IA41

Concepts fondamentaux en Intelligence Artificielle et langages dédiés

CM #1

Introduction à l'IA : définitions, genèse, domaines, représentation des connaissances et langages dédiés

Fabrice LAURI

utbm

Organisation d'IA41

Enseignants

- Fabrice Lauri fabrice.lauri@utbm.fr
- Olivier Grunder
- Marina Piercy
- Claude Renaud

Horaires

- Cours : Jeudi 8h-10h
- TD 1 : Jeudi 10h15-12h15 TD 2 : Mardi 10h15-12h15
- TD 3 : Vendredi 8h-10h TD 4 : Lundi 8h-10h
- TD 5 : Vendredi 10h15-12h15
- TP 1 : Mercredi 10h15-12h15 TP 2 : Mardi 14h-16h
- TP 3 : Vendredi 14h-16h TP 4 : Mardi 16h15-18h15
- TP 5 : Lundi 10h15-12h15

Plan du cours

1. Définitions, genèse, problèmes typiques et domaines de recherche de l'IA

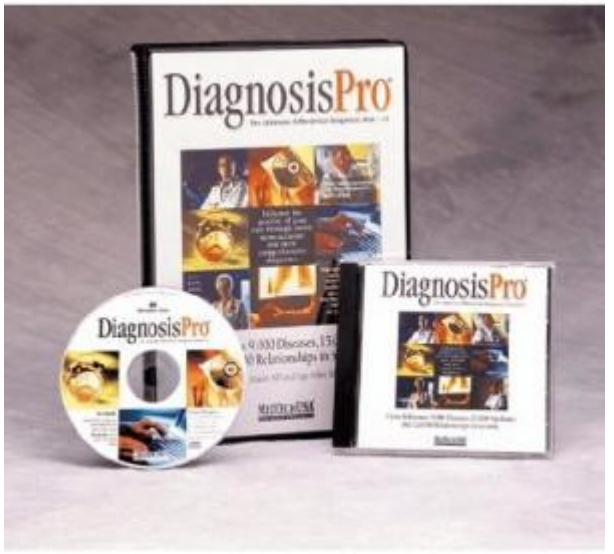
2. Démarche générale pour la résolution de problèmes
   Modes de représentation des connaissances
   Récursivité

3. Introduction aux langages dédiés à la résolution de problèmes d'IA : PROLOG et LISP

Définitions, genèse, problèmes typiques et domaines de recherche de l'Intelligence Artificielle

Question

Que doit posséder chacun des systèmes suivants :



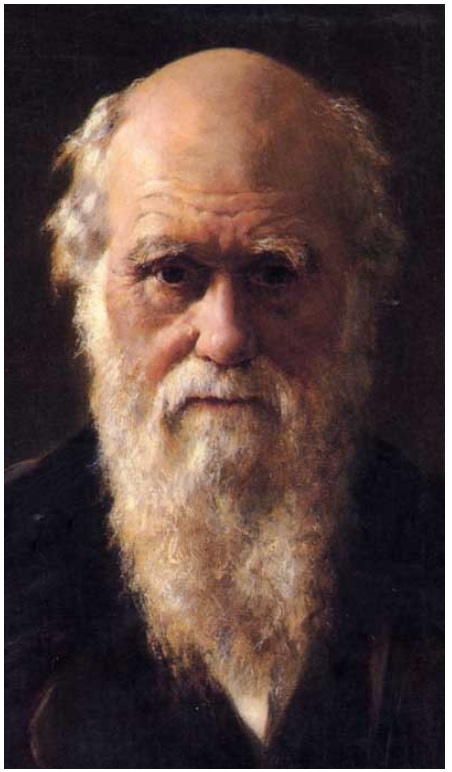
Réponse

... une certaine once d'**intelligence** !

Mais qu'est-ce que l'**intelligence** ?

Cela dépend à qui on pose la question...

J. Piaget



C. Darwin

H.E. Gardner

A. Binet

Réponses

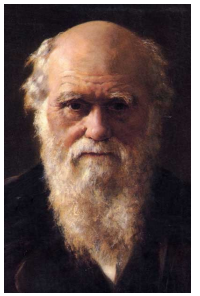
L'intelligence, c'est selon :

- *J. Piaget* (psychologue, biologiste et logicien) :

« *l'adaptation du sujet à son milieu.* » Deux types d'intelligence : sensori-motrice et verbale ou réfléchie.

- *C. Darwin* (biologiste) :

« *ce qui permet la survie de l'individu* »



- *A. Binet* (pédagogue et psychologue) :

« *ce que mesurent les tests d'intelligence* »...

Réponses

- *H.E. Gardner* (psychologue) : théorie de l'intelligence multiple.

Plusieurs types d'intelligence coexiste chez chaque être humain :

- logico-mathématique,
- visuo-spatiale,
- verbo-linguistique,
- corporelle-kinesthésique,
- intrapersonnelle,
- interpersonnelle,
- naturaliste,
- musicale,
- existentielle ou spirituelle.

Mais encore...

Avant eux, d'autres philosophes, psychologues, biologistes, etc. ont également voulu comprendre comment l'*Homo sapiens sapiens* pense, perçoit, agit.

Le domaine de l'IA s'inspire de ses travaux et les enrichit :
en plus de vouloir **comprendre** l'intelligence, elle tente également d'en **construire**.

L'IA systématise et automatise les tâches intellectuelles.

Une brève genèse de l'IA

La préhistoire : 1945-1955

Pendant la guerre : décryptage

Techniques de décryptage → traduction automatique

Echec, mais :

- découverte de l'importance des connaissances non exprimées
- représentation des connaissances
- acquisition des connaissances

Une brève genèse de l'IA

Les débuts : 1955-1970

Acte de naissance : en 1956 à une conférence au Dartmouth College par J. McCarthy

Fondation : « *toute activité intelligente est modélisable et reproductible par une machine.* »

1956 : 1er programme d'IA, *Logic Theorist*, par *Newell, Shaw* et *Simon* pour la démonstration automatique.

1957 : programme de jeu d'échecs, *NSS*, et système de résolution de problèmes généraux, *GPS*.

1960 : développement de LISP par McCarthy

Développement de DENDRAL, un des 1ers systèmes experts

Une brève genèse de l'IA

Les débuts : 1955-1970

Hypothèses fondatrices de l'IA symbolique par *Simon* et *Newell* :

« *Tout système de symboles possède les moyens nécessaires et suffisants pour une action intelligente de caractère général.* »

et

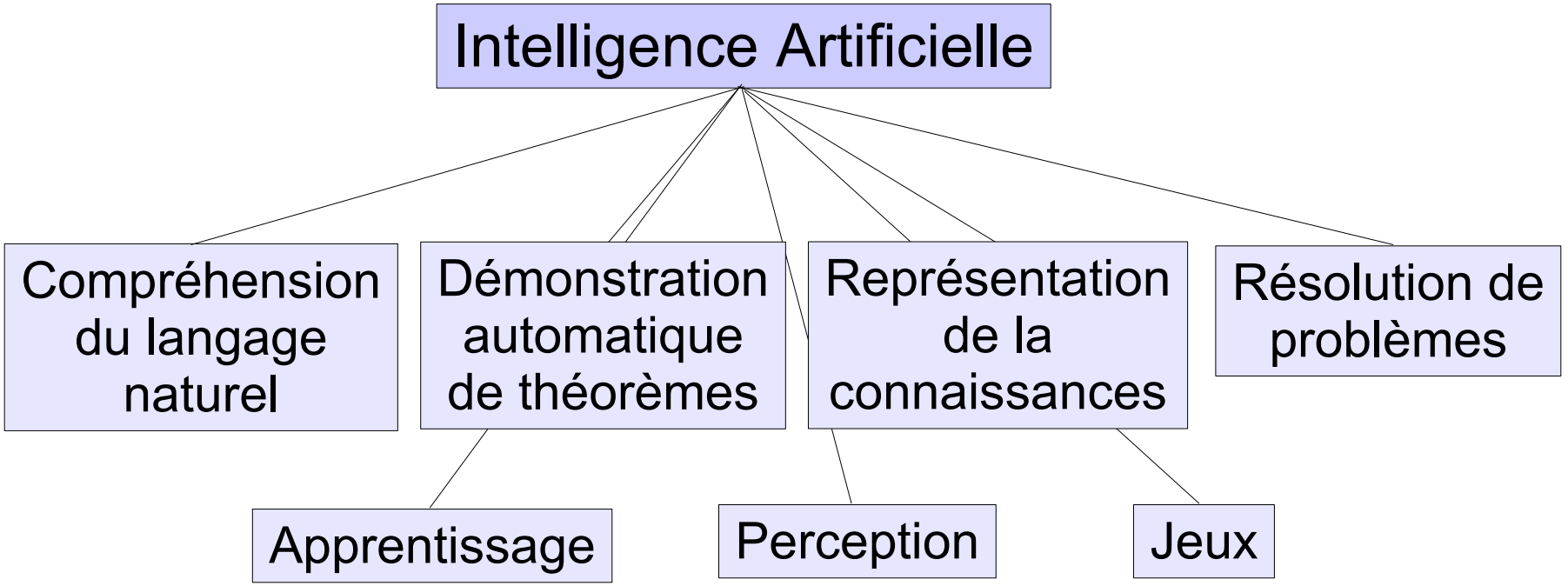
« *Un système de symbole est une machine qui produit dans le temps un assemblage évolutif de structures de symboles.* »

Une brève genèse de l'IA

La spécialisation : 1970-1980

Premiers systèmes à base de connaissances

1965 : principe de résolution par *Robinson*, à la base de PROLOG.



Une brève genèse de l'IA

Une reconnaissance : 1980-1990

Projet *Cinquième Génération* au Japon : développer des technologies et des techniques « efficaces »

→ des avancées dans les architectures parallèles, bases de données, langages de programmation, traitement des informations génétiques...

→ Programme *Strategic Computer Initiative* par le DARPA : 1 milliards de dollars

Ordinateur d'échecs : *DEEP THOUGHT*

Systèmes experts : *PROSPECTOR*, *R1/XCON*

Qu'est-ce que l'Intelligence Artificielle ?

- M. Minsky :

« *... the science of making machines do things that would require intelligence if done by humans.* »

- J.L. Laurière :

« *Tout problème pour lequel aucune solution algorithmique n'est connue, relève a priori de l'intelligence artificielle.* »

- D. McDermott & E. Charniak :

« *l'étude des facultés mentales à l'aide de modèles de type calculatoire.* »

Qu'est-ce que l'Intelligence Artificielle ?

- P. Winston :

« *l'étude des idées qui permettent aux ordinateurs d'être intelligents.* »

- H. Prade :

consiste à « *donner à des machines des capacités leur permettant d'effectuer des tâches ou des activités réputées intelligentes (car jusqu'à présent uniquement réalisées par des humains)*. »

- C. Pellegrini :

« ... *une forme de réaction à l'affirmation longtemps admise stipulant que les ordinateurs ne peuvent pas réaliser des tâches requérant de l'intelligence.* »

Domaines typiques de l'IA

Les domaines typiques de l'IA ont deux caractéristiques en commun :

- manipulent des **informations symboliques** ou **numériques** :
  lettres, mots, signes, dessins, nombres...

- impliquent des **choix** : à certains instants, faire face à plusieurs possibilités → **non déterminisme**, composante essentielle de l'intelligence

Domaines typiques de l'IA

- La perception et la reconnaissance des formes
- Les mathématiques et la démonstration automatique de théorèmes
- Les jeux
- La résolution de problèmes
- La compréhension du langage

Problèmes typiques d'IA

- Problèmes entièrement spécifiables et difficiles

  Exemples : problème du voyageur de commerce, tournée de véhicules, les jeux (puzzles, échecs, dames...)

- Problèmes couvrant un domaine précis : systèmes experts

  → Description spécifiable partiellement

  → Mais buts (très) clairs

  Exemples : MYCIN, DENDRAL, R1/XCON

→ **Résolution non déterministe** : la recherche de la solution est arborescente et non plus séquentielle.

Exemples de systèmes d'IA

- Robots
- Programme d'échecs
- Système de reconnaissance vocale
- Système de synthèse de la parole
- Système de traduction automatique
- Système de reconnaissance de forme
- Système de diagnostic médical
- Programme de preuve de théorème
- Programme de mots croisés
- Programme d'élaboration de tournées de facteurs
- ...

Les deux dimensions de l'IA

Systématise et automatise les tâches intellectuelles pour élaborer des machines capables de :

Penser comme un humain (approche cognitive)	**Penser rationnellement** (approche basée sur une logique mathématique)
Agir comme un humain (approche pragmatiste)	**Agir rationnellement** (domaine de l'optimisation)

Penser comme un humain

Approche cognitive de l'IA : réaliser des programmes imitant dans leur fonctionnement l'esprit humain.
→ sciences cognitives

Sciences cognitives : ont pour but de « décrire, expliquer et le cas échéant, simuler les principales dispositions de l'esprit humain - langage, raisonnement, perception, coordination motrice, planification. »

Encyclopedia Universalis

Penser comme un humain

L'IA de la machine est alors construite à partir de modèles et techniques empruntés à plusieurs domaines (psychologie, linguistique, philosophie, neuroscience...) :

- états mentaux
- structures de mémoire
- apprentissage de symboles
- ...

Agir comme un humain

Approche pragmatiste de l'IA : développer des théories permettant d'améliorer notre capacité à programmer efficacement un ordinateur. Si possible, cherche à obtenir de meilleurs résultats que ceux que pourraient obtenir un être humain.

Une I.A. = boîte noire manipulant les données d'entrée pour obtenir des résultats en sortie.

Boîte intelligente = si elle réussit un certain nombre de tests, qui, réussis par un être humain, permettrait de dire qu'il est intelligent.

Agir comme un humain

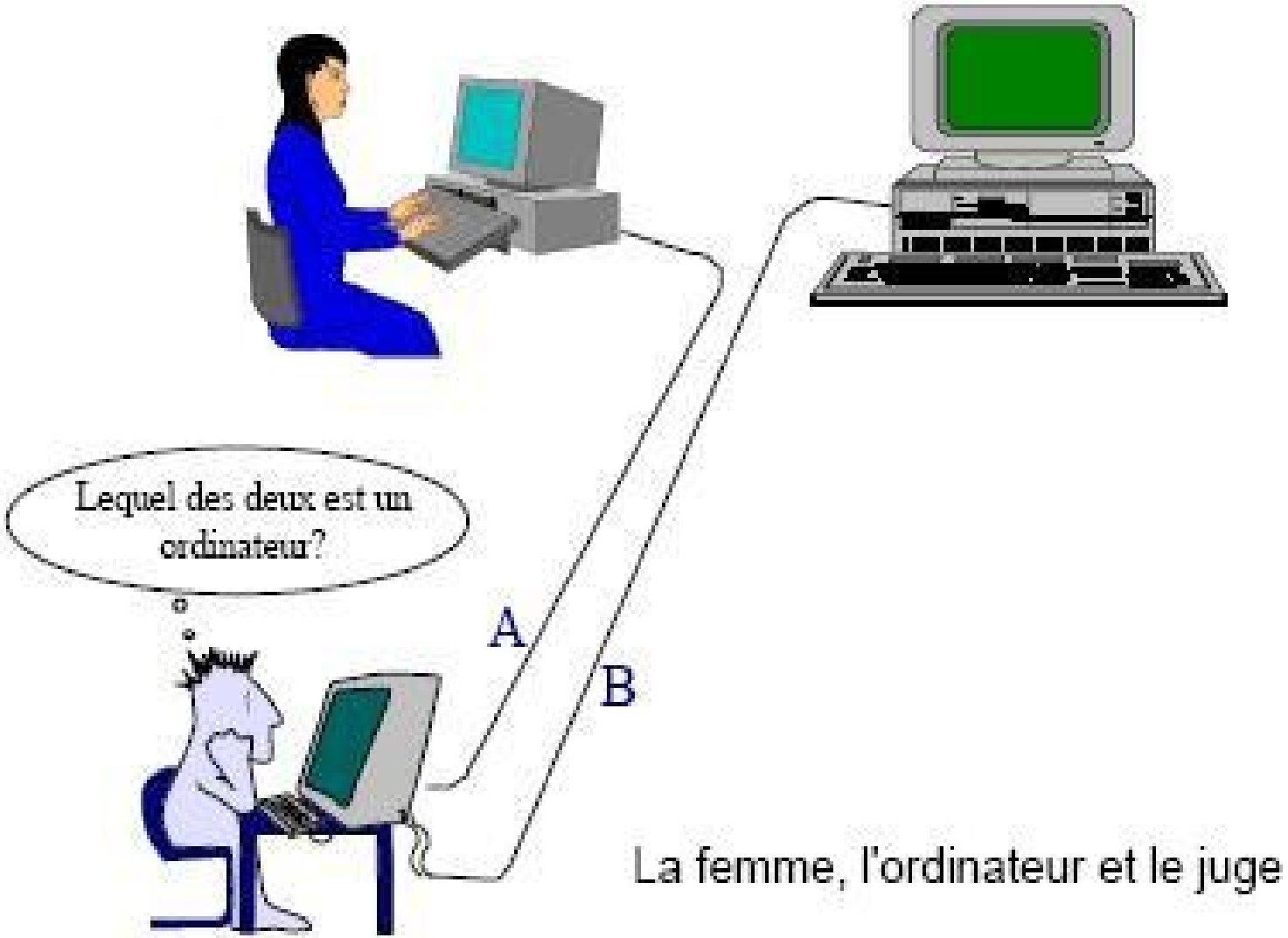
Méthodologie :

- identifier une tâche ou activité pour laquelle l'homme est meilleur que la machine
- la faire réaliser par la machine

Exemples :

- Prouver un théorème
- Jouer aux échecs
- Diagnostiquer une maladie
- Planifier ses déplacements
- Se déplacer dans un bâtiment
- Test de Turing...

Agir comme un humain : le test de Turing



La femme, l'ordinateur et le juge

Penser / agir rationnellement

Objectif :

Prendre toujours la meilleure décision possible compte tenu des éléments disponibles et connus (informations/connaissances, temps, ressources)

Raisonnement logique lorsque :

- connaissances parfaites
- ressources illimitées

Rationnalité limitée lorsque :

- connaissances imparfaites
- ressources limitées

→ Techniques empruntées à la recherche opérationnelle, la théorie du contrôle, l'économie.

→ Ignore le rôle de la conscience et des émotions sur le processus de raisonnement

Penser rationnellement

Aristote : quels sont les processus corrects de la pensée ?

Logique mathématique :

- permet de représenter des énoncés du monde et de raisonner en les manipulant de manière formelle
- à la base de systèmes modernes de résolution de problèmes complexes

Difficultés :

- tous les comportements ne sont pas le résultat de raisonnements logiques
- difficile d'exprimer le monde réel en terme de faits uniquement vrais ou faux

Agir rationnellement

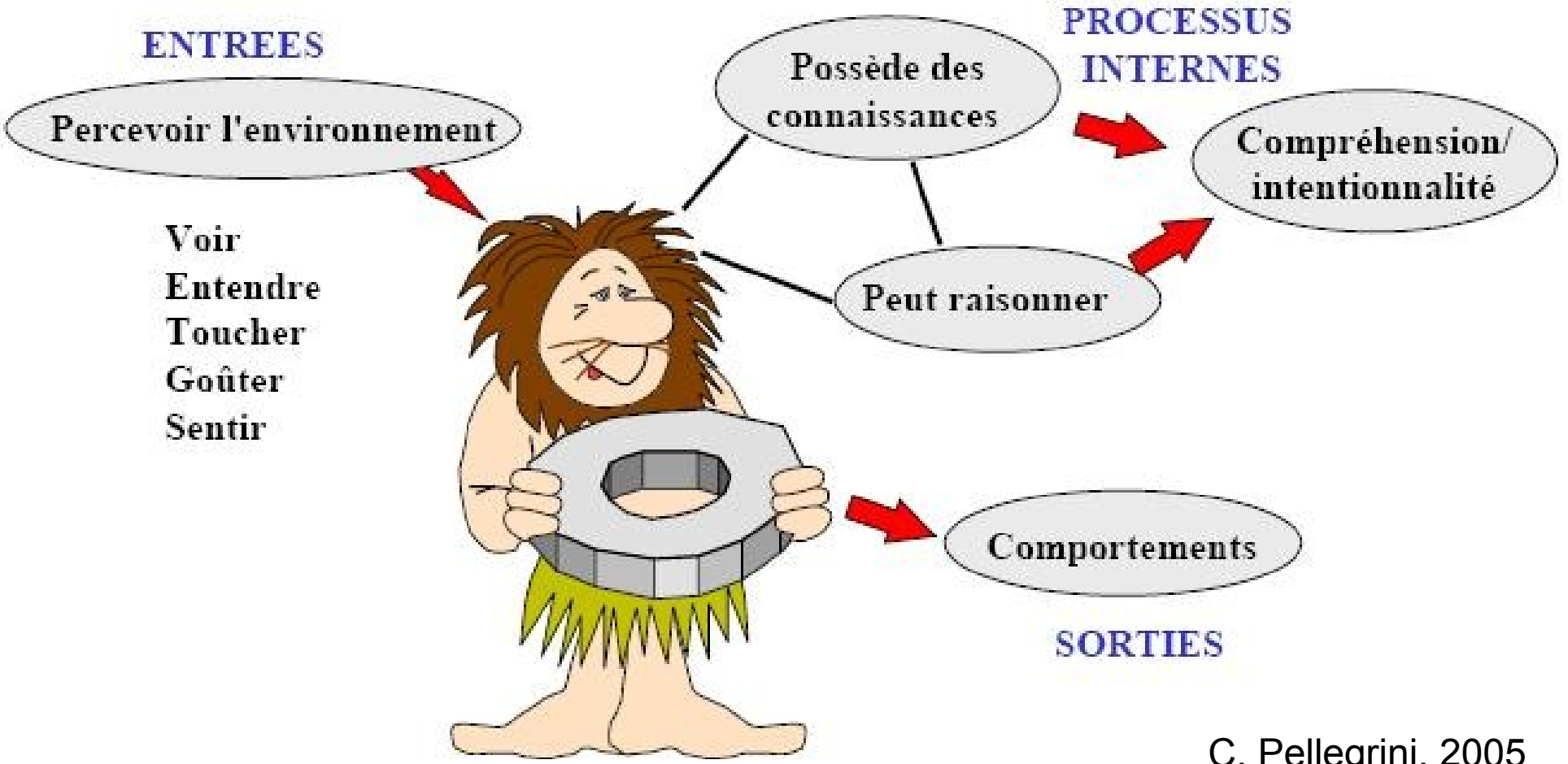
Comportement rationnel = effectuer l'action correcte, la plus adéquate étant donné ce que l'on sait de la situation actuelle.

Action correcte : celle qui est supposée optimiser un objectif donné

Comportement rationnel : ne requiert pas nécessairement le fait de penser (ex. : cligner des yeux) mais la pensée doit être au service de l'action rationnelle.

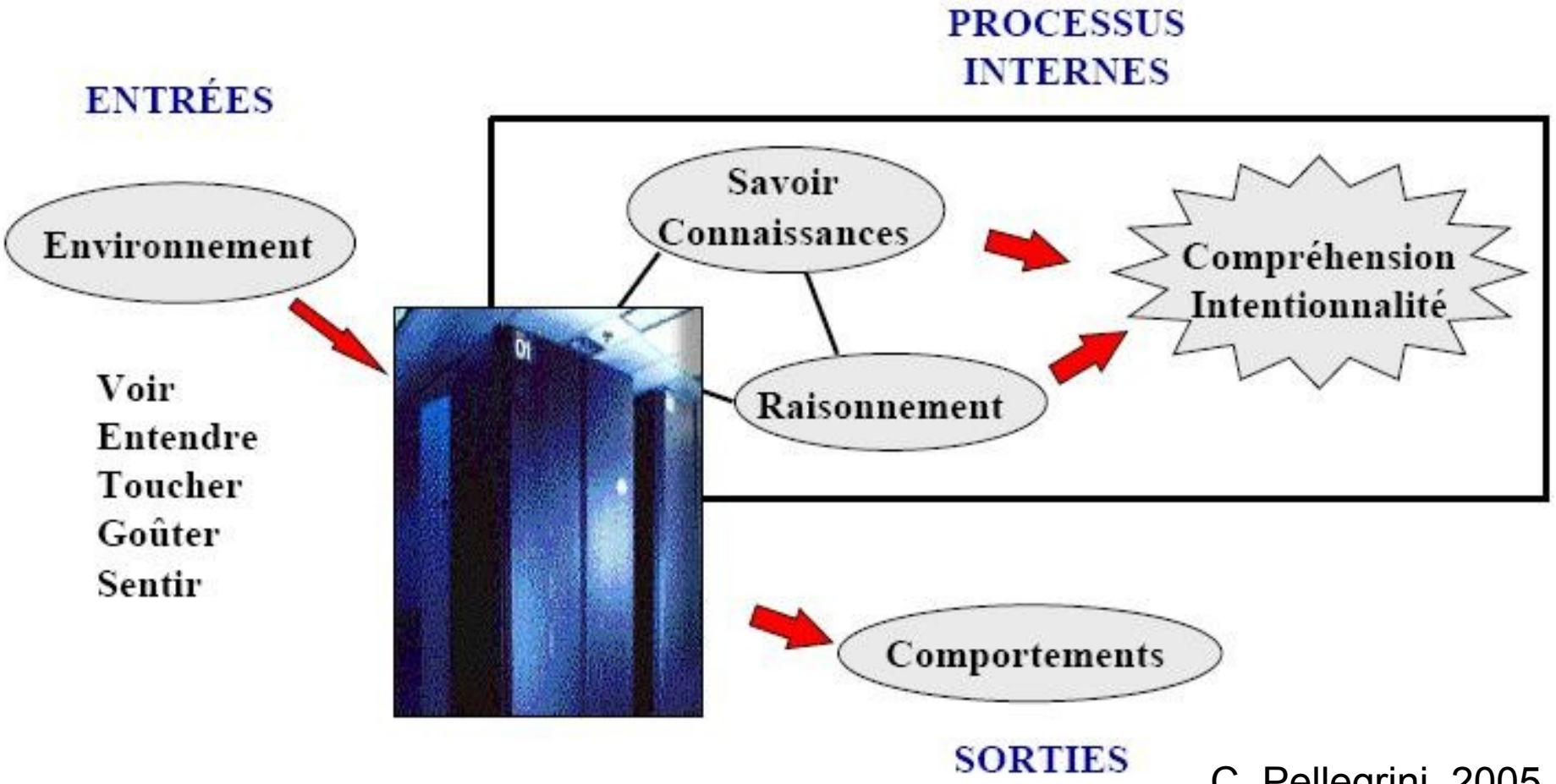
Exemple : agent rationnel, régit par une fonction f : $P^* \rightarrow A$, qui en fonction des percepts reçus choisit la meilleure action à réaliser.

Une entité intelligente



C. Pellegrini, 2005

Un agent intelligent



C. Pellegrini, 2005

Prédictions et réalité

Dans les années 60, un célèbre professeur du MIT disait :
« *à la fin de l'été, on aura construit un oeil électronique.* »

En 2007, il n'y a toujours pas de système de vision par ordinateur capable de comprendre une scène complexe.

Mais des systèmes informatiques effectuent quotidiennement :

- Surveillance de trafic routier,
- Reconnaissance de visages
- Analyse d'images médicales
- ...

Prédictions et réalité

En 1958, H. Simon (CMU) prédisait que dans 10 ans, un programme informatique serait capable de battre le meilleur joueur aux échecs.

Cette prédiction s'est vérifiée en 1997 : Deep Blue a battu Kasparov : 3,5 contre 2,5.

Aujourd'hui, les ordinateurs ont gagné des titres de champion du monde aux jeux de dames, Othello, échecs.

Ils restent encore très mauvais au jeu de Go, jeu hautement stratégique...

Prédictions et réalité

Dans les années 1970, beaucoup croyaient que les robots informatisés seraient incontournables, présents dans les usines aussi bien qu'au domicile.

Aujourd'hui, quelques industries (automobiles, électronique) sont très robotisées, mais les robots domestiques sont encore utopiques.

Des robots ont explorés Mars, d'autres réalisent des opérations chirurgicales (cerveau, coeur) et des robots humanoïdes sont disponibles à la location au Japon.

(voir http://world.honda.com/news/2001/c011112.html)

Prédictions et réalité

Démarche générale pour la résolution de problèmes, représentation des connaissances et notions de récursivité

Caractéristiques de l'énoncé d'un problème

Qu'est-ce qu'un « problème » :
Situation dans laquelle on ne voit pas immédiatement la suite d'actions à effectuer pour atteindre un objectif.

La plupart des problèmes ne sont pas posés de manière formelle (rigoureuse), car exprimés en langage naturel, qui est :

- **Incomplet** : énoncé contient des informations implicites
- **Ambigu**
- **Redondant** : énoncé peut insister sur des informations qui ne traduisent pas forcément les difficultés du problème
- **Incorrect**

Démarche générale pour la résolution de problèmes (1/2)

E1 : Compréhension de la totalité de l'énoncé

E2 : Inférences immédiates

E3 : Identification de la difficulté du problème

E4 : Incubation (pause,distraction)

E5 : Reformulation, choix d'une représentation

E6 : Résolution partielle (retour E2) ou totale

E7 : Vérifier

E8 : Généraliser

Comprendre

Concevoir un plan

Exécuter le plan

Examiner la solution

Démarche générale pour la résolution de problèmes (2/2)

Généraliser :

1) Peut-on appliquer la méthode proposée sur une généralisation du problème

2) Existe-t'il d'autres problèmes où la même méthode s'appliquerait avec succès ?

3) Existe-t'il d'autres méthodes pour ce problème ?

Les représentations

« Une représentation ne vaut que par la facilité de mise en oeuvre des procédures de traitement. » J.L. Laurière

Un mode de représentation des connaissances inclut :
- une structure de données codant la connaissance
- un mécanisme d'exploitation de la connaissance codée

Un mode de raisonnement doit posséder trois propriétés :
- Clarté,
- Puissance d'expression,
- Efficacité du mécanisme d'exploitation

Représentation **déclarative** ≠ représentation **procédurale**

Quelques représentations

Plusieurs représentations en fonction du problème à résoudre :

- Représentations logiques
- Réseaux sémantiques
- Règles de production
- *Frames*

Représentations logiques

Connaissance = formule construite selon une syntaxe précise.

Règles de raisonnement : *modus ponens*, *modus tollens*, résolution...

Exemples :

- $(A \wedge B)$
- $(A \wedge B) \vee \neg C$
- $(A \wedge B) \Rightarrow C$
- $\forall x\ f(x) \Rightarrow g(x)$
- $\exists x\ f(x) \wedge g(x)$

Réseaux sémantiques

- Hiérarchie de concepts, dictionnaire sous forme de graphe
- Noeud = concept
- Arc = relation entre concepts
- Mécanisme d'héritage de propriétés

Exemple :

« Clyde est un moineau. Un moineau est une sorte d'oiseau. Un oiseau a des ailes. »

Règles de production

Connaissance déclarative de la forme :

Si <condition> Alors <conclusion> (α)

Exemple :

« Si le moteur cale
et l'allumage est correct
et le réservoir d'essence n'est pas vide
Alors vérifier la carburation. »

Les Frames

Représentation d'expériences passées
Granularité importante : connaissances structurées relatives à un objet, concept ou situation

Exemple :

<frame vin :
sorte de : boisson
appellation : (domaine : AOC/vin de pays)
(défaut: vin de pays)
(si besoin : demander appellation)
nature : (domaine : sec/demi-sec/doux/liquoreux/corsé)
degré alcool : (intervalle : 9 à 15)
(défaut : 12)
producteur : (si besoin : trouver nom sur étiquette)
robe : ...
>

Notation des représentations

- Notations linéaires :
  - infixées,
  - préfixées,
  - postfixées

- Notations non linéaires :
  - sous forme d'arbres
  - sous forme de graphes

Définition de la récursivité

La récursivité est la possibilité de faire apparaître dans la définition d'une entité une référence à elle-même. Dans ce cas, on dit que l'entité possède une définition récursive ou qu'elle est intrinsèquement récursive.

En programmation, on distingue deux types d'entités récursives :

- les fonctions récursives
    - fonction factorielle, fonction puissance
    - fonction de *Fibonacci*

- les structures de données récursives
    - les nombres
    - les listes
    - les arbres
    - les objets fractals...

Approche pour la construction d'algorithmes récursifs (1/2)

Un problème se prête bien à l'analyse récursive lorsqu'il peut être décomposé en sous-problèmes de taille plus petite et de même nature que le problème initial.

L'analyse récursive est constituée de trois étapes :

1) paramétrage du problème

2) recherche d'un cas trivial et de sa solution

3) décomposition du cas général

Approche pour la construction d'algorithmes récursifs (2/2)

1) Paramétrage du problème

Il s'agit d'identifier tous les paramètres du problème, en particulier ceux dont la taille décroît à chaque appel récursif.

2) Recherche d'un cas trivial et de sa solution

Un cas trivial est un sous-problème qui peut être résolu **SANS appel récursif**. Il correspond souvent au cas où la taille est nulle.

3) Décomposition du cas général

Cette étape a pour but de ramener le problème donné à l'instant t vers un ou plusieurs sous-problèmes que l'on suppose déjà traités à des instants précédents et de taille plus petite.

Application de l'approche : définition de la fonction factorielle

Par exemple, définir la fonction factorielle revient à spécifier les trois étapes suivantes :

1) Paramétrage (profil) de la fonction

$$factorielle : N \rightarrow N$$

2) Cas trivial

$$Pour\ n < 2,\ factorielle(\ n\ ) = 1$$

3) Cas général (appel récursif)

$$Pour\ n >= 2,\ factorielle(\ n\ ) = n * factorielle(\ n-1\ )$$

Notations pour l'écriture d'une fonction : exemple avec la fonction factorielle

1. Profil :

factorielle : N → N

2. Définition formelle de factorielle(n) :

{ n < 2 } → factorielle(n) = 1
*{ n >= 2 } → factorielle(n) = n * factorielle(n-1)*

3. Jeu d'essais :

*factorielle(3) = 3 * factorielle(2)*
*= 3 * 2 * factorielle(1)*
*= 3 * 2 * 1*
= *6* (voir page suivante)

Exécution d'une fonction récursive : exemple avec la fonction factorielle

factorielle(3)

Exécution d'une fonction récursive : exemple avec la fonction factorielle

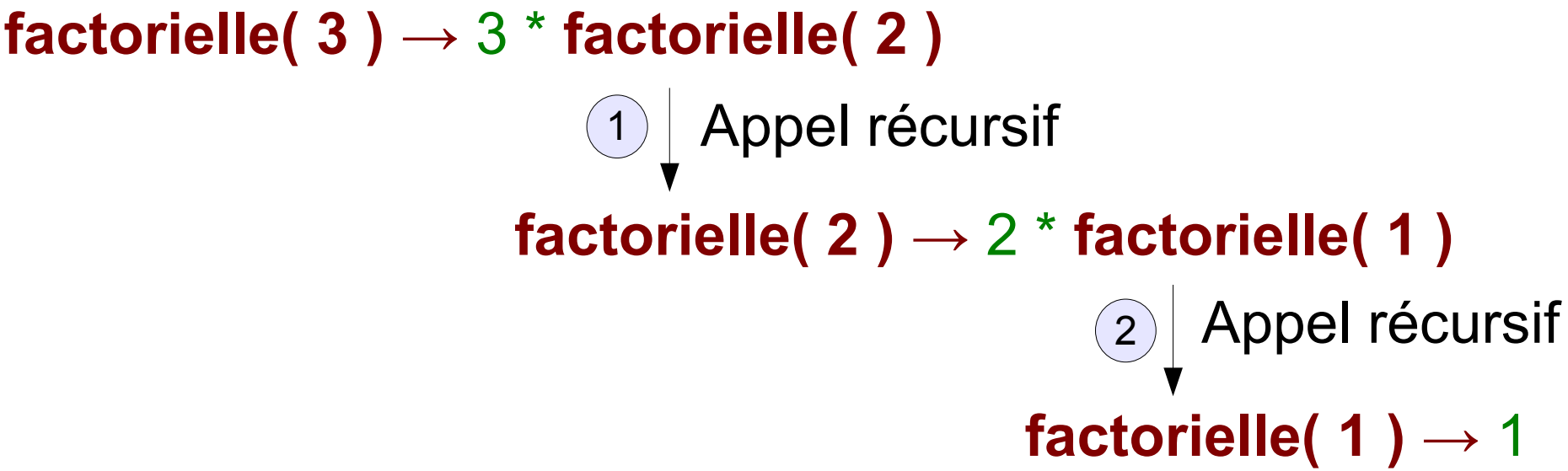
factorielle(3) → 3 * **factorielle(2)**

Exécution d'une fonction récursive : exemple avec la fonction factorielle

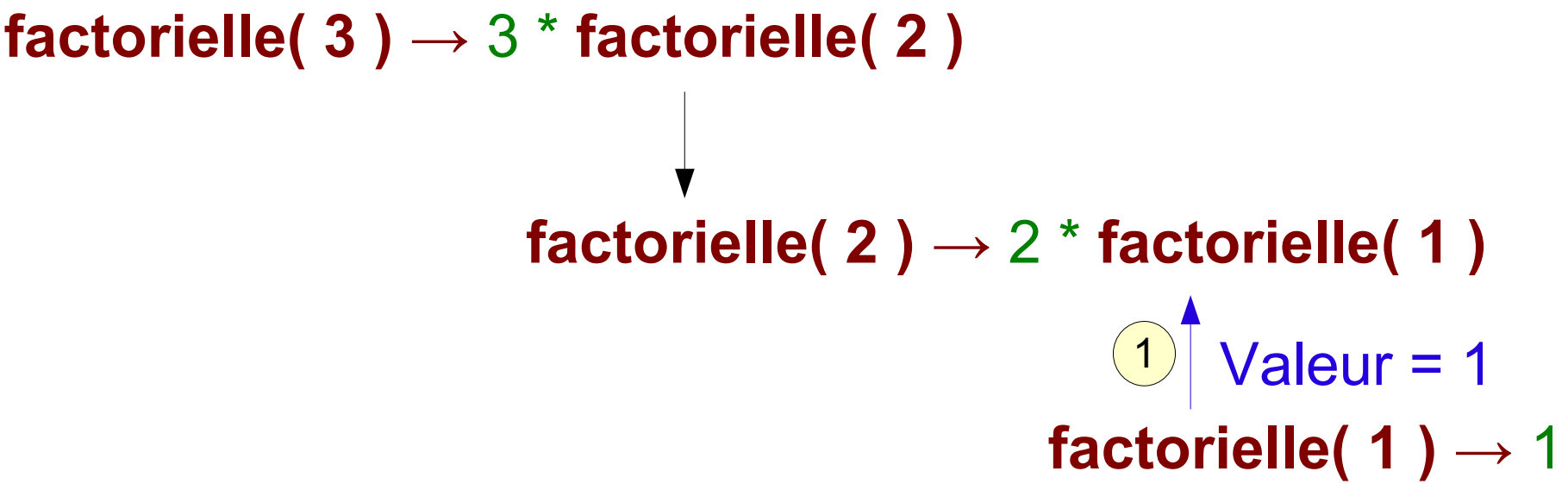
factorielle(3) → 3 * **factorielle(2)**

factorielle(2) → 2 * **factorielle(1)**

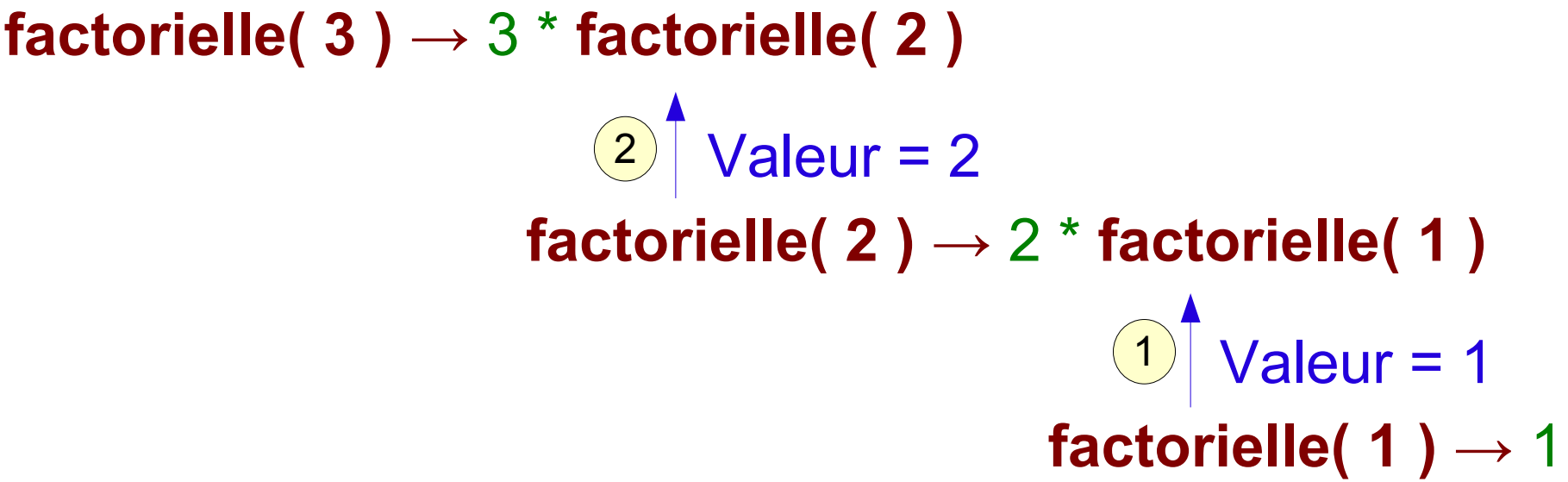
Exécution d'une fonction récursive : exemple avec la fonction factorielle



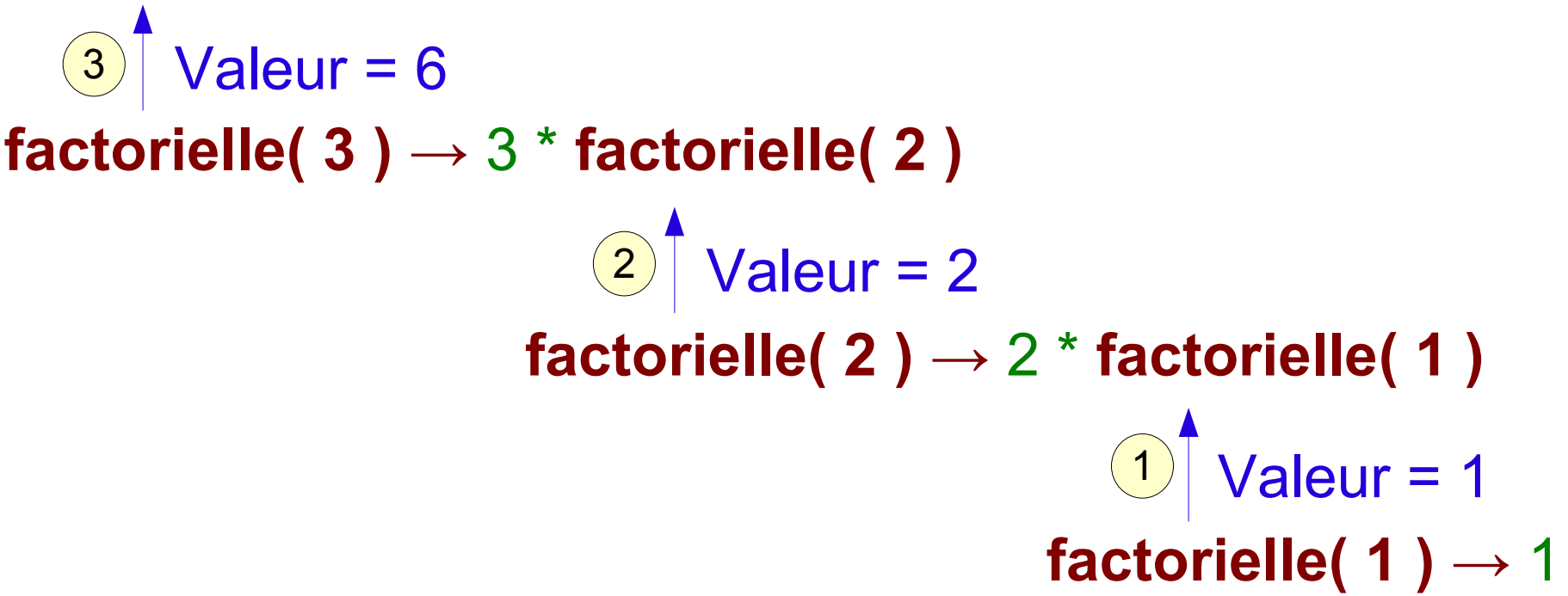
Exécution d'une fonction récursive : exemple avec la fonction factorielle



Exécution d'une fonction récursive : exemple avec la fonction factorielle



Exécution d'une fonction récursive : exemple avec la fonction factorielle



Langages dédiés à la résolution de problèmes d'IA

Catégories de langages

- **Langages procéduraux** (Fortran, C, Pascal, ADA...)
  - Programmation impérative : programme indique par son algo. quelles sont les étapes permettant d'aboutir à une solution.
  - Programme = données + actions
  - Exécution d'un programme = changement d'état de la mémoire centrale

- **Langages orientés objets** (Smalltalk, C++, Java...)
  - Généralement basés sur la programmation impérative
  - Basés sur la notion de classes et d'objets (instances de classes)
  - Encapsulation des données (attributs) et des fonctions (opérations) manipulant ces données
  - Une classe n'est pas exécutable : elle doit être instanciée
  - La même classe peut être instanciée plusieurs fois : les attributs des différentes instances peuvent alors avoir des valeurs différentes.

Catégories de langages

- **Langages logiques** (Prolog)
  - Programmation déclarative
  - Programme = base de faits et règles
  - Aucune description du processus de résolution
  - Exécution d'un programme = indication de but(s) sous forme de requête(s)
  - Moteur d'inférences tente de satisfaire le(s) but(s) à partir des faits et des règles
  - Basés sur la logique des prédicats du premier ordre

- **Langages fonctionnels** (Lisp, Caml)
  - Programme = description d'une fonction qui, appliquée à des arguments, retournera un résultat.
  - Possibilité de créer des fonctions d'ordre supérieur : fonctions acceptant des fonctions en arguments et retournant des fonctions comme résultat.
  - Basés sur le λ-calcul.

Exemple #1 de programme PROLOG

On désire disposer d'une base de faits évoquant les relations suivantes :

Jean-Claude est le père d'André
Léa est la mère d'André
Hugues est le père de Jean-Claude
Hugues est le grand-pere d'André

Pour transformer ces énoncés en PROLOG, il faut :
• identifier les objets
• identifier les relations entre objets. En PROLOG, ces relations s'appelle des ***prédicats***. Un prédicat est soit vrai, soit faux.

Exemple #1 de programme PROLOG

On désire disposer d'une base de faits évoquant les relations suivantes :

Jean-Claude est le père d'André
Léa est la mère d'André
Hugues est le père de Jean-Claude
Hugues est le grand-pere d'André

En vert : les objets
En rouge : les relations.

Exemple #1 de programme PROLOG

Programme :

```
L1    pere( 'jean-claude', andre ).
L2    mere( lea, andre ).
L3    pere( hugues, 'jean-claude' ).
L4    grand_pere( X,Y ) :- pere( X, Z ), pere( Z, Y ).
```

Faits

Règle

Signification :

- *'jean-claude'*, *andre*, *lea* et *hugues* sont des **objets constants**
- *pere*, *mere* et *grand_pere* sont des prédicats, c'est-à-dire des relations entre objets qui sont supposées vraies
- X, Y et Z sont des variables pouvant être instanciées par les constantes définies précédemment.

Exemple #1 de programme PROLOG

Programme :

L1 *pere('jean-claude', andre).*
L2 *mere(lea, andre).*
L3 *pere(hugues, 'jean-claude').*
L4 *grand_pere(X,Y) :- pere(X, Z), pere(Z, Y).*

} Faits
} Règle

Signification :

- L1 : Jean-Claude est le père d'André
- L2 : Léa est la mère d'André
- L3 : Hugues est le père de J.-C.
- L4 : X est grand-père de Y si il existe un Z tel que X est père de Z et Z est père de Y.

Exemple #2 de programme PROLOG

Programme :

L1 factorielle(0, 1).
L2 fatorielle(1, 1).
*L3 factorielle(N, R) :- N > 1, N1 is N-1, factorielle(N1,R1), R is N*R1.*

Signification :

- *factorielle* est un prédicat qui relie un premier entier à un autre
  - Le premier entier représente le paramètre d'entrée de la fonction
  - Le second entier représente le résultat de la factorielle
- L1 : la factorielle de 0 est 1
- L2 : la factorielle de 1 est 1
- L3 : si N > 1, la factorielle de N est R, avec R = N* (N-1)!
- *is* est un prédicat prédéfini de PROLOG qui évalue le terme à droite et affecte sa valeur à la variable de gauche.

Exemple de programme LISP

Programme :

```
L1    (setq personnes '("jean-claude" "andre" "lea" "hugues"))
L2    (car personnes)
L3    (car (cdr personnes))
L4    (+ 1 5 6 8)
L5    (* (+ 5 2 3) (- 5 2))
```

Signification :

- L1 : *personnes* est une liste contenant les chaînes de caractères...
- L2 : retourne le premier élément de la liste
- L3 : retourne le premier élément du reste de la liste
- L4 : calcule la somme de 1,5,6 et 8 et l'affiche
- L5 : affiche 30

Plan des cours

1. Introduction à l'IA
2. Concepts fondamentaux des SBC et de PROLOG
3. Programmation PROLOG
4. Systèmes formels et logique propositionnelle
5. Logique des prédicats du premier ordre
6. Introduction à la théorie des langages formels
7. Introduction aux problèmes de satisfaction de contraintes
8. Introduction au Lambda calcul et à LISP
9. LISP
10. Planification et stratégie de recherche dans les graphes
11. Théorie des jeux et algorithmes
12. Pratique de la programmation d'IA dans les jeux
13. Introduction à l'apprentissage automatique
14. Bilan des apports de l'IA et références bibliographiques