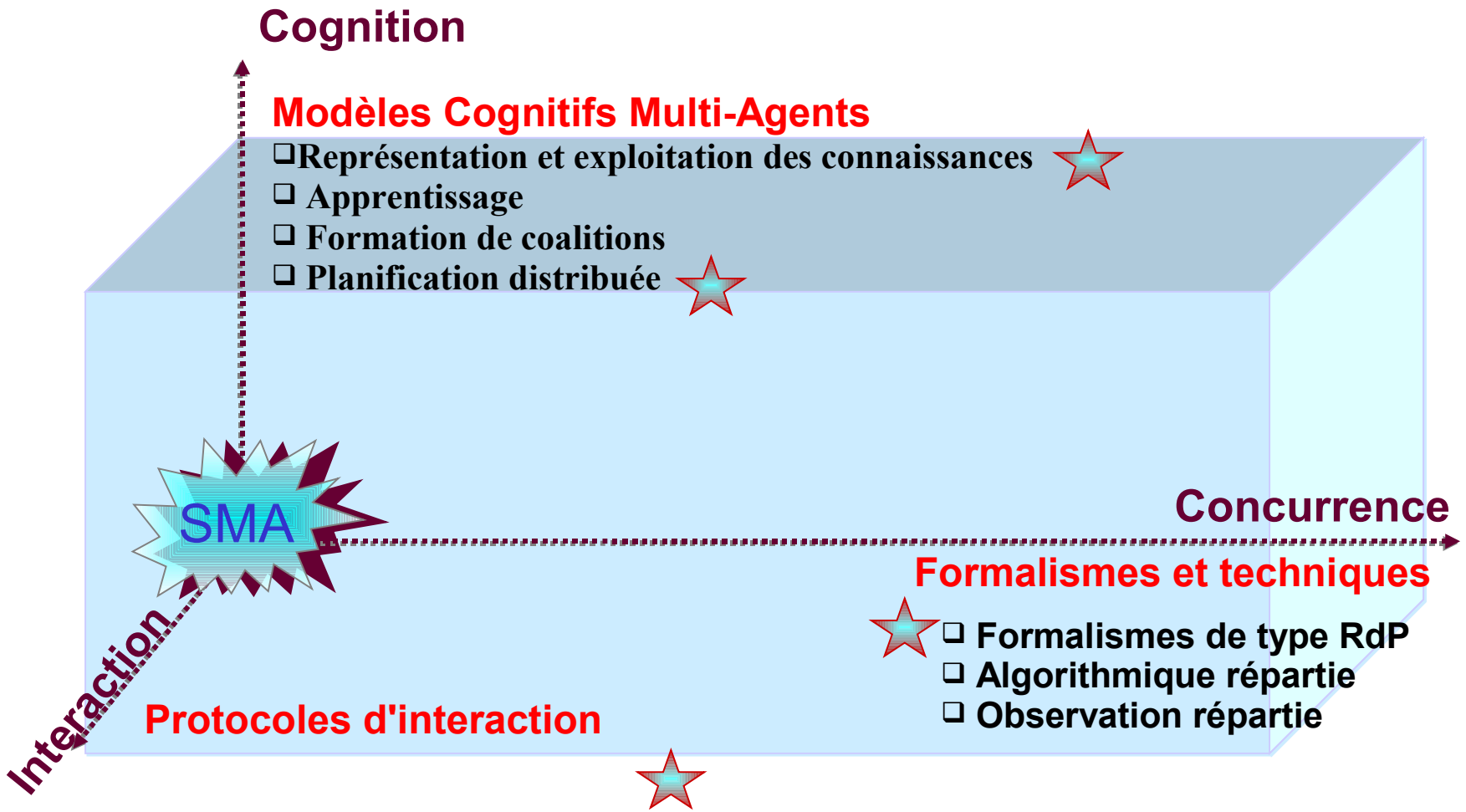
SMA et techniques

Tarek Melliti
Laboratoire IBISC
(Informatique Biologie Intégrative et Systèmes Complexes)
tarek.melliti@ibisc.univ-evry.fr

SMA et technique : Les trois dimensions [Amal el Fallah]



Représentation de connaissances

Tarek Melliti
Laboratoire IBISC
(Informatique Biologie Intégrative et Systèmes Complexes)
tarek.melliti@ibisc.univ-evry.fr

Introduction : définition d'un agent intelligent

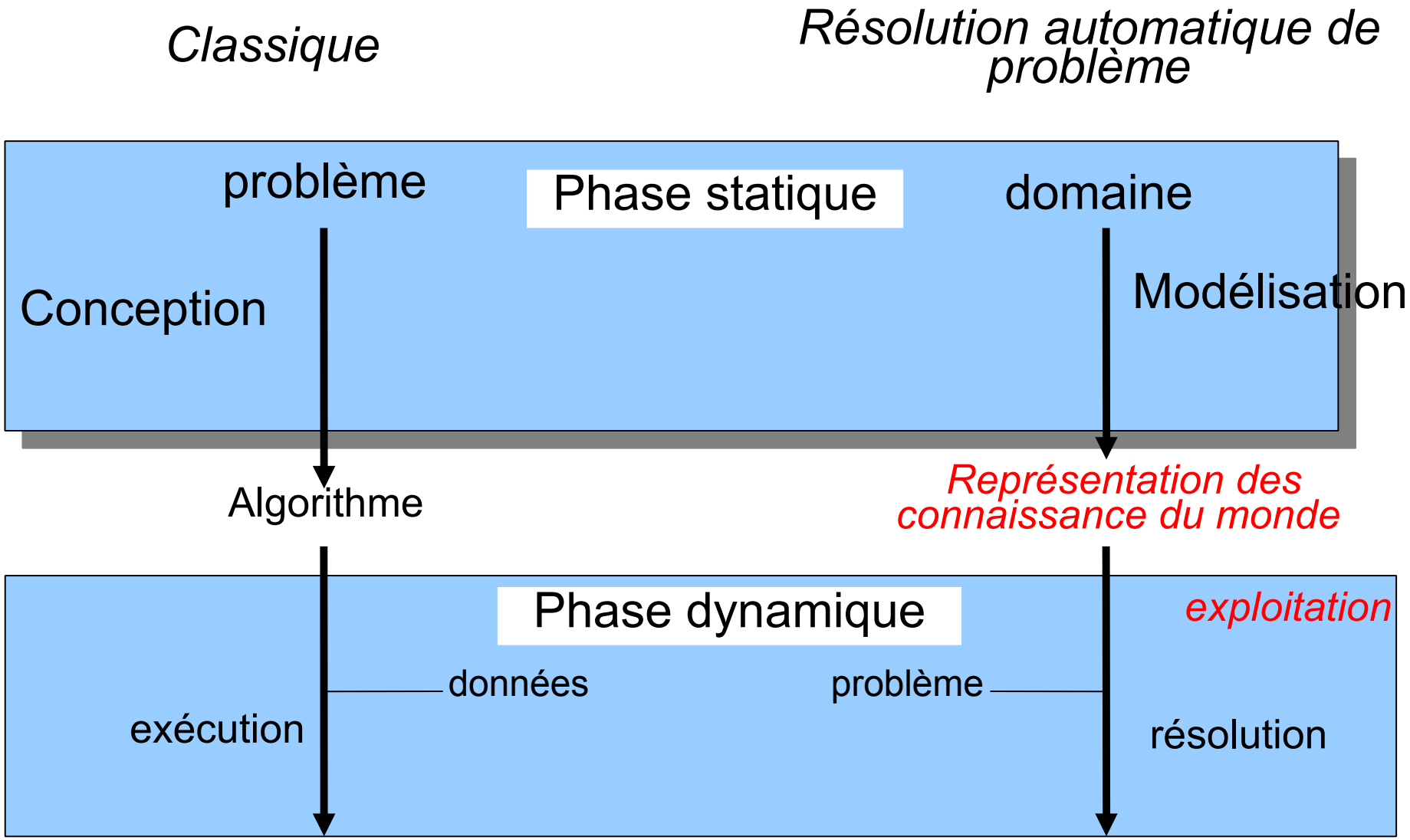
- Un agent intelligent (par opposition à agent reactif):
  - Autonomie
  - Pro-activity
  - Guidé par ses objectifs
  - Coopére
  - Etc.

- Composante essentielle => l'agent est capable de résoudre des problèmes tout seul ou collectivement.

- =>Pas un developpement classique.

- comment on a pu mettre en place un système capable de résoudre automatiquement des problèmes qui lui sont posés ? => Intelligence Artificielle

Introduction : Résolution de problème



Ce qui va suivre ..

- Dans cette partie du cours on va voir deux types d'approche de résolution pour deux classes de problèmes différents.
- Le premier : Systèmes experts
  - Type de problème: classification, diagnostic, réponse à des questions.
  - Exemple: est ce que la “baleine bleue” est un mammifère ?
- Le deuxième : planification
  - Définir la suite d'actions pour atteindre un état donné
  - Exemple: comment aller à la fac ?

Les Systèmes Experts

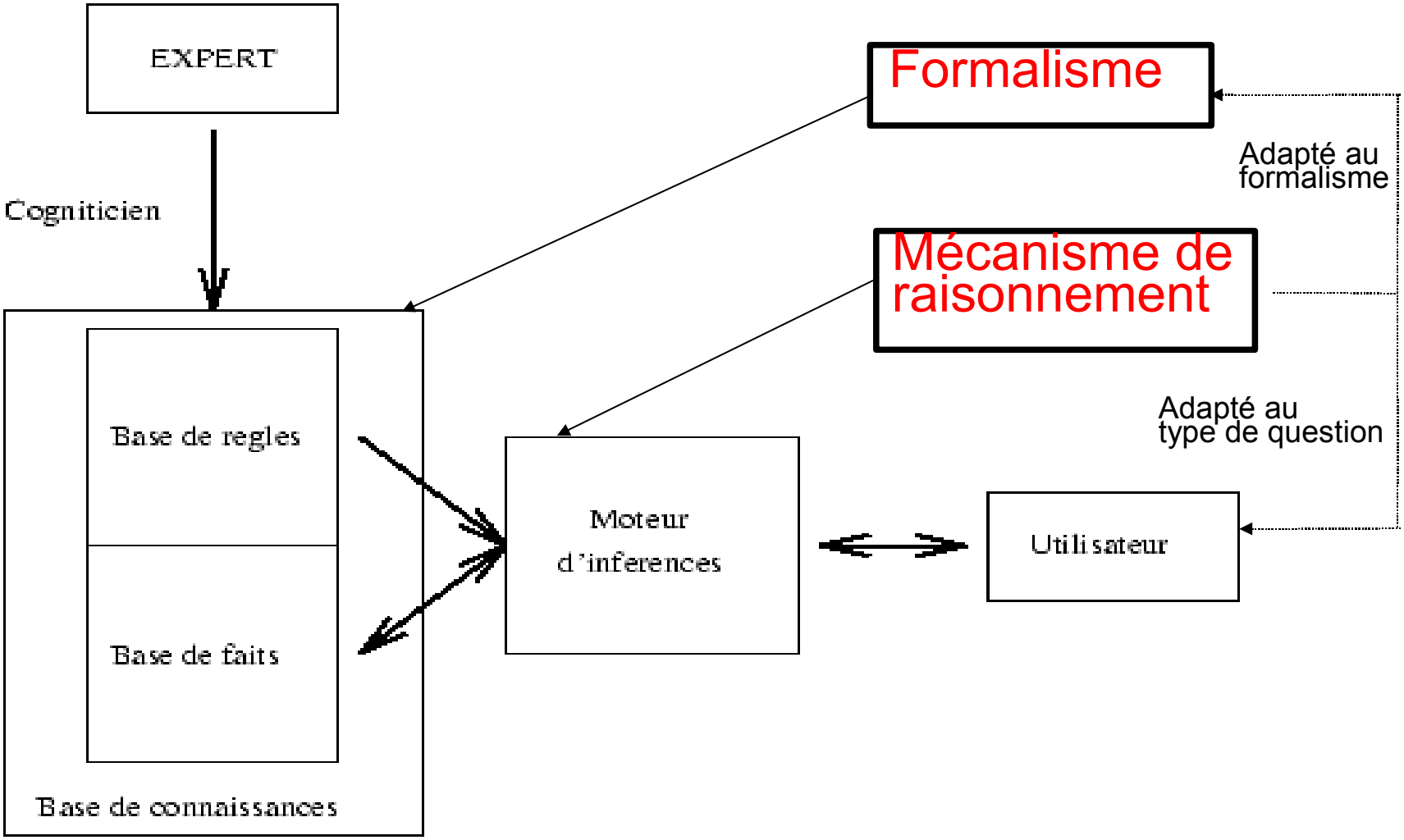
Les systèmes experts

- Un système expert est un logiciel qui reproduit le comportement d'un expert humain accomplissant une tâche intellectuelle dans un domaine précis. On peut souligner les points suivants :

  - Les systèmes experts sont généralement conçus pour résoudre des problèmes de *classification* ou de *décision* (diagnostic médical, prescription thérapeutique, régulation d'échanges boursiers, ...)

  - Les systèmes experts sont des outils de l'Intelligence Artificielle, c'est-à-dire qu'on ne les utilise que lorsqu'aucune méthode algorithmique exacte n'est disponible ou praticable

Vision Globale



La connaissances ? (1)

- Différence entre donnée, information, connaissance.
  - la donnée transporte l'information : ce sont des signaux non interprétés ;
  - l'information est une interprétation de la donnée ;
  - La connaissance utilise l'information dans le cadre d'actions, dans un but précis. Les actions peuvent être la prise de décisions, la création de nouvelles informations, etc...
- Questions fondamentales
  1. Acquisition de la connaissance du domaine
  2. Représentation de la connaissance (un formalism syntaxe + sémantique)
  3. Contrôle du raisonnement (complétude et validité)

La connaissances ? (2)

- Connaissance => capacité à mobiliser des informations pour agir
- Le passage de INFORMATION à CONNAISSANCE est lié à l'expérience de l'action => pas de frontière parfaitement définie
- Définition : Connaissance = Information (donnée) qui influence un processus.
- Pas de classement universel des différents types de connaissances (voir la tentative de Porphyre)
- Connaissance : ensemble des notions et des principes qu'une personne acquiert par l'étude, l'observation ou l'expérience et qu'elle peut intégrer à des habiletés . (dénition sur le web)

- synonyme de Génie cognitif :
  - Discipline étudiant l'extraction et la formalisation de connaissances provenant d'un expert humain en vue d'automatiser le raisonnement.
- En informatique, l'ingénierie des connaissances évoquerait les techniques pour manipuler des connaissances sur ordinateur.

La représentation de connaissance : vers un formalisme

- La représentation des connaissances est le support préalable aux traitements ultérieurs que l'on souhaite effectuer sur ces connaissances:
  1. Organiser, classer,...
  2. Chercher, extraire,...
  3. Déduire, établir des contradictions, réviser,...

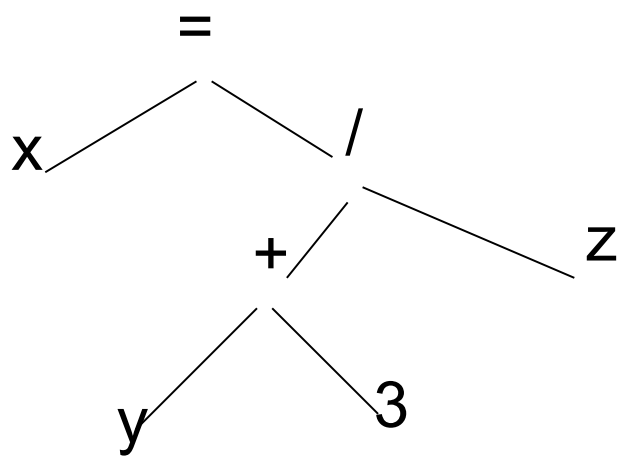
- D'une certaine manière, la représentation des connaissances explicites dans un formalisme vise la recherche de connaissances implicites mais inhérentes aux faits de base.

La représentation ? (1)

- Dire que A « représente » B
  - Ne suffit pas pour que ce soit « vrai »
  - Il convient de vérifier que si B a un certain effet sur un processus P, A démontre un effet « équivalent » sur un processus « équivalent »
- A n'est cependant pas « équivalent » à B
  - « Une carte n'est pas le territoire » (heureusement!)
  - Une carte « représente » le territoire dans le cadre d'un processus de recherche d'un itinéraire (par exemple)
- Séparation l'objet de la représentation, le monde réelle, et le monde artificiel utilisé pour.

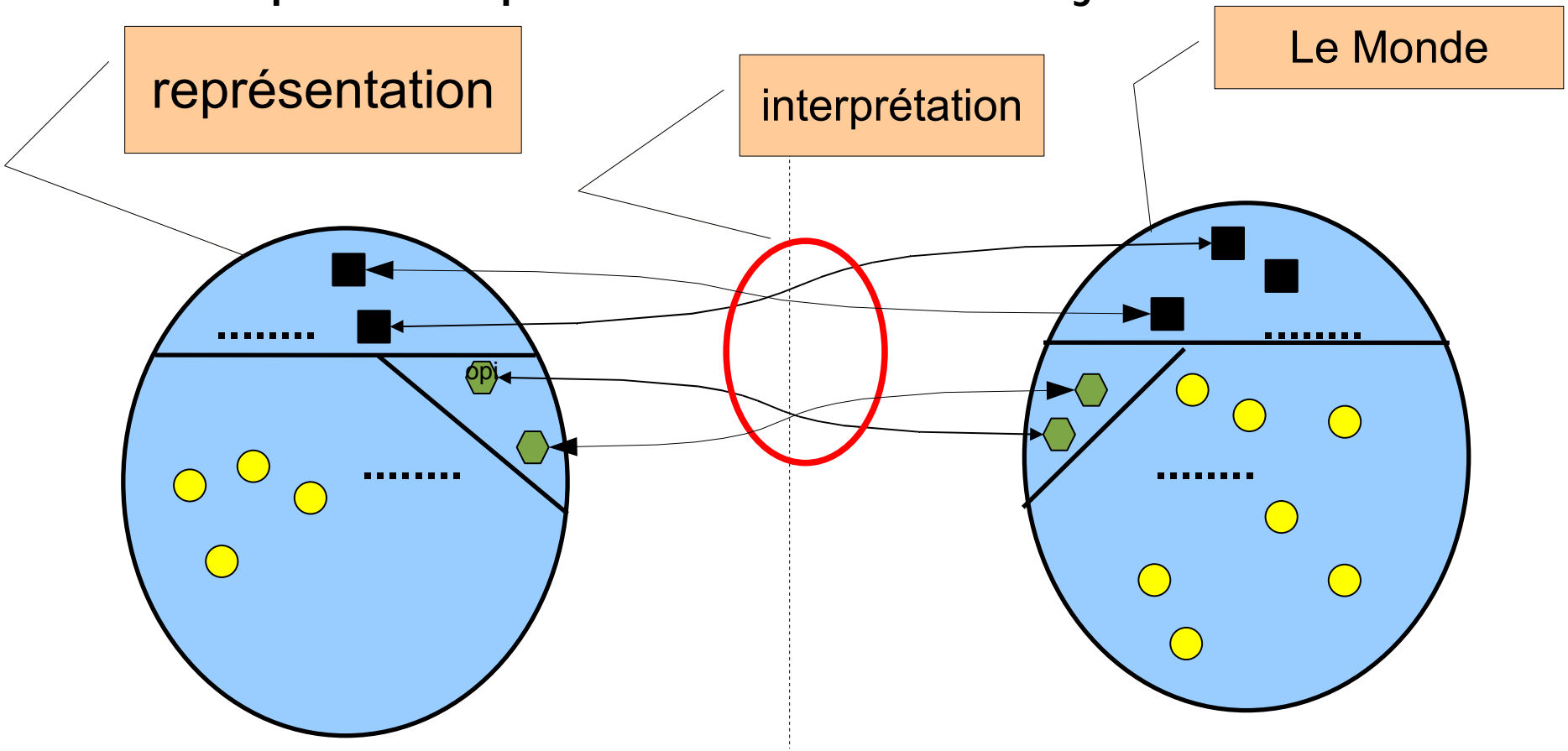
Représenttaion (2)

- Représenter ⇔ Approximer dans le contexte d'une tâche (activité?) particulière
- Représenter ⇔ Structure de symboles pour « décrire » une approximation du « monde » (un modèle du monde) dans le contexte d'une tâche particulière.
- Interpréter une structure (une représentation) ⇔ Composition de l'interprétation des différents symboles la constituant
- Exemple
  - X=(y+2)/z



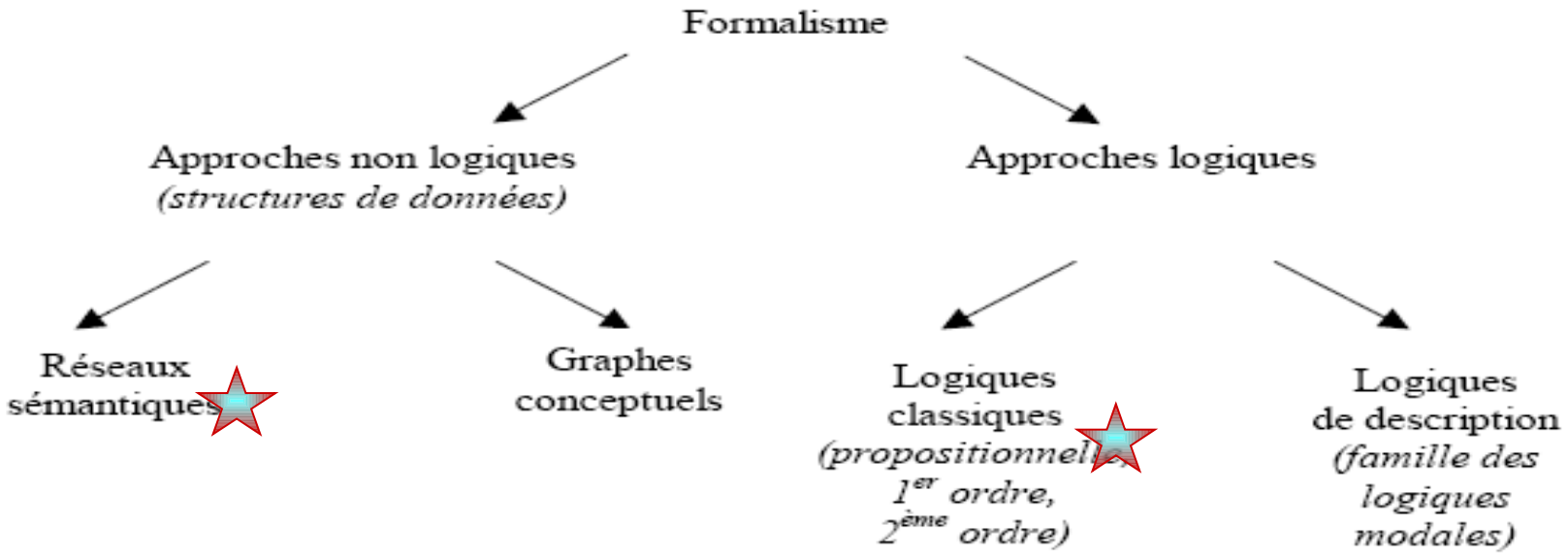
Représentation (3)

La notion d'interprétation présuppose que le modèle (du monde) est constitué d'objets, et que parmi les symboles, il en est qui s'interprètent comme des objets du modèle.



Représentaion de connaissances : les formlismes

- Il existe une pléthore de formalismes.
- Le choix du formalisme à utiliser dépend à la fois du domaine d'application, des opérations à mettre en oeuvre sur ces connaissances (type de problème à resoudre) et ... de la culture du modélisateur.



Premier formalisme : logique propositionnelle

Définition d'une logique formelle

- *Définition 1.1 : D'une façon générale, une* logique formelle L *est constituée :*
  1. *d'une famille de formules:*
     - *formules de L sont les objets sur lesquels portent les raisonnements, et sont habituellement des mots formés sur un certain alphabet*
  2. *d'une notion de preuve :*
     - *raisonnements valides, et sont généralement des suites de formules de L obéissant à des règles syntaxiques particulières*
  3. *d'une notion de vérité ou sémantique :*
     - faisant référence à une interprétation à l'aide de structures extérieures *au monde des formules.*
- *1 & 2 : Syntaxe de la logique*
- *3 : Sémantique de la logique*

Définition et syntaxe(1)

- Syntaxe :
  - Un ensemble de mot représentant des propositions atomiques notées par Xi....
  - Des connecteurs
    - $\neg$
    - $\wedge$
    - $\vee$
    - $\rightarrow$
    - $\leftrightarrow$
    - Séparateurs : les parenthèse avec la notion de porté classique ()
  - les formules propositionnelles L_* , F, G,... Obéit à la syntaxe suivante :
    - $S:: P|...|Q| \neg (F)|(F \wedge G) | (F \vee G) | (F \rightarrow G) | (F \leftrightarrow G)$
  - On designe par $\boldsymbol{L_*}$ l'ensemble des formules produites par cette grammaire.

Définition et syntaxe(2)

- ***Proposition 1.1****: Pour montrer qu'une propriété P est vraie pour toute formule propositionnelle, il suffit de montrer*
  - *que P est vraie pour toute variable Xi*
  - *que, si P est vraie pour F, alors elle est vraie aussi pour ¬F,*
  - *et que, si P est vraie pour F et G, alors elle est vraie aussi pour F∧G, F∨G, F⇒G, et F⇔G.*

- *Par exemple, on déduit de ce qui précède que toute formule qui n'est pas une variable s'écrit de façon unique (aux parenthèses près) sous la forme ¬F, ou F∧G, ou F∨G, ou F→G, ou F↔G.*

- *De cette unicité résulte que, pour définir une application f sur les formules propositionnelles, il suffit de définir f(Xi), puis de définir f(F∧G), . . . , f(F↔G) en fonction de f(F) et f(G). Par exemple, on construit ainsi l'ensemble Var(F) des variables apparaissant dans une formule, défini par :*
  - *Var(Xi)={Xi}, Var(¬ F) = Var(F)*
  - *Var (F * G)=Var(F) ∪ Var (G) avec *={¬F, ou F∧G, ou F∨G, ou F→G, ou F↔G}*

Preuve : objectif

- *L'objectif de la théorie de la preuve est de donner une caractérisation finie des formules valides, en se basant sur un (petit) ensemble de `vérités premières : les axiomes, et un (petit) ensemble de règles permettant de produire toutes les vérités : les règles d'inférence. (saturation des connaissances implicites)*

Preuve : axiomes (1)

- ***Définition 1.2: L'ensemble des sous-formules d'une formule A est le plus petit ensemble tel que***
  - A est une sous-formule de A.
  - Si ($\neg$B) est une sous-formule de A alors B est une sous-formule de A.
  - Si (B $\wedge$ C) est une sous-formule de A alors B et C sont des sous-formules de A.
  - Si (B v C) est une sous-formule de A alors B et C sont des sous-formules de A.
  - Si (B -> C) est une sous-formule de A alors B et C sont des sous-formules de A.
- ***Définition 1.3:*** Une substitution (ou substitution uniforme) associe à une variable propositionnelle p une formule A . Elle est notée [p\A].L'application de [p\A] à une formule B , notée (B)[p\A], est le résultat du remplacement simultané de toutes les occurrences de p dans B par A. (A)[p\B] est appelé une instance de A.

Preuve : Axiomes (de Hilbert) (2)

- Axiomes:
  1) A -> (B -> A)
  2) (A -> B) -> ((A -> (B -> C)) -> (A -> C))
  3) A -> (B -> A ∧ B)
  4) (A ∧ B) -> A
  5) (A ∧ B) -> B
  6) A -> A v B
  7) B -> A v B
  8) (A -> C) -> ((B -> C) -> (A v B -> C))
  9) (A -> B) -> ((A -> ~B) -> ~A)
  10) ¬¬A -> A

Règle d'inférence

- Il existe plusieurs schémas de déduction, ou syllogismes, dans le raisonnement naturel. Les preuves par coupure sont définies par référence à un unique schéma, la *coupure*, ou *modus ponens*.
- ***Définition (règle inférence):*** Soient F, G, H des formules propositionnelles. On dit que H est déduite *par inférence à* partir de F et G si G est la formule F ->H.
- Par exemple, supposons F = X1*v* X2 et G = (X1*v* X2)⇒X1. Alors X1 se déduit par coupure à partir de F et G.

La preuve (suite)

- *Définition (preuve). Soit A une formule. Une preuve de A est une liste finie de formules $(A_1, \ldots, A_n)$ t.q.*
  - $A_n = A$
  - *pour i = 1, ..., n, la formule A_i est*
    - *soit l'instance d'un axiome,*
    - *soit obtenue par application de la règle de **Modus Ponens** à partir de deux prémisses A_j, A_k précédant Ai dans la liste.*
- *Définition (déduction) : Une déduction d'une formule A à partir d'hypothèses $B_1,\ldots,B_m$ (noté $B_1,\ldots,B_m \vdash A$) est une liste finie de formules $(A_1, \ldots, A_n)$ t.q.*
  - $A_n = A$
  - *pour i = 1, ...,n, la formule A_i est*
    - *soit un axiome,*
    - *soit égal à une des hypothèses B_j,*
    - *soit obtenue par application de la règle de **Modus Ponens** à partir de deux prémisses A_j, A_k précédant A_i dans la liste.*

Exemple de preuve

- Exemples de preuve
  - Preuve de A -> A :
    1. A -> (A -> A)
       instance de A -> (B -> A)
    2. (A -> (A -> A)) -> ((A -> ((A -> A) -> A)) -> (A -> A))
       instance de (A -> B) -> ((A -> (B -> C)) -> (A -> C))
    3. ((A -> ((A -> A) -> A)) -> (A -> A))
       obtenue par Modus Ponens à partir de 1. et 2.
    4. A -> ((A -> A) -> A)
       instance de A -> (B -> A)
    5. A -> A
       obtenue par Modus Ponens à partir de 3. et 4.

Sémantique

- Définir une sémantique pour une logique, (ou pour un formalisme en général) c'est attacher à chaque formule (ou toute production du formalisme) une interprétation faisant référence à un contexte externe. Ici le domaine de discours qu'on veut modéliser.

- *Le but de la logique propositionnelle est de modéliser le raisonnement naturel — ou tout au moins l'un des aspects de celui-ci sur des valeurs de vérité des propositions.*
  - Attribuer une valeur de vérité qui est soit « vrai » ou « faux » à une formule atomique donnée.
  - Attribuer une fonctionn de vérité des connecteurs par rapport au raisonnement naturel qu'ils sont supposés mimer (tableau de vérité).
  - Ceci grace à la proposition 1.1
  - => on peut claculer la valeur de vérité de n'importe quel formule

Interprétation ou affectation

- On appelle affectation de valeurs booléennes, ou simplement *affectation*, pour une formule propositionnelle F une fonction *v* de Var∗ dans {0, 1} dont le domaine inclut Var(F) ; l'é*valuation* de F en *v*, notée $\text{eval}_v(F)$, est alors définie récursivement par les clauses
  - $\text{eval}_v(Xi) = v(Xi)$, $\text{eval}_v(\neg F) = f_\neg(\text{eval}v(F))$, $\text{eval}v(F\ c\ G) = f_c(\text{eval}_v(F), \text{eval}_v(G))$,

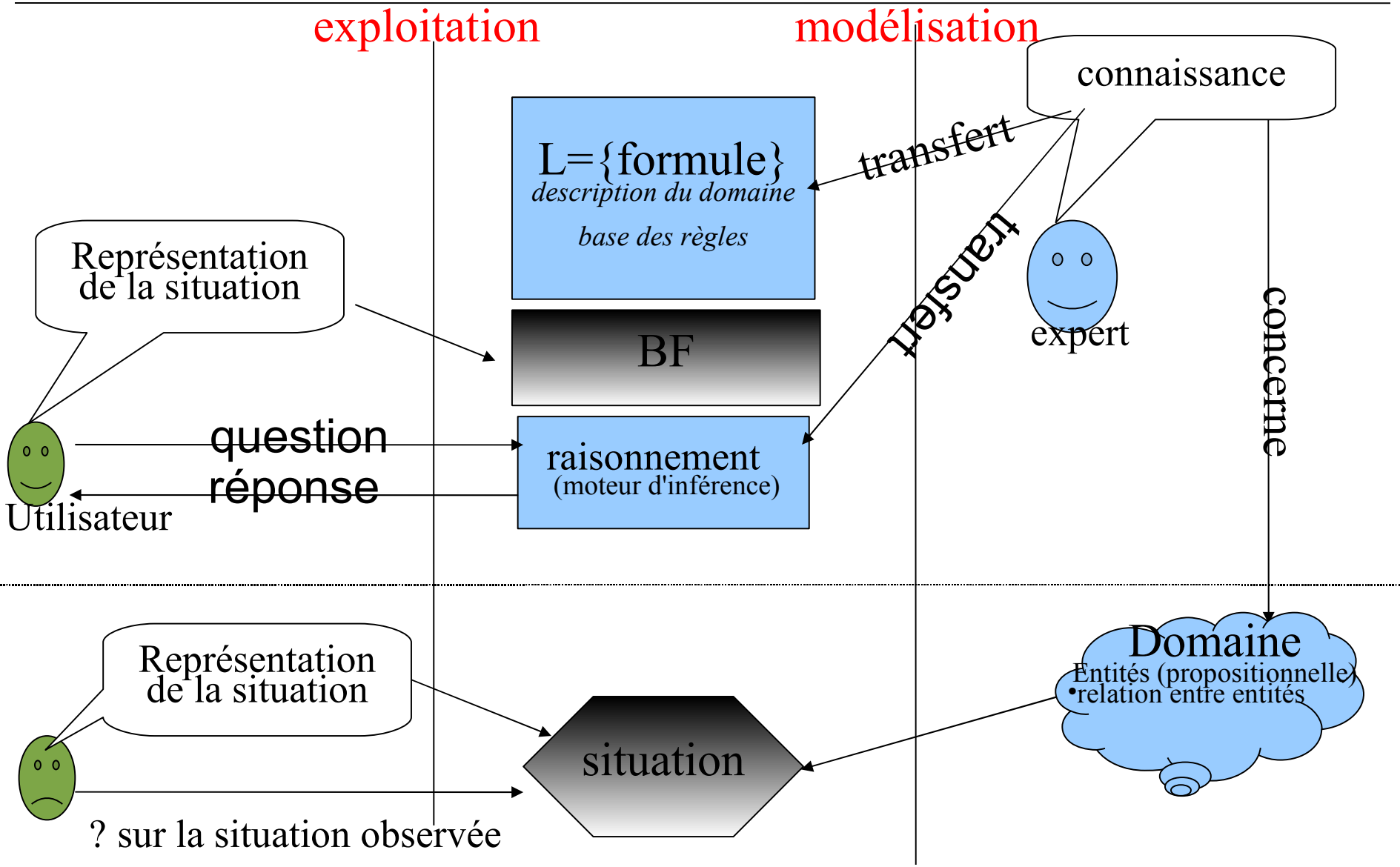
Satisfaction, validité, équivalence, conséquence

- Soit L une logique
  - F et G deux formules de la logiques et T une famille de formule.
  - Soit I la classe de toutes les interprétations (affectation) possible avec v une affectation des instance d'interprétation (v_I)
- Statifesabilité: v |= F ssi $eval_v$(F) = « vrai » .
- Validité : |= F ssi v |= F pour toute affection v de I.
- Equivalence et conséquence : On dit que F, G sont *équivalentes* si v |= F équivaut à v |= G; on dit que G est *conséquence* de F si v |= F entraîne *v* |= G.
- Par exemple, la formule X∨¬X est valide, puisque satisfaite par toute affectation

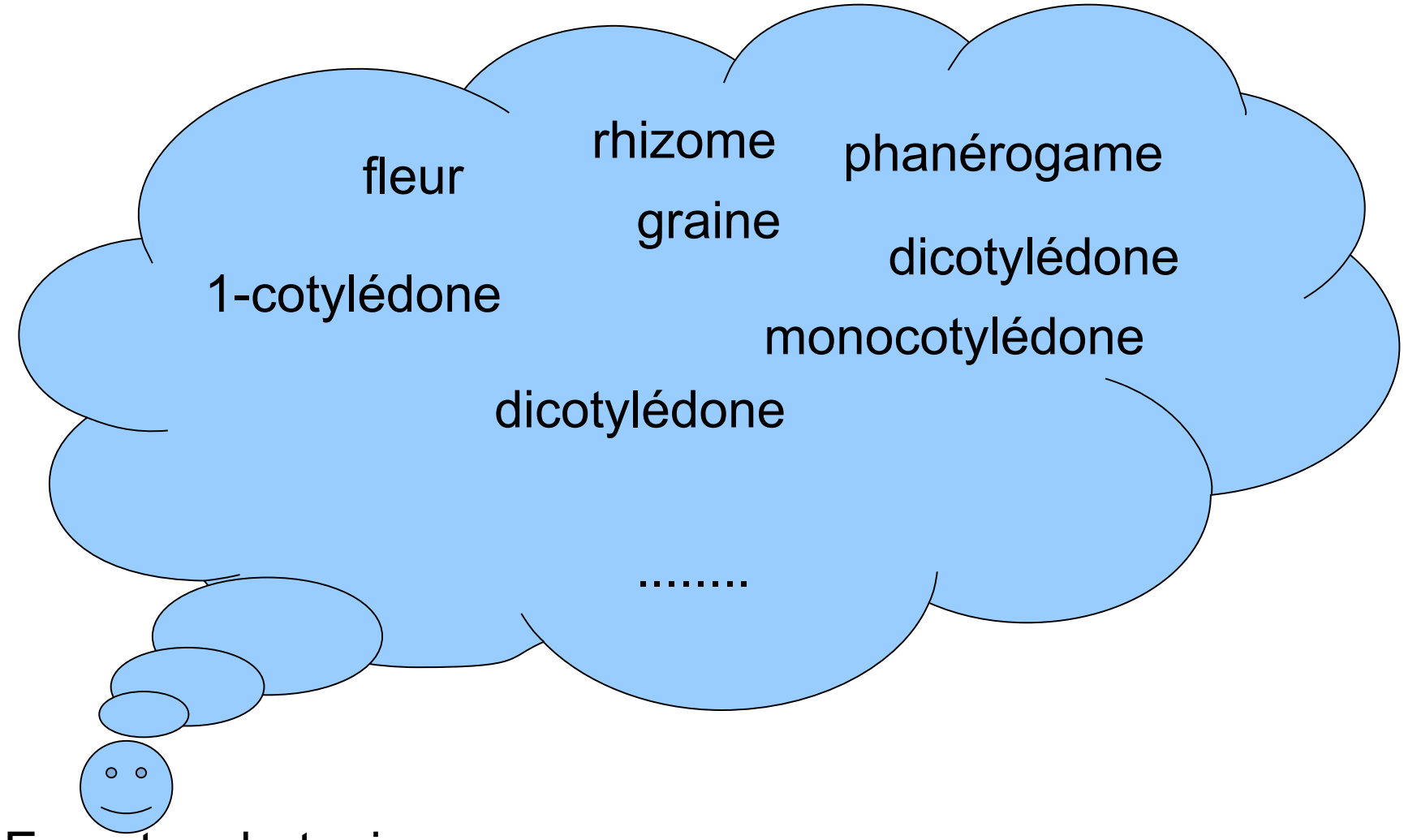
Complétude et correction

- Notez que le système de preuve est interne aux syntaxes
  - Axiomes syntaxique + règle d'inférences (pas interprétation)
- Une sémantique est données pour les proposition ainsi que pour les connecteurs en fonction d'un raisonnement naturelle qu'on veux capter.
- Notez également que la validité (etc.) fait appel au monde externe les interprétations.
- Il est essentiel pour l'automatisation du raisonement que ces deux monde dual soit conforme l'un vers l'autre. Deux proprité :
  - Correction : tous ce qui est syntaxiquement porouvable doit être valide : "|- *F entraîne* |=*F*"
  - Complétude : tous ce qui est valide doit être prouvable "|=F entraine |-F"

Recentrons nos propos



Exemple


fleur
rhizome
phanérogame
graine
dicotylédone
1-cotylédone
monocotylédone
dicotylédone
.....
Expert en botanique

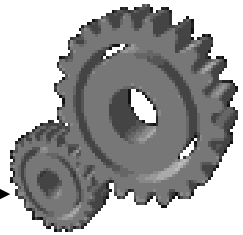
Exemple : expert en botanique(2)

- L'Expert voit les choses comme ça :
- Les plantes ont des propriétés (presentes (vrai) ou abscentes)
  - Exemple :
    - À fleur ou sans fleur
    - À graine ou sans graine
    - Pour celles qui sont à graine on distingue : graine nue et non
- Les plantes sont classées selon leurs propriétés :
  - Exemple :
    - Si une plante a des fleurs et des graines elle appartient à la famille des phanérogames.
    - Etc.
- On a des propositions qui décrivent des entités ainsi que des règles qui régissent le lien entre des phrases (ou des formules sur les propositions).

Example : expert en botanique

J'ai une plante qui a une fleur et elle produit des graines aussi est ce c'est une **phanérogame** ?

graine,fleur



Fleur ∧ graine→ phanérogame

graine, fleur, phanérogame

J'ai une plante qui a une fleur et elle produit des graines aussi qu'est ce que vous pouvez me dire

pas1

fleur ∧ graine→ phanérogame, fleur,graine

Modus ponens

pas2

fleur ∧ graine→ phanérogame, fleur,graine, phanérogame

Systèmes experts propositionnels

- Un système expert = Base de connaissances + moteur d'inférence:
- Une base de connaissances elle-même composée :
  - d'une base de règles qui modélise la connaissance du domaine considéré ;
  - d'une base de faits qui contient les informations concernant le cas que l'on est en train de traiter.
- Moteur d'inférence :
  - Un module qui implémente des algorithmes capables de raisonner à partir des informations contenues dans la base de connaissances, de faire des déductions, etc.
- Note : L'objectif est de résoudre des problèmes. Les données du problème sont les observations (les faits) et on veut expliciter et déduire ce qui est implicite dans les règles=> modus ponens

Systèmes experts propositionnels : les règles (1)

- Elle rassemble la connaissance et le savoir-faire de l'expert. Elle n'évolue donc pas au cours d'une session de travail.

- Une règle est de la forme:

  <conjonction de formule> $\rightarrow$ *<conclusion>*
  où les conclusions sont de la forme
  <Fait> = <valeur>.

- Exemples :
  - ***si population > 200000 et ville-universitaire alors cinémaArtEtEssai=vrai***
  - ***si revenu-imposable = connu et quotient-familial = connu alors calculerMontantImpot=vrai***
  - Le dernier exemple montre comment un système expert peut être utilisé en association avec des programmes classiques.
    - On peut supposer que le passage à la valeur vrai du fait booléen *calculerMontantImpot* déclenche une procédure calculant le montant de l'impôt en question et l'attribuant au fait réel *MontantImpôt*.

Systèmes experts propositionnels : les règles (2)

- La signification Logique est la conjonction de la signification logique de chacune des règles. En particulier, on peut aisément coder dans le formalisme précédent des règles de la forme :
  - si A ou B alors C
    ou
  - si A alors B et C
    Il n'en est par contre pas de même de
  - ~~si A alors B ou C~~
- Déclenchable : une règle est déclenchable quand la pré-condition est évaluée à vrai en accord avec l'interprétation donnée dans la base des faits.

La base des faits

- Ce sont en quelque sorte les formules (les faits) dont le contenu (valeur de vérité) est connu par le système (ou l'utilisateur).
- On peut dire que les faits consistent à attribuer une interprétation qui correspond à une instance du domaine considéré.
- C'est un espace de travail, modifié par les nouveaux faits déduits par le raisonnement. Toute formule déduite ou prouvée est considérée comme un fait.

Moteur d'inférence

- Essentiellement un raisonnement par règles d'inférence (modus ponens).
- On désigne par :
  - BR un {} des règles.
  - BF un {} des faits.
  - Et F un fait à prouver (appelé but)
- Une moteur d'inférence doit
  - BR U BF |– F ?
- En utilisant les règles d'inférences.

Trois algorithmes d'inférence : R U F ⊨ G ?

- On distingue essentiellement trois modes principaux de fonctionnement des moteurs d'inférences :
  - le chaînage avant
  - le chaînage arrière
  - et le chaînage mixte

Chaînage avant

- pour déduire un fait particulier:
  - on déclenche les règles dont les prémisses sont connues
  - jusqu'à
    - ce que le fait à déduire soit également connu
    - ou qu'aucune règle ne soit plus déclenchable.

```
ALGORITHME DU CHAINAGE AVANT
ENTREE : BF, BR, F
DEBUT
   TANT QUE F n'est pas dans  BF ET
   QU'il existe dans BR une règle applicable FAIRE
      choisir une règle applicable  R (étape de résolution de
            conflits, utilisation d'heuristiques, de métarègles)
      BR = BR - R  (désactivation de R)
      BF = BF union concl(R) (déclenchement de la règle R, sa
            conclusion est rajoutée à la base de faits)
   FIN DU TANT QUE
   SI F appartient à BF ALORS
      F est établi
   SINON
      F n'est pas établi
```

Exercices

a) $fleur \wedge graine \rightarrow phanérogame$

b) $phanérogame \wedge graine\text{-}nue \rightarrow sapin$

c) $phanérogame \wedge 1\text{-}cotylédone \rightarrow monocotylédone$

d) $phanérogame \wedge 2\text{-}cotylédone \rightarrow dicotylédone$

e) $monocotylédone \wedge rhizome \rightarrow muguet$

f) $dicotylédone \rightarrow anémone$

g) $monocotylédone \wedge \neg rhizome \rightarrow lilas$

h) $feuille \wedge fleur \rightarrow cryptogame$

i) $cryptogame \wedge \neg racine \rightarrow mousse$

j) $cryptogame \wedge racine \rightarrow fougère$

k) $\neg feuille \wedge plante \rightarrow thallophyte$

l) $thallophyte \wedge chlorophylle \rightarrow algue$

m) $thallophyte \wedge \neg chlorophylle \rightarrow champignon$

n) $\neg feuille \wedge \neg fleur \wedge \neg plante \rightarrow colibacille$

déterminer la plante ayant les caractéristiques suivantes: *rhizome, fleur, graine, 1-cotylédone?*

Chaînage avant : propriété

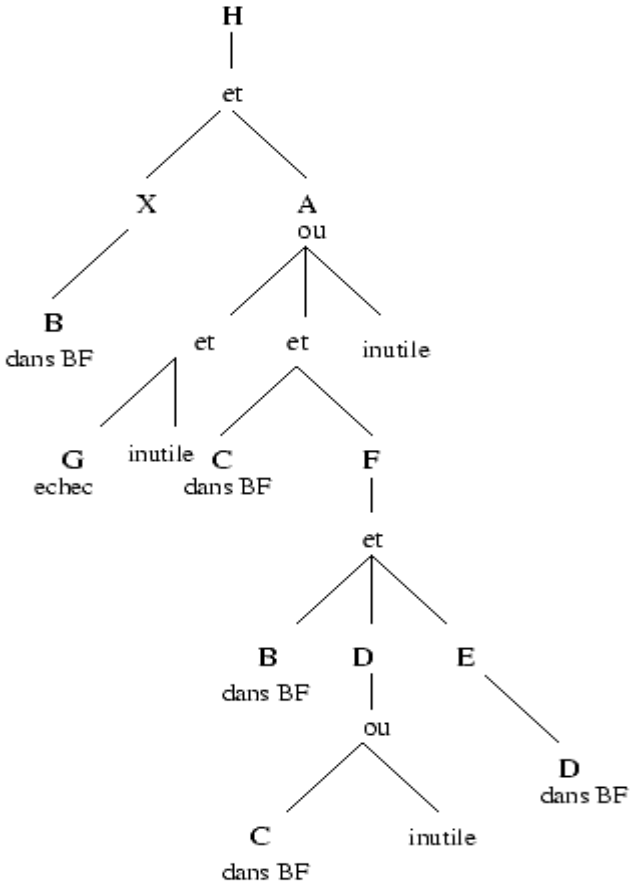
- Cet algorithme peut être très pénalisant pour certaines bases
- L'algorithme de chaînage avant s'arrête toujours
  - si l'on utilise des règles dont les conclusions peuvent être des faits négatifs, pour tout fait F, il peut se produire 4 situations :
    - F ∈ BF : le fait est établi.
    - ¬ F ∈ BF : la négation du fait est établie.
    - ni F, ni ¬ F ne sont dans BF : le système ne déduit rien à propos du fait. C'est une troisième valeur de vérité qui apparaît naturellement et dont l'interprétation peut être diverse selon les cas.
    - F et ¬ F Î BF : la base est incohérente. On peut prévoir un fait Base_incohérente et une méta-règle de la forme : si il existe un fait qui appartient, ainsi que sa négation, à BF alors Base_incohérente.

Chaînage arrière

- Le mécanisme de chaînage arrière consiste à partir du fait que l'on souhaite établir, à rechercher toutes les règles qui concluent sur ce fait, à établir la liste des faits qu'il suffit de prouver pour qu'elles puissent se déclencher puis à appliquer récursivement le même mécanisme aux faits contenus dans ces listes.
- Ces faits deviennent à leur tour des faits à démontrer. La récurssion s'arrête quand le but à établir appartient à la base des faits.

Arbre ET-OU

- Soit BF = {B,C}, Fait = H et BR composée des règles :
  - B $\wedge$ D et E $\rightarrow$ F
  - G $\wedge$ D $\rightarrow$ A
  - C $\wedge$ F $\rightarrow$ A
  - B $\rightarrow$ X
  - D $\rightarrow$ E
  - X $\wedge$ A $\rightarrow$ H
  - C $\rightarrow$ D
  - X $\wedge$ C $\rightarrow$ A
  - X $\wedge$ B $\rightarrow$ D



Algorithme de chaînage

- Problème de terminaison :
  - S'il n'a pas de mémoire avec deux règle récursive deux fait apparaissant l'un et l'autre dans les pré-condition mutuel.
- Avantages le chaînage arrière:
  - Gain en rapidité on déclenche que les règles qui on un rapport de dépendance avec le fait à prouvé:
  - constitue une approche intéressante pour définir des systèmes d'interaction puissant avec l'utilisateur:
    - Principes: définir des faits comme « *demandables* » (i.e on peut demander les valeur de vérité à l'utilisateur.
    - Si on a un fait à prouvé :
    - On régresse ce fait et on favorise les règle où les condition contiennent le plus de fait « *demandable* » ou encore des but avec des faits « *demandable* ».
    - Ceci permet de calculer l'ensemble minimal des questions pertinente pour prouvé des faits.

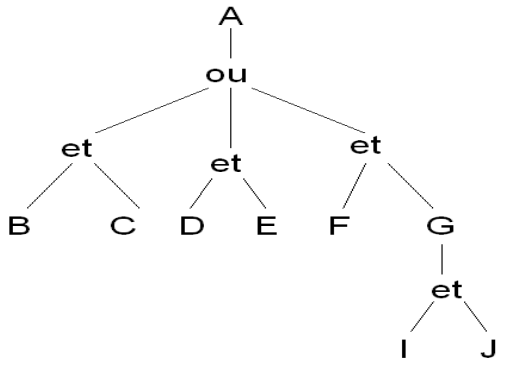
Chaînage arrière avec fait demandable

- $\underline{B} \wedge C \rightarrow A$
- $\underline{D} \wedge E \rightarrow A$
- $\underline{F} \wedge G \rightarrow A$
- $\underline{I} \wedge J \rightarrow G$
- $J \rightarrow \neg E$

- On suppose que les faits B, D, F et I sont les seuls faits demandables.
- La mémoire de travail est initialisée avec l'information J est vrai.
- La question posée au système est : A est-il vrai ? Quelles sont les questions pertinentes à poser à l'utilisateur ?

Chaînage arrière avec fait demandable

- $\underline{B} \wedge C \rightarrow A$
- $\underline{D} \wedge E \rightarrow A$
- $\underline{F} \wedge G \rightarrow A$
- $\underline{I} \wedge J \rightarrow G$
- $J \rightarrow \neg E$



- "B est-il vrai" n'est pas un question pertinente.
  - aucune règle ne conclut sur le fait C qui n'est pas non plus demandable.
  - le fait B ne peut être utilisé que conjointement à C, la valeur de vérité de B n'apportera aucune information sur celle de A
- "D est-il vrai" n'est pas non plus pertinente. En effet, comme on sait que J est vrai, que cela implique que E est faux et que D n'est utilisé que conjointement à E.
- "F est-il vrai" est pertinente. En effet, le fait G est encore déductible. Mais si la réponse à cette question est NON,
- "I est-il vrai" n'est plus pertinente car la valeur de G ne sert plus à rien

Deuxième formalisme: les réseaux sémantiques.

Logiques modales

- Les chose ne sont pas si simple.
- Il est facile de mettre en place un calcule sur les valeur de vérité
- C'est loin de la réalité de la forme de connaissance au quel on fait face pour prendre des décision « heureusement ;-) !!!».

Comment exprimer :

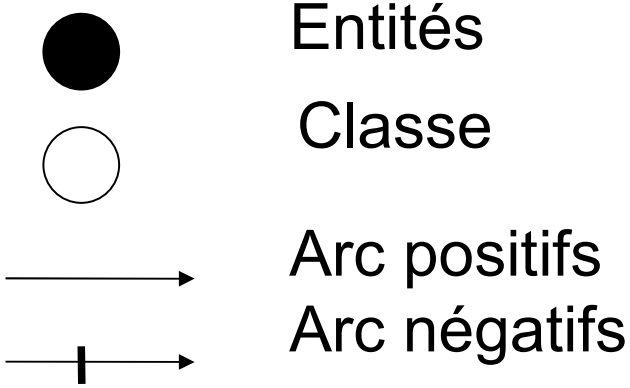
« Lionel affirme que la constitution européenne est une bonne chose »
« Il y a peu de chances que la vie existe ailleurs que sur Terre »
« Généralement, les mollusques ont une coquille »

Pourquoi un deuxième formalisme

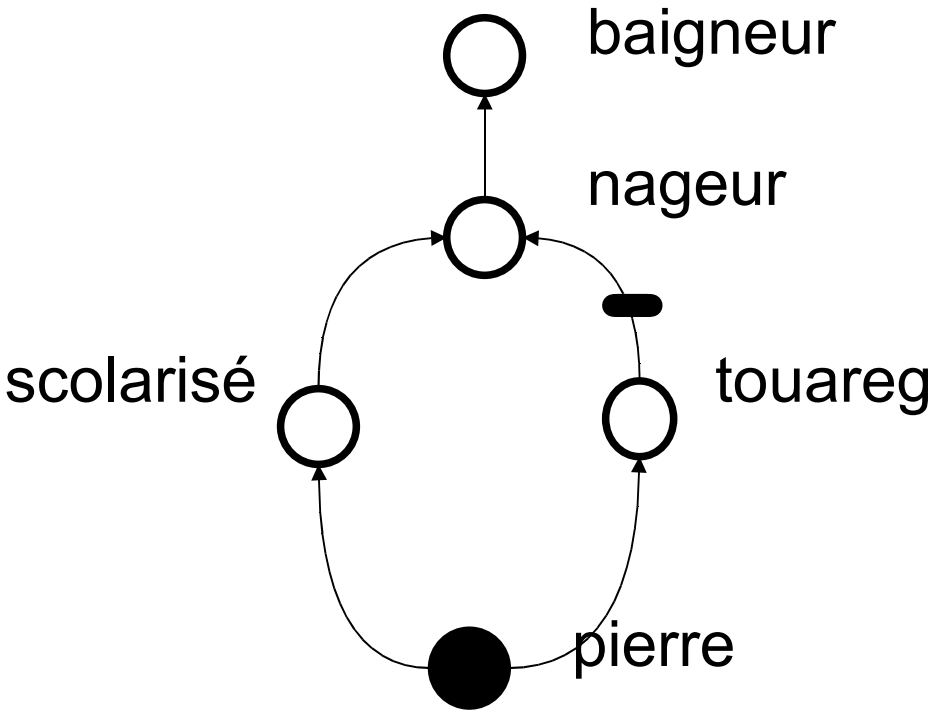
- Y a pas que les logiques comme formalisme pour accueillir les connaissances.
- Difficultés de la représentation à base de modèle logique
  - Système décidable ⇔ logique des propositions, mais … temps de décision exponentiels !
  - Autres logiques ⇔ plus expressives, mais semi-décidables, voire indécidables !
- Comment rendre les inférences efficaces ?
  - Restreindre la logique
  - Abandonner l'exigence de complétude !
- Rendre + facile la « lecture » de la représentation ?

Les réseaux sémantiques :syntaxe

- Un réseau sémantique est un graphe orienté acyclique dont :
  - les noeuds sont soit des classes d'entités, soit des entités spécifiques,
  - les arcs sont de deux types arcs positifs et arcs négatifs.
- De plus :
  - Une entité n'est jamais à l'extrémité d'un arc.
- Notation Graphique :



Example



Sémantique

- **Interprétation**
  - Un arc positif reliant un noeud `C` à un noeud `D` signifie : (un) `C` est généralement un `D`.
  - Un arc négatif reliant un noeud `C` à un noeud `D` signifie : (un) `C` n'est généralement pas un `D`.
  - L'article indéfini est présent dans le cas où `C` est une classe.
- On comprend aisément que la finalité de ce type de modèle est de *classer* entités et classes en supportant des *exceptions*.
- Dans la suite, on remarquera qu'algorithmiquement il n'y a pas de différence entre entité et classe. Cependant cette distinction peut être exploitée dans une interaction entre l'utilisateur et le système lors des réponses aux questions et des modifications des connaissances.

Preuve : questionnement

- La question évidente que l'on désire poser est l'existence d'une relation de spécialisation que peut entretenir une classe C envers les autres classes du réseau (ou une entité envers les autres classes) avec trois réponses possibles:
  1. (Un) C est généralement un D.
  2. (Un) C n'est généralement pas un D.
  3. Le réseau ne permet pas de conclure.
- Ce type de question possède donc un unique paramètre C.

manipulation des graphes : Premier algorithme naïf (1)

- Si C est généralement un D et que D est généralement un E.
  - On a envie de conclure que C est généralement un E.
- Si C est généralement un D et que D n'est pas généralement un E.
  - On a envie de conclure que C n'est pas généralement un E.
- Pour répondre a ce type de question une idée consiste à saturer le graphe en terme de relation positive et négatif.
  - Ceci revient à une fermeture transitive du graphe sur les arcs.
  - Notez bien ici : syntaxe est un formalisme (les graphes) sur le quel on sait des choses (acyclique, les algorithmes etc..)
  - On a donné une sémantique et on prcède aà un mécanisme de raisonnement en s'inspirant de ce qu'on peut faire sur les graphe
-  N'hésitez pas à créer vos formalismes

manipulation des graphes : Premier algorithme naïf (1)

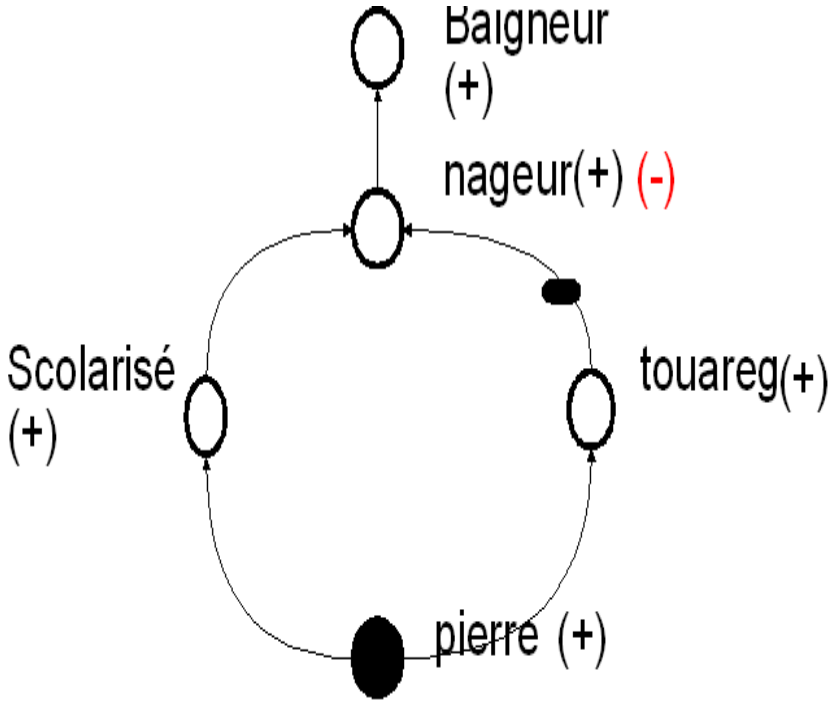
// On appelle ici suivant positif
// un suivant par un arc positif
Pile=∅;
Marquer les suivants positifs de C;
Empiler les suivants positifs de C;
Tant que la pile est non vide **faire**
Dépiler un noeud D;
Pour tout suivant positif E de D non marqué **faire**
E est un successeur positif de C;
Marquer E;
Empiler E;
Fin pour
Fin tant que

// On appelle ici suivant négatif
// un suivant par un arc négatif

Pour tout successeur positif E de C **faire**
Pour tout suivant négatif D de E **faire**
D est un successeur négatif de C à D;
Fin pour
Fin pour

ambiguïté

- Examinons le résultat de ces deux algorihtmes sur l'exemple appliqué à Pierre comme noeud de notre étude.
- Le marquage noire pour l'algo positive
- Le marquage rouge pour algo négative



1. Pierre est un élève scolarisé, touareg, nageur et baigneur.

2. Pierre n'est pas un nageur.

Ambiguïté : trie topologique

- Pour traiter correctement l'ambiguïté, il importe de traiter simultanément les conclusions négatives et positives.
- L'Idée est de marquer les successeurs de C positive, négatif, ambiguë ou manque d'information.
- Mais on marque un noeud N ssi on a déjà marqué ses prédécesseurs pour pouvoir en décidé.
- Solution:
  - Commencer par un tri topologique du graphe
  - Notez que le graphe est acyclique donc un tel tri est possible

Ambiguïté: Trie topologique

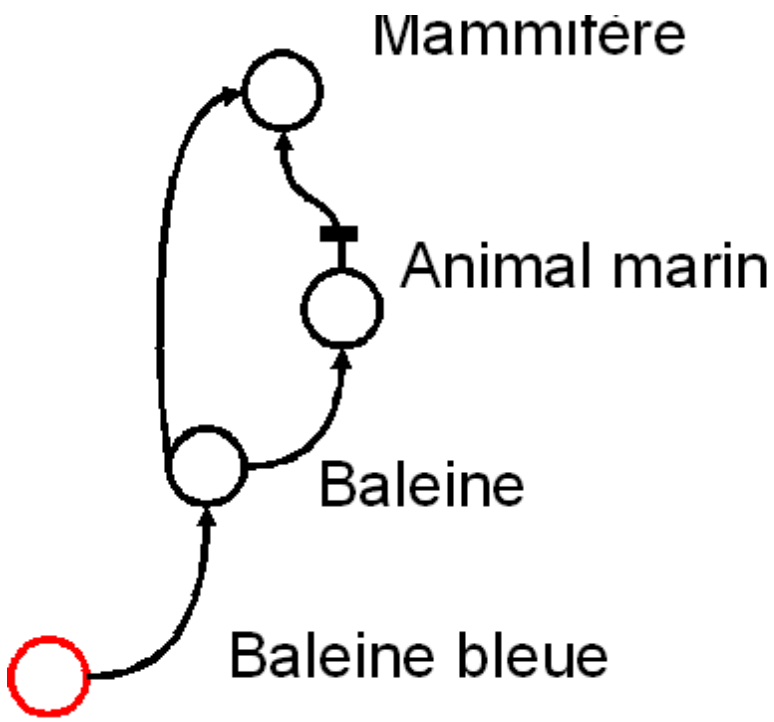
ind=1;
Pile=∅;
Empiler C;
Tant que la pile est non vide **faire**
    Dépiler un noeud D;
    Indice de D=ind;
    ind++;
    Pour tout arc de D à E **faire**
        Supprimer l'arc de D à E;
        Si E n'est plus l'extrémité d'aucun arc **alors**
        Empiler E;
        Fin si
    Fin pour
Fin tant que

Ambiguïté: algorithme de marquage

On affecte à C l'état positif
// L'examen des successeurs de C se fait
// dans l'ordre topologique
Pour tout successeur D de C **faire**
Si pour tout arc de E vers D,
l'état de E n'est pas positif **alors**
D <= {¤}
Si ∃ un arc positif de E vers D et
un arc négatif de F vers D et
les états de E et de F sont positifs **alors**
D <= {~}
Sinon si ∃ un arc négatif de E vers D et
l'état de E est positif **alors**
D<= {-} l'état négatif
Sinon
D <= {+}
Fin pour

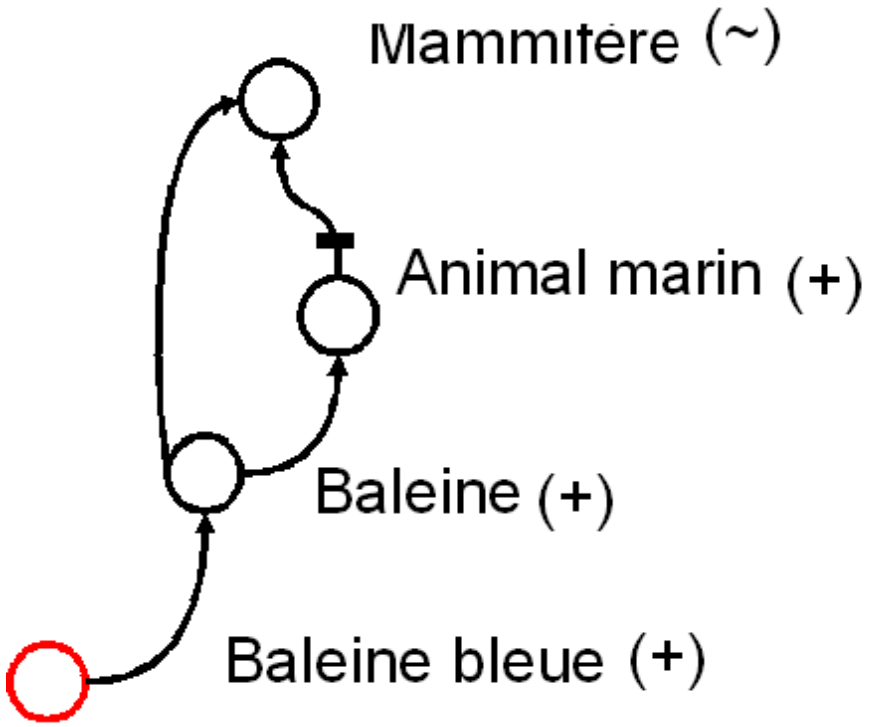
La spécificité ou un traitement des exceptions

- Voyons ce cas : baleine bleue est un mammifère?



La spécificité ou un traitement des exceptions

- Si on applique l'algorithme précédent

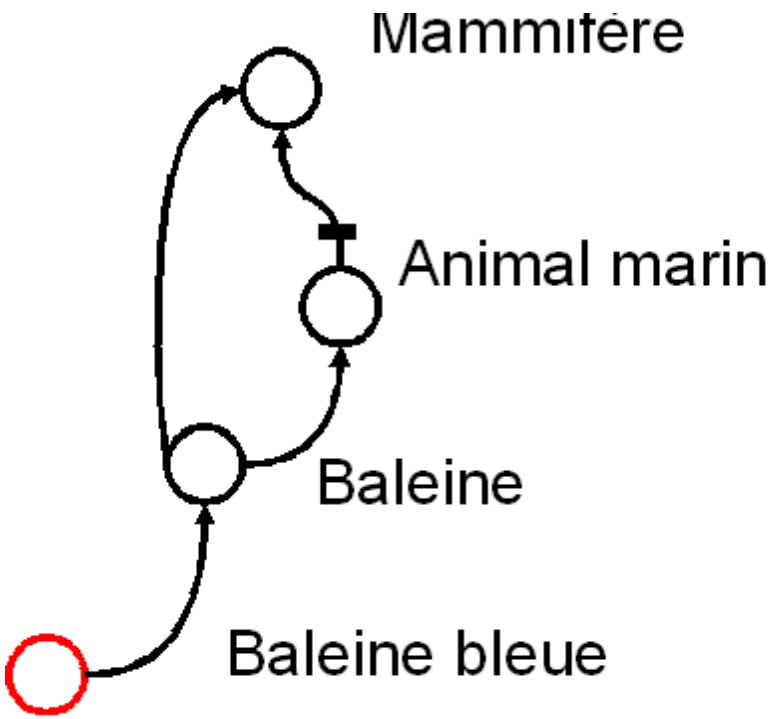


Question > est ce que la **baleine bleue** est un **Mammifère?**
réponse > c'est ambiguë

La spécificité ou un traitement des exceptions

- Mais ce n'est pas dans l'esprit du modélisateur il veux plutôt dire que...

"Les animaux marins ne sont généralement pas des mammifères excepté les baleines".



Les exceptions

- On veux pouvoir identifier ce type de connaissance:

- Autrement dit, l'ambiguïté apparente est levée en considérant que l'arc baleine-mammifère est plus spécifique que le chemin (baleine)-(animal marin)-(mammifère).

Analysons ce que c'est une spécificité

- Il s'agit maintenant de définir précisément ce qu'est la spécificité dans le modèle:

- Réexaminon un cas d'ambiguïté:
  - *Un noeud D est ambiguë pour un noeud C ssi ∃ A et B telque A(+) et B(+) avec*
    - *A -|-> D*
    - *B--> D*
    - *C=>A et C=>B*

- Examinons les relations qu'entretiennent ces noeuds positives *A* et *B*

- Revenons à notre Examples mammifère est ambiguë pour C :
  - Équivalent à A et B? Baleine et animal marin
  - Si on supprime baleine alors animal marin n'est plus accessible de baleine bleue
  - On peut conclure que animal marin n'est pas pertinent pour ...

Algo : principe (pour l'algo en détail voir TD N° 1)

- **cette version se distingue de la précédente uniquement par la sélection des noeuds pertinents**
- **D'autre part, pour détecter la condition de pertinence, il suffit de vérifier l'accessibilité du noeud en question à partir de C par un chemin d'arcs positifs passant par des noeuds positifs sans rencontrer d'autres noeuds de Ens.**
- **Les marqueur sont attribué en fonction de de ça comme suit**
- **Seul les noeuds prédécesseur pertinent entre en considération pour marquer un noeud.**
- **Et donc on considère que les chemins spécifique**

conclusion

- Pour une résolution automatique de problème une représentation explicite de connaissance s'impose.
- On a vue deux formalismes de représentation de connaissances
  - Logique formelle : cas propositionnelle
  - Les réseaux sémantiques
- Un formalisme est composé d'une syntaxe et un ensemble de propriété de cette syntaxe qui permet de réaliser le type d'exploitation voulus (axioms ou encore les proprité de graphe) .
-

conclusion

- Utiliser un formalisme pour représenté un monde consiste à
  - associer les symboles de la syntaxe avec des entités du monde en question.
  - Il faut que ça coïncide et que formalise support des traitement qui nous permette de reproduire le raisonnement naturelle.
  - Et ceci en vérifiant deux propriétés importantes :
    - Correction
    - Complétude

Ce cours est basé sur...

- *Cours de logique pour l'IA de Serge Haddad*
  - *Que je remercie*
- *Cours de SMA l'AFIA de Amal El fallah segrouchni*
  - *Que je remercie pour le schémas des trois dimensions des SMAs*
- *Cours d'intelligence artificielle Systèmes experts de François Denis*

Référence bibliographique Utiles

- Système Expert:
  - *J. GIARRATANO, G. RILEY, Expert systems Principles and Practice, PWS Publishing, Boston, 1993.*
  - *DELAHAYE, Jean-Paul, Systèmes experts : organisation et programmation des bases de connaissances en calcul propositionnel, Eyrolles, 1987*
- *Réseaux sémantique et autre formalisme :*
  - *D. Kayser "La représentation des connaissances" Hermès collection informatique Paris 1997*
  - *J.F. Sowa "Knowledge Representation: Logical, Philosophical, and ComputationalFoundations" Brooks Cole Publishing Co., Pacific Grove, CA. 2000*
  - *.....*