Initiation à l'Intelligence Artificielle

Philippe Beaune, *Gauthier Picard*, Laurent Vercouter

École Nationale Supérieure des Mines de Saint-Étienne

gauthier.picard@emse.fr

Pôle XXI 2010-2011

Sommaire

1. Introduction
2. Logique des propositions
3. Logique des prédicats
4. Prolog

Objectifs et déroulement

Objectifs de ce cours

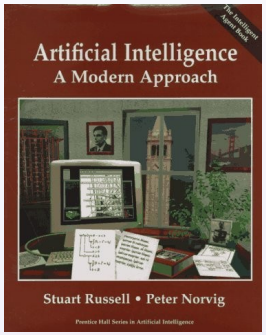
- Avoir un aperçu (forcément partiel) de l'Intelligence Artificielle
- Être capable de découvrir d'autres champs de l'Intelligence Artificielle

Déroulement

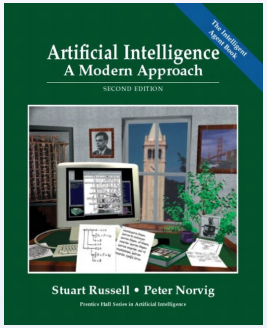
- 1 cours de 1h30
- 3 séances de TP en Prolog

Le livre de référence

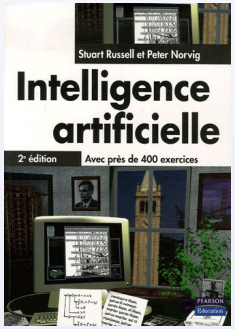
Artificial Intelligence : A Modern Approach, Stuart Russell & Peter Norvig



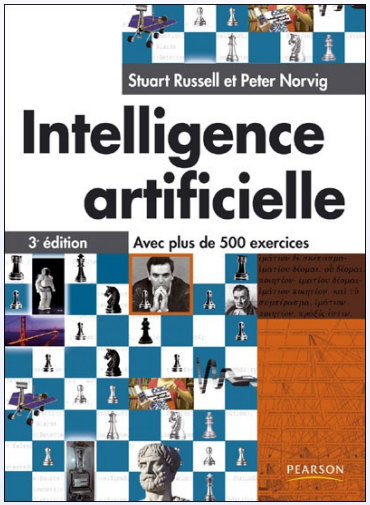
1st edition
1995

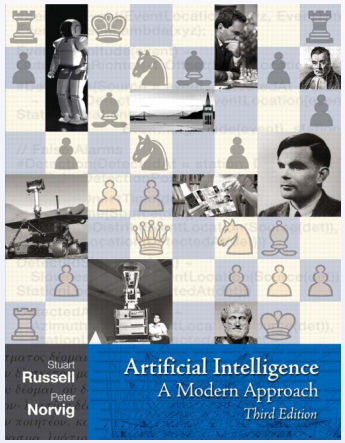


2nd edition
2003



2006



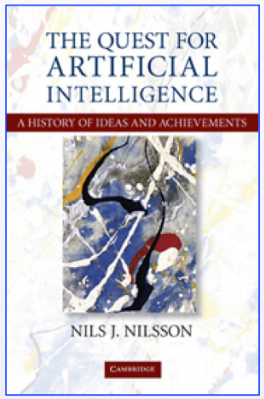


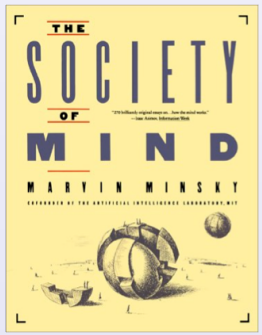
3rd edition
2010

- `http://aima.cs.berkeley.edu/`
- Plein de ressources sur le site du livre

Quelques autres ressources

- AFIA : http://www.afia-france.org/
- Revue d'IA : http://ria.revuesonline.com/
- AAAI : http://www.aaai.org/
- AI Magazine : http://www.aaai.org/Magazine
- ACM SIGART : http://www.sigart.org/
- Nils J. Nilsson : http://ai.stanford.edu/~nilsson/
- John McCarthy : http://www-formal.stanford.edu/jmc/
- Marvin Minsky : http://web.media.mit.edu/~minsky/
- JAIR : http://www.jair.org/
- IJCAI : http://www.ijcai.org/
- AI Journal : http://www.ida.liu.se/ext/aijd/
- ECCAI, ECAI : http://www.eccai.org/
- AI/Alife Howto : http://zhar.net/howto/
- ETAI : http://www.etaij.org/
- …liste non exhaustive, bien évidemment





Logique des propositions (1)

- On va s'intéresser à des énoncés soit vrais soit faux, et aux relations entre ces énoncés, avec une présentation *simplifiée*

Syntaxe

- Vocabulaire
  - Chaînes de caractères représentant les atomes
  - Connecteurs : $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$, $\neg$, ...
  - Parenthésage : $($, $)$
- Règles de construction des formules bien formées (*fbf*)
  - Un atome est une *fbf*
  - Si F est une *fbf*, alors (F) est une *fbf*
  - Si G est une *fbf*, alors $\neg G$ est une *fbf*
  - Si F et G sont deux *fbf*, alors $F \vee G$, $F \wedge G$, $F \Rightarrow G$, $F \Leftrightarrow G$ et $F \Rightarrow \neg G$ sont des *fbf*
- (Règles de priorité entre connecteurs)

Logique des propositions (2)

Composition

- La valeur de vérité d'une *fbf* dépend uniquement des connecteurs et de la valeur de vérité de chaque atome

Tables de vérités

F	G	$\neg F$	$F \wedge G$	$F \vee G$	$F \Rightarrow G$	$F \Leftrightarrow G$
faux	faux	vrai	faux	faux	vrai	vrai
faux	vrai	vrai	faux	vrai	vrai	faux
vrai	faux	faux	faux	vrai	faux	faux
vrai	vrai	faux	vrai	vrai	vrai	vrai

Interprétation

Fonction $\mathcal{I}$ de {atomes} vers {vrai, faux}

Logique des propositions (3)

Quelques formules

$$\begin{aligned} P \Rightarrow Q &\equiv (\neg P \wedge Q) \\ P \Leftrightarrow Q &\equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P) \equiv (P \wedge Q) \vee (\neg P \wedge \neg Q) \\ \neg\neg P &\equiv P \end{aligned}$$

- Loi de de Morgan :

$$\begin{aligned} \neg(P \wedge Q) &\equiv \neg P \vee \neg Q \\ \neg(P \vee Q) &\equiv \neg P \wedge \neg Q \end{aligned}$$

- Commutativité, associativité, distributivité, ...
- Contradiction : $P \wedge \neg P \equiv FAUX$
- Tiers-exclus : $P \vee \neg P \equiv VRAI$
- Absorption :

$$\begin{aligned} P \vee (P \wedge Q) &\equiv P \\ P \wedge (P \vee Q) &\equiv P \end{aligned}$$

Logique des propositions (4)

Quelques règles d'inférence

Modus Ponens ou élimination de $\Rightarrow$:

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

Elimination du $\wedge$:

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

Introduction du $\wedge$:

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

Introduction du $\vee$:

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

Elimination de la double nég. :

$$\frac{\neg\neg\alpha}{\alpha}$$

Résolution unitaire :

$$\frac{\alpha \wedge \beta, \quad \neg\beta}{\alpha}$$

Résolution :

$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma} \text{ ou } \frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Logique des propositions (5)

Définitions

- Un modèle d'une *fbf* (resp. d'un ensemble F de *fbf*) est une interprétation qui rend vraie cette *fbf* (resp. chaque *fbf* de F)
- S'il existe un modèle m d'une *fbf* (resp. d'un ensemble de *fbf*), on dit que la *fbf* (resp. l'ensemble de *fbf*) est satisfiable, sinon elle (resp. il) est inconsistant
- Si une *fbf* (resp. un ensemble de *fbf*) est satisfiable pour tout modèle, on dit qu'elle (resp. il) est valide.
- Une *fbf* A est fbf conséquence logique d'un ensemble F de *fbf* si tout modèle de F est modèle de A, et on note $F \models A$
  - Exemples : $\{A, B\} \models A$, $\{A, B\} \models A \wedge B$, $\{A, A \Rightarrow B\} \models B$
- Si A est valide, on note $\models A$

Théorème de déduction

$A \models B$ ssi $\models (A \Rightarrow B)$

Réfutation

$A \models B$ ssi $(A \wedge \neg B)$ est inconsistant

Logique des propositions (6)

Algorithmes d'inférence

- Soit KB un ensemble de *fbf*, F une *fbf*, et i un algorithme d'inférence :
  - Si F est dérivée de KB par i alors on note : $KB \vdash_i F$
- Si i dérive seulement des conséquences logiques alors i est dit sain :
  - Si $KB \vdash_i F$ alors $KB \models F$
- Si toute conséquence logique peut être dérivée par i, alors i est dit complet :
  - Si $KB \models F$ alors $KB \vdash_i F$

➥ *Méthode avec tables de vérité ? Si n atomes alors 2^n lignes !*

Logique des propositions (7)

Forme normale conjonctive

- Un littéral est un atome (A) ou la négation d'un atome ($\neg A$)
- Une *fbf* est mise sous forme normale conjonctive (*FNC*) si elle est sous la forme $F_1 \wedge F_2 \wedge \ldots \wedge F_n$ où chaque F_i est une disjonction de littéraux
- Les F_i sont des clauses
- La forme clausale est l'ensemble des clauses
- Toute *fbf* peut être mise sous forme normale conjonctive (et donc clausale) :
  1. Éliminer $\Leftrightarrow$ puis $\Rightarrow$
  2. Lois de de Morgan
  3. Éliminer les doubles négations
  4. Appliquer les règles de distributivité

Logique des propositions (8)

Principe de réfutation

Pour montrer que $F_1, F_2, \ldots, F_n \vDash C$, il faut et il suffit de montrer que la formule de réfutation $F_1 \wedge F_2 \wedge \ldots \wedge F_n \wedge \neg C$ est inconsistante

Règle de résolution

Soient C_1 et C_2 deux clauses d'une formule F, s'il existe un atome A tel que $A \in C_1$ et $\neg A \in C_2$ alors la clause $\left(C_1 \setminus \{A\}\right) \cup \left(C_2 \setminus \{\neg A\}\right)$, est dite résolvante de C_1 et C_2, et elle est conséquence logique de F

Logique des propositions (9)

Méthode de résolution (Robinson, 1965)

- Pour montrer qu'une formule F est inconsistante, il faut et il suffit de produire la clause vide par résolution à partir de l'ensemble des clauses issues de F mise sous forme clausale
- D'où la méthode, pour montrer que $KB \models A$:
  1. Construire la formule de réfutation
  2. La mettre sous forme clausale : F
  3. Construire une résolvante (tant que c'est possible) et l'ajouter à F jusqu'à obtenir la clause vide
  4. Si la clause vide est obtenue alors $KB \models A$ sinon $KB \not\models A$

Logique des propositions (10)

Exemple sur le modus ponens

- Montrer que $\{A, A \Rightarrow B\} \models B$
- Formule de réfutation : $A \wedge (A \Rightarrow B) \wedge \neg B$
- Forme clausale : $\{A, (\neg A \vee B), \neg B\}$
- 1e solution :
  - résolvante de A et $(\neg A \vee B)$: B, puis
  - résolvante de B et $\neg B$: $\varnothing$
- 2e solution :
  - résolvante de $(\neg A \wedge B)$ et $\neg B$: $\neg A$, puis
  - résolvante de A et $\neg A$: $\varnothing$

- *Dans quel ordre prendre les clauses pour obtenir les résolvantes successives ?*
- *Plusieurs algo qui garantissent la complétude*
- *NP-complet sauf classe P pour certains cas dont les clauses de Horn*

Une énigme à résoudre

Vous êtes perdus sur une piste dans le désert. Vous arrivez à une bifurcation. Chacune des deux pistes est gardée par un sphinx que vous pouvez interroger. Les pistes peuvent soit conduire à une oasis, soit se perdre dans le désert profond (au mieux, elle conduisent toutes à une oasis, au pire elles se perdent toutes les deux).

1. *Le sphinx de droite vous répond : « Une au moins des deux pistes conduit à une oasis. »*
2. *Le sphinx de gauche vous répond : « La piste de droite se perd dans le désert. »*
3. *Vous savez que les sphinx disent tous les deux la vérité, ou bien mentent tous les deux.*

tiré de Notes de cours de Jérôme Champavert :

`http://www.grappa.univ-lille3.fr/~champavere/Enseignement/0607/l2miashs/ia/logique.pdf`

Logique des prédicats (1)

Limitation de la logique des propositions

- La logique des propositions a un pouvoir d'expression limité
- Comment exprimer que si *Sylvain* est fils de *Philippe*, et *Philippe* fils de *Jean*, alors *Jean* est grand-père de *Sylvain*, ainsi que de *Marion*, fille aussi de *Philippe*, et que cela est vrai dans plein d'autres cas, sans avoir à énumérer tous les liens de parentés pour toutes les familles ?

Logique des prédicats (1)

Limitation de la logique des propositions

- La logique des propositions a un pouvoir d'expression limité
- Comment exprimer que si *Sylvain* est fils de *Philippe*, et *Philippe* fils de *Jean*, alors *Jean* est grand-père de *Sylvain*, ainsi que de *Marion*, fille aussi de *Philippe*, et que cela est vrai dans plein d'autres cas, sans avoir à énumérer tous les liens de parentés pour toutes les familles ?

Introduction de prédicats et de variables

$Fils(x,y) \wedge Fils(y,z) \Leftrightarrow Grand\_pere(z,x)$

$\forall x, Gentil(x) \wedge Beau(x)$ `/* tt le monde est gentil et beau */`

$\exists x, Fatigue(x)$ `/* quelqu'un est fatigué */`

Logique des prédicats (2)

Syntaxe

- Termes : constantes (majuscules : A), variables (minuscules : x), fonctions (minuscules : $f(A, X, f(g(e)))$)
- Formules atomiques
  - prédicats dont les arguments sont des termes (majuscules : $P(x, t, f(u, g(s), R), Z)$)
  - terme = terme
- Formules bien formées (*fbf*)
  - Une formule atomique est une *fbf*
  - Si F est une *fbf*, (F) et $\neg F$ sont des *fbf*
  - Si F et G sont des *fbf* : $F \wedge G$, $F \vee G$, $F \Rightarrow G$ et $F \Leftrightarrow G$ sont des *fbf*
  - Si F est une *fbf* et x une variable :
    - $\forall x.F$ est une *fbf*
    - $\exists x.F$ est une *fbf*

Logique des prédicats (3)

Les quantificateurs

- L'ordre peut être important :

$$\forall x.(\exists y.Aime(x,y)) \text{ vs. } \exists x.(\forall y.Aime(x,y))$$

- Loi de de Morgan :

$$\begin{aligned} \neg\forall x.F &\equiv \exists x.\neg F \\ \neg\exists x.F &\equiv \forall x.\neg F \\ \forall x.F &\equiv \neg\exists x.\neg F \\ \exists x.F &\equiv \neg\forall x.\neg F \end{aligned}$$

- Une variable est dite libre dans F si toutes ses occurrences dans F sont hors de portée des quantificateurs, sinon elle est liée
- Une formule est fermée (ou close) si elle ne contient aucune variable libre, sinon elle est ouverte

Logique des prédicats (4)

Interprétation

On se donne :

- un domaine de valeurs pour les constantes
- une application qui donne une valeur à chaque variable
- une application qui associe à toute fonction d'arité n et à tout n-uplet de termes une valeur dans le domaine de valeurs
- une application qui associe à tout prédicat d'arité n et à tout n-uplet de termes une valeur dans {vrai,faux}

- $\forall x.P$ est vrai ssi P est vrai pour toute interprétation de x
- $\exists x.P$ est vrai ssi P est vrai pour au moins une interprétation de x

Logique des prédicats (5)

Forme de skolem

- Une formule F est sous forme prenex ssi elle est sous la forme $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n . A$ où Q_i est un quantificateur et A ne contient aucun quantificateur
- Pour toute formule F il existe une formule F' sous forme prenex telle que $F \equiv F'$
- Soit F une formule sous forme prenex de la forme

$$\forall x_1 \ldots \forall x_i \exists x_{i+1} Q_{i+2} x_{i+2} \ldots Q_n x_n . A$$

et f un nouveau symbole d'une fonction i-aire, la formule F'

$$\forall x_1 \ldots \forall x_i Q_{i+2} x_{i+2} \ldots Q_n x_n . A\{x_{i+1}/f(x_1, \ldots, x_i)\}$$

est la skolémisation partielle de F et F est satisfiable ssi F' l'est

- Si une formule prenex F a n quantificateurs $\exists$, la forme de skolem F' de F est obtenue par n applications de la skolémisation partielle et F est satisfiable ssi F' l'est.

Logique des prédicats (6)

Mise sous forme clausale (principales étapes)

1. Mise sous FNC comme en logique des propositions
2. Mise sous forme prenex
3. Skolémisation
4. Mise sous forme clausale : élimination des $\forall$

Substitution

Une substitution est application σ de l'ensemble des variables vers l'ensemble des termes. Par extension on note $\sigma(F)$ la formule F dans laquelle on a appliqué σ

Unification

F_1 et F_2 sont unifiables s'il existe une substitution σ telle que $\sigma(F_1) = \sigma(F_2)$
σ est alors appelée unificateur de F_1 et F_2

Logique des prédicats (7)

Résolution

Comme en logique des propositions mais en passant par l'unification :

- Soient F_1 et F_2 deux clauses : elles sont résolvables ssi elles contiennent une paire opposée de formules atomiques $P(x_1, \ldots, x_n)$ et $\neg P(x'_1, \ldots, x'_n)$ et si elles peuvent être unifiées par un unificateur σ
- La résolvante est alors $\sigma(F_1 \setminus \{P\}) \cup \sigma(F_2 \setminus \{\neg P\})$
- Exemple : $F(x) \wedge G(Toto)$ et $\neg F(y) \wedge G(z)$

Logique des prédicats (7)

Résolution

Comme en logique des propositions mais en passant par l'unification :

- Soient F_1 et F_2 deux clauses : elles sont résolvables ssi elles contiennent une paire opposée de formules atomiques $P(x_1, \ldots, x_n)$ et $\neg P(x'_1, \ldots, x'_n)$ et si elles peuvent être unifiées par un unificateur σ
- La résolvante est alors $\sigma(F_1 \setminus \{P\}) \cup \sigma(F_2 \setminus \{\neg P\})$
- Exemple : $F(x) \wedge G(Toto)$ et $\neg F(y) \wedge G(z)$

➥ La résolution est saine et complète (au sens de la réfutation)

Logique des prédicats (8)

Décidabilité

- La logique des propositions est décidable (on peut montrer en un nombre fini d'opérations qu'une formule est valide ou contradictoire)
- La logique des prédicats est indécidable (Gödel, 1931)
- La logique des prédicats est semi-décidable : on peut montrer en un nombre fini d'opérations si une formule est valide mais pas si elle est contradictoire
- La logique des prédicats réduite aux clauses de Horn est décidable (cf. Prolog)

Prolog (1)

Clauses de Horn

- Disjonction de littéraux dont un seul au plus est positif
  - ex. : $p(X) \Leftarrow q(a) \wedge r(Z) \wedge s(Z) \wedge t(toto)$
- Clause avec exactement un littéral positif est dite clause définie

Constituants de Prolog

- Base de règles : ensemble de clauses définies non réduites à un littéral positif
- Base de faits : ensemble de littéraux positifs
  - ex. : $\{p(truc), r(machin), s(X, Y)\}$
- Question : clause négative
  - ex. : $q(X, toto) \wedge w(truc)$?

Prolog (2)

Moteur d'inférence

- Ordre 1 : à base de la logique des prédicats
- Chaînage arrière : raisonnement guidé par les buts
- Principe de résolution (par SLD-*resolution*) avec stratégie en profondeur d'abord
- Régime par tentatives : *backtrack* si échec
- Non-monotone
- Négation par l'échec (*SLDNF-resolution*)
  - $not(p)$ réussit si p n'est pas démontrable

Prolog (3)

SLD-resolution

- Chaque étape de résolution doit prendre une clause négative (initialement, la question) et une clause définie (prise dans le programme)
- SLD-resolution (Linear resolution for Definite clauses with Selection function) :
  - Prendre un littéral de la clause négative (lequel ?) et tenter une unification avec le littéral positif d'une clause définie (**unificateur le plus général**)
  - Si une telle unification est trouvée alors remplacer le littéral choisi de la clause négative par les éventuels littéraux négatifs de la clause définie qui a réussi l'unification (*Linear*)
  - Si l'unification échoue, reporter cet échec à l'unification de niveau supérieur
  - Si la clause négative est vide : succès !

Prolog (4)

Stratégie de Prolog : profondeur d'abord

- Choix du 1er littéral de la clause négative
- Backtrack aux feuilles de l'arbre
- Choix des clauses définies dans l'ordre d'écriture du programme
- Conséquences :
  - Une stratégie en profondeur est efficace (en largeur ce serait gourmand en taille mémoire)
  - Mais il y a un risque de boucle infinie (attention à l'ordre d'écriture des règles) : donc Prolog n'est pas complet (même si la *SLD-resolution* est complète pour la réfutation)

Prolog (5)

Syntaxe

- Constantes : entiers, flottants, ou chaînes de caractères commençant par une minuscule
- Variables : chaînes de caractères commençant par une majuscule
- Prédicats :
  - Nom commençant par une minuscule
  - Arguments pouvant être des constantes, des variables et des prédicats
- Listes :

```
[a,b,c] = [a | [b | c]] = [a,b |[c]]
```

- Faits :

```
toto(truc, machin).
```

- Règles :

```
titi(X) :- toto(X,machin),
bidule(foo).
```

Prolog (6)

Exemple familial

```
homme(jean).
homme(pierre).
homme(luc).
femme(marie).
femme(anne).
femme(marion).
pere(Papa,Enfant) :- parent(Papa,Enfant), homme(Papa).
mere(X,Y) :- parent(X,Y), femme(X).
grand_pere(X,Y) :- pere(X,Z), parent(Z,Y).
grand_mere(X,Y) :- mere(X,Z), parent(Z,Y).
frere(X,Y) :- parent(Z,X), parent(Z,Y), X\=Y, homme(X).
oncle(O,N) :- frere(O,X), parent(X,N).
parent(jean,pierre).
parent(jean,marie).
parent(anne,marion).
parent(luc,jean).
parent(luc,anne).

:- oncle(X,Y) ?
```

Prolog (7)

Un pas en avant

- Être une liste ou pas :

```
list(X) :- X=[Y].
```

ce qui peut se simplifier en :

```
list([Y]).
```

ce qui peut encore se simplifier en :

```
list([_]).
```

hé ! on oublie les listes vides, il faut ajouter :

```
list([]).
```

- Récursivité :
  - Être élément ou ne pas être :

```
elem(X,[X|_]).
elem(X,[_|Y]) :- elem(X,Y).
```

  - Être premier élément, être dernier élément,… à vous de jouer…
  - Concaténation de 2 listes :

```
concat([],X,X).
concat([X|S],Y,[X|R]) :- concat(S,Y,R).
```

Prolog (8)

Divers

- Affectation :

```
X is 2+3
```

- Comparaisons :
  - X = Y réussit s'il unifie les 2 termes ou s'ils sont identiques
  - X == Y réussit si les 2 termes sont équivalents (sans unification)
  - X =@= Y réussit si les 2 termes sont structurellement équivalents (sans unification)
  - …

Prolog (9)

Toujours penser déclaratif, mais néanmoins...

- `fail` : échoue toujours
- `true` : réussit toujours
- `!` (cut) : bloque le backtracking

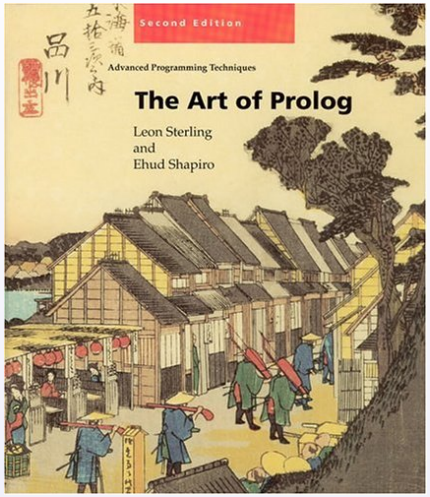
```
not(X) :- X , ! , fail.
not(X).
```

… ne pas recourir trop souvent au *cut* !

Divers

- Certains Prolog vont plus loin, notamment CSP
- Compilation (WAM, 1983), standard ISO (1995), …

La suite en T.P. …



- *The art of Prolog*
  L. Sterling & E. Shapiro,
  1994 (VF de 1990 chez Masson)
- *Logic, Programming and Prolog* (2ed)
  Ulf Nilsson and Jan Maluszynski,
  2000
  `http://www.ida.liu.se/~ulfni/lpp/`