INFf3
Programmation logique

Organisation de l'UE

CM + TD (sauf 1ère semaine) / TP / CM + TD / TP / CM + TD / …

CM (1h30) : Benoît Lemaire

TD (1h30) : Benoît Lemaire

TP (2h) : Yasser Bourahla, Jérôme Euzenat

1 partiel + 1 projet + 1 examen

Partie 1
-
Introduction

Les principaux paradigmes en programmation

- Programmation impérative (Java, L1)
- Programmation fonctionnelle (Scheme, L1)
- Programmation orientée objet (Java, L2)
- *Programmation déclarative (Prolog, L2)*

Programmation impérative

- Démarche algorithmique qui décrit la façon de traiter les données pour atteindre un résultat par une série d'actions (instructions).
- L'ordre d'exécution des instructions est impératif : déterminé à l'avance.
- Le déroulement du programme est parfaitement déterministe.
- Importance des structures de contrôle
- Exemples : Java, C, Basic, Pascal, Fortran…

Programmation fonctionnelle

- La programmation consiste à faire de la composition de fonctions.
- Proche de la programmation impérative ; cependant ses fondements (λ-calcul) ainsi que l'absence de branchements et d'affectation (dans sa forme théorique) en fait un mode de programmation à part.
- Dans la programmation fonctionnelle, on distingue deux grandes familles de langages :
  - les langages fortement typés, ex : ML, Caml, Haskell
  - les langages non typés, ex : Lisp, Scheme

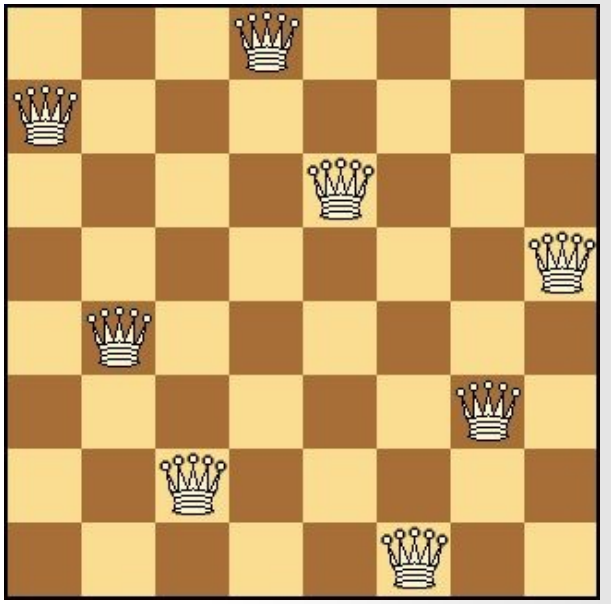
Programmation déclarative

Le programmeur ne s'occupe pas de la manière d'obtenir le résultat; par contre, il doit décrire le problème à résoudre en explicitant les objets concernés, leurs propriétés et les relations qu'ils vérifient.

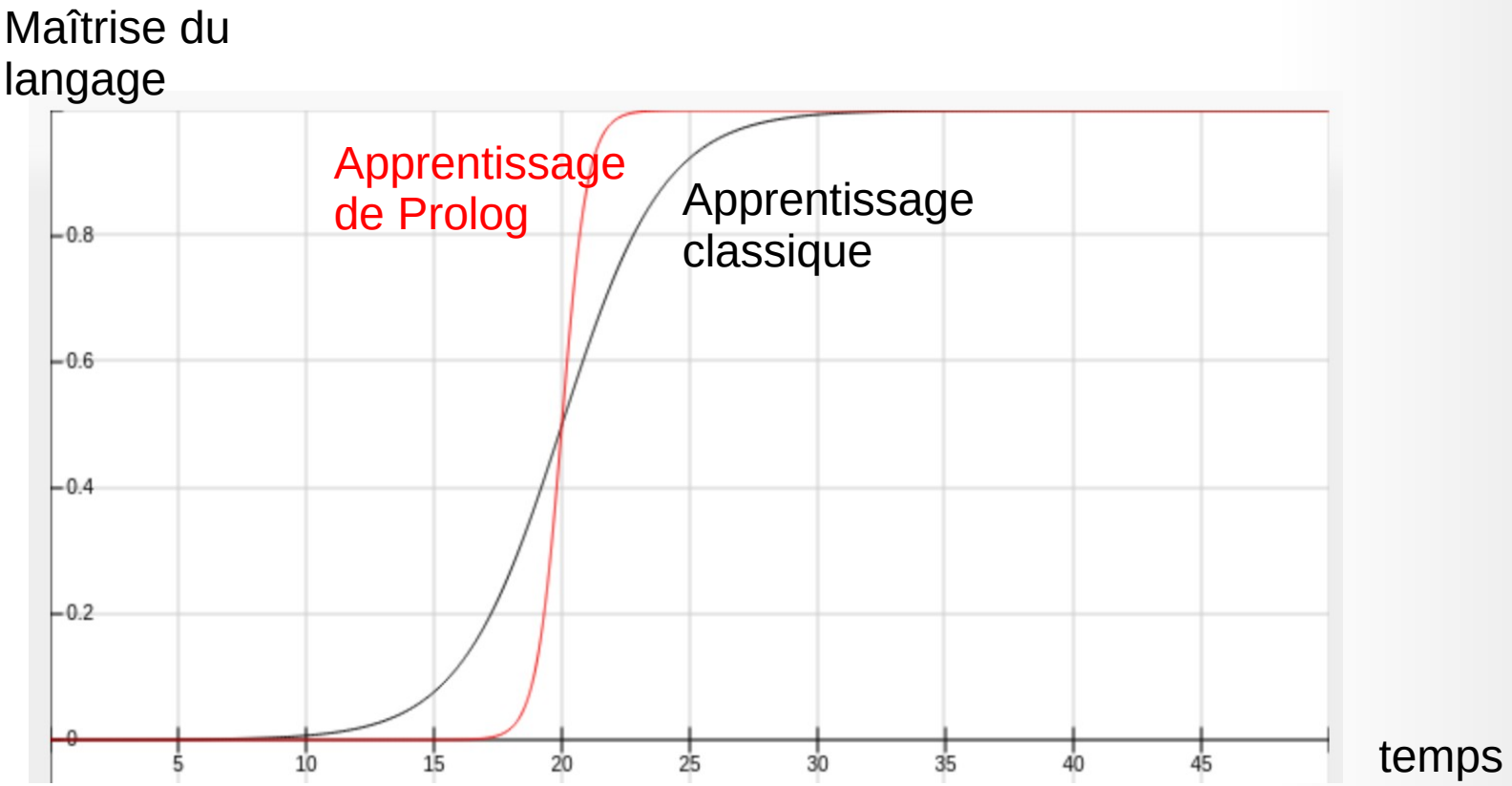
- Cette description constitue la **base de connaissance**
- Ensuite, le mécanisme de résolution intégré au langage, général et universel, parcourt de façon non déterministe toutes les possibilités du problème et calcule les solutions.
- Le mode de représentation est la logique des prédicats

Prolog

- Prolog = **Pro**grammation **Log**ique (Colmerauer, 1972)
- Utilisé quand il n'existe pas d'algorithme pour résoudre un problème, mais que les contraintes qui définissent la solution sont connues :
  - problème des 8 reines ;
  - SEND + MORE = MONEY ;
  - problème de satisfaction de contraintes
  - ...
- Un programme Prolog est composé de :
  - **faits** ;
  - **règles**
- et on l'exécute avec des **requêtes.**



Apprendre Prolog



Un programme est composé de

- **faits** (p. ex., le soleil est une étoile, la lune est satellite de la Terre, la somme des éléments d'une liste vide est 0, ...)
- **règles** (p. ex., X est une planète si X est un astre et E est une étoile et X est satellite de E, la somme des éléments d'une liste est le premier + la somme des éléments du reste de la liste).

Exécuter un programme, c'est

- faire une **requête** (p. ex., la Terre est-elle une planète ? ou Quelles sont toutes les planètes ?)

Faits

- On utilise la représentation logique d’ordre 1, ou logique des prédicats : **prédicat(arg1, arg2, …).**
- Un prédicat n'est pas une fonction ! (ou une fonction uniquement booléenne).
- Lien avec l'UE "Mathématiques pour l'informatique"
- Exemples :
  - `etoile(soleil).`
  - `capitale(france, paris).`
  - `temperaturesSemaine14([5,7,7,4,2,4,10]).`
  - `position(roi,b,7).`

Règles

- On utilise la représentation logique d'ordre 1 :

p1(…) ∧ p2(…) ∧… ∧ pn(...) ⇒ r(….)

ou

r(….) :- p1(…), p2(…), …, pn(…).

- Exemples :
  - `planete(X) :- astre(X),etoile(E),satellite(X,E).`
  - `fibo(X,R) :- X1 is X-1, X2 is X-2, fibo(X1,R1), fibo(X2,R2), R is R1+R2.`
  - `tousDifferents([X|L]):-not(member(X,L)), tousDifferents(L).`

Requêtes (exécution du programme)

- Une requête a la forme **prédicat(arg1,…, argn).**
- Le premier objectif est de déterminer si la requête est vraie ou fausse
  - `factorielle(4,24).`

    `true`
  - `planete(lune).`

    `false`
- S'il y a des variables dans la requête, Prolog indique ensuite leurs valeurs pour lesquelles la requête est vraie
  - `factorielle(4,X).`

    `X=24`

Exemples de requêtes à partir de faits

```
mange.
herbivore(chevre).
mange(loup, chevre).
```

```
?- mange.
true.
?- herbivore(chevre).
true.
?- herbivore(loup).
false.
?- carnivore(loup)
ERROR: unknown procedure carnivore/1
```

Exemples de requêtes à partir de faits et règles

```
mange(loup, chevre).
cruel(loup) :- mange(loup, chevre).
carnivore(loup) :- mange(loup, chevre), animal(chevre).
```

```
?- cruel(loup).
true.
?- carnivore(poireau).
false.
?- carnivore(loup).
ERROR: carnivore/1: Unknown procedure: animal/1
```

Pas d'individu poireau => Pas d'évaluation du prédicat carnivore => false

Individu loup => Evaluation du prédicat carnivore => prédicat non défini: animal

Règles Prolog

- Il peut y avoir plusieurs conditions derrière le « :- », séparées par des virgules ou des points-virgules
  - La virgule correspond à un ET logique (conjonction)
  - Le point-virgule correspond à un OU logique (disjonction)
  - Exemple :

    La relation telle que si on est le père du père ou de la mère de quelqu'un alors on est son grand-père se traduit par :

```
grandpere(xavier,yves) :- pere(xavier,joe), pere(joe,yves).
grandpere(xavier,yves) :- pere(xavier,joe), mere(joe,yves).
```

ou encore par :

```
grandpere(xavier,yves) :- pere(xavier,joe), (pere(joe,yves) ;
   mere(joe,yves)).
```

Exemple de requêtes avec variables

```
grandpere(X,Y) :- pere(X,P), pere(P,Y).
grandpere(X,Y) :- pere(X,P), mere(P,Y).
```

Exemple de requêtes avec variables

```
mange(loup, chevre).
carnivore(X) :- mange(X, Y), animal(Y).
cruel(X) :- mange(X, _).
```

```
?- cruel(loup)
true
?- carnivore(loup)
ERROR: carnivore/1: Unknown procedure: animal/1
?- carnivore(poireau)
false
```

Prédicat

- Un programme **Prolog** est constitué d'une suite de clauses regroupées en paquets.
- Chaque paquet définit un **prédicat** ; il est constitué d'un ensemble de clauses qui ont même symbole de prédicat et même arité.
- L'ordre dans lequel les clauses sont définies est important.

Exemple d'exécution

```
planete(X) :- astre(X), étoile(Y), satellite(X,Y).

astre(lune).

astre(terre).

astre(soleil).

astre(venus).

satellite(venus, soleil).

satellite(lune, terre).

satellite(terre, soleil).

étoile(vega).

étoile(soleil).
```

Constantes vs Variables

- Constantes: commencent par une minuscule (ou entre apostrophes), entiers ou flottants
- Variables : commencent par une majuscule ou par _
  - Les variables utilisées dans un fait ou une règle sont universellement quantifiées par Prolog, avec le quantificateur ∀ (« quel que soit » ou « pour tout »).
  - Les variables utilisées dans une requête sont existentiellement quantifiées par Prolog, avec le quantificateur ∃ (« existe-t-il »).

Variable anonyme

- Notée '_' (tiret bas)
- Représente un objet dont on ne souhaite pas connaître la valeur, c’est une variable muette.
- ?- invite(X,_).
  Prolog ne donnera que les valeurs pour X
- Attention, une variable commençant par '_' et de longueur >= 2 n'est pas muette par défaut en SWI-Prolog.

Exercice

- Exprimer en Prolog les propositions suivantes, identifier les objets, les faits, les règles
  - *la chèvre est un animal herbivore*
  - *le loup est un animal cruel*
  - *toute chose cruelle est carnivore*
  - *un animal carnivore mange de la viande et un animal herbivore mange de l'herbe*
  - *un animal carnivore mange des animaux herbivores*
  - *les carnivores et les herbivores boivent de l'eau*
  - *un animal consomme ce qu'il boit ou ce qu'il mange*
  - Question : y a-t-il un animal cruel et que consomme-t-il ?

Solution possible

```
animal(chevre).
animal(loup).
herbivore(chevre).
cruel(loup).

carnivore(X) :- cruel(X).

mange(X,viande):- animal(X), carnivore(X).
mange(X,herbe):- animal(X), herbivore(X).
mange(X,Y):- animal(X), animal(Y), carnivore(X), herbivore(Y).

boit(X,eau):- animal(X), (carnivore(X); herbivore(X)).

consomme(X,Y) :- mange(X,Y); boit(X,Y).
```

?- *animal(X), cruel(X), consomme(X,Y).*

la chèvre est un animal herbivore

le loup est un animal cruel

toute chose cruelle est carnivore

un animal carnivore mange de la viande et un animal herbivore mange de l'herbe

un animal carnivore mange des animaux herbivores

les carnivores et les herbivores boivent de l'eau

un animal consomme ce qu'il boit ou ce qu'il mange

Question : y a-t-il un animal cruel et que consomme-t-il ?

Quelques opérateurs utiles pour le TD 1

- On peut utiliser les comparaisons arithmétiques (<, >, =<, >=, =:=, =\=)
  - p. ex., X<5, Heure > 12, N<=T, ...
  - attention, **=<** est inversé !
  - attention, l'égalité est **=:=** ! L'opérateur = ne fait pas les calculs
- On peut faire des calculs avec `is` :
  - X is 3+2, Y is N*4+1
  - Attention, X = 5+2 ne fait pas le calcul !
- L'opérateur modulo est `mod` :
  - X is mod(13,4).

    X=1