

# PROLOG

## Programmation Logique

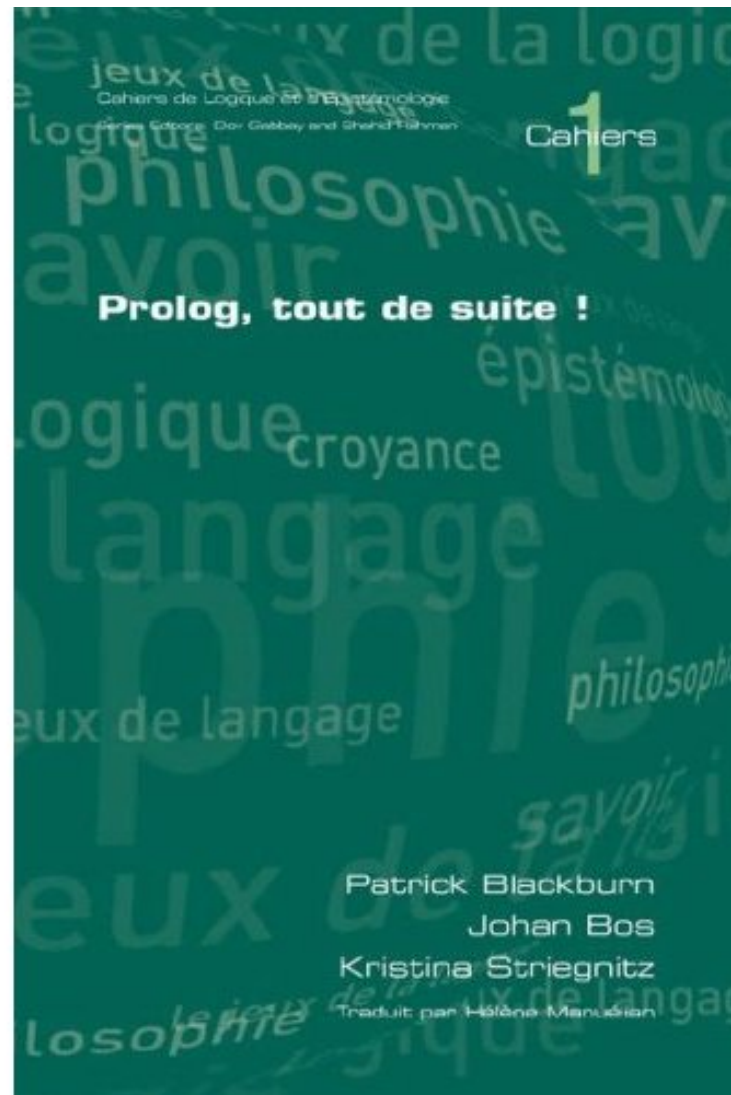
Yannick Estève  
Université du Maine

# Programmation logique

---

- Sources pédagogiques
  - Prolog, tout de suite !
    - Patrick Blackburn, Johan Bos, Kristina Striegnitz
  - Interpréteur Prolog : SWI Prolog
    - Autre possibilité :
      - GNU Prolog

# Prolog, tout de suite !



# SWI Prolog

---

- Interpréteur Prolog disponible sous licence LGPL
- Utilisable avec
  - Linux,
  - Windows, ou
  - Mac OS
- Beaucoup d'autres interpréteurs Prolog
- Tous ne sont pas ne respectent pas la norme ISO

# Cours 1

---

- Théorie
  - Introduction à Prolog
  - Faits, Règles et Requêtes;
  - Syntaxe de Prolog
- Exercices
  - Exercices du chapitre 1 de Prolog, TdS
  - TP

# Objectifs de cette séance

---

- Donner des exemples simples de programmes écrits en Prolog
- Aborder les 3 constructions de bases en Prolog :
  - Faits
  - Règles
  - Requêtes

# Objectifs de cette séance

---

- Introduire d'autres concepts :
  - Le rôle de la logique
  - L'unification avec l'aide de variables
- Commence l'étude systématique de Prolog en définissant :
  - Les termes
  - Les atomes
  - Les variables

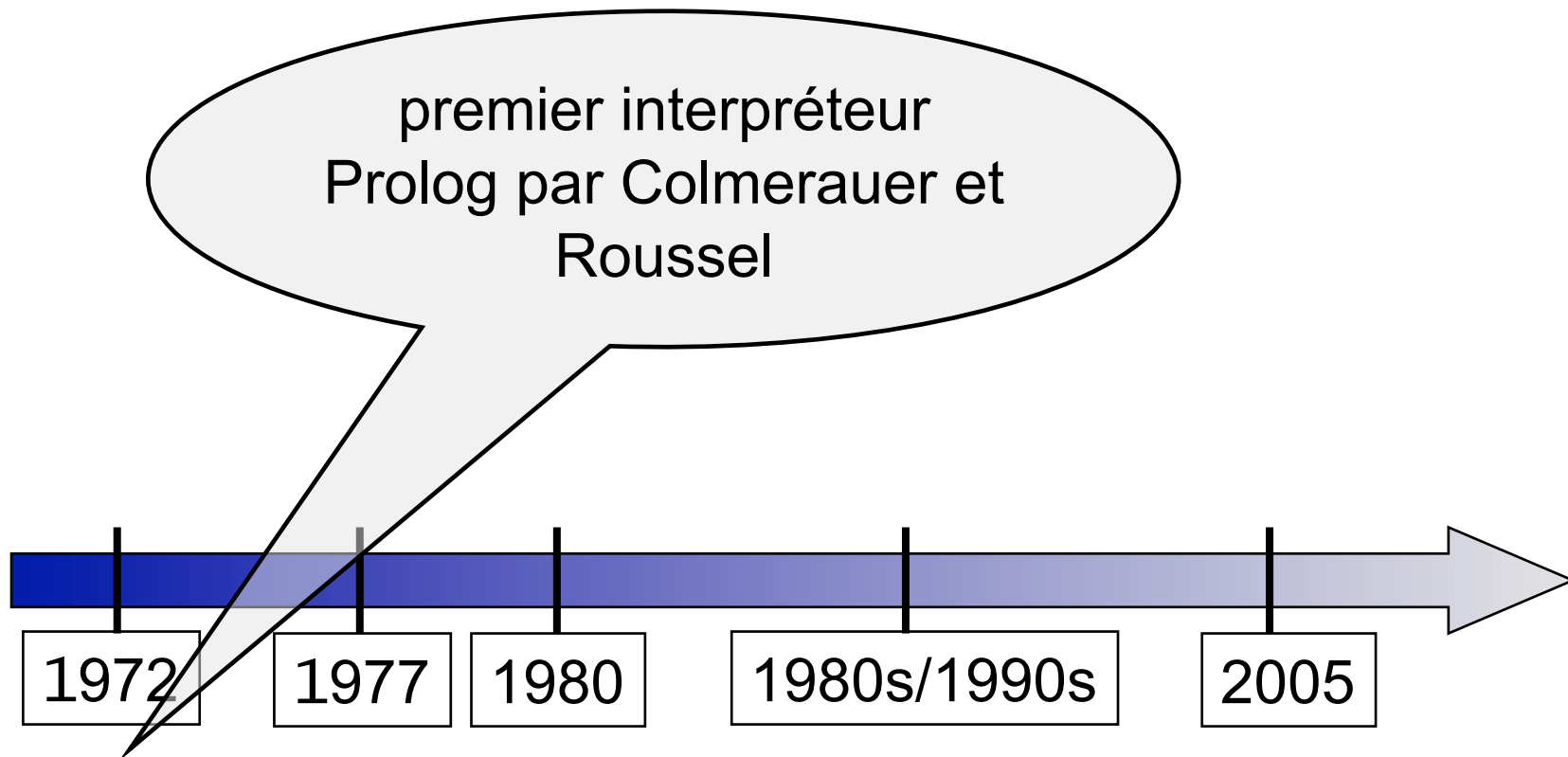
# Prolog

---

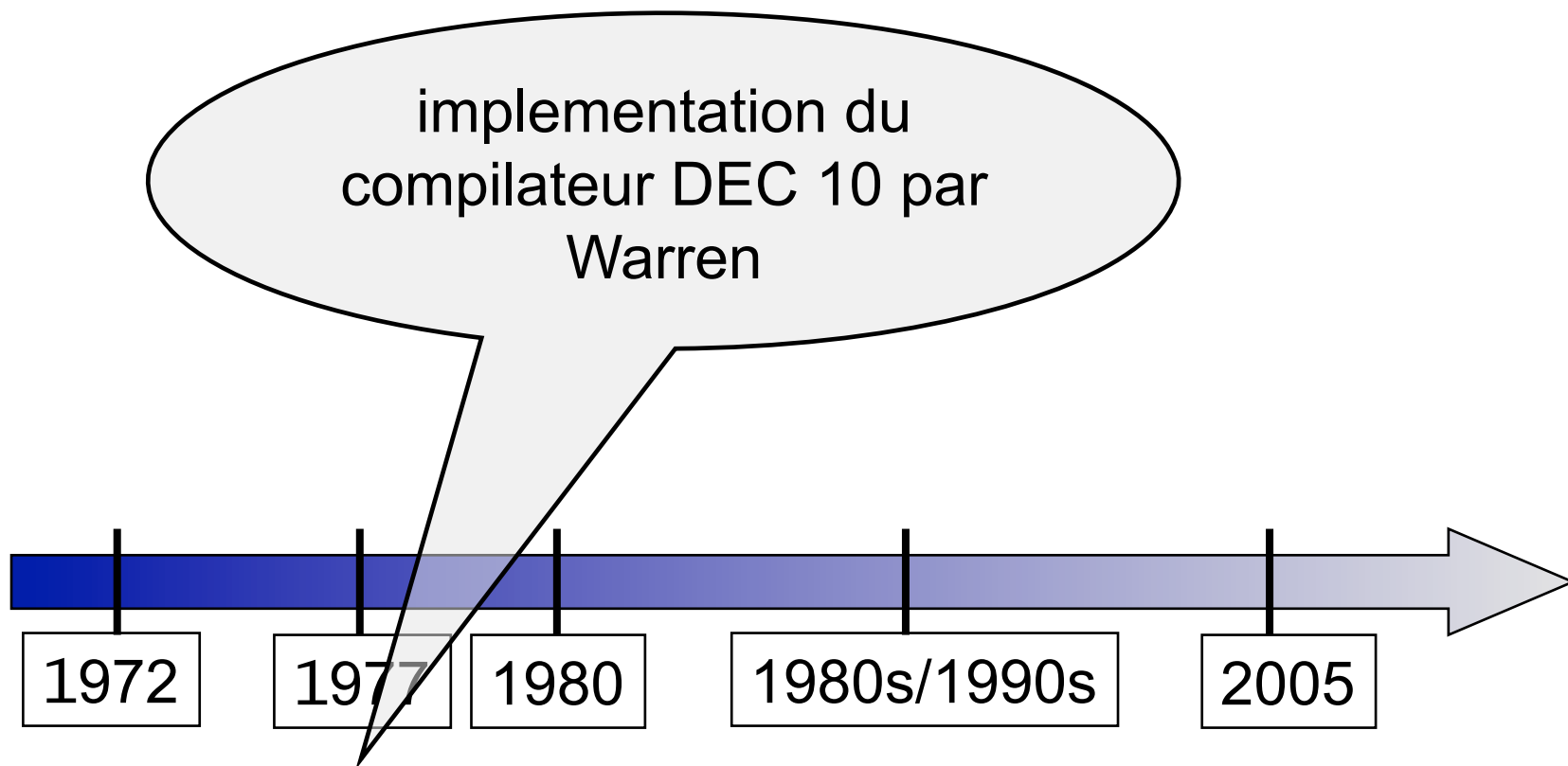
- "Programmation logique"
- Langage déclaratif
- Très différent d'autres langages de programmation (procéduraux)
- Pertinent pour des applications nécessitant des connaissances riches



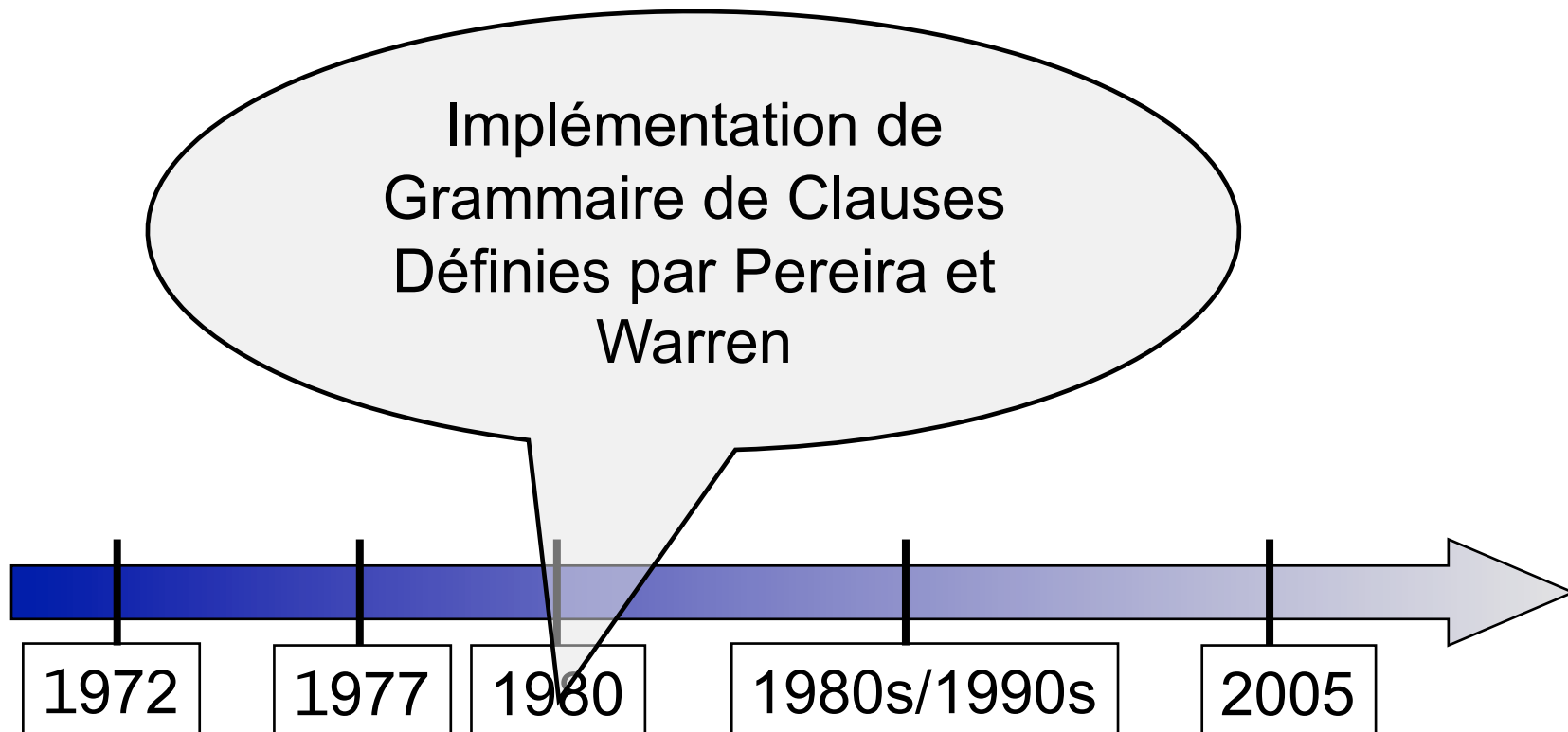
# Histoire de Prolog



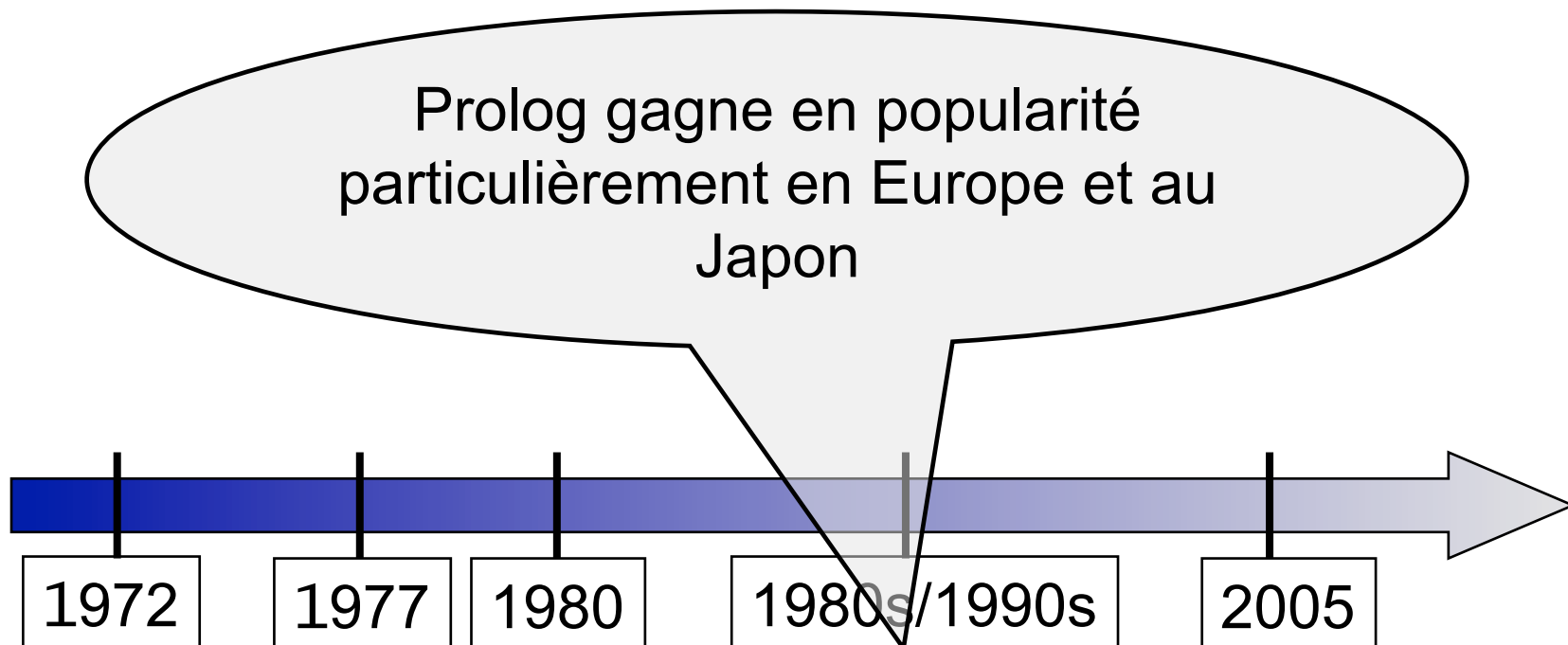
# Histoire de Prolog



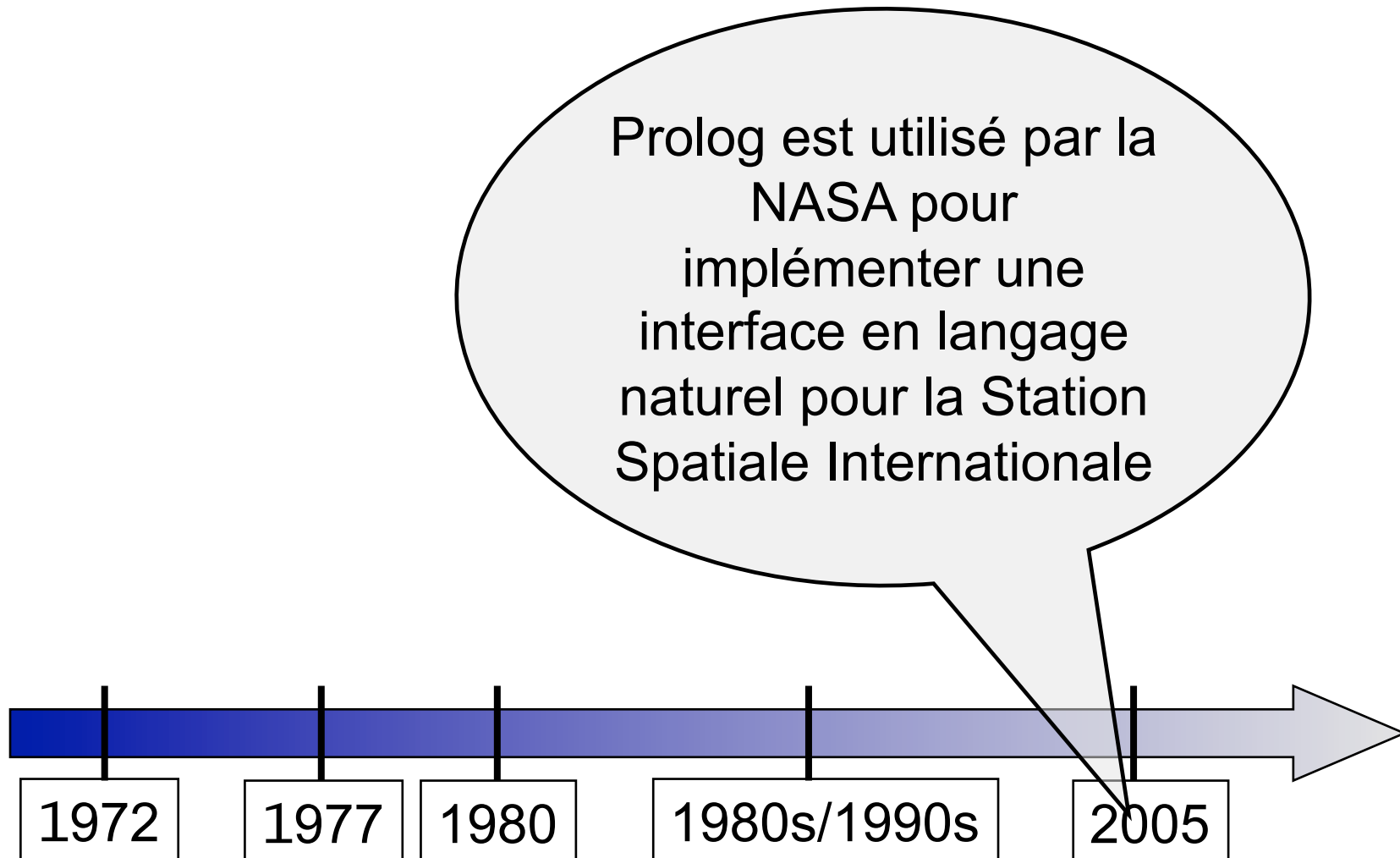
# Histoire de Prolog



# Histoire de Prolog



# Histoire de Prolog



# Idée de base de Prolog

---

- Décrire la situation considérée
- Poser une question
- Prolog déduit logiquement de nouveaux faits sur la situation que nous avons décrite
- Prolog nous rend ses déductions comme réponses

# Conséquences

---

- Penser de manière déclarative, pas de manière procédurale
  - Challenge
  - Nécessite un état d'esprit différent
- Langage de haut niveau
  - Pas aussi efficace que le C (par ex.)
  - Pertinent pour du prototypage rapide
  - Utile pour un grand nombre d'applications en intelligence artificielle

# Base de connaissances 1

```
femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.
```



# Base de connaissances 1

femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.

?-

# Base de connaissances 1

femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.

?- femme(mia).

# Base de connaissances 1

```
femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.
```

```
?- femme(mia).  
yes  
?-
```

# Base de connaissances 1

```
femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.
```

```
?- femme(mia).  
yes  
?- joueAirGuitar(jody).
```

# Base de connaissances 1

```
femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.
```

```
?- femme(mia).  
yes  
?- joueAirGuitar(jody).  
yes  
?-
```

# Base de connaissances 1

```
femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.
```

```
?- femme(mia).  
yes  
?- joueAirGuitar(jody).  
yes  
?- joueAirGuitar(mia).  
no
```

# Base de connaissances 1

femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.

?- tatouee(jody).

# Base de connaissances 1

```
femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.
```

```
?- tatouee(jody).  
no  
?-
```



# Base de connaissances 1

```
femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.
```

```
?- tatouee(jody).  
ERROR: predicate tatouee/1 not defined.  
?-
```

# Base de connaissances 1

femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.

?- fete.

# Base de connaissances 1

femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.

?- fete.  
yes  
?-

# Base de connaissances 1

femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.

?- concertDeRock.

# Base de connaissances 1


femme(mia).  
femme(jody).  
femme(yolanda).  
joueAirGuitar(jody).  
fete.

?- concertDeRock.  
no  
?-

# Base de connaissances 2

```
cool(yolanda).  
ecoute2LaMusique (mia).  
ecoute2LaMusique (yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique (mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique (yolanda).
```

# Base de connaissances 2

cool(yolanda).  fait  
ecoute2LaMusique (mia).  
ecoute2LaMusique (yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique (mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique (yolanda).

# Base de connaissances 2

cool(yolanda).

fait

ecoute2LaMusique(mia).

fait

ecoute2LaMusique(yolanda):- cool(yolanda).

joueAirGuitar(mia):- ecoute2LaMusique(mia).

joueAirGuitar(yolanda):- ecoute2LaMusique(yolanda).



# Base de connaissances 2

cool(yolanda).

fait

ecoute2LaMusique(mia).

fait

ecoute2LaMusique(yolanda):- cool(yolanda).

règle

joueAirGuitar(mia):- ecoute2LaMusique(mia).

joueAirGuitar(yolanda):- ecoute2LaMusique(yolanda).

# Base de connaissances 2

The diagram shows a list of Prolog facts and rules. Callout boxes labeled 'fait' (fact) point to the first two lines, and callout boxes labeled 'règle' (rule) point to the last two lines.

```
cool(yolanda).  
ecoute2LaMusique(mia).  
ecoute2LaMusique(yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique(mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique(yolanda).
```

# Base de connaissances 2

The diagram shows a list of Prolog clauses within a light gray box. Callout boxes with pointers identify specific parts of the clauses:

- `cool(yolanda).` is followed by a callout box labeled **fait**.
- `ecoute2LaMusique(mia,` is followed by a callout box labeled **fait**.
- `ecoute2LaMusique(yolanda):- cool(yolanda),` is followed by a callout box labeled **règle**.
- `joueAirGuitar(mia):- ecoute2LaMusique(mia),` is followed by a callout box labeled **règle**.
- `joueAirGuitar(yolanda):- ecoute2LaMusique(yolanda),` is followed by a callout box labeled **règle**.

```
cool(yolanda).
ecoute2LaMusique(mia,
ecoute2LaMusique(yolanda):- cool(yolanda),
joueAirGuitar(mia):- ecoute2LaMusique(mia),
joueAirGuitar(yolanda):- ecoute2LaMusique(yolanda),
```

# Base de connaissances 2

```
cool(yolanda).  
ecoute2LaMusique (mia).  
ecoute2LaMusique (yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique (mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique (yolanda).
```

tête

corps

# Base de connaissances 2

```
cool(yolanda).  
ecoute2LaMusique (mia).  
ecoute2LaMusique (yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique (mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique (yolanda).
```

?-

# Base de connaissances 2

```
cool(yolanda).  
ecoute2LaMusique (mia).  
ecoute2LaMusique (yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique (mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique (yolanda).
```

```
?- joueAirGuitar(mia).  
yes  
?-
```

# Base de connaissances 2

```
cool(yolanda).  
ecoute2LaMusique (mia).  
ecoute2LaMusique (yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique (mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique (yolanda).
```

```
?- joueAirGuitar (mia).  
yes  
?- joueAirGuitar (yolanda).  
yes
```

# Clauses

```
cool(yolanda).  
ecoute2LaMusique (mia).  
ecoute2LaMusique (yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique (mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique (yolanda).
```

*Il y a cinq clauses dans cette base de connaissances :*

*deux faits, et trois règles.*

*La fin d'une clause est marquée par un point.*



# Prédicats

```
cool(yolanda).  
ecoute2LaMusique (mia).  
ecoute2LaMusique (yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique (mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique (yolanda).
```

*Il y a trois **prédicats** dans cette base de connaissances :*

*cool, ecoute2lamusique, and joueAirGuitar*

# Base de connaissances 3

```
cool(vincent).  
ecoute2LaMusique(butch).  
joueAirGuitar(vincent):- ecoute2LaMusique(vincent), cool(vincent).  
joueAirGuitar(butch):- cool(butch).  
joueAirGuitar(butch):- ecoute2LaMusique(butch).
```

# Expression de la conjonction

```
cool(vincent).  
ecoute2LaMusique(butch).  
joueAirGuitar(vincent):- ecoute2LaMusique(vincent), cool(vincent).  
joueAirGuitar(butch):- cool(butch).  
joueAirGuitar(butch):- ecoute2LaMusique(butch).
```

*La virgule "," indique une conjonction en Prolog*

# Base de connaissances 3

```
cool(vincent).  
ecoute2LaMusique(butch).  
joueAirGuitar(vincent):- ecoute2LaMusique(vincent), cool(vincent).  
joueAirGuitar(butch):- cool(butch).  
joueAirGuitar(butch):- ecoute2LaMusique(butch).
```

```
?- joueAirGuitar(vincent).  
no  
?-
```

# Base de connaissances 3

```
cool(vincent).
ecoute2LaMusique(butch).
joueAirGuitar(vincent):- ecoute2LaMusique(vincent), cool(vincent).
joueAirGuitar(butch):- cool(butch).
joueAirGuitar(butch):- ecoute2LaMusique(butch).
```

```
?- playsAirGuitar(butch).
yes
?-
```

# Expression de la Disjonction

```
cool(vincent).  
ecoute2LaMusique(butch).  
joueAirGuitar(vincent):- ecoute2LaMusique(vincent), cool(vincent).  
joueAirGuitar(butch):- cool(butch).  
joueAirGuitar(butch):- ecoute2LaMusique(butch).
```

```
cool(vincent).  
ecoute2LaMusique(butch).  
joueAirGuitar(vincent):- listens2music(vincent), happy(vincent).  
joueAirGuitar(butch):- cool(butch); ecoute2LaMusique(butch).
```

# Prolog et Logique

---

- Prolog a clairement un lien avec la logique
- Opérateurs
  - Implication  $:-$
  - Conjonction  $,$
  - Disjonction  $;$
- Utilisation du modus ponens
- Négation

# Base de connaissances 4

femme(mia).

femme(jody).

femme(yolanda).

aime(vincent, mia).

aime(marsellus, mia).

aime(mon\_chou, lapin).

aime(lapin, mon\_chou).



# Variables Prolog

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- woman(X).
```

# Instanciation de Variables

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- femme(X).  
X=mia
```

# Réponses Alternatives

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- femme(X).  
X=mia;
```

# Réponses Alternatives

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- femme(X).  
X=mia;  
X=jody
```

# Réponses Alternatives

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- femme(X).  
X=mia;  
X=jody;  
X=yolanda
```

# Réponses Alternatives

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- femme(X).  
X=mia;  
X=jody;  
X=yolanda;  
no
```

# Base de connaissances 4

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- aime(marsellus,X), femme(X).
```

# Base de connaissances 4

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- aime(marsellus,X), femme(X).  
X=mia  
yes  
?-
```



# Base de connaissances 4

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- aime(pumpkin,X), femme(X).
```

# Base de connaissances 4

```
femme(mia).  
femme(jody).  
femme(yolanda).
```

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).
```

```
?- aime(pumpkin,X), femme(X).  
no  
?-
```

# Base de connaissances 5

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).  
  
jaloux(X,Y):- aime(X,Z), aime(Y,Z).
```

# Base de connaissances 5

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).  
  
jaloux(X,Y):- aime(X,Z), aime(Y,Z).
```

```
?- jaloux(marsellus,W).
```

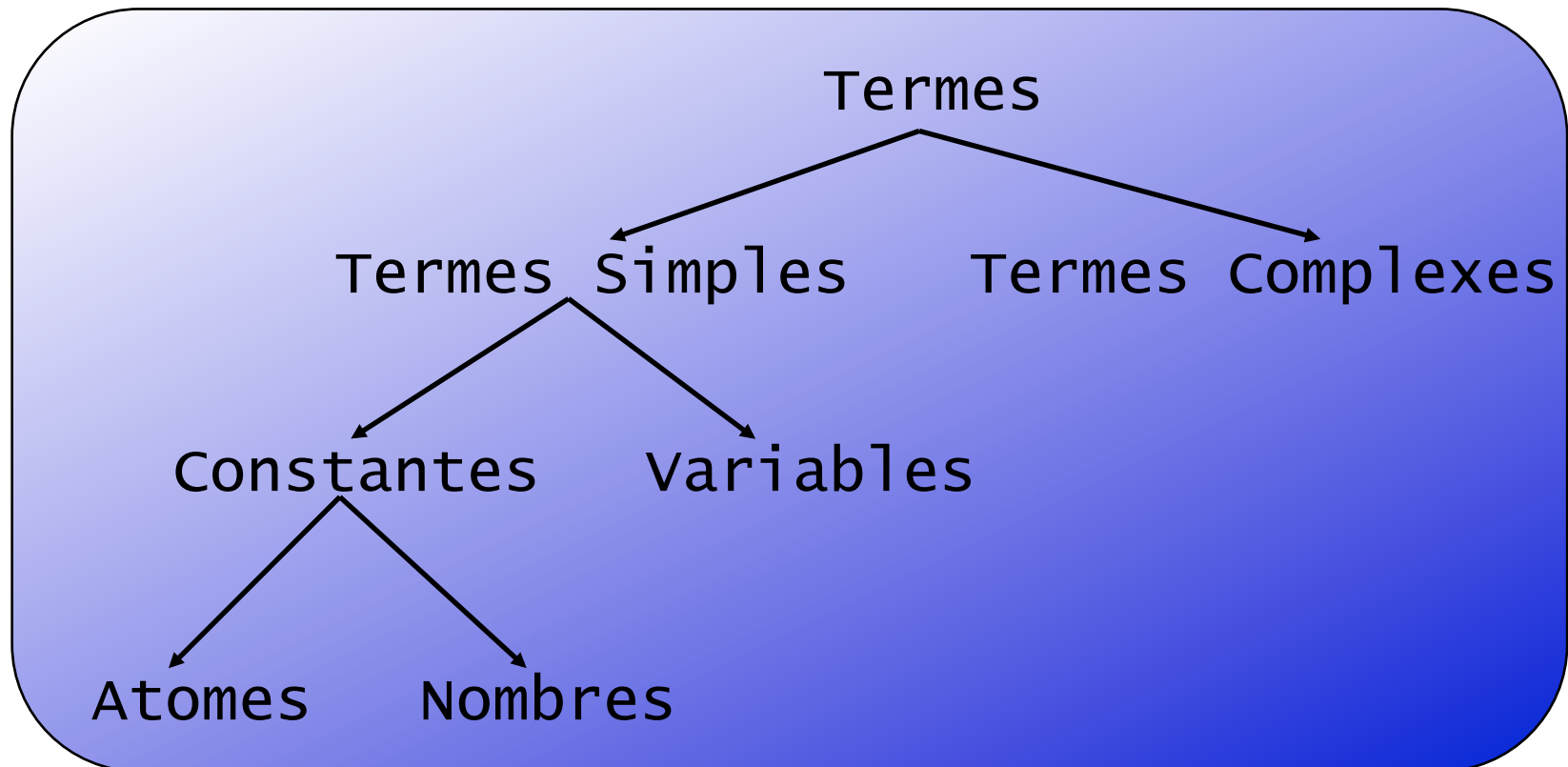
# Base de connaissances 5

```
aime(vincent, mia).  
aime(marsellus, mia).  
aime(mon_chou, lapin).  
aime(lapin, mon_chou).  
  
jaloux(X,Y):- aime(X,Z), aime(Y,Z).
```

```
?- jaloux(marsellus,W).  
W=vincent  
?-
```

# Syntaxe de Prolog

- À partir de quels éléments sont construits les faits, règles et requêtes ?



# Atomes

- Une séquence de caractères de lettres minuscules ou majuscules, chiffres, ou souligné, commençant par une minuscule
  - *Exemples:* **butch**, **big\_kahun\_burger**, **joueGuitare**
- Une séquence arbitraire de caractères entourés de simples quotes
  - *Exemples:* **'Vincent'**, **'Five dollar shake'**, **'@\$%'**
- Une sequence de caractères spéciaux
  - *Exemples:* **:**, **,**, **;**, **.**, **:-**

# Nombres

---

- Entiers : 12, -34, 22342
- Flottants : 34573.3234



# Variables

---

- Une séquence de caractères de lettres minuscules ou majuscules, chiffres, ou souligné, commençant par une majuscule ou un souligné
- Exemples:

**X, Y, Variable, Vincent, \_tag**

# Termes Complexes

---

- Atomes, nombres et variables sont les briques pour construire des termes complexes
- Les Terms complexes sont construits par un foncteur directement suivi par une séquence d'arguments
- Les arguments sont entre parathèses, séparés par des virgules
- Le foncteur doit être un atome

# Exemples de termes complexes

---

- Exemples déjà rencontrés:
  - joueAirGuitar(jody)
  - aime(vincent, mia)
  - jaloux(marsellus, W)
- Termes complexes à l'intérieur de termes complex :
  - cache(X,pere(pere(pere(butch))))

# Arité

- Le nombre d'arguments d'un terme complexe est appelé son arité
- Exemples:

**femme(mia)** est un terme d'arité 1  
**aime(vincent,mia)** a comme arité 2  
**pere(father(butch))** arité 1

# L'arité est importante

---

- En Prolog, il est possible de définir deux prédicats avec le même foncteur mais avec une arité différente
- Prolog les traitera comme deux prédicats différents
- Dans la doc. Prolog l'arité d'un prédicat is généralement indiqué par le suffixe "/" suivi d'un nombre représentant l'arité

# Exemple d'Arité

```
cool(yolanda).  
ecoute2LaMusique(mia).  
ecoute2LaMusique(yolanda):- cool(yolanda).  
joueAirGuitar(mia):- ecoute2LaMusique(mia).  
joueAirGuitar(yolanda):- ecoute2LaMusique(yolanda).
```

- Cette base de connaissances définit :
  - cool/1
  - ecoute2LaMusique/1
  - joueAirGuitar/1

# Exercices

# Résumé de cette séance

---

- Exemples simple de programmes en Prolog
- Les trois constructions de base présentés :
  - Les Faits
  - Les Règles
  - Les Requêtes
- Autre concepts abordés, comme
  - le role de la logique
  - L'unification avec l'aide des variables
- Définition des briques de base de Prolog :  
termes, atomes, et variables



# Prochaine séance

---

- **L'unification** dans Prolog
- Stratégie de recherche de Prolog