Chapitre 1 Introduction
Quel est le problème ?
Un peu d'histoire.
Applications en informatique.

Première Partie : La logique des propositions (LP)

Chapitre 2 Le langage de la logique des propositions
Alphabet (Intuition des connecteurs, les détails suivront)
    Ensemble PROP des propositions : p,q,r,... éventuellement indicées
    Elément absurde (bottom)
    Connecteurs (unaire, binaire, arité)
    Parenthèses
Les abréviations T, $\leftrightarrow$ et $\oplus$
Simplification du parenthésage (externes et autour des négations de propositions)
Définition par induction de l'ensemble FORM des formules (bien formées)
Preuves par induction et exemple simple
Arbre d'une formule (terminologie : noeud, racine, feuille, fils, père,...)
Occurrence dans une formule
Exemple d'occurrences
Substitution d'une occurrence
Substitution de toutes les occurrences

Chapitre 3 La théorie des modèles : sémantique de la logique des propositions (LP)
Les tables de vérité de chaque connecteur
Tautologie, formule contradictoire
Les valuations
Validité, Satisfiabilité, Insatisfiabilité d'une formule / Modèles et contre-modèles d'une formule
Validité, Satisfiabilité, Insatisfiabilité d'un ensemble (éventuellement infini) de formules
La conséquence logique et l'équivalence logique

Equivalences logiques remarquables

Théorème de substitution des équivalents

LES METHODES DE PREUVE (chapitres 4, 5 et 6)
Chapitre 4 La méthode des tableaux (sémantiques)
Présentation informelle avec exemples
Présentation complète et formelle
Reprise des exemples
Terminaison. Adéquation et complétude de la méthode (Preuve en une demie-heure ?)

Chapitre 5 La déduction naturelle
Utilisation de la conséquence logique et de ses propriétés : Vers la théorie de la preuve
Système de DN complet
Quelques exemples
Adéquation et complétude admises

Chapitre 6 Réfutation par résolution
Formes Normales Disjonctives et Conjonctives. Ensemble de Clauses.
Principe de résolution

Chapitre 7 Extension de la logique classique des propositions
(Brève incursion du côté des logiques non-classiques)
Abandon de la bivalence : exemple de logique trivaluée (sujet de septembre 03)
Abandon de la vérifonctionnalité : exemple de logique modale (sujet de janvier 01)
Abandon de l'atomicité des propositions : la logique des prédicats

Deuxième Partie : La logique du premier ordre (LP1) ou logique des prédicats

Chapitre 8 Introduction
Analyse des propositions en terme d'objets (ou n-uples d'objets) et de propriétés sur ces objets

Chapitre 9 Le langage
L'alphabet :
VAR (ens. de variables),
CONST (ens. de constantes),
FONC_n (ens de fonctions d'arité n),
PRED_n (ens de prédicats d'arité n)
Définition des termes
Définition des formules
Occurrences libres/liées de variables - Substitution des occurrences libres (noté [t/x])
Notion de substitution saine : [t/x] est saine dans A si aucune vble de t n'est capturée par un quantificateur de A lors de la subtitution, contre-exemple : [y/x] n'est pas saine dans $\exists y\ (y > x)$

Chapitre 10 La théorie des modèles
Approche informelle
Interprétation = Domaine + valuation = <D,v> où :
v : VAR U CONST $\rightarrow$D,
v: FONC_n $\rightarrow 2\char`^(D^n$ --> D)
v: PRED_n $\rightarrow 2\char`^(D^n$ --> {0,1})
Interprétation d'une formule atomique de la forme $P(t_1,...,t_n)$
Interprétation d'une formule complexe

Chapitre 11 La théorie de la preuve
Notation : Free(A) : ens des variables libres de A
Notation : HYP(A) : ens des hypothèses non-déchargées dont dépend A
Notation : Free(HYP(A)) : ens des vbles libres des hypothèses non-déchargées dont dépend A
Règle d'introduction du $\exists$
Règle d'introduction du $\forall$
Règle d'élimination du $\forall$
Règle d'élimination du $\exists$
Exemples :
$\forall x\ (F) \leftrightarrow \neg\exists x\ (\neg(F))$
$(\forall x\ (F) \vee G)$ <--> $(\forall x\ (F \vee G))$ si $x \notin$ Free(G)
$(\exists x\ (F) \wedge G)$ <--> $(\exists x\ (F \wedge G))$ si $x \notin$ Free(G)
$\forall x$ (F) --> $\forall y$ (F[y/x]) si [y/x] est une substitution saine (théorème du renommage)

Enoncé du théorème d'adéquation et complétude de la déduction naturelle

Chapitre 12 : Théories du premier ordre

La théorie (infinitaire) $Th_=$ de l'égalité
Axiomes de réflexivité (R), symétrie (S) et transitivité (T) et schéma d'axiome de substitutivité des égaux (SE(F)) : $(F \wedge x=y) \rightarrow F[y/x]$ si [y/x] est saine dans F

$Th_= = \{R,S,T\} \cup \{SE(F) / F \in FORM\}$

La théorie $Th_\leq$ des ordres
Axiomes de réflexivité (R), de transitivité (T) et d'antisymétrie (AS): $\forall x(\forall y((x\leq y \ \&\ y\leq x) \rightarrow x=y)$

$Th_{\leq} = Th_{=} \cup \{R,T,AS\}$

L'arithmétique de Peano

Langage

$Arith_0$: Peano sans induction

Constat : on peut prouver t+t'=t'+t pour tout couple de termes t et t' sans variables du langage, mais on ne peut dériver $\forall x(\forall y(x+y=y+x))$

$Arith_1 = Arith_0$ U {Ind(F) / F $\in$ FORM}

où Ind(F) = (F[0/x] $\wedge$ (F[y/x] --> F[S(y)/x])) $\rightarrow$ $\forall$x(F), si [y/x] est saine dans F

Modèles non-standard de $Arith_1$: présentation informelle

- Enoncé du théorème d'incomplétude et son argument diagonal : Card(Valid(**N**,0,+,*,<)) > Card(Th($Arith_1$))

.

Première Partie : la logique des propositions

Chapitre 1 Le langage de la logique des propositions

Ce sont les stoïciens qui identifient la notion de « proposition » et se proposent d'étudier comment les propositions élémentaires se combinent pour former des propositions complexes. Au sens large, une proposition est un énoncé qui est soit vrai, soit faux (sans que l'on ait besoin de savoir s'il est l'un ou l'autre). Parmi les propositions, certaines (« Il pleut et, soit il fait jour, soit la lumière est allumée ») sont composées de propositions plus simples (ici, « Il pleut » et « Soit il fait jour, soit la lumière est allumée ») qui sont « assemblées » par un mot qui joue un rôle particulier (en gros ce sont les conjonctions de coordination et de subordination de la grammaire française) et que nous appellerons connecteurs. Une proposition élémentaire, et qui n'est pas formé de propositions plus simples. Dans notre exemple « Il pleut » est une proposition élémentaire, mais pas « Soit il fait jour, soit la lumière est allumée » qui elle-même est composée de deux propositions élémentaires « Il fait jour » et « La lumière est allumée » reliées par « Soit…soit ». On verra de combien de connecteurs différents on a besoin pour prendre en compte tous les rapports possibles entre deux propositions.

Dans le cadre de la logique des propositions, on utilisera des variables notées p,q,r,… qui seront sensées représenter des propositions. Ces variables seront appelées propositions par abus de langage, il s'agit en fait de symboles de propositions : en soi p n'est pas vraie ou fausse, par contre « Il pleut » est vraie ou pas suivant le temps qu'il fait dehors (au moment où on lit ces lignes…). De même, on utilisera des symboles particuliers pour faire référence aux connecteurs, ceci afin d'éviter toute ambiguïté que pourraient introduire les connecteurs du français (le mot « et » introduit parfois une notion de temps).

Préambule (si nécessaire)

Cardinalité des ensembles : Ensemble finis / infinis dénombrables / infinis non-dénombrables

1 L'alphabet

Le langage de la logique des propositions est constitué :

- D'un ensemble PROP (éventuellement infini mais dénombrable) de lettres propositionnelles destinées à représenter les propositions : elles seront notées par des lettres minuscules (p, q, r …) éventuellement indicées (p_0, q_1, …). Par abus de langage, on appellera propositions les éléments de PROP.

- D'un ensemble de connecteurs binaires $\{\wedge, \vee, \rightarrow\}$, du connecteur unaire $\neg$, et du symbole $\perp$

- Des parenthèses gauche et droite.

Abréviations $\mathsf{T} = \neg\perp$ $\quad (A\leftrightarrow B) = ((A\rightarrow B) \wedge (B\rightarrow A))$ $\quad (A\oplus B) = ((A \vee B) \wedge \neg (A \wedge B))$

Simplifier le parenthésage (parenthèses externes et négation prioritaire).

2 Définitions et preuves par induction

Parmi toutes les séquences (les suites) composées des symboles de l'alphabet présenté ci-dessus, nous serons amené à en examiner de particulières, composant des sous-ensembles de l'ensemble de toutes les séquences possibles.

Pour cela, nous utiliserons des « définitions par induction » qui permettent à leur tour d'établir des « preuves par induction ».

Définition par induction : le schéma général en est le suivant, supposons que nous voulions définir un sous-ensemble E parmi toutes les séquences (ou mots) composées avec l'alphabet, une définition de E par induction procède en deux étapes (notez la ressemblance avec une définition par récurrence) :

Etape 1 (d'initialisation) : quelles sont les séquences dont on peut directement dire qu'elles sont dans E ?

cette étape définit la base notée E_0
Supposons que E_n ait été défini :
Etape 2 (d'induction) : on définit E_{n+1} en expliquant quels nouveaux éléments peuvent être obtenus à partir des éléments des E_i précédents.

Finalement $E = \bigcup_{i\in \mathbf{N}} E_i$, c'est-à-dire que $x\in E$ si et seulement si $\exists i\in \mathbf{N}$ tel que $x\in E_i$

Définition de l'ensemble FORM des formules :
Initialisation : $PROP\subseteq FORM$ et $\perp\in FORM$
Induction :

si $A\in FORM$	alors	$\neg A$	$\in FORM$
si $A,B\in FORM$	alors	$(A\wedge B)$	$\in FORM$
		$(A\vee B)$	$\in FORM$
		$(A\to B)$	$\in FORM$

Toute formule de FORM peut être obtenue par application des étapes ci-dessus.

Ainsi, $((p\wedge\perp) \to(\neg(r\wedge s))) \in$ FORM car :
$p,\perp,r,s \in FORM$ (Initialisation)
donc $(p\wedge\perp)$ et $(r\wedge s)\in FORM$ (Induction)
donc $\neg(r\wedge s)\in FORM$ (Induction)
et donc $((p\wedge\perp) \to(\neg(r\wedge s)))$

On peut aussi bien *définir l'ensemble FORMBIN des formules ne contenant que des connecteurs binaires et donc pas de négation ($\neg$)* :
Initialisation : $PROP\subseteq FORMBIN$ et $\perp\in$ FORMBIN
Induction :

si $A,B\in$ FORMBIN	alors	$(A\wedge B)$	$\in$ FORMBIN
		$(A\vee B)$	$\in$ FORMBIN
		$(A\to B)$	$\in$ FORMBIN

Toute formule de FORMBIN peut être obtenue par application des étapes ci-dessus.

On peut aussi définir (par induction) diverses quantités, ensembles,… comme la longueur lg(A) d'un élément d'un ensemble F défini inductivement, en suivant les étapes permises pour F.

Exemples :
Définition de la longueur d'une formule A de FORM :
Initialisation : $lg(p)=1$, pour tout $p\in PROP$
$lg(\perp)=1$
Induction :
$lg(\neg A)=lg(A)+1$
$lg((A\wedge B))=lg(A)+lg(B)+3$ – 3 car il y a une « (», une «) » et un « $\wedge$ »
$lg((A\vee B))=lg(A)+lg(B)+3$
$lg((A\to B))=lg(A)+lg(B)+3$
NB : on peut regrouper les 3 derniers cas par : $lg((A*B))=lg(A)+lg(B)+3$ où $*\in \{\wedge,\vee,\to\}$

Définition du nombre de parenthèses nbp(A) d'une formule A de FORM :
Initialisation : $nbp(p)=0$, pour tout $p\in PROP$
$nbp(\perp)=0$
Induction :
$nbp(\neg A)=lg(A)+1$
$nbp ((A*B))= nbp (A)+ nbp (B)+2$ où $*\in \{\wedge,\vee,\to\}$

Définition du nombre de connecteurs (binaires) b(A) d'une formule A de FORMBIN :
Initialisation : $b(p)=0$, pour tout $p\in PROP$
$b(\perp)=0$
Induction :
$b ((A*B))= b (A)+ b (B)+1$ où $*\in \{\wedge,\vee,\to\}$

Définition de l'ensemble SF(A) des sous-formules d'une formule A de FORM :

Initialisation : $SF(p)=\{p\}$, pour tout $p\in PROP$

$SF(\perp)=\{\perp\}$

Induction :

$SF ((A*B))= \{(A*B)\}\cup SF(A)\cup SF(B)$ où $*\in\{\wedge,\vee,\rightarrow\}$

Preuves par induction : soit P une propriété, F un ensemble défini par induction, une façon de démontrer que tout élément de F vérifie la propriété P est la suivante : il faut démontrer que

i) D'une part, les éléments de F_0 vérifient P

ii) D'autre part, si les éléments de F_n vérifient P alors ceux de F_{n+1} aussi

Si i) et ii) sont vérifiés alors on peut en conclure que tout élément de F vérifie P par le principe d'induction (notez la ressemblance avec le principe de récurrence).

Dans le cas des formules, on peut être plus précis : pour prouver que toute formule A de FORM vérifie P, on procède ainsi :

i) pour tout $p\in PROP$, il faut montrer que P(p) est vérifié, ainsi que $P(\perp)$

ii) Hypothèses d'induction : pour tout A et B $\in$ FORM, P(A) et P(B) sont vérifiées,

Il faut démontrer que : $P(\neg A)$

$P((A\wedge B))$

$P((A\vee B))$

$P((A\rightarrow B))$

sont tous quatre vérifiés

Si i) et ii) sont démontrés alors on conclut par le principe d'induction que toute formule $A\in FORM$ vérifie P

Remarque : afin d'éviter des choses comme le paradoxe du sorite, il convient de s'entendre sur ce qu'on appelle une propriété. Pour l'instant, nous dirons que P est une propriété si et seulement l'ensemble {A/ P(A) est vérifié} est (informellement) défini.

Exemple très simple : nbp(A) mod 2 = 0 pour tout $A\in FORM$ (toute formule a un nombre pair de parenthèses).

i) pour tout $p\in PROP$: nbp(p) = 0 – par déf. de nbp – et 0 mod 2 = 0, idem pour $\perp$

ii) Hypothèses d'induction : nbp(A) mod 2 = 0 et nbp(B) mod 2 = 0

$nbp(\neg A) = nbp(A)$ – par déf. de nbp – et nbp(A) mod 2 = 0 par HI

nbp((A*B)) = nbp(A) + nbp(B) + 2 – par déf. de nbp, et nbp(A) mod 2 = nbp(B) mod 2 = 0 par HI

donc nbp(A) + nbp(B) + 2 est pair puisque somme de nombres pairs.

3 Arbre d'une formule : définitions informelle et inductive

L'arbre d'une formule est un arbre dont la racine contient une proposition (ou $\perp$) et a des sous-arbres vides, ou contient « le connecteur principal » et son/ses sous-arbres sont les arbres de ses sous-formules. Soit Conn l'ensemble des connecteurs alors les opérations de construction des arbres de formules sont :

$0 : \rightarrow$ Arb (arbre vide)

<_,_,_> : Arb x Conn x Arb (nouvel arbre construit)

et l'arbre d'une formule A, dénoté arb(A) est défini inductivement par :

$Arb (p)=<0,p,0>$

$Arb(\perp)=<0,\perp,0>$

$Arb(\neg B)=<Arb(B),\neg, 0>$

$Arb((B*C))=<Arb(B),*,Arb(C)>$ où $*\in\{ \vee \rightarrow \leftrightarrow ...\}$

4 Occurrences et substitutions

Occurrence dans une formule :

Une occurrence est mot du langage (1+2)* (présenté informellement en cours comme une suite de 1 et de 2, l'occurrence vide étant représentée par λ)

Cette suite de 1 et de 2 permet d'accéder à tout noeud de l'arbre d'une formule et donc à toute sous-formule, la suite vide (λ) donnant accès à la racine (Par défaut quand le fils est unique, il est à gauche).

Définition : Soit $k\in\{1,2\}$, alors $k\lambda=k$.

Définition (Concaténation) : Soit E un ensemble de suites de 1 et de 2, et soit $k\in\{1,2\}$, alors $k.E$ dénote l'ensemble des suites de E préfixées par k, i.e. $k.E=\{km \mid m\in E\}$. NB : $k.\{\lambda\} = k$, alors que $k.\emptyset = \emptyset$

Exemple : Si $E=\{\lambda, 22, 121\}$ et $k=2$, alors $k.E=\{2, 222, 2121\}$. NB : $k.(E\cup F) = k.E \cup k.F$

Définition (Ensemble des occurrences) : Soit A une formule, l'ensemble $Occ(A)$ des occurrences de A est défini inductivement par :
$Occ(p) = \{\lambda\}$; pour toute proposition $p\in PROP$
$Occ(\bot)= \{\lambda\}$;
$Occ((\neg A1)) = 1.Occ(A1) \cup\{\lambda\}$;
$Occ((A1 * A2)) = 2.Occ(A1) \cup 1.Occ(A2) \cup \{\lambda\}$.

Exemple : $Occ((q \wedge (p \rightarrow (\neg p))))$
$= 1.Occ(q) \cup 2.Occ(p \rightarrow (\neg p)) \cup \{\lambda\}$
$= 1.\{ \lambda\} \cup 2.(1.Occ(p) \cup 2.Occ(\neg p) \cup \{\lambda\}) \cup \{\lambda\}$
$= 1.\{ \lambda\} \cup 2.1.Occ(p)) \cup 2.2.(1.Occ(p) \cup \{\lambda\}) \cup \{2. \lambda\}) \cup \{\lambda\}$
$= \{1. \lambda\} \cup 2.1.\{ \lambda\} \cup 2.2.1. \{\lambda\} \cup \{2.2. \lambda\} \cup \{2. \lambda\} \cup \{\lambda\}$
$= \{1,21,221,22,2, \lambda\}$

Définition (Occurrence) : Soit A une formule, m une occurrence de A (i.e. $m\in Occ(A)$), l'occurrence m dans A, notée $A[m]$, est définie par :
$A1[\lambda] = A1$;
$(\neg A1)[1.m'] = A1[m']$, NB : $(\neg A)[2.m']$ n'est pas défini
$(A1 * A2)[1.m'] = A1[m']$; pour $*\in \{\rightarrow,\vee,\wedge\}$
$(A1 * A2)[2.m'] = A2[m']$; pour $*\in \{\rightarrow,\vee,\wedge\}$

Exemple : $(q \wedge (p \rightarrow (\neg p)))\ [22]$
$= (p \rightarrow (\neg p)))\ [2]$
$= (\neg p)\ [\lambda]$
$= (\neg p)$

Définition (Substitution d'une occurrence) : Soit A une formule, et m l'une de ses occurrence A (i.e. $m\in Occ(A)$) alors le résultat du remplacement de cette occurrence m dans A par B, noté $A[B/m]$ – on notera parfois $A[B/m]^1$ l'exposant 1 servant à rappeler que l'on ne substitue QU'UNE occurrence - est défini récursivement par :
$A1[B/\lambda]^1 = B$
$(\neg A1)[B/1.m']^1 = (\neg A1[B/m']^1)$;
$(A1 * A2)[B/1.m']^1 = (A1[B/m']^1 * A2)$; pour $*\in \{\rightarrow,\vee,\wedge\}$
$(A1 * A2)[B/2.m']^1 = (A1 * A2[B/m']^1)$; pour $*\in \{\rightarrow,\vee,\wedge\}$
Par convention, si $m\notin Occ(A)$ alors $A[B/m] = A$.

Exemple : $(q \wedge (p \rightarrow (\neg p)))\ [(r\vee s)/22]$
$= (q \wedge (p \rightarrow (\neg p))\ [(r\vee s)/2]\ )$
$= (q \wedge (p \rightarrow (\neg p)\ [(r\vee s)/ \lambda]))$
$= (q \wedge (p \rightarrow (r\vee s)\ ))$

Définition (Substitution de toutes les occurrences): Soient A, B, C trois formules, on note $A[B/C]$ le résultat de la substitution de TOUTES les occurrences de C dans A par B et on le définit par : autrement dit, si $m\in Occ(A)$ et si $A[m]=C$, alors $A[B/C][m] = B$ et $A[B/C][m] = A[m]$ sinon :
$p\ [B/C]=B$ si $C=p$ et p sinon (pour tout $p\in PROP$)
$\bot\ [B/C]=B$ si $C=\bot$ et $\bot$ sinon
$(\neg A1)[B/C]=B$ si $C = (\neg A1)$ et $(\neg A1[C/B])$ sinon
$(A1 * A2)[B/C]=B$ si $C= (A1 * A2)$ et $(A1[C/B] * A2[C/B])$ sinon

Exemple : $(q \wedge (p \rightarrow (\neg p)))\ [(r\vee s)/p]$
$= (q[(r\vee s)/p] \wedge (p \rightarrow (\neg p))\ [(r\vee s)/p])$
$= (q \wedge (p[(r\vee s)/p] \rightarrow (\neg p)\ [(r\vee s)/p]))$
$= (q \wedge ((r\vee s) \rightarrow (\neg p[(r\vee s)/p]))) = (q \wedge ((r\vee s) \rightarrow (\neg (r\vee s))))$

Chapitre 2 La théorie des modèles : sémantique de la logique des propositions

Postulats :

a) Bivalence — Choix de deux valeurs de vérité, ici 0 et 1
b) Vérifonctionnalité — La valeur de vérité d'une formule ne dépend QUE de celles des propositions qui la composent.

Donc la donnée des valeurs de vérité des propositions suffit pour déterminer celle de toute formule.

1 Les tables de vérité

On donne ici les tables de vérité des connecteurs $\wedge$, $\vee$, $\rightarrow$et $\neg$, et du symbole $\perp$, en insistant bien sur celle de $\rightarrow$

2 Formalisation des tables de vérité : interprétations / valuations et modèles

De fait, une ligne d'une table de vérité d'une formule A correspond à une situation possible c'est-à-dire à une valeur possible parmi {0,1} de chacune des propositions y apparaissant : c'est donc une fonction qui à chaque proposition associe une valeur de {0,1}. Une telle fonction est appelée une **valuation** (on dit aussi une **interprétation**), c'est donc une fonction de l'ensemble PROP des propositions dans l'ensemble {0,1}.
Pour pouvoir associer une valeur de vérité à une formule et non pas seulement aux propositions il faut de plus que v satisfasse les propriétés données dans la définition qui suit et qui reflètent les tables de vérité :

Définition 1 : Une valuation est une fonction de PROP dans {0,1} qui de plus satisfait les propriétés suivantes :

- $v[\perp]$ $=0$
- $v[\neg B]$ $=1-v[B]$
- $v[(B\wedge C)]$ $= \min(v[B] , v[C]))$ $= v[B] * v[C]$
- $v[(B\vee C)]$ $= \max(v[B] , v[C]))$ $= 1-((1-v[B]) * (1-v[C]))$
- $v[(B\rightarrow C)]$ $= \max(1-v[B],v[C])$
- $v[(B\leftrightarrow C)]$ $= v[(B\rightarrow C)\wedge (C\rightarrow B)]$ $= 1-\max(v[A]-v[B],v[B]-v[A])$

Définition 2 : Soit A une formule, H un ensemble de formules, et v une valuation,

- un **modèle** de A est une valuation v telle que v[A] = 1
  on dit que **v satisfait A**
- un **contre-modèle** de A est une valuation v telle que v[A] = 0
  on dit que **v falsifie A**
- un **modèle** de H est une valuation v telle que v[B] = 1 pour tout $B\in H$
  on dit que **v satisfait H**
- un **contre-modèle** de H est une valuation v telle que v[B] = 0 pour au moins un $B\in H$
  on dit que **v falsifie H**

exemple 1 Si l'on applique la valuation v telle que : v[p]=1, v[q]=0 et v[r]=1 à la formule $((p\wedge q)\vee\neg r)$, on obtient :

$v[((p\wedge q)\vee\neg r)] = \max(v[(p\wedge q) , v[(\neg r)]] = \max(v[p] * v[q] , 1-v[(r)]] = \max(1*0,1-1) = 0.$
Cette valuation v est donc un contre-modèle de $((p\wedge q)\vee\neg r)$.

Définition 3 : Soit A une formule,

- A est **valide** / est une **tautologie (**en notation **|=A)** ssi pour toute valuation v, on a v[A]=1
  Autrement dit, A est valide ssi toute valuation est un modèle de A
- A est **satisfiable ou cohérente** ssi il existe une valuation v telle que v[A]=1
- A est **insatisfiable ou contradictoire** ssi il n'existe pas de valuation v telle que v[A]=1
  Autrement dit, A est insatisfiable ssi toute valuation est un contre-modèle de A
- A est **invalide** ssi elle n'est pas valide
- Ces définitions s'étendent naturellement à un ensemble de formules.

Remarque : A est valide ssi $\neg$A est insatisfiable

Nous n'avons pour l'instant défini que la validité d'une conclusion seule. Mais à quelles conditions peut-on affirmer (dans le cadre de la logique des propositions bien entendu) qu'une conclusion C s'ensuit logiquement d'un ensemble d'hypothèses H ? C'est le cas seulement si dans chaque situation où les hypothèses sont vérifiées, la conclusion l'est aussi. En terme de modèles, on dira que C s'ensuit logiquement de l'ensemble H d'hypothèses si et seulement si tout modèle de chacune des hypothèses de H est aussi un modèle de C.

Définition 4 :

C est une conséquence logique de H={B1,B2,…,Bn} ssi tout modèle de H est aussi un modèle de C, et ceci est noté H |= C – ou B1,B2,…,Bn |= C.

Remarque : Notons le cas particulier où H est vide : que signifie ∅ |= C ? Littéralement, d'après la définition donnée, si v est un modèle de ∅ (ce qui est toujours le cas, sinon il y aurait une formule de ∅ dont v ne serait pas un modèle ce qui n'est pas possible) alors v est un modèle de C ; autrement dit, toute valuation v est un modèle de C, et donc C est valide. Par conséquent, ∅ |= C signifie la même chose que |= C, ce qui justifie a posteriori cette notation.

Définition 5 :

C est logiquement équivalente à B ssi tout modèle de B est aussi un modèle de C, et réciproquement. Ceci est noté C ≡ B.

Les symboles |= et ≡ **appartiennent au métalangage mathématique** et ne font pas partie de l'alphabet des formules !! |= exprime une relation entre un ensemble de formules et une formule, et ≡ met en relation deux formules. Néanmoins, ces symboles mathématiques sont liés avec les connecteurs de l'implication et de l'équivalence :

Théorème 1 : A|=B ssi |=A→B (ce théorème est appelé théorème de la déduction) et A≡B ssi |= (A↔B).

3 Quelques équivalences remarquables

Soient A, B et C trois formules quelconques :

(A→A) ≡ T	-	(A↔A) ≡ T	-
(A∨¬A) ≡ T	Tiers-exclus	¬(A∧¬A) ≡ T	Non-contradiction
A ≡ ¬¬A	-	-	-
A ≡ (A∨A)	Idempotence du ∨	A≡(A∧A)	Idempotence du ∧
(A∨B) ≡ (B∨A)	Commutativité du ∨	(A∧B) ≡ (B∧A)	Commutativité du ∧
((A∨B)∨C) ≡ (A∨(B∨C))	Associativité du ∨	((A∧B) ∧C) ≡ (A∧ (B∧C))	Associativité du ∧
((A∨B)∧C) ≡ ((A∧C)∨(B∧C))	Distributivité de ∨ sur ∧	((A∧B)∨C) ≡ ((A∨C) ∧ (B∨C))	Distributivité de ∧ sur ∨
((A↔B) ≡ (B↔A))	Commutativité de ↔	Le → n'est pas commutatif ! !	-
(((A↔B) ↔C) ≡ (A↔ (B↔C)))	Associativité de ↔	Le → n'est pas associatif ! !	-
(A→(B→C)) ≡ ((A→B)→(A→C))	Auto-distributivité de →	(A→(B→C)) ≡ (B→(A→C))	Permutabilité de →
(A→(B→C)) ≡ ((A∧B)→C)	Import-export	(((A→B)∧(B→C))→(A→C))	Transitivité de →
(A∨⊥) ≡ A	Élément neutre du ∨	(A∧T) ≡ A	Élément neutre du ∧
(A∨ T) ≡ T	Élément absorbant du ∨	(A∧⊥) ≡ ⊥	Élément absorbant du ∧
(A→B) ≡ (¬B→¬A)	Contraposition	(A↔B) ≡ (¬A↔¬B)	Contraposition
((A→B)∧(A→¬B)) ≡ ¬A	Absurde	(A→¬A) ≡ ¬A	Absurde
	rapports entre		**rapports entre**
(A→B) ≡ (¬A∨B)	→ et ∨	(A→B) ≡ ¬(A∧¬B)	→ et ∧
(A∨B) ≡ ¬(¬A∧¬B)	∨ et ∧	(A∧B) ≡ ¬(¬A∨¬B)	∧ et ∨
(A⊕B) ≡ ¬(A↔B)	⊕ et ↔	(A⊕B) ≡ ((A∧¬B)∨(¬A∧B))	⊕ et ∧ / ∨
Lois de simplification :			
(A↔⊥) ≡ ¬A	(A↔T) ≡ A		
(A→⊥) ≡ ¬A	(A→T) ≡ T	(⊥→A) ≡ T	(T → A) ≡ A
Divers :			
((A∨B)→C) ≡ ((A→C)∧(B→C))	(A∧(A∨B)) ≡ A	(A∧(A→B)) ≡ (A∧B)	
(A→(B∧C)) ≡ ((A→B)∧(A→C))	(A∨ (A∧B)) ≡ A		

Ce sont les lois de De Morgan qui, à cause des rôles symétriques joués par le $\vee$ et le $\wedge$ ont consacré le choix du $\vee$ (« ou inclusif ») comme connecteur principal au lieu du $\oplus$ (« ou exclusif »).
La dualité entre le $\vee$ et le $\wedge$ s'exprime plus généralement dans le théorème suivant :

Théorème 2 : Si A est une formule ne contenant que des $\wedge$, des $\vee$ et des $\neg$, et que A' résulte de A en remplaçant dans A les $\wedge$ par des $\vee$, les $\vee$ par des $\wedge$, et toute proposition p par sa négation $\neg$p alors $\models$A ssi $\models\neg$A'.

Les tautologies rassemblées ci-dessus permettent de simplifier une formule en vertu du fait que l'on peut au sein d'une formule A, remplacer une sous-formule par une formule équivalente, et obtenir une formule équivalente à A. C'est l'objet du théorème :

Définition 6 :

Rappel : soit A une formule, B l'une de ses sous-formules, d'occurrence m et C une autre formule alors

- **A[C/m]** désigne la formule obtenue en remplaçant l'occurrence m de B dans A par C, et
- **A[C/B]** désigne la formule obtenue en remplaçant **toutes les** occurrences de B par C dans A.

Théorème 3 : Soit A, B, C trois formules telles que B≡C, et m une occurrence de A, alors $\models$ A $\leftrightarrow$ A[C/m], et donc bien sûr $\models$ A $\leftrightarrow$ A[C/B]. Ce théorème est appelé théorème du remplacement des équivalents.

Démonstration :
1er cas : m$\notin$ Occ(A) en ce cas A[C/B] = A et le théorème est trivialement vérifié.
2ème cas : m$\in$ Occ(A) et on procède par **induction sur m :**
Initialisation de l'induction :
m = λ : alors, par définition des occurrences et substitutions, on a A=A[λ]=B et A[C/λ]=C, donc A $\equiv$ A[C/m]
Etape d'induction :
1) m = 1m' : ceci implique que

i) A = ($\neg$A1), dans cas, A[C/m] = ($\neg$A1) [C/1m'] = A1[C/m'], et par HI A1 $\equiv$ A1[C/m'], et il est facile de vérifier que ceci entraîne ($\neg$A1) $\equiv$ ($\neg$A1[C/m']), ce qui entraîne à son tour ($\neg$A1) $\equiv$ ($\neg$A1) [C/1m'].

ou que

ii) A = (A1*A2), dans ce cas, A[C/m] = (A1*A2) [C/1m'] = A1[C/m'], et par HI A1 $\equiv$ A1[C/m'], et il est facile de vérifier que ceci entraîne (A1*A21) $\equiv$ (A1[C/m']*A2), qui entraîne (A1*A21) $\equiv$ (A1[C/m']*A2), qui, finalement implique (A1*A2) $\equiv$ (A1*A2) [C/1m'].

1) m = 2m' : ce cas se traite comme le 1.ii) ci-dessus.

Chapitre 3 La méthode des tableaux

Pour déterminer si une formule donnée est ou non une tautologie, ou pour déterminer si elle est ou non conséquence logique d'un ensemble de formules, on peut utiliser les tables de vérité mais cette méthode a l'inconvénient d'être exhaustive : c'est-à-dire que l'on vérifie *toutes* les valuations possibles, et pour une formule contenant *n* propositions, la table correspondante aura 2^n lignes. Dans la présente section, nous allons présenter une méthode dont le principe est de chercher à construire **un** modèle, et n'est donc pas exhaustive. Avec cette méthode – dite **méthode des tableaux** – pour déterminer si une formule A est une tautologie, on cherchera un modèle de sa négation ($\neg$A) – et donc un contre-modèle de A – si on en trouve un c'est que A n'est pas une tautologie. Cette méthode, aussi appelée méthode des arbres, est donc une méthode par l'absurde : on suppose que ($\neg$A) est satisfiable et on cherche à aboutir à une contradiction (si A est valide).

1 Présentation informelle de la méthode des tableaux

Cas d'une formule valide : on veut savoir si la formule A0=((p$\to$q)$\to$($\neg$q$\to\neg$p)) est valide (c'est-à-dire v[A0]=1 pour toute valuation v). On procède par l'absurde : supposons que sa négation $\neg$A0=$\neg$((p$\to$q)$\to$($\neg$q$\to\neg$p)) admette un modèle v0 (c'est-à-dire que v0[$\neg$A0]=1 et donc v[A0]=0)

i) pour que v0[$\neg$((p$\to$q)$\to$($\neg$q$\to\neg$p))]=1 il faudrait que
ii) v0[(p$\to$q)]=1, et que v0[($\neg$q$\to\neg$p)]=0 par définition des valuations, et pour cela il faudrait que
iii) v0[(p$\to$q)]=1, et que v0[$\neg$q]=1 et v0[$\neg$p]=0, et donc v0[p]=1, c'est-à-dire que
iv) v0[(p$\to$q)]=1, v0[q]=0 et v0[p]=1, donc par définition, il faudrait
v) 1-v0[p]=1 ou v0[q] = 1, et de plus v0[q]=0 et v0[p]=1, et donc :
   a) ou bien 1-v0[p]=1 et donc v0[p]=0, et de plus v0[q]=0 et v0[p]=1, **ce qui n'est pas possible**
   b) ou bien v0[q]=1, et de plus v0[q]=0 et v0[p]=1, **ce qui n'est pas possible non plus.**

En conclusion, il est absurde de supposer que $\neg A0$ admet un modèle, et donc A0 est valide.

Cas d'une formule invalide : on veut savoir si la formule $B0=((p\rightarrow q)\rightarrow \neg p)$ est. On procède par l'absurde : supposons que sa négation $\neg B0=\neg((p\rightarrow q)\rightarrow \neg p)$ admette un modèle v1 :

i) pour que $v1[\neg((p\rightarrow q)\rightarrow \neg p)]=1$, il faudrait que
ii) $v1[(p\rightarrow q)]=1$, et que $v1[\neg p]=0$ par définition des valuations, et pour cela il faudrait que
iii) $v1[(p\rightarrow q)]=1$, et que $v1[p]=1$, et pour cela il faudrait que
iv) $v1[p]=0$ ou $v1[q]=1$, et que de plus $v1[p]=1$, et donc :
a) ou bien, $v1[p]=0$, et $v1[p]=1$, **ce qui n'est pas possible**,
b) ou bien $v1[q]=1$, et $v1[p]=1$ **ce qui est possible**, et comme il n'y a pas d'autres contraintes on peut en conclure que $\neg B0$ est satisfiable.

En conclusion, $\neg B0$ admet un modèle, et donc B0 n'est pas valide, v1 (cas b) en est un contre-modèle.

2 Présentation « formelle » de la méthode des tableaux

Où l'on va préciser comment formaliser la démarche décrite dans le paragraphe précédent. La méthode consiste à construire un arbre, dans lequel chaque niveau représente l'une des étapes i), ii), iii), … ci-dessus. L'algorithme ci-dessous décrit la construction de cet arbre pour une formule quelconque C :

Initialisation : on commence avec un arbre réduit à un seul nœud, ne contenant que la formule C

Tant que c'est possible appliquer l'un des 3 points ci-dessous :

Point 1 : si le long d'une branche de l'arbre en cours de construction il y a un nœud contenant $\perp$, cette branche sera dite *fermée* (on la marque d'une croix X à son extrémité)

Point 2 : si toutes les branches sont fermées STOP

Point 3 : choisir un nœud **n** dans une branche non fermée et contenant une formule C non encore examinée, et, en chaque feuille dont **n** est ancêtre, créer un (ou plusieurs) nouveau nœud-fils contenant les formules issues de l'application d'une des règles ci-dessous.

Alors ou bien toutes les branches sont fermées et le tableau est dit **fermé**, ou il ne l'est pas et il est dit **ouvert**.

Fin de l'algorithme .

A la suite de cet algorithme,

- si le tableau est fermé c'est que la formule de départ est insatisfiable,
- sinon, il est ouvert, et la formule initiale est satisfiable.

Pour obtenir un modèle de cette formule, il suffit de relever les atomes le long d'une branche ouverte et de définir la valuation v telle que : $v[p]=1$ si p est présent le long de cette branche, et $v[p]=0$ si $\neg p$ est présent, et $v[p]$ quelconque sinon.

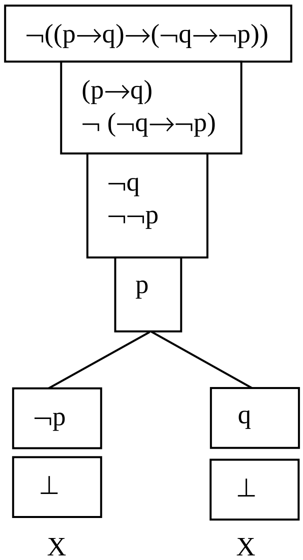
3 Les règles des tableaux

Règle	Un noeud contient	Faire
Fermeture	$\perp$	Fermer la branche contenant ce nœud
Règle	**Une branche contient**	**Créer (à la feuille de cette branche)**
$\perp$	A et $\neg A$	Un fils contenant $\perp$
Règle	**Un nœud contient**	**Créer (à CHAQUE feuille de CHAQUE branche (non fermée) passant par ce nœud)**
$\wedge$	$(A\wedge B)$	Un fils contenant A et B
$\neg\vee$	$\neg(A\vee B)$	Un fils contenant $\neg A$ et $\neg B$
$\neg\rightarrow$	$\neg(A\rightarrow B)$	Un fils contenant A et $\neg B$
$\leftrightarrow$	$(A\leftrightarrow B)$	Un fils contenant $(A\rightarrow B)$ et $(B\rightarrow A)$
$\neg\neg$	$\neg\neg A$	Un fils contenant A
$\vee$	$(A\vee B)$	Deux fils contenant l'un A et l'autre B
$\neg\wedge$	$\neg(A\wedge B)$	Deux fils contenant l'un $\neg A$ et l'autre $\neg B$
$\rightarrow$	$(A\rightarrow B)$	Deux fils contenant l'un $\neg A$ et l'autre B
$\neg\leftrightarrow$	$\neg (A\leftrightarrow B)$	Deux fils contenant l'un $\neg(A\rightarrow B)$ et l'autre $\neg(B\rightarrow A)$

Remarque importante : les 6 premières règles ne créant pas d'embranchement, il est préférable de les appliquer en priorité sur celles qui en créent un, de façon à limiter la taille de l'arbre obtenu.

Reprise des exemples informels

Pour savoir si A0 est valide, construisons le tableau de ¬A0 :



⇒ règle ¬→

⇒ règle ¬→

⇒ règle ¬¬

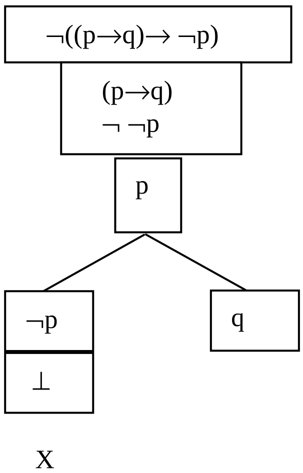
⇒ règle →

⇒ règle ⊥, deux fois

Toutes les branches sont fermées : Stop.

Toutes les branches sont fermées, le tableau est fermé, la formule initiale ¬((p→q)→(¬q→¬p)) est donc insatisfiable, par conséquent, A0=((p→q)→(¬q→¬p)) est valide.

Pour savoir si B0 est valide, construisons le tableau de ¬B0 :



⇒ règle ¬→

⇒ règle ¬¬

⇒ règle →

⇒ règle ⊥,

Il n'y a plus rien à faire. Stop.

Une seule branche est fermée, le tableau est ouvert, la formule initiale ¬((p→q)→¬p) est donc satisfiable, par conséquent, ((p→q)→ ¬p) est invalide, un contre-modèle est fourni par v telle que :

- **v[p]=1**
- **v[q]=1**

Récapitulation : **|= A ssi**
¬A est insatisfiable ssi
le tableau de A est fermé

Un lien utile : http://www.umsu.de/logik/trees/ il s'agit d'une applet qui implémente la méthode des tableaux.

4 Présentation formelle de la méthode des tableaux

Définition :

- Une *branche* est un ensemble de formules.
- Un *tableau intermédiaire* est un ensemble de branches.
- Un *tableau (terminé)* est le dernier d'une suite de tableaux intermédiaires, suite obtenue par application de règles de réécriture sur les branches du tableau intermédiaire précédent, le premier élément de la suite étant un tableau intermédiaire initial.
- Une branche est *fermée* ssi elle contient la formule $\perp$, elle est *ouverte* sinon.
- Un tableau (terminé) est *fermé* ssi toutes ses branches le sont, il est *ouvert* sinon.

Soit H un ensemble de formules, le tableau de H, noté T(H), est défini comme suit :

On calcule $T_{i+1}(H)$ à partir de $T_i(H)$ en appliquant de manière non déterministe l'une des règles de réécriture ci-dessous sur une branche de $T_i(H)$ (on suppose que $T_i(H) = \{Br_1,\ldots,Br_n\}$). Chacune de ces règles prend en entrée une branche de $T_i(H)$ et « renvoie » un ensemble de branches (à cause des disjonctions).
NB : Par commodité on adoptera les notations suivantes :

- B_k, X,Y pour signifier que les formules X etY sont éléments de Br_k
- B_k+X+Y au lieu de $B_k \cup \{X,Y\}$
- $B_k \Rightarrow B_k'$ pour signifier que B_k se réécrit en B_k'

(Règle $\perp$)	B_k , $X,\neg X$	$\Rightarrow$	$\{B_k + \perp\}$ – on ajoute $\perp$ à B_k si elle contient X et $\neg$X
(Règle $\neg$)	B_k , $\neg\neg X$	$\Rightarrow$	$\{B_k + X\}$
	-- règles conjonctives		
(Règle $\wedge$)	B_k , $X\wedge Y$	$\Rightarrow$	$\{B_k + X + Y\}$
(Règle$\neg\vee$)	B_k , $\neg(X\vee Y)$	$\Rightarrow$	$\{B_k + \neg X + \neg Y\}$
(Règle$\neg\rightarrow$)	B_k , $\neg(X\rightarrow Y)$	$\Rightarrow$	$\{B_k + X + \neg Y\}$
(Règle$\leftrightarrow$)	B_k , $X\leftrightarrow Y$	$\Rightarrow$	$\{B_k + (X\rightarrow Y) + (Y\rightarrow X)\}$
	-- règles disjonctives		
(Règle$\vee$)	B_k , $(X\vee Y)$	$\Rightarrow$	$\{B_k + X , B_k + Y\}$
(Règle$\neg\vee$)	B_k , $\neg(X\vee Y)$	$\Rightarrow$	$\{B_k + \neg X , B_k + \neg Y\}$
(Règle $\rightarrow$)	B_k , $(X\rightarrow Y)$	$\Rightarrow$	$\{B_k + \neg X , B_k + Y\}$
(Règle $\neg\leftrightarrow$)	B_k , $\neg(X\leftrightarrow Y)$	$\Rightarrow$	$\{B_k + \neg (X\rightarrow Y) , B_k + (X\rightarrow Y)\}$

L'algorithme complet est :

Initialisation : $T_0(H) = \{H\}$ – tableau constitué d'une seule branche,
Induction :
  Tant que $T_0(H)$ n'est pas terminé **Faire**
  -- Calcul de de $T_{i+1}(H)$
  Si une branche B ouverte de $T_i(H)$ se réécrit en E par application de la règle * parmi les règles ci-dessus (règle choisie de manière non déterministe)
    Alors $T_{i+1}(H) = T_i(H) \setminus B \cup E$ (on « met à jour » la branche B)
    -- sinon ne rien faire
  Si $T_i(H) = T_{i+1}(H)$
    Alors $T_{i+1}(H)$ est terminé
  Fin tant que ;
    --$T_i(H)$ est terminé, et est soit ouvert, soit fermé.

Propriétés de l'algorithme de calcul des tableaux

Théorème : l'algorithme ci-dessus termine (i.e. il existe $0 \leq k$ tel que $T_k(H)$ est terminé) et fournit en sortie le tableau (terminé) de H noté $T(H)$.

Théorème (adéquation) : si $T(H)$ est fermé, alors H est insatisfiable.

Lemme: si B est une branche ouverte de $T(H)$ alors la valuation v telle que $v(p) = 1$ ssi $p \in B$ est un modèle de H.

Théorème (Complétude, conséquence du lemme) : si $T(H)$ ouvert, alors H est satisfiable.

Corollaire : Une formule A est valide si et seulement si $T(\{\neg A\})$ de A est fermé.
Corollaire : Une formule A est conséquence logique de H si et seulement si $T(H \cup \{\neg A\})$ est fermé.

Corollaire :Le problème de la satisfiabilité d'un ensemble de formules de la logique des propositions est décidabe.

NB : en pratique, puisqu'on représentera les branches par des suites ordonnées et les tableaux (intermédiaires ou terminés) par des arbres constitués de ces branches, on n'aura pas besoin de supprimer B dans T_i, il suffira d'étendre par un ou deux successeurs la branche B. De plus, on rayera les formules déjà traitées.

Reprise des exemples informels

Pour savoir si A0 est valide, construisons le tableau de $\neg$A0 :

$T_0(A0) = \{ \{\neg((p\to q)\to(\neg q\to\neg p)) \} \}$

(règle $\neg\to$) $T_1(A0) = \{ \{ \cancel{\neg((p\to q)\to(\neg q\to\neg p))}, (p\to q), \neg(\neg q\to\neg p) \} \}$

(règle $\neg\to$) $T_2(A0) = \{ \{\cancel{\neg((p\to q)\to(\neg q\to\neg p))}, (p\to q), \cancel{\neg(\neg q\to\neg p)}, \neg q, \neg\neg p\} \}$

(règle $\neg\neg$) $T_3(A0) = \{ \{\cancel{\neg((p\to q)\to(\neg q\to\neg p))}, (p\to q), \cancel{\neg(\neg q\to\neg p)}, \neg q, \cancel{\neg\neg p}, p\} \}$

(règle $\to$) $T_4(A0) = \{ \{\cancel{\neg((p\to q)\to(\neg q\to\neg p))}, \cancel{(p\to q)}, \cancel{\neg(\neg q\to\neg p)}, \neg q, \cancel{\neg\neg p}, p, \neg p\},$
$\{\cancel{\neg((p\to q)\to(\neg q\to\neg p))}, \cancel{(p\to q)}, \cancel{\neg(\neg q\to\neg p)}, \neg q, \cancel{\neg\neg p}, p, q\}\}$

(règle $\perp$) $T_5(A0) = \{ \{\cancel{\neg((p\to q)\to(\neg q\to\neg p))}, \cancel{(p\to q)}, \cancel{\neg(\neg q\to\neg p)}, \neg q, \cancel{\neg\neg p}, p, \neg p, \perp\},$
$\{\cancel{\neg((p\to q)\to(\neg q\to\neg p))}, \cancel{(p\to q)}, \cancel{\neg(\neg q\to\neg p)}, \neg q, \cancel{\neg\neg p}, p, q, \perp\} \}$

Fin

(Toute nouvelle application de règle porterait sur une formule déjà traitée, et on constaterait bien un point fixe : en particulier, on aurait $T_6(A0) = T_5(A0)$).

T(A0) est fermé, A0 est donc insatisfiable.

Chapitre 4 La théorie de la preuve : déduction naturelle

1 Propriétés de la conséquence logique

On va ici, examiner quelques propriétés remarquables de la conséquence logique. Ces propriétés seront la source d'une autre approche à la question de la validité d'un raisonnement, approche qui sera systématisée dans la section « Théorie de la preuve ». Ces propriétés s'énoncent sous la forme de théorèmes dont on ne prouvera que certains.

Théorème 4

- Si $H \models A$ et $H \subseteq H'$ alors $H' \models A$ (propriété de monotonie)
  Preuve : tout modèle de H' est nécessairement un modèle de H (puisque, par hypothèse, H contient moins de formules à satisfaire) et tout modèle de H est un modèle de A (par hypothèse), donc tout modèle de H' est un modèle de A.
- Si $\models A$ alors $H \models A$ (cas particulier de la monotonie)
- $B_1,\ldots,B_n \models A$ ssi $(B_1\wedge\ldots\wedge B_n) \models A$
- si $\models A$ et $\models A \rightarrow B$ alors $\models B$ (théorème du *modus ponens*)
- $H,B \models A$ ssi $H \models B \rightarrow A$ (Théorème de la déduction)
- $H,A \models A$ (réflexivité)
  Preuve : on sait que $\models A\rightarrow A$, et donc $H \models A\rightarrow A$ (monotonie), et donc $H,A \models A$ (déduction).
- Si $H \models B$ et $H',B \models C$ alors $H,H' \models C$ (transitivité)
  Preuve : Hypothèse 1 : tout modèle de H est un modèle de B
  Donc, tout modèle de $H\cup H'$ est un modèle de $H'\cup B$
  Hypothèse 2 : tout modèle de $H'\cup B$ est un modèle de C
  Donc, tout modèle de $H\cup H'$ est un modèle de C.
- Si $A \models B$ et $B \models C$ alors $A \models C$ (cas particulier de la transitivité)
- $H \models \neg A$ ssi $H,A \models \perp$ (raisonnement par l'absurde)
  Commentaire : Attention, le sens de $H, A \models \perp$ est que tout modèle de H,A est un modèle de $\perp$. Or il n'existe pas de modèle de $\perp$ (sinon il devrait attribuer la valeur 1 à $\perp$, ce qui irait à l'encontre de la définition des valuations). Donc ceci signifie que H,A n'admet aucun modèle.
  Preuve : si $H, A \models \perp$ alors H,A n'admet aucun modèle, et donc pour toute valuation v : $v[A]=0$ ou il existe une formule $B \in H$ telle que $v[B]=0$, ou ce qui revient au même : $v[\neg A]=1$ ou il existe une formule $B \in H$ telle que $v[B]=0$.
  Par ailleurs, $H \models \neg A$ revient à : pour toute valuation v, ou il existe une formule $B \in H$ telle que $v[B]=0$ (et donc v n'est pas un modèle de H), ou $v[\neg A]=1$.
  Il y a donc bien équivalence.
- $H \models A$ et $H \models \neg A$ ssi $H \models \perp$ (principe de contradiction)
- $H,A,A\rightarrow B \models C$ ssi $H,A,B \models C$
- Si $H,A \models C$ et $H,B \models C$ alors $H,A\vee B \models C$ (analyse des cas)

Ces propriétés (parmi beaucoup d'autres) vont nous permettre de nous approcher du raisonnement sous hypothèses : en effet, il devient possible de prouver que $H \models A$ sans jamais faire appel aux valuations mais en faisant seulement appel aux manipulations formelles que permettent ces propriétés.

2 Présentation informelle

Grâce à la notion de conséquence logique, nous pouvons d'ores déjà formaliser les questions du type « S'ensuit-il que C sous les hypothèses A et B ? » par « C est-elle une conséquence logique de A et B (c-à-d $A,B \models C$) ? ». Mais comme nous l'avons souligné dans le cadre de la méthode des tableaux, nous ne disposons que de moyens permettant une réponse oui/non, alors que l'on souhaiterait parfois avoir une « trace » du raisonnement effectué (cela est encore plus crucial pour la logique des prédicats où il n'existe pas de méthode permettant à coup sûr de savoir si oui ou non une formule est conséquence logique d'un ensemble d'hypothèses). Prenons un exemple pour éclaircir notre propos :
Le raisonnement suivant est-il valide ?

- Quand il neige, il fait froid
- Quand il y a du verglas, il fait froid
- Or (en ce moment) il neige ou il y a du verglas (ou les deux)

- De plus, en été il ne fait pas froid
- Donc on n'est pas en été

Si l'on suppose que :

- « Il neige» est représentée par n, que
- « Il y a du verglas» est représentée par v, que
- « Il fait froid » est représentée par f, et que
- « On est en été » est représenté par é

Cela revient à se demander si $\neg$é est conséquence logique des hypothèses n$\rightarrow$f (Hyp 1) et v$\rightarrow$f (Hyp 2), n$\vee$v (Hyp 3) et é$\rightarrow\neg$f (Hyp 4), c-à-d :

- n$\rightarrow$f , v$\rightarrow$f, n$\vee$v, é$\rightarrow\neg$f |= $\neg$é ?

La méthode des tableaux permet de répondre par l'affirmative, mais ne permet pas de représenter le raisonnement suivant :

- Puisque (Hyp 3) il neige ou il y a du verglas, examinons les deux cas possibles :
  - a: Il neige (Hyp 5) - Hypothèse formulée temporairement
    Or, quand il neige, il fait froid (Hyp 1)
    Donc d'après (1) et (5), il fait froid dans ce cas
  - b: Il y a du verglas (Hyp 6) - Hypothèse formulée temporairement
    Or, quand il y a du verglas, il fait froid (Hyp 2)
    Donc d'après (2) et (6), il fait froid dans ce cas aussi
- Dans les deux cas, il fait froid.
- Maintenant, supposons qu'on soit en été (Hyp 7) - Hypothèse formulée temporairement
- Si on est en été, il ne fait pas froid (Hyp 4)
- Donc, d'après (7) et (4), il ne fait pas froid
  ce qui est en contradiction avec : il fait froid, obtenu plus haut
  Autrement dit, sous les hypothèses 1, 2, 3, 4 et 7 on aboutit à une contradiction.
- Par conséquent, avec les hypothèses 1, 2, 3 et 4, l'hypothèse 7 ne peut être soutenue car elle mène à une contradiction. On en conclut donc le contraire : on est pas en été.

La théorie de la preuve s'intéresse tout particulièrement à cet aspect des choses : formuler un cadre formel dans lequel appréhender la structure du raisonnement, et non simplement la réponse oui/non à sa validité ; autrement dit, si l'on considère que les hypothèses et la conclusion sont les points de départ et d'arrivée d'un trajet possible (si la conclusion est conséquence logique des hypothèses) ; pendant que la méthode des tableaux s'intéresse seulement à la question de savoir s'il existe un chemin qui va de l'un à l'autre, la théorie de la preuve s'intéresse, elle, au chemin lui-même.

Remarque :

Pourquoi appeler de la même manière (en l'occurrence hypothèses) les phrases « Quand il neige, il fait froid » et « Il neige ou il y a du verglas » alors que la première semble énoncer une loi générale (toujours vraie) et la seconde ne fait que constater un fait (faux aujourd'hui, vrai demain) ? Rappelons que nous avons distinguées les vérités factuelles et les vérités logiques : ces dernières sont vraies mêmes dans les univers imaginaires, dans toutes les situations imaginables. Par contre, on serait certainement très étonné qu'il se mette à neiger alors qu'il fait chaud, et on se dirait qu'il y a quelque chose qui cloche MAIS on peut l'imaginer. Du point de vue strictement logique, les lois générales et les faits jouent le même rôle : ce sont des hypothèses. La logique ne s'occupe pas de savoir si elle sont vraies ou fausses mais seulement de savoir ce qui se passe si on les suppose vraies.

Néanmoins, on verra que l'on peut faire une distinction : les lois générales comme « Quand il neige, il fait froid » peuvent être regroupées ensembles avec les tautologies et forment alors ce qu'on appelle une **théorie** (ici, une théorie « météorologique »). Seules les hypothèses qui ne font pas partie de la théorie seront appelées hypothèses : en pratique, seules celles-ci seront éventuellement remises en question en cas de contradiction, ainsi, dans l'exemple ci-dessus, on ne remettra pas en cause les phrases comme « Quand il neige, il fait froid » aussi facilement que « Il neige ou il y a du verglas ». Du point de vue météorologique, cette dernière n'est pas toujours vraie. Du point de vue logique aucune n'est plus vraie que l'autre. Une théorie est donc constituée d'**axiomes logiques** (les tautologies) et d'**axiomes non-logiques** que l'on tient pour toujours vrais non pas pour des raisons logiques mais soit parce qu'ils expriment des définitions (« L'énergie mécanique est la somme de l'énergie potentielle et de l'énergie cinétique » ou « Un nombre premier est un nombre qui n'est divisible que par 1 et par lui-même »), soit parce qu'ils correspondent à ce que l'on observe (« Quand il neige, il fait froid » ou « L'énergie mécanique se conserve »), soit encore parce qu'ils expriment l'idée que l'on se fait de concepts (« Par deux points, il passe une droite »). Les formules que l'on peut déduire à partir de ces axiomes s'appellent des **théorèmes** de cette théorie (« Il y a une infinité de nombre premier », « En été, il n'y a ni neige, ni verglas »).

Si l'on reprend maintenant le raisonnement ci-dessus, en remplaçant les propositions (comme « Il neige ») par des lettres propositionnelles (comme n) afin de s'abstraire du sens des énoncés, en numérotant les lignes et en faisant référence à ces numéros pour justifier le raisonnement, on obtient :

1.	$n \vee v$	(Hyp 3)	
2.		n	(Hyp 5) – hypothèse temporaire
3.		$n \to f$	(Hyp 1)
4.		f	(d'après les lignes 2 et 3)
5.		v	(Hyp 6) – hypothèse temporaire
6.		$v \to f$	(Hyp 2)
7.		f	(d'après les lignes 5 et 6)
8.	f	(puisque sous les deux cas – lignes 2 et 5, on arrive à la même conclusion – lignes 4 et 7. Cette conclusion intermédiaire ne dépend plus des hypothèses 4 et 5 mais seulement de 1, 2 et 3.)	
9.		é	(Hyp 7) - hypothèse temporaire
10.		$\text{é} \to \neg f$	(Hyp 4)
11.		$\neg f$	(d'après les lignes 10 et 11)
12.		$\bot$	(d'après les lignes 8 et 11) On aboutit à une contradiction
13.	$\neg \text{é}$	Puisque l'hypothèse de la ligne 9 mène à une contradiction, on considère que sa négation est prouvée	

Ainsi, en ayant besoin uniquement des hypothèses 1, 2, 3 et 4 (les autres n'apparaissant que temporairement dans le raisonnement), on prouve que s'ensuit la conclusion. Si on utilise une hypothèse temporaire (p.ex. Hyp 7), au moment où l'on sort de la partie du raisonnement où elle est utilisée (les lignes 9 à 12 pour Hyp 7) , on dit que l'on **décharge** l'hypothèse (on lui enlève sa charge d'hypothèse).
Ce que nous appelerons une **preuve** (ou encore une **démonstration)** est précisément constitué par le « texte » des lignes 1 à 13, présentée de cette façon (avec des lignes numérotées, …) ou d'une autre : nous choisirons une présentation plus visuelle basée sur des arbres.
La déduction naturelle que nous allons présenter ci-dessous, se veut être une formalisation du raisonnement lui-même et non plus seulement de sa validité, dans l'esprit de ce qui a été montré dans la section précédente. C'est pour des raisons de lisibilité que plutôt qu'une présentation avec des lignes numérotées, nous utiliserons des arbres.

Définition 7 : Une preuve en déduction naturelle (en abrégé DN) d'une formule A à partir des hypothèses H1, …, Hn est un arbre dont :

- les feuilles sont les Hi ou des hypothèses temporaires qui ont été **déchargées** au moins une fois
- la racine contient A (la formule à prouver)
- les nœuds contiennent des formules obtenues par application de règles (voir plus loin) sur des nœuds déjà obtenus

Définition 8 : Le fait qu'il existe une preuve en déduction naturelle d'une formule A à partir des hypothèses (dites hypothèses principales) H1, …, Hn sera noté H1,…, Hn |- A. On dira que A est ***déductible*** *de l'ensemble { H1, …, Hn }. On dira parfois que A est un* ***théorème*** *de la théorie H. Dans le cas particulier où H est vide, on dira que A est un* ***théorème*** *(de la logique des propositions).*

Les règles permettent de fixer les règles implicitement utilisée dans le raisonnement ci-dessus (des lignes 14 : é et 15 : $\text{é} \to \neg f$ on s'autorise à conclure la ligne 16 : $\neg f$, p.ex.).
Pour chaque connecteur (y compris $\bot$), on va préciser une règle **d'introduction** qui spécifie à quelles conditions faire apparaître le connecteur, et une règle **d'élimination** qui spécifie à quelles conditions faire disparaître le connecteur (dans l'exemple des lignes 14, 15 et 16, on élimine le connecteur $\to$).

Les noms des règles sont abrégés ainsi : (E$\rightarrow$) désigne la règle pour l'élimination (E) du connecteur $\rightarrow$, (I$\vee$) désigne la règle pour l'introduction (I) du connecteur $\vee$…

3 Les règles de déduction naturelle

Nous utiliserons les conventions graphiques suivantes où « nœud » signifie aussi bien « nœud interne » que « feuille ».

1.
$$\frac{X}{Y}$$
Qui se lit « si un nœud contient la formule X : lui créer un fils contenant la formule Y »

2.
$$\frac{X \qquad Z}{Y}$$
Qui se lit « si deux nœuds contiennent les formules X et Z : leur créer un fils contenant la formule Y »

3.
$$\frac{\begin{array}{c} X \quad (i) \\ \dots \\ Y \end{array}}{Z}\ (i)$$
Qui se lit « si une feuille contient la formule X, et qu'il existe un chemin de cette feuille à un nœud qui contient Y alors lui créer un fils contenant Z et **décharger** X dans **toutes les feuilles** la contenant, car X peut être présente dans plusieurs nœuds.
L'indice (i) sert à préciser la formule à décharger : en effet, il peut y avoir plusieurs feuilles (contenant d'autres formules que X) ayant un chemin de X à Y». Notons tout de suite qu'une formule peut très bien être déchargée plusieurs fois.

Il faut penser que ce dernier cas inclut le cas dégénéré où les deux nœuds en question sont en fait le même. Le schéma devient alors :

4.
$$\frac{X \quad (i)}{Z}\ (i)$$

Définition 9 : Dans le schéma 3 ci-dessus,

On dit que la formule Y **dépend** de l'hypothèse X. Bien entendu, Y **dépend** aussi de toutes les hypothèses H contenues dans des feuilles qu'un chemin lient à Y. La formule Z, elle, ne dépend plus de l'hypothèse X puisque on est sorti de la zone où elle était utilisée en la déchargeant. Par contre elle dépend toujours des hypothèses H.

Préliminaires

Toutes ces règles sont accompagnées de commentaires du type « Avoir prouvé A et avoir prouvé A$\rightarrow$B sous H suffit pour prouver B sous H ». A et B sont des formules, et H représente l'ensemble d'hypothèses dont dépendent les formules concernées (ici A et A$\rightarrow$B).

Les règles d'élimination ne déchargeant pas d'hypothèses

$$\frac{A \qquad A \rightarrow B}{B}\ (E\rightarrow)$$
Avoir prouvé A et avoir prouvé A$\rightarrow$B sous H suffit pour prouver B sous H (c'est le **Modus Ponens**)

$$\frac{A \wedge B}{A}\ (E\wedge)$$
Une preuve de A$\wedge$B sous H suffit pour prouver A sous H

$$\frac{A\wedge B}{B}\,(E\wedge)$$

Une preuve de $A\wedge B$ sous H suffit pour prouver B sous H

$$\frac{A \qquad A\leftrightarrow B}{A}\,(E\leftrightarrow)$$

Avoir prouvé A et avoir prouvé $A\leftrightarrow B$ sous H suffit pour prouver A sous H

$$\frac{A \qquad A\leftrightarrow B}{B}\,(E\leftrightarrow)$$

Avoir prouvé A et avoir prouvé $A\leftrightarrow B$ sous H suffit pour prouver B sous H

$$\frac{\neg\neg A}{A}\,(E\wedge)$$

Avoir prouvé $\neg\neg A$ sous H suffit pour prouver A sous H

La règle d'élimination du $\vee$ est renvoyée à plus tard.

Les règles d'introduction ne déchargeant pas d'hypothèses

$$\frac{A \qquad B}{A\wedge B}\,(I\wedge)$$

Avoir prouvé A et avoir prouvé B sous H suffit pour prouver $A\wedge B$ (ou de $B\wedge A$) sous H

$$\frac{A}{A\vee B}\,(I\vee)$$

Avoir prouvé A sous H suffit pour prouver $A\vee B$ (ou $B\vee A$) sous H

$$\frac{A \qquad \neg A}{\bot}\,(I\bot)$$

Avoir prouvé A et avoir prouvé $\neg A$ sous H mène à une contradiction ($\bot$) sous H

$$\frac{A\to B \qquad B\to A}{A\leftrightarrow B}\,(I\leftrightarrow)$$

Une preuve de $A\to B$ et une preuve de $B\to A$ sous H donnent une preuve de $A\leftrightarrow B$ sous H

Les règles déchargeant des hypothèses

De manière générale, ces règles sont de la forme : si on a obtenu une preuve de X **sous l'hypothèse** Y (plus éventuellement d'autres hypothèses), alors conclure Z en **déchargeant** Y : on obtient de la sorte une preuve de Z qui ne dépend plus de Y. Pour clarifier quelles hypothèses on décharge à certains moments, on va les numéroter. Commençons par la plus simple : $(I\to)$

$$\frac{\begin{array}{c} A \quad (i) \\ \dots \\ B \end{array}}{A\to B}\,(I\to)(i)$$

Une preuve de B sous H et A (numérotée i) fournit une preuve de $A\to B$ (sans l'hypothèse (i) que l'on décharge, en précisant le numéro de l'hypothèse déchargée, ici i) On obtient donc une preuve de $A\to B$ sous H

$$\frac{\begin{array}{c} A \quad (i) \\ \dots \\ \bot \end{array}}{\neg A}\,(I\neg)(i)$$

Une preuve de $\bot$ sous H et A (numéro i) (H et A mènent à une contradiction) fournit une preuve de $\neg A$ (sans l'hypothèse (i) que l'on décharge). On obtient donc une preuve de $\neg A$ sous H

La règle la plus complexe est celle de (E∨) qui nécessite une convention graphique à elle toute seule :

```
                A (j)          B (k)
                …              …
A∨B             C              C
-------------------------------------------- (E∨)(j,k)
                     C
```

Avoir prouvé A∨B sous H,
avoir prouvé C sous H et A,
avoir prouvé C sous H et B,
suffisent à prouver C sous H.
On décharge donc les hypothèses temporaires j et k

Reprenons l'exemple de la neige et du verglas ; on construit une preuve de ¬é sous les hypothèses 1, 2, 3 et 4 :

```
                 n (5)   n→f (1)          v (6)   v→f (2)
                 ------------------ (E→)  ----------------- (E→)
n∨v (3)             f                     f                      é (7)    é→¬f (4)
-------------------------------------------------------------- (E∨)(5,6)  ------------------ (E→)
                         f                                                      ¬f
                         ---------------------------------------------------------- (I⊥)
                                          ⊥
                                             ------ (I¬)(7)
                                              ¬ é
```

NB : il est nécessaire qu'à la fin, toutes les hypothèses temporaires (5, 6 et 7) aient pu être déchargées. De plus, elles peuvent avoir été déchargées plusieurs fois chacune.

Commentaires : la difficulté dans ce type d'approche est qu'il n'y pas a priori de systématicité (contrairement à la méthode des tableaux qui est automatique) et c'est normal dans la mesure où ce système est sensé reproduire le raisonnement lui-même. En pratique, l'intuition des hypothèses à formuler est guidée par l'allure de la formule à prouver. Ainsi, pour prouver une formule de la forme A→B, on pose l'hypothèse A (temporairement) et on cherche à obtenir une preuve de B sous A. Puis on utilise la règle (I→) d'introduction du →.

Définition 10 : S'il n'existe pas de preuve de H |- A on dira que ¬A est ***consistante*** *avec H ou encore que H∪{¬A} est* ***consistant****. Par voie de conséquence, si un ensemble de la forme H∪{¬A} n'est pas consistant (=****inconsistant****), c'est qu'il existe une preuve de H|-A : A est déductible de H si et seulement si H∪{¬A} est consistant.*

Nous verrons un peu plus loin que les notions de *satisfiable, valide, insatisfiable, conséquence logique de* correspondent respectivement à celles de *consistante, déductible sans hypothèse, inconsistante, déductible de.*

Les règles dérivées (ou admissibles) :

Une règle dérivée est une règle du type de celles vues ci-dessus qui peut être ajoutée aux autres sans pour autant augmenter le nombre de formules déductibles. On montre qu'elle est dérivée en prouvant que toute preuve où elle interviendrait peut être écrite sans elle.

```
A       ¬A∨B
-----------------        (Syllogisme disjonctif)
        B

¬A          A→B
--------------------     (Modus Tollens)
        ¬B

--------     (Introduction d'un théorème, si on a déjà démontré A,
   A         cette règle évite d'avoir à recopier la preuve de A préalablement obtenue)
```

4 Une approche similaire : les séquents

L'objet de base du calcul est le *séquent*, qui est un couple de **listes** (et donc ordonnées) finies (éventuellement vides) de formules. Les séquents sont usuellement notés :

$$\Gamma \mathrel{|\!\!-\!\!-} \Delta$$

avec Γ=(A_1,…, A_n) et Δ=(B_1,…, B_p) et où les A_i sont les *hypothèses* et les B_j sont les *conclusions* du séquent. Le séquent ci-dessus se lit «
L'interprétation d'un séquent est la suivante : la virgule à gauche s'interprète comme une conjonction, la virgule à droite comme une disjonction et le symbole |-- comme une implication, si bien que le séquent ci-dessus peut se comprendre comme une notation pour la formule :

$$(A_1 \wedge\ldots\wedge A_n) \to (B_1 \vee\ldots\vee B_p)$$

Le séquent ci-dessus se lit « Δ est déductible de Γ ». On relèvera un risque de confusion au sujet du symbole |-- et de la notion de déductibilité avec la DN, mais l'équivalence des deux systèmes fait que cette ambiguïté est inoffensive.

Règles

Les règles du calcul des séquents spécifient comment à partir d'un certain nombre (éventuellement nul) de séquents *prémisses*, on peut dériver un nouveau séquent *conclusion*. Dans chacune des règles les lettres grecques Gamma, Delta, etc. dénotent des suites de formules, on fait figurer les séquents prémisses au-dessus, séparés par un trait horizontal du séquent conclusion. Elles se divisent en trois groupes : le groupe *identité*, le groupe *structurel* et le groupe *logique*.

Groupe identité

$$\frac{}{A \vdash A}\ (\text{axiome}) \qquad \frac{\Gamma \vdash A , \Delta \qquad \Gamma', A \vdash \Delta'}{\Gamma , \Gamma' \vdash \Delta , \Delta'}\ (\text{coupure})$$

Une discussion sur la règle de coupure figure plus bas. Remarquons que la règle axiome est la seule de tout le calcul qui n'a pas de séquent prémisse.
Pour les groupes structurel et logique, les règles viennent par paire selon le côté, gauche ou droit, du séquent où elles agissent.

Groupe structurel

(π dénote une permutation des éléments de la liste en argument)

$$\frac{\Gamma \vdash \Delta}{\pi(\Gamma) \vdash \Delta}\ (\text{échange gauche}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \pi(\Delta)}\ (\text{échange droite})$$

$$\frac{\Gamma \vdash \Delta}{\Gamma,A \vdash \Delta}\ (\text{affaiblissement gauche}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta,A}\ (\text{affaiblissement droite})$$

$$\frac{\Gamma,A,A \vdash \Delta}{\Gamma,A \vdash \Delta}\ (\text{contraction gauche}) \qquad \frac{\Gamma \vdash \Delta,A,A}{\Gamma \vdash \Delta,A}\ (\text{contraction droite})$$

Les règles d'échanges sont responsables de la *commutativité* de la logique.
Les règles de contraction et d'affaiblissement expriment une forme d'idempotence des opérateurs $\vee$ et $\wedge$ de la logique : une formule *A* est équivalente aux formules $A\vee A$ et $A\wedge A$.

Groupe logique

$$\frac{}{\Gamma,\bot \vdash \Delta}\ (\bot\ \text{- gauche}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta,\bot}\ (\bot\ \text{- droite})$$

$$\frac{\Gamma \vdash A,\Delta}{\Gamma,\neg A \vdash \Delta}\ (\neg\ \text{- gauche}) \qquad \frac{\Gamma,A \vdash \Delta}{\Gamma \vdash \neg A,\Delta}\ (\neg\ \text{- droite})$$

$$\frac{\Gamma,A,B \mid\!-\!- \Delta}{\Gamma,A\wedge B \mid\!-\!- \Delta}\ (\wedge\text{ - gauche}) \qquad \frac{\Gamma \mid\!-\!- A,\Delta \qquad \Gamma \mid\!-\!- B,\Delta}{\Gamma \mid\!-\!- \Delta,A\wedge B}\ (\wedge\text{ - droite})$$

$$\frac{\Gamma,A \mid\!-\!- \Delta \qquad \Gamma,B \mid\!-\!- \Delta}{\Gamma,A\vee B \mid\!-\!- \Delta}\ (\vee\text{ - gauche}) \qquad \frac{\Gamma \mid\!-\!- \Delta,A,B}{\Gamma \mid\!-\!- \Delta,A\vee B}\ (\vee\text{ - droite})$$

$$\frac{\Gamma \mid\!-\!- A,\Delta \qquad \Gamma,B \mid\!-\!- \Delta}{\Gamma,A\to B \mid\!-\!- \Delta}\ (\to\text{ - gauche}) \qquad \frac{\Gamma,A \mid\!-\!- \Delta,B}{\Gamma \mid\!-\!- \Delta,A\to B}\ (\to\text{ - droite})$$

$$\dfrac{\dfrac{\dfrac{}{n \mid\!-\!- n}(ax)}{v\to f,é\to\neg f,n \mid\!-\!- n,\neg é}(affD+G) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{é \mid\!-\!- é}(ax)}{v\to f,n,f,é \mid\!-\!- é}(affG)}{v\to f,n,f \mid\!-\!- é,\neg é}(\neg D) \qquad \dfrac{\dfrac{\dfrac{}{f \mid\!-\!- f}(ax)}{v\to f,n,f \mid\!-\!- f,\neg é}(affD+G)}{v\to f,n,f,\neg f \mid\!-\!- \neg é}(\neg G)}{v\to f,é\to\neg f,n,f \mid\!-\!- \neg é}(échG+\to G)}{n\to f,v\to f,é\to\neg f,n \mid\!-\!- \neg é}(échG+\to G) \qquad \dfrac{}{n\to f,v\to f,é\to\neg f,v \mid\!-\!- \neg é}(\text{voir partie gauche})$$
$$\overline{\qquad n\vee v,n\to f,v\to f,é\to\neg f \mid\!-\!- \neg é \qquad}\ (échG + \vee G)$$

Un exemple de preuve (il manque la partie droite, similaire à celle de gauche)

Il existe plusieurs systèmes de séquents : on peut, par exemple, considérer que la règle (axiome) est la suivante :

$$\frac{}{\Gamma,A,\Gamma' \mid\!-\!- \Delta,A,\Delta'}$$

ce qui évite de multiplier les applications des règles de contraction, affaiblissement et échange. De manière plus générale, on peut formuler les règles ci-dessus en y incluant directement les contractions et affaiblissements nécessaires de façon à simplifier les preuves obtenues.

Ressources informatiques : Nous renvoyons le lecteur à l'applet : http://bach.istc.kobe-u.ac.jp/seqprover/ qui élabore des preuves de type séquent, en utilisant un jeu de règles un peu plus riche que celui présenté ici.
Signalons aussi l'excellent ProofDisplay qui construit automatiquement des preuves en déduction naturelle dans un format similaire au nôtre (« Box tree display ») mais aussi dans d'autres formats, lien ici :
http://www.phil.cmu.edu/projects/apros/index.php?page=generator&subpage=proofdisplay

5 Résultat généraux.

Théorème 5 Le système de déduction naturel présenté ci-dessus est adéquat et complet pour les formules valides de la logique des propositions, c'est-à-dire : H |-- C si et seulement si H |= C et ce même si H est infini.

Théorème 6 Le système de séquents présenté ci-dessus est adéquat et complet pour les formules valides de la logique des propositions, c'est-à-dire : H |-- C si et seulement si H |= C et ce même si H est infini.

Théorème 7 (*Hauptsatz de Gentzen, 1934).* Pour toute preuve d'un séquent Γ |-- Δ, il existe une preuve du même séquent ne faisant pas intervenir la règle de coupure (seule règle ne respectant pas la propriété de sous-formule) ce qui permet de rechercher des preuves de manière automatique.

Chapitre 5 Réfutation par résolution

1 Formes normales disjonctives et conjonctives, Ensemble de clauses

Sur le plan logique, cette section paraîtra quelque peu anecdotique, mais la connaissance de ces formes normales (conjonctives tout particulièrement) est nécessaire pour aborder le langage de programmation PROLOG. En effet un programme écrit dans ce langage est en fait une formule donnée en forme normale conjonctive (en fait plus que cela comme on le verra). Le principe de fonctionnement de PROLOG repose sur l'application du principe de résolution présenté en section 2 de ce chapitre.

Définition 11 : Un littéral est soit une proposition, soit la négation d'une proposition, ou $\perp$, ou T. L'ensemble des littéraux est donc défini par LIT = $\{p, \neg p \,/\, p \in PROP\} \cup \{\perp, T\}$

Définition 12 : Une formule est dite en forme normale disjonctive (en abrégé FND) ssi elle est de la forme :

- $\Phi = (\chi_{11} \wedge \chi_{12} \wedge \dots \wedge \chi_{1n}) \vee (\chi_{21} \wedge \chi_{22} \wedge \dots \wedge \chi_{2n}) \vee \dots \vee (\chi_{k1} \wedge \chi_{k2} \wedge \dots \wedge \chi_{kn})$
  (où chaque χ_{ij} est un littéral)

Une formule est dite en forme normale conjonctive (en abrégé FNC) ssi elle est de la forme :

- $\Phi = (\chi_{11} \vee \chi_{12} \vee \dots \vee \chi_{1n}) \wedge (\chi_{21} \vee \chi_{22} \vee \dots \vee \chi_{2n}) \wedge \dots \wedge (\chi_{k1} \vee \chi_{k2} \vee \dots \vee \chi_{kn})$
  (où chaque χ_{ij} est un littéral)

Comme on l'a vu auparavant, pour chaque formule il existe une formule en FND qui lui est équivalente, il suffit d'observer sa table de vérité. En pratique, dresser la table de vérité peut s'avérer fastidieux, pour déterminer une FND ou une FNC, on utilisera plutôt le Théorème 3 et une partie de la liste des « tautologies utiles » permettant par remplacements successifs d'obtenir la forme normale voulue. L'algorithme ci-dessous permet de calculer une FND ou une FNC:

1. Remplacer les sous-formules $(A \leftrightarrow B)$ par $(A \rightarrow B) \wedge (B \rightarrow A)$
2. Remplacer les sous-formules $(A \rightarrow B)$ par $(\neg A \vee B)$
3. Remplacer les sous-formules $\neg(A \vee B)$ par $(\neg A \wedge \neg B)$
4. Remplacer les sous-formules $\neg(A \wedge B)$ par $(\neg A \vee \neg B)$ et reprendre en 3 tant que c'est possible
5. Eliminer les doubles négations : remplacer les sous-formules $\neg\neg A$ par A
6.
   Pour une FND : Remplacer les sous-formules $A \wedge (B \vee C)$ et $(B \vee C) \wedge A$ par $(A \wedge B) \vee (A \wedge C)$ et reprendre en 6.
   Pour une FNC : Remplacer les sous-formules $A \vee (B \wedge C)$ et $(B \wedge C) \vee A$ par $(A \vee B) \wedge (A \vee C)$ et reprendre en 6.

A tout moment on peut bien sûr appliquer des simplifications comme $(A \wedge T) \leftrightarrow A$, …

L'algorithme ci-dessus ne procédant que par remplacement d'équivalents la formule A' obtenue finalement est évidemment équivalente à la formule initiale A. De plus il est clair qu'elle est en FND ou FNC selon. Donc :

Théorème 8 (Théorème de la FND – resp. de la FNC) : Pour toute formule A, il existe une formule A' équivalente (c-à-d $\models A \leftrightarrow A'$) qui soit en FND – resp. en FNC.

Définition (Clause) : Une clause est un ensemble de littéraux distincts, ensemble éventuellement vide (Intuitivement, une clause correspond à la disjonction des littéraux qui la composent). Une clause $\{\chi_1, \chi_2, \dots, \chi_n\}$ est dite satisfaite par une valuation v si et seulement si la formule $v(\chi_1 \vee \chi_2 \vee \dots \vee \chi_n) = 1$. Un ensemble de clauses $E=\{C_1, C_2, \dots, C_m\}$ est satisfait par une valuation v si et seulement si $v(C_i)=1$ pour toute clause C_i de E.

Définition (Clause tautologique) : Une clause contenant deux littéraux complémentaires (comme le sont p et $\neg$p) est appelée *clause tautologique*.

Définition (Ensemble de clauses d'une formule) : Soit A une formule, A0 l'une de ses FNC avec $A0 = (\chi_{11} \vee \chi_{12} \vee \dots \vee \chi_{1n}) \wedge (\chi_{21} \vee \chi_{22} \vee \dots \vee \chi_{2n}) \wedge \dots \wedge (\chi_{k1} \vee \chi_{k2} \vee \dots \vee \chi_{kn})$ et telle que $\forall i,j,k : \chi_{ij} \neq \chi_{ik}$ (i.e. un littéral donné n'apparaît au plus qu'une fois dans une disjonction), alors l'ensemble de clauses $\{C_1, \dots, C_k\}$ où $C_i = \{\chi_{i1} \vee \chi_{22} \vee \dots \vee \chi_{in}\}$ est un *ensemble de clauses* pour A.

Remarque importante : Puisqu'une clause est un ensemble, l'ordre des littéraux n'a pas d'importance. De même, l'ordre des clauses n'importe pas vis-à-vis de l'ensemble de clauses. Ainsi, les deux FNC suivantes :

$$(p \vee q \vee \neg r) \wedge (r \vee q \vee \neg s)$$

$$(q \vee \neg s \vee r) \wedge (p \vee \neg r \vee q)$$

admettent toutes deux $\{\{p,q,\neg r\},\{r,q,\neg s\}\}$ comme ensemble de clauses.

Puisque, par commutativité et idempotence du $\vee$ et du $\wedge$, les formules en FNC qui ne diffèrent que par l'ordre des littéraux au sein des disjonctions ou par l'ordre des conjonctions, ou par le nombre d'occurrence d'un même littéral au sein d'une disjonction, ou encore par le nombre de disjonctions équivalentes sont elles-mêmes équivalentes, il est naturel de représenter toutes ces formules équivalentes par un même ensemble de clauses.
Dorénavant, on supposera qu'une clause ne contient au plus qu'une occurrence de chaque littéral, et que chaque clause n'apparaît qu'une fois au plus dans un ensemble de clauses.
De plus, si χ est un littéral et A une clause, on notera $\chi \vee A$ la clause obtenue en ajoutant χ à A au lieu de $A \cup \{\chi\}$.

2 Le principe de résolution

Définition (Résolvante) : Une clause C est la résolvante des clauses A et B si et seulement si il existe une proposition p et deux clauses A0 et B0, tels que : $A = p \vee A0$, $B = \neg p \vee B0$, et $C = A0 \vee B0$.

NB : si A0 et B0 sont vides, alors C est vide aussi.
NB : On vérifie facilement que $\models ((p \vee A0) \wedge (\neg p \vee B0)) \rightarrow (A0 \vee B0)$, donc que la résolvante de deux clauses est une conséquence logique de leur conjonction.

Définition (Déduction par résolution) : Soit **C** *un ensemble de clauses, et B une clause. Une déduction par résolution de B à partir de* **C** *est une séquence S1,S2,…,Sn de clauses telles que Sn = B et pour tout i ($1 \leq i \leq n$) Si est dans* **C** *ou Si est la résolvante de deux clauses Sj et Sk telles que j<i et k<i. Une réfutation de* **C** *(par résolution) est une déduction de $\perp$* (la clause vide) à partir de **C**. On écrira **C** $|\!\!-\!\!-_{\text{rés}}$ B (ou simplement **C** |-- B s'il n'y a pas d'ambiguïté) pour exprimer qu'il existe une déduction par résolution de B à partir de **C**.

Exemple : Soit ***C*** l'ensemble de clauses {C1, C2, C3} avec :

C1 = ¬p, q	C3 = ¬q, ¬r	C5 = ¬s
C2 = ¬p, r, s	C4 = p	

C6 = q	(résolvante de C1 et C4)
C7 = ¬r	(résolvante de C3 et C6)
C8 = r, s	(résolvante de C2 et C4)
C9 = r	(résolvante de C5 et C8)
C10 = $\perp$	(résolvante de C7 et C9)

On l'a vu, cela signifie que $\perp$ est une conséquence logique de ***C***. Donc si on obtient la clause vide, c'est que l'ensemble de clauses de départ est insatisfiable. Qu'en est-il de la réciproque ?

Théorème de la résolution (Robinson, 1965) : Soit ***C*** un ensemble de clauses, ***C*** est insatisfiable si et seulement si ***C*** $|\!\!-\!\!-_{\text{rés}} \perp$.
Démonstration : On l'a vu, l'un des sens est acquis (si ***C*** $|\!\!-\!\!-_{\text{rés}} \perp$ alors ***C*** est insatisfiable : la résolution est adéquate, si elle aboutit à une contradiction c'est que l'ensemble de clauses de départ est contradictoire). Il reste à prouver la complétude de la résolution (qu'elle permet toujours de prouver l'insatisfiabilité d'un ensemble de clauses) : il faut démontrer que si ***C*** est insatisfiable alors ***C*** $|\!\!-\!\!-_{\text{rés}} \perp$.

Deuxième Partie : la logique du premier ordre (LP1)

ou logique des prédicats

Chapitre 6 Le langage

Langage de la logique des prédicats.

1) Alphabet
   Connecteurs booléens : $\{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\} \cup \{\neg\} \cup \{\bot\}$
   Séparateurs : ( )
   Ensembles PRED_n de prédicats n-aires, n entier naturel
   Ensembles FON_n de fonctions n-aires, n entier naturel
   Ensemble CST de constantes
   Ensemble VAR de variables
   Quantificateurs : $\forall$ et $\exists$

2) L'ensemble TERM des termes du premier ordre (sur cet alphabet) :
   Soit T_0 = CST $\cup$ VAR et T_n+1 = {f(t_1,…,t_n) / t_i$\in$ T_n et f$\in$FON_n}
   TERM est le plus petit ensemble contenant T_i pour i quelconque.

3) Formules, l'ensemble FOR1 des formules prédicatives est défini par :
   Soit F_0 = {P(t_1,…,t_n) / t_i $\in$ TERM et P $\in$ PRED_n} $\cup$ {$\bot$}
   F_n+1 = F_n
   $\cup$ {(A * B) / A,B $\in$ FOR_n, * connecteur binaire}
   $\cup$ {$\neg$A / A $\in$ FOR_n}
   $\cup$ {($\forall$x A), ($\exists$x A) / x$\in$VAR et A $\in$ F_n}
   et FOR1 est le plus petit ensemble contenant F_i pour i quelconque.

3 Occurrences et substitutions de variables / Variables libres et liées

Les définitions relatives aux occurrences de sous-formules et à leur substitution par d'autres formules, telles que définies dans la partie « Logique des propositions », restent valables, les quantificateurs se traitant comme la négation (une seule sous-formule, à droite par convention). Par contre, les arguments des prédicats et fonctions n-aires ne peuvent plus être atteint avec simplement des suite de 1 et de 2, on va donc utiliser les entiers.

Occurrence dans une formule :
Une occurrence est mot du langage (n)* où n est l'ensemble des entiers, l'occurrence vide étant représentée par λ. Cette suite d'entiers permet d'accéder à tout noeud de l'arbre d'une formule et à toute sous-formule, à tout terme et à tout sous-terme, la suite vide (λ) donnant accès à la racine (Par défaut quand le fils est unique, il est en position 1).

Rappel (Concaténation) : Soit E un ensemble de suites finies d'entiers, et soit k$\in$**N**, alors k.E dénote l'ensemble des suites de E préfixées par k, i.e. k.E={km / m$\in$E}.

Exemple : Si E={λ, 11, 212, 231} et k=1, alors k.E={1, 111, 1212, 1231}

Définition (Ensemble des occurrences dans une formule) : L'ensemble Occ(A) des occurrences de A est défini inductivement par :

Occ(x) = {λ} si t est une variable ou une constante
Occ($\bot$)={λ} ;
Occ(p)= {λ} ; pour toute proposition p$\in$PROP
Occ(p(t1,...,tn) = 1.Occ(t1) $\cup$ 2.Occ(t2) ... $\cup$ n.Occ(tn) $\cup$ {λ} si p est un prédicat
Occ(f(t1,...,tn) = 1.Occ (t1) $\cup$ 2.Occ (t2) ... $\cup$ n.Occ (tn) $\cup$ {λ} si f est une fonction
Occ($\neg$A1) = 1.Occ(A1) $\cup$ {λ} ;
Occ((A1 * A2)) = 1.Occ(A1) $\cup$ 2.Occ(A2) $\cup$ {λ}
Occ($\forall$x (A1)) = 1.Occ(A1) $\cup$ {λ}
Occ($\exists$x (A1)) = 1.Occ(A1) $\cup$ {λ}

Exemple : Occ(p(x,f(y,<u>a</u>)) et l'occurrence soulignée est désignée par 2.2
= 1.Occ(x) $\cup$ 2. Occ(f(y,a)) $\cup$ {λ}
= {1.λ} $\cup$ 2.(1. Occ(y) $\cup$ 2. Occ(a) $\cup$ {λ}) $\cup$ {λ}

$= \{1.\lambda\} \cup 2.(1.\{\lambda\} \cup 2.\{\lambda\} \cup \{\lambda\}) \cup \{\lambda\}$
$= \{1, 2.1, 2.2.2, 2, \lambda\}$

Définition (Type d'occurrence) : Soit A une formule, m une occurrence de A (i.e. $m \in Occ(A)$), si A[m] est un terme on parlera d'une occurrence de terme, si c'est une formule on parlera d'occurrence de sous-formule.

Définition (Occurrence d'un terme) : Soit A une formule, m une occurrence de terme de A (i.e. $m \in Occ(A)$), l'occurrence m dans A, notée A[m], est définie par :
$A[\lambda] = t$;
$(\neg A1)[1.m'] = A1[m']$;
$(A1 * A2)[1.m'] = A1[m']$; pour $* \in \{\rightarrow,\vee,\wedge\}$
$(A1 * A2)[2.m'] = A2[m']$; pour $* \in \{\rightarrow,\vee,\wedge\}$
$(\forall x\ (A1))[1.m'] = A1[m']$;
$(\exists x\ (A1))[1.m'] = A1[m']$;
$F(t1,...,tn)[i.m'] = ti[m']$ si F est un prédicat ou une fonction

Exemple : $(\forall x\ (p(x,f(y,a)) \rightarrow \exists y\ (q(y) \wedge r(x,y))))$ [1.1.2]
$= (p(x,f(y,a)) \rightarrow \exists y\ (q(y) \wedge r(x,y)))$ [1.2] $= p(x,f(y,a))$ [2] $= f(y,a)$ $[\lambda] = f(y,a)$

Définition (Champ d'un quantificateur): (ici, Q désigne indifféremment $\forall$ et $\exists$). Soient A une formule, et (Qx B) une sous-formule de A d'occurrence m, le champ de cet occurrence du quantificateur Qx est l'ensemble des occurrences m' dont m est un préfixe strict (c-à-d tel que $m' = m.m''$ avec $m'' \neq \lambda$, autrement dit, c'est l'ensemble des sous-formules et sous-termes de B)
Exemple : dans $\forall x\ (p(x,f(y,a)) \rightarrow \exists y\ (q(y) \wedge r(x,y)))$ le champ de $\exists y$ est {222,2221,2222,22211,22221,22222}

Définition (Variables : occurrences libres/liées): Soient A une formule, et x une variable d'occurrence m dans A, on dira que cette occurrence de x est liée si et seulement si elle dans le champ d'un quantificateur $\exists x$ ou $\forall x$, dans le cas contraire on dira qu'elle est libre. Une variable est dite libre dans A si l'une au moins de ses occurrences y est libre, et liée dans A si l'une au moins de ses occurrences y est liée (NB : une variable peut donc être à la fois libre et liée pourvu que l'une de ses occurrences soit libre et une autre liée).

On note Free(A) l'ensemble des variables libres de A, Bound(A) celui des variables liées de A et var(A) l'ensemble de toutes les variables de A, NB : $var(A) = Free(A) \cup Bound(A)$.

Définition (Clôtures): Soient A et x1,…xn les variables libres de A, on appelle clôtures universelle et existentielle de A les formules $\forall x1,\dots,\forall xn.\ A$ et $\exists x1,\dots,\exists xn.\ A$.

Définition (Substitution des occurrences libres de variables): Soient A une formule, t un terme et x une variable libre de A, on note A[t/x] le résultat de la substitution des occurrences **libres** (et seulement celles-ci) de x dans A par t, elle est définit inductivement par (Q désigne indifféremment $\forall$ ou $\exists$) :
$x[t/x] = t$;
$(\neg A1)\ [t/x] = (\neg A1[t/x])$;
$(A1 * A2)\ [t/x] = (A1[t/x] * A2[t/x])$; pour $* \in \{\rightarrow,\vee,\wedge\}$
$(Qx\ A1)[t/x] = (\forall x\ A1)$; car les occurrences de x dans $(\forall x\ A1)$ sont nécessairement liées
$(Qy\ A1)[t/x] = (\forall y\ A1[t/x])$; si $y \neq x$
$f(t1,...,tn)[t/x] = f(t1[t/x],\dots,tn[t/x])$ si f est un prédicat ou une fonction

Définition (Substitution saine): Soient A une formule, t un terme et x une variable libre de A, on dira que la substitution [t/x] est **saine pour x dans A** si et seulement si aucune variable de t n'est « capturée » par un quantificateur de A. Formellement, si et seulement si il n'y a dans A aucune occurrence libre de x qui soit dans le champ d'un quantificateur Qy et tel que y soit une variable ayant une occurrence dans t.
Exemple : dans $(p(x,f(y,a)) \rightarrow \exists y\ (q(y) \wedge r(x,y)))$ la substitution [f(y)/x] n'est pas saine (f(y) se fait capturer par $\exists y$), alors que [f(z)/x] est saine.

On peut se passer de la notion de substitution saine si l'on se restreint aux substitutions précédées d'un renommage (= substitutions sur les variantes alphabétiques).

Définition (Variante alphabétique): Soient A une formule, V un ensemble de variables, on dira A' est une *variante alphabétique* (ou encore un *renommage* de A) loin de V si et seulement si pour toute sous-formule QxB de A d'occurrence m, A' contient à cette occurrence une sous-formule QyB[y/x] (où y est une « nouvelle variable », c'est-à-dire que $y \notin V$).

Plutôt que de se restreindre aux substitutions saines, on peut définir les substitutions de la manière suivante :

Définition (Substitution des occurrences libres de variables sur une variante alphabétique): Soient A une formule, t un terme et x une variable libre de A, on note A'[t/x] le résultat de la substitution des occurrences **libres** (et seulement celles-ci) de x par t dans une variante alphabétique A' de A **loin de var(t)**. Il s'agit donc d'une substitution ordinaire précédée d'un renommage.

Théorème 9 Une formule est équivalente à n'importe laquelle de ses variantes alphabétiques. (On sera en mesure de démontrer ceci dans la section suivante).

Théorème 10 Soient A une formule, t un terme, x une variable libre de A, et A' une variante alphébétique de A, alors la substitution [t/x] est toujours **saine pour x dans A'**. C'est ce qui nous permettra de nous passer de la notion de substitution saine.

Ces notions sont indispensables à la présentation du système de déduction naturelle qu'on verra plus tard.

A La théorie des modèles

On se souvient ici de ce qu'est une valuation dans le cadre de la logique des propositions : une fonction de l'ensemble PROP (ensemble des propositions) dans l'ensemble {0,1} qui respecte en plus des contraintes (comme $v[(B \wedge C)] = \min(v[B] , v[C])$)). Cette définition des valuations est issue des postulats de bivalence (qui impose qu'il n'y ait que deux valeurs de vérité ; on a pris 0 et 1 arbitrairement) et de vérifonctionnalité (qui impose que la valeur de vérité d'une formule ne dépende que de celle de ses composants. Ses composants étant des propositions, la manière dont elle en dépend est prise en charge par les contraintes comme $v[(B \wedge C)] = \min(v[B] , v[C])$).

Dans le cadre de la logique des prédicats, on reprend ces deux postulats : bivalence et vérifonctionnalité. Mais les choses se compliquent dans la mesure où les composants d'une formule ne sont pas que des propositions, mais aussi des constantes, des variables et plus généralement des termes auquel on ne peut pas attribuer une valeur de vérité.

Ainsi, quelle valeur une valuation va-t-elle attribuer à la constante s vue plus haut ? Cette constante doit être comprise comme représentant un objet (le personnage Socrate, le système de réservation de la SNCF, le chien de mon voisin, mon poisson rouge, ou n'importe quoi d'autre) et cela est justement fixé par la valuation : une valuation en logique des prédicats attribue donc à une constante le nom d'un objet (rappelons encore qu'objet est ici pris en un sens très large incluant les êtres vivants, les objets mathématiques, les abstractions, …). En fait, il en va de même pour tous les termes : une valuation doit permettre d'attribuer un nom d'objet à chaque terme du langage de la logique des prédicats. Dans un second temps, une valuation doit aussi attribuer une valeur de vérité à chaque formule atomique et respecter des contraintes qui permettent de déduire la valeur de vérité d'une formule complexe à partir de celles des formules atomiques qui la composent…

Donc, une constante doit être interprétée par un nom d'objet… mais parmi lesquels ? C'est la première chose à préciser : dans quel domaine on se place (les nombres entiers, les objets géométriques, les personnes, les chats et les nombres réels, …).

1 Interprétation / valuation / modèle

Une valuation est, en résumé, la donnée de deux choses : un domaine (appelé aussi univers du discours) et une valuation proprement dite, c'est un couple <D,v> où :

- D est un ensemble non-vide
- v est une fonction qui attribue une valeur de D à chaque terme et une valeur de vérité à chaque formule atomique

Plus précisément, la fonction v doit associer en même temps :

- un élément de D à chaque constante ($v : CONS \rightarrow D$)
- un élément de D à chaque variable ($v : VAR \rightarrow D$)
- une fonction de D^n dans D à chaque fonction à n arguments ($v : FONC^n \rightarrow (D^n \rightarrow D)$)
- une fonction de D^n dans {0,1} à chaque prédicat à n arguments ($v : PRED^n \rightarrow (D^n \rightarrow \{0,1\})$)

- et, pour des raisons « techniques », si $d \in D$ alors $v(d) = d$. On supposera donc que $D \subseteq CONS$.[1]

Interprétation des termes et des formules atomiques

Etant donnée une valuation <D,v>, un terme t, la valeur attribuée à t par v s'obtient par :

- v(t) si t est une constante ou une variable
- sinon, c'est que t = f(t1,…,tn) et alors v(t) = v(f) (v(t1),…,v(tn)), en effet, par définition v(f) donne une fonction de D^n dans D et chaque v(ti) donne un élément de D.

Soit P(t1,…,tn) une formule atomique, sa valeur de vérité s'obtient par :

- v(P(t1,…,tn)) = v(P) (v(t1),…,v(tn)), en effet, par définition v(P) donne une fonction de D^n dans {0,1} et chaque v(ti) donne un élément de D.

Exemple abstrait :
Soit la fonction binaire (à deux arguments) $\oplus$, trois constantes a, b et c, et le prédicat binaire #. A l'aide de ces éléments, on peut constituer des termes comme $(a \oplus b)$, $((b \oplus c) \oplus b)$, … et des formules atomiques comme
$(a \oplus b) \# ((b \oplus c) \oplus c)$, $(a \oplus b) \# c$, … (NB : $\oplus$ et # sont utilisés en notation infixe)
Considérons les valuations suivantes :
i. <N,v0> où N est l'ensemble des entiers et v0 est définie par :

- v0(a) = 12, v0(b) = 24 et v0(c) = 36
- v0($\oplus$) est l'addition sur N (qui est bien une fonction de N^2 dans N)
- v0(#) est l'égalité dans N (qui est bien une fonction de N^2 dans {0,1})

Dans cette valuation, $v0(a \oplus b) = v0(\oplus)(v0(a),v0(b)) = +(12,24) = 36$
et $v0((a \oplus b) \# ((b \oplus c) \oplus c)) = v0(\#)(v0(a \oplus b),v0((b \oplus c) \oplus c))) = [=(36,96)] = 0$, autrement dit cette formule atomique est fausse dans cette interprétation/valuation.

ii. <R,v1> où R est l'ensemble des réels et v1 est définie par :

- $v1(a) = \pi$, $v1(b) = 2/3$ et $v1(c) = \sqrt{1/2}$
- v1($\oplus$) est la multiplication sur R (qui est bien une fonction de R^2 dans R)
- v1(#) est la relation > dans R (qui est bien une fonction de R^2 dans {0,1})

Dans cette valuation, $v1(a \oplus b) = v1(\oplus)(v1(a),v1(b)) = *(\pi,2/3) = 2\pi/3$
et $v1((a \oplus b) \# ((b \oplus c) \oplus c)) = v1(\#)(v1(a \oplus b),v1((b \oplus c) \oplus c))) = [>(2\pi/3,1/3)] = 1$, autrement dit cette formule atomique est vraie dans cette interprétation/valuation.

iii. <A,v2> où A est l'ensemble des séquences de lettres de l'alphabet français et v2 est définie par :

- v2(a) = « zé», v2(b) = « bu » et v2(c) = « zo »
- v2($\oplus$) est la concaténation[2] sur A noté . (qui est bien une fonction de R^2 dans R)
- v2(#) est la relation $\neq$ dans A (qui est bien une fonction de R^2 dans {0,1})

Dans cette valuation, $v2(a \oplus b) = v2(\oplus)(v2(a),v2(b))$ = zé.bu = zébu
et $v2((a \oplus b) \# ((b \oplus c) \oplus c)) = v2(\#)(v2(a \oplus b),v2((b \oplus c) \oplus c)))$ = [$\neq$ (zébu,buzozo)] = 1, autrement dit cette formule atomique est vraie dans cette interprétation/valuation.

On voit donc que le rôle d'une valuation est de donner un sens à chaque constante, variable, terme et prédicat, de manière à pouvoir affirmer sous une interprétation particulière que telle ou telle formule atomique est vraie ou fausse. Il ne reste plus qu'à préciser comment on obtient la valeur de vérité d'une formule complexe à partir de celle de ses composantes (qui sont des formules atomiques reliées par des connecteurs).

Interprétation d'une formule complexe

Soit <D,v> une valuation (ou interprétation) telle que décrite ci-dessus. De plus, v doit respecter les contraintes énoncées ci-dessous :

Définition 13 :

- $v[\bot] = 0$
- $v[\neg B] = 1 - v[B]$
- $v[(B \wedge C)] = \min(v[B], v[C]))$ $= v[B] * v[C]$

[1] Attention, ceci est abusif si D est infini non-dénombrable. Ce dont on a besoin, c'est de disposer dans l'alphabet d'un symbole de constante pour représenter chaque constante de D effectivement utilisée (en général en nombre fini ou au pire dénombrable).
[2] Concaténation : mise bout à bout. La concaténation des mots « parle » et « ment » donne le mot « parlement ».

- $v[(B \vee C)]$ = $\max(v[B], v[C]))$ = $1-((1-v[B]) * (1-v[C]))$
- $v[(B \rightarrow C)]$ = $\max(1-v[B],v[C])$
- $v[(B \leftrightarrow C)]$ = $v[(B \rightarrow C) \wedge (C \rightarrow B)]$ = $1-\max(v[A]-v[B],v[B]-v[A])$
- $v[\forall xA]$ = $\min(v(A[d/x])$: pour tout $d \in D)$ – n'est vraie que si vraie pour tout d
- $v[\exists xA]$ = $\max(v(A[d/x])$: pour tout $d \in D)$ – n'est vraie que si vraie pour au moins un d

On fera dorénavant (et abusivement) la confusion entre la valuation <D,v> et la seule fonction v.

Remarque : la formule $p(x) \vee \neg p(x)$ semble devoir toujours être vraie, quelque soit l'interprétation que l'on peut faire de la variable x et du prédicat p, or si le domaine D pouvait être vide p(x) serait indéfini et il serait techniquement compliqué d'affirmer que $p(x) \vee \neg p(x)$ est défini. C'est pour éviter cette difficulté que l'on impose la non-vacuité de D.

Définition 14 : Soit A une formule,

- un **modèle** de A est une valuation v telle que $v[A] = 1$
  on dit que **v satisfait A**
- un **contre-modèle** de A est une valuation v telle que $v[A] = 0$
  on dit que **v falsifie A**

Soit H un ensemble de formules, et v une valuation

- un **modèle** de H est une valuation v telle que $v[B] = 1$ pour tout $B \in H$
  on dit que **v satisfait H**
- un **contre-modèle** de H est une valuation v telle que $v[B] = 0$ pour au moins un $B \in H$
  on dit que **v falsifie H**

Comme pour la logique des propositions, on a les définitions suivantes :

Définition 15 : Soit A une formule,

- A est **valide** ssi pour toute valuation v, on a $v[A]=1$
  Autrement dit, A est valide ssi toute valuation est un modèle de A
- A est **satisfiable** ssi il existe une valuation v telle que $v[A]=1$
- A est **insatisfiable** ssi il n'existe pas de valuation v telle que $v[A]=1$ (A est **contradictoire**)
  Autrement dit, A est insatisfiable ssi toute valuation est un contre-modèle de A
- A est **invalide** ssi elle n'est pas valide
- Ces définitions s'étendent naturellement à un ensemble de formules.

Remarques

- A est valide ssi $\neg A$ est insatisfiable
- A est invalide ssi $\neg A$ est satisfiable

Notation Le fait que A soit valide sera noté $\models A$.

Théorème 11 $\models A$ si et seulement si sa clôture universelle est valide (idem pour la clôture existentielle).

Définition 16 : Les définitions de conséquence logique et d'équivalence logique sont exactement les mêmes que celles concernant la logique des propositions. Les notations sont également les mêmes : $H \models C$ (C est conséquence logique de H) et $B \equiv C$ (B et C sont logiquement équivalentes).

Exercices :
Prouvez que les formules suivantes sont valides :
1 $\forall x\, P(x) \leftrightarrow \neg \exists x \neg P(x)$
2 $\forall x\, (A \wedge B) \leftrightarrow (\forall x\, A \wedge \forall x\, B)$ …
Soit A' une variante alphabétique de A loin de var(A) : montrez que $A \equiv A'$.
Prouvez que les formules suivantes ne sont pas valides, en en donnant un contre-modèle :
$\exists x\, (A \wedge B) \leftrightarrow (\exists x\, A \wedge \exists x\, B)$ …
Donnez un modèle de la formule suivante :
$(\forall x \exists y\, x\#y) \wedge (\forall x\, (\neg\, x\#x)) \wedge (\forall x \forall y \forall z\, ((x\#y \wedge y\#z) \rightarrow x\#z))$
Montrez que A est satisfiable si et seulement si sa clôture universelle (resp. existentielle) l'est.

B La théorie de la preuve

Problème : |=A (A est valide) ssi A est vraie dans toute valuation, or le nombre de valuation est infini (et même infini non-dénombrable) et ce, contrairement au cas de la logique des propositions où le nombre de valuations possibles pour une formule est fini et donc, susceptible d'un examen systématique comme c'est le cas dans la méthode des arbres sémantiques ou dans les tables de vérité).
Plutôt que de chercher à vérifier si A est valide pour toute valuation, on va chercher à en donner une preuve en déduction naturelle. On garde tout ce qui vient de la déduction naturelle en logique des propositions (définition d'une preuve comme étant un arbre… , des hypothèses principales, des hypothèses temporaires, les règles d'introduction et d'élimination des connecteurs $\wedge$, $\vee$, $\rightarrow$, $\perp$, $\neg$, $\leftrightarrow$ puisque ceux-ci sont interprétés de la même façon qu'en logique des propositions). A cela, il est bien sûr nécessaire d'ajouter des règles d'introduction et d'élimination pour $\forall$ et $\exists$. Attention ces nouvelles règles devront donner naissance à un système adéquat et complet : elles ne doivent pas permettre d'établir la preuve d'une formule non valide (c'est-à-dire pour laquelle il existe une structure dans laquelle elle est fausse) mais être suffisamment riche pour permettre de prouver toute formule valide. Les voici, de la plus simple à la plus complexe :

1 La déduction naturelle en logique des prédicats

Afin d'en faciliter la présentation, on introduit ici quelques notations :
Soit A une formule obtenue dans une preuve en DN, on note HYP(A) l'ensembles des hypothèses non-déchargées utilisées pour obtenir A, c'est-à-dire des hypothèses non-déchargées qui sont des feuilles de la preuve dont A est la conclusion. Et on notera Free(HYP(A)) l'ensemble des variables libres des hypothèses non-déchargées dont dépend A, en symboles : $\mathrm{Free}(\mathrm{HYP}(A)) = \bigcup_{B\in \mathrm{HYP}(A)} \mathrm{Free}(B)$.

Les règles de déduction naturelle

Introduction du $\exists$:

$$\frac{A[t/x]}{\exists x\, A}\,(I\exists)$$

Commentaire : si on a A[t/x] comme hypothèse ou comme conclusion intermédiaire (ce qui signifie qu'on a prouvé A pour la valeur t de x), alors on est en droit d'en conclure qu'il existe une valeur de x pour laquelle A est prouvée.

Elimination du $\forall$:

$$\frac{\forall x\, A}{A[t/x]}\,(E\forall)$$

Commentaire : si on a $\forall$x A comme hypothèse ou comme conclusion intermédiaire alors on est en droit de conclure que A est prouvée pour toute valeur particulière t **A CONDITION TOUTEFOIS** que [t/x] soit saine dans A. En effet, si l'on ne respecte pas cette restriction, on peut aboutir à des aberrations, sans elle la règle n'est pas adéquate comme on peut le vérifier dans le (mauvais) exemple ci-dessous :

$$\dfrac{\dfrac{\forall x\exists y\,(x\neq y)\ (1)}{(\exists y\,(x\neq y))[y/x]}\,(E\forall) \qquad \exists y\,(y\neq y)}{\forall x\exists y\,(x\neq y) \rightarrow (\exists y\ y\neq y)}\,(I\rightarrow)\ (1)$$

a) hypothèse vraie dans **N**
b) élimination abusive car [y/x] n'est pas saine dans $\exists$y (x≠y)
c) qui se réécrit en la ligne d'en dessous
d) conclusion fausse dans **N**

Ici l'étape b) est incorrecte.

Signalons toutefois le cas particulier, très souvent utilisé, où il n'y a rien à vérifier et qu'on appellera (E$\forall$') :

$$\frac{\forall x\, A}{A}\,(E\forall')$$

Commentaire : ici, A = A[x/x], et x est toujours libre pour x dans A, pour toute formule A.

Comme nous l'avions annoncé, on peut remplacer la règle (E$\forall$) par la suivante :

$$\frac{\forall x\, A}{A'[t/x]}\,(E\forall)$$

Commentaire : A' est une variante alphabétique de A loin de var(t).

Si l'on reprend le « mauvais exemple » donné ci-dessus, voyons comment l'usage de variantes alphabétiques empêche d'engendrer le mauvais raisonnement signalé ci-dessus :

$\forall x \exists y\ (x \neq y)$ (1)	a) hypothèse vraie dans **N**
$\forall x \exists u\ (x \neq u)$ (1)	b) variante alphabétique loin de var(y) préparant la substitution [y/x]
——— (E∀)	c) élimination correcte puisqu'on travaille sur une variante
$(\exists u\ (x \neq u))[y/x]$	d) qui se réécrit en la ligne d'en dessous
$\exists u\ (y \neq u)$	
——— (I→) (1)	
$\forall x \exists y\ (x \neq y) \rightarrow (\exists u\ y \neq u))$	e) conclusion vraie dans **N**, on ne peut pas engendrer d'erreur !

Introduction du ∀ :

$$\frac{A}{\forall x\ A}\ (E\forall)$$

Commentaire : si on a A comme hypothèse ou comme conclusion intermédiaire, à quelles conditions est-on en droit de conclure que $\forall x\ A$ est prouvée ? A la condition que x représente une valeur quelconque. Ceci est pris en compte par la contrainte suivante sur l'applicabilité de la règle : x ne doit pas être une variable libre d'une hypothèse non-déchargée dont dépend A. Autrement dit, **la règle est correcte ssi $x \notin$ Free(HYP(A))**. A nouveau, si l'on ne respecte pas cette restriction, on peut aboutir à des aberrations, sans elle la règle n'est pas adéquate comme on peut le vérifier dans l'exemple ci-dessous :

$x=0$ (1)	a) hypothèse parfois vraie dans **N** (dans le cas précis où x vaut 0)
——— (I∀)	b) introduction abusive car x est libre dans x=0
$(\forall x\ x=0)$	
——— (I→) (1)	
$(x=0 \rightarrow (\forall x\ x=0))$	
——— (I∀)	c) introduction justifiée cette fois car il n'y a plus d'hypothèses non déchargées
$(\forall x\ (x=0 \rightarrow (\forall x\ x=0)))$	
——— (E∀)	d) élimination justifiée car 0 est libre pour x dans
$(0=0 \rightarrow (\forall x\ x=0))$	e) formule bien sûr toujours fausse dans **N**

Ici l'étape b) est incorrecte.

Et enfin, la plus délicate à énoncer :
Elimination du ∃ :

$$\frac{\exists x\ A \qquad \begin{matrix} A\ (i) \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ C \end{matrix}}{C}\ (E\exists)(i)$$

Commentaire : à condition que x ne soit pas une variable libre de C, ni d'une hypothèse non-déchargée –et autre que A– dont dépend C. Autrement dit, la règle est correcte ssi $x \notin \mathrm{Free}(C) \cup \mathrm{Free}(\mathrm{HYP}(C) \setminus A)$: on ne considère que les hypothèses non-déchargées autres que A (Rappel HYP(C)\A est l'ensemble HYP(C) privé de l'élément A) : bref, x doit être « quelconque ».

Attention, ici on décharge l'hypothèse (i) que l'on aura posée temporairement, le temps d'obtenir C.

On pourrait être tenté de proposer la règle suivante, plus simple, malheureusement, elle n'est pas adéquate :

$$\frac{\exists x\ A}{A[y/x\ ]}\ (E\exists')$$

Avec pour justification que si on a $\exists x\ A$ comme hypothèse ou conclusion intermédiaire, il existe une valeur pour laquelle c'est vrai. Appelons-la y, et on obtient alors une preuve de A[y/x]. Mais avec une telle règle, on pourrait obtenir une preuve de la formule $\exists x\ A \rightarrow A[y/x]$ qui n'est pas valide.

Des exemples

a) Formalisez l'argument « Les humains sont mortels, or Socrate est humain, donc Socrate est mortel » en logique des prédicats, puis donnez-en une preuve en déduction naturelle.
On introduit les prédicats :
h(x) se lisant « x est humain » et m(x) se lisant « x est mortel »
et la constante s se lisant « Socrate »
L'argument est formalisé par : $\forall x(h(x) \rightarrow m(x))$, h(s) |-- ? m(s)

$\forall x(h(x) \rightarrow m(x))$ (1)
———— (E∀) –[s/x] saine dans (1)
$h(s) \rightarrow m(s)$ h(s)(2)
———————— (E→)
m(s)

b) |--- ? $(\forall x\, \forall y\, A \rightarrow \forall x\, \forall y\, A)$

$\forall x\, \forall y\, A$(1) -- car membre gauche d'une implication
———— (E∀')
$\forall y\, A$ -- cas particulier de (E∀) sans condition d'application
———— (E∀')
A -- cas particulier de (E∀) sans condition d'application
———— (I∀)
$\forall x\, A$ -- Ok car x ∉ Free(HYP(A))
———— (I∀)
$\forall y\, \forall x\, A$ -- Ok car y ∉ Free(HYP(∀x A))
———————— (I→) (1)
$(\forall x\, \forall y\, A \rightarrow \forall x\, \forall y\, A)$

c) |--- ? $(\exists x\, \exists y\, A \rightarrow \exists x\, \exists y\, A)$

A (3)
—— (I∃)
$\exists y\, A$
—— (I∃)
$\exists y\, \exists x\, A$
$\exists y\, A$ (2)
———————————— (E∃) (3) OK car x∉Free(∃x ∃y A) et x ∉ Free(HYP(∃x ∃y A) \ A)
$\exists y\, \exists x\, A$
$\exists x\, \exists y\, A$ (1)
———————————— (E∃) (2) OK car x∉Free(∃x ∃y A) et x ∉ Free(HYP(∃x ∃y A) \ ∃y A)
$\exists y\, \exists x\, A$

d) |--- ? $\exists x\, A \rightarrow \neg\forall x \neg A$

A (2) $\forall x \neg A$ (3) -- raisonnement par l'absurde
———— (E∀')
$\neg A$
———————— (I⊥)
$\perp$
———————— (I¬) (3)
$\neg\forall x \neg A$
$\exists x\, A$ (1)
———————————— (E∃) (2) -- car x ∉ Free(¬∀x ¬ A) et x∉ Free(HYP(¬∀x ¬ A)\A)
$\neg\forall x \neg A$

Traitement du prédicat =

Les règles données jusqu'à maintenant s'applique à tous les prédicats sans distinction, on a besoin pour un prédicat comme = de donner des règles spécifiques à ce prédicat afin de spécifier en quoi il n'est pas un prédicat quelconque. Les règles le concernant sont :

$$\frac{}{t = t}\ (I=)$$

Pour tout terme t. Remarque : cette règle s'applique sans numérateur car il n'y a besoin de rien pour l'appliquer.

$$\frac{t1 = t2 \qquad A[t1 / x]}{A[t2 / x]}\ (E=)$$

Pour tous termes t1 et t2. Commentaire : cette règle exprime que si A est prouvée pour t1 et que t1=t2 alors A est prouvée pour t2.

Encore des exemples

e) Muni des règles concernant le prédicat =, on peut démontrer que |-- t1 = t2 → t2 = t1, autrement dit, que le prédicat = est symétrique.

$$\dfrac{\dfrac{t1 = t2\ (1) \qquad \dfrac{}{t1 = t1}\ (I=)}{(x = t1)\,[t2/x]}\ (E=)}{t1 = t2 \rightarrow t2 = t1}\ (I\rightarrow)(1)$$

Or t1=t1 est identique à (x=t1)[t1/x] : la prémisse t1 = t1 s'écrit (x = t1) [t1/x].

La conclusion (x = t1) [t2/x] est identique à t2 = t1.

Exercice : à l'aide des règles concernant le prédicat =, montrez qu'il est transitif.

2 Les séquents pour la logique des prédicats

De même que pour la logique des propositions, une approche alternative à la déduction naturelle existe, c'est une extension des séquents. Toute les définitions sont reprises à l'identique, on ajoute simplement des règles permettant de gérer les quantificateurs, dans le groupe de règles dites logique.

$$\frac{\Gamma, A[t/x] \vdash \Delta}{\Gamma, \forall x A \vdash \Delta}\ (\forall\text{ - gauche}) \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash \forall x A, \Delta}\ (\forall\text{ - droite})$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, \exists x A \vdash \Delta}\ (\exists\text{ - gauche}) \qquad \frac{\Gamma \vdash A[t/x], \Delta}{\Gamma \vdash \exists x A, \Delta}\ (\exists\text{ - droite})$$

Les règles de quantificateurs sont soumises à des restrictions : dans les règles de ∀-droite et ∃-gauche, on demande que la variable *x* ne figure dans aucune des formules de Γ et Δ. On impose que la substitution [t / x] est saine.

C Le principe de résolution

1 Forme normale de Skolem

Forme normale prénexe

Il est souhaitable d'étendre le principe de résolution vu en partie propositionnelle au cas de la logique des prédicats car ce principe est particulièrement adapté à un traitement automatique. Son extension à la logique des prédicats étant d'ailleurs le fondement théorique du langage déclaratif PROLOG.

Le premier obstacle est la définition d'une FNC. Pour y parvenir, il faut tout d'abord transformer une formule en une formule équivalente dans laquelle tous les quantificateurs sont en tête, qu'on appelle forme normale prénexe (FNP). Pour cela on utilise les équivalences suivantes :
$\forall x\ (A \wedge B) \equiv (\forall xA \wedge \forall xB)$ et $\exists x\ (A \vee B) \equiv (\exists xA \vee \exists xB)$ qui permettent de « factoriser » un $\forall$ou un $\exists$ (resp. devant une conjonction ou une disjonction). Pour les autres cas de figure, on utilisera i) $Qx\ (A*B) \equiv (QxA*B)$ si $x \notin Free(B)$ avec $Q=\forall$ ou $\exists$.

Ainsi, le traitement d'une formule de la forme : $\forall xA \vee\ B$ requiert un renommage préalable de x dans $\forall xA$ et son remplacement par une variable n'apparaissant pas dans B afin de pouvoir utiliser i) ce qui donne $\forall yA'[y/x] \vee B$ où A' est une variante de A loin de y et $y \notin Free(B)$.
La variable y n'étant pas libre dans B, on obtient $\forall y\ (A'[y/x] \vee\ B)$ en vertu de i) qui est équivalente à $\forall xA \vee B$. En procédant de même pour les cas de la forme : $B \vee \forall xA$, $B \wedge \exists xA$, $\exists xA \wedge B$ (après avoir remplacé les $\rightarrow$, les $\leftrightarrow$, ... par des conjonctions et des disjonctions) on parvient à factoriser les quantificateurs en tête de formule, le reste de la formule, ne contenant pas de quantificateurs, peut donc être mis en FNC de la même manière que dans le cas propositionnel, par conséquent :

Théorème 12 Soit A une formule du langage du premier ordre, il existe une formule A' équivalente à A, de la forme $Q_1x_1...Q_nx_n\ M$ où les Q_i sont des quantificateurs et M, qu'on appelle matrice, ne contient aucun quantificateur et est en FNC. On dit que la formule A' est une forme normale prénexe de A (cette FNP n'est pas unique à cause de l'ordre dans lequel on factorise les quantificateurs).

Forme normale de Skolem

Pour introduire les formes normales de Skolem, nous allons utiliser les fonctions des ensembles FON_n de l'alphabet de la logique des prédicats. L'idée (car nous n'allons pas outre mesure entrer dans les détails) est la suivante : considérons une formule de la forme $\forall x \exists y\ A$, elle exprime (du point de vue des modèles) que pour toute valeur vx attribuée à x, il y a une valeur vy attribuée à y telle que $A[vx/x,vy/y]$ soit vraie. Or vy dépend évidemment de vx, c'est pour garder trace de cette dépendance que sont utilisées les fonctions de Skolem, la sémantique d'une telle fonction est de « choisir » pour chaque valeur vx une valeur vy à attribuer à y telle que $A[vx/x,vy/y]$ soit vraie.

Définition 17 : Soit $Q_1x_1...Q_nx_n\ M$ une formule en FNP, sa forme normale de Skolem (FNS) est définie via une fonction F définie inductivement : $FNS(Q_1x_1...Q_nx_n\ M)=F(Q_1x_1...Q_nx_n\ M, liste\_vide)$ et

- $F(\forall x_i\ Reste, liste) = (\forall x_i\ F(Reste, ajout\text{-}en\text{-}queue(liste, x_i))$ -- NB : ajout-en-tête fait aussi bien l'affaire, on ajoute x_i à la liste des variables universelles déjà identifiées et dont vont dépendre les existentielles.

- $F(\exists x_i\ Reste, (x_{i1},...x_{ik})) = F(Reste\ [f_i\ (x_{i1},...x_{ik}) / x_i]\ , (x_{i1},...x_{ik}))$ où f_i est une « nouvelle »[3] fonction, et x_i est remplacée par un terme contenant les variables universelles qui la précédaient dans la formule.

- $F(Reste) = Reste$ si $Reste$ n'est pas préfixé de quantificateurs.

Malheureusement, si A est en FNP, il se trouve que $FNS(A)$ ne lui est pas équivalente (exercice : vérifiez que $\forall x \exists y P(x,y)$ n'est pas équivalente à $\forall x P(x,f(x))$ qui est la FNS de la première, pour cela, trouvez une interprétation dans laquelle l'une est vraie et pas l'autre). Et donc, ce n'est pas parce que l'une est valide que l'autre l'est !!! Puis vérifiez que si l'une est satisfiable l'autre l'est aussi : heureusement, la skolémisation si elle ne préserve pas la validité, préserve la satisfiabilité.

Théorème 13 Une formule A est satisfiable si et seulement si $FNS(A)$ l'est. Pour la validité, on procèdera donc par réfutation : A valide ssi $\neg A$ insatisfiable ssi $FNS(\neg A)$ insatisfiable.

Définition (Fragments): Plus qu'en logique des propositions, on est souvent amenés à ne pas considérer l'intégralité des formules, mais uniquement un sous-ensemble de celles-ci qu'on appelle alors un fragment (ou champ de production). Citons parmi les fragments connus :

[3] = pas déjà utilisée dans Reste.

c) le fragment monadique (PRED = PRED_1 : il n'y a que des prédicats à UN argument), satisfiabilité NP-complet.
d) le fragment $\exists^*\forall^*$ (= une suite de $\exists$ suivi d'une suite de $\forall$, aussi connue sous le nom de classe de Bernays-Schönfinkel) dont les formules sont de la forme $\exists x1,\ldots,\exists xn.\ \forall y1,\ldots,\forall ym.\ A)$ où A ne contient aucun quantificateur et dont les termes sont des variables (pas de fonctions) : satisfiabilité NEXPTIME-complète. Une fois mises en FNS, les formules de ce fragment engendrent des clauses dont les termes sont écrits sans fonctions. Le fragment clauses de Horn de ce fragment est décidable en temps doublement exponentiel (DEXPTIME).
e) la classe d'Ackermann : $\exists^*\forall\exists^*$ (un seul $\forall$ au milieu de $\exists$), pas de fonctions : DEXPTIME.
f) le fragment à deux variables (décidable), le fragment à trois variables (indécidable).
g) le fragment gardé (décidable NEXPTIME)

Clauses, résolvantes, unification, factorisation, clauses de Horn, complétude résolution binaire avec clauses de Horn, et binaire+factorisation avec clauses générales.

Equivalences remarquables (Rappel **:** (A⊕B) est une abréviation de ((A∨B)∧¬(A∧B)), ⊕ représente le ou exclusif, et (A↔B) est une abréviation de ((A→B) ∧ (B→A)) et une ⊤ abréviation de ¬⊥.

Règles pour le calcul des tableaux Rappel **:** H |= A ssi Tab(H∪¬A) est fermé. Si le tableau est ouvert, une de ses branches l'est donc, et elle fournit un modèle de (H∪¬A).

Règle	**Un noeud contient**	**Faire**
Fermeture	⊥	Fermer la branche contenant ce nœud
Règle	**Une branche contient**	**Créer (à la feuille de cette branche)**
⊥	A et ¬A	Un fils contenant ⊥
Règle	**Un nœud contient**	**Créer (à CHAQUE feuille de CHAQUE branche (non fermée) passant par ce nœud)**
∧	(A∧B)	Un fils contenant A et B
¬∨	¬(A∨B)	Un fils contenant ¬A et ¬B
¬→	¬(A→B)	Un fils contenant A et ¬B
↔	(A↔B)	Un fils contenant (A→B) et (B→A)
¬¬	¬¬A	Un fils contenant A
∨	(A∨B)	Deux fils contenant l'un A et l'autre B
¬∧	¬(A∧B)	Deux fils contenant l'un ¬A et l'autre ¬B
→	(A→B)	Deux fils contenant l'un ¬A et l'autre B
¬↔	¬ (A↔B)	Deux fils contenant l'un ¬(A→B) et l'autre ¬(B→A)

Définitions (H est un ensemble de formules, A une formule, v une valuation) : 1) v est un **modèle** de H ssi v(B)=1 pour tout B de H., 2) H est **satisfiable** ssi il existe un modèle de H, 3) A est une **conséquence logique** de H, noté H |= A, ssi tout modèle de H est un modèle de A, 4) H est **insatisfiable** ssi il n'est pas satisfiable, 5) A est **valide**, noté |= A, ssi toute valuation est un modèle de A, 6) A est **déductible** de H, noté H |-- A, ssi il existe une preuve en déduction naturelle de A à partir des hypothèses (dites principales) de H.

Déduction naturelle

Logique des propositions (LP0)	
A∧B ------- (E∧) A (resp. B)	A→B A ----------------- (E→) B
¬¬A ------- (E¬) A	⊥ ------- (E⊥) A
A (j) B (k) … … A∨B C C -------------------------------------- (E∨)(j,k) C	
A B -------------- (I∧) A∧B (resp. B∧A)	A ------- (I∨) A∨B (resp. B∨A)
A ¬A --------------- (I⊥) ⊥	
A (i) … B -------- (I→)(i) A→B	A (i) … ⊥ ---------- (I¬)(i) ¬A

Logique des prédicats (LP1)
∀xA ---- (E∀) si [t/x] saine A[t/x]
A[t/x] ---- (I∃) si [t/x] saine ∃xA
A (i) … si x∉Free(Hyp(C) \ A) ∃xA C et x∉Free(C) ------------- (E∃)(i) C
A ---- (I∀) si x∉Free(Hyp(A)) ∀x A

Règle additionnelle de réutilisation d'un théorème déjà démontré
------- (I |--) si |-- A A

Rappel : lorsqu'on décharge une hypothèse c'est dans **toutes** les feuilles du sous-arbre concerné et seulement dans celles-ci ! Par ailleurs, on peut très bien décharger plusieurs fois la même hypothèse.

NB : ↔ se traite comme une conjonction, ainsi :

A→B B→A ---------------------- (I∧) A↔B	A↔B ------- (E∧) A→B (resp. B→A)

Résolution

Soit C_1= {p,R_1} et C_2={¬p,R_2} deux clauses, dites complémentaires, la clause Rés(C_1,C_2)={R_1,R_2} est une résolvante de C_1 et C_2. **Cas particulier** : la résolvante de deux clauses {p} et {¬p} est donc {}=∅ (clause vide) qui est notée ⊥. **Attention** ! La résolvante de deux clauses ayant DEUX littéraux complémentaires est une clause tautologique (équivalente à T), et n'a aucun intérêt (T est trivialement conséquence logique de ces clauses). Exemple : {q,¬q,R_1,R_2}≡{T} est une résolvante de C_1= {p,q,R_1} et C_2={¬p,¬q,R_2}.
Soit E_0 un ensemble de clauses, soit E_{i+1}= E_i ∪ {Rés(C_1,C_2) non tautologique/ C_1,C_2 ∈E_i et C_1,C_2 sont complémentaires}. **Théorème** : E_0 est insatisfiable si et seulement si il existe n tel que E_n contient ⊥.

Equivalences remarquables en logique des prédicats.
Indifféremment Q dénote ∀ ou ∃, et * dénote ∧ ou ∨. Les substitutions sont supposées saines.

∀x (A∧B) ≡ (∀xA ∧∀xB)	Distributivité de ∀ sur ∧	∃x (A∨B) ≡ (∃xA ∨ ∃xB)	Distributivité de ∃ sur ∨
∀xA ≡ ¬∃x¬A	Dualité	∃xA ≡ ¬∀x¬A	Dualité
QxA ≡ Qy(A[y/x])	Renommage	Qx (A*B) ≡ (QxA*B) si x∉Free(B)	(pour → voir plus bas)
∀x∀y A ≡ ∀y∀x A	Permutation	∃x∃y A ≡ ∃y∃x A	Permutation
Qx A ≡ A si x∉Free(A)	Vacuité	∀x (A→B) ≡ ((∃xA)→B) si x∉Free(B)	-
∀x A → ∃xA	-	∃x (A→B) ≡ ((∀xA)→B) si x∉Free(B)	-