

**IA41**

**Concepts fondamentaux en Intelligence Artificielle  
et langages dédiés**

**CM #5**

**Logique des prédicats du  
premier ordre**

**Fabrice LAURI**

# Logique des prédicats du premier ordre (PP)

- Introduction
- Définitions
- Théorie des modèles
- Théorie de la démonstration
- Propriétés fondamentales de (PP)
- Décidabilité de (PP)
- Principe de résolution

# Logique des prédicats du premier ordre (PP)

- **Introduction**
- **Définitions**
- **Théorie des modèles**
- **Théorie de la démonstration**
- **Propriétés fondamentales de (PP)**
- **Décidabilité de (PP)**
- **Principe de résolution**

# Introduction

(LP) permet de rendre compte du syllogisme suivant :

Si Jean est malade, il ne sort pas	$J \Rightarrow \neg S$
Jean est malade	$J$

---

Jean ne sort pas	$\neg S$
------------------	----------

Mais pas du syllogisme suivant :

Si un homme est malade, il ne sort pas	$M \Rightarrow \neg S$
Jean est un homme malade	$R$

---

Jean ne sort pas	$\neg S'$
------------------	-----------

# Introduction

## Nouveaux concepts dans (PP) :

- prédicats
- variables
- quantificateurs :
  - « quel que soit » :  $\forall$  (ensemble d'objets)
  - « il existe » :  $\exists$  (certains objets d'un ensemble)

## Exemple :

### « *Sébastien a les cheveux longs.* » :

- « *avoir les cheveux longs* » est un *prédicat*
- *Sébastien* est le *sujet du prédicat*
  
- Ce prédicat peut se noter  $p(x)$  : « *x a les cheveux longs* »
- $x$  est une variable, sujet du prédicat, qui peut être remplacée par Sébastien pour rendre  $p$  vrai
- Cet énoncé peut alors s'écrire  $p(\text{Sébastien})$ .

# Introduction

Énoncés quantifiés :

« Tout est éphémère » :  $\forall x E(x)$

« Rien est éphémère » :  $\forall x \neg E(x)$

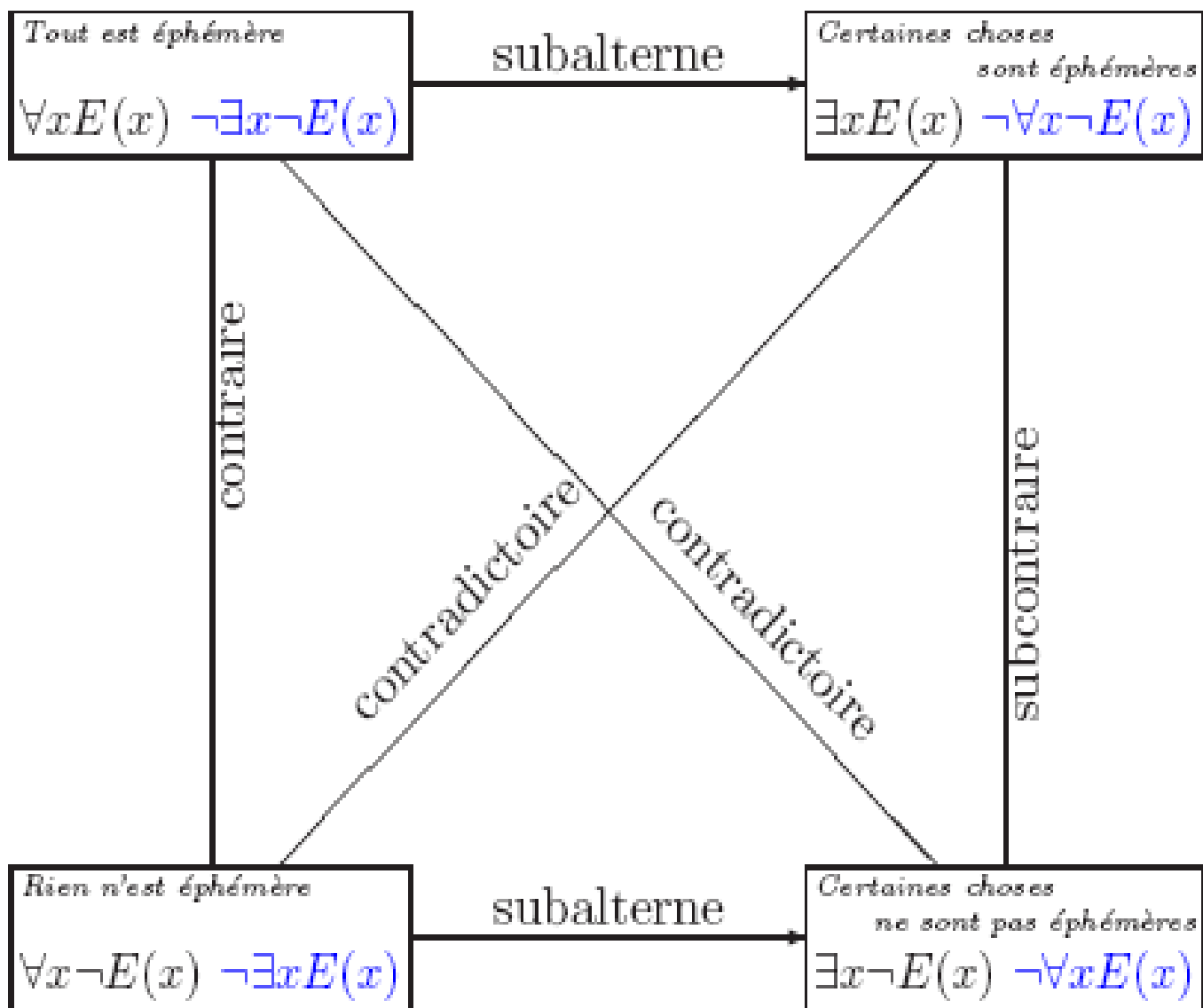
$\exists$  peut se définir à partir de  $\forall$  :

« Rien est éphémère »      « Il n'existe pas de chose éphémère »

$$\forall x \neg E(x) \quad \equiv \quad \neg \exists x E(x)$$

# Introduction

Carré d'opposition d'Aristote :



# Introduction

Notion d'univers du discours :

« *Tous les philosophes sont assis.* »

$$\forall x (P(x) \Rightarrow A(x))$$

Bonne formulation

$$\forall x (P(x) \wedge A(x))$$

Mauvaise formulation

« *Quelques philosophes sont assis.* »

$$\exists x (P(x) \wedge A(x))$$

Bonne formulation

$$\exists x (P(x) \Rightarrow A(x))$$

Mauvaise formulation

# Logique des prédicats du premier ordre (PP)

- Introduction
- Définitions
- Théorie des modèles
- Théorie de la démonstration
- Propriétés fondamentales de (PP)
- Décidabilité de (PP)
- Principe de résolution

# Définition de (PP)

Une théorie du premier ordre consiste en :

- un **alphabet**
- un **langage** ou ensemble de mots formés à partir de l'alphabet
- un ensemble d'**axiomes**
- un ensemble de **règles d'inférence**

# Alphabet de (PP) (1/5)

Il est constitué des ensembles disjoints suivants :

- **les constantes d'individus** : chaînes alphanumériques commençant par une lettre minuscule (*underscore* autorisé)

Exemples : *socrate*, *{ a, b, c, ... }*, *127*, *x\_20*

Les constantes représentent des entités individuelles.

- **les symboles de fonctions** : un foncteur est une chaîne alphanumérique commençant par une minuscule, comme *f*, *g*, *h*, ...

Chaque foncteur possède une arité, qui est le nombre d'arguments auxquels le foncteur peut s'appliquer.

Exemples : - *carré( x )* d'arité 1,  
- *plus( 3, 4.2 )* d'arité 2

**Remarque** : les constantes peuvent donc être considérées comme des foncteurs d'arité 0.

# Alphabet de (PP) (2/5)

Les foncteurs servent à construire les termes du langage.

Terme :  $\underbrace{\langle \text{symbole fonctionnel} \rangle}_{\text{Arité } n} ( \underbrace{\langle \text{nom\_indiv}_1 \rangle, \dots, \langle \text{nom\_indiv}_n \rangle}_{\text{Chacun de ces objets peut lui-même être composé}} )$

- **les symboles de variables** : chaînes alphanumériques commençant par une majuscule, comme  $\{ X, Y, Z, \dots \}$ .

Une variable peut être substituée par un terme.

# Alphabet de (PP) (3/5)

- **les symboles de prédicat** : permettent de formaliser les relations entre objets. Ce sont des chaînes alphanumériques commençant par une minuscule, comme p, q, r...

A un symbole de prédicat est associé une arité, qui indique le nombre de termes auxquels il s'applique.

Exemples : - *aime( X, femme( X ) )* : X aime sa femme

Nom de la femme  
de l'objet X

- *aime( X, chocolat )* : X aime le chocolat

- *p( f( g( X ) ), a, h( X, v( a ) ) )*

# Alphabet de (PP) (4/5)

## - les connecteurs

$\neg$  : négation

$\wedge$  : conjonction

$\vee$  : disjonction

$\Rightarrow$  : implique

$\Leftrightarrow$  : est équivalent

- **les symboles de ponctuation** { ( , ) } utilisés pour améliorer la lisibilité et lever les ambiguïtés

## - les quantificateurs

$\forall$  : universel

$\exists$  : existentiel

# Alphabet de (PP) (5/5)

**Attention au risque d'ambiguïté entre symboles de prédicat et symboles de fonction.**

Exemple : *père( georges )*

Cela peut être interprétée comme la fonction qui, à l'objet *georges*, associe le nom de son père.

Mais cela peut également exprimer le fait que son argument est ou n'est pas père.

La différence s'établit au niveau syntaxique :

- un symbole de fonction ne peut jamais apparaître en tête d'un atome et peut apparaître à des niveaux plus internes,
- un symbole de prédicat ne peut apparaître qu'en tête d'un atome.

# Le langage (PP) (1/7)

- **les termes** sont définis par les règles suivantes :

t1) toute variable est un terme

t2) toute constante est un terme

t3) si  $t_1, t_2, \dots, t_n$  sont des termes et  $f$  est un symbole fonctionnel d'arité  $n$ , alors  $f(t_1, t_2, \dots, t_n)$  est un terme

Exemples : *127* (*constante*)

*ALPHA\_3* (*variable*)

*plus(3, x)* (*terme structuré*)

} Objets (complexes)

- **les atomes** (littéral ou formule atomique) sont une construction syntaxique de la forme :  $p(t_1, t_2, \dots, t_n)$  où  $p$  est un symbole de prédicat d'arité  $n$  et où  $t_1, t_2, \dots, t_n$  sont des termes.

Exemples : *homme( Socrate )*

*précède( 5, -2 )*

} Propriétés des objets

# Le langage (PP) (2/7)

- **les formules** lient des atomes à l'aide des connecteurs, selon les règles suivantes :

f1) tout atome est une formule

f2) si  $\varphi$  et  $\psi$  sont deux formules, alors :

$(\neg\varphi)$ ,  $(\varphi \wedge \psi)$ ,  $(\varphi \vee \psi)$ ,  $(\varphi \Rightarrow \psi)$ ,  $(\varphi \Leftrightarrow \psi)$  sont des formules

f3) si  $\varphi$  est une formule et  $X$  une variable, alors :

$(\forall X \varphi)$ ,  $(\exists X \varphi)$  sont des formules.

Le langage du premier ordre est l'ensemble des formules correctement construites à partir des symboles de son alphabet.

# Le langage (PP) (3/7)

## Remarques :

- La formule  $(\forall X \varphi)$  permet de signifier que tout objet substituable à la variable  $X$  possède la propriété décrite par la formule  $\varphi$ .
- La formule  $(\exists X \varphi)$  permet de signifier qu'il existe au moins un objet substituable à la variable  $X$  qui possède la propriété décrite par la formule  $\varphi$ .

$X$  = variable quantifiée (ou variable liée par un quantificateur).

# Le langage (PP) (4/7)

## Définitions :

- La **portée** de  $\forall X$  (respectivement  $\exists X$ ) dans la formule  $(\forall X \varphi)$  (respectivement  $(\exists X \varphi)$ ) est la formule  $\varphi$ .
- Une **occurrence liée** d'une variable dans une formule est l'occurrence suivant immédiatement le quantificateur ou une occurrence dans la portée du quantificateur. Toute autre occurrence est **libre**.

Exemple :  $\varphi = \exists X p(X) \wedge q( g( y(a, f( X ) ) ) )$

Occurrences liées

Occurrence libre

- Une **formule close** est une formule sans variables libres, comme

$$\forall X \exists Y ( p(X, Y) \vee q(X) )$$

# Le langage (PP) (5/7)

## Définitions (suite) :

- Une **formule terminale** (*ground formula*) est une formule sans aucune variable.

La règle de priorité des connecteurs permet d'alléger l'écriture des formules :

Priorité la plus forte	$\forall, \exists$
	$\neg$
	$\wedge, \vee$
Priorité la plus faible	$\Rightarrow, \Leftrightarrow$



# Le langage (PP) (7/7)

Similitudes avec les langages à structure de blocs :

Bloc A

Var X

Bloc B

Var Y  
Y = X

Variable locale au bloc B = occurrence liée de Y dans B par la déclaration Var Y.

Occurrence libre de X dans B mais liée dans A.

FinBloc B

X = Z

Occurrence libre de Z dans A.

FinBloc A

Occurrence liée de X dans A.

# Description de structures élémentaires dans (PP)

- Expression des propriétés des listes
- Expression des propriétés des arbres binaires

# Expression des propriétés des listes dans (PP) (1/5)

Introduction d'un foncteur d'arité 2 :  $.$  (constructeur de listes).  
Utilisé en notation infixée :  $H.T$  est la liste formé d'un élément  $H$   
et d'une liste  $T$ .

Exemple :  $.(a, nil)$  désigne la liste comportant un seul élément  $a$ ;  
 $nil$  désigne la liste vide.

## 1) Axiomatisation de la définition d'une liste

$L$  est une liste ssi :

- $L$  est la liste vide ou
- $L$  se décompose en un élément suivi d'une liste

$$\forall L (\text{liste}(L) \Leftrightarrow (L = \text{nil}) \vee \exists U \exists L_1 ((L = U.L_1) \wedge (\text{liste}(L_1))))$$

# Expression des propriétés des listes dans (PP) (2/5)

## 2) Prédicat d'appartenance d'un élément à une liste

X appartient à la liste L ssi :

- X est le premier élément de L ou
- X figure dans le reste  $L_1$  de la liste  $Y.L_1$

$$\forall X \forall L (\text{dans}(X, L) \Leftrightarrow \exists L_1 (\text{liste}(L_1) \wedge ((L = X.L_1) \vee \exists Y ((L = Y.L_1) \wedge \text{dans}(L_1, X))))))$$

# Expression des propriétés des listes dans (PP) (3/5)

## 3) Prédicat exprimant la longueur d'une liste

N est la longueur de L ssi :

- L est la liste vide et N est égal à 0 ou
- L s'écrit  $X.L_1$  et N est la longueur de  $L_1$  augmentée de 1.

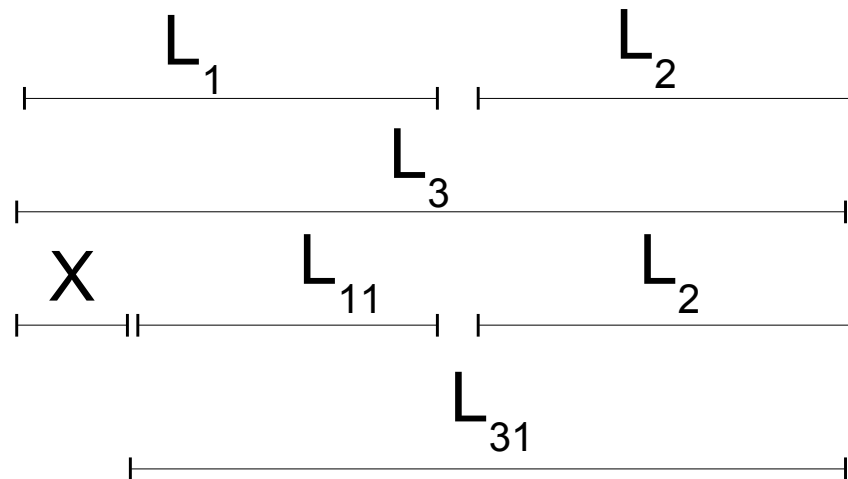
$$\begin{aligned} \forall L \forall N (\text{long}(L, N) \Leftrightarrow & \\ ((L = \text{nil}) \wedge (N = 0)) \vee & \\ \exists X \exists L_1 \exists N_1 ((\text{liste}(L_1) \wedge & \\ ((L = X.L_1) \wedge & \\ (\text{long}(L_1, N_1)) \wedge & \\ (\text{add}(N_1, 1, N)))))) & \end{aligned}$$

# Expression des propriétés des listes dans (PP) (4/5)

## 4) Prédicat exprimant la concaténation de deux listes

$L_3$  est la concaténation de  $L_1$  devant  $L_2$  ssi :

- $L_1$  est la liste vide et  $L_2 = L_3$  ou
- si le reste  $L_{31}$  de  $L_3$  est la concaténation du reste  $L_{11}$  de  $L_1$  devant  $L_2$  et si  $L_3$  est produit par juxtaposition de la tête de  $L_1$  et de  $L_{31}$ .



# Expression des propriétés des listes dans (PP) (5/5)

## 4) Prédicat exprimant la concaténation

$$\begin{aligned} & \forall L_1 \forall L_2 \forall L_3 (\text{conc}(L_1, L_2, L_3) \Leftrightarrow \\ & \quad ((L_3 = \text{nil}) \wedge (L_2 = L_3)) \vee \\ & \quad \exists X \exists L_{11} \exists L_{31} (\text{liste}(L_{11}) \wedge (\text{liste}(L_{31}) \wedge \\ & \quad (L_1 = X.L_{11}) \wedge (L_3 = X.L_{31}) \wedge \text{conc}(L_{11}, L_2, L_{31}))) \end{aligned}$$

# Expression des propriétés des arbres binaires dans (PP) (1/2)

## 1) Axiomatisation de la structure d'arbre binaire

On introduit le foncteur d'arité 3 .

Un arbre binaire peut être vide, auquel cas il est représenté par la constante *nil*.

Sinon, il se décompose en un sous-arbre gauche, une racine et un sous-arbre droit.

D'où la définition :

$$\forall T (\text{arbrebin}( T ) \Leftrightarrow (T = \textit{nil}) \vee$$

$$\exists Tg \exists Td \exists R (\text{arbrebin}( Tg ) \wedge \text{arbrebin}( Td ) \wedge (T = t( Tg, R, Td )))$$

# Expression des propriétés des arbres binaires dans (PP) (2/2)

## 2) Prédicat d'appartenance à un arbre binaire

X est dans T ssi :

- X est la racine de T ou
- X apparaît soit dans le sous-arbre gauche Tg, soit dans le sous-arbre droit Td

$$\forall T \forall X (\text{dans}(T, X) \Leftrightarrow$$

$$\exists Tg \exists Td (T = t(Tg, X, Td)) \vee$$

$$\exists Tg \exists Td \exists R ((T = t(Tg, R, Td)) \wedge (\text{dans}(Tg, X) \vee (\text{dans}(Td, X))))))$$

# Logique des prédicats du premier ordre (PP)

- Introduction
- Définitions
- **Théorie des modèles**
- Théorie de la démonstration
- Propriétés fondamentales de (PP)
- Décidabilité de (PP)
- Principe de résolution

# Théorie des modèles

## Objectif :

Utiliser une méthode permettant de dégager une interprétation sémantique de nos formules (vraies ou fausses ?).

## Problèmes :

Tables de vérité ne peuvent plus être utilisées, car les variables possèdent un **domaine de valeur**.

## Solution :

Utiliser une fonction d'interprétation permettant de donner une valeur de vérité à un prédicat en fonction de la valeur des termes qui le composent.

Exemple : prédicat  $p(X) = \text{« } X \text{ est bleu »}$  opérant sur le domaine de valeurs  $\{ \text{extraterrestre, terrien, schtroumpf} \}$ .

Interprétation possible de  $p...$

# Interprétation des langages du premier ordre (1/6)

## Définition #1 : définition d'une interprétation

Soit  $L$  un langage du premier ordre.

L'interprétation pour le langage  $L$  est définie par :

- a) un ensemble  $D$  non vide = le domaine d'interprétation
- b) A chaque symbole de constante individuelle, on associe un élément de  $D$ .
- c) A chaque symbole de fonction d'arité  $n$  ( $n \geq 1$ ) de  $L$ , on associe une fonction de  $D^n \rightarrow D$ .  
Si  $f$  est ce symbole de fonction, la fonction associée sera noté  $f'$ .
- d) A chaque symbole de prédicat d'arité  $n$  ( $n \geq 1$ ) de  $L$ , on associe une application de  $D^n \rightarrow \{ \text{vrai, faux} \}$ .  
Si  $p$  est ce symbole de prédicat, l'application associée sera notée  $p'$ .

# Interprétation des langages du premier ordre (2/6)

**Définition #2 : affectation d'une variable relativement à une interprétation**

Soit  $I$  une interprétation pour un langage du premier ordre  $L$ .  
 $I$  est définie par a), b), c) et d) comme précédemment.

Une affectation des variables de  $L$  relativement à  $I$  est l'association à chacune des variables de  $L$  d'un élément de  $D$ , domaine d'interprétation de  $I$ .

# Interprétation des langages du premier ordre (3/6)

**Définition #3 : association d'une valeur à un terme de langage**

Soient : L un langage du premier ordre,  
I une interprétation de L,  
D le domaine d'interprétation de I,  
A une affectation des variables de L relativement à I.

La valeur du terme  $t$  relativement à I est définie par :

- si  $t$  est la constante  $a$ , alors sa valeur  $t'$  est l'élément  $a' \in D$ .
- si  $t$  est la variable  $x$  alors  $t'$  est l'élément de  $D$  qui est associé à  $x$  par l'affectation  $A$ .
- si  $t$  est de la forme  $f(t_1, t_2, \dots, t_n)$ , alors  $t'$  sera  $f'(t'_1, t'_2, \dots, t'_n)$ , où  $f$  symbole de fonction d'arité  $n$  et  $t_1, t_2, \dots, t_n$  des termes.  
 $t'_1, t'_2, \dots, t'_n$  sont des valeurs calculées selon les règles a), b), c) et d), avec  $t'_1, t'_2, \dots, t'_n \in D$  et  $f'(t'_1, t'_2, \dots, t'_n) \in D$ .

# Interprétation des langages du premier ordre (4/6)

## Définition #4 : interprétation des formules (1/3)

A une formule de L on peut associer une valeur de vérité comme suit :

a) si la formule est l'atome  $p(t_1, t_2, \dots, t_n)$  alors la valeur de vérité associée sera  $p'(t'_1, t'_2, \dots, t'_n) \in \{ \text{vrai, faux} \}$ , où :

- p est un symbole de prédicat d'arité n,
- $t_1, t_2, \dots, t_n$  sont des termes
- $t'_1, t'_2, \dots, t'_n$  sont les affectations de  $t_1, t_2, \dots, t_n$

# Interprétation des langages du premier ordre (5/6)

## Définition #4 : interprétation des formules (2/3)

b) si la formule est  $\neg f$ ,  $f \wedge g$ ,  $f \vee g$ ,  $f \Rightarrow g$  ou  $f \Leftrightarrow g$ , où  $f$  et  $g$  sont des formules, la valeur de vérité de la formule résultante est donnée par la table suivante :

f	g	$\neg f$	$f \wedge g$	$f \vee g$	$f \Rightarrow g$	$f \Leftrightarrow g$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

# Interprétation des langages du premier ordre (6/6)

## Définition #4 : interprétation des formules (3/3)

- c) si la formule est de la forme  $\exists x f$  alors sa valeur de vérité sera VRAI s'il existe  $d \in D$  tel que  $f$  a pour valeur de vérité VRAI quand on associe à  $x$  la valeur  $d$ .
- d) si la formule est de la forme  $\forall x f$  alors sa valeur de vérité sera VRAI si  $\forall d \in D$ ,  $f$  a la valeur de vérité VRAI lorsque l'on associe à  $x$  n'importe quelle valeur  $d$ .

# Exemple d'interprétation de formule

Soit la formule  $A = \forall x (p(x) \Rightarrow q)$  où  $p$  est défini sur le domaine  $D = \{ 1, 2 \}$ .

# Définitions et modèles (1/2)

## **Formule valide :**

Une formule est valide ssi elle est vraie pour toute interprétation  $I$ .  
On note alors  $\models A$ .

## **Formule satisfiable :**

Une formule  $A$  est dite satisfiable ssi elle est vraie pour au moins une interprétation  $I$ .  
On dit aussi que  $I$  est un **modèle** de  $A$ .

## **Formule insatisfiable :**

Une formule est insatisfiable ssi elle n'admet pas de modèle.

## **Conséquence logique :**

Soient une formule  $B$  et une famille de  $n$  formules  $A_1, \dots, A_n$ .  
 $B$  est conséquence logique des  $A_i$  si pour tout modèle  $M$  des  $A_i$ ,  
 $M$  est aussi un modèle de  $B$ . On note alors  $A_1, \dots, A_n \models B$ .

# Définitions et modèles (2/2)

Pour résumer, il y a 4 classes de formules :

- les **tautologies**, valides quelque soit l'interprétation
- les **satisfiables**, qui admettent au moins un modèle
- les **falsifiables**, où il existe au moins une interprétation qui n'est pas un modèle
- les **insatisfiables**, qui n'admettent pas de modèles.

# Logique des prédicats du premier ordre (PP)

- Introduction
- Définitions
- Théorie des modèles
- **Théorie de la démonstration**
- Propriétés fondamentales de (PP)
- Décidabilité de (PP)
- Principe de résolution

# Les axiomes dans (PP) (1/5)

Soient A, B et C des formules bien formées.

A1) Introduction de l'implication :

$$(A \Rightarrow (B \Rightarrow A))$$

A2) Distribution de l'implication :

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

A3) Contraposition :

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

A1), A2, A3) correspondent aux axiomes de (LP)

# Les axiomes dans (PP) (2/5)

Pour introduire A4), définissons ce qu'est le « *renommage de variable* ».

Soit  $A(X)$  une formule ayant  $X$  comme variable libre et  $t$  un terme. On note  $A(t)$  la formule obtenue en substituant le terme  $t$  à chaque occurrence de la variable  $X$  dans  $A$ .

La substitution peut être précédée d'une phase de renommage du nom des variables liées de  $A$  de manière à ce qu'aucune des variables de  $t$  ne se retrouve liée après la substitution.

Exemple : On veut substituer  $t = f(Y, u)$  à  $X$  dans :

$$A(X) = (p(X) \vee \forall Y \forall X r(X, Y))$$



Occurrences libres de  $X$

Occurrences liées de  $X$  et de  $Y$

# Les axiomes dans (PP) (3/5)

Exemple : On veut substituer  $t = f( Y, u )$  à la formule

$$A( X ) = ( p( X ) \vee \forall Y \forall X r( X, Y ) )$$

On remplace les occurrences liées :

- $X$  est renommée en  $X_1$
- $Y$  est renommée en  $Y_1$

On obtient la formule :

$$A( X ) = ( p( X ) \vee \forall Y_1 \forall X_1 r( X_1, Y_1 ) )$$

et on peut désormais sans risque effectuer la substitution de  $t = f( Y, u )$  à la variable  $X$  :

$$A( t ) = ( p( f( Y, u ) ) \vee \forall Y_1 \forall X_1 r( X_1, Y_1 ) )$$

# Les axiomes dans (PP) (4/5)

A4) Particularisation :

$$(\forall X A( X )) \Rightarrow A( t )$$

Exemple :

Avec A4), la formule  $\forall Y \exists Z \text{hait}( Y, Z )$  est dérivée en  $\exists Z \text{hait}( \text{mère}( X ), Z )$ . On a substitué à  $Y$  le terme  $\text{mère}( X )$ . Si on avait sans précaution substitué à  $Y$  le terme  $\text{mère}( Z )$ , on obtiendrait  $\exists Z \text{hait}( \text{mère}( Z ), Z )$ , ce qui exprime une situation beaucoup plus forte que celle que l'on était autorisé à déduire de l'antécédent.

Pour que la substitution soit réalisée correctement, il faut renommer l'occurrence liée de  $Z$ , d'où :

$$\exists Z_1 \text{hait}( \text{mère}( Z ), Z_1 )$$

# Les axiomes dans (PP) (5/5)

A5) Spécialisation universelle :

$$((\forall X) (A \Rightarrow B)) \Rightarrow (A \Rightarrow (\forall X) B)$$

si X n'est pas libre dans A

# Les règles d'inférence dans (PP)

R1a) Modus Ponens

$$\frac{\begin{array}{l} \mathbf{A} \\ \mathbf{A} \Rightarrow \mathbf{B} \end{array}}{\mathbf{B}}$$

R1b) Modus Tollens

$$\frac{\begin{array}{l} \neg \mathbf{A} \\ \mathbf{B} \Rightarrow \mathbf{A} \end{array}}{\neg \mathbf{B}}$$

R2) Généralisation

$$\frac{\mathbf{A}}{\forall X \mathbf{A}} \quad \text{où } X \text{ est une variable libre dans } A$$

# Exemple de démonstration dans (PP)

Soit le théorème suivant à démontrer :

$$(\forall y) (\forall x) D(x, y) \Rightarrow (\forall x) (\forall y) D(x, y)$$

**Démonstration :**

$$A4) (\forall y) (\forall x) D(x, y) \Rightarrow (\forall x) D(x, y)$$

$$A4) (\forall x) D(x, y) \Rightarrow D(x, y)$$

$$R2) D(x, y) \Rightarrow (\forall y) D(x, y)$$

$$R2) (\forall y) D(x, y) \Rightarrow (\forall x) (\forall y) D(x, y)$$

On pourra démontrer de façon semblable d'autres théorèmes :

$$1) (\exists y) (\forall x) D(x, y) \Rightarrow (\forall x) (\exists y) D(x, y)$$

$$2) (\forall x) D(x) \Rightarrow (\exists x) D(x)$$

$$3) \neg((\forall x) (\forall y) D(x, y)) \Leftrightarrow (\exists x) (\exists y) (\neg D(x, y))$$

$$4) \neg((\exists x) (\exists y) D(x, y)) \Leftrightarrow (\forall x) (\forall y) (\neg D(x, y))$$

# Dérivation dans (PP) (1/2)

Soit  $\{ A_1, A_2, \dots, A_p \}$  un ensemble de formules.

Une formule  $B$  est dérivée de  $\{ A_1, A_2, \dots, A_p \}$ , cette dérivation étant notée  $\{ A_1, A_2, \dots, A_p \} \vdash B$ , s'il existe une suite finie de formules  $B_1, B_2, \dots, B_n$  telles que  $B = B_n$  et  $\forall i \in \{ 1, \dots, n-1 \}$  :

- soit  $B_i$  est un axiome logique, c-à-d A1), A2), A3), A4) ou A5),
- soit  $B_i$  est un élément de  $\{ A_1, A_2, \dots, A_p \}$ , c-à-d une hypothèse,
- soit  $B_i$  est obtenue par application d'une des 2 règles d'inférence à une ou plusieurs formules de  $B_1, B_2, \dots, B_{i-1}$ .

# Dérivation dans (PP) (2/2)

Exemple :

Soit l'hypothèse  $\forall x \forall y p( x, y )$ , on veut en déduire  $\forall z p( z, z )$ .

1)  $\forall x \forall y p( x, y )$

2)  $\forall x \forall y p( x, y ) \Rightarrow \forall y p( z, y )$

3)  $\forall y p( z, y )$

4)  $\forall y p( z, y ) \Rightarrow p( z, z )$

5)  $p( z, z )$

6)  $\forall z p( z, z )$

A4) en substituant  $z$  à  $x$   
Modus Ponens 1) et 2)

A4) en substituant  $z$  à  $y$   
Modus Ponens 3) et 4)

R2) à 5)

# Exemple : l'arithmétique formelle (ou théorie des nombres) (1/4)

Ce système formel (AF), proposé par *Peano*, est une extension fondamentale de (PP).

Neuf axiomes sont ajoutés, faisant intervenir :

- une constante unique, le zéro : 0
- quatre opérateurs :
  - successeur  $\sigma$  (d'arité 1) avec :
    - $\sigma(n)$  noté  $(n+1)$ ,
    - $\sigma(0)$  noté 1.
  - somme  $+$  (d'arité 2)
  - multiplication  $*$  (d'arité 2)
  - égalité  $=$  (d'arité 2)

# Exemple : l'arithmétique formelle (ou théorie des nombres) (2/4)

Soient  $x, y, z$  toute suite de symboles de (AF) ne contenant pas le signe  $=$  :

$$(A6) (\forall x) x+0 = x$$

$$(A7) (\forall x) x*0 = 0$$

$$(A8) (\forall x) \neg (\sigma(x) = 0)$$

$$(A9) (\forall x) (\forall y) x+\sigma(y) = \sigma(x+y)$$

$$(A10) (\forall x) (\forall y) x*\sigma(y) = x*y + x$$

$$(A11) (\forall x) (\forall y) (\sigma(x) = \sigma(y)) \Rightarrow (x = y)$$

$$(A12) (\forall x) (\forall y) (x = y) \Rightarrow (\sigma(x) = \sigma(y))$$

$$(A13) (\forall x) (\forall y) (\forall z) (x = y) \Rightarrow ((x = z) \Rightarrow (y = z))$$

$$(A14) (A(0) \text{ et } (\forall u) (A(u) \Rightarrow A(\sigma(u)))) \Rightarrow (\forall u) A(u)$$

(A14) formalise le raisonnement par récurrence (induction formelle)

# Exemple : l'arithmétique formelle (ou théorie des nombres) (3/4)

Ce système formel de l'arithmétique formelle (AF) a une importance capitale.

En effet, l'arithmétisation peut être effectué pour tout système formel.

C'est en faisant appel à (AF) que les métathéorèmes de (DH) et de (JP) ont pu être énoncés.

Exemple pour (JP) :

(A) :  $a \square a$

R :  $m_1 \square m_2 \Rightarrow bm_1 \square bm_2$

$a$ ,  $b$  et  $\square$  ne sont que des symboles qui peuvent être remplacés par d'autres symboles, comme 1, 2 et 3 (symboles de AF)...

Par exemple, (A) peut être transformé en 1 3 1.

# Exemple : l'arithmétique formelle (ou théorie des nombres) (4/4)

(R) peut être transformé en :  $m_1 \ 3 \ m_2 \Rightarrow 2 \ m_1 \ 3 \ 2 \ m_2$ .

Mais l'opération « *être suivi de* », qui porte sur des suites de symboles, possède une correspondance dans (AF).

En effet, dans l'arithmétique usuelle, pour former le nombre  $x$  suivi du nombre  $y$  en utilisant la notation décimale, il suffit de multiplier  $x$  par une puissance de 10 et d'ajouter  $y$ .

Ainsi  $x \ y \equiv 10^n * x + y$ , avec  $n$  le nombre de chiffres de  $y$ .

L'étude du système (JP) par exemple peut ainsi complètement être ramenée à celle d'un sous-système de (AF).

# Logique des prédicats du premier ordre (PP)

- Introduction
- Définitions
- Théorie des modèles
- Théorie de la démonstration
- **Propriétés fondamentales de (PP)**
- Décidabilité de (PP)
- Principe de résolution

# Propriétés fondamentales de (PP) (1/2)

## Adéquation :

Une logique est dite adéquate si tout théorème  $A$  ( $\vdash A$ ) est une formule valide ( $\models A$ ).

**(PP) est adéquat : si  $\vdash A$  alors  $\models A$ .**

## Consistance :

Une logique est consistante s'il n'existe aucune formule  $A$  telle que  $\vdash A$  et  $\vdash \neg A$ .

**(PP) est consistant.**

# Propriétés fondamentales de (PP) (2/2)

## Complétude forte :

Une logique est dite fortement complète ssi la relation de conséquence valide correspond à la relation de dérivabilité : soit  $E$  un ensemble de formules, si  $E \models A$  alors  $E \vdash A$ .

## Complétude faible :

Une logique est faiblement complète si toute formule valide est un théorème : quand  $\models A$  alors  $\vdash A$ .

La complétude forte implique la complétude faible.

**(PP) est fortement complet.**

# Logique des prédicats du premier ordre (PP)

- Introduction
- Définitions
- Théorie des modèles
- Théorie de la démonstration
- Propriétés fondamentales de (PP)
- **Décidabilité de (PP)**
- Principe de résolution

# Décidabilité de (PP)

## Décidabilité :

Une logique est décidable s'il existe un algorithme capable de déterminer en un nombre fini d'étapes si une formule donnée est ou n'est pas un théorème.

**(PP) est indécidable, ou plus précisément semi-décidable.**

# Logique des prédicats du premier ordre (PP)

- Introduction
- Définitions
- Théorie des modèles
- Théorie de la démonstration
- Propriétés fondamentales de (PP)
- Décidabilité de (PP)
- Principe de résolution

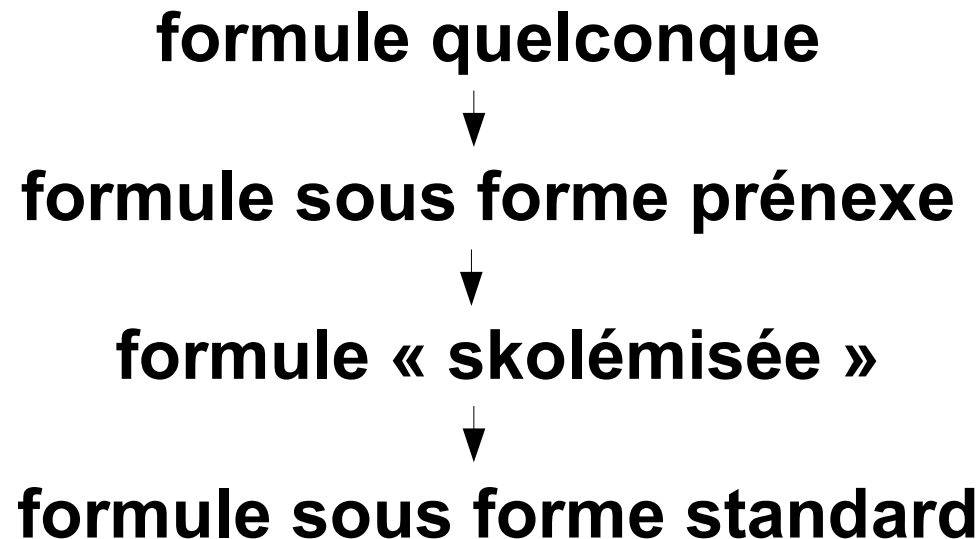
# Préambule au principe de résolution

## Remarques préalables :

Une démonstration (ou une déduction) dans (PP) peut être très fastidieuse à réaliser.

Pour réaliser plus simplement une démonstration, le principe de résolution utilise des formules mises sous **forme standard**.

Trois étapes sont nécessaires pour obtenir une telle formule :



# Forme prénexe d'une formule

## Matrice :

Une matrice est une formule du calcul des prédicats sans quantificateurs.

## Forme prénexe :

Une forme prénexe est une formule de la forme :

$Q_1 x_1 Q_2 x_2 \dots Q_n x_n M$  où  $Q_i$  désigne  $\forall$  ou  $\exists$  et  $M$  est une matrice.

## Forme prénexe équivalente :

Toute formule admet une forme prénexe équivalente.

# Mise sous forme prénexe d'une formule

1. remplacer les connecteurs  $\Rightarrow$  et  $\Leftrightarrow$  par des  $\wedge$  et  $\vee$
2. renommer certaines variables liées de manière à n'avoir plus de variables quantifiées deux fois
3. supprimer les quantificateurs inutiles, s'il y en a
4. transférer toutes les occurrences de la négation en utilisant les règles de réécriture suivantes :

$$4a. \quad \neg(A \wedge B) \quad \text{se réécrit en} \quad (\neg A \vee \neg B)$$

$$4b. \quad \neg(A \vee B) \quad \text{se réécrit en} \quad (\neg A \wedge \neg B)$$

$$4c. \quad \neg\forall A \quad \text{se réécrit en} \quad \exists\neg A$$

$$4d. \quad \neg\exists A \quad \text{se réécrit en} \quad \forall\neg A$$

5. transférer les quantificateurs en tête de formule, en utilisant la commutativité, l'associativité et le renommage de variables si nécessaire :

$$5a. \quad (\forall x A \wedge B) \quad \text{se réécrit en} \quad \forall x (A \wedge B)$$

$$5b. \quad (\exists x A \wedge B) \quad \text{se réécrit en} \quad \exists x (A \wedge B)$$

$$5c. \quad (\forall x A \vee B) \quad \text{se réécrit en} \quad \forall x (A \vee B)$$

$$5d. \quad (\exists x A \vee B) \quad \text{se réécrit en} \quad \exists x (A \vee B)$$

# Exemple de mise sous forme prénexe

Soit la formule :

$$\forall x p(x) \wedge \exists y q(y) \Rightarrow \exists y p(y)$$

Quelle est sa forme prénexe ?

# Skolémisation et forme standard

## Algorithme de skolémisation :

- transformer la formule  $F$  en forme prénexe
- remplacer toute variable quantifiée existentiellement par un symbole de fonction dont les arguments sont les variables qui sont quantifiées universellement et qui précèdent notre variable
- supprimer les quantificateurs (universels et existentiels), devenus inutiles.

Exemple : Skolémisation de l'exemple précédent.

## Forme standard :

Une formule sous forme standard est une matrice formée d'une conjonction de clauses.

# Principe et algorithme de résolution (1/3)

## Principe de résolution :

Soient  $N$  une forme standard et  $C_1$  et  $C_2$  deux clauses de  $N$ .

Soient 2 littéraux  $l_1 \in C_1$  et  $\neg l_2 \in C_2$ ,  $\theta$  une substitution de renommage telle que  $\theta(C_1)$  et  $C_2$  n'aient aucune variable libre commune.

Soient  $\sigma$  un unificateur de  $\theta(C_1)$  et  $C_2$  et  $R$  la résolvente :

$$R = \sigma( \theta( C_1 \setminus \{ l_1 \} ) \vee ( C_2 \setminus \{ \neg l_2 \} ) )$$

Alors  $N$  et  $N \cup \{ R \}$  sont logiquement équivalentes.

## Remarque :

Ce principe ne suffit pas à inférer la clause vide quand elle est conséquence logique d'un ensemble de clauses.

On adjoint à ce principe une règle de factorisation.

# Principe et algorithme de résolution (2/3)

## Règle de factorisation :

Soit  $C_1$  une clause de la forme  $l_1 \vee l_2 \vee \dots \vee l_n$  telle que  $l_1$  et  $l_2$  sont deux littéraux unifiables. Soit  $\sigma$  un unificateur de  $l_1$  et  $l_2$ .

La clause  $C'_1 = \sigma( C_1 \setminus \{ l_2 \} )$  est une conséquence logique de  $C_1$ .

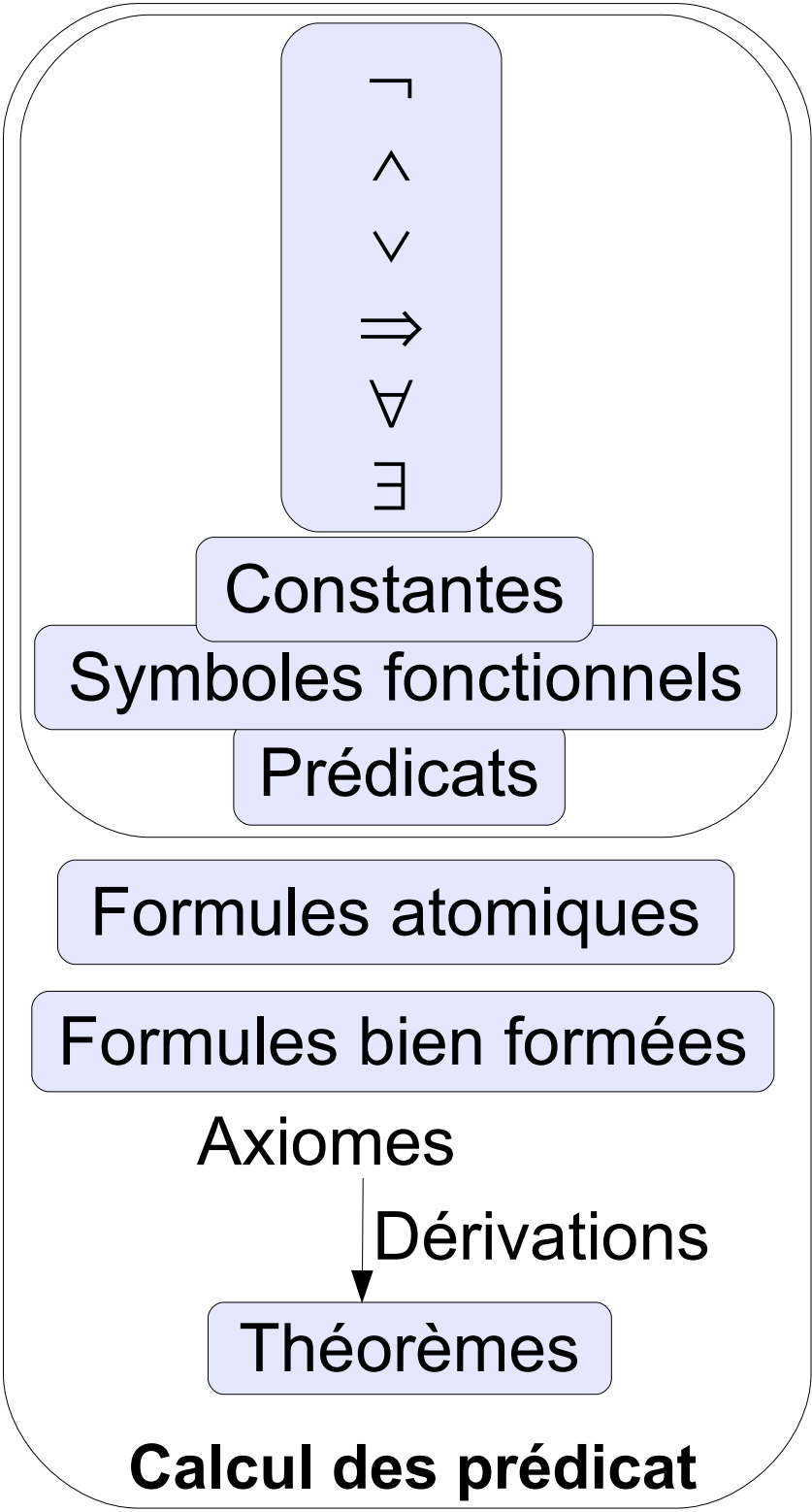
$C'_1$  est appelé facteur de  $C_1$ .

# Principe et algorithme de résolution (3/3)

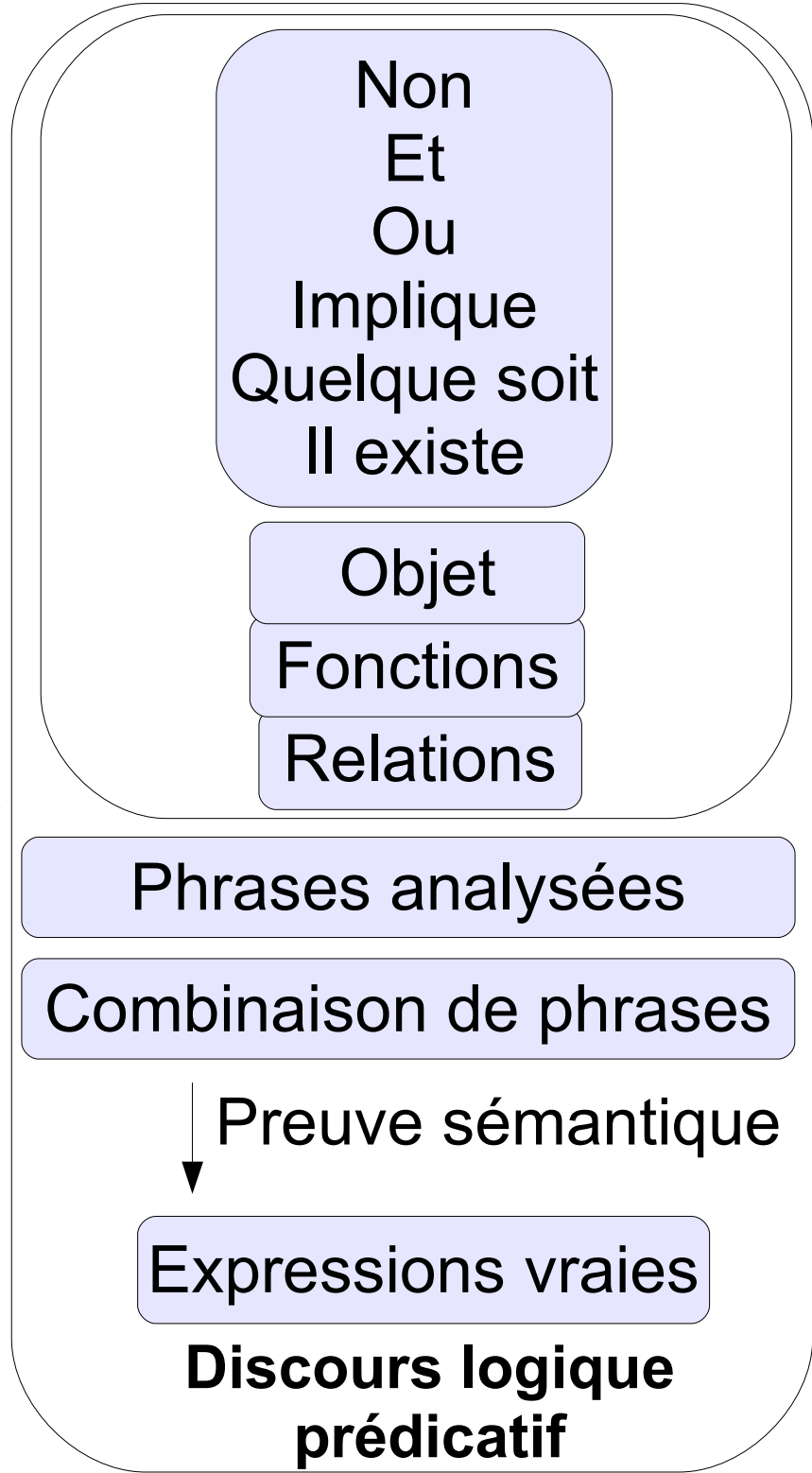
## Algorithme de résolution :

Soit  $N$  une formule sous forme standard.

1. Si  $\emptyset \in N$ , l'ensemble est inconsistant et la résolution terminée.
2. Prendre  $C_1$  et  $C_2$  deux clauses ou facteurs de clauses dans  $N$ ,  
telles que  $l_1 \in C_1$ ,  $\neg l_2 \in C_2$  et  $l_1, l_2$  unifiables.
3. Trouver un unificateur de  $l_1$  et  $l_2$  et calculer la résolvente  $R$ .
4. Recommencer en 1 en remplaçant  $N$  par  $N \cup \{ R \}$ .



INTERPRETATION



# Définition d'un système formel du 1er ordre

Tout système formel qui comporte les axiomes de (PP) est dit du premier ordre.

Le qualificatif **premier ordre** tient au fait que dans (PP), seules les **variables prédicatives** (symboles les plus internes des formules) peuvent être **quantifiées**.

A l'ordre 2, les prédicats eux-mêmes peuvent être quantifiés.

Ordres supérieurs : les quantificateurs agissent sur les prédicats de prédicats.