LOGIQUE COMPUTATIONNELLE

COURS 4

CALCUL DES PRÉDICATS II

Ch. Baskiotis

LAPI – EISTI

5 octobre 2009

Ce qu'on sait - I

– Constantes : $\Xi = \{a, b \ldots\}$

Ce qu'on sait - I

- Constantes : $\mathbf{\Xi} = \{a, b \ldots\}$
- Variables (libres et liées) : $\mathbf{V} = \{X, Y \ldots\}$

Ce qu'on sait - I

- Constantes : $\mathbf{\Xi} = \{a, b \ldots\}$
- Variables (libres et liées) : $\mathbf{V} = \{X, Y \ldots\}$
- Foncteurs : $f : \mathbf{V} \times \mathbf{\Xi} \rightarrow \mathbf{\Xi}$

Ce qu'on sait - I

- Constantes : $\mathbf{\Xi} = \{a, b \ldots\}$
- Variables (libres et liées) : $\mathbf{V} = \{X, Y \ldots\}$
- Foncteurs : $f : \mathbf{V} \times \mathbf{\Xi} \rightarrow \mathbf{\Xi}$
- Prédicats : $p = \mathbf{V} \times \mathbf{\Xi} \rightarrow \{vrai,\ faux\}$

Ce qu'on sait - I

- Constantes : $\mathbf{\Xi} = \{a, b \ldots\}$
- Variables (libres et liées) : $\mathbf{V} = \{X, Y \ldots\}$
- Foncteurs : $f : \mathbf{V} \times \mathbf{\Xi} \rightarrow \mathbf{\Xi}$
- Prédicats : $p = \mathbf{V} \times \mathbf{\Xi} \rightarrow \{vrai,\ faux\}$
- Quantificateurs : universel $\forall$, existentiel $\exists$

Ce qu'on sait - II

– Termes $\mathbb{T}$: constantes, variables, foncteurs avec comme arguments des termes.

Ce qu'on sait - II

- Termes $\mathbb{T}$: constantes, variables, foncteurs avec comme arguments des termes.
- Formules bien formées – fbf $\mathbb{F}$: Construites à partir des termes et en utilisant les connecteurs et les quantificateurs.

Ce qu'on sait - II

- Termes $\mathbb{T}$: constantes, variables, foncteurs avec comme arguments des termes.
- Formules bien formées – fbf $\mathbb{F}$: Construites à partir des termes et en utilisant les connecteurs et les quantificateurs.
- *Alphabet* $\Sigma_1 = \{\mathbf{V}, \mathbf{\Xi}, \mathbf{F}, \mathbf{P}, L\}$

Ce qu'on sait - II

- Termes $\mathbb{T}$: constantes, variables, foncteurs avec comme arguments des termes.
- Formules bien formées – fbf $\mathbb{F}$: Construites à partir des termes et en utilisant les connecteurs et les quantificateurs.
- *Alphabet* $\Sigma_1 = \{\mathbf{V}, \mathbf{\Xi}, \mathbf{F}, \mathbf{P}, L\}$
- *Langage formel du premier ordre* $\mathcal{L}_1 = \{\Sigma_1, \mathbb{T}, \mathbb{F}\}$

Ce qu'on sait - III

Interprétation sémantique

C'EST QUOI ? Donner un sens aux fbf

Ce qu'on sait - III

Interprétation sémantique

C'EST QUOI ? Donner un sens aux fbf

$\Rightarrow$ connaître leur valeur de vérité.

Ce qu'on sait - IV

Présentation de l'interprétation

1. À chaque constante de $\mathcal{L}$ correspond une constante a_I de $\mathcal{U}$

Ce qu'on sait - IV

Présentation de l'interprétation

1. À chaque constante de $\mathcal{L}$ correspond une constante a_I de $\mathcal{U}$
2. À chaque foncteur $f(t_1, \ldots t_n)$ de $\mathcal{L}$ correspond une fonction $f_I((t_1)_I, \ldots, (t_n)_I))$ de $\mathcal{U}$ avec $(t_k)_I$ l'interprétation du terme t_k par I.

Ce qu'on sait - IV

Présentation de l'interprétation

1. À chaque constante de $\mathcal{L}$ correspond une constante a_I de $\mathcal{U}$
2. À chaque foncteur $f(t_1, \ldots t_n)$ de $\mathcal{L}$ correspond une fonction $f_I((t_1)_I, \ldots, (t_n)_I))$ de $\mathcal{U}$ avec $(t_k)_I$ l'interprétation du terme t_k par I.
3. À chaque prédicat $p(t_1, \ldots t_n)$ de $\mathcal{L}$ correspond une application $p_I((t_1)_I, \ldots, (t_n)_I))$ de $\mathcal{U}^n$ dans $\{vrai, faux\}$.

4. À chaque variable X correspond, via l'assignation $\overline{\phi}$, un élément variable de l'univers du discours $\mathcal{U}$

Ce qu'on sait - V

Satisfiabilité. Modèles

– Une fbf F qui est une conséquence logique d'une interprétation I est dite *satisfiable*, I est un *modèle* de F et l'on note $I \models F$.

Ce qu'on sait - V

Satisfiabilité. Modèles

– Une fbf F qui est une conséquence logique d'une interprétation I est dite *satisfiable*, I est un *modèle* de F et l'on note $I \models F$.

– Une fbf F qui est satisfaite par toute interprétation est une *formule valide*. Notation $\models F$.

Ce qu'on sait - V

Satisfiabilité. Modèles

- Une fbf F qui est une conséquence logique d'une interprétation I est dite *satisfiable*, I est un *modèle* de F et l'on note $I \models F$.
- Une fbf F qui est satisfaite par toute interprétation est une *formule valide*. Notation $\models F$.
- Une fbf qui n'a pas de modèle est *insatisfiable*.

Ce qu'on sait - V

Satisfiabilité. Modèles

- Une fbf F qui est une conséquence logique d'une interprétation I est dite *satisfiable*, I est un *modèle* de F et l'on note $I \models F$.
- Une fbf F qui est satisfaite par toute interprétation est une *formule valide*. Notation $\models F$.
- Une fbf qui n'a pas de modèle est *insatisfiable*.
- Théorème (de l'insatisfiabilité).- Soient F un ensemble des fbf closes et G une fbf close.
  On a $F \models G$ ssi $F \bigcup \{\neg G\}$ est insatisfiable.

Évaluation syntaxique

Procédé mécanique

Évaluation syntaxique

Procédé mécanique

Fondé sur

– des axiomes ;

Évaluation syntaxique

Procédé mécanique

Fondé sur

- des axiomes ;
- des règles d'inférence

Évaluation syntaxique

Procédé mécanique

Fondé sur

- des axiomes ;
- des règles d'inférence

Établit la cohérence syntaxique d'une fbf.

$\Rightarrow$ Méthodologie de démonstration

Notion du théorème

Une fbf A est un *théorème* si

– A est un axiome ;

Notion du théorème

Une fbf A est un *théorème* si

– A est un axiome ;

– A est obtenu par application de la règle d'inférence sur d'autres théorèmes

Notion du théorème

Une fbf A est un *théorème* si

– A est un axiome ;

– A est obtenu par application de la règle d'inférence sur d'autres théorèmes

Notation : $\vdash A$

Axiomes

A1 Introduction de l'implication

$$A \to (B \to A)$$

Axiomes

A1 Introduction de l'implication

$$A \to (B \to A)$$

A2 Distributivité de l'implication

$$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$$

Axiomes

A1 Introduction de l'implication

$$A \to (B \to A)$$

A2 Distributivité de l'implication

$$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$$

A3 Négation

$$(\neg B \to \neg A) \to ((\neg B \to A) \to B)$$

Axiomes

A1 Introduction de l'implication

$$A \to (B \to A)$$

A2 Distributivité de l'implication

$$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$$

A3 Négation

$$(\neg B \to \neg A) \to ((\neg B \to A) \to B)$$

A4 Exemplification (instanciation) universelle :

$$\forall X\, A(X) \to A(c)$$

Axiomes

A1 Introduction de l'implication

$A \rightarrow (B \rightarrow A)$

A2 Distributivité de l'implication

$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

A3 Négation

$(\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B)$

A4 Exemplification (instanciation) universelle :

$\forall X\, A(X) \rightarrow A(c)$

A5 Généralisation universelle : $((A \rightarrow B) \rightarrow (A \rightarrow \forall X B))$

Règle d'inférence

Modus ponens

$$\frac{\vdash A, \vdash A \rightarrow B}{\vdash B}$$

Démonstration

La *démonstration* d'un théorème A dans une théorie T est une suite finie $(A_1, A_2, \ldots, A_n, A)$

Démonstration

La *démonstration* d'un théorème A dans une théorie T est une suite finie $(A_1, A_2, \ldots, A_n, A)$ où chaque A_i est

– soit un axiome ;

Démonstration

La *démonstration* d'un théorème A dans une théorie T est une suite finie
$(A_1, A_2, \ldots, A_n, A)$
où chaque A_i est

- soit un axiome ;
- soit le résultat d'une règle d'inférence appliquée sur les éléments A_j précédemment obtenus (c-à-d $j < i$).

Démonstration

La *démonstration* d'un théorème A dans une théorie T est une suite finie $(A_1, A_2, \ldots, A_n, A)$ où chaque A_i est

- soit un axiome ;
- soit le résultat d'une règle d'inférence appliquée sur les éléments A_j précédemment obtenus (c-à-d $j < i$).

$\Rightarrow$ la démonstration est un procédé mécanisable.

Consistance

Une logique est *syntaxiquement consistante* s'il n'existe aucune formule du langage telle que nous avons en même temps $\vdash A$ et $\neg \vdash A$.

Consistance

Une logique est *syntaxiquement consistante* s'il n'existe aucune formule du langage telle que nous avons en même temps $\vdash A$ et $\neg \vdash A$.

THÉORÈME : Le calcul des prédicats est syntaxiquement consistant.

Équivalence entre modèles et théorie de démonstration

La logique du premier ordre est

– *adéquate* $\vdash A \Rightarrow \models A$.

Équivalence entre modèles et théorie de démonstration

La logique du premier ordre est

– *adéquate* $\vdash A \Rightarrow \models A$.

– *complète* $\models A \Rightarrow \vdash A$.

Équivalence entre modèles et théorie de démonstration

La logique du premier ordre est

– *adéquate* $\vdash A \Rightarrow \models A$.

– *complète* $\models A \Rightarrow \vdash A$.

THÉORÈME (de completude) : Pour toute fbf p on a :

$A \vdash p \leftrightarrow A \models p$.

Équivalence entre modèles et théorie de démonstration

La logique du premier ordre est

– *adéquate* $\vdash A \Rightarrow \models A$.

– *complète* $\models A \Rightarrow \vdash A$.

THÉORÈME (de completude) : Pour toute fbf p on a : $A \vdash p \leftrightarrow A \models p$.

THÉORÈME (de refutation) : Soit E un ensemble des fbf et A une fbf. On a $E \vdash A$ ssi $E \cup \{\neg A\} \vdash \bot$.

Démonstration automatique

Démonstration automatique de l'implication $E \vdash A$

Démonstration automatique

Démonstration automatique de l'implication $E \vdash A$
$\Rightarrow E$ et A sous forme conjonctive normale (fcn).

Démonstration automatique

Démonstration automatique de l'implication $E \vdash A$
$\Rightarrow E$ et A sous forme conjonctive normale (fcn).

On connaît déjà le passage en fcn d'une fbf avec des connecteurs.

Démonstration automatique

Démonstration automatique de l'implication $E \vdash A$
$\Rightarrow E$ et A sous forme conjonctive normale (fcn).

On connaît déjà le passage en fcn d'une fbf avec des connecteurs.

Quid des quantificateurs ?

Quantificateur universelle et fcn

La fbf

$$\forall X p(X)$$

est équivalente à

$$p(X)$$

Quantificateur universelle et fcn

La fbf

$$\forall X p(X)$$

est équivalente à

$$p(X)$$

$\Rightarrow$ Suppression des quantificateurs universels

Quantificateur universelle et fcn

La fbf

$$\forall X p(X)$$

est équivalente à

$$p(X)$$

$\Rightarrow$ Suppression des quantificateurs universels sans modification de la sémantique de la fbf.

Quantificateur universelle et fcn

La fbf

$$\forall X p(X)$$

est équivalente à

$$p(X)$$

$\Rightarrow$ Suppression des quantificateurs universels sans modification de la sémantique de la fbf.

$\Rightarrow$ Pas de quantificateur universel dans les fcn.

Quantificateur existentiel et fonctions de Skolem

Soit la fbf

$$\forall X \; \exists Y \; p(X, Y)$$

Quantificateur existentiel et fonctions de Skolem

Soit la fbf

$$\forall X \; \exists Y \; p(X, Y)$$

$\Rightarrow$ On dispose d'un mécanisme qui pour tout $X = X_0$ particulier, il peut produire une constante a tq $p(X_0, a)$.

Quantificateur existentiel et fonctions de Skolem

Soit la fbf

$$\forall X \; \exists Y \; p(X, Y)$$

$\Rightarrow$ On dispose d'un mécanisme qui pour tout $X = X_0$ particulier, il peut produire une constante a tq $p(X_0, a)$.

a depend de X_0.

Quantificateur existentiel et fonctions de Skolem

Soit la fbf

$$\forall X \; \exists Y \; p(X, Y)$$

$\Rightarrow$ On dispose d'un mécanisme qui pour tout $X = X_0$ particulier, il peut produire une constante a tq $p(X_0, a)$.

a depend de X_0.

a est une fonction de X : $a = s_Y(X)$ qui s'appelle *fonction de Skolem*.

Quantificateur existentiel et fonctions de Skolem

Soit la fbf

$$\forall X \; \exists Y \; p(X, Y)$$

$\Rightarrow$ On dispose d'un mécanisme qui pour tout $X = X_0$ particulier, il peut produire une constante a tq $p(X_0, a)$.

a depend de X_0.

a est une fonction de X : $a = s_Y(X)$ qui s'appelle *fonction de Skolem*.

fcn correspondante : $p(X, s_Y(X))$.

Fonction de Skolem - Exemple

Soit le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Fonction de Skolem - Exemple

Soit le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Considérons la fbf $\forall X \; \exists Y \; pp(X, Y)$

Fonction de Skolem - Exemple

Soit le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Considérons la fbf $\forall X \ \exists Y \ pp(X, Y)$

Cette fbf est vraie si nous avons un mécanisme tel que pour tout X il pourrait nous construire un Y avec $X < Y$.

Fonction de Skolem - Exemple

Soit le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Considérons la fbf $\forall X \ \exists Y \ pp(X, Y)$

Cette fbf est vraie si nous avons un mécanisme tel que pour tout X il pourrait nous construire un Y avec $X < Y$.

En fait au lieu d'élaborer ce mécanisme, on en stipule son existence !

Fonction de Skolem - Exemple

Soit le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Considérons la fbf $\forall X \ \exists Y \ pp(X, Y)$

Cette fbf est vraie si nous avons un mécanisme tel que pour tout X il pourrait nous construire un Y avec $X < Y$.

En fait au lieu d'élaborer ce mécanisme, on en stipule son existence !

Ce mécanisme prend le nom de *fonction de Skolem* et sera noté $s_Y(X)$.

Fonction de Skolem - Le piège

Soit la fbf

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

Fonction de Skolem - Le piège

Soit la fbf

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

fcn correspondante :

$$p(X, s_Y)$$

Fonction de Skolem - Le piège

Soit la fbf

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

fcn correspondante :

$$p(X, s_Y)$$

La fonction de Skolem s_Y est sans arguments

Fonction de Skolem - Le piège

Soit la fbf

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

fcn correspondante :

$$p(X, s_Y)$$

La fonction de Skolem s_Y est sans arguments

$\Rightarrow$ Elle est une constante.

Fonction de Skolem - Exemple

Soit toujours le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Fonction de Skolem - Exemple

Soit toujours le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Considérons la fbf $\exists Y \ \forall X \ pp(X, Y)$

Fonction de Skolem - Exemple

Soit toujours le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Considérons la fbf $\exists Y \ \forall X \ pp(X, Y)$

Cette fbf est vraie si nous avons un mécanisme tel que il pourrait nous construire un Y avec $X < Y$ pour tout X.

Fonction de Skolem - Exemple

Soit toujours le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Considérons la fbf $\exists Y \ \forall X \ pp(X, Y)$

Cette fbf est vraie si nous avons un mécanisme tel que il pourrait nous construire un Y avec $X < Y$ pour tout X.

Ici Y doit vérifiée le prédicat $\forall X \ \Rightarrow$ elle ne dépend pas du choix particulier de X.

Fonction de Skolem - Exemple

Soit toujours le prédicat $pp(X, Y)$ qui est vrai si X est plus petit que Y.

Considérons la fbf $\exists Y \ \forall X \ pp(X, Y)$

Cette fbf est vraie si nous avons un mécanisme tel que il pourrait nous construire un Y avec $X < Y$ pour tout X.

Ici Y doit vérifiée le prédicat $\forall X \ \Rightarrow$ elle ne dépend pas du choix particulier de X.

Donc la fonction de Skolem $s_Y(X)$ n'a pas d'argument $X \ \Rightarrow$ elle est une constante s_Y.

Fonction de Skolem - L'explication

$\forall X \; \exists Y \; p(X, Y) \;\; \Rightarrow \;\; p(X, s_Y(X))$

Y se trouve sous la portée de $\forall X$

$\Rightarrow Y$ est fonction de X

Fonction de Skolem - L'explication

$\forall X \; \exists Y \; p(X, Y) \;\; \Rightarrow \;\; p(X, s_Y(X))$

Y se trouve sous la portée de $\forall X$

$\Rightarrow Y$ est fonction de X

$\exists Y \; \forall X \; p(X, Y) \;\; \Rightarrow \;\; p(X, s_Y)$

Y n'est pas sous la portée

d'un quantificateur universel

$\Rightarrow Y$ est une constante

PROLOG

PROLOG = *PRO*grammation *LOG*ique

PROLOG

PROLOG = *PRO*grammation *LOG*ique

Langage de programmation qui utilise les clauses de Horn et la logique du premier ordre.

Déclaratif versus procédural

Prog. Procédurale	Prog. Déclarative
Le programme décrit la façon de *procéder* pour obtenir le résultat souhaité, en imitant le fonctionnement de l'ordinateur	Le programme *déclare* le résultat souhaité que l'ordinateur doit obtenir, sans se soucier de la technique utilisée.

Déclaratif versus procédural

Prog. Procédurale	Prog. Déclarative
Le programme décrit la façon de *procéder* pour obtenir le résultat souhaité, en imitant le fonctionnement de l'ordinateur	Le programme *déclare* le résultat souhaité que l'ordinateur doit obtenir, sans se soucier de la technique utilisée.
Compilation	Logique 1er ordre

Déclaratif versus procédural

Prog. Procédurale	Prog. Déclarative
Le programme décrit la façon de *procéder* pour obtenir le résultat souhaité, en imitant le fonctionnement de l'ordinateur	Le programme *déclare* le résultat souhaité que l'ordinateur doit obtenir, sans se soucier de la technique utilisée.
Compilation	Logique 1er ordre
Résultat	Résultat

Connaître la famille – Les ancêtres

AbSys Langage d'assertions (1969)

Premier langage de programmation logique

Connaître la famille – Les ancêtres

AbSys Langage d'assertions (1969)

Premier langage de programmation logique

Golux Langage équationnel (1973)

Tentative de concilier la programmation déclarative avec la programmation procédurale.

Connaître la famille – Les parents

1973 Présentation de Prolog par A. Colmerauer et P. Roussel (Un. de Marseille)

Connaître la famille – Les parents

1973 Présentation de Prolog par A. Colmerauer et P. Roussel (Un. de Marseille)

1976 Prolog sur micro-ordinateur.

Connaître la famille – Les parents

1973 Présentation de Prolog par A. Colmerauer et P. Roussel (Un. de Marseille)

1976 Prolog sur micro-ordinateur.

1977 Prolog d'Édimbourg par D. Warren. Premier Prolog puissant.

Connaître la famille – Les descendants

Gödel Langage de métaprogrammation (c-à-d. programmation qui traite la représentation d'autres programmes) logique (Lloyd 1994)

Connaître la famille – Les descendants

Gödel Langage de métaprogrammation (c-à-d. programmation qui traite la représentation d'autres programmes) logique (Lloyd 1994)

Oz Langage multiparadigme, c'est-à-dire logique, fonctionnel, avec contraintes, objet, concurrent. (G. Smolka 1991)

Connaître la famille – Les descendants

Gödel Langage de métaprogrammation (c-à-d. programmation qui traite la représentation d'autres programmes) logique (Lloyd 1994)

Oz Langage multiparadigme, c'est-à-dire logique, fonctionnel, avec contraintes, objet, concurrent. (G. Smolka 1991)

λProlog Langage de logique d'ordre supérieur

Que peut-on faire avec ?

Raconter l'histoire de sa vie ou celle de l'humanité !

Que peut-on faire avec ?

Raconter l'histoire de sa vie ou celle de l'humanité !

Règle : *X est mortel si X est un homme*

Traduction en Prolog `mortel(X) :- homme(X).`

Règle : *Il existe un homme si X est un homme*

Traduction en Prolog `existeHomme :- homme(X).`

Un fait : On affirme que Socrate est un homme

Traduction en Prolog `homme(socrate).`

Un autre fait :

`homme(aristote).`

Question avec réponse `yes`

? mortel(socrate). -

Une autre question

avec réponses `X=socrate; Y=aristote`

? mortel(X). -

Avant le prochain TD

- Installer la dernière version de SWI-Prolog.

Site : `http ://www.swi-prolog.org/`

Avant le prochain TD

- Installer la dernière version de SWI-Prolog.

  Site : `http ://www.swi-prolog.org/`
- Installer l'éditeur associé

• Site pour Windows :

`http ://lernen.bildung.hessen.de/informatik/swiprolog/indexe.htm`

Avant le prochain TD

- Installer la dernière version de SWI-Prolog.

  Site : `http ://www.swi-prolog.org/`
- Installer l'éditeur associé

• Site pour Windows :

`http ://lernen.bildung.hessen.de/informatik/swiprolog/indexe.htm`

• Site pour Linux :

`http ://turing.ubishops.ca/home/bruda/emacs-prolog/`