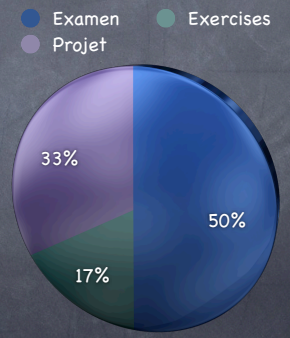
Systèmes Experts 3

Richard.Moot@labri.fr
http://www.labri.fr/perso/moot/SE/

Systèmes Experts

- Exercices à faire et à rendre pendant le TD du 21 octobre 2009.
- Projet: développement et programmation d'un mini système expert en Prolog
- Examen



Plan de cours 3

- Rappel de la résolution,
- Résolution avec des clauses Horn,
- Introduction relativement informelle à la logique des prédicats et clauses de Horn pour un partie de cette logique.
- Exemples d'applications.

Prochain cours

- Calcul de séquents pour faire des preuves formelles en logique des prédicats.
- Suivi par des exercices dans le TD du 14 octobre qui vont vous aider à bien faire les exercices du 21 octobre (partie du note final pour ce cours).

Exercises

- Rappel de la résolution,
- Résolution avec des clauses Horn,
- Introduction relativement informelle à la logique des prédicats et clauses de Horn pour un partie de cette logique.
- Prochain cours (et TD de la semaine prochaine): calcul de séquents.

Résolution

- La résolution est un des stratégies de base pour la recherche de preuve automatique,
- Prolog utilise un forme de la résolution pour la démonstration automatique.

Résolution

- On veut démontrer que X est un tautologie.
- On transforme la formule $\neg X$ en une formule Y qui est équivalent et en forme normale conjonctive $C_1 \wedge \ldots \wedge C_n$. On parle d'une conjonction de clauses.
- Chaque des C_i est du forme $A_1 \vee \ldots \vee A_n$, on parle d'une disjonction des littéraux.

Résolution

- Chaque des Ci est du forme $A_1 \vee \ldots \vee A_n$, on parle d'une disjonction des littéraux (des formules atomiques et leurs négations).
- Chaque fois qu'on trouve un clause $p \vee A_1 \vee \ldots \vee A_n$ et un clause $\neg p \vee B_1 \vee \ldots \vee B_n$ on ajoute un clause $A_1 \vee \ldots \vee A_n \vee B_1 \vee \ldots \vee B_n$.

Résolution

- Chaque des C_i est du forme $A_1 \vee \ldots \vee A_n$, on parle d'une disjonction des littéraux (des formules atomiques et leurs négations).
- Chaque fois qu'on trouve une clause $p \vee A_1 \vee \ldots \vee A_n$ et une clause $\neg p \vee B_1 \vee \ldots \vee B_n$ on ajoute un clause $A_1 \vee \ldots \vee A_n \vee B_1 \vee \ldots \vee B_n$. Ceci est la seule règle! Peut surprenant cette règle s'appelle la résolution.

Résolution

- On continue jusqu'au moment ou:
  - on ne peut plus appliquer la règle de résolution pour trouver des nouvelles clauses
  - on a démontré la clause vide: $\perp$
- Etant donné que à partir de $\neg X$ on déduit $\perp$, X est un tautologie. (pourquoi?)

Résolution

- Si à partir de $\neg X$ on déduit $\perp$, $\neg X$ est une contradiction et X est un tautologie. (pourquoi?)
- Sinon, X n'est pas un tautologie: X est peut-être une formule valable, peut-être une contradiction.

Résolution

- $\neg X \leftrightarrow Y$
- $Y \rightarrow \perp$
- $\neg X \rightarrow \perp$
- X

Forme Normale Conjonctive/Disjonctive

Conjonction $\wedge$

Forme Normale Conjonctive/Disjonctive

Conjonction $\wedge$

de disjonctions $\vee$ ...

Forme Normale Conjonctive/Disjonctive

Conjonction $\wedge$

de disjonctions $\vee$ ...

des atomes p

Forme Normale Conjonctive/Disjonctive

Conjonction $\wedge$

de disjonctions $\vee$ ...

des atomes $\neg p$ p

et leur négations

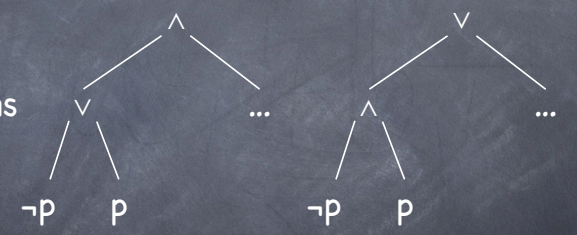
Forme Normale Conjonctive/Disjonctive

Conjonction

de disjonctions

des atomes

et leur négations



On appelle les atomes et leur négations <u>des littéraux</u>

Transformer en forme normal conjonctive

- Utiliser les définitions de → et ↔ pour assurer que la formule contient que ¬ , ∧ et ∨.
- Utiliser les lois De Morgan pour bouger les négations vers les formules atomiques.
- Utiliser les lois de distribution pour bouger les conjonctions vers l'extérieur.

Résolution: Exemple Forme Normale Conjonctive

On démontre:

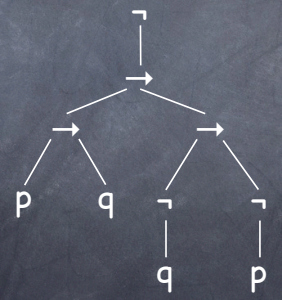
$(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$

A partir de sa négation

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

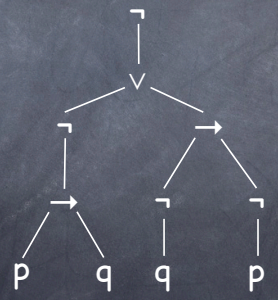
Résolution: Exemple Forme Normale Conjonctive

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



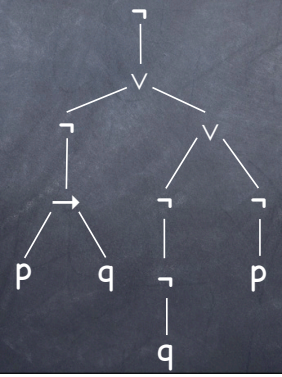
Résolution: Exemple
Forme Normale Conjonctive

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



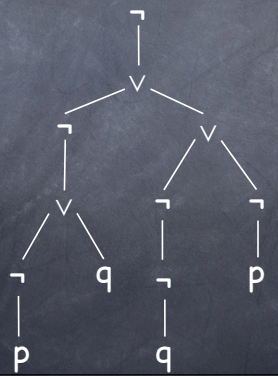
Résolution: Exemple
Forme Normale Conjonctive

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



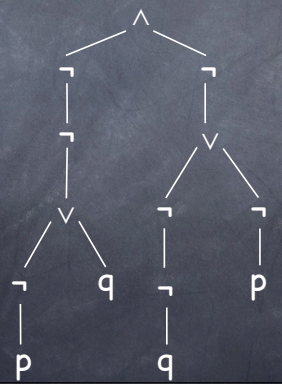
Résolution: Exemple
Forme Normale Conjonctive

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



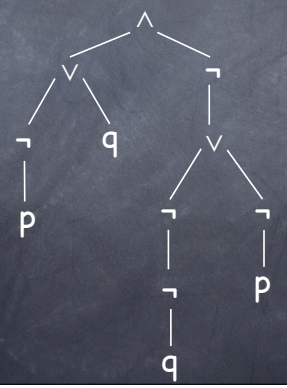
Résolution: Exemple
Forme Normale Conjonctive

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



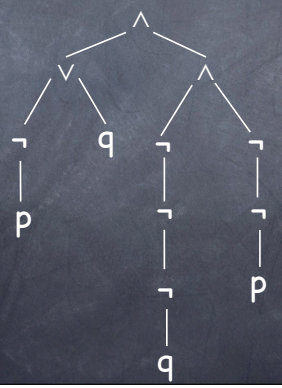
Résolution: Exemple
Forme Normale Conjonctive

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



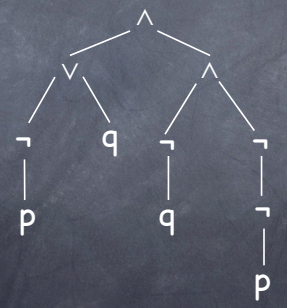
Résolution: Exemple
Forme Normale Conjonctive

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



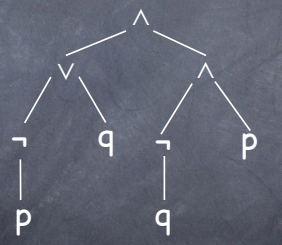
Résolution: Exemple
Forme Normale Conjonctive

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



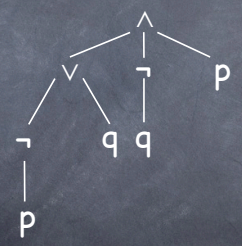
Résolution: Exemple
Forme Normale Conjonctive

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



Résolution: Exemple
Forme Normale Conjonctive

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



$\neg p \vee q \ \wedge \ \neg q \ \wedge \ p$

Résolution: Exemple
Clauses

$\neg p \vee q \ \wedge \ \neg q \ \wedge \ p$

Résolution: Exemple
Clauses

1. $\neg p \vee q \ \wedge$
2. $\neg q \ \wedge$
3. p

Résolution: Exemple
Résolution

1. $\neg p \vee q \ \wedge$
2. $\neg q \ \wedge$
3. p
4. q Résolution (1,3)

Résolution: Exemple Résolution

1. $\neg p \vee q \ \wedge$
2. $\neg q \ \wedge$
3. p
4. q Résolution (1,3)
5. $\neg p$ Résolution (1,2)

Résolution: Exemple Résolution

1. $\neg p \vee q \ \wedge$
2. $\neg q \ \wedge$
3. p
4. q Résolution (1,3)
5. $\neg p$ Résolution (1,2)
6. $\bot$ Résolution (3,5)

Résolution: Exemple Résolution

1. $\neg p \vee q \ \wedge$
2. $\neg q \ \wedge$
3. p
4. q Résolution (1,3)
5. $\neg p$ Résolution (1,2)
6. $\bot$ Résolution (3,5)
   ou Résolution (2,4)

Clauses de Horn

- Un clause de Horn H est une disjonction qui contient exactement un littéral positif.
- C'est-à-dire: un clause de Horn a une unique formule atomique sans négation, les autres formules ont toutes une négation.
- Une question (anglais: query) Q est une disjonction qui contient que des littéraux négatifs.

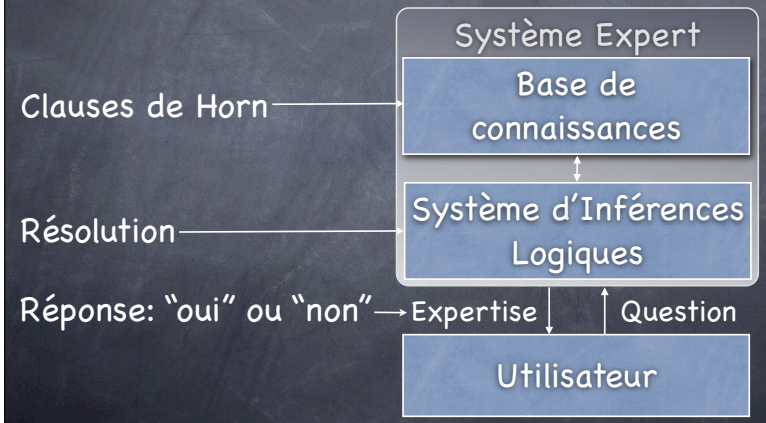
Clauses de Horn

- La résolution pour des clauses de Horn utilise les même principes que la résolution pour des clauses de disjonctions. (pourquoi?)
- Un programme P est une conjonction de clauses de Horn, c'est-à-dire $P \equiv H_1 \wedge \ldots \wedge H_n$.
- La réponse à une question Q pour un programme P est "vrai" si et seulement si on déduit la clause vide à partir de $P \wedge Q$.

Clauses de Horn

- Dans le contexte de systèmes experts, le clauses de Horn sont la base de connaissances,
- Une question interroge la base de connaissances et donne une réponse de "oui" si et seulement si la question est une conséquence de la base des connaissances.
- Ceci nous donne déjà un système expert version "light".
- On verra pourquoi

Clauses de Horn, questions et résolution



Résolution: Exemple Clauses de Horn

1. $\neg p \vee q$
2. $\neg q$
3. p

Résolution: Exemple Clauses de Horn

1. $\neg p \vee q$
2. $\neg q$ ⟵ Question
3. p Que des littéraux négatifs

Résolution: Exemple Clauses de Horn

Programme $\neg p \vee q$
p

Question $\neg q$

Résolution: Exemple Clauses de Horn

Programme $\neg p \vee q$
p

Question $\neg q$

Les clauses de Horn s'écrivent comme des implications

Résolution: Exemple Clauses de Horn

Programme $p \rightarrow q$
p

Question $\neg q$

De plus:
en Prolog, on inverse la direction de l'implication.
Ceci est juste une différence de notation.

Résolution: Exemple Clauses de Horn

Programme $q \leftarrow p$

p

On distingue entre les faits qui n'ont pas d'implication (ou sont du forme $p \leftarrow T$, si vous préférez)

Question $\neg q$

Résolution: Exemple Clauses de Horn

Programme $q \leftarrow p$

p

Et les règles, qui ont un implication

Question $\neg q$

Résolution: Exemple Clauses de Horn

Programme $q \leftarrow p$

p

Pour montrer que $\neg q$ donne une contradiction avec les clauses du programme on utilise la règle de résolution comme avant

Question $\neg q$

Résolution: Exemple Clauses de Horn

Programme $q \leftarrow p$

p

Pour montrer que $\neg q$ donne une contradiction avec les clauses du programme on utilise la règle de résolution comme avant

Question $\neg q$

La question contient que des littéraux négatifs et un clause de Horn contient exactement un littéral positif (la conclusion de l'implication).

Résolution: Exemple Clauses de Horn

Programme $q \leftarrow p$
p

Question $\neg q$

Pour montrer que $\neg q$ donne une contradiction avec les clauses du programme on utilise la règle de résolution comme avant

La question contient que des littéraux négatifs et un clause de Horn contient exactement un littéral positif (la conclusion de l'implication).

Alors, quand on applique la règle de résolution on remplace un littéral négatif de la question par les littéraux négatifs d'un règle.

Résolution: Exemple Clauses de Horn

Programme $q \leftarrow p$
p

Question $\neg p$

Pour montrer que $\neg q$ donne une contradiction avec les clauses du programme on utilise la règle de résolution comme avant

La question contient que des littéraux négatifs et un clause de Horn contient exactement un littéral positif (la conclusion de l'implication).

Alors, quand on applique la règle de résolution on remplace un littéral négatif de la question par les littéraux négatifs d'un règle.

Résolution: Exemple Clauses de Horn

Programme $q \leftarrow p$
p

Question $\neg p$

Pour un fait, on élimine simplement le littéral négatif qui en correspond.

On finit avec la question vide, qui correspond à une contradiction, comme avant.

Résolution: Exemple Clauses de Horn

Programme $q \leftarrow p$
p

Question $\bot$

Pour un fait (un Horn clause sans implication), on élimine simplement le littéral négatif qui en correspond.

On finit avec la question vide, qui correspond à une contradiction, comme avant.

Résolution: Exemple
Clauses de Horn

Programme $q \leftarrow p$
p

Question $\bot$

Pour un fait (un Horn clause sans implication), on élimine simplement le littéral négatif qui en correspond.

On finit avec la question vide, qui correspond à une contradiction, comme avant.

Souvent, pour nous simplifier la vie, on ne met pas les négations pour la question.

Résolution
Clauses de Horn

$\neg s \vee b \vee t$
$\neg b \vee r$
$\neg b \vee m$
$\neg a \vee r$
$\neg a \vee m$
$\neg r$
s

$\neg t$

On reprend l'exemple du dernier cours

s: il est venu seul
t: il a pris le train
a: il a pris son automobile
b: il a pris le bus
r: il est arrivé en retard
m: il a manqué la réunion

Résolution
Clauses de Horn

$\neg s \vee b \vee t$
$\neg b \vee r$
$\neg b \vee m$
$\neg a \vee r$
$\neg a \vee m$
$\neg r$
s

$\neg t$

On reprend l'exemple du dernier cours

On a déjà vu la traduction en forme normale conjonctive.

Il y a deux clauses qui ne sont pas des clauses de Horn (indiqué en bleu).

La question est "a-t-il pris le train?"

Résolution
Clauses de Horn

$\neg s \vee b \vee t$
$\neg b \vee r$
$\neg b \vee m$
$\neg a \vee r$
$\neg a \vee m$
$\neg r$
s

$\neg t$

On reprend l'exemple du dernier cours

Même s'il y a quelques clauses qui ne sont pas de clauses de Horn, on peut — pour ce programme et cette question, mais pas toujours ! — obtenir de clauses de Horn en faisant des résolutions.
Ceci est un forme de compilation

Résolution
Clauses de Horn

¬s∨b∨t
¬b∨r
¬b∨m
¬a∨r
¬a∨m
¬r
s

¬t

On reprend l'exemple du dernier cours

Alors, étant donné qu'on a ¬r, on peut résoudre ce fait avec ¬b∨r et ¬a∨r

Résolution
Clauses de Horn

¬s∨b∨t
¬b
¬b∨m
¬a
¬a∨m
¬r
s

¬t

On obtient deux nouveaux clauses: ¬b et ¬a qui ne sont pas des clauses de Horn non plus!

Maintenant, le clause ¬r n'a aucun formule atomique en commun avec un autre clause ou la question, alors on peut le supprimer

Résolution
Clauses de Horn

¬s∨b∨t
¬b
¬b∨m
¬a
¬a∨m
s

¬t

On peut résoudre ¬b comme on a fait pour ¬r et obtenir un clause ¬s∨t

Maintenant, ni ¬a ni ¬b peuvent résoudre contre la question ou un des autres clause, alors on peut les supprimer.

Résolution
Clauses de Horn

¬s∨t
¬b∨m
¬a∨m
s

¬t

Résolution
Clauses de Horn

$\neg s \vee t$
$\neg b \vee m$
$\neg a \vee m$
s

$\neg t$

Maintenant le programme contient que des clauses de Horn et on l'écrit avec des implications.

Résolution
Clauses de Horn

$t \leftarrow s$
$m \leftarrow b$
$m \leftarrow a$
s

t

Les clauses de Horn s'écrivent, naturellement en forme d'implication.

Prolog utilise des clause de Horn pour encoder sa base de connaissances et la résolution pour en déduire de réponses aux questions.

Notez encore qu'on inverse les implications et qu'on laisse la négation pour la question implicite.

Résolution
Clauses de Horn

$t \leftarrow s$
$m \leftarrow b$
$m \leftarrow a$
s

t

Il est très important que ce programme n'est pas équivalent au premier programme (pourquoi pas?)

Existe-t-il un programme en clauses de Horn qui est équivalent au premier programme? (pourquoi/pourquoi pas?)

Par contre, pour la question "t?" ce programme donne la bonne réponse (pourquoi?)

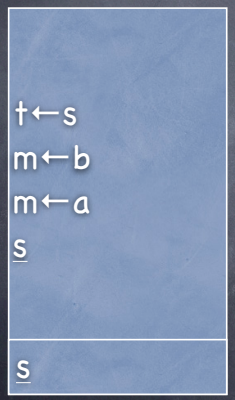
Résolution
Clauses de Horn

$t \leftarrow s$
$m \leftarrow b$
$m \leftarrow a$
s

t

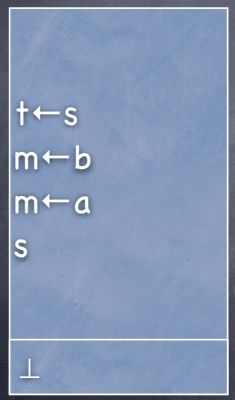
Par contre, pour la question "t?" ce programme donne la bonne réponse (pourquoi?)

Résolution
Clauses de Horn



Par contre, pour la question "t?" ce programme donne la bonne réponse (pourquoi?)

Résolution
Clauses de Horn



Par contre, pour la question "t?" ce programme donne la bonne réponse (pourquoi?)

Logique des prédicats

- Pour beaucoup d'applications (y compris la modélisation des connaissances!) la logique prépositionnel est trop simple.
- La logique des prédicats est une extension de la logique prépositionnel.
- C'est-à-dire qu'on garde tout les choses qu'on avait déjà, mais on en ajoute des autres.

Logique des prédicats

- Notamment, on ajoute des constants et des variables. Des constants sont des noms pour des entités dans notre domaine d'interprétation.
- De plus un a des relations entre des entités dans notre domaine.
- Finalement, on ajoute les constructions qui correspondent à "pour chaque x" et "il existe un x tel que"

Logique des prédicats

- Ce qu'on va pouvoir faire c'est de dire des chose comme:
  - "Jean aime Marie" aime(jean,marie)
  - "chaque perruche est un oiseau" $\forall x.$ [perruche(x) $\rightarrow$ oiseau(x)]
  - "chaque ordinateur moderne a un microprocesseur" $\forall x.$ [ordinateur(x) $\wedge$ moderne(x)) $\rightarrow$ $\exists y.$ [microprocesseur(y) $\wedge$ partie_de(x,y)]]

Logique des prédicats Termes

- un constante a, b, c (on écrira souvent "jean", "marie" etc. pour rendre les formules plus lisibles) est un terme,
- un variable x, y, z est un terme
- si f est un fonction qui prend n arguments et t_1, ..., t_n sont des termes, alors f(t_1, ..., t_n) est un terme (on écrira souvent "racine_carre_de" et "pere_de"),
- rien d'autre n'est un terme.

Logique des prédicats Termes

- L'intuition est que les termes vont représenter les individus dans notre domaine: des personnes, du matériel, etc.
- Il y en a qui ont des noms (les constantes: a, "jean", "marie") et des autres qui sont anonymes (les variables x, y, z, dont la valeur dépend du contexte)
- Finalement, on peut avoir des fonctions, qui donnent un entité qui dépend de leurs arguments, comme "microprocesseur_de(x)".

Logique des prédicats Formules Atomiques

- si p est un prédicat qui prend n arguments et t_1, ..., t_n sont des termes, alors p(t_1, ..., t_n) est un formule atomique,
- si p est un prédicat qui ne prend aucun argument, on l'appelle une proposition,
- si t_1 et t_2 sont de termes, alors $t_1 = t_2$ et $t_1 \neq t_2$ sont des formules atomiques.
- rien d'autre n'est une formule atomique

Logique des prédicats Formules Atomiques

- la différence entre un prédicat et une fonction est qu'un fonction donne une entité comme résultat et un prédicat "vrai" ou "faux". Alors "aime(jean,marie)" est une formule qui est vrai si est seulement si Jean aime Marie dans notre modèle.
- $t_1 = t_2$ est bien sûr l'identité entre deux termes et est vrai si les deux termes dénotent le même entité. $t_1 \neq t_2$ dénote l'inégalité et est vrai ssi $t_1 = t_2$ est faux

Logique des prédicats Formules

1. Une formule atomique est une formule,
2. Si X est une formule (pas nécessairement atomique) alors $\neg X$ — "non X" ou "la négation de X" — est une formule,

Logique des prédicats Formules

3. Si X et Y sont des formules alors,

   3.a. $(X \wedge Y)$ "X et Y"

   3.b. $(X \vee Y)$ "X ou Y"

   3.c. $(X \rightarrow Y)$ "si X alors Y"

   3.d. $(X \leftrightarrow Y)$ "X si et seulement si Y"

   sont de formules

Logique des prédicats Formules

4. Si x est un variable et F est une formule alors $\forall x.[F]$ et $\exists x.[F]$ sont des formules
5. Rien d'autre n'est une formule.

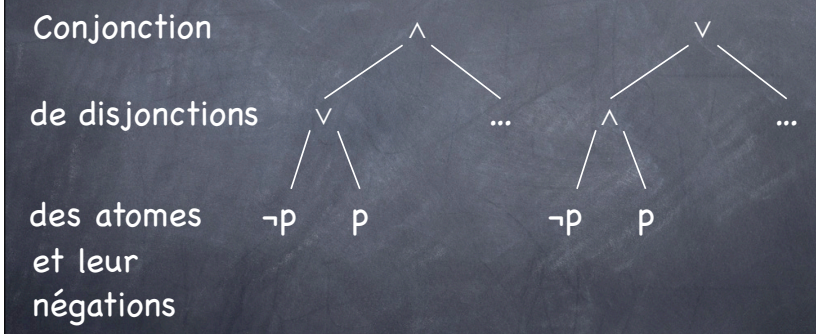
Logique des prédicats
Formules

- La formule ∀x.[F] est vrai si et seulement si pour chaque entité de notre domaine de discours F est vrai.
- La formule ∃x.[F] est vrai si et seulement s'il existe un entité dans notre domaine pour lequel F est vrai.

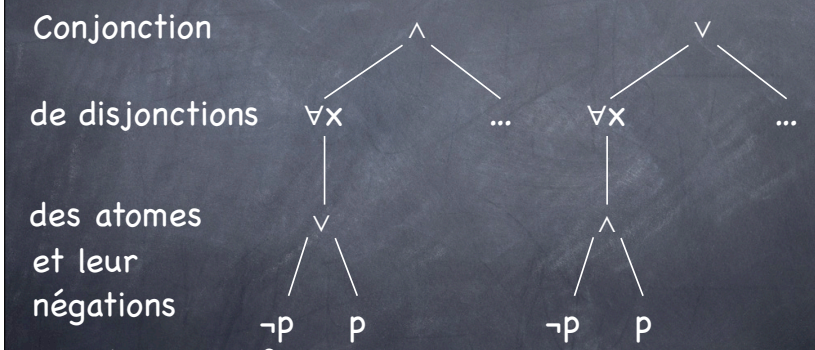
Logique des prédicats
Déduction

- Alors, comment décide-t-on si une formule en calcul des prédicats est un tautologie?
- Sous certains conditions on peut adapter la résolution au calcul des prédicats.
- Mais il y a des autres méthodes de calcul aussi (comme le calcul des séquents dans les notes de cours).

Forme Normale
Conjonctive/Disjonctive



Forme Normale
Conjonctive/Disjonctive



que des quantificateurs universels,
avec portée sur les disjonctions

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

aime(jean,marie)

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

aime(jean,marie)

X = marie

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

aime(marie,marie)

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

aime(marie,marie)

X = marie

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

⊥

Réponse: oui

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

aime(romeo,juliet)

X = romeo

X = juliet

Echec d'unification: X ne peut pas être à la fois "romeo" et "juliet"

Alors, réponse: non

Résolution
Clauses de Horn

Remarque important:
"non" veux simplement dire "je ne sais pas le démontrer" ou "la question n'est pas un conséquence logique du programme". Alors il faut toujours faire attention à une réponse "non" qui peut aussi être un conséquence d'un simple manque de connaissances.

Alors, réponse: non

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

aime(jean,X)

Pour quel X est-il vrai que Jean aime X?

Résolution
Clauses de Horn

aime(jean,Y) ←
    aime(marie,Y).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

aime(jean,X)

X = Y

Pour quel X est-il vrai que Jean aime X?

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(Y,Y).
aime(robert,vin).

aime(marie,X)

Y = X

Y = marie

Pour quel X est-il vrai que Jean aime X?

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(Y,Y).
aime(robert,vin).

⊥

Y = X

Y = marie

X = marie

Pour quel X est-il vrai que Jean aime X?

Réponse: X = marie

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,Y) ←
    aime(Y,vin).
aime(X,X).
aime(robert,vin).

aime(jean,X)

Y = X

Pour quel X est-il vrai que Jean aime X?

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(Y,Y).
aime(robert,vin).

aime(X,vin)

Y = X
Y = vin
X = vin

Pour quel X est-il vrai que Jean aime X?

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(Y,Y).
aime(robert,vin).

⊥

Y = X
Y = vin
X = vin

Pour quel X est-il vrai que Jean aime X?

Réponse: X = vin

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

aime(X,vin)

X = robert

Pour quel X est-il vrai que Jean aime X?

Résolution
Clauses de Horn

aime(jean,X) ←
    aime(marie,X).
aime(jean,X) ←
    aime(X,vin).
aime(X,X).
aime(robert,vin).

⊥

X = robert

Pour quel X est-il vrai que Jean aime X?

Réponse: X = robert

Exemple avec des termes

- 0 dénote le nombre 0,
- si X dénote le nombre N, s(X) dénote le nombre N + 1,

0	0
1	s(0)
2	s(s(0))
3	s(s(s(0)))

Bien sûr que ceci n'est pas un façon très efficace pour coder les entiers !

Résolution Clauses de Horn

```
addition(0,X,X).
addition(s(X),Y,s(Z)) ←
    addition(X,Y,Z).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

Résolution Clauses de Horn

```
addition(0,X,X).
addition(s(X),Y,s(Z)) ←
    addition(X,Y,Z).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(s(s(0),s(s(0)),Z).
```

Résolution Clauses de Horn

```
addition(0,X,X).
addition(s(X),Y,s(V)) ←
    addition(X,Y,V).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(s(s(0),s(s(0)),Z).
```

X = s(0)
Y = s(s(0))
Z = s(V)

Z = s(V)

Résolution
Clauses de Horn

```
addition(0,X,X).
addition(s(X),Y,s(W)) ←
    addition(X,Y,W).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(s(0),s(s(0)),V).
```

X = 0
Y = s(s(0))
V = s(W)

Z = s(V)

Résolution
Clauses de Horn

```
addition(0,X,X).
addition(s(X),Y,s(W)) ←
    addition(X,Y,W).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(s(0),s(s(0)),V).
```

X = 0
Y = s(s(0))
V = s(W)

Z = s(s(W))

Résolution
Clauses de Horn

```
addition(0,X,X).
addition(s(X),Y,s(W)) ←
    addition(X,Y,W).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(0,s(s(0)),W).
```

W = X
X = s(s(0))

Z = s(s(W))

Résolution
Clauses de Horn

```
addition(0,X,X).
addition(s(X),Y,s(W)) ←
    addition(X,Y,W).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(0,s(s(0)),W).
```

W = X
X = s(s(0))

Z = s(s(s(s(0))))

Résolution
Clauses de Horn

```
addition(0,X,X).
addition(s(X),Y,s(Z)) ←
    addition(X,Y,Z).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(X,s(0),s(s(0))).
```

Un avantage de la formulation des propriétés en logique comme ça est qu'il n'y a pas vraiment d'entrée et de sortie et qu'on peut demander des questions comme on veut.

Résolution
Clauses de Horn

```
addition(0,X,X).
addition(s(V),Y,s(Z)) ←
    addition(V,Y,Z).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(X,s(0),s(s(0))).
```

X = s(V)
Y = s(0)
Z = s(0)

Résolution
Clauses de Horn

```
addition(0,X,X).
addition(s(V),Y,s(Z)) ←
    addition(V,Y,Z).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(V,s(0),s(0)).
```

X = s(V)
Y = s(0)
Z = s(0)

X = s(V)

Résolution
Clauses de Horn

```
addition(0,W,W).
addition(s(X),Y,s(Z)) ←
    addition(X,Y,Z).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(V,s(0),s(0)).
```

V = 0
W = s(0)

X = s(V)

Résolution
Clauses de Horn

```
addition(0,W,W).
addition(s(X),Y,s(Z)) ←
    addition(X,Y,Z).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(V,s(0),s(0)).
```

V = 0

W = s(0)

X = s(0)

Résolution
Clauses de Horn

```
addition(0,Z,Z).
addition(s(V),Y,s(Z)) ←
    addition(V,Y,Z).

multiplication(0,X,0).
multiplication(s(X),Y,Z) ←
    multiplication(X,Y,V),
    addition(V,Y,Z).
```

```
addition(X,Y,s(s(0))).
```

X = s(V)

Y = Z

Z = s(s(0))

Résolution
Clauses de Horn

```
travail(usine(A,B,C,D)) ←
    competent(A, m2),
    competent(B, m13),
    competent(C, m17),
    competent(D, m18).

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).
```

```
travail(U).
```

Interprétation:
dans notre petit usine on a 4 machines (m2, m13, m17 et m18) et 4 employées (em19, em27, em36 et em40). Malheureusement, pas tous nos employées peuvent opérer chaque des nos machines.

Résolution
Clauses de Horn

```
travail(usine(A,B,C,D)) ←
    competent(A, m2),
    competent(B, m13),
    competent(C, m17),
    competent(D, m18).

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).
```

```
travail(U).
```

Interprétation:
notre plus ancien employée em19 sait opérer tous le machines. em27 et em36 savent opérer un seul machine et grâce à un cours de soir qu'il vient de finir, em40 sait opérer 2 machines

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
    competent(A, m2),
    competent(B, m13),
    competent(C, m17),
    competent(D, m18).

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

travail(U).

Interprétation:
L'usine est représenté par usine(A,B,C,D) ou A est l'employée qui travaille au machine m2 etc.
On cherche à trouver, pour chaque employée un machine qui lui convient.

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
    competent(A, m2),
    competent(B, m13),
    competent(C, m17),
    competent(D, m18).

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

travail(U).

U = usine(A,B,C,D)

U = usine(A,B,C,D)

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
    competent(A, m2),
    competent(B, m13),
    competent(C, m17),
    competent(D, m18).

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(A,m2),c(B,m13),c(C,m17),c(D,m18)

U = usine(A,B,C,D)

U = usine(A,B,C,D)

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
    competent(A, m2),
    competent(B, m13),
    competent(C, m17),
    competent(D, m18).

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(A,m2),c(B,m13),c(C,m17),c(D,m18)

A = em19

X = m2

U = usine(A,B,C,D)

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),
competent(C, m17),
competent(D, m18).

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(B,m13),c(C,m17),c(D,m18)

A = em19
X = m2

U = usine(em19,B,C,D)

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),
competent(C, m17),
competent(D, m18).

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(B,m13),c(C,m17),c(D,m18)

B = em19
X = m13

U = usine(em19,em19,C,D)

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),
competent(C, m17),
competent(D, m18).

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(B,m13),c(C,m17),c(D,m18)

On semble d'avoir oublié quelque chose !

B = em19
X = m13

U = usine(em19,em19,C,D)

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(B,m13), em19≠B, c(C,m17), ...

A = em19

B = em19

U = usine(
em19,
em19,
C,
D)

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

em19≠em19, c(C,m17), ...

A = em19

B = em19

U = usine(
em19,
em19,
C,
D)

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(B,m13),em19≠B, c(C,m17), ...

Echec !

On cherche une autre affectation pour m13.

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(A,m2),c(B,m13),em19≠B, c(C,m17),

Echec !

On cherche une autre affectation pour m2.

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(A,m2),c(B,m13),A≠B, c(C,m17), ...

On cherche une autre affectation pour m2.

A = em27

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(A,m2),c(B,m13),A≠B, c(C,m17), ...

On cherche une autre affectation pour m2.

A = em27

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(B,m13),em27≠B, c(C,m17), B≠C,...

On cherche une autre affectation pour m2.

A = em27

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(B,m13),em27≠B, c(C,m17), B≠C,...

On cherche une autre affectation pour m2.

A = em27

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

em27≠em19, c(C,m17), em19≠C,...

On cherche une autre affectation pour m2.

X = m13

B = em19

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(C,m17), em19≠C,em27≠C, ...

On cherche une autre affectation pour m2.

A = em27

B = em19

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

em19≠em36,em27≠em36, c(D,m18)...

On cherche une autre affectation pour m2.

A = em27

B = em19

C = em36

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

em27≠em36, c(D,m18)...

On cherche une autre affectation pour m2.

A = em27

B = em19

C = em36

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(D,m18), D≠C,D≠B,D≠A.

On cherche une autre affectation pour m2.

A = em27

B = em19

C = em36

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

c(D,m18), D≠em36,D≠em19,D≠em27.

On cherche une autre affectation pour m2.

A = em27
B = em19
C = em36
D = em40

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

em40≠em36,em40≠em19,em40≠em2

On cherche une autre affectation pour m2.

A = em27
B = em19
C = em36
D = em40

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

em40≠em19,em40≠em27.

On cherche une autre affectation pour m2.

A = em27
B = em19
C = em36
D = em40

Résolution
Clauses de Horn

travail(usine(A,B,C,D)) ←
competent(A, m2),
competent(B, m13),A≠B,
competent(C, m17),B≠C,A≠C,
competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).

em40≠em27.

On cherche une autre affectation pour m2.

A = em27
B = em19
C = em36
D = em40

Résolution
Clauses de Horn

```
travail(usine(A,B,C,D)) ←
    competent(A, m2),
    competent(B, m13),A≠B,
    competent(C, m17),B≠C,A≠C,
    competent(D,m18),D≠C,D≠B,D≠A.

competent(em19, X).
competent(em27, m2).
competent(em36, m17).
competent(em40, m17).
competent(em40, m18).
⊥
```

On cherche une autre affectation pour m2.

A = em27
B = em19
C = em36
D = em40

Prolog et Résolution

- Résolution comme fait par Prolog est très simple.
- Alors, si votre programme marche intelligemment, c'est uniquement grâce a vous.
- En programmation avec des contraints, on peut utiliser le contraint "all_different" pour dire que toutes les variables dans un ensemble ont des valeurs différents.

Lois De Morgan

- $\neg\forall x.F \equiv \exists x.\neg F$
- $\neg\exists x.F \equiv \forall x.\neg F$

Forme Prénex

- $\forall x.[\ F\ ] \wedge G \equiv \forall x.[\ F \wedge G]$ Pas d'occurrences de x en G
- $\forall x.[\ F\ ] \vee G \equiv \forall x.[\ F \vee G]$ Pas d'occurrences de x en G
- $\forall x.[\ F\ ] \rightarrow G \equiv \exists x.[\ F \rightarrow G]$ Pas d'occurrences de x en G
- $\forall x.G \rightarrow [\ F\ ] \equiv \forall x.[\ G \rightarrow F]$ Pas d'occurrences de x en G

Forme Prénex

- $\exists x.[\ F\ ] \wedge G \equiv \exists x.[\ F \wedge G]$ Pas d'occurrences de x en G
- $\exists x.[\ F\ ] \vee G \equiv \exists x.[\ F \vee G]$ Pas d'occurrences de x en G
- $\exists x.[\ F\ ] \rightarrow G \equiv \forall x.[\ F \rightarrow G]$ Pas d'occurrences de x en G
- $\exists x.G \rightarrow [\ F\ ] \equiv \exists x.[\ G \rightarrow F]$ Pas d'occurrences de x en G