Polytech'Lille - GIS4

Logique & Web Sémantique

S. Janot

Première partie: Représentation des Connaissances et Raisonnement

* Logique propositionnelle

* Logique du premier ordre

* Bases de connaissance, systèmes à base de règles

Deuxième partie: Web Sémantique

* ***Représentation de données: graphes RDF***

* ***SPARQL***

* ***Ontologies***

Logique propositionnelle

Logique

Syntaxe

ensemble de symboles et de règles, pour écrire des formules logiques

Sémantique

attribue une valeur de vérité à chaque formule

Syntaxe

* atomes

  chaînes de caractères, commençant par une lettre minuscule

  valeurs spéciales: V, F

* connecteurs logiques

  $\neg$ (non)

  $\wedge$ (et)

  $\vee$ (ou)

  $\rightarrow$ (implique)

* formules

- un atome est une formule
- si e est une formule, alors $\neg$e est une formule
- si e_1 et e_2 sont des formule, alors:

e1 $\wedge$ e2 , e1 $\vee$ e2, e1 $\rightarrow$ e2 sont des formules

* exemples

p $\wedge$ q

((p $\wedge$ q) $\wedge$ r)

(p $\rightarrow$ q) $\rightarrow$ (p $\vee$ q)

(p $\vee$) $\wedge$ q

p $\rightarrow$ $\neg$ $\neg$($\wedge$ q)

Sémantique

définit le sens du langage

Interprétation

- chaque atome est associé à une **proposition** (la dénotation)
- chaque **proposition** a une valeur: Vrai ou Faux

Exemple:
p: "il fait beau", vrai

p: atome
"Il fait beau" : proposition, dénotation de p
vrai: valeur de vérité de "Il fait beau"

Exemple:

Chaque midi, trois étudiants, Xavier ,Yann et Zoé mangent ensemble. Les trois affirmations suivantes sont vraies :
- si Xavier commande un dessert, alors Yann aussi
- chaque jour, soit Yann, soit Zoé commande un dessert, mais pas les deux
- Xavier ou Zoé ou les deux commandent chaque jour un dessert
- si Zoé commande un dessert, alors Xavier aussi

Qui prend un dessert ?

Exemple:

Chaque midi, trois étudiants, Xavier ,Yann et Zoé mangent ensemble. Les trois affirmations suivantes sont vraies :
- si Xavier commande un dessert, alors Yann aussi
- chaque jour, soit Yann, soit Zoé commande un dessert, mais pas les deux
- Xavier ou Zoé ou les deux commandent chaque jour un dessert
- si Zoé commande un dessert, alors Xavier aussi

Qui prend un dessert ?

x: Xavier commande un dessert

y: Yann commande un dessert

z: Zoé commande un dessert

Exemple:

Chaque midi, trois étudiants, Xavier ,Yann et Zoé mangent ensemble. Les trois affirmations suivantes sont vraies :
- si Xavier commande un dessert, alors Yann aussi
  f1: $\mathbf{x \rightarrow y}$
- chaque jour, soit Yann, soit Zoé commande un dessert, mais pas les deux
  f2: $\mathbf{(y \vee z) \wedge (\neg y \vee \neg z)}$
- Xavier ou Zoé ou les deux commandent chaque jour un dessert
  f3: $\mathbf{x \vee z}$
- si Zoé commande un dessert, alors Xavier aussi
  f4: $\mathbf{z \rightarrow x}$

Qui prend un dessert ?

x: Xavier commande un dessert

y: Yann commande un dessert

z: Zoé commande un dessert

Tables de vérité

* définition de la sémantique des connecteurs logiques:

ω_1	ω_2	$\omega_1 \wedge \omega_2$	$\omega_1 \vee \omega_2$	$\neg \omega_1$	$\omega_1 \rightarrow \omega_2$
V	V	V	V	F	V
V	F	F	V	F	F
F	V	F	V	V	V
F	F	F	F	V	V

Valeur de vérité d'une formule :

étant donnée une **interprétation**, la valeur de n'importe quelle formule peut être calculée en utilisant une table de vérité:

table de vérité pour **$p \wedge \neg q$**:

p	q	$\neg q$	$p \wedge \neg q$
V	V	F	F
V	F	V	V
F	V	F	F
F	F	V	F

table de vérité pour **$(p \rightarrow \neg q) \wedge p$**:

p q $\neg q$ $p \rightarrow \neg q$ $(p \rightarrow \neg q) \wedge p$

p	q	$\neg q$	$p \rightarrow \neg q$	$(p \rightarrow \neg q) \wedge p$

Chaque midi, trois étudiants, Xavier ,Yann et Zoé mangent ensemble. Les trois affirmations suivantes sont vraies :

- si Xavier commande un dessert, alors Yann aussi

  f1: $\mathbf{x \rightarrow y}$

- chaque jour, soit Yann, soit Zoé commande un dessert, mais pas les deux

  f2: $\mathbf{(y \vee z) \wedge (\neg y \vee \neg z)}$

- Xavier ou Zoé ou les deux commandent chaque jour un dessert

  f3: $\mathbf{x \vee z}$

- si Zoé commande un dessert, alors Xavier aussi

  f4: $\mathbf{z \rightarrow x}$

Qui prend un dessert ?

* table de vérité avec f1, f2, f3 et f4
* valeurs de x, y et z pour lesquelles on a f1, f2, f3 et f4 vraies

X	Y	Z	**x →y**	**y ∨ z**	**¬y ∨¬z**	f2	**x ∨ z**	**z → x**

Satisfiabilité, modèles d'une formule ou d'un ensemble de formules

Une interprétation ***satisfait*** une formule ω si cette formule a la valeur V sous cette interprétation.

Une interprétation ***satisfait*** un ensemble de formules Δ si chaque formule de Δ a la valeur V sous cette interprétation.

Une interprétation qui satisfait une formule est appelée un ***modèle*** de cette formule.

Une formule (ou un ensemble de formules) qui a au moins un modèle est ***satisfiable***.

Une formule est ***valide*** si elle est vraie dans toute interprétation.

Une formule (ou un ensemble de formule) est ***insatisfiable (ou invalide)*** si elle est fausse dans toute interprétation

Exemples:

$p \vee \neg p$	valide
$p \wedge \neg p$	invalide
$p \rightarrow (q \vee \neg p)$	satisfiable
$p \rightarrow (q \rightarrow p)$	?
$((p \rightarrow q) \rightarrow p) \rightarrow p$	?
$(\neg p \wedge q) \rightarrow (q \rightarrow p)$?	
$\neg p \wedge \neg (p \rightarrow q)$	?

Exemples:

$p \vee \neg p$	valide	***toujours vraie***
$p \wedge \neg p$	invalide	***jamais vraie***
$p \rightarrow (q \vee \neg p)$	satisfiable	***parfois vraie***

$p \rightarrow (q \rightarrow p)$ *vraie si p Vrai, vraie si p Faux*: **valide**

$((p \rightarrow q) \rightarrow p) \rightarrow p$

$(\neg p \wedge q) \rightarrow (q \rightarrow p)$

$\neg p \wedge \neg (p \rightarrow q)$

Lois d'équivalence

Deux formules sont équivalentes si et seulement si elles ont les mêmes modèles.

Lois de De Morgan

$$\neg\ (\omega_1\ \vee\ \omega_2)\ \equiv\ \neg\omega_1\ \wedge\ \neg\omega_2$$
$$\neg\ (\omega_1\ \wedge\ \omega_2)\ \equiv\ \neg\omega_1\ \vee\ \neg\omega_2$$

Contraposition

$$(\omega_1\ \rightarrow\ \omega_2)\ \equiv\ (\neg\omega_2\ \rightarrow\ \neg\omega_1)$$

Associativité

$$(\omega_1\ \vee\ \omega_2)\ \vee\ \omega_3\ \equiv\ \omega_1\ \vee\ (\omega_2\ \vee\ \omega_3\ )$$
$$(\omega_1\ \wedge\ \omega_2)\ \wedge\ \omega_3\ \equiv\ \omega_1\ \wedge\ (\omega_2 \wedge \omega_3\ )$$

Distributivité

$$\omega_1\ \vee\ (\omega_2 \wedge \omega_3\ )\ \equiv\ (\omega_1\ \vee\ \omega_2)\ \wedge\ (\omega_1\ \vee\ \omega_3)$$
$$\omega_1\ \wedge\ (\omega_2\ \vee\ \omega_3\ )\ \equiv\ (\omega_1\ \wedge\ \omega_2)\ \vee\ (\omega_1\ \wedge\ \omega_3)$$

Démonstration: tables de vérité

Forme Normale Conjonctive **(FNC)** :

Toute formule peut-être transformée en une formule équivalente sous la forme d'une conjonction de clauses: $c_1 \wedge c_2 \wedge ... \wedge c_n$

où chaque clause ci est de la forme

$$a_1 \vee a_2 \vee ... \vee a_p$$

et chaque termes a_k est un atome ou la négation d'un atome

(de façon analogue, toute formule peut-être transformée sous forme normale disjonctive)

Exemples:

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

$(a \vee b \vee c) \wedge (a \vee \neg b) \wedge (a \vee \neg c)$

Mise sous forme normale conjonctive

1- enlever les implications en appliquant l'équivalence:

$$(\omega_1 \rightarrow \omega_2) \equiv \neg\omega_1 \vee \omega_2$$

2- appliquer les lois de De Morgan pour réduire la portée des négations ($\neg$)

$$\neg (\omega_1 \vee \omega_2) \equiv \neg\omega_1 \wedge \neg\omega_2$$
$$\neg (\omega_1 \wedge \omega_2) \equiv \neg\omega_1 \vee \neg\omega_2$$

3- appliquer la loi de distributivité pour transformer les disjonctions en conjonctions:

$$\omega_1 \vee (\omega_2 \wedge \omega_3) \equiv (\omega_1 \vee \omega_2) \wedge (\omega_1 \vee \omega_3)$$

Exemples :

$p \rightarrow (q \vee \neg p)$

$\neg p \vee (q \vee \neg p)$

$\neg \mathbf{p \vee q}$

$(\neg p \wedge q) \rightarrow (q \rightarrow p)$

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

$(p \vee \neg q) \vee (\neg q \vee p)$

$\mathbf{p \vee \neg q}$

Exercice :

Mettre sous forme normale conjonctive la formule $\neg(p \rightarrow q) \vee (r \rightarrow p)$

Conséquence logique

Si une formule **ω est vraie** pour toute **interprétation qui satisfait un ensemble de formules Δ**, alors **ω est conséquence logique de Δ**. On note $\Delta \Rightarrow \omega$.

Exemples:

$\{p\} \Rightarrow p$

$\{p, q\} \Rightarrow p \wedge q$

$F \Rightarrow \omega , \forall \omega$

Théorème de déduction

Si $\{\omega_1, \omega_2, .. , \omega_v\} \Rightarrow \omega$, alors $(\omega_1 \wedge \omega_2 \wedge \ldots \wedge \omega_v) \rightarrow \omega$ est valide et réciproquement

Raisonnement par l'absurde

Soit Δ est un ensemble de formules, et ω une formule. Si **Δ est satisfiable** et $\Delta \cup \{\neg\omega\}$ n'a pas de modèle alors **ω est conséquence logique de Δ**.

Démonstration automatique

Règles d'inférence

Une règle d'inférence est une règle syntaxique permettant la production de nouvelles formules

on note:

$\alpha \vdash \beta$: β peut être est inféré à partir de α

$\alpha_1 , \alpha_2 \vdash \beta$: β est inféré à partir de α_1 et α_2

<u>Exemple :</u>

$\omega_1 , \omega_2 \vdash \omega_1 \wedge \omega_2$ (introduction du et)

* Règles d'inférences usuelles en logique propositionnelle

$\omega_1 \ , \ \omega_1 \rightarrow \omega_2 \ \vdash \omega_2$ (modus ponens)

$\omega_1 \ , \ \omega_2 \ \vdash \ \omega_1 \ \wedge \ \omega_2$ (introduction du et)

$\omega_1 \ \wedge \ \omega 2 \ \vdash \omega_1$ (élimination du et)

$\omega_1 \ \vdash \omega_1 \ \vee \ \omega_2$ (introduction du ou)

Preuve, déduction

Une séquence de formules $\{\omega_1, \omega_2, .. , \omega_n\}$ est **une preuve de ω_n** à partir d'un ensemble de formules Δ si , $\forall$ i $\in$ [1,n] :

- **ω_i appartient à Δ**
- **ou ω_i peut être inférée à partir des formules $\{\omega_1, \omega_2, .. , \omega_{i-1}\}$**

On note $\Delta \vdash \omega_n$

ω_n est un **théorème** de Δ

Exemple:

* $\Delta = \{ p, r, p \rightarrow q \}$

preuve de $q \wedge r$ à partir de Δ?

p	$\in \Delta$
$p \rightarrow q$	$\in \Delta$
q	*modus ponens*
r	$\in \Delta$
$q \wedge r$	*introduction du* **et**

$\{p, p \rightarrow q, q, r, q \wedge r\}$ est une preuve de $q \wedge r$

Sémantique et Inférence

sémantique: modèles, valeurs de vérité d'une formule

règles d'inférences, preuve:

* méthodes de calcul pour produire des nouvelles formules

Sémantique et Inférence

sémantique: modèles, valeurs de vérité d'une formule

règles d'inférences, preuve:

* méthodes de calcul pour produire des nouvelles formules

** Une formule déduite à partir d'un ensemble Δ en appliquant les règles d'inférence est-elle toujours conséquence logique de Δ?* **correction**

** Toute formule conséquence logique de Δ peut-elle être déduite à partir de Δ en appliquant les règles d'inférence?* **complétude**

*Un ensemble de règles d'inférences R est **correct** si :*

Pour tout ensemble de formules Δ, si ω peut-être inférée à partir de Δ en utilisant un ensemble de règles R, alors ω est conséquence logique de Δ.

*Un ensemble de règles d'inférences est **complet** si :*

Pour tout ensemble de formules Δ et pour toute formule ω, si ω est conséquence logique de Δ, alors il existe une preuve de ω à partir de Δ.

L'ensemble des règles d'inférences mentionné est correct

La ***résolution par réfutation*** (chapitre suivant) est complète pour la logique propositionnelle restreinte à un certain type de formules logiques

Résolution en logique propositionnelle

méthode de déduction automatique (règle d'inférence)
qui s'applique à des ensembles de **clauses.**

Une clause est une disjonction de littéraux, un littéral est un atome ou la négation d'un atome.

Exemples:

p ∨ q ∨ ¬r

p ∨ ¬q ∨ ¬r

Règle de résolution:

A partir d'une clause **p** ∨ c1 et d'une clause ¬**p** ∨ c2, on déduit c1 ∨ c2

(c1 ∨ c2) est le résolvant

cas particulier

la clause vide est notée ***nil***

correspond à un ensemble inconsistant (insatisfiable): p et ¬p

La résolution est *correcte :*

Si **p** ∨ **c1** est vraie (relativement à une interprétation I) et ¬**p** ∨**c2** est vraie (relativement à I) alors **c1** ∨ **c2** est vraie (relativement à I).

Exemple :

a partir de :

p ∨ ¬q ∨ ¬r et ¬**p** ∨ s

on déduit:

¬q ∨ ¬r ∨ s

Chaque midi, trois étudiants, Xavier ,Yann et Zoé mangent ensemble. Les trois affirmations suivantes sont vraies :

- si Xavier commande un dessert, alors Yann aussi

  f1: $\mathbf{x \rightarrow y}$

- chaque jour, soit Yann, soit Zoé commande un dessert, mais pas les deux

  f2: $\mathbf{(y \vee z) \wedge (\neg y \vee \neg z)}$

- Xavier ou Zoé ou les deux commandent chaque jour un dessert

  f3: $\mathbf{x \vee z}$

- si Zoé commande un dessert, alors Xavier aussi

  f4: $\mathbf{z \rightarrow x}$

Qui prend un dessert ?

* déduction par résolution pour trouver les valeurs x, y et z à partir de f1, f2, f3 et f4

Résolution par réfutation

Clauses de Horn

forme particulière de clauses, comportant au plus un littéral **positif**.
méthodes de déduction efficaces pour ce type de clauses.

Trois types de clauses de Horn:

"fait" : clause composée uniquement d'un atome

"règle" : un atome positif et au moins un littéral négatif

$p \vee \neg q$

$p \vee \neg q \vee \neg r$

correspond à des formules du type:

$a \wedge b \wedge c \rightarrow d$

"but": uniquement des littéraux négatifs

$\neg p \vee \neg q$

Exemple :

a.

b.

$a \wedge b \rightarrow c$

$a \wedge c \rightarrow d$

$a \wedge b \wedge d \rightarrow e$

Restrictions :

pas de négation

pas de disjonction dans la partie conclusion

Preuve par réfutation

Pour prouver qu'une formule donnée **ω** est conséquence logique d'un ensemble de clauses **satisfiables Δ,** il suffit de montrer que **Δ** ∪ {¬**ω**} est insatisfiable :

Soit **Δ** est un ensemble de formules satisfiable et **ω** une formule.

ω est conséquence logique de **Δ**. si et seulement si il existe une résolution qui donne la clause vide à partir de **Δ** ∪ {¬**ω**}

Exemple

Réfutation pour montrer que **c** est conséquence logique de **(a** ∧**b)** et ¬**b** ∨ **c**

Réfutation à partir de {**a**, **b** ,¬**b** ∨ **c** } ∪ **{¬c}**

Procédure de résolution par réfutation

Pour prouver **ω** à partir d'un ensemble (satisfiable) de formules **Δ** :

- convertir **Δ** en clauses
- convertir la négation de ω en clause et l'ajouter à l'ensemble

- appliquer la résolution à l'ensemble jusqu'à l'obtention de la clause vide ou jusqu'à ce qu'il n'y ait plus de résolvants possibles

Stratégies de résolution

Comment choisir les résolutions à effectuer pour obtenir une méthode efficace?

- éviter les résolutions inutiles
- trouver la dérivation la plus courte pour arriver à la clause vide.

* calcul exhaustif :

  ex: *saturation* par niveau

* construction d'un arbre de dérivation à partir d'une clause

Exemple

$\Delta = \{ a \vee b,\ a \vee \neg b,\ \neg a \vee b \}$

$\omega = a \wedge b$

Résolution linéaire

Une déduction **linéaire** de racine c_0 à partir d'un ensemble Δ contenant c_0 est une déduction de la forme $c_0c_1c_2.....c_n$ telle que:

$\mathbf{c}_i$ est obtenue à partir de $\mathbf{c}_{i-1}$ et d'une clause de $\Delta \cup \{c_0, c_1, c_2, ... c_i\}$

restriction: utilisation de la dernière clause produite

La résolution linéaire est complète :

Théorème :

Soit Δ est un ensemble de formules satisfiable. Si $\Delta \cup \{\neg\omega\}$ est insatisfiable, alors il existe une déduction linéaire de la clause vide de racine $\neg\omega$.

Exemple: $\{ a \vee \neg b, a \vee \neg c \vee \neg d, c, d \vee \neg c\} \Rightarrow a$

Résolution input linéaire

Chaque nouvelle clause est déduite à partir de la dernière clause produite (linéaire) et d'une clause de l'ensemble initial (input)

La résolution input linéaire est :

- incomplète en général
- complète pour les clauses de Horn (clauses avec au plus un littéral **positif)**

Théorème*:*

Si Δ est un ensemble satisfiable de clauses comportant au plus un littéral positif, ω un atome, alors ω est conséquence logique de Δ si et seulement si il existe une dérivation **input linéaire** *à partir de $\Delta \cup \{\neg\omega\}$ qui donne la clause vide.*

Exemple:

a.

b.

$a \wedge b \rightarrow c$

$a \wedge c \rightarrow d$

$a \wedge b \wedge d \rightarrow e$

* e conséquence logique?

- mise sous forme clausale:

- réfutation input linéaire à partir de **¬e**:

Trois logiciens entrent dans un bar.

Le serveur demande : est-ce que tout le monde prendra une bière?

Le premier répond : Je ne sais pas

Le deuxième répond : Je ne sais pas

Le troisième répond : Oui

Logique du premier ordre

Langage et sémantique

Variables, quantificateurs: $\forall$, $\exists$

Règles générales : $\forall X\ (\text{homme}(X) \rightarrow \text{mortel}(X))$

Description du "monde" à l'aide de relations :

parent(arthur, berthe).

parent(berthe, claude).

$\forall X\, \forall Y\, \forall Z\ (\text{parent}(X,Y) \wedge \text{parent}(Y,Z) \rightarrow \text{grand-parent}(X,Z))$

"Tous les ordinateurs de la salle C101 sont en panne"

"Dans toutes les salles, il y a au moins un ordinateur en panne"

"Si un chien a faim, alors il aboie, mais il y a des chiens qui aboient et qui n'ont pas faim"

"Tous les ordinateurs de la salle C101 sont en panne"

$$\forall X\ [\ (ordi(X)\ \wedge\ salle(X, c101))\ \rightarrow\ panne(X)]$$

"Dans toutes les salles, il y a au moins un ordinateur en panne"

$$\forall Y\ [salle(Y)\ \rightarrow\ \exists X\ \ (ordi(X)\ \wedge\ salle(X, Y) \wedge panne(X))]$$

"Si un chien a faim, alors il aboie, mais il y a des chiens qui aboient et qui n'ont pas faim"

Langage

* ensemble infini de constantes (chaînes de caractères, commençant par une lettre minuscule)

* variables : X, Y (chaînes de caractères commençant par une lettre majuscule)

* fonctions :

    f, g, h (symboles de fonctions)

    f(a,b), g(a), h(a,b,c)

* symboles de prédicat :

    p, q , r

Connecteurs logiques :

- $\neg$ (non)
- $\wedge$ (et)
- $\vee$ (ou)
- $\rightarrow$ (implique)

Quantificateurs :

- $\forall$: "quelque soit"
- $\exists$: "il existe"

Termes :

- une constante est un terme
- une variable est un terme
- si f est un symbole de fonction d'arité n et t_1, t_2, ... t_n sont des termes, alors f(t_1, t_2, ... , t_n) est un terme

Formules (bien formées) :

- si p est un symbole de prédicat et t_1, t_2, ... t_n sont des termes, alors p(t_1, t_2, .. t_n) est une formule
- si ω est une formule, alors $\neg\omega$ est une formule
- si ω_1 et ω_2 sont des formules, alors :

  $\omega_1 \wedge \omega_2$, $\omega_1 \vee \omega_2$, $\omega_1 \rightarrow \omega_2$ sont des formules

Exemples :

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

$p(f(a,b),c)$

$\forall X\ (homme(X) \rightarrow mortel(X)\ ) \wedge homme(socrate)$

$\forall X\ (\ \exists Y\ [\ (p(X,Y) \rightarrow\ \ q(f(X))\ ) \vee q(Y)\ ]\ )$

$\forall X\ [(chien(X) \wedge faim(X)) \rightarrow aboie(X)] \wedge \exists X\ (chien(X) \wedge (aboie(X) \wedge \neg faim(X))$

livre(auteur(oneil,cathy), ref(penguin_books, 2017),"Weapons of math destruction")

Sémantique

Définition d'un interprétation en logique des prédicats :

$$\forall X \, (\, \exists Y \, [\, (\, p(X,Y) \rightarrow \quad q(f(X) \,) \; \vee \; q(Y)] \;)$$

interprétation I :

Définition d'un interprétation en logique des prédicats :

Une interprétation d'une formule en logique du premier ordre est définie par:

- un domaine D, ensemble non vide
- l'assignation d'une application de D^n dans D à chaque fonction d'arité n
- l'assignation d'une application de D^n dans {vrai, faux} à chaque prédicat d'arité n

$$\forall X \, (\exists Y \, [\, (\, p(X,Y) \rightarrow \quad q(f(X)) \; \vee \; q(Y)] \;)$$

interprétation I

D={a,b}	*définition du domaine*
f: f(a)=a, f(b)=a	*définition des fonctions*
p(a,a): V , p(a,b): F p(b,a): F, p(b,b): V q(a): V, q(b): V	*définition des prédicats*

Satisfiabilité, modèles

Une interprétation ***satisfait*** une formule ω si cette formule a la valeur V sous cette interprétation.

Une interprétation qui satisfait une formule est appelée un ***modèle*** de cette formule.

Une formule qui n'a pas de modèle est ***insatisfiable*** (inconsistante)

Une formule qui a au moins un modèle est ***satisfiable***.

Une formule est ***valide*** si elle est vraie dans toute interprétation.

Évaluation d'une formule relativement à une interprétation :

- signification des connecteurs: table de vérité
- quantificateurs :

$\forall X$ f : vraie si f est vraie pour tout élément de D

$\exists X$ f : vraie s'il existe une valeur de D pour laquelle f est vraie

Exemples

Formule : $(\forall X\ p(X)) \wedge (\exists X\ \neg p(X))$

Interprétation I :

D={a,b}

p(a) : V

p(b) : F

Valeur de la formule : Faux

Formule : $\forall X \ (p(X) \rightarrow q(f(X), a))$

Exemple d'interprétation:

domaine : D={a,b}

définition des fonctions :

f(a) : b, f(b) : a

valeurs des prédicats :

p(a) : F, p(b) : V, q(a,a) : V, q(a,b) : V, q(b,a) : F, q(b,b) : V

Autre façon de définir l'interprétation :

D={a,b}, f(a)=b, f(b)=a, {p(b), q(a,a), q(a,b), q(b,b)}

Valeur de la formule ?

Inférence en logique du premier ordre

*** Règles d'inférence de la logique propositionnelle**

ω_1 , $\omega_1 \rightarrow \omega_2 \vdash \omega_2$ (modus ponens)

ω_1 , $\omega_2 \vdash \omega_1 \wedge \omega_2$ (introduction du et)

$\omega_1 \wedge \omega_2 \vdash \omega 2 \wedge \omega_1$ (commutativité du et)

$\omega_1 \wedge \omega 2 \vdash \omega_1$ (élimination du et)

$\omega_1 \vdash \omega_1 \vee \omega_2$ (introduction du ou)

$\omega_1 \vee \omega_2 \vdash \omega_2 \vee \omega_1$ (commutativité du ou)

*** Règles prenant en compte les quantificateurs**

Instanciation :

A partir de $\forall X\ (\omega(X))$, on infère $\omega(a)$, où a est une constante et $\omega(a)$ est la formule obtenue en substituant a à X dans $\omega(X)$

Généralisation existentielle :

A partir de $\omega(a)$, on infère $\exists X\ \omega(X)$

Equivalences :

$\neg\ \forall X\ \omega(X) \equiv \exists\ X\ \neg\omega(X)$

$\neg\ \exists X\ \omega(X) \equiv\ \forall\ X\ \neg\omega(X)$

Résolution en calcul des prédicats

* déduction par résolution en logique propositionnelle :
à partir de :

$$\mathbf{p} \vee c1 \ , \ \neg \mathbf{p} \vee c2$$

on déduit :

$$c1 \vee c2$$

* logique du premier ordre :

$\forall X \ (\neg \ homme(X) \vee mortel(X))$ $\qquad$ $homme(socrate)$

mortel(socrate)

$\forall X \forall Y (p(X) \vee q(X,Y))$ $\qquad$ $\forall Z \ (\neg p(f(a,b)) \vee r(Z))$

Unification

* Si on a deux littéraux complémentaires de la forme $\lambda(X)$ et $\neg\lambda(t)$,

où *t* est un terme et *X* une variable, on peut substituer *t* à *X* dans la première clause et effectuer une résolution entre les deux clauses.

unification : calcul d'une substitution qui permet de faire "coincider" deux littéraux

Substitution

Un *composant de substitution* est une expression de la forme *(X/t)*, où *X* est une variable et t un terme quelconque.

Si *A* est une formule, on note *(X/t)A* la formule obtenue en remplaçant *X* par *t* dans *A*.

Une *substitution* est une application *s* qui à une formule *A* associe $c_1c_2...c_k(A)$, où c_1, c_2..., c_k sont des composants de substitutions. On note $s=c_1\,c_2...c_k$

Exemple :

s=(X/f(a))(Y/f(X))

s(p(X,Y))=

Définition :

Deux termes ***t1*** et ***t2*** sont unifiables s'il existe une substitution s telle que ***s(t1)=s(t2)***

On appelle unificateur de ***t1*** *et* ***t2*** toute substitution s telle que :

s(t1)=s(t2)

Exemple : t1=p(X,f(X)), t2=p(a,Y)

unificateur :

Plus grand unificateur (unificateur le plus général)

Soit *t1 et t2 deux* termes, *U* l'ensemble des unificateurs de *t1 et t2*.

Soit *s* une substitution de *U*. Alors *s* est un plus grand unificateur de *A* si :

$\forall u \in U, \exists v \in U(A), u = v \circ s$

(u est une substitution "moins générale" que s)

Algorithme d'unification : calcul du plus grand unificateur de t1 et t2

s ← ⊘
tant que **s(t1) ≠ s(t2)**
- déterminer le premier symbole de s(t1) différent du symbole de s(t2) de même rang
- déterminer les sous-termes de s(t1) et s(t2) commençant à ce symbole

si aucun n'est une variable
alors échec, t1 et t2 ne sont pas unifiables
sinon
soit **X** la variable et **t** l'autre terme,
s ← (X/t) s
finsi
fin tant que

Exemples :

p(X, f(X), a) et p(d,Y,Z)

s=(X/d)

p(d, f(d),a) p(d,Y,Z)

…

p(X,f(X),a) et p(d,Y, Y) ?

Mise sous forme clausale d'une formule

Toute formule en logique des prédicats peut-être transformée en un ensemble de clause :

- en appliquant les mêmes transformations qu'en logique propositionnelle
- en "éliminant" les quantificateurs

<u>suppression des implications et des négations portant sur une formule :</u>

1- Enlever les implications en appliquant l'équivalence :

$(p \rightarrow q) \equiv \neg p \vee q$

2- Réduire la portée des $\neg$:

- appliquer les règles d'équivalence :

  remplacer $\neg \forall X\ \omega(X)$ par $\exists X\ \neg\omega(X)$

  remplacer $\neg \exists X\ \omega(X)$ par $\forall X\ \neg\omega(X)$

- remplacer $\neg(\neg F)$ par F
- appliquer les lois de De Morgan :

  $\neg (\omega_1 \vee \omega_2) \equiv \neg\omega_1 \wedge \neg\omega_2$

  $\neg (\omega_1 \wedge \omega_2) \equiv \neg\omega_1 \vee \neg\omega_2$

variables et quantificateurs :

3- Renommer les variables, pour que chaque quantificateur porte sur un nom différent

exemple : ($\forall X\ p(X) \vee \exists X\ q(X)$) est remplacé par: $\forall X\ (p(X) \vee \exists Y\ q(Y))$

4- Eliminer les quantificateurs **existentiels** (mise sous forme de Skolem) :

- remplacer chaque variable quantifiée existentiellement :

- par une fonction dont les arguments sont les variables quantifiées universellement dont elle dépend (fonction de Skolem).

$\forall X\ \exists Y$ (poids(X,Y)): Y dépend de X, on remplace Y par f(X).

$\forall X$ (poids(X,f(X)))

- ou par une constante si elle ne dépend d'aucune variable (quantifiée universellement)

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

5- Mettre sous forme prénexe

placer tous les quantificateurs (universels) en tête de la formule

6- Supprimer les quantificateurs: toutes les variables restantes sont quantifiées universellement, on peut donc omettre les quantificateurs

forme normale conjonctive :

7- Mettre sous forme normale conjonctive en appliquant les lois de distributivité et associativité

8 - Mise sous forme d'un ensemble de clauses en "supprimant" les connecteurs

Exemples :

1) $\forall X \ \forall Y \ [\ (\exists Z \ (p(X,Z) \wedge p(Y,Z)) \rightarrow (\exists U \ q(X,Y,U)) \ ]$

2) On considère l'énoncé suivant :

Tout homme est un primate

Les dauphins ne sont pas des primates

Il y a des dauphins qui sont intelligents

Résolution en logique du premier ordre

Soient *F* et *G* deux clauses sans variables communes (les variables sont renommées si nécessaire). Alors la clause *H* est un résolvant de *F* et *G* si :

- *F* est de la forme ***A*** $\vee$ *F1*
- *G* est de la forme $\neg$***B*** $\vee$ *G1*
- ***A* et *B*** sont unifiables, avec *s* un plus grand unificateur de *A* et *B*
- *H* est la clause *s(F1* $\vee$ *G1)*

Exemples :

$\neg$ homme(X) $\vee$ mortel(X)) homme(socrate)

p(X)$\vee$*q(X,Y)* $\neg$p(f(a,b))$\vee$ r(Z)

Correction et complétude

Correction

La résolution en logique des prédicats est correcte :

s'il existe une preuve par résolution de F à partir d'un ensemble de clauses Δ, alors F est conséquence logique de Δ.

Complétude

Comme en logique propositionnelle, la procédure de résolution n'est pas complète en général : on ne peut pas déduire toutes les conséquences logiques d'un ensemble de formules par résolution.

Réfutation

La résolution par réfutation est **correcte** et **complète** :

Si Δ est un ensemble de clauses satisfiables, alors ω est conséquence logique de Δ si et seulement si la résolution par réfutation à partir de $\Delta \cup \{\neg\omega\}$ produit la clause vide.

Réfutation pour les clauses de Horn

Définition: Une clause de Horn est une clause comportant un seul littéral positif.

Exemple :

$$p(X) \vee \neg q(Y) \vee \neg p(Y,Z)$$

$$q(Y) \wedge p(Y,Z) \rightarrow p(X)$$

Théorème

La résolution par réfutation **input linéaire** est complète pour les **clauses de Horn** :

Si $\boldsymbol{\Delta}$ *est un ensemble de clauses comportant* ***un seul littéral positif****,* $\boldsymbol{\omega}$ *un atome, alors* $\boldsymbol{\omega}$ *est conséquence logique de* Δ *si et seulement si il existe une dérivation input linéaire de* $\boldsymbol{\Delta \cup \{\neg \omega\}}$ *qui donne la clause vide.*

Remarque

Si c n'est pas conséquence logique de Δ, il se peut que la procédure de résolution par réfutation ne termine pas.

Exemple

$p(X) \vee \neg r(X)$

$p(X) \vee \neg q(f(X)) \vee \neg q(X)$

$q(f(X)) \vee \neg q(X)$

$q(a)$

Réfutation pour montrer que p(a) est conséquence logique de la base :

Exercice

A partir des affirmations suivantes :

Tout homme est un primate

Les dauphins ne sont pas des primates

Il y a des dauphins qui sont intelligents

effectuer une résolution pour montrer que :

on peut ne pas être un homme et être intelligent

Prolog

Langage de programmation basé sur la logique des prédicats

Représentation de connaissances en logique du premier ordre sous forme de clauses de Horn :

$p(X,Y) \wedge r(Y) \rightarrow q(X)$

équivalent à:

$\neg p(X,Y) \vee \neg r(Y) \vee q(X)$

On note:

```
q(X) :-
    p(X,Y), r(Y).
```

Règles, faits.

* Faits: un seul terme positif, sans variables (terme clos) :

```
p(a,b).
```

représente des données

* Règles: clauses de Horn, un seul terme positif :

```
q(X) :-
      p(X,Y)
```

Interrogation de la base :

```
?- q(X)
```

Programme Prolog :

ensemble de clauses de Horn, faits et règles

Exécution d'un programme:

Construction d'un arbre de résolution par réfutation pour prouver un but

Parcours de l'arbre en utilisant une stratégie particulière

Exemple

Soit la base suivante :

```
parent(patricia , bruno).
parent(thomas , bruno).
parent(thomas, elisabeth).
parent(bruno, anne).
parent(bruno, marie).
parent(marie,jean).
femme(patricia).
femme(elisabeth)
femme(marie).
femme(anne).
homme(bruno).
homme(thomas).
```

Interrogation de la base :

?- parent(bruno,marie).

oui

Calcul effectué: réfutation à partir de la base et ¬parent(bruno,marie)

Réponse: oui si on obtient la clause vide

?-parent(bruno,jean).

non

?-parent(bruno,X).

X=anne;

X=marie;

Questions avec plusieurs buts :

Trouver les petits-enfants de patricia :

trouver tous les Y tels que parent(patricia,X) **et** parent(X,Y) :

```
?-parent(patricia,X),parent(X,Y).
X=bruno, Y=anne
X=bruno, Y=marie
```

Introduction de règles:

```
pere(X,Y) :- parent(X,Y), homme(X).
mere(X,Y) :- parent(X,Y), femme(X).
grand_parent(X,Y):- parent(X,Z), parent(Z,Y).
```

Exemple de questions :

est-ce que bruno est le père de anne

```
?-pere(bruno,anne)
```

trouver tous les couples (X,Y) tels que parent(X,Y) et femme(X) soient dans la base

```
?-mere(X,Y)
```

trouver tous les grands-parents de marie

```
?-grand_parent(X,marie)
```

trouver tous les petits-enfants de bruno

```
?-grand_parent(bruno,X)
```

Mécanisme de résolution de Prolog

Résolution par réfutation :

dérivation input linéaire à partir du "but"

Exemple

But:
p(a)
Règle:
p(X) :- q(X),r(X).

X et a sont unifiables, avec comme unificateur (X/a), on peut appliquer la règle c'est à dire effectuer une résolution entre :
¬p(a) et p(X) ∨ ¬q(X) ∨ ¬r(X)

On obtient une nouvelle clause :
¬q(a) ∨ ¬r(a)

Calcul en Prolog :

* dérivation input linéaire en choisissant le premier sous-but de la liste.
* construction et exploration de l'arbre de dérivation correspondant en profondeur d'abord avec backtracking (retour arrière).

Chaque clause vide trouvée correspond à une réponse.

Exemple:

grand_parent(thomas,X):

Règles récursives

Exemple:

définir la relation ancêtre

X est un ancêtre de Y si X est parent de Y

Règles:

```
ancetre(X,Y) :-
        parent(X,Y).
ancetre(X,Y) :-
        parent(X,Z),
        ancetre(Z,Y).
```

Exemples de questions:

```
?-ancetre(patricia,X).
?-ancetre(X,Y).
```

Problème: terminaison du programme.

* cycle dans le graphe de la relation

* branches infinies possibles dans l'arbre de dérivation, selon l'ordre des sous-buts et l'ordre de règles

```
ancetre(X,Y) :-
        ancetre(Z,Y),
        parent(X,Z).
ancetre(X,Y) :-
        parent(X,Y).
```

Equivalent d'un point de vue logique, mais boucle...

Représentation de données structurées

données en Prolog:

termes en logique des prédicats

représentation de données structurée à l'aide de fonctions

exemples:

date_et_heure(date(J,M,A), heure(H,M,S))

personne(etatcivil(Nom,Prenom,datenaiss(J,M,A)), adresse(Num,Rue,Ville))

livre(auteur(Nom, Prenom),ref(Editeur, Annee), Titre)

unification: permet d'extraire des données à partir de termes structurés :

```
nom(X) :-
        personne(etatcivil(X,Y,Z),W).
```

tous les noms:

```
?-nom(X).
```

noms et prénoms:

```
?-personne(etatcivil(X,Y,_),_).
```

Exemple:

livre(auteur(nilsson,nils) , ref(morgan_kaufman , 1998) ,"Artificial Intelligence: A new Synthesis").

livre(auteur(poole,david), ref(oxford_press, 1998), "Computational Intelligence: A logical approach").

livre(auteur(russel, stuart),ref(prentice_hall, 2009), "Artificial Intelligence: A modern approach, 3rd Edition").

livre(auteur(russel, stuart),ref(pearson, 2010), "Intelligence Artificielle, 3ième édition")

livre(auteur(blackburn, patrick),ref(College Publications, 2006), "Learn Prolog Now!").

livre(auteur(bratko,ivan),ref(Addison Wesley, 2011),"Prolog Programming for Artificial Intelligence").

livre(auteur(allemag,dean),ref(morgan_kaufmann, 2011),"Semantic Web for the Working Ontologist")

livre(auteur(ducharme,bob),ref(o_reilly, 2013),"Learning Sparql,2nd edition")

livre(auteur(oneil,cathy), ref(penguin_books, 2017),"Weapons of math destruction")

Toutes les livres de Stuart Russel :

```
?-livre(auteur(russel,stuart), X, Y).
```

tous les titres:

```
?-livre(_ , _, X).
```

tous les titres des livres parus en 2011:

tous les titres parus après 2010:

Listes

Codage d'une liste à l'aide d'un terme structuré:

- symbole de fontion (liste)
- constante représentant la liste vide (nil)

liste(a,liste(b,liste(c,nil))): a,b,c

liste(a,nil): a

liste(a,L): liste commençant par a

Notation particulière en Prolog:

[ ] : liste vide

[a,b,c,d] : liste énumérée

[X | L] : liste commençant par X

Exemple:

```
appartient(X,[X|L]).
appartient(X,[Y|L]) :-
   appartient(X,L).
```

Exemples d'utilisation:

?-appartient(b,[a,b,c]).

?- appartient(X, [a,b,c]).

Prédicats prédéfinis

Comparaison de termes : \==

```
frere(X,Y) :- parent(Z,X), parent(Z,Y), X\==Y.
```

Arithmétique : **is**

```
nbelem([],R,R).
nbelem([X|L],N,R) :-
        NN is N+1,
        nbelem(L,NN,R)
```

Logique et représentation des connaissances:

Prolog
Système à base de règles
Bases de connaissances

Systèmes à base de règles

Systèmes experts

Logiciel supposé reproduire le raisonnement d'un expert, dans un domaine donné.

- base de connaissances: faits, règles

- moteur d'inférences pour effectuer des déductions

exemple:

BF={a, b}.

si a et b alors c

si a et c alors d

Systèmes à base de règles

Application: gestion des règles métiers

Implémentation des "règles métiers" ("business rules") dans une application

Règles métiers (Business Rules)

Spécifications décrivant les opérations et stratégies d'un métier:
stratégie marketing, politique de prix, gestion de la relation client

Intérêt de la représentation par règle:

- langage déclaratif, utilisable par un expert du domaine
- directement exécutable

Application dans les domaines où ces règles doivent pouvoir changer rapidement et fréquemment

Exemples (ILOG):

IF the shopping cart contains more than $100 worth of CDs
THEN give the customer $5 off the next purchase

IF Customer traveled to Europe within the last year,
THEN send notification of European travel promotions.

IF destination is preferred number
THEN give a 50% discount on call.

"Un système de gestion de règles métier (BRMS, business rule management system) permet de définir, déployer et gérer les politiques métier, et les décisions opérationnelles qui en découlent, indépendamment du code applicatif.

En externalisant les règles métier et en offrant les outils pour les gérer, un BRMS permet aux experts métier de définir et de gérer les décisions qui régissent le comportement des systèmes, réduisant ainsi le temps et le travail nécessaires pour mettre à jour les systèmes de production, et augmentant la capacité de l'entreprise à réagir aux changements environnants. "

https://www-01.ibm.com/software/fr/websphere/products/business-rule-management/whatis/

BRMS: Business rules management systems

* ***IBM Operational Decision Manager***

https://www.ibm.com/cloud/automation-software/business-decision

**** DROOLS***

https://www.drools.org/

**** BLAZE ADIVSOR***

http://www.fico.com/en/products/fico-blaze-advisor-decision-rules-management-system

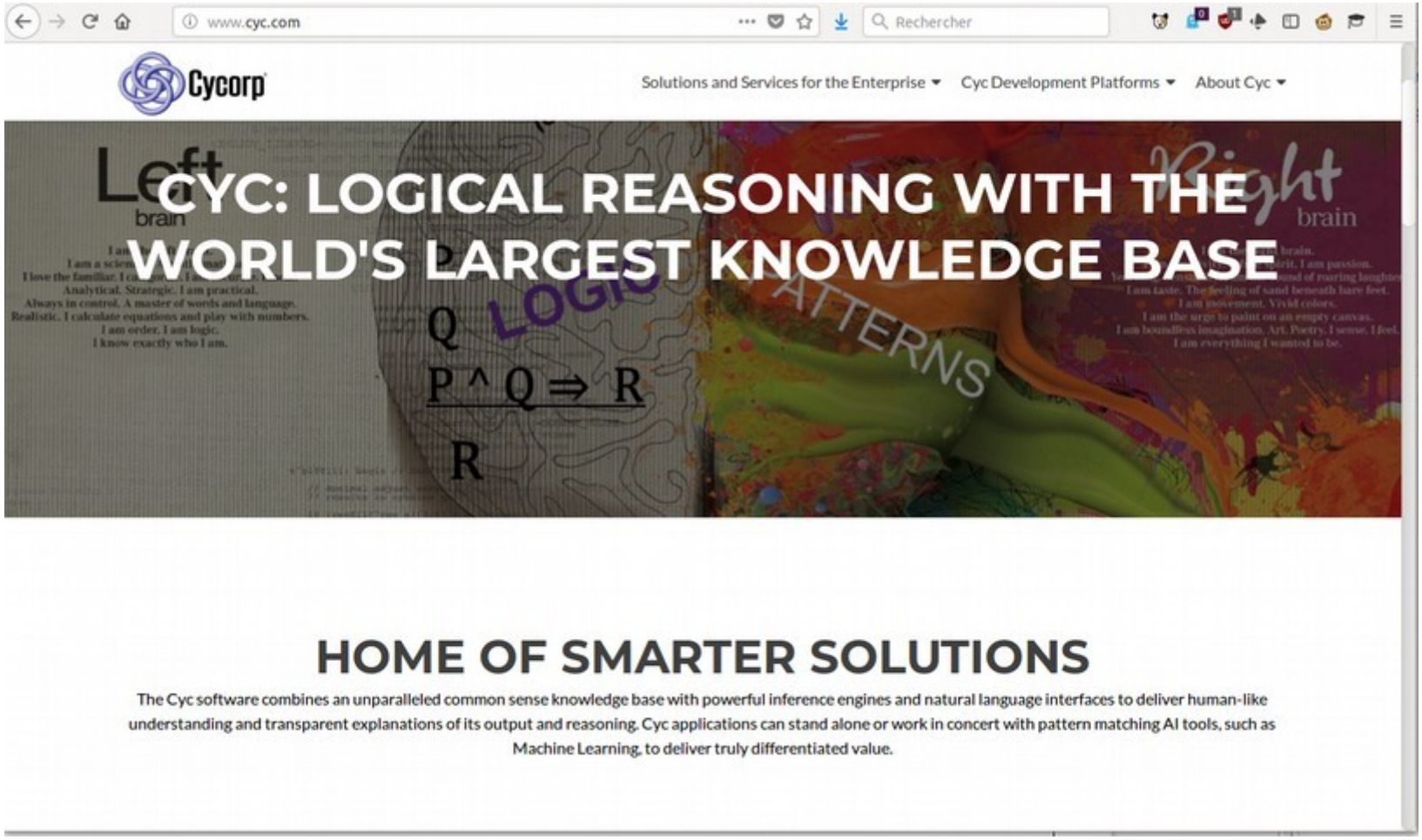
Systèmes à base de connaissance

* base de connaissance:
    * données, informations
    * règles

* moteur d'inférence:
    déduction, raisonnement sur les connaissances

Base de connaissances: CYC

www.cyc.com



Vers une base de connaissance universelle?

projet en cours depuis 1984, Doug Lenat - Cycorp (1994)

Modéliser et incorporer toutes les connaissances implicites et explicites

Développement d'une base de connaissance et de mécanismes d'inférences

Pour simuler le "raisonnement de sens commun"

pour créer des programmes permettant de "raisonner" intelligemment dans un contexte quelconque

Cyc ontologie: plusieurs centaines de milliers de termes, plusieurs millions de relations

Cyc Inference Engine, ***Cyc Knowledge Base Browse****r*.

Exemple: analyse du langage naturel

Fred saw *the plane flying* over Zurich.
***Fred* saw the mountains *flying* over Zurich.**

Exemple: analyse du langage naturel

Fred saw *the plane flying* over Zurich.
Fred **saw the mountains *flying* over Zurich.**

IBM DeepQA, Watson

Créer un système informatique capable de répondre à des questions ouvertes et variées

** formulées en langage naturel*

** pouvant porter sur un grand nombre de domaines*

Watson: programme capable de jouer contre des humains au jeu télévisé "Jeopardy"

"Watson is powered by 10 racks of IBM Power 750 servers running Linux, and uses 15 terabytes of RAM, 2,880 processor cores and is capable of operating at 80 teraflops"

http://www.research.ibm.com/deepqa/deepqa.shtml

https://fr.wikipedia.org/wiki/Watson_(intelligence_artificielle)

IBM Watson: Beyond Jeopardy! Q&A (*https://learning.acm.org/webinar/lally.cfm)*

Q: In what programming language(s) are the algorithms implemented?
A: Watson is implemented mostly in Java, but there are a few C components and some Prolog.

Q: How much Prolog or Horn clause-type programming does Watson use?
A: *Mainly we used Prolog for pattern matching over natural language parse trees. Here is a link to an article that provides more detail of how we used Prolog: http://www.cs.nmsu.edu/ALP/2011/03/natural-language-processing-with-prolog-in-the-ibm-watson-system/.*

Q: Did you use multiple ontologies or did you build a Jeopardy! task-specific ontology?
A: *Watson doesn't require an ontology, but evidence scoring algorithms can make use of ontologies. For example, we did make extensive use of Dbpedia and the YAGO-type ontology for Jeopardy!. However such structured data was just one small part of Watson. Watson gets most of its evidence from unstructured text.*

DBPEDIA

YAGO

KNOWLEDGE GRAPH