Approche logique de l'Intelligence Artificielle-Partie 2

INSA 4IF/4IFA
Sylvie Calabretto

Plan du cours

- *Logique Classique*
  - Systèmes Formels
  - Calcul des Propositions/ Logique ordre 0
  - Calcul des Prédicats/ Logique ordre 1
  - Calcul des Prédicats/ Logiques ordres supérieurs
- *Logiques Multivaluées*
  - Logique Floue

Introduction

- Le raisonnement logique suivant :

  Tous les hommes sont mortels

  Socrate est un homme

  Donc Socrate est mortel

- N’est pas modélisable en logique des propositions

Syntaxe de la logique des Prédicats

- **Alphabet** composé
  - de constantes (0, 1, …, a, b, …)
  - de variables (x, y, z, …)
  - de symboles de fonctions (f, g, …)
  - de symboles de relations ou prédicats (p, q, …)
  - de connecteurs ($\neg, \vee, \wedge, \rightarrow, \leftrightarrow$)
  - de quantificateurs ($\forall$, $\exists$)

Règles de construction des formules

- Les **termes** sont définis par :
  - les constantes et les variables sont des termes
  - si $t_1, t_2, \ldots, t_n$ sont des termes et f une fonction d'arité n, alors $f(t_1, t_2, \ldots, t_n)$ est un terme
- Les **formules atomiques** sont définies par :
  - si $t_1, t_2, \ldots, t_n$ sont des termes et R une relation d'arité n, alors $R(t_1, t_2, \ldots, t_n)$ est une formule atomique
- Les **formules** sont définies par :
  - les formules atomiques sont des formules
  - si ϕ est une formule alors $\neg \phi$ est une formule
  - si ϕ est une formule et x une variable alors $\forall x (\phi)$ est une formule ainsi que $\exists x (\phi)$
  - si ϕ et φ sont des formules alors $\phi \vee \varphi$, $\phi \wedge \varphi$, $\phi \rightarrow \varphi$, $\phi \leftrightarrow \varphi$ sont des formules

Exemples de formules

- Formules atomiques
  - plus_petit(x, y) ou x< y
  - père(x, y) ou x pèreDe y
  - dans(x, E) ou x∈E

- Formules composées
  - ∀x, ∀z, (x oncle_maternel_de z) ↔ ∃y, (x frère_de y) **et** (y mère_de z)

Variables libres/liées

- Une occurrence d'une variable x est **liée** dans une formule ϕ si et seulement si il existe une sous-chaîne de ϕ contenant x qui est une sous-formule de ϕ qui débute par $\forall x$ ou $\exists x$ (elle est **libre** sinon).

- Une variable x est une **variable libre** (resp. **liée**) d'une formule ϕ si et seulement si x a une occurrence libre (resp. liée) dans ϕ.

- Exemples :
  - Dans : $\forall x \exists y P(x,y)$, x et y sont liées
  - Dans P(x), x est libre

Substitution d'une variable par un terme

- Soit x une variable et t un terme, la **substitution** de t à x, ϕ [t / x] est la formule obtenue en remplaçant toutes les occurrences libres de x dans ϕ par t.

- Exemple :
  - $\phi = P(x) \vee Q(x)$
  - ϕ [t / x] = $P(t) \vee Q(t)$

Règles d'inférence

- Toutes les règles d'inférence du calcul des propositions (dont Modus Ponens)
- Généralisation Existentielle (GE) :
  $E(t) \Rightarrow \exists x\ E(x)$ où t est un terme clos
- Généralisation Universelle (GU) :
  $E \Rightarrow \forall x\ E(x)$ où x n'est pas quantifié dans E.
- Spécification Existentielle (SE) :
  $\exists x\ E(x) \Rightarrow E(k)$ où k est une constante nouvelle
- Spécification Universelle (SU) :
  $\forall x\ E(x) \Rightarrow E(t)$ où t est un terme quelconque
- Dualité : $\forall x\ E(x) \Leftrightarrow \neg\ \exists x\ \neg E(x)$

Exercice

Trouver la conclusion des 7 assertions suivantes :

1/ Les animaux sont toujours mortellement offensés si je ne fais pas attention à eux.

2/ Les seuls animaux qui m'appartiennent se trouvent dans ce pré.

3/ Aucun animal ne peut résoudre une devinette s'il n'a pas reçu une formation convenable dans une école.

4/ Aucun des animaux qui se trouvent dans ce pré n'est un raton laveur.

5/ Quand un animal est mortellement offensé, il se met toujours à courir en tout sens et à hurler.

6/ Je ne fais pas attention à un animal qui ne m'appartient pas.

7/ Aucun animal qui a reçu dans une école une formation convenable ne se met jamais à courir en tout sens et à hurler.

Exercice : corrigé

- 1/ $\forall x, \neg \text{Attention}(x) \rightarrow \text{MortellementOffensé}(x)$
- 2/ $\forall x, \neg \text{Pré}(x) \rightarrow \neg \text{Appartient}(x)$
- 3/ $\forall x, \neg \text{Formation}(x) \rightarrow \neg \text{RésoudDevinette}(x)$
- 4/ $\neg \text{Pré}(\text{raton laveur})$
- 5/ $\forall x, \text{MortellementOffensé}(x) \rightarrow \text{CourtEtHurle}(x)$
- 6/ $\forall x, \neg \text{Appartient}(x) \rightarrow \neg \text{Attention}(x)$
- 7/ $\forall x, \text{CourtEtHurle}(x) \rightarrow \neg \text{Formation}(x)$

- En utilisant le Modus Ponens et les règles 4, 2, 6, 1, 5, 7 et 3 respectivement, on en conclut que les ratons laveurs ne peuvent pas résoudre une devinette !

Interprétation de la Logique des Prédicats

- Une interprétation du langage L des prédicats du 1er ordre nécessite les notions suivantes :
  - un domaine d'interprétation D : un ensemble de valeurs que peuvent prendre les variables
  - une interprétation
    - des constantes
    - des fonctions
    - des prédicats/relations

- Exemple :
  - D(x,y) est vrai si et seulement si « y divise x »
  - P(x,y) est le produit de x par y

Valuation

On appelle valuation :

- Pour une interprétation I donnée, on appelle valuation v des variables relatives à I, toute application de l'ensemble des variables dans D.
- Pour une valuation v donnée, l'interprétation d'une formule logique contenant des variables libres est obtenue en substituant aux variables libres leurs valeurs dans D.

Formules valides

- Une formule ϕ est **universellement valide** si et seulement si pour toute interprétation $\mathcal{D}$ et pour toute valuation ν, ϕ est vraie.
- Une formule ϕ est **valide** si et seulement si il existe une interprétation $\mathcal{D}$ telle que pour toute valuation ν, ϕ est vraie.
- Une formule ϕ est **satisfaisable** si et seulement si il existe une interprétation $\mathcal{D}$ et une valuation ν telles que ϕ est vraie.

Exemples

- $\phi = \forall x\ (\ p(x) \rightarrow p(f(x))) \rightarrow \forall y\ (\ p(y) \rightarrow p(f(y)))$ est une formule universellement valide
- Soit $\phi = \forall x\ (\exists y\ D(x,y) \rightarrow \exists z\ E(x,p(z,w))$ où :
  - D(x,y) est vrai si et seulement si « y divise x »
  - E(x,y) est vrai si et seulement si « x et y sont égaux »
  - P(x,y) est le produit de x par y
- ϕ est satisfaisable
- Preuve :
  - Soit $\mathcal{D}$ = { 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 14 }
  - Seule variable libre = w, pour une valuation assignant la valeur 2 à w, la formule est vraie
- Conclusion : Tout nombre x admettant un diviseur dans $\mathcal{D}$ est un nombre pair

Principe de résolution et Prolog

- Plus complexe que pour calcul des propositions car prise en compte de variables

- Sert de base au langage Prolog mais ce langage se limite à un type particulier de formules

- Applications de Prolog : Spécification de Logiciels, Systèmes Experts, Systèmes à Base de Connaissances, Interfaces en langage naturel et Représentation de Connaissances, Développement d'Intelligences Artificielles pour des jeux, …

Principe de résolution en Logique des Prédicats

- Pour pouvoir appliquer la méthode de résolution, les formules sont mises sous forme clausale.

- La résolution consiste à apparier des clauses satisfaisant à certaines conditions pour produire de nouvelles clauses jusqu'à obtenir la clause vide.

Littéraux et clauses

- Un **littéral** est une formule atomique (littéral de signe positif) ou une formule atomique précédée du symbole de négation (littéral de signe négatif).

- Une **clause** est une disjonction de littéraux.

Méthode de résolution (M. Robinson, 1965)

- Avec 2 clauses A(x) et ¬A(t), s'il existe une substitution s les alignant, on a la **règle de résolution de Robinson** :

  $A(x) \vee B(x,y), \neg A(t) \vee C(t,u) \Rightarrow B(st,sy) \vee C(st,su)$

  $sx = st$

  Exemple :

  ¬P(x) ∨ Q(x)
  et P(a)
  où a est une constante, donnent la résolvante :
  Q(a)
  Avec la substitution x = a

Règle de la résolution

- L'équivalence $\neg X \vee \neg Y \vee Z \leftrightarrow ((X \wedge Y) \rightarrow Z)$ suggère que cette règle est une sorte de *syllogisme*.
- Les règles *modus ponens* et *modus tollens* en sont des cas particuliers.
- Cette règle d'inférence est nécessaire et suffisante donc bien adaptée aux moteurs d'inférence
- La méthode de résolution exploite cette règle au cœur d'une démonstration par l'absurde (Herbrand/Robinson)

Application de la résolution à la démonstration d'une formule

- Soit Γ un ensemble de clauses, et C une clause,

  C est un théorème de Γ si et seulement si la clause vide se déduit de l'ensemble constitué de Γ et de ¬C.

- On dit aussi que C est une conséquence logique de Γ.

Exemple

- Montrer que l 'ensemble de clauses

  $\{C_1 = S(a,y) \vee T(x,y),\ C_2 = \neg T(z,t) \vee U(a,z),$

  $C_3 = \neg S(a,b),\ C_4 = \neg U(u,v)\}$ est contradictoire.

  Il faut donc montrer que de $\{C_1, C_2, C_3, C_4\}$, on déduit la clause vide.

- Preuve :
  - $C_1, C_3 \Rightarrow T(x,b) = C_5$ ([y/b])
  - $C_2, C_4 \Rightarrow \neg T(z,t) = C_6$ ([u/a], [v/z])
  - $C_5, C_6 \Rightarrow \square$ ([z/x], [t/b])

Mise en œuvre de la résolution

- Mettre la formule sous forme clausale.

- Tester des substitution de termes pour appliquer la règle de résolution jusqu'à obtenir une contradiction par conjonction d'un littéral et de sa négation (la clause vide).

Mise sous forme de clauses

- Elimination des connecteurs $\leftrightarrow$, $\rightarrow$ en utilisant les équivalences : $(\phi \leftrightarrow \varphi) \equiv (\phi \rightarrow \varphi) \wedge (\varphi \rightarrow \phi)$ et
  $(\phi \rightarrow \varphi) \equiv (\neg\phi \text{ v } \varphi)$
- Transfert des négations devant les atomes en utilisant les équivalences :$\neg\neg\phi \equiv \phi$, $\neg(\exists x\ \phi) \equiv \forall x\ (\neg\phi)$ et $\neg(\forall x\ \phi) \equiv \exists x\ (\neg\phi)$
- Standardisation des variables
- Mise sous forme prénexe (tous les quantificateurs à gauche)
- Skolémisation (élimination des quantificateurs existentiels)
- Distributivité pour obtenir une forme conjonctive
- Passage à la forme clausale : Suppression des quantificateurs universels + 1 clause par disjonction

Skolémisation

- On remplace toutes les variables gouvernées par un quantificateur existentiel :
  - par une constante si elles ne sont pas dans la portée d'un quantificateur universel
  - par une nouvelle fonction $f(x_1, x_2, \ldots, x_n)$ si elles sont dans la portée de n quantificateurs universels portant sur les variables $x_1, x_2, \ldots, x_n$

- Exemple :
  - $\exists y\ P(y)$ ➡ $P(a)$ où a est une constante de Skolem
  - $\forall x\ \exists y\ P(x,y)$ ➡ $\forall x\ P(x,f(x))$ où f est une fonction de Skolem

Exercice 1

- Démontrer, en utilisant la méthode de résolution de Robinson, qu'on peut ne pas être un homme et être intelligent à partir des énoncés suivants :

  - Tout homme est un primate.
  - Les dauphins ne sont pas des primates.
  - Il y a des dauphins qui sont intelligents.

Exercice 1 : solution

- On pose les prédicats suivants :
  - H(x) : x est un homme
  - P(x) : x est un primate
  - D(x) : x est un dauphin
  - I(x) : x est intelligent

- Traduction des formules :
  - $\forall x, H(x) \rightarrow P(x)$
  - $\forall x, D(x) \rightarrow \neg P(x)$
  - $\exists x, D(x) \wedge I(x)$
  - Conclusion : $\exists x, \neg H(x) \wedge I(x)$
  - Négation de la conclusion : $\forall x, H(x) \vee \neg I(x)$

Exercice 1 : solution

- Mise sous forme de clauses :
  - 1/ $\neg H(x) \vee P(x)$
  - 2/ $\neg D(x) \vee \neg P(x)$
  - 3/ $D(a)$
  - 4/ $I(a)$
  - 5/ $H(x) \vee \neg I(x)$

- Résolution :
  - Rés(1,2) : $\neg H(x) \vee \neg D(x)$: 6
  - Rés(6, 5) : $\neg D(x) \vee \neg I(x)$: 7
  - Rés(7, 3) avec (x/a) : $\neg I(a)$: 8
  - Rés(8, 4) : □

Exercice 2

Etablir que :

C : Il y a des gens malhonnêtes non arrêtés

à partir des axiomes suivants :

A1 : Pour tout crime, il y a quelqu'un qui l'a commis,

A2 : Seuls les gens malhonnêtes commettent des crimes,

A3 : Ne sont arrêtés que les gens malhonnêtes,

A4 : Les gens malhonnêtes arrêtés ne commettent pas de crimes,

A5 : Il y a des crimes.

Le langage PROLOG (A. Colmerauer, 1970)

- Le langage PROLOG est un démonstrateur automatique de théorème reposant sur une application systématique de la résolution de Robinson

- Un programme PROLOG est composé de clauses de Horn = une clause ayant au plus un littéral positif telle que :

  $\neg\phi_1 \vee \neg\phi_2 \vee \ldots \vee \neg\phi_n \vee\varphi$

- Cette clause est logiquement équivalente à : $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n \rightarrow \varphi$

PROLOG et démonstration automatique

- But de l'exécution d'un programme Prolog = démontrer à l'aide de la résolution de Robinson qu'une conjonction de littéraux (le but) est le conséquent d'un ensemble de clauses de Horn C.

- Le but se déduit de C pour certaines substitutions des variables. Toutes ces substitutions sont reportées par Prolog en cas de succès pour en informer l'utilisateur.

Prolog : forme (standard Edimbourg)

- **Règles**
  1 : grandMère(X,Z) :- mère(X,Y),pèreOuMère(Y,Z).
  2 : pèreOuMère(X,Y) :- mère(X,Y).
  3 : pèreOuMère(X,Y) :- père(X,Y).

- **Faits**
  4 : mère(marie,pierre)
  5 : mère(marie,jacques)
  6 : mère(marie,jean)
  7 : père(jean,sophie)
  8 : père(pierre,albert)

- **Requête**
  grandMere(x, y)

A :- B,C,D. se lit **A si B et C et D** pour $(B \wedge C \wedge D) \rightarrow A$

Résolution

La requête codée *suppose le faux pour savoir le vrai.*

9: ¬grandMere(x, y)

inférence par la règle de résolution utilisée entre clauses spécifiques (dont les faits),et clauses générales

10: ¬mere(x1, z1)∨ ¬pereOuMere(z1, y1) <x1=x, y1=y> 9,1

11: ¬pereOuMere(z1, y1) <marie=x1=x, pierre=z1, y1=y> 10,4

12: ¬mere(x2, y2) <x2=pierre=z1, y2=y1=y, marie=x1=x > 11,2

improductif : pierre n'est la mère de personne

13: ¬pere(x2, y2) <x2=pierre=z1, y2=y1=y, marie=x1=x > 11,3

14: ⊥ <x2=pierre=z1, albert=y2=y1=y, marie=x1=x> 13,8

⇒ *grandMere(x,y) établi pour x=marie, y=albert*

15: … (on obtiendra de même x=marie, y=sophie)

Exemple application PROLOG (Morpion)

% The game state will be represented by a list of 9 elements

% board(_,_,_,_,_,_,_,_,_) at the beginning

% eg board(_,_,'x',_,_,_,_,_,_) after the first round

% eg board(_,_,'x',_,_,_,'o',_,_) after the second round

% ...

% until someone wins or the board is fully instanciated

Un extrait du programme

Exemple application PROLOG (Morpion)

```
%%%% Test is the game is finished %%%
gameover(Winner) :- board(Board), winner(Board,Winner), !.  % There exists a winning configuration: We cut!


%%%% Test if a Board is a winning configuration for the player P.
winner(Board, P) :- Board = [P,Q,R,_,_,_,_,_,_], P==Q, Q==R, nonvar(P). % first row
winner(Board, P) :- Board = [_,_,_,P,Q,R,_,_,_], P==Q, Q==R, nonvar(P). % second row
winner(Board, P) :- Board = [_,_,_,_,_,_,P,Q,R], P==Q, Q==R, nonvar(P). % third row
winner(Board, P) :- Board = [P,_,_,Q,_,_,R,_,_], P==Q, Q==R, nonvar(P). % first column
winner(Board, P) :- Board = [_,P,_,_,Q,_,_,R,_], P==Q, Q==R, nonvar(P). % second column
winner(Board, P) :- Board = [_,_,P,_,_,Q,_,_,R], P==Q, Q==R, nonvar(P). % third column
winner(Board, P) :- Board = [P,_,_,_,Q,_,_,_,R], P==Q, Q==R, nonvar(P). % first diagonal
winner(Board, P) :- Board = [_,_,P,_,Q,_,R,_,_], P==Q, Q==R, nonvar(P). % second diagonal
```

Un extrait du programme

Exemple application PROLOG (Morpion)

```
%%%% Recursive predicate that checks if all the elements of the List (a board)
%%%% are instanciated: true e.g. for [x,x,o,o,x,o,x,x,o] false for [x,x,o,o,_G125,o,x,x,o]
isBoardFull([]).
isBoardFull([H|T]):- nonvar(H), isBoardFull(T).


%%%% Artificial intelligence: choose in a Board the index to play for Player (_)
%%%% This AI plays randomly and does not care who is playing: it chooses a free position
%%%% in the Board (an element which is an free variable).
ia(Board, Index,_) :- repeat, Index is random(9), nth0(Index, Board, Elem), var(Elem), !.
```

Un extrait du programme

Logiques typées

Très souvent, dans une théorie formelle, les :

$$\forall x\ \forall y\ \forall z\ ((p(x) \wedge q(y) \wedge r(x,y,z)) \rightarrow s(y,z))$$

comportent des *conditions de nature* telles que p(x) ou q(y), qu'on pourrait noter $\forall x \in \mathbf{P}\ \forall y \in \mathbf{Q}\ \forall z\ ((r(x,y,z)) \rightarrow s(y,z))$

On considère alors que D, domaine (implicite) de la théorie formelle, se fragmente en divers domaines $D_1, D_2, \ldots D_n$, support des *sortes* (ou *types formels*) $s_1, s_2, \ldots s_n$

- toute constante *c* est attachée à une sorte *s*
- une variable *v* est attachée à une sorte *s* par une notation telle que *v:s*
- toute fonction f d'arité k a maintenant un profil ou une signature $s_1 * s_2 * .. * s_k \rightarrow s_f$
- tout prédicat p d'arité k a maintenant un profil ou une signature $s_1 * s_2 * .. * s_k \rightarrow V$

Logiques typées

- La construction des termes et des énoncés doit alors respecter sortes et profils

- On procède de même à des remplacements uniformes cohérents lors des passages aux interprétations et modèle(s)

- Une hiérarchie des sortes pourra rendre compte formellement de compatibilités génériques : si s1<s2 signifie que *s1 est sous-sorte de s2*, tout élément de s1 sera accepté en lieu et place d'un élément de s2 ; la contrainte correspondante devra être satisfaite par les différents modèles

Applications de la logique classique

- La Logique Classique justifie la conception de Prolog et de nombreux moteurs d'inférence, et par là sous-tend
  - Pour l'utilisateur final, de nombreux logiciels de conseil et de diagnostic (Systèmes Experts/Systèmes à Base de Connaissances).
  - Pour le développeur
    - de nombreuses *maquettes et prototypes,* où l'on travaille près des spécifications (le QUOI) plutôt qu'avec beaucoup de moyens algorithmiques (le COMMENT) :

      Bases de données déductives, interface d'interrogation en langue naturelle, traducteurs, ...
    - la possibilité de *modules-experts*

      Couplage PROLOG/BD

Plan du cours

- *Logique Classique*
  - Systèmes Formels
  - Calcul des Propositions/ Logique ordre 0
  - Calcul des Prédicats/ Logique ordre 1
  - Calcul des Prédicats/ Logiques ordres supérieurs
- *Logiques Multivaluées*
  - Logique Floue

Logiques d'ordres supérieurs

- Ces logiques se manifestent par des énoncés utilisant des propriétés, relations ou prédicats en mêlant leurs propriétés factuelles et leurs propriétés génériques.

  - **Exemple 1 :** une relation est dite transitive si, pour tout triplet d'objets x, y, z, quand elle est satisfaite par (x,y) et par (y,z), alors elle l'est par (x,z).

    ou simple énoncé logique d'ordre 2 ?

    $\forall R \ (\text{transitive}(R) \rightarrow \ \forall x \forall z \ (\exists y (R(x,y) \wedge R(y,z)) \rightarrow R(x,z)) \ )$

  - **Exemple 2 :** principe de récurrence : *approche méta-logique* :
    Règle d'inférence supplémentaire pour l'ordre 1

    $P(0), \ \forall n \ (P(n) \rightarrow P(n+1) \ ) \Rightarrow \ \forall \ n \ P(n)$

    ou simple énoncé logique d'ordre 2 ?

    $\forall \ P \ \ (\ (P(0) \wedge (\ \forall n \ (\ P(n) \rightarrow P(n+1) \ ) \ ) \ ) \ \rightarrow \ \ \forall \ n \ P(n) \ )$

Echelle de Russel / 1905

- **Principe**

  Termes simples : ordre 0

  Tout prédicat se référant à un objet d'ordre n est d'ordre n+1

Exemple : relations et leur théorie

- **Ordre 1**
  - $\forall x \ \forall y \ (r(x,y) \rightarrow \ (app(x,X) \rightarrow app(y,X)))$
- **Ordre 2**
  - $\forall R \ \forall S \ \ (pf(R,S) \leftrightarrow (\forall x \ \forall y \ \{R(x,y) \rightarrow S(x,y)\}))$ *; déf. finesse*
  - $\forall R \ (\text{transitive } (R) \rightarrow \ \forall x \forall z \ (\exists y(R(x,y) \wedge R(y,z)) \rightarrow R(x,z)))$
  - $\forall R \ ((\text{transitive } (R) \wedge \text{réflexive}(R) \wedge \text{antisymétrique}(R)) \leftrightarrow \text{ordre}(R))$
- **Ordre 3**
  - **ordre(pf)** ; fait : la finesse entre relations est un ordre

Systèmes d'ordre 2

- PROLOG d'ordre 2
- PROLOG normal : les énoncés d 'ordre supérieur définis comme des macros-énoncés sont éliminés par un pré-traitement
- MOLOG (IRIT)
- Utilisation de la réification

Simulation de l'ordre 2 par réification (généalogie)

/* simulation **ordre supérieur** par **réification** des relations : en notant fait([p, x, y]) un fait p(x,y), la "relation" p est mise au niveau des *termes* x et y ; on peut alors définir ainsi la fermeture transitive P d'une relation Q : */

```
fait([P, X, Y]):- fermetureTransitive(P, Q), fait([Q, X, Y]).
fait([P, X, Y]):- fermetureTransitive(P, Q), fait([Q, X, Z]), fait([P, Z, Y]).
        /* liaison entre faits secondaires réifiés et faits primaires développés */
fait([pereOuMere, X, Y]):- pere(X, Y); mere(X, Y).
                /* ancêtre comme fermeture transitive de pereOuMere */
fermetureTransitive(ancetre, pereOuMere).
```

La requête fait([ancetre,X,Y]) donne 26 solutions

- X=jeanantonin, Y=louis
- X=jeanantonin, Y=charles
- X=jean, Y=marielouise
- X=jean, Y=monique
- X=jean, Y=edmond
- X=jean, Y=martine
- X=jean, Y=yves
- X=louis, Y=pierre
- X=louis, Y=thérèse2

pères/enfants

- X=mariethérèse, Y=louis
- X=mariethérèse, Y=charles
- X=thérèse1, Y=marielouise
- X=thérèse1, Y=monique

mères/enfants

- X=thérèse1, Y=edmond
- X=thérèse1, Y=martine
- X=thérèse1, Y=yves
- X=monique, Y=pierre
- X=monique, Y=thérèse2

mères/enfants

- X=jeanantonin, Y=pierre
- X=jeanantonin, Y=thérèse2
- X=jean, Y=pierre
- X=jean, Y=thérèse2

grand-pères/ petits-enfants

- X=mariethérèse, Y=pierre
- X=mariethérèse, Y=thérèse2
- X=thérèse1, Y=pierre
- X=thérèse1, Y=thérèse2

grand-mères/ petits-enfants

Ordre 0+

- Principe : Ajouter à l'ordre 0
  - Des variables globales
  - Des comparateurs

- Exemple :
  - Couleur du ciel = bleu
  - Pression des pneus < 2

- Exemple de règle d'ordre 0+ :
  - Si Pression des pneus < 2 Alors Regonfler les pneus

Représentation des connaissances en logique

- **Faits :**
  - Ordre 0 : Calcul des Propositions
  - Ordre 0+ : ex) Température cuve > 120
  - Ordre 1 : Calcul des Prédicats
  - Ordre 2 : ex) R est transitive

- **Règles :**
  - Ordre 0 : unexpectedBehaviour et dataCorruption → diskOverflow
  - Ordre 0+ : Température cuve > 120 → état = très chaud
  - Ordre 1 : x partie-de y et y partie-de z → x partie-de z
  - Ordre 2 : x R y et y R z et R est transitive → x R z