Prolog – TP n°1
— Faits et règles simples —

Pierre-Alexandre FAVIER

2005 - 2006

1 Rappel : la syntaxe Prolog

1.1 Faits et règles

Une clause valide peut être soit un fait, soit une règle.
Toute clause se termine par un point.

La déclaration d'un fait consiste donc en l'écriture de ce fait suivie d'un point. Ce fait est désigné sans ambiguïté par son foncteur et son arité. Un foncteur valide commence impérativement par une lettre minuscule, les majuscules et le symbole _ sont réservés aux noms de variables.
Les différentes clauses d'un même prédicat doivent être regroupées dans le fichier source.

Par exemple, pour déclarer que Prolog est un langage :
`langage(prolog).`

Il s'agit du prédicat `langage/1` (*foncteur/arité*).

Un fait déclaré est *formellement* vrai, un fait ne peux pas être déclaré faux : il n'est pas déclaré.

Une règle possède une tête et un corps, séparés par l'opérateur `:-`
La règle suivante :
`a :- b , c.`
se lit : "*a si b et c*"

La virgule dénote la conjonction (*et*) alors que le point-virgule dénote la disjonction (*ou*) :
`a :- b; c.`
se lit : "*a si b ou c*"
Comme un même prédicat peut être défini par plusieurs clauses qui sont autant d'alternatives possibles pour la démonstration, on préférera toujours l'écriture sous forme de clauses distinctes qui est plus lisible. La disjonction précédente sera donc exprimées

comme suit :

```
a :- b.
a :- c.
```

1.2 Les variables

Les noms des variables commencent par une majuscule ou par le symbole _

```
programmer(X) :- langage(X).
```

Une variable dont le nom commence par le caractère _ sera traitée en interne : Prolog n'indiquera pas la valeur que doit prendre cette variable pour satisfaire le prédicat.

Exemple :

```
?- langage(X).
X = prolog

yes
?- langage(_).
yes
?-
```

2 Un petit réseau routier

On considère un réseau routier décrit par la base de faits suivante :

```
route(s,a).  % une route relie la ville s a la ville a
route(s,d).
route(a,b).
route(a,d).
route(b,c).
route(b,e).
route(d,e).
```

1. Représenter le réseau ainsi défini.
2. Quels types de questions peut–on poser lors de la consultation de cette base de faits ?
3. Définir le prédicat
   voisines(X,Y) *% Il existe une route directe entre* **X** *et* **Y**

3 Une base de données simple

On considère une base de faits décrivant les vols d'une compagnie aérienne :

```
vol(numero du vol,
    ville de depart,
    ville d'arrivee,
    heure de depart,
    heure d'arrivee,
    nombre de passagers)
```

1. Décrire plusieurs vols selon le format précédent.
   Exemples de vols :

   ```
   vol(1,bordeaux,paris,1400,1500,100).
   vol(2,bordeaux,lyon,0600,0800,25).
   ...
   ```

2. Quelles questions PROLOG doit–on poser pour déterminer les vols (identifiés par leurs numéros) :
   (a) au départ d'une ville donnée ?
   (b) qui arrivent dans une ville donnée ?
   (c) au départ d'une ville donnée et qui partent avant midi ?
   (d) arrivant dans une ville donnée à partir de 14 h ?
   (e) qui ont plus de 100 passagers et qui arrivent dans une ville donnée avant 17 h ?
   (f) qui partent à la même heure de deux villes différentes ?
   (g) qui durent plus de deux heures ?

4 Arbre généalogique

On considère un arbre généalogique décrit par la base de faits suivante :

– **homme/1**

```
homme(jean).                    % jean est un homme
homme(fabien).
homme(jerome).
homme(emile).
homme(franck).
homme(bruno).
homme(marc).
```

– **femme/1**

```
femme(evelyne).               % évelyne est une femme
femme(louise).
femme(julie).
femme(anne).
```

```
femme(aurelie).
femme(sophie).
femme(marie).
femme(eve).
```

– **pere/2**

```
pere(emile,jean).          % émile est le père de jean
pere(jean,fabien).
pere(fabien,eve).
pere(jean,jerome).
pere(bruno,evelyne).
pere(bruno,franck).
pere(franck,sophie).
pere(franck,aurelie).
```

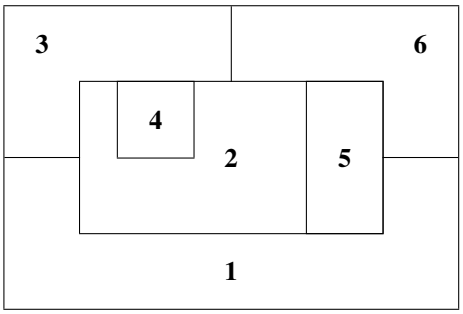
– **mere/2**

```
mere(louise,jean).         % louise est la mère de jean
mere(julie,evelyne).
mere(julie,franck).
mere(evelyne,fabien).
mere(evelyne,jerome).
mere(anne,sophie).
mere(anne,aurelie).
mere(marie,eve).
mere(sophie,marc).
```

1. Définir les prédicats suivants :
   (a) **parent(X,Y)** *% X est un des parents de Y*
   (b) **fils(X,Y)** *% X est le fils de Y*
   (c) **gdPere(X,Y)** *% X est le grand-pere de Y*
   (d) **ancetre(X,Y)** *% X est l'ancetre de Y*

5 Coloriage d'une carte

On dispose de 4 couleurs (rouge,jaune,vert,bleu) pour colorier la carte représentée ci–dessous.



1. Ecrire un prédicat qui choisit 2 couleurs valides différentes.
2. Ecrire un programme PROLOG qui permette d'associer une couleur (rouge, jaune, vert, bleu) à une région (1,2,3,4,5,6) de telle manière que deux régions adjacentes ne soient pas de la même couleur.

6 Petit interpréteur PROLOG

Ecrire un interpréteur PROLOG simplifié (**prolog(+But)**) qui permette de prouver un but, une conjonction de buts et/ou une disjonction de buts.

Les règles seront écrites sous la forme :

regle(Conclusion,Conditions)

et les faits sous la forme :

regle(Fait,vrai)

`Exemple de programme :`

regle(pere(e,j),vrai).
regle(pere(j,f),vrai).
regle(gdPere(X,Y),(pere(X,Z),pere(Z,Y))).

7 Entiers naturels

Traduire en PROLOG les axiomes définissant les nombres entiers non négatifs ($\mathcal{N}$) et les principales opérations les concernant.

Le vocabulaire initial de la théorie des entiers non négatifs comprend :
- une constante z qui représente l'entier 0 (zéro) ;
- une fonction unaire $s(X)$ qui traduit la notion de successeur d'un entier **X** ;
- un prédicat unaire **entier(X)** (**X** *est un entier*).

1. **Génération :**
$$z \in \mathcal{N} \text{ et } \forall x \in \mathcal{N}, s(x) \in \mathcal{N}$$
2. **Addition :**
$$\forall x \in \mathcal{N}, x + z = x \text{ et } \forall x, y \in \mathcal{N}, x + s(y) = s(x + y)$$
3. **Multiplication :**
$$\forall x \in \mathcal{N}, x \cdot z = z \text{ et } \forall x, y \in \mathcal{N}, x \cdot s(y) = x \cdot y + x$$
4. **Prédécesseur :**
$$\forall x \in \mathcal{N}, p(s(x)) = x$$
5. **Inférieur ou égal :**
$$\forall x \in \mathcal{N}, x \leq z \Leftrightarrow x = z \text{ et } \forall x, y \in \mathcal{N}, x \leq y \Leftrightarrow x = y \text{ ou } x \leq p(y)$$
6. **Strictement inférieur :**
$$\forall x, y \in \mathcal{N}, x < y \Leftrightarrow x \leq y \text{ et } x \neq y$$

8 Dérivation symbolique

Définir le prédicat **d(+Expression,+Variable, ?Derivee)** vrai si et seulement si **Derivee** est la dérivée de l'**Expression** par rapport à la **Variable** (variable au sens mathématique du terme).

On rappelle les principaux résultats concernant les dérivées de fonctions :

- $\frac{dC}{dx} = 0$ si $C = Cte_{/x}$
- $\frac{d(x^n)}{dx} = nx^{n-1}$
- $\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx}$
- $\frac{d(u \cdot v)}{dx} = v \cdot \frac{du}{dx} + u \cdot \frac{dv}{dx}$
- $\frac{d(f(u))}{dx} = \frac{du}{dx} \cdot \frac{d(f(u))}{du}$
- $\frac{d(sin(x))}{dx} = \cos(x)$
- ...
- $\frac{d(\log(x))}{dx} = \frac{1}{x}$
- ...

9 Arithmétique

1. Définir les prédicats PROLOG correspondant aux suites récurentes suivantes :

   (a) **Suite factorielle :**

   $u_0 = 1$
   $u_n = n \cdot u_{n-1}\ \forall n \in \mathcal{N}^*$

   (b) **Suite de Fibonacci :**

   $u_0 = 1$
   $u_1 = 1$
   $u_n = u_{n-1} + u_{n-2}\ \forall n \in \mathcal{N}, n > 1$

10 Arbres binaires

Un arbre binaire est une structure soit vide, soit composée de trois éléments :
– une racine **Val**,
– un sous–arbre binaire **Gauche**,
– un sous–arbre binaire **Droite**.
Un arbre binaire vide sera représenté par le terme atomique **[]**, un arbre non vide par le terme composé **bt(Gauche,Val,Droite)**.

1. Définir le prédicat **arbreBinaire(?Terme)** vrai si et seulement si **Terme** est un arbre binaire.
2. Définir les prédicats suivants concernant les arbres binaires :
   (a) **dansArbre(X,A)** *% **X** est un élément de l'arbre binaire **A***
   (b) **profondeur(A,N)** *% **N** est la profondeur de l'arbre binaire **A***

11 Termes de base

1. Définir le prédicat **substituer(?Ancien, ?Nouveau,+Terme, ?NouveauTerme)** vrai si et seulement si **NouveauTerme** correspond au **Terme** dans lequel tous les arguments **Ancien** ont été remplacés par le sous–terme **Nouveau**.
   Exemple :

```
?- substituer(b,a, a(a, b, c,a),X).

X = a(a, a , c,a)

Yes
```

2. Définir le prédicat **substituer(?Ancien, ?Nouveau,+Terme, ?NouveauTerme)** vrai si et seulement si **NouveauTerme** correspond au **Terme** dans lequel toutes les occurences du sous–terme **Ancien** ont été remplacées par le sous–terme **Nouveau**, y compris les foncteurs. Exemple :

```
?- substituer(a, b, a(a, t(g), b(a), a(f)),X).

X = b(b, t(g), b(b), b(f))

Yes
```

12 Exercices supplémentaires

12.1 Arbre généalogique

En vous basant sur la représentation du 4, écrire les prédicats suivants :

1. Définir les prédicats suivants :
   (a) **parent(X,Y)** *% X est un des parents de Y*
   (b) **fils(X,Y)** *% X est le fils de Y*
   (c) **fille(X,Y)** *% X est la fille de Y*
   (d) **femme(X,Y)** *% X est la femme de Y*
   (e) **mari(X,Y)** *% X est le mari de Y*
   (f) **gdPere(X,Y)** *% X est le grand-pere de Y*
   (g) **gdMere(X,Y)** *% X est la grand-mere de Y*
   (h) **gdParent(X,Y)** *% X est un des grand-parents de Y*
   (i) **aGdPere(X,Y)** *% X est l'arriere grand-pere de Y*
   (j) **aGdMere(X,Y)** *% X est l'arriere grand-mere de Y*
   (k) **frereSoeur(X,Y)** *% X est le frere ou la soeur de Y*
   (l) **frere(X,Y)** *% X est le frere de Y*
   (m) **soeur(X,Y)** *% X est la soeur de Y*
   (n) **beauFrere(X,Y)** *% X est le beau-frere de Y*
   (o) **belleSoeur(X,Y)** *% X est la belle-soeur de Y*
   (p) **ancetre(X,Y)** *% X est l'ancetre de Y*

12.2 Entiers naturels

A partir de l'exercice 7 Traduire en PROLOG les axiomes définissant les nombres entiers non négatifs ($\mathcal{N}$) :

1. **Exponentiation :**

$$\forall x \in \mathcal{N}, x^z = s(z) \text{ et } \forall x, y \in \mathcal{N}, x^{s(y)} = x^y \cdot x$$

2. **Soustraction :**

$$\forall x \in \mathcal{N}, x - z = x \text{ et } \forall x, y \in \mathcal{N}, s(x) - s(y) = x - y$$

3. **Quotient et reste :**

$$\forall x \in \mathcal{N}, \forall y \in \mathcal{N}^*, x < y \Rightarrow quot(x, y) = 0 \text{ et } quot(x + y, y) = s(quot(x, y))$$

$$\forall x \in \mathcal{N}, \forall y \in \mathcal{N}^*, x < y \Rightarrow rest(x, y) = x \text{ et } rest(x + y, y) = rest(x, y)$$

4. **Division :**

$$\forall x \in \mathcal{N}, \forall y \in \mathcal{N}^*, x \div y \Leftrightarrow \exists q \in \mathcal{N}^* \text{ tel que } x \cdot q = y$$