peuvent être écrits dans ce langage. Elle est aussi définie comme la logique du premier ordre.² En logique des prédicats, les éléments de base du langage ne sont plus des propositions mais des prédicats.

1.2.3 Interprétations et modèles

La logique a besoin d'un langage, de phrases pour la décrire. Cette section couvrira donc la sémantique à utiliser.

1.2.4 Théorie de preuve

Nous pouvons manipuler une phrase en logique pour obtenir un résultat. Par exemple, si A et B sont vrais, nous pouvons en déduire que A est vrai. Il y a des règles d'inférences à utiliser pour prendre une phrase en logique et en déduire une autre. Une preuve mathématique est une séquence de phrases liées par des règles d'inférences.

1.2.5 Algorithme de preuve

C'est l'algorithme le plus puissant qui existe en logique des prédicats. Néanmoins, il est inefficace seul. Afin de le rendre efficace, il faut poser des hypothèses. Nous approfondirons ce problème dans le cadre de cette section.

2. Il existe d'autres formes de logiques plus expressives mais plus difficiles à utiliser. Exemple : la logique du deuxième ordre, ...

1.2.6 Théorie logique

Il est possible de formaliser tout objet mathématique avec une théorie logique qui lui est propre. En exemple, citons la théorie des ensembles, des fonctions et des ordres partiels.

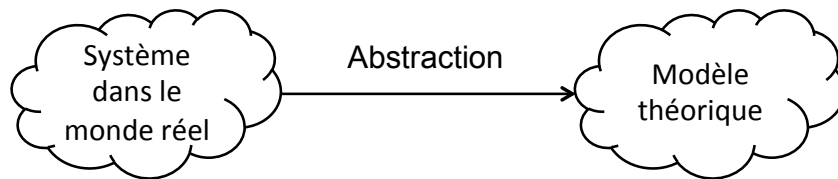
1.2.7 Programmation logique

Le rêve serait de pouvoir exprimer toute chose logique en langage de programmation efficace. Il s'agira d'appliquer ce principe avec l'algorithme de preuve, sur base d'hypothèses.

2 Contexte : la méthode scientifique

2.1 Formalisation d'un système

Comment pouvons-nous formaliser un système dans le monde réel tel que les champs magnétiques ou la gravitation ?

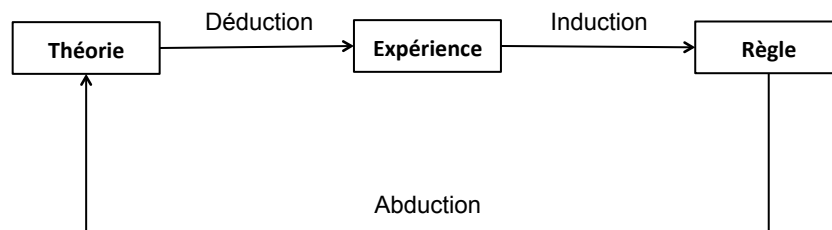


Afin de formaliser un système dans le monde réel, nous devons faire une abstraction vers un modèle théorique. Ce modèle théorique est un ensemble de phrases logiques dont il est possible de tirer des prédictions. Il n'est intéressant que s'il se comporte comme le vrai système.

Un exemple de cette formalisation pourrait être les équations de Maxwell qui sont le modèle théorique correspondant à la trajectoire des électrons et protons.

2.2 Boucle de raisonnement

Il existe trois formes de raisonnement : la déduction, l'induction et l'abduction. Ces trois formes de raisonnement peuvent être liées dans une boucle de raisonnement de la façon suivante :



2.2.1 Déduction

Il s'agit de faire des calculs et des raisonnements logiques par rapport à la théorie. On en conclut une expérience.

2.2.2 Induction

L'induction est le fait de trouver une règle générale qui décrit les résultats d'expériences. On choisit en général une règle moyenne qui deviendra la règle générale. Il faut souligner que les résultats expérimentaux ne sont pas totalement fiables ou incomplets. Dès lors, la règle trouvée n'est pas nécessairement exacte. Par exemple, si par induction nous avons trouvé la règle, "les oiseaux volent", cela est vrai tant que l'on a pas vu un pingouin. Autre exemple, nous pouvons supposer que demain le soleil va se lever comme depuis des milliers d'années, même si rien ne l'assure.

2.2.3 Abduction

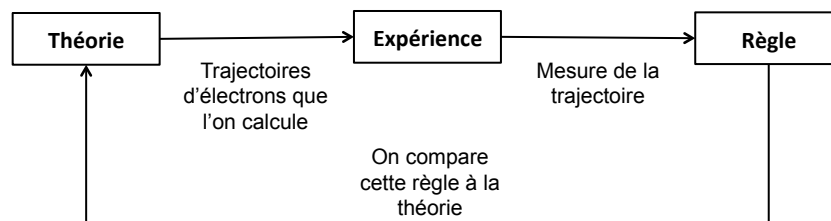
On compare la règle générale trouvée lors de l'induction avec la théorie. Si cela se révèle différent, comme l'on suppose que tout est parfait dans les calculs/expériences, c'est la théorie qui est fausse. Il faut alors corriger la théorie existante ou en inventer/deviner une nouvelle. Ce type de raisonnement est l'abduction. On applique l'abduction couramment dans notre vie de tous les jours ; par exemple, lorsqu'un élève entre trempé dans la classe, nous supposons qu'il pleut dehors.

Sur ces trois concepts seul la déduction est un raisonnement sûr, les deux autres sont encore mal compris. Nous nous focaliserons dans ce cours uniquement sur la déduction.

2.3 Exemples

2.3.1 Loi de Maxwell

Nous illustrons dès à présent le fonctionnement de la boucle de raisonnement à l'aide de l'exemple cité plus haut, c'est-à-dire les équations de Maxwell :



Par déduction, grâce à la théorie et aux conditions initiales que nous fixons, nous trouvons la trajectoire d'un électron. Nous effectuons ensuite des mesures dans le monde réel. Nous allons, par exemple, mesurer la trajectoire plusieurs fois avec des méthodes différentes et, par induction, nous trouvons une règle qui est la loi de comportement de la particule. Nous comparons ensuite, par abduction, cette règle à la théorie, et nous la corrigeons si besoin.

2.3.2 Sac de billes

Afin d'illustrer les 3 formes de raisonnement de manière plus formelle, considérons un sac de billes pouvant contenir des billes noires ou blanches. Notons que $sac(x)$ signifie "la bille x est dans le sac" et que $blanc(x)$ signifie "la bille x est blanche".

Déduction

1. Règle : $\forall x, sac(x) \Rightarrow blanc(x)$
2. Cas : $sac(a), sac(b), \dots$

3. Résultat : $blanc(a), blanc(b), \dots$

Si toutes les billes se trouvant dans le sac sont blanches et que l'on pioche une bille de ce sac, cette bille sera blanche. Cette déduction est forcément correcte.

Induction

1. Cas : $sac(a), sac(b), \dots$
2. Résultat : $blanc(a), blanc(b), \dots$

3. Règle : $\forall x, sac(x) \Rightarrow blanc(x)$

Si toutes les billes qu'on pioche du sac sont blanches, alors nous pouvons établir comme règle que toutes les billes dans le sac sont blanches. Cette induction n'est pas forcément correcte.

Abduction

1. Règle : $\forall x, sac(x) \Rightarrow blanc(x)$
2. Résultat : $blanc(a), blanc(b), \dots$

3. Cas : $sac(a), sac(b), \dots$

Si toutes les billes se trouvant dans le sac sont blanches et que nous trouvons des billes blanches à côté du sac, nous pouvons penser qu'elles viennent du sac. Cette abduction n'est pas forcément correcte.

3 La logique propositionnelle

« *They don't even need to know what they're talking about.* » —
Richard Feynman à propos des mathématiciens.

La logique propositionnelle est la plus simple des formes de logique. Elle permet de formaliser des expressions telles que les suivantes :

1. « S'il fait beau, alors je vais dehors. »
2. « Cet homme est grand et fort. »
3. « Il fait jour mais pas nuit. »

Plus précisément, nous partons d'un ensemble de propositions premières. Par exemple :

1. il fait beau ;
2. je vais dehors ;
3. cet homme est grand ;
4. cet homme est beau ;
5. il fait jour ;
6. il fait nuit.

Le contenu exact de ces propositions n'a en fait aucune importance. C'est pourquoi elles seront remplacées par des lettres majuscules :

1. $A =$ « il fait beau » ;
2. $B =$ « je vais dehors » ;
3. $C =$ « cet homme est grand » ;
4. $D =$ « cet homme est beau » ;
5. $E =$ « il fait jour » ;
6. $F =$ « il fait nuit ».

Une proposition logique est alors :

- soit une des propositions premières ;
- soit une combinaison de propositions logiques connectées par des connecteurs logiques.

Ainsi, les exemples de propositions précédentes peuvent être réécrites comme ceci (la signification précise des différents symboles sera décrite plus loin) :

1. $A \Rightarrow B$;
2. $C \wedge D$;

3. $E \wedge \neg F$.

L'avantage de cette notation par rapport aux phrases en français est qu'elle nous permet d'effectuer des raisonnements formels sur les propositions logiques. En particulier, nous pouvons précisément définir :

1. une **grammaire** qui définit ce qui est une proposition logique et ce qui ne l'est pas ;
2. une **sémantique** qui donne un sens à chaque proposition logique ;
3. une **théorie de preuve** permettant, en sachant qu'une proposition est vraie, de trouver d'autres propositions vraies (par exemple à partir de $A \Rightarrow B$ on peut trouver $\neg B \Rightarrow \neg A$).

3.1 La grammaire

La logique propositionnelle est un **langage** formel. Il peut être défini à l'aide d'une grammaire sur un **alphabet**. L'alphabet est l'ensemble des symboles qui composent une proposition logique, c'est-à-dire :

- les lettres majuscules représentant les différentes propositions premières : A, B, C , etc. ;
- true et false qui représentent des propositions qui sont respectivement toujours vraies et toujours fausses ;
- les différents connecteurs logiques :
 - Conjonction : \wedge
 - Disjonction : \vee
 - Négation : \neg
 - Implication : \Rightarrow
 - Equivalence : \Leftrightarrow
- les caractères de ponctuation « (» et «) ».

Cependant, toutes les séquences composées de ces caractères ne sont pas des phrases propositionnelles. La grammaire suivante permet de donner les règles que les phrases propositionnelles doivent respecter :

```
<identificateur> = A | B | C | D | ...
<proposition>    = true
                  | false
                  | <identificateur>
                  | (<proposition>)
                  | ¬<proposition>
                  | <proposition> ∧ <proposition>
                  | <proposition> ∨ <proposition>
                  | <proposition> ⇒ <proposition>
                  | <proposition> ⇔ <proposition>
```


Remarquez que seules les séquences de symboles qui respectent cette grammaire sont des phrases propositionnelles. Ainsi, $p \Leftrightarrow q$ n'est pas une phrase propositionnelle parce que les propositions premières sont des lettres majuscules ; de même les phrases en français — ou en klingon, ou encore dans d'autres formalismes mathématiques — comme « s'il fait beau alors je vais dehors » ne respectent pas la grammaire précédente et ne sont donc pas des phrases propositionnelles.

Notez aussi que le discours à propos des phrases propositionnelles n'est pas une phrase propositionnelle. La grammaire précédente, la description de l'alphabet, et cette explication parlent de propositions logiques sans en être, et font donc partie de ce qui est appelé le **métalangage**.

3.2 Les tables de vérité

La grammaire permet de définir précisément l'ensemble des phrases propositionnelles, mais pas de leur donner un sens, c'est à dire de définir quand une proposition est vraie ou fausse.

Pour cela, il faut décider lesquelles des propositions premières sont vraies et lesquelles sont fausses. Rappelez-vous que la signification des propositions premières n'a aucune importance, ce choix est donc complètement arbitraire (le choix qui décrit au mieux le monde réel n'est qu'un des choix possibles parmi tous les autres). On sait également que les propositions true et false sont, respectivement, toujours vraies et toujours fausses.

Les autres propositions sont construites à partir de propositions plus simples. Leur véracité est fonction de celle des propositions qui les composent. Prenons par exemple $p \wedge q$, où p et q sont d'autres propositions. La véracité de $p \wedge q$ est une fonction de celle de p et de q : $p \wedge q$ est vrai si et seulement si p et q sont vrais aussi (\wedge est un « et » logique). Cette relation peut être exprimée à l'aide de la table de vérité suivante :

p	q	$p \wedge q$
true	true	true
true	false	false
false	true	false
false	false	false

Voici la table de vérité des autres connecteurs logiques :

p	q	$p \vee q$
true	true	true
true	false	true
false	true	true
false	false	false

p	q	$p \Leftrightarrow q$
true	true	true
true	false	false
false	true	false
false	false	true

p	$\neg p$
true	false
false	true

p	q	$p \Rightarrow q$
true	true	true
true	false	false
false	true	true
false	false	true

Remarquez que le dernier tableau est correct. Il n'est parfois pas intuitif que la proposition $A \Rightarrow B$ (qui pourrait s'exprimer en français par « si A , alors B ») soit toujours vraie quand A est faux, mais c'est pourtant le cas : la proposition dit que quand A est vrai, B doit l'être aussi, mais elle ne donne aucune information sur le cas où A est faux.

Les parenthèses, quant à elles, servent à distinguer des propositions telles que $A \wedge (B \vee C)$ et $(A \wedge B) \vee C$, qui s'écriraient de la même manière sans parenthèses alors qu'elles n'ont pas la même table de vérité :

A	B	C	$A \wedge (B \vee C)$	$(A \wedge B) \vee C$
true	true	true	true	true
true	true	false	true	true
true	false	true	true	true
true	false	false	false	false
false	true	true	false	true
false	true	false	false	false
false	false	true	false	true
false	false	false	false	false

3.3 Les interprétations

Une autre façon plus puissante de définir si une proposition est vraie ou fausse est d'utiliser une **interprétation**. Si E_P est l'ensemble des propositions premières, alors une interprétation I définit la fonction $\text{val}_I : E_P \rightarrow \{\text{true}, \text{false}\}$ ³ qui permet de savoir si ces propositions premières sont vraies ou fausses. Par exemple, on pourrait écrire ceci :

3. La notation $f : A \rightarrow B$ signifie que f est une fonction depuis l'ensemble A vers l'ensemble B .

$$\text{val}_I(A) = \text{true}$$

$$\text{val}_I(B) = \text{false}$$

$$\text{val}_I(C) = \text{true}$$

Étant donné la fonction val_I , il est possible de définir la fonction $\text{VAL}_I : P \rightarrow \{\text{true}, \text{false}\}$, qui est une extension de val_I à P , l'ensemble de toutes les propositions. L'équivalent des tables de vérité pourrait être des expressions telles que :

$$\forall p \in E_P. \text{VAL}_I(p) = \text{val}_I(p)$$

$$\text{VAL}_I(p \wedge q) = \begin{cases} \text{true} & \text{si } \text{VAL}_I(p) \text{ et } \text{VAL}_I(q) = \text{true} \\ \text{false} & \text{sinon} \end{cases}$$

$$\text{VAL}_I(p \vee q) = \begin{cases} \text{false} & \text{si } \text{VAL}_I(p) \text{ ou } \text{VAL}_I(q) = \text{false} \\ \text{true} & \text{sinon} \end{cases}$$

$$\text{VAL}_I(\neg p) = \begin{cases} \text{false} & \text{si } \text{VAL}_I(p) = \text{true} \\ \text{true} & \text{si } \text{VAL}_I(p) = \text{false} \end{cases}$$

$$\text{VAL}_I(p \Leftrightarrow q) = \begin{cases} \text{true} & \text{si } \text{VAL}_I(p) = \text{VAL}_I(q) \\ \text{false} & \text{sinon} \end{cases}$$

$$\text{VAL}_I(p \Rightarrow q) = \begin{cases} \text{false} & \text{si } \text{VAL}_I(q) = \text{false} \text{ alors que } \text{VAL}_I(p) = \text{true} \\ \text{true} & \text{sinon} \end{cases}$$

Prenons un exemple concret, utilisons une interprétation pour étudier la phrase suivante : « S'il fait beau à midi, j'irai promener le chien ». Nous devons d'abord traduire cette phrase en l'une des propositions du formalisme que nous avons défini, en commençant par identifier les propositions premières :

1. B = « Il fait beau » ;
2. M = « Il est midi » ;
3. P = « Je vais promener le chien » ;

Identifions également les connecteurs à employer : « s'il fait beau \wedge qu'il est midi \Rightarrow j'irai promener le chien ». En combinant ces deux résultats, nous obtenons la phrase propositionnelle $(B \wedge M) \Rightarrow P$.

Interprétons désormais notre proposition à l'aide de l'interprétation suivante :

1. Il fait beau : $\text{val}_I(B) = \text{true}$;
2. Il est midi : $\text{val}_I(M) = \text{true}$;
3. Je n'irai pas promener le chien : $\text{val}_I(P) = \text{false}$.

Nous pouvons alors effectuer le développement suivant :

$$\begin{aligned} \text{VAL}_I((B \wedge M) \Rightarrow P) &= \begin{cases} \text{false si } \text{VAL}_I(P) = \text{false alors que } \text{VAL}_I(B \wedge M) = \text{true} \\ \text{true sinon} \end{cases} \\ \text{or } \text{VAL}_I(B \wedge M) &= \begin{cases} \text{true si } \text{VAL}_I(B) \text{ et } \text{VAL}_I(M) = \text{true} \\ \text{false sinon} \end{cases} \\ &= \text{true} \\ \text{donc } \text{VAL}_I((B \wedge M) \Rightarrow P) &= \begin{cases} \text{false si } \text{VAL}_I(P) = \text{false} \\ \text{true sinon} \end{cases} \\ &= \text{false} \end{aligned}$$

Nous pouvons donc conclure que, dans cette interprétation, la personne ayant fait cette affirmation a menti. Notons néanmoins que notre homme n'aurait pas mentit en partant promener le chien alors qu'il pleuvait à midi, rien n'ayant été dit sur ce qu'il ferait dans le cas où il ne ferait pas beau.

3.4 Les modèles logiques

À partir de la notion d'interprétation, nous pouvons définir ce qu'est un **modèle**. Soit $B = \{b_1, b_2, \dots, b_n\}$ un ensemble de propositions logiques. Une interprétation I est un modèle de B si et seulement si $\forall b_i \in B. \text{VAL}_I(b_i) = \text{true}$. Autrement dit, I décrit un univers qui respecte toutes les règles se trouvant dans l'ensemble B .

Donc, dans notre exemple précédent, l'interprétation choisie n'est pas un modèle de la proposition analysée, celle-ci n'étant pas validée, mais par exemple l'interprétation telle que $\text{val}_I(B) = \text{true}$, $\text{val}_I(M) = \text{true}$ et $\text{val}_I(P) = \text{true}$ est bien un modèle de la proposition utilisée comme exemple. Remarquez aussi que cela ne change rien au fait que l'interprétation choisie est le modèle d'autres propositions que celle étudiée (par exemple $B \wedge M$).

Les tautologies : Pour certaines propositions, toute interprétation est un modèle, c'est à dire que ces propositions sont toujours vraies. Par exemple, true est évidemment une tautologie, de même que $A \Rightarrow A$ ou encore $A \vee \neg A$. Le fait qu'une proposition p est une tautologie se note $\models p$.

Les contradictions : Pour d'autres propositions, il n'existe aucun modèle, c'est à dire qu'elles sont toujours fausses. Par exemple $A \wedge \neg A$ est une contradiction. Le fait qu'une proposition p est une contradiction se note $\not\models p$.

Les contingences : Toutes les autres propositions sont des contingences. Il existe des interprétations qui sont des modèles et d'autres qui n'en sont pas. Par exemple $A \wedge B$ est vrai pour l'interprétation I telle que $\text{val}_I(A) = \text{val}_I(B) = \text{true}$, mais faux dans tous les autres cas.

3.4.1 Conséquence logique

p est conséquence logique de q si et seulement si $p \Rightarrow q$ est une tautologie. En d'autres termes, si

$p \models q$ q est valide dans tous les modèles de p

alors

$\models (p \Rightarrow q)$ $p \Rightarrow q$ est une tautologie.

On peut donc écrire

$p \Rightarrow q$ p est conséquence logique de q .

Cependant, la conséquence logique (\Rightarrow) n'est pas une proposition logique (cf. syntaxe d'une proposition).

3.4.2 Équivalence logique

Par le raisonnement ci-dessus, on peut dire que p est logiquement équivalent à q si et seulement si

$p \models q$ q est valide dans tous les modèles de p
 $q \models p$ p est valide dans tous les modèles de q

et donc

$\models (p \Rightarrow q)$ $p \Rightarrow q$ est une tautologie et
 $\models (q \Rightarrow p)$ $q \Rightarrow p$ est une tautologie.

On peut donc écrire

$p \Leftrightarrow q$ p est conséquence logique de q .

L'équivalence logique n'est pas non plus une proposition logique.

Il ne faut pas non plus oublier la différence entre phrase propositionnelle (p, q, s, \dots) et propositions primaires (P, Q, S, \dots) (cf. syntaxe d'une proposition) :

$p \Rightarrow q$	n'est pas une proposition
mais	
$P \wedge Q \Rightarrow R \wedge \neg S$	en est bien une.

4 Les preuves

Une preuve est une manière de partir d'une formule et de dire si celle-ci est vraie ou fausse.

il y a 3 approches possibles :

- Table de vérité
- Preuve transformationnelle
- Preuve déductive (la plus générale)

4.1 Preuve avec table de vérité

Prouvons que $\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$ est vrai :

P	Q	$\neg P$	$\neg Q$	$(\neg P \vee \neg Q)$	$P \wedge Q$	$\neg(P \wedge Q)$
F	F	T	T	T	F	T
T	F	F	T	T	F	T
F	T	T	F	T	F	T
T	T	F	F	F	T	F

On voit bien ici, que la table de $\neg(P \wedge Q)$ est équivalente à celle de $(\neg P \vee \neg Q)$. La preuve a donc vérifié la véracité de la proposition.

Il y a néanmoins un problème avec cette preuve car s'il y a N propositions premières, on a 2^N lignes dans la table.

4.2 Preuve transformationnelle

On va utiliser des "Lois", c'est-à-dire des équivalences.

$p \Leftrightarrow p \vee p$	Idempotence
$p \vee q \Leftrightarrow q \vee p$	Commutativité
$(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$	Associativité
$\neg\neg p \Leftrightarrow p$	Double Négation
$p \Rightarrow q \Leftrightarrow \neg p \vee q$	Implication
$\neg p \wedge q \Leftrightarrow \neg p \vee \neg q$	1 ^{ere} loi de De Morgan
$p \Leftrightarrow q \Leftrightarrow (p \Rightarrow q) \wedge (q \Rightarrow p)$	Équivalence

On y ajoute 2 règles supplémentaires pour cette technique :

Transitivité de l'équivalence

Si $p \Leftrightarrow q$ et $q \Leftrightarrow r$, alors $p \Leftrightarrow r$.

Substitution

Il est autorisé de remplacer une formule par une formule équivalente à l'intérieur d'une autre formule. Autrement dit :

Soit p, q, r des formules propositionnelles.

Si $p \Leftrightarrow q$ et $r(p)$, alors $r(p) \Leftrightarrow r(q)$.

On peut remplacer p par q car elles sont équivalentes. Il faut faire attention à ces règles, car elles sont en méta-langage. Elles expliquent seulement ce qui est autorisé faire.

Exemple

On veut prouver : $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$

$$\begin{aligned} p \wedge (q \wedge r) &\Leftrightarrow p \wedge \neg\neg(q \wedge r) \\ &\Leftrightarrow p \wedge \neg(\neg q \vee \neg r) \\ &\Leftrightarrow \neg\neg(p \wedge \neg(\neg q \vee \neg r)) \\ &\Leftrightarrow \neg(\neg p \vee \neg\neg(\neg q \vee \neg r)) \\ &\Leftrightarrow \neg(\neg p \vee (\neg q \vee \neg r)) \\ &\Leftrightarrow \neg((\neg p \vee \neg q) \vee \neg r) \\ &\vdots \\ &\text{effectuer les mêmes lois dans le sens contraire} \\ &\vdots \\ &\Leftrightarrow (p \wedge q) \wedge r \end{aligned}$$

Le problème dans cette méthode de preuve c'est qu'il faut de l'idée, de la créativité et donc ce n'est pas forcément mieux que les tables de vérités, surtout si c'est compliqué de trouver l'astuce.

4.3 Preuve déductive

A partir des lois logiques (lois d'équivalence) et des règles d'inférence, On va construire des preuves.

Une preuve est un objet formel défini avec précision.

Lois de logique :

$p \Leftrightarrow p \vee p$	Idempotence de \vee
$p \vee q \Leftrightarrow q \vee p$	Commutativité de \vee
$(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$	Associativité de \vee
$\neg\neg p \Leftrightarrow p$	Double Négation
$p \Rightarrow q \Leftrightarrow \neg p \vee q$	Implication
$\neg p \wedge q \Leftrightarrow \neg p \vee \neg q$	1 ^{ere} loi de De Morgan
$\neg p \vee q \Leftrightarrow \neg p \wedge \neg q$	2 ^{eme} loi de De Morgan
$(p \wedge q) \vee r \Leftrightarrow (p \vee r) \wedge (q \vee r)$	Distributivité de \vee

Mais également l'idempotence, la commutativité, l'associativité et la distributivité de \wedge .

Règle d'inférence :

A la différence de la preuve transformationnelle, les règles d'inférences ont une direction : Elles commencent par les prémisses et se terminent par la conclusion.

Conjonction :	$\frac{\begin{array}{l} p \quad \text{prémisse} \\ q \quad \text{prémisse} \end{array}}{p \wedge q \quad \text{Conclusion}}$	Simplification :	$\frac{p \wedge q}{p}$
Addition :	$\frac{p}{p \vee q}$	Contradiction :	$\frac{p}{\frac{\neg p}{q}}$
Double Négation :	$\frac{\neg\neg p}{p}$	Transitivité de l'équivalence :	$\frac{p \Leftrightarrow q}{\frac{q \Leftrightarrow r}{p \Leftrightarrow r}}$
Modus Ponens :	$\frac{p \Rightarrow q}{p}$ q	Modus Tollens :	$\frac{p \Rightarrow q}{\frac{\neg p}{\neg q}}$
Loi d'équivalence :	$\frac{p \Leftrightarrow q}{q \Leftrightarrow p}$		

On y ajoute 2 règles spéciales :
— Le théorème de déduction,
— La preuve par contradiction.

Théorème de déduction

Pour prouver $s \Rightarrow t$, On suppose s vrai. On déduit t et donc on sait que l'hypothèse $s \Rightarrow t$ est vraie et on l'évacue.

On note ce théorème $s \vdash t$ (t peut être prouvé à partir de s , ce qui signifie qu'on peut construire une preuve (objet mathématique) qui est vraie si on applique bien les règles ci-dessus).

Remarque : $p \models t \neq p \vdash t$

- $p \models t$ est une notion de vérité, et donc de sémantique,
- $p \vdash t$ est une notion syntaxique, on peut prouver qu'une preuve est vraie.

Notion de prouvabilité : on peut créer une preuve, et donc on peut créer une séquence de prémisses et de conclusions.

$$\frac{p, \dots, r, s \vdash t}{p, \dots, r \vdash s \Rightarrow t}$$

Ceci permet notamment de formaliser des théorèmes.

Preuve par contradiction

C'est une preuve indirecte :

Implicitement, on suppose que p jusqu'à q n'a pas de problème, c'est-à-dire qu'on ne peut pas prouver une contradiction pour ces formules propositionnelles.

$$\frac{\begin{array}{l} p, \dots, q, r \vdash s \\ p, \dots, q, r \vdash \neg s \end{array}}{p, \dots, q \vdash \neg r}$$

Ceci signifie qu'il y a une erreur dans les prémisses, et on suppose ici que c'est la formule propositionnelle r qui est fautive. On justifie qu'il n'y a aucune contradiction dans p, \dots, q car on part du principe qu'il existe un modèle de p, \dots, q .

Tout ceci est une formalisation de choses qu'on connaît déjà, mais ça nous donne une notion précise du raisonnement à avoir.

4.4 Exemple de preuve propositionnelle

Voici les propositions premières :

A : tu manges bien

B : ton système digestif est en bonne santé
 C : tu pratiques une activité physique régulière
 D : tu es en bonne forme physique
 E : tu vis longtemps

On peut maintenant faire une théorie et on espère qu'elle aura un modèle.
 $A \Rightarrow B, C \Rightarrow D, B \vee D \Rightarrow E, \neg E$
 On aimerait prouver que $\neg A \wedge \neg C$ est vrai.

Preuve :

1. $A \Rightarrow B$	prémisse
2. $C \Rightarrow D$	prémisse
3. $B \vee D \Rightarrow E$	prémisse
4. $\neg E$	prémisse
5. A	hypothèse
6. B	modus ponens (1)
7. $B \vee D$	addition (6)
8. E	modus ponens (7)
9. $\neg A$	preuve indirecte
10. C	hypothèse
11. D	modus ponens (2)
12. $D \vee B$	addition (11)
13. $B \vee D$	commutativité (12)
14. E	modus ponens (9)
15. $\neg C$	preuve indirecte
16. $\neg A \wedge \neg C$	conjonction (9,15)

Grâce à la déduction on a donc pu prouver que tu ne manges pas bien et que tu ne pratiques pas d'activité physique régulière. On aimerait maintenant pouvoir automatiser les preuves quand elles existent. Mais il faut savoir s'il peut tout résoudre ou pas. On va donc construire un algorithme nous permettant de résoudre automatiquement les preuves en logique propositionnelle.

5 Deux règles plus sophistiquées

5.1 Théorème de déduction

- Pour prouver $s \Rightarrow t$
- On suppose s vrai
- On déduit t
- Ensuite on évacue l'hypothèse

Notation : $p \vdash t$ (on peut prouver t à partir de p)

5.1.1 Prémisse :

$$\frac{p \dots r, s \vdash t}{p \dots r, s \vdash (s \Rightarrow t)} \quad (1)$$

5.1.2 Conclusion :

Déduire une implication

5.2 Preuve par contradiction (ou preuve indirecte)

On prend une hypothèse, et on peut prouver qu'elle est vraie ou fausse, d'où l'hypothèse n'est pas bonne.

5.2.1 Prémisse :

on suppose que $p \dots q$ n'a pas de problème

$$\frac{\begin{array}{l} p \dots q, r, s \vdash s \\ p \dots q, r, s \vdash \neg s \end{array}}{p \dots q \vdash \neg r} \quad (2)$$

5.2.2 Conclusion :

si $p \dots q$ n'a pas de problème, on se focalise alors sur r

6 Exemples de Preuves

Une preuve est une séquence de pas où chaque pas est une application de règles d'inférences et de lois logiques. Il faut justifier à chaque étape le nom de

la règle / loi, et indenter les éléments de la preuve en preuve conditionnelle indirecte.

6.1 Prémisses :

$$p \wedge q \vee r$$

6.2 Conclusion :

$$\neg p \Rightarrow r$$

6.3 Exemple sans preuve conditionnelle

1. $(p \wedge q) \vee r$ *Prémisse*
2. $r \vee (p \wedge q) \vee r$ *Commutativité en 1*
3. $(r \vee p) \wedge (r \vee q)$ *Associativité en 2*
4. $(r \vee p)$ *Simplification en 3*
5. $(p \vee r)$ *Commutativité en 4*
6. $\neg \neg p \vee r$ *Loi de la négation en 5*
7. $\neg p \Rightarrow r$ *Implication en 6*

6.4 Exemple avec preuve conditionnelle

1. $(p \wedge q) \vee r$ *Prémisse*
2. $\neg \neg(p \wedge q) \vee r$ *Double négation en 1*
3. $\neg (\neg p \wedge \neg q)$ *Loi De Morgan en 2*
4. $\neg p \vee \neg q \Rightarrow r$ *Implication en 3*
 - (a) $\neg p$ *Hypothèse*
 - (b) $\neg p \vee \neg p$ *Addition sur 5*
 - (c) r *Modus Ponens sur 4 et 6*
5. $\neg p \Rightarrow r$ *Evacuation de l'hypothèse*

6.5 Exemple de Preuve par contradiction

1. $(p \wedge q) \vee r$ *Prémisse*
2. $(p \vee r) \wedge (q \vee r)$ *Distributivité sur 1*
3. $(p \vee r)$ *Simplification en 2*
 - (a) $\neg (\neg p \Rightarrow r)$ *Hypothèse*

- (b) $\neg (\neg \neg p \vee r)$ *Implication en 4*
- (c) $\neg (p \vee r)$ *Négation en 5*
- 4. $\neg \neg (\neg p \Rightarrow r)$ *Preuve par contradiction*
- 5. $\neg p \Rightarrow r$ *Négation en 7*

7 Quelques concepts

7.1 Principe de dualité

7.1.1 Dans les formules sans \rightarrow :

$$1 \leftrightarrow T \qquad \qquad \qquad true \leftrightarrow false \qquad (3)$$

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

7.1.2 Formule quelconque :

$$1 \leftrightarrow T \qquad \qquad \qquad true \leftrightarrow false \qquad (4)$$

Justification en raisonnant sur les modèles :

$$\begin{aligned} p1, \dots, pn \models ssi \models (p1, \dots, pn \wedge \neg q) &\leftrightarrow false \\ ssi \models (\neg p1 \vee \dots \vee pn) &\leftrightarrow true \end{aligned}$$

7.2 Forme Normale

- Conjonctive : $(p \vee q) \wedge (q \vee a) \wedge (s \vee r)$
- Disjonctive

7.2.1 terminologie

- Littéral : $P \vee \neg P \cong L$
- Clause : $\vee Li = (L1 \vee L2 \vee L3 \dots \vee Li)$

7.2.2 Forme normale conjonctive FNC

7.3 Algorithme de normalisation

1. Eliminer les \rightarrow et \leftrightarrow

2. Déplacer les négations vers l'intérieur (dans les propositions premières) De Morgan
3. Déplacer les disjonctions (\vee) vers l'intérieur
4. Simplifier ($P \vee \neg P$)

7.3.1 Exemple

$$\begin{aligned}
& (p \rightarrow (Q \rightarrow R)) \rightarrow ((P \wedge S) \rightarrow R) \\
& \neg(\dots) \vee (\dots) \\
& \neg(\neg P \vee (\neg Q \vee R)) \vee (\neg(P \wedge S) \vee R) \\
& (\neg\neg P \wedge \neg(\neg Q \vee R)) \vee ((\neg P \vee \neg S) \vee R) \\
& (P \wedge (Q \wedge \neg R)) \vee (\neg P \vee \neg S \vee R) \\
& (P \vee \neg P \vee \neg S \vee R) \wedge (Q \vee \neg P \vee \neg S \vee R) \wedge (\neg R \vee \neg P \vee \neg S \vee R) \\
& (Q \vee \neg P \vee \neg S \vee R)
\end{aligned}$$

8 Algorithme de preuve

Nous allons maintenant introduire un algorithme qui permet de trouver une preuve en logique des prédicats. Cet algorithme est une automatisation de la *démonstration par l'absurde* qui est basé sur une seule règle d'inférence, la *résolution*.

8.1 La résolution

On veut quelque chose de simple, sans toutes les règles que nous avons vues auparavant, mais le plus puissant possible. Nous n'utiliserons qu'une seule règle : la **résolution**. On peut faire des résolutions de preuves propositionnelles rien qu'en ayant cette règle. Cette règle utilise la forme normale conjonctive. On utilise les preuves indirectes (preuves par l'absurde), car c'est le plus simple.

Commençons par un exemple de résolution.

8.1.1 Exemple de résolution

Prenons comme propositions premières :

P_1 : il neige
 P_2 : la route est dangereuse
 P_3 : on prend des risques
 P_4 : on va vite
 P_5 : on va lentement
 P_6 : on prend le train

$$\left. \begin{array}{l} 1. P_1 \Rightarrow P_2 \\ 2. P_2 \Rightarrow \neg P_3 \\ 3. P_4 \Rightarrow P_3 \vee P_6 \\ 4. P_4 \vee P_5 \\ 5. P_1 \end{array} \right\} B : \text{notre théorie}$$

On va utiliser B + modus ponens + résolution..

1. $P_1 \Rightarrow P_2$ 2. $P_2 \Rightarrow \neg P_3$ 3. $P_4 \Rightarrow P_3 \vee P_6$ 4. $P_4 \vee P_5$ 5. P_1 3'. $\neg P_3 \Rightarrow \neg P_4 \vee P_6$ 6. P_2 7. $\neg P_3$ 8. $\neg P_4 \vee P_6$ 9. $P_5 \vee P_6$	1-5 : B : notre théorie que l'on utilise comme prémisse réécriture de 3 modus ponens (1,5) modus ponens (6,2) modus ponens (7,3') résolution (4,8)
---	---

La ligne 3 n'étant pas symétrique, nous pouvons la transformer pour obtenir une proposition symétrique et donc choisir le membre qui est à gauche de l'implication. Pour rappel, $P_4 \Rightarrow P_3 \vee P_6$ peut être réécrit : $\neg P_4 \vee P_3 \vee P_6$ (loi de l'implication), qui est logiquement équivalent à $\neg P_3 \Rightarrow \neg P_4 \vee P_6$ (loi de l'implication). C'est de cette manière que nous avons obtenu la ligne 3'.

On peut fusionner les lignes 4 et 8 grâce à la résolution. La **résolution** est une règle qui prend deux disjonctions avec une proposition première et sa négation, et qui les fusionne en retirant cette proposition première. On peut prouver que cela fonctionne de plusieurs manières. Par exemple : si P_4 est vrai, P_6 doit être vrai. Si P_4 est faux, P_5 doit être vrai. Donc on sait que P_5 ou P_6 doit être vrai car on sait que dans tous les cas de figure, c'est soit l'un soit l'autre qui doit être vrai.

8.1.2 Principe de résolution

$$\begin{array}{l}
 p_1 \vee q \\
 p_2 \vee \neg q \\
 \hline
 p_1 \vee p_2
 \end{array}$$

Cette règle représente la base de l'algorithme de résolution. On peut la vérifier en utilisant le métalangage. De plus, cette règle est aussi utilisée dans la logique des prédicats.

8.1.3 La résolution préserve les modèles

Tout ce qui est modèle des deux premières disjonctions sera aussi modèle de la résultante.

$$p : \bigwedge_{1 \leq i \leq n} C_i \qquad C_i : \text{disjonction} : \bigvee_{1 \leq j \leq n} L_j \quad \{C_1, \dots, C_n\}$$

$C_1, C_2 =$ deux disjonctions

On doit prouver : $\{C_1, \dots, C_n\} \models r$ avec $r = C_1 - \{P\} \vee C_2 - \{\neg P\}$. r est une nouvelle disjonction à partir de deux autres disjonctions. On doit prouver que r est toujours vrai.

On considère que P est dans C_1 et que $\neg P$ est dans C_2 .

Pour prouver cela, on utilise la sémantique. On fait une preuve en métalangage, ce n'est pas formalisé.

$$\text{Val}_I(P) = \begin{cases} T \\ F \end{cases}$$

Dans les deux cas de figure, on doit démontrer que quand on a un modèle, une interprétation qui rend vrai p , le r sera vrai aussi. Si P est vrai alors $\neg P$ est faux, donc C_2 sera vrai et donc r sera vrai. Quand P est faux, le C_1 doit être vrai, donc r est vrai. r est donc vrai dans les deux cas.

8.2 Algorithme

C_i : clause $\bigvee_i L_i$

L_i : P ou $\neg P$

$\{C_i, \dots, C_n\} \models C$

s.s.i

$\{C_i, \dots, C_n, \neg C\} \models \text{false}$

C_i : axiomes

C : candidat théorème

Ce que nous voulons prouver : $\{C_i, \dots, C_n\} \vdash C$

Il existe une preuve avec les règles d'inférence $\{C_i, \dots, C_n\}$ tel qu'on obtient C .

$S = \{C_i, \dots, C_n, \neg C\}$

But : Déterminer si S est inconsistent. On veut faire des déductions jusqu'à arriver sur false.

8.2.1 Pseudocode

```

while false  $\notin S$  et  $\exists ?$  clauses résolubles non-résolues
do choisir  $C_1, C_2 \in S$  tel que  $\exists P \in C_1, \neg P \in C_2$ 
   calculer  $r := C_1 - \{P\} \vee C_2 - \{\neg P\}$ 
   calcul  $S := S \cup \{r\}$ 

```

end

if false \in S then *C est prouvé* else *C n'est pas prouvé*

La subtilité dans cet algorithme est de choisir correctement les clauses C_1 et C_2 car l'efficacité de l'algorithme en dépend.

8.3 Exemples

8.3.1 Exemple 1

$C_1 : P \vee Q$
 $C_2 : P \vee R$
 $C_3 : \neg Q \vee \neg R$
 $C : P$ $\{C_1, C_2, C_3, \neg C\}$

Quelques pas de résolution :

$C_1 + \neg C \rightarrow Q$ (C_5)
 $C_2 + \neg C \rightarrow R$ (C_6)
 $C_3 + C_5 \rightarrow \neg R$ (C_7)
 $C_6 + C_7 \rightarrow$ false (\in S donc C est prouvé)

8.3.2 Exemple 2

$p_1 : \text{Mal de tête} \wedge \text{Fièvre} \Rightarrow \text{Grippe}$
 $p_2 : \text{Gorge blanche} \wedge \text{Fièvre} \Rightarrow \text{Angine}$
 $p_3 : \text{Mal de tête}$
 $p_4 : \text{Fièvre}$

Algorithme

- Normalisation en forme normale
- Pseudocode avec résolution

Question : Grippe ?

8.4 Conclusion

Nous pouvons tirer des conclusions sur la logique des propositions et sur notre algorithme.

Pour toute théorie $B = \{c_1, \dots, c_n\}$ et p ,

- si $B \vdash p$ alors $B \models p$ (Adéquat - *Soundness*);
- si $B \models p$ alors $B \vdash p$ (Complet - *Completeness*);
- $\forall B, p$, l'exécution de l'algorithme se termine après un nombre fini d'étapes. (Décidable - *Decidable*)

Cet algorithme est donc très puissant, même s'il n'est peut être pas très efficace. Mais au moins on est sûr qu'il s'arrêtera toujours à un moment.

Malheureusement, la logique des propositions n'est pas très expressive. On va donc essayer de faire la même chose mais avec une logique plus puissante : la logique des prédicats. Mais nous n'arriverons pas à avoir un algorithme aussi puissant pour la logique des prédicats, car c'est une logique trop forte.

9 La logique des prédicats

9.1 Introduction

Nous allons maintenant étudier une logique beaucoup plus expressive que la logique propositionnelle, la logique des prédicats, qui est aussi appelée la logique de premier ordre.⁴ Voici un premier tableau qui montre les différences entre la logique propositionnelle vue jusqu'à présent et la logique des prédicats que nous allons étudier.

Logique Propositionnelle	Logique des prédicats
Propositions premières P, Q, R \hookrightarrow Pas de Relations	Prédicats $P(x,y)$ Quantificateurs : $\exists x, \forall y$ \hookrightarrow Relation

On note dans la logique des prédicats $P(x,y)$ avec x,y qui sont des variables, les arguments du prédicat P . Dans la logique propositionnelle, chaque proposition est "toute seule" alors que dans les prédicats on peut lier plusieurs prédicats ensemble.

Exemple	Logique propositionnelle	Logique des prédicats
Socrate est un philosophe	P	Phil(Socrate)
Platon est un philosophe	Q	Phil(Platon)

En logique propositionnelle il n'y a aucunes relations entre P et Q, alors qu'en logique des prédicats on peut lier Socrate et Platon avec le prédicat Philosophe qui prend en argument Socrate ou Platon. Phil(Socrate) est donc vrai. On peut donc dire grâce aux prédicats que Socrate et Platon sont "la même chose", des philosophes.

Un autre exemple de prédicat :

$$\forall \alpha \text{ Phil}(\alpha) \Rightarrow \text{Savant}(\alpha)$$

\hookrightarrow ... *c'est un résumé d'un très grand nombre de faits. ça marche pour tous arguments, ça peut être un ensemble infini !*

Comme Socrate est un philosophe, on peut déduire que Socrate est un savant aussi !

Pour dire la même chose en logique de proposition cela est beaucoup plus compliqué :

4. Il existe des logiques d'ordres supérieures mais ils ne feront pas l'objet de ce cours.

"Socrate est un savant" Proposition "R"
 "Platon est un savant" Proposition "S"

On va donc noter en logique propositionnelle

$$(P \Rightarrow R) \cup (Q \Rightarrow S) \cup \dots (\text{potentiellement infini})$$

On doit tout énumérer et il n'y a aucune relations entre les différentes propositions. Si il y a un nombre infini ça ne marche pas. Il y a donc de grandes limitations dans la logique propositionnelle.

Néanmoins parfois la logique propositionnelle est utile. Il existe des outils informatiques qui utilisent la logique propositionnelle et donc parfois ça suffit. Typiquement on peut prendre l'exemple de "SAT solver" qui donne equation booleenes très compliquées et va trouver des valeurs propositions primitives qui rendent vrais cette porposition. C'est donc assez utilisé! La logique propositionnelle est utile mais si on veut faire du raisonnement sur plus que vrai et faux, avec des relations plus que vrai faux et avec des relations entre des propositions alors la logique propositionnelle ne marche pas. Si on veut faire un logiciel qui montre une certaine intelligence alors la logique des propositions ne marche pas et il faut utiliser la logique des prédicats.

Autre exemple :

Exemple	Logique propositionnelle	Logique des prédicats
Tout adulte peut voter	P	$\forall x \text{ adulte}(x) \Rightarrow \text{voter}(x)$
John est un adulte	Q	$\text{adulte}(\text{john})$
—	—	—
John peut voter	?R?	$\text{voter}(\text{john})$

Ce genre de raisonnement est très difficile à faire en logique propositionnelle alors qu'en logique des prédicats c'est beaucoup plus simple! Le john en ligne 3 et en ligne 4 sont les même john! Ceci montre donc bien qu'il nous faut la logique des prédicats pour faire des relations de ce type.

9.2 Quantificateurs

Les expressions "pour tout x " ($\forall x$) et "il existe x tel que" ($\exists x$) sont appelés des quantificateurs en logique des prédicats. Les quantificateurs permettent d'instancier les variables dans une formule. La notion de portée d'un quantificateur est un concept très important auquel il faut faire très attention, car il peut changer complètement le sens d'une formulation.

$$\forall x (\text{enfants}(x) \wedge \text{intelligents}(x) \Rightarrow \exists y \text{ aime}(x,y))$$

$$\forall x (\text{enfants}(x) \wedge \text{intelligents}(x)) \Rightarrow \exists y \text{ aime}(x,y)$$

Ces deux formules peuvent paraître équivalente, mais en réalité elles ont un sens tout à fait différent. En effet, dans le deuxième cas on remarque que le quantificateur $\forall x$ ne porte pas sur la dernière variable x qui est en argument du prédicat $\text{aime}(x,y)$.

Il faut donc faire bien attention à quel quantificateur une variable s'identifie lorsqu'on manipule des formules.

- $\forall x P(x) \wedge \exists x Q(x)$: contient deux variables différentes
- $\forall x \exists x P(x) \wedge Q(x)$: est une forme incorrecte, conflit des noms de variables

Pour résoudre ces conflits on fait appel à une nouvelle opération, le renommage. Cette opération permet de changer le nom des variables tout en conservant le sens de la formule. Ainsi on obtient :

$$\forall x \exists z P(x) \wedge Q(z) \text{ renommage } (2)$$

Le concept de variables, de leur portées ainsi que d'opérateurs en logique des prédicats fait fortement penser au langage de programmation

Une comparaison entre un code et une formule est tout à fait envisageable. Prenons un code tout à fait banal comprenant des variables différentes avec des portées différentes qui ont le même identificateur ainsi qu'une formule correspondante.

```

1.  begin {
2.      var x,y: int;
3.      x := 4;
4.      y := 2;
5.
6.      begin {
7.          var x: int;
8.          x := 5;
9.          x := x*y;
10.     end }
11.     x := x*y;
12. end }
```

$$\forall x \forall y p(x) \wedge (\exists x q(x,y) \vee r(x,y))$$

En analysant morceau par morceau de la formule :

- " $\forall x \forall y p(x) \wedge$ " correspond aux points {2, 3, 4} du code
- " $\exists x q(x,y) \vee$ " correspond aux points {7, 8, 9}
- " $r(x,y)$ " correspond au point {11}

Cet exemple illustre parfaitement la ressemblance et le lien entre le monde de la programmation et celui de la logique des prédicats

9.3 Syntaxe

Symboles logiques	quantificateurs connecteurs logiques parenthèses variables true, false	$\forall \exists$ $\wedge \vee \neg \Rightarrow \Leftrightarrow$ () x, y, z
Symboles non-logiques	symboles de prédicats symboles de fonction	$P \varphi R + arguments \geq 0$ $+ arguments \geq 0$

9.4 Grammaire

9.4.1 Règles de formation

$$\begin{aligned}
 < formule > ::= < formule atomique > \\
 & \quad | \neg < formule > \\
 & \quad | < formule > < connecteur > < formule > \\
 & \quad | \forall < var > . < formule > \\
 & \quad | \exists < var > . < formule > \\
 < formule atomique > ::= true, false \\
 & \quad | < predicat > (< terme > *) \\
 < terme > ::= < constante > \\
 & \quad | < var > \\
 & \quad | < fonction > (< terme > *) \\
 < connecteur binaire > ::= \wedge | \vee | \Rightarrow | \Leftrightarrow
 \end{aligned}$$

9.5 Sémantique

Dans la logique des prédicats, nous gardons les notions de modèle et d'interprétation qui sont déjà définis dans la logique propositionnelle. Même si la logique des prédicats est beaucoup plus puissante, nous gardons une sémantique assez similaire à la logique propositionnelle.

L'idée dans l'interprétation c'est de dire si la formule est vraie ou fausse. Tout comme pour la logique propositionnelle on va faire la même chose pour les prédicats avec les relations.

Illustrons par un exemple :

$$p : P(b, f(b)) \Rightarrow \exists y P(a, y)$$

On suppose que le a et le b sont des constantes et que le f est une fonction. Il faut donner un sens à cette formule et pour cela il faut donner un sens à :

- $P : val_I(P) = \geq$ (*Considéré comme un vrai prédicat*)
- $a : val_I(a) = 2$
- $b : val_I(b) = \pi$
- $f : val_I(f) = f_i \quad f_i = \mathbb{R} \rightarrow \mathbb{R} : d \rightarrow \frac{d}{2}$

Avec ces éléments, on peut donc trouver l'interprétation :

$$\text{Si } \pi \geq \frac{\pi}{2}, \text{ alors } \exists d \in \mathbb{R} \text{ tel que } \sqrt{2} \geq d$$

Cette phrase n'est pas très utile mais avec cette interprétation, la phrase logique donne ce sens. Une autre interprétation donnerais un sens totalement différent à la phrase logique. Voici une autre interprétation totalement différente :

- $a : val_I(a) = \text{"Barack Obama"}$
- $b : val_I(b) = \text{"Vladimir Putin"}$
- $f : val_I(f) = f_i \quad f_i \rightarrow \text{père}(d)$
- $P : val_I(P) = P_I \quad d_1 \text{ est enfant de } d_2$

Avec ce nouveau sens, on trouve l'interprétation suivante :

Si Vladimir Putin est l'enfant du père de Vladimir Putin alors \exists une personne telle que Barack Obama est l'enfant de cette personne.

La seconde interprétation est très différente de la première malgré le fait que ce soit la même formule à l'origine ! La connexion entre une formule et son sens permet de garder une certaine souplesse dans le sens où l'on peut choisir ça. C'est un peu comme dans la logique propositionnelle mais

avec encore plus de souplesse. Si l'on fait ce raisonnement-ci, ce dernier sera toujours vrai pour toutes les interprétations qui sont vraies.

On peut se demander si ces deux interprétations sont des modèles de la formule ?

La première interprétation est un modèle de la formule car le sens de la formule est vrai dans l'interprétation. En effet, $\pi \geq \frac{\pi}{2}$ et $\exists d \in \mathbb{R}$ tel que $\sqrt{2} \geq d$. On voit donc que le modèle est vrai.

La deuxième interprétation est aussi un modèle de la formule car l'interprétation trouvée est vrai aussi. Cela peut paraître bizarre mais c'est correct.

10 Logique des prédicats

10.1 Détails techniques

La première question que l'on se pose est : Comment faire des preuves en logique des prédicats ? Il faut noter que les quantificateurs \forall et \exists rendent le raisonnement plus subtile. Il faudra des règles pour pouvoir raisonner sur ces quantificateurs.

10.2 Sémantique

La sémantique en logique des prédicats est très proche de celle en logique propositionnelle. Cependant, l'interprétation, I , est plus précise qu'en logique propositionnelle mais est également plus compliquée à utiliser à cause des variables et des symboles de fonction. cette interprétation peut être décrite comme une paire : $I = \text{pair}(D_I, \text{val}_I)$ avec D_I le domaine de discours I et la fonction val qui est l'interprétation de tous les symboles. $D_I \neq \emptyset \forall s \in S$ avec s soit un symbole de prédicat soit une fonction. $\text{val}_I(S) = P_I$ une fonction $P_I : D_I^n \rightarrow (\text{True}, \text{False})$ avec S symbole d'une fonction. Il existe aussi une vraie fonction $f_I : D_I^n \rightarrow D_I$ avec n le nombre d'arguments tel que $\text{val}_I(S) = f_I$. Cela implique que dans le domaine de discours chaque fonction correspond à une vraie fonction et chaque prédicat correspond à un vrai prédicat.

Si on rajoute une variable, x par exemple, l'expression devient : $(\text{var}, x) \rightarrow \text{val}_I(x) = x_I \in D_I$. L'interprétation d'une variable, x_i est un élément (n'importe lequel en fonction de l'interprétation) de D_I . La fonction VAL_I est la même fonction mais pour les formules et pas uniquement pour les symboles comme avant. Néanmoins on ne doit pas redéfinir VAL_I car elle existe à partir du moment où val_I et D_I existent. Définition : $VAL_I : \text{TERM} \cup \text{PRED} \rightarrow D_I \cup (\text{True}, \text{False})$ avec TERM l'ensemble des termes et PRED l'ensemble des prédicats (toutes les formules en logique des prédicats). Les termes peuvent être définis comme $t \rightarrow VAL_I(t)$ et les prédicats comme $P \rightarrow VAL_I(P)$ avec P une formule. Il y a une relation entre

val_I et VAL_I : $\text{val}_I((P(T_1, \dots, T_n))) = P_I(VAL_I(t_1), \dots, VAL_I(t_n))$ avec $P_I = VAL_I(P)$ La première partie de l'égalité est un prédicat avec des arguments tandis que la seconde est formée de plusieurs prédicats avec un argument.

Définissons d'autres formules. $VAL_I(P \wedge Q)$ est true si $VAL_I(P) = \text{True}$ et $VAL_I(Q) = \text{True}$. false sinon. Donc il faut $VAL_I(\forall x.P) = \text{True}$ avec P une formule pouvant dépendre de x (il faut remplacer toutes les occur-

rences de x dans P et la formule doit restée vraie. Si pour chaque $d \in D_I$ $I' = I \cup (x \leftarrow d)$ et $val_I(x) = d$ alors l'interprétation de x doit être : $VAL'_I(P) = True$. Pour $VAL_I(\exists x.P) = True$ le raisonnement est identique en remplaçant \forall par \exists .

10.3 Différence avec la logique des propositions

Les trois différences entre les deux logiques sont : les variables, les quantificateurs et les symboles de fonction (moins important). Imaginons un modèle $B : \{ P_1, \dots, P_n \}$ Si nous utilisons une interprétation I pour B cela donne : $\forall P_I \in BVAL_I(P_I) = True$ qui est très générale car P_I peut avoir des variables, des quantificateurs, etc...

10.4 Preuves avec règles

Il est possible de faire des preuves avec des règles de preuve. Une preuve est une simple manipulation de symboles qui sont des règles. Mais il faut justifier ces règles pour obtenir des résultats vrais. Elles sont justifiées en raisonnant sur les interprétations pour vérifier si elles sont correctes ou non. Tout cela est la sémantique. C'est la base pour pouvoir faire des preuves.

Beaucoup de choses restent les mêmes que dans la logique des propositions. La preuve est toujours un objet mathématique, une séquence avec des formules, des justifications, une application des règles. Elle commence avec des prémisses et finit par une conclusion.

Il est possible de faire des preuves manuelles, mais aussi des preuves automatisées. C'est une généralisation de l'approche de la logique des propositions.

Il y a toujours :

- Une règle de résolution pour les preuves automatisées, mais celle-ci est plus générale. Elle va utiliser un concept appelé "unification". Ce nouveau concept est introduit à cause des variables. En effet, celles-ci peuvent être différentes, il faut donc trouver un nouveau moyen de les fusionner.
- Une forme normale, mais elle est plus compliquée à cause des quantificateurs et des symboles de fonctions. Mais il est encore possible de l'obtenir.
- un algorithme avec ses propriétés. Mais il est moins fort/complet que l'algorithme développé pour la logique des propositions. Il ne sera plus

décidable mais seulement semi-décidable. C'est-à-dire que parfois il tournera en boucle. Cela est dû aux variables et aux quantificateurs. La logique des prédicats est beaucoup plus riche que la logique des propositions, mais en contrepartie l'algorithme arrive à prouver moins de choses. Cependant, l'algorithme sera toujours adéquat mais pas forcément complet. Il ne sera pas toujours possible de trouver une preuve, même quand elle existe parce qu'elle sera trop compliquée.

Qu'est-ce qu'il est possible de faire avec ce genre d'algorithme qui est moins fort ?

Il y a deux possibilités :

1) Faire un assistant de preuve

C'est un outil qui aide les gens à faire des preuves formelles. Deux exemples d'assistants de preuves sont Coq et Isabelle. C'est un outil très sophistiqué mais qui a permis de prouver des choses de manière totalement formelle, alors qu'avant des preuves prenaient des dizaines voire des centaines de pages de preuves mathématiques. Mais cet assistant ne fait pas tout parce que l'algorithme est moins bon. Cependant il aide beaucoup. C'est à l'être humain de lui donner des coups de pouce sous forme de lemmes, hypothèses, chemins, stratégies ... Ensuite l'algorithme s'occupe de la manipulation des symboles.

Un exemple très célèbre est le théorème de la coloration d'une carte. La question est : Est-il toujours possible de colorier chaque pays avec une couleur, de façon à ce que deux pays limitrophes n'aient pas la même couleur et en utilisant un certain nombre de couleurs différentes ? Ce n'est pas évident à prouver et ça a demandé beaucoup de travail aux mathématiciens. Mais récemment, Georges Gonthier (un informaticien) a réussi à formuler ce problème avec l'assistant de preuves. Ce fût un tour de force. Désormais, il existe une preuve complètement formalisée, sans erreur pour ce théorème.

2) L'utiliser dans les langages de programmation

L'algorithme peut être considéré comme le moteur d'un programme. C'est ce qu'on appelle maintenant la programmation logique. Elle consiste à utiliser la logique dans un programme. Le langage le plus célèbre qui a suivi cette approche est Prolog. Ce fût un énorme succès car les gens ne croyaient pas que c'était possible de faire un programme en logique qui pouvait tourner.

Cela a donné naissance à la programmation par contraintes (une contrainte est une relation logique). Cette discipline est très utile pour les optimisations, par exemple dans le cas du "voyageur de commerce".

La logique des prédicats n'est donc pas quelque chose de seulement théorique, destiné uniquement aux mathématiciens. Les gens ont vraiment essayé avec succès d'utiliser la logique dans l'exécution des programmes.

11 Preuves en logique des prédicats

On va généraliser l'approche de la logique propositionnelle car le langage des prédicats est beaucoup plus riche. Il ajoute entre autres :

- Des variables
- Des constantes
- Des fonctions (détaillé plus tard)
- Des prédicats
- Des quantificateurs

Les preuves en logique des prédicats ressemblent très fort aux preuves en logique propositionnelles. Il y a encore des prémisses, des formules avec leurs justificatifs et une conclusion. On peut aussi utiliser des preuves indirectes et des preuves conditionnelles. Cela reste un objet formel.

1.	<div>...</div>	<i>Premisses</i>
2.	<div>Formule, règle</div>	<i>Justification</i>
...
<i>n.</i>	<div>Conclusion</div>	<i>Justification</i>

11.1 Exemple

La méthode pour passer de $\forall x \cdot P(x) \wedge Q(x)$ (prémisse) à $\forall x \cdot P(x) \wedge (\forall x \cdot Q(x))$ (conclusion) est la suivante :

1. Enlever les quantificateurs pour avoir des variables libres
2. Raisonner sur l'intérieur
3. Remettre les quantificateurs

Les étapes difficiles à réaliser correctement sont les étapes 1 et 3. Voici la preuve en "français" :

En retirant les quantificateurs des prémisses, cela donne : "Alors, $P(x) \wedge Q(x)$ est vrai pour tout x donc $P(x)$ est vrai pour tout x ". De là on peut remettre les quantificateurs pour obtenir $\forall x \cdot P(x)$. De façon similaire, on obtient $\forall x \cdot Q(x)$. Et on conclut en remettant les quantificateurs : $\forall x \cdot P(x) \wedge \forall x \cdot Q(x)$, en utilisant la conjonction.

En preuve formelle cela donne :

1.	$\forall x \cdot P(x) \wedge Q(x)$	Prémisses
2.	$P(x) \wedge Q(x)$	Elimination de \forall
3.	$P(x)$	Simplification
4.	$\forall x \cdot P(x)$	Introduction de \forall
5.	$Q(x)$	Simplification \forall
6.	$\forall x \cdot Q(x)$	Introduction de \forall
7.	$\forall x \cdot P(x) \wedge \forall x \cdot Q(x)$	Conjonction

On a donc utilisé 4 règles en plus par rapport aux preuves formelles en logique propositionnelle (les règles de la logique propositionnelle restent valables en logique des prédicats) :

- Elimination de \forall
- Introduction de \forall
- Elimination de \exists
- Introduction de \exists

Certaines de ces règles sont simples d'utilisation, d'autres sont plus difficiles. Il est également possible d'utiliser d'autres règles (certaines plus générales que d'autres⁵).

Note :

Il est possible d'utiliser les quantificateurs dans les formules mathématiques. Typiquement, on ne les note pas, car ils sont présents de manière implicite. Par exemple :

- $\forall x \cdot \sin(2x) = 2 \cdot \sin(x) \cdot \cos(x)$
- $\forall x \cdot x + x = 2x$
- $\exists x \cdot \sin(x) + \cos(x) = 0,5$
- $\exists x \cdot x + 5 = 9$

On peut remarquer que pour les deux premiers cas, x est une véritable variable, on peut donc ajouter un quantificateur universel \forall . Pour les deux cas suivants, on remarque que x est une inconnue, car il y a une équation à résoudre et une solution à trouver, on peut donc

5. Voir "Inference logic" ou "Predicate logic"

ajouter un quantificateur existentiel \exists .

Dans certains cas, les quantificateurs existentiels et universels sont utilisés au sein de la même formule mathématique.

$$\forall a \cdot \forall b \cdot \forall c \cdot \exists x \cdot ax^2 + bx + c = 0$$

Dans l'exemple ci-dessus, nous avons 4 variables : a, b, c, x . Les 3 premières sont des véritables variables, on peut les affecter à n'importe quelle valeur, tandis que la dernière est une inconnue, c'est la solution à trouver. Il faut donc trouver x pour toutes les valeurs possibles de a, b, c .

12 Règles en logique des prédicats

En logique des prédicats, pour trouver une preuve, on va faire des manipulations de formules.

12.1 La substitution

Une manipulation fréquente en logique des prédicats est la **substitution**. Elle consiste à prendre une formule et remplacer une partie par une autre.

Si on a une formule $p[x/t]$ (où p veut dire "toutes les formules"). On va remplacer toutes les occurrences libres de x par t . On peut aussi écrire cela de la manière suivante :

$p[x/t]$ possède deux portées :

La première se note $p[x]$ et correspond à une partie de la règle.

La seconde se note $p[t]$ et correspond à l'autre partie de la règle.

Exemple :

1.	$P(x) \rightarrow \forall y \cdot (P(x) \wedge R(y))$	$[x/y]$ veut dire qu'on va remplacer toutes les occurrences libres de x par y .
2.	$P(y) \rightarrow \forall y \cdot (P(y) \wedge R(y))$	En remplaçant x par y , on a changé le sens de la formule car avant, x n'était pas dans la portée du quantificateur alors que maintenant il l'est. Ce changement de sens s'appelle une capture de variable , car la variable y est capturée par le quantificateur. Pour résoudre ce problème, on va effectuer un renommage .
3.	$P(y) \rightarrow \forall z \cdot (P(y) \wedge R(z))$	Résultat après renommage (pour éviter la capture de variable).

13 Elimination de \forall

$\forall x \bullet P(x) \rightarrow P(a)$ a est une constante
 $\rightarrow P(y)$ y est une variable ($P_I(y_I)$ est vrai $y_I \in P_I$)
 $\forall =$ pour tout $x_I \in P_I : P_I(x_I)$ est vrai

Règle :

$$\frac{\forall x:p}{p[x/t]}$$

Substitution : t remplace x

/!\ Il faut faire du renommage dans certains cas !

Exemple :

1. $\forall x \bullet \forall y \bullet P(x,y)$ Prémisse
2. $\forall y \bullet P(x,y)$ Elimination de \forall
3. $P(x,x)$ Elimination de $\forall \rightarrow$ Pas de renommage car pas de capture
4. $\forall x \bullet P(x,x)$ Introduction de \forall

14 Elimination de \exists

$\exists x \bullet P(x) \rightarrow P(x)$ x = nouvelle constante qui apparait nulle part ailleurs
 $(val_I(a) = x_I)$

Il existe un $x_I \in D_I$ avec $P_I(x_I)$ est vrai

$\rightarrow P(y)$ y = variable qui existe déjà dans la preuve

$\rightarrow P(z)$ z = nouvelle variable dans la preuve $val_I(z) = x_I$

Exemple 1 :

1. $\exists x \bullet \text{chef}(x)$ Prémisse
2. $\exists x \bullet \text{voleur}(x)$ Prémisse
3. $\text{chef}(y)$ Elimination de \exists
4. ~~$\text{voleur}(y)$~~ Elimination de \exists y n'est pas une nouvelle variable dans la preuve
5. $\text{chef}(y) \wedge \text{voleur}(y)$ Conjonction
6. $\exists y \bullet \text{chef}(y) \wedge \text{voleur}(y)$ Introduction de \exists

Exemple 2 :

- | | |
|---|---------------------------|
| 1. $\exists x \bullet \text{chef}(x)$ | Prémisse |
| 2. $\exists x \bullet \text{voleur}(x)$ | Prémisse |
| 3. $\text{chef}(y)$ | Elimination de \exists |
| 4. $\text{voleur}(z)$ | Elimination de \exists |
| 5. $\text{chef}(y) \wedge \text{voleur}(z)$ | Conjonction |
| 6. $\exists y \exists z \text{chef}(y) \wedge \text{voleur}(z)$ | Introduction de \exists |

15 Introduction de \exists

Règle :

$$\frac{p[t]}{\exists x \bullet p[x]}$$

Il y a une substitution $p[x/t]$

Exemple :

$$\frac{P(y,y)}{\exists x \bullet P(x,x)}$$

$$\frac{P(y,x)}{\exists x \bullet P(x,x)}$$

Ceci n'est pas correct !

\Rightarrow Il doit être possible de retrouver la formule original en remplaçant.

16 Introduction de \forall

Règle :

$$\frac{p}{\forall x \bullet p}$$

- Si p n'a pas d'occurrence libre de x alors c'est OK
- Si p contient une occurrence libre de x : on doit s'assurer que la preuve jusqu'à cet endroit marchera pour toutes valeurs affectées à x
 - \hookrightarrow Aucune formule dans la preuve jusqu'à cet endroit ne doit mettre une contrainte sur x !

Deux conditions :

- x n'est pas libre dans une formule dans la preuve jusqu'à cet endroit obtenu par élimination de \exists
- x n'est pas libre dans une prémisse (x est déjà connu au début donc il possède déjà une valeur)

Exemple :

1. $\forall x \exists y \text{ parent}(y,x)$ Prémisse
2. $\exists y \text{ parent}(y,x)$ Elimination de \forall
3. $\text{parent}(y,x)$ Elimination de $\exists \rightarrow$ N'est valable que pour ce y et ce x , pas pour tous
4. ~~$\forall x \text{ parent}(y,x)$~~ Introduction de $\forall \rightarrow$ On ne peut pas faire ça car il y a une contrainte sur x . Là on dit que ce y est parent de tous !

17 Partie 9

On va terminer le point traité au cours précédent par un exemple un peu plus conséquent d'une preuve manuelle en logique des prédicats avant d'introduire l'algorithme permettant d'effectuer des preuves de manière automatisée.

Exemple de preuve manuelle

Il est important de pouvoir faire des preuves manuellement, car cela permet de bien comprendre toutes les étapes de raisonnement d'une preuve, même si par la suite on utilise un algorithme plutôt que de faire les preuves à la main.

L'exemple suivant est inspiré de l'Empire romain :

Prémisses

- Les maîtres et esclaves sont tous des hommes adultes
- toutes les personnes ne sont pas des hommes adultes

Note : on voit dans les prémisses qu'il y a des quantificateurs : tous, toutes.

A prouver

- il existe des personnes qui ne sont pas des maîtres

Preuve

1. $\forall x (maitre(x) \vee esclave(x) \implies adulte(x) \wedge homme(x))$ prémisses
2. $\neg \forall x (adulte(x) \wedge homme(x))$ prémisses
3. $\exists x \neg (adulte(x) \wedge homme(x))$ théorème négation
S' il n'est pas vrai que toutes les personnes sont des hommes adultes alors il existe une personne qui n'est pas un homme adulte
4. $\neg (adulte(x) \wedge homme(x))$ \exists elim
On élimine le quantificateur existentiel : on peut le faire, car on introduit une variable x qu'on choisit comme étant une personne rendant vraie la proposition.
5. $(maitre(x) \vee esclave(x) \implies adulte(x) \wedge homme(x))$ \forall elim
On peut retirer le \forall en réduisant le champ de x aux x rendant vraie la proposition.
6. $\neg (maitre(x) \vee esclave(x))$ modus tollens
7. $\neg maitre(x) \wedge \neg esclave(x)$ de Morgan

8. $\neg \text{maitre}(x)$ simplification
9. $\exists y \neg \text{maitre}(y)$ \exists intro
Comme dans l'interprétation, x est une personne qui rend valable cette proposition, on peut dire qu'il existe une personne rendant valable cette proposition et réintroduire le quantificateur \exists

C'était un exemple très simple ne faisant que quelques pas, mais la logique est assez expressive pour permettre des preuves plus complexes (par exemple, formaliser les mathématiques), le nombre de pas serait alors beaucoup plus important.

Instant Histoire :

A la fin du 19ème siècle, début du 20ème :

- création de la logique de 1er ordre (Gottlob Frege)
- Deux personnes ont essayé de formaliser toutes les mathématiques. Principia Mathematica (Alfred Whitehead, Bertrand Russell)

Lors que l'arrivée des ordinateurs, fin du 20ème siècle (années 50-60 et fin du siècle) on a essayé de formaliser la logique via des algorithmes :

- Création de l'Algorithme de Preuves (1965) :
 - Alan Robinson crée La Règle de Résolution (qui va être expliquée aujourd'hui)
 - Création de prouveurs (assistants de preuve) par exemple Coq et Isabelle en 1972
 - Création de la logique de programmation qui aide à l'élaboration de la programmation par contraintes : Prolog (1972)
- Création de la sémantique Web : OWL (Web Ontology Language)

18 Algorithme de preuve pour la logique des prédicats

- Cet algorithme s'inspire de l'algorithme de réfutation de la logique des propositions [résolution forme clausale]
- Pour la logique des prédicats, c'est un peu plus compliqué, mais ça marche !

On peut le faire marcher malgré la complexité des variables et des quantificateurs ce qui est assez étonnant, car c'est une logique très expressive. Arriver à trouver un algorithme permettant de traiter la logique des prédicats était une sorte de Graal au 20ème siècle.

18.1 3 transformations :

On va commencer par faire les transformations de normalisation. Il y a 3 transformations à effectuer :

1. formule \rightarrow forme prénexe :

$$(\dots \forall \dots \exists \dots \forall) \implies \forall \exists \forall (\dots)$$

Tous les quantificateurs sont mis en tête de la formule. Les quantificateurs étant très compliqués à gérer, on transforme la formule pour les extraire de celle-ci.

Les modèles sont conservés durant cette transformation.

2. forme prénexe \rightarrow forme Skolem (élimination des \exists) :

$$\forall \exists \forall (\dots) \implies \forall \forall \forall (\dots)$$

Les quantificateurs existentiels sont très embêtants, car ils sont restrictifs. Ils disent qu'il existe des éléments, mais ne précisent pas lesquels, on va donc les éliminer.

Cette transformation préserve l'existence des modèles, mais pas les modèles eux-mêmes. Ils doivent être modifiés pour conserver la même signification.

3. forme Skolem \rightarrow forme normale conjonctive :

$$\forall \dots \forall \wedge_i (\forall_j L_{ij})$$

(*meme algorithme que pour la logique des propositions*)

Cette transformation est la même que celle effectuée dans la logique des propositions.

Les modèles sont préservés lors de cette transformation.

18.2 Résolution

En logique des propositions :

$$\frac{L \vee C_1, \neg L \vee C_2}{C_1 \vee C_2}$$

Cela fonctionne en logique des propositions, car il n'y a pas de variables, mais ici, on peut avoir $L_1 \vee C_1$ et $\neg L_2 \vee C_2$ avec L_1 et L_2 qui ont des variables différentes. Par exemple $P(x,a)$ et $P(y,z)$.

Pour pouvoir faire la résolution, il va falloir en quelque sorte les rendre identiques.

On va donc dire : ce ne sont peut-être pas toujours les mêmes, mais, pour certaines valeurs, ils sont identiques. Si $x=y$ et $a=z$ alors on peut faire la résolution.

Unification

$L_1 : P(x,a)$

$L_2 : P(y,z)$

Pour que L_1 et L_2 soient identiques, on va restreindre les variables et faire une substitution.

(a : constante, x, y, z : variables)

$$\sigma = \{(x, y), (z, a)\}$$

$$P(x, a) \rightarrow P(y, a)$$

$$P(y, z) \rightarrow P(y, a)$$

Cette résolution marche pour toutes les valeurs qui sont limitées par la substitution. Le résultat ne sera donc pas général. Cette opération s'appelle l'unification et utilise la substitution σ (sigma). On peut maintenant faire la résolution en appliquant le même algorithme de réfutation que pour la logique des prédicats :

$$\frac{L_1 C_1, \neg L_2 \vee C_2}{(C_1 \vee C_2)\sigma}$$

18.3 Propriétés de cet algorithme

- Cet algorithme est moins fort que pour la logique des propositions, car la logique des prédicats est beaucoup plus expressive.
- adéquat : Si $B \vdash T \rightarrow B \models T$
si l'on trouve une preuve de T avec les axiomes B alors T sera vrai dans tous les modèles de B
- complet : Si $B \models T \rightarrow B \vdash T$
Si quelque chose est vrai dans tous les modèles alors on va trouver une preuve
- L'algorithme possède les propriétés d'un algorithme semi-décidable :
 - Si $B \vdash T \rightarrow$ l'algorithme trouve une preuve.
 - Si $B \not\vdash T \rightarrow$ il peut tourner en rond indéfiniment.*Si ce qu'on tente de prouver est vrai dans tous les modèles, l'algorithme va finir par trouver une preuve, mais si ce n'est pas vrai, l'algorithme va tourner en rond et ne jamais se terminer. Le problème est donc que quand l'algorithme prend trop de temps à trouver une preuve, on doit l'arrêter et l'on n'est jamais certain du résultat. On ne peut jamais être sûr que l'algorithme n'aurait pas trouvé une preuve si on l'avait laissé tourner plus longtemps. Il est donc semi-décidable, car ses résultats ne sont totalement fiables que dans le cas où une preuve est trouvée.*

18.4 Transformation de la formule de base vers la forme pré-nexe

Etapes de la transformation en formule logiquement équivalente :

1. Éliminer \Leftrightarrow et \implies
2. Renommer les variables.

- Chaque quantificateur ne porte que sur une variable, il faudra en créer de nouvelles si besoin en prenant soin de conserver l'équivalence de la formule .
 - Attention : ne jamais garder le même nom de variable pour une variable libre et une variable liée.
 - Supprimer les quantificateurs si possible.
3. Migrer les négations (\neg) vers l'intérieur, vers les prédicats. On peut faire cela, car $\neg\exists$ peut être transformé en $\forall\neg$ et vice versa.
 4. On peut mettre tous les quantificateurs de la logique des prédicats à l'avant de la formule.

Exemple d'une transformation en forme prénexe

1. $\forall x[p(x) \wedge \neg(\exists y)\forall x(\neg q(x, y)) \Rightarrow \forall z\exists v \bullet p(a, x, y, v)]$
Expression de base
2. $\forall x[p(x) \wedge \neg(\exists y)(\forall x)(\neg \neg q(x, y) \vee \forall z\exists v \bullet r(a, x, y, v))]$
Suppression des \Rightarrow
3. $\forall x[p(x) \wedge \neg(\exists y)(\forall u)(\neg \neg q(u, y) \vee \forall z\exists v \bullet r(a, u, y, v))]$
Renommage des variables et suppression des quantificateurs inutiles
4. $\forall x[p(x) \wedge \forall y \neg (\forall u)(\neg \neg q(u, y) \vee \exists v \bullet r(a, u, y, v))]$
 $\neg\exists y$ devient $\forall y \neg$ et simplification des \neg
5. $\forall x[p(x) \wedge \forall y \neg (\exists u \neg (q(u, y) \vee \exists v \bullet r(a, u, y, v)))]$
 $\neg\forall u$ devient $\exists u \neg$
6. $\forall x[p(x) \wedge \forall y \exists u (\neg q(u, y) \wedge \neg(\exists v) \bullet r(a, u, y, v))]$
Distribution des \neg (de Morgan)
7. $\forall x[p(x) \wedge \forall y \exists u (\neg q(u, y) \wedge (\forall v) \bullet \neg r(a, u, y, v))]$
 $\neg\exists v$ devient $\forall v \neg$
8. $\forall x \forall y \exists u \forall v \bullet [p(x) \wedge (\neg q(u, y) \wedge \neg r(a, u, y, v))]$
Extraction des quantificateurs.

Algorithme de preuve pour la logique des prédicats

19 Introduction

19.1 3 transformations

19.2 Résolution

19.3 Propriétés de l'algorithme

20 Transformation en forme prénexe

21 Transformation en forme Skolen

21.1 Intuition

Cette transformation consiste à éliminer toutes les occurrences de quantificateurs existentiels.

$$(\forall x)(\forall y)(\exists u)(\forall v)[P(x) \wedge \neg Q(u, y) \wedge \neg R(a, u, y, v)]$$

Dans ce cas-ci, la valeur de u dépend des valeurs de x et y . Lorsqu'on a choisi x et y , on est alors libre de choisir u . On peut donc supposer qu'une fonction $g(x, y)$ fournit cet élément de façon à conserver la satisfaisabilité de la formule tout en supprimant $(\exists u)$.

$$(\forall x)(\forall y)(\forall v)[P(x) \wedge \neg Q(g(\mathbf{x}, \mathbf{y}), y) \wedge \neg R(a, g(\mathbf{x}, \mathbf{y}), y, v)]$$

Après la transformation, l'existence des modèles est préservée.

21.2 Règle

Pour chaque élimination d'un quantificateur existentiel $(\exists x)$, on remplace sa variable quantifiée par une fonction $f(x_1, \dots, x_n)$ dont les arguments sont les variables des quantificateurs universels dont x est dans la portée.

Justification par un exemple :

$$p : \forall x \forall y \exists z [\neg P(x, y) \vee Q(x, z)]$$

$$p_s : \forall x \forall y [\neg P(x, y) \vee Q(x, f(\mathbf{x}, \mathbf{y}))]$$

Les modèles de $p \neq$ modèles p_s .

Pour p

— Interprétation I

— $D_I = \text{Professeur} \cup \text{Université}$

- $Val_I(P) = P_i = \text{“a enseigné à l’université”}$
- $Val_I(Q) = Q_i = \text{“est diplômé de l’université”}$
- $Val_I(f)$ n’existe pas.

Pour p_s

- Étendre I
- $I' = \{F \vdash F_i\} \circ I$
- $f(a, b) = \text{“l’université ayant dû diplômer } a \text{ pour que } a \text{ puisse enseigner à } b\text{”}$

p admet un modèle (I) SSI p_s admet un modèle (I') .

Que ça ne soit exactement le même modèle ne pose pas de problème pour notre algorithme. L’algorithme par réfutation continue à itérer jusqu’à trouver une contradiction (*false*). S’il n’y a pas de modèle pour p_s , il n’y a pas de modèle pour p et ça suffit.

22 Transformation en forme normale conjonctive

Même manipulations qu’en logique des propositions.

23 La règle de résolution

$$\frac{L_1 \vee C_1, \neg L_2 \vee C_2}{(C_1 \vee C_2)\sigma}$$

Cette règle de résolution ne fonctionne que si L_1 et L_2 sont identiques.

- $L_1 = P_1(a, y, z)$
- $L_2 = P_1(x, b, z)$

Dans un modèle il y a un prédicat qui correspond au symbole P_1 et il y a un ensemble de triplets qui rendent vrai P_1 .

L’unification de L_1 et L_2 donne L qui représente l’intersection des deux ensembles. On écrit :

- $L_1\sigma = L$
- $L_2\sigma = L$

L_1 et L_2 sont unifiables s’il existe une substitution σ telle que $L_1\sigma = L_2\sigma$.

- $\sigma = \{(x, a), (y, b)\}$
- $L_1\sigma = P(a, b, z)$
- $L_2\sigma = P(a, b, z)$

Exemple :

— $L_1 = P_1(a, x)$

— $L_2 = P_2(b, x)$

Il n'y pas de substitution qui existe car on a deux constantes différentes, l'intersection des deux ensembles est vide.

Exemple 2 :

— $L_1 = P(f(x), z)$

— $L_2 = P(y, a)$

Dans ce cas-ci, il y a beaucoup de substitutions possibles telles que :

— $\sigma_1 : \{(y, f(a)), (x, a), (z, a)\} \Rightarrow L_1\sigma_1 = L_2\sigma_1 = p(f(x), a)$

— $\sigma_2 : \{(y, f(x)), (z, a)\} \Rightarrow L_1\sigma_2 = L_2\sigma_2 = p(f(x), a)$

Dans ce cas-ci, σ_2 est plus général.

— On préfère alors la substitution σ la plus générale.

— On peut démontrer qu'il existe un unificateur plus général U.P.G.

— U.P.G. est calculable.

Règle de résolution :

— p_1, p_2 clauses

— $p_1 = L^+ \vee C_1$

— $p_2 = \neg L^- \vee C_2$

— L^+ et L^- ont les mêmes symboles de prédicat.

— $\{L^+, L^-\}$ unifiable par σ U.P.G.

Alors

$$\frac{L^+ \vee C_1, \neg L^- \vee C_2}{(C_1 \vee C_2)\sigma}$$

24 Algorithme

$S := \{\Delta x_1, \dots, \Delta x_i, \neg Th\}$ dont chaque formule est en forme normale conjonctive (FNC).

while : **false** $\ni S$ et il existe une paire de clauses résolubles et non-résolues.

do : choisir p_i, p_j dans S et L tel que :

— L^+ dans p_i

— L^- dans p_j

— $\{L^+, L^-\}$ unifiable par σ U.P.G.

Calculer :

— $r := (p_i - [L^+] \vee p_j - [\neg L^-])\sigma$

— $S_i = S \cup \{r\}$

end

If **false** $\ni S$ then Th prouv  . else Th non prouv  . end

25 Exemple

- $(\forall x)homme(x) \wedge fume(x) \Rightarrow mortel(x)$
- $(\forall x)animal(x) \Rightarrow mortel(x)$
- $homme(Ginzburg)$
- $fume(Ginzburg)$
- **Candidat-th  or  me** : $mortel(Ginzburg)$

25.1 Initialisation de S

- P1 : $(\forall x)homme(x) \wedge fume(x) \Rightarrow mortel(x)$
- P2 : $(\forall x)animal(x) \Rightarrow mortel(x)$
- P3 : $homme(Ginzburg)$
- P4 : $fume(Ginzburg)$
- P5 : $\neg mortel(Ginzburg)$

25.2 It  rations

A

- P1 + P5
- $\sigma = \{(x, Ginzburg)\}$
- $r = P6 = \neg homme(Ginzburg) \vee \neg fume(Ginzburg)$

B

- P3 + P6
- $\sigma = \{\}$
- $r = P7 = \neg fume(Ginzburg)$

C

- P4 + P7
- $\sigma = \{\}$
- $r = false$. Inconsistence donc le candidat th  or  me est prouv  .

25.3 Non-d  terminisme

Importance des choix qu'on fait.

A

- P2 + P5
- $\sigma = \{(x, Ginzburg)\}$
- $r = \neg animal(Ginzburg)$

Si on avait fait ce choix-ci pour la première itération, l'algorithme ne peut plus continuer et on doit faire marche-arrière.

26 Stratégies

- Quelles paires p_i, p_j choisir ?
- Quelles L^+, L^- choisir ?

Les assistants de preuves utilisent des stratégies existantes, avec l'input de l'humain.

Le langage **Prolog**, inventé en 1972 par Alain Colmerauer et Robert Kowalski, utilise volontairement des stratégies naïves qui permettent de rendre l'algorithme prévisible. Les axiomes deviennent un programme, c'est la programmation logique. Un exemple de stratégie naïve est la stratégie LUSH qui choisit de haut vers le bas les paires p_i, p_j , et de gauche à droite dans p_i .

Etude des structures discrètes

Exemples de structures discrètes : entiers positifs, chaînes, arbres, ensembles, relations, fonctions, ...

On peut définir ces structures avec la logique des prédicats et faire des raisonnements sur celles-ci en utilisant des règles d'inférence.

Utilisation :

- On peut programmer avec ces structures. *Prolog* permet d'écrire les axiomes directement. Il faut cependant faire attention de bien les choisir ;
- On peut les utiliser dans les assistants de preuve (exemples d'assistants de preuve : *Coq*, *Isabelle*)

Théorie du premier ordre

Intuition : définition logique d'une structure mathématique

Définition d'une théorie

- Sous-language de la logique du premier ordre
 - **vocabulaire** : constantes, fonctions, prédicats ;
 - règles syntaxiques et sémantiques sur ce vocabulaire ;
- Ensemble d'axiomes (formules fermées, c'est-à-dire formules ne contenant pas de variables libres) ;
- Ensemble de règles d'inférence.

Exemple : théorie des liens familiaux (FAM)

1. Vocabulaire :

- 2 symboles de fonctions : $p/1$, $m/1$
- 3 symboles de prédicats : $P/2$, $GM/2$, $GP/2$

On peut interpréter les fonctions p et m comme "père de" et "mère de", et les prédicats P , GM et GP comme "parent de", "grand-mère de" et "grand-père de". On donnera plus de précisions sur cette interprétation dans la suite.

2. Axiomes :

$(\forall x) (P(x, p(x)))$	(père)
$(\forall x) (P(x, m(x)))$	(mère)
$(\forall x)(\forall y) (P(x, y) \Rightarrow GP(x, p(y)))$	
$(\forall x)(\forall y) (P(x, y) \Rightarrow GM(x, p(y)))$	

3. Règles :

Les règles sont uniquement celles de la logique des prédicats.

Première interprétation

D_I : personnes

$\text{val}_I(p) = \text{"père de"}$

$\text{val}_I(m) = \text{"mère de"}$

$\text{val}_I(P) = \text{"Parent"}$

$\text{val}_I(GP) = \text{"Grand-père"}$

$\text{val}_I(GM) = \text{"Grand-mère"}$

père de : Pers \rightarrow Pers : $d \rightarrow \text{"père de" } d$

mère de : Pers \rightarrow Pers : $d \rightarrow \text{"mère de" } d$

Parent(d_1, d_2) = V ssi d_2 est un parent de d_1

Grand-père(d_1, d_2) = V ssi d_2 est un grand-père de d_1

Grand-mère(d_1, d_2) = V ssi d_2 est une grand-mère de d_1

Cette interprétation est un modèle de FAM car les axiomes sont tous vérifiés.

On remarque la ressemblance avec une théorie scientifique, où les axiomes correspondent à la théorie et l'interprétation à ce que celle-ci signifie dans le monde réel.

Une théorie peut avoir plusieurs modèles.

Seconde interprétation

Cette interprétation est également un modèle de FAM.

$D_J : \mathbb{N}$

$\text{val}_J(p) = \text{"}p_J\text{"}$

$\text{val}_J(m) = \text{"}m_J\text{"}$

$\text{val}_J(P) = \text{"}P_J\text{"}$

$\text{val}_J(GP) = \text{"}GP_J\text{"}$

$\text{val}_J(GM) = \text{"}GM_J\text{"}$

$p_J : \mathbb{N} \rightarrow \mathbb{N} : d \rightarrow 2d$

$m_J : \mathbb{N} \rightarrow \mathbb{N} : d \rightarrow 3d$

$P_J(d_1, d_2)$ ssi $d_2 = 2d_1$ ou $d_2 = 3d_1$

$GP_J(d_1, d_2)$ ssi $d_2 = 4d_1$ ou $d_2 = 6d_1$

$GM_J(d_1, d_2)$ ssi $d_2 = 6d_1$ ou $d_2 = 9d_1$

Propriétés des théories

- Une formule fermée p est **valide** dans la théorie Th si elle est vraie dans chaque modèle de Th . On écrit :

$$\models_{Th} p$$

Soit l'ensemble des axiomes $Ax = \{Ax_1, \dots, Ax_n\}$. On a bien que $\models_{Th} Ax_i$.

- q est une **conséquence logique** de p dans la théorie Th si q est vraie dans tous les modèles de Th qui rendent p vraie. On écrit :

$$p \models_{Th} q$$

- Une théorie est **consistante** si elle a au moins un modèle (> 0 modèles).
- Une théorie est **inconsistante** si elle n'a pas de modèle (0 modèles).

Comment peut-on faire pour établir $\models_{Th} p$? Il y a deux approches différentes :

1. **l'approche sémantique** : on prend un modèle quelconque de Th et on évalue $VAL_I(p)$ en utilisant le fait que $VAL_I(Ax_i) = V$;
2. **l'approche syntaxique** : théorie de preuve : on essaye de construire une preuve de p à partir des axiomes, en appliquant les règles de Th .

En pratique, la deuxième approche est utilisée beaucoup plus souvent.

Preuve (esquisse)

La preuve qui suit est une esquisse. Il manque plusieurs étapes.
On veut montrer :

$$\models_{FAM} (\forall x)(\exists z)GM(x, z)$$

$(\forall x)(\forall y) (P(x, y) \Rightarrow GM(x, m(y)))$	(Ax)
$(\forall x) (P(x, p(x)) \Rightarrow GM(x, m(p(x))))$	(Elimination de $\forall y$ et substitution $y/p(x)$)
$(\forall x P(x, p(x))) \Rightarrow \forall x GM(x, m(p(x)))$	(Distribution \forall / \Rightarrow)
$\forall x GM(x, m(p(x)))$	(Modus ponens)
$\forall x \exists y GM(x, y)$	(Introduction de \exists)

Qualité d'une théorie

Certaines qualités sont directement issues de la logique :

1. **consistante** : il est impossible de déduire p et $\neg p$ de la même théorie.
2. **minimale** : les axiomes sont indépendants : $\{Ax_1, \dots, Ax_n\} \not\models Ax_k$

Vérification : Construction d'interprétations.

$$VAL_J(Ax_k) = False$$

$$VAL_J(Ax_i) = True \text{ pour } i \neq k$$

3. **complète** : les axiomes suffisent pour prouver la propriété d'intérêt.
Sinon il faut en ajouter.

Exemple : qualité de deux théories

1. **Système de Copernic** : Chaque axiome de chaque planète est indépendant car elles tournent toutes autour du soleil.
2. **Système de Ptolémée** : Chaque axiome de chaque planète dépendent de ceux de la Terre car elle représente le centre du l'univers mais est également une planète et dispose donc d'axiomes.

27 Extension d'une théorie

Lorsqu'on a une théorie déjà existante, on souhaite parfois l'étendre afin de la rendre plus complète. Pour ce faire, on ajoute des axiomes et on étend le vocabulaire. Voici deux exemples d'extension de théorie pour mieux comprendre comment effectuer cette opération. Ils étendent tous les deux la théorie des liens familiaux (FAM) décrite dans une section précédente.

Exemple 1

Considérons le nouvel axiome suivant, que nous noterons Ax.

$$(\forall x)\neg P(x, x)$$

La nouvelle théorie ainsi étendue que nous noterons FAM* possède un axiome de plus : Ax. Cette théorie FAM* est **consistante**, c'est-à-dire qu'il existe au moins un modèle qui valide cette théorie. Pour s'en convaincre, il suffit de considérer la première interprétation de la théorie FAM de la section précédente, qui utilise les liens familiaux.

En revanche, si on considère la deuxième interprétation (deuxième modèle, noté J) de FAM qui associe les symboles p et m aux fonctions mathématiques $p_J : \mathbb{N} \rightarrow \mathbb{N} : d \rightarrow 2d$ et $m_J : \mathbb{N} \rightarrow \mathbb{N} : d \rightarrow 3d$, on observe une contradiction. En effet, dans le modèle J on a la définition suivante du prédicat P :

$$P_J(d_1, d_2) \text{ ssi } d_2 = 2d_1 \text{ ou } d_2 = 3d_1$$

Il suffit de choisir $x = 0$ dans notre nouvel axiome Ax pour constater que le modèle J ne valide pas la théorie étendue FAM*. De manière générale, l'extension d'une théorie peut donc réduire l'ensemble des modèles de celle-ci.

Exemple 2

Considérons à présent le nouvel axiome suivant, que nous noterons Adam.

$$(\forall y)\neg P(a, y)$$

où a est une constante arbitraire. Notons la théorie étendue $\text{FAM}' = \text{FAM} + \text{Adam}$. Dans cet exemple, on peut observer que FAM' est **inconsistante** car aucun modèle ne peut valider cette théorie. En effet, en partant du premier axiome de FAM (appelé "père"), nous effectuons quelques étapes pour obtenir une contradiction.

$$\begin{array}{ll} (\forall x)P(x, p(x)) & \\ \iff P(a, p(a)) & \text{Elimination } \forall \\ \iff (\exists y)P(a, y) & \text{Intro } \exists \end{array}$$

Ci-dessus, le premier axiome de FAM reformulé (père), qui est en contradiction avec le nouvel axiome (Adam), que nous reformulons ci-dessous.

$$\begin{aligned} & (\forall y) \neg P(a, y) \\ \iff & \neg(\exists y) P(a, y) \end{aligned}$$

Par la règle de preuve par contradiction, on démontre qu'aucun modèle n'est possible pour la théorie étendue FAM'. Étendre une théorie équivaut à faire de la manipulation syntaxique. Il est important de bien vérifier ce que notre modification a comme conséquence sur le nombre de modèles que la théorie accepte.

28 Liens entre théories

Dans cette section nous abordons la comparaison de différentes théories : inclusion, équivalence et quelques corollaires, ainsi que la théorie des ordres partiels stricts. Dans ce qui suit, on note Th_1 et Th_2 deux théories.

Inclusion

On dit que Th_1 est **contenue** dans Th_2 si

- Le vocabulaire de Th_1 est inclus dans le vocabulaire de Th_2 .
- Toute formule valide dans Th_1 l'est aussi dans Th_2 .

Attention, on peut donc avoir deux théories qui "parlent de la même chose" mais qui possèdent des axiomes totalement différents. TODO : expliquer pourquoi / Ajouter un exemple

Equivalence

On dit que Th_1 et Th_2 sont **équivalentes** si elles sont contenues l'une dans l'autre. Cela signifie que les deux théories "disent la même chose" et que tout modèle d'une des théories est également modèle de l'autre.

Il est important de bien faire la différence entre les **liens** entre les théories et l'**extension** d'une théorie. Le premier concept exprime ce que modélisent les théories tandis que le deuxième n'est que de la manipulation syntaxique.

Corollaires

On note respectivement V_i , M_i et Ax le vocabulaire, les modèles et les axiomes d'une théorie i .

Si $V_{Th_1} \subseteq V_{Th_2}$ et $M_{Th_2} \subseteq M_{Th_1}$ alors Th_1 est contenue dans Th_2 .

Si $V_{Th_1} \subseteq V_{Th_2}$ et tout axiome de Th_1 est aussi axiome de Th_2 alors Th_1 est contenue dans Th_2 .

Si $V_{Th_1} = V_{Th_2}$ et $\forall i, j \quad \models_{Th_2} Ax_{i,1}$ et $\models_{Th_1} Ax_{j,2}$ alors Th_1 et Th_2 sont équivalentes.

Si p une formule fermée telle que $\models_{Th_1} p$ et $Th_2 = Th_1 \cup \{p\}$ alors Th_1 et Th_2 sont équivalentes.

29 Théorie des ordres partiels stricts

Nous allons donner un autre exemple de théorie ainsi que deux interprétations différentes de cette théorie.

Vocabulaire

- Le symbole P

Axiomes

- $(\forall x) \neg P(x, x)$ (irréflexivité) (OPS1)
- $(\forall x)(\forall y)(\forall z)(P(x, y) \wedge P(y, z) \Rightarrow P(x, z))$ (transitivité) (OPS2)

Première interprétation

Soit l'interprétation I de cette théorie. On a :

- $D_I = \mathbb{Z}$
- $val_I(P) = "<"$

Lorsque l'on écrit $val_I(<) = "<"$, le premier symbole $<$ est un mot de vocabulaire dans la théorie tandis que le deuxième est une fonction. Cette fonction confère un sens au symbole. On remarque que cette interprétation est un modèle de notre théorie. En effet, la fonction $<$ sur les entiers est irréflexive et transitive. Si on a $x < y$ et $y < z$, on peut en déduire que $x < z$.

Deuxième interprétation

Soit l'interprétation J de cette théorie. On a :

- $D_J = \mathbb{Z}$
- $val_J(P) = "\neq"$

Ici on change le sens du symbole $<$:

$$val_J(<) = "\neq"$$

Cette interprétation n'est pas un modèle. En effet, par exemple, pour $x = 5, y = 3, z = 5$, on a :

$$5 < 3 \wedge 3 < 5 \not\Rightarrow 5 < 5$$

Ceci est un contre-exemple car 5 n'est pas différent de 5 (irréflexivité), et donc cette interprétation ne vérifie pas l'axiome sur la transitivité!

29.1 Exemples de programmes Prolog

Programme (code Prolog) Ce petit programme permet de calculer la factorielle d'un nombre. Il s'agit de clauses exprimant des faits, des règles et des questions. Le code commence par un fait : $0! = 1$. Ensuite, il définit une clause pour les factorielles de façon généralisée. Attention, les virgules et les points sont importants. Les virgules sont les séparateurs entre les littéraux dits négatifs tandis que le point marque la fermeture de la clause.

```
fact(0,1)
fact(N,F) :- N>0,
    N1 is N-1
    fact(N1,F1),
    F is N* F1 .
```

Requête Nous allons maintenant exécuter le programme `fact` afin de trouver la factorielle de 5 et stocker la réponse dans la variable `F`. Le prompt Prolog est représenté par `"|?- "` et la sortie standard par `"->"`. Les commentaires sont après les `"%"` ou entre `"/* */"`. Il ne faut pas oublier le point à la fin de la requête.

```
|?- fact (5,F). %requête
-> F=120 % réponse
```

Forme clausale Le programme réécrit sous forme clausale.

$$\begin{aligned} & fact(0,1) \\ & \wedge \\ & (\neg n > 0 \\ & \vee \neg minus(n1, n, 1) \\ & \vee \neg fact(n1, f1) \\ & \vee \neg times(f, n, f1) \\ & \vee fact(n, f)) \end{aligned}$$

Exécution Le but de l'exécution est de prouver que $G \equiv fact(5, r)$. Pour y arriver, Prolog va faire une suite de substitutions σ afin de vérifier les affirmations contenues dans r . La substitution finale contiendra le résultat final.

$$\begin{aligned}
G &\equiv fact(5, r) \\
r &=< fact(5, r) > \\
\% (1) \\
\sigma &= \{(n, 5), (f, r)\} \\
r &=< (5 > 0), minus(n1, 5, 1), fact(n1, f1), times(r, 5, f1) > \sigma \\
\% (2) \\
r &=< minus(n1, 5, 1), fact(n1, f1), times(r, 5, f1) > \\
\sigma' &= \{(n1, 4) \cup \sigma\} \\
r &=< fact(4, f1), times(r, 5, f1) > \\
[\dots] \\
\sigma_{res} &= \{(r, 120), \dots\}
\end{aligned}$$

- (1) avec clause 2
(2) a>b existe dans le système prédéfini -> true

29.1.1 Exemple 2

Le but de cet exemple est de faire un "append" de deux listes :
append(L1, L2, L2)

Prolog :

- append ([] , L , L)
- append ([X|L1], L2, [X|L3]) :- append (L1,L2,L3)

Clausal Le programme réécrit sous forme clausale.

append(nil,l',l')

\wedge

(\neg append(l_1, l_2, l_3) \wedge append (cons(x, l_1, l_2 , cons(x, l_3)))

Execution Attention, l'exécution du programme est une preuve en soit.

G = append (cons(1,nil), cons(2, nil), l)

1. $r = < \text{append}(\text{cons}(1, \text{nil}), \text{cons}(2, \text{nil}), l) >$ et $\sigma_1 = (x, 1), (l_1, \text{nil}), (l_2, \text{cons}(2, \text{nil})), (\text{cons}(x, l_3), l)$
2. $r = < \text{append}(\text{nil}, \text{cons}(2, \text{nil}), l_3) >$ et $\sigma_2 = (l', \text{cons}(2, \text{nil}), (l_3, l'))$
3. $r = < >$

Résultat $l = \text{con}(1, l_3) \quad l_3 = l' \quad l' = \text{cons}(2, \text{nil}) \quad l = \text{con}(1, \text{cons}(1, \text{nil}))$

Références

- [Nis] Nimal Nissanke. *Introductory Logic and Sets for Computer Scientists*.
- [LPP] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets : Reasoning About a Highly Connected World*.