3.3. Sémantique déclarative *(VIDEO-PPT 2.3)*

Prolog a pour fondement théorique la logique du 1er ordre. Il en est proche mais il est plus restreint (notamment : perte de la négation, preuves limitées à des littéraux positifs).

La logique propositionnelle et la logique du 1er ordre ont été étudiées en INFO 1. Dans cette section, il s'agit de montrer le lien entre Prolog et la logique des prédicats (*hors récursivité*).

La sémantique déclarative de Prolog consiste à donner une signification logique aux clauses Prolog.

Forme clausale

Problème : une expression logique peut prendre différentes formes équivalentes :

- $A \Rightarrow B \equiv \neg A \vee B$
  $A \wedge B \Rightarrow C \equiv \neg A \vee \neg B \vee c$

☞ *La manipulation automatique de telles formules conduit à les représenter d'une seule manière appelée forme clausale, c'est une formule logique composée d'une disjonction de termes positifs et de termes négatifs. Les variables sont quantifiées universellement.*

$L_1 \vee L_2 \dots \vee \neg L_p \vee \neg L_{p+1} \vee \dots$

$\forall x$ homme(x) $\vee$ femme(x) $\vee$ ¬humain(x)

La clause vide, notée □, ne contient aucun littéral : elle est toujours fausse.

Clause de Horn

☞ *Une clause de Horn est une forme clausale avec au plus un littéral positif*

$L_1 \vee \neg L_2 \dots \vee \neg L_n$

- La clause de Horn : ∀ X,Y,Z grandPere(X,Y) ∨ ¬pere(X,Z) ∨ ¬pere(Z,Y)
  peut se réécrire : ∀X,Y,Z grandPere(X,Y) ⇐ pere(X,Z) ∧ pere(Z,Y)
  ou : ∀X,Y grandPere(X,Y) ⇐ ∃Z (père(X,Z) ∧ pere(Z,Y))

Formulation logique des règles

Une clause *Prolog* est une clause *de* Horn. Notation *Prolog* : L_1 :- L_2 , … , L_n.

- $\underbrace{\texttt{grandPere(X,Y)}}_{\text{conclusion}}$:- $\underbrace{\texttt{pere(X,Z) , pere(Z,Y)}}_{\text{hypothèses}}$.

conclusion ⇐ hypothèses

Si dans une règle *Prolog* une variable apparait avant ou (avant et après) l'opérateur :-, le quantificateur associé est ∀. Quand une variable n'apparait qu'après l'opérateur :- le quantificateur associé est ∃.

- ∀ `X,Y` si ∃`Z` tel que (`pere(X,Z)` et `pere(Z,Y)`) est vrai alors `grandPere(X,Y)`est vrai.

Un fait correspond à une clause de *Horn* sans littéral négatif mais avec seulement un littéral positif (clauses de Horn positive). En Prolog un fait correspond à une clause sans queue.

Résolution logique

En logique, pour prouver que la formule `f` est une conséquence logique des formules $\{f_1, f_2, \dots, f_n\}$, on démontre que l'ensemble $\{f_1, f_2, \dots, f_n, \neg f\}$ est inconsistant. Autrement dit, on démontre que le système $\{f_1, f_2, \dots, f_n, \neg f\}$ conduit à une contradiction (n'est pas satisfiable). C'est une démonstration par l'absurde ou **réfutation**, on va chercher à établir que système permet d'inférer$\{f, \neg f\}$qui est un sous-ensemble inconsistant (on ne peut avoir quelque chose et son contraire).

Pour montrer qu'un système d'équations logiques $\{f_1, f_2, \dots, f_n, \neg f\}$ est inconsistant, on utilise la règle de résolution :

$$\frac{P \vee Q \ , \ \neg P \vee R}{Q \vee R}$$

dans laquelle les formules P et R peuvent être absentes et donc dans le cas où elles le sont toutes les deux l'obtention de la réfutation :

$$\frac{P \ , \ \neg P}{\square}$$

Rappel de logique : Si, en introduisant la négation de `f` dans $\{f_1, f_2, \dots, f_n\}$, on peut déduire une contradiction $\{f_i, \neg f_i\}$(qui se réduit en la clause vide notée □), alors on peut conclure que `f` se déduit de$\{f_1, f_2, \dots, f_n\}$, `f` est donc vrai.

Ainsi, poser en Prolog une question « est-ce que `Q` est vrai ? » revient à montrer : `Q` soit ¬□ => `Q` (¬□ notant « vrai ») qui correspond à l'emploi de la méthode de résolution logique dans sa contraposée ¬`Q` => □ (on insère l'hypothèse ¬`Q` et on établit l'inconsistance □).

L'implication recherchée ¬□ => `Q` s'écrit en Prolog `Q :- ¬□`. En Prolog on omet l'écriture de `:- ¬□` , une question `Q` est une clause sans queue (puisque celle-ci est vraie).

- Application sur l'exemple déjà vu en sémantique opérationnelle :

```
1.  ¬mere(X,Z) ∨ ¬pere(Z,Y) ∨ grand_mere(X,Y).
2.  mere(rose, john)
3.  mere(rose, robert)
```

4. `pere(john, caroline).`

Pour montrer `grand_mere(rose, caroline)`, on introduit dans le système la négation

5. `¬grand_mere(rose, caroline).`

De `1` et `5` avec `{X = rose, Y = caroline}`, on déduit :

6. `¬mere(rose,Z) ∨ ¬pere(Z,caroline)`

De `6` et `4` avec `{Z = john}`, on déduit :

7. `¬mere(rose, john)`

De `7` et `2` on déduit la clause vide □ (faux/contradiction). Donc `grand_mere(rose, caroline)` se déduit des équations `1.`, `2.`, `3.` et `4.` C'est une proposition vraie.

Résumé

☞ *Poser une question, c'est introduire dans le système, la négation de la question et montrer que l'ensemble des clauses est inconsistant (démonstration par l'absurde ou réfutation). Pour cela, il faut réussir à engendrer la clause vide en appliquant la règle de résolution autant de fois que nécessaire.*

Il n'y a pas de mini-TP sur cette section établissant la liaison entre Prolog et logique mathématique.