Algorithmes & Raisonnement

2023-2024

TD1 : Prise en main de Prolog

0.1 Installation de SWI Prolog

1. Télécharger swi-prolog : https://www.swi-prolog.org/download/stable selon votre plateforme.

   Selon votre plateforme, procédez à l'installation :

   ◦ **Windows** : exécuter swipl-xxx-64.exe. Cela installera Prolog comme une application windows quelconque.

   ◦ **Mac** : ouvrir swiplxxx.dmg et procédez à l'installation dans le dossier Applications

   ◦ **Linux** : sudo apt install swi-prolog

2. Lancez l'exécutable *swi prolog* (selon les plateformes) :

   ◦ Windows et Mac : lancez l'exécutale **swipl** (vous savez lancer une application installée sur votre plateforme !)

   ◦ Linux : dans un efenetre "terminal" (console), lancez l'exécutale **swipl**

3. Taper la requête **set_prolog_flag(editor, pce_emacs).** (n'oubliez pas le '.')

   ➛ Nous aurons besoin une seule fois d'exécuter cette requête pour dire notre choix de l'éditeur.

   ➛ Lors des prochaines utilisations, Prolog s'en souviendra.

4. Plus de documentation et aide sur https://www.swi-prolog.org/pldoc/man?section=pceemacs.

5. D'une manière générale, vous pouvez sélectionner l'éditeur que vous voulez (à la place de *pce_emacs*).

   ☞ Par exemple, sous Windows, vous pouvez également demander **set_prolog_flag(editor, wordpad).**. Voir également plus bas si votre éditeur n'est ni *pce_emacs* ni *wordpad*.

6. Quelque soit votre plateforme, sous Prolog, vous disposez des commandes 'cd', 'pwd', ...

   Par exemple, pour savoir dan srépertoire vous-êtes, taper :

   ```
   pwd.        % le répertoire courant
   ```

   Pour vous placer dans votre répertoire usuel (répertoire de votre compte) taper :

   ```
   cd('~/')      % vous placera dans votre répertoire principal
   ```

   Edition de programmes : 1e solution

7. Sous Prolog, taper la requête suivante pour éditer / créer le fichier 'prog1.pl'.

   edit(file('prog1.pl')). % ➛ true : la réponse de swi

   Cette requête lance l'éditeur. Il s'agit d'un éditeur élémentaire mais utile.

   ☞ Cette requête crée le fichier 'prog1.pl'. Si 'prog1.pl' existe déjà dans le répertoire actuel, taper simplement **"edit('prog1.pl')."**. Néanmoins, *edit(file('prog1.pl')).* fonctionnera toujours !

8. Sur MacOs, si l'exécution de cette dernière commande échoue, c'est parce que vous n'avez pas déjà installé tous les modules nécessaires sur votre machine.

   Edition de programmes : 2e solution

   Dans ce cas, utilisez un éditeur quelconque hors Prolog puis consultez votre programme. Par exemple, utilisez l'éditeur 'textmate' (version gratuite existe); éditez 'prog1.pl'; sauvegardez et sous prolog, chargez le programme par `['prog1.pl']`.

   Vous pouvez également utiliser les éditeurs VS-Code / VS-Codium.

9. Si vous avez sélectionné un autre éditeur que *pce_emacs* ou *wordpad* (p. ex, si votre éditeur est 'vi' sous Mac ou Linux), après l'édition et la sauvegarde de votre fichier 'prog1.pl', une fois revenu sous Prolog, vous serez obligé de consulter votre programme *'prog1.pl'* par la requête `['prog1.pl']` pour le charger explicitement.

10. Sous l'éditer, taper les lignes suivantes (pour tester) :

```
mortel(X) :- homme(X).

homme(socrates).

homme(aristote).

homme(man).
```

11. Sauvegardez le fichier (Menu File / Save)

12. Si vous avez utilisé l'éditeur *pce_emacs*, dans le menu de l'éditeur, sélectionner "compile buffer" (ou "build").

    ➛ Cela charge votre programme sous Prolog.

    ➛ Si vous avez des erreurs, elles sont signalées; corrigez !

    Sinon, sous Prolog, taper `['prog1.pl'].`

    N.B. : même si vous avez utilisé l'éditeur *pce_emacs*, vous pouvez toujours charger votre code par `['prog1.pl'].`

13. Maintenant, taper la requête

    `mortel(X).`

    ➛ Pour avoir les autres réponses, appuyez sur ";".

    ➛ Un retour-chariot (touche "entrée") arrête les réponses.

14. Tester le mode trace en tapant "trace." puis retaper la requête `mortel(X).` pour avoir une trace.

15. Terminer la trace avec "notrace".

16. Pour quitter Prolog, on tapera la commande "**halt.**" .

Deux remarques (utiles pour plus tard; voir avec l'enseignant) :
1) Pour avoir accès aux contraintes du domaine fini, il faut importer la librairie concernée. On exécutera la requête

```
use_module(library(clpfd)).
```

D'une manière plus générale, insérer au début de vos fichiers Prolog la requête

```
:- use_module(library(clpfd)).
```

2) Par défaut, si vous essayez d'exécuter une requête qui appelle un prédicat inexistant, vous aurez une erreur. Par exemple, toujours sous Prolog, taper la requête

```
femme(X).        % Vous aurez un message d'erreur "prédicat inexistant"
```

Si vous voulez obtenir un échec (*fail*) à la place de l'erreur, il faudra insérer au début de votre fichier de programme :

```
:- set_prolog_flag(unknown, fail).
```

➛ On ne peut exécuter la même requête sous Prolog (on aura un refus). Par contre, vous pouvez réaliser le même effet en utilisant la requête prédéfini `dynamic`.
➛ L'utilisation de `dynamic nom_prédicat` est préférable `:- set_prolog_flag(unknown, fail).`.

On peut connaître l'état de ce paramètre par la requête `current_prolog_flag(unknown, X).`
3) La négation du prédicat `p` se dit `not(p)` ou $\backslash + p$

Les séances de BE de ce module utilisent le langage **Prolog**.
Avant de commencer le BE, vérifiez en section 0.1 en page 1 que vous avez bien installé le langage Prolog selon les plateformes Linux, Mac ou Windows.

On suppose ci-dessous que vous savez éditer et charger un programme Prolog.

0.1.1 Premiers Tests

Comme en TD1, un premier programme peut contenir :

```
mortel(X) :- homme(X).   % une règle :  "∀X, X est mortel si X est un homme"
homme(socrates).          % un fait :  "socrates est un homme".
homme(aristote).     % laisser en minuscule (Aristote avec 'A' = une variable)
homme(man).
```

NB : ce programme contient une **règle** et 3 **faits**.

*- Une **règle** est de la forme : partie gauche :- partie droite.*
- Intuitivement, pour démontrer la partie gauche d'une règle, il faudra démontrer sa partie droite.
*- Un **fait** est avéré (c'est comme une règle dont la partie droite = true.)*
- Une variable commence par une lettre MAJUSCULE (ou par '_').

Maintenant, on va poser des questions :

```
| ?- mortel(X).      % Qui est mortel ?  (∃X telle que mortel(X)=vrai ?)
→ X=socrates ?        % taper ";"  devant '?'  pour avoir la réponse suivante.
→ X=aristote  ?
→ X=man
→ yes
```

Remarques :

1- Sous Prolog, utiliser les **4 flèches** du pavé à droite pour rappeler les requêtes précédentes.

Par exemple, à cet endroit, la flèche haute rappelle "mortel(X)." tapée en dernier.

2- Pour obtenir les réponses à une requête :

- si la requête n'a pas de réponse, on obtient "false" ou "no",
- si la requête a une seule réponse, on obtient la réponse suivi de "True" ou "yes"
- si la requête a plusieurs réponse, la première est donnée et Prolog attend notre réaction :
  - on appuie sur le "point virgule" pour la réponse suivante
  - on appuie sur "retour charriot" pour stopper les autres réponses.
- Controle-C + "retour chariot" arrête les choses brutalement (abandon)

3- Un autre intérêt d'avoir les réponses une-par-une est de pouvoir arrêter facilement une résolution en cas de *boucle infinie*

0.1.2 Rappel : comment charger son programme

Sous l'éditeur, éditer un fichier fourni : par exemple 'ex1.pl'.

Comment / où placer vos programme Prolog ?

1. Vous pouvez placer vos fichiers Prolog (d'extension ".pl") dans le répertoire où vous avez installé Prolog. L'inconvénient est de mélanger vos fichiers avec ceux du Prolog !

2. Vous pouvez décider d'un répertoire (disons "/xx/yy/BE") et y placer vos programmes Prolog. Admettons que vous avez dans ce répertoire le programme "ex1.pl" (extension ".pl" obligatoire).
   Pour le charger, vous pourrez taper (sous Prolog):
   consult('/xx/yy/BE/ex1.pl'). ou ['/xx/yy/BE/ex1.pl'].
   Sous Windows : consult('C : \\xx\\yy\\BE\\ex1.pl'). ou ['C : \\xx\\yy\\BE\\ex1.pl'].

3. Vous pouvez changer de répertoire par la requête "**cd('nom répertoire').**"
   ☞ Sous swi-prolog, en plus simple : $cd('C : /xx/yy/BE')$.
   ☞ Sous Prolog, : $cd('C : \xx\yy\BE')$.
   Une fois placé dans dans le répertoire de vos programmes, vous aurez juste besoin de donner le nom du programme. Par exemple, *consult('ex1.pl').* ou *['ex1.pl'].*
   ➛ Vous pouvez également obtenir la liste des fichiers du répertoire courant par la requête "**dir.**".

4. Aussi, vous pouvez vous placer dans le répertoire de vos BEs ("/xx/yy/BE") et de lancer Prolog. Pour cela, il faut placer le chemin d'accès à Prolog dans la variable PATH.
   Demandez à votre enseignant comment faire. La Doc de Prolog l'explique également.

... On reprend ... Charger ce programme par :

```
| ?- ['ex1.pl'].
```
Ou par :
```
| ?- consult('ex1.pl').
```

Vous pouvez aussi écrire "consult(" puis glisser le fichier "ex1.pl" après la parenthèse puis ajouter ").".

➛ Le chargement d'un programme Prolog se fait par l'une des commandes Prolog suivantes :

```
consult(ex1).  %pour un nom de fichier simple, sinon, entourez de '
consult('ex1.pl').
consult('/home/moi/TPs/ex1.pl').
consult('C:/Documents and Setting/moi/TPs/ex1.pl').
```

ou de manière équivalente, en tapant l'une des requêtes suivantes (n'oubliez pas le '.' final) :

```
[ex1].
['ex1.pl'].
['/home/moi/TPs/ex1.pl'].
```

L'extension par défaut des programmes Prolog est ".pl".
- Le caractère '**%**' marque le début d'un ***commentaire*** sur la même ligne.

On peut aussi insérer des commentaires sur plusieurs lignes par */* .....sur plusieurs lignes ..... */*

- Un programme prolog est constitué d'un ensemble de *prédicats*.

Un prédicat = un paquet de règles / faits dont le nom commence par le même symbole de prédicat et le même nombre de paramètres. Par exemple, le prédicat" homme (.)".

La convention Prolog est de noter cela " homme/1 " : nom du prédicat = " homme " et il a un seul paramètre. Un autre exemple était " entier/1 " ci-dessus...

- Si votre programme contient des erreurs ou si vous devez changer son contenu, faites les modifications sous l'éditeur et recharger le programmer à l'aide de la commande Prolog "*consult*" ou par *[...]* (comme ci-dessus).

Poser des questions :

```
| ?- homme(X).
```

→ `....` ***les réponses.***