

## 2.2. Des prédicats primaires définis par des faits

(VIDEO-PPT 1.2)

Un *fait* est une relation entre des objets qui est toujours vraie.

Exemple : « *Joseph est le père de John* ».

- ➔ Une relation : « *père* »
- ➔ Des objets : « *Joseph* », « *John* »

Ce qui se représente par le fait `pere(joseph, john)`.

Vocabulaire :

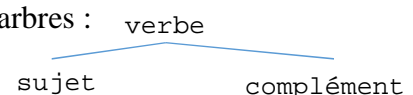
- `pere` est un prédicat (ici binaire) primaire (construit sur des faits)
- `joseph` et `john` sont des arguments (ici des constantes symboliques)

☞ *La syntaxe requiert que prédicats et constantes symboliques commencent par une minuscule.*

Autres exemples de Langue Naturelle traduits en Prolog :

Expression en LN	Notation Gprolog
Toto parle à Luc	<code>parleA(tom, luc)</code>
L'or est précieux	<code>estPrecieux(or)</code>
Max donne Milou à Jean	<code>donne(max, milou, jean)</code>

L'ordre d'écriture des arguments de la relation que décrit un prédicat est arbitraire mais il est choisi une fois pour toutes par le programmeur. L'habitude d'écriture est : `verbe(sujet, complément)`. En représentation interne, Prolog manipule des arbres :



Prolog n'attache pas de sens aux mots, il s'appuie sur des traitements syntaxiques. Ainsi, si pour une expression, le programmeur décide que le premier argument de la relation correspond au sujet, il faut garder ce positionnement pour toute la base de connaissances.

Les faits :

```
parleA(toto, luc).
parleA(luc, toto).
```

expriment deux choses différentes. Prolog voit deux arbres dont les arguments sont différents et ne gère pas la commutativité.

Créons maintenant une véritable base de connaissances, nous nous servirons pour l'exemple de la généalogie de John Kennedy, extrait de `familleKennedy.pl` :

```

pere(joseph, john).
pere(joseph, robert).
pere(joseph, edward).
pere(john, john_john).
pere(john, caroline).
pere(robert, kathleen).

mere(rose, edward).
mere(rose, john).
mere(rose, robert).
mere(rose, rose_marie).

```

☞ Tous les faits définissant un même prédicat sont regroupés en séquence dans l'ordre énoncé par l'utilisateur. Ils forment un paquet de clauses. Cet ordre n'a pas de sens logique (et donc pas d'importance) mais, d'un point de vue pratique, cela facilite les recherches de l'interpréteur.

### 2.3. Des prédicats secondaires construits par des règles

Les règles expriment des relations conditionnelles entre des prédicats. Ainsi :

« X est grand-mère de Y s'il existe Z tel que X est mère de Z

et (Z est mère de Y ou Z est père de Y) »

soit :

$$(\forall x \forall y \forall z (mère(x,z) \wedge mère(z,y)) \Rightarrow grand\_mère(x,y))$$

$\wedge$

$$(\forall x \forall y \forall z (mère(x,z) \wedge père(z,y)) \Rightarrow grand\_mère(x,y))$$

ce qui se développe avec deux implications à « but » unique avec des conjonctions implicites, soit des expressions de la forme :

$$sous-but_1 \text{ sous-but}_2 \dots sous-but_n \Rightarrow but$$

ce qui s'écrit en Prolog avec deux règles d'effacement de but (noter l'inversion entre prémisses et conclusion) :

```


grand_mere(X,Y) :- mere(X,Z), mere(Z,Y).
grand_mere(X,Y) :- mere(X,Z), pere(Z,Y).

```

et se lit pour la première règle « *pour prouver grand\_mere(X,Y), il faut prouver mere(X,Z) puis mere(Z,Y).* »

*miniTP* : établissez que

- 1) Rose est la grand-mère de Caroline mais pas de Robert
- 2) Robert est le frère de John
- 3) John est le frère de Robert !

 Une question fermée (pas de variable en entrée) peut être vérifiée de plusieurs façons. Puisque la réponse est juste binaire (vrai/faux, sans variable en sortie), seul le premier succès importe et il suffit de stopper la recherche d'autres possibilités en répondant **<RC>** au ? de l'interpréteur.