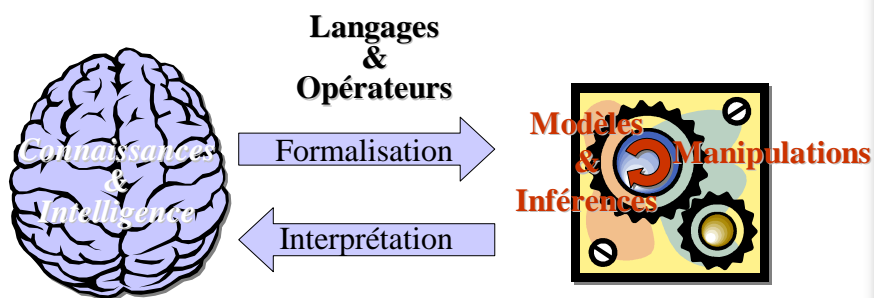


Langages et outils de formalisation

- ❖ Le but de la représentation des connaissances est de rendre compte d'un domaine particulier de telle sorte que cette représentation soit manipulable par la machine.
- ❖ Nécessité de langages de modélisation
- ❖ La sémantique des langages étant fixée les opérations possibles sont contrôlées



❖ Langages de formalisation

- ▼ Expressivité
- ▼ Dilemme expressivité / performance
- ▼ Expressivité réduite : aide à la décision

❖ Grandes familles

- ▼ Logiques (propositions, prédicats, descriptions)
- ▼ Graphes et Réseaux sémantiques
- ▼ Langages à objets

❖ Problème du choix d'un langage.

❖ Objectifs:

- ▼ tour d'horizon des langages (survol)
- ▼ accent sur leurs particularités
- ▼ pointeurs vers outils

❖ Exemple d'un système symbolique ⁽¹⁶⁾

- ▼ symboles : **u p e**
- ▼ axiome (posé comme valide) : **x p u e u x**
- ▼ règle de production : si **x p y e z** existe alors **x p y u e z u** existe aussi (théorème)
- ▼ exemple de chaîne bien formée **uuu p uu e uuuuu**
- ▼ chaîne mal formée **uu p uu e u**
- ▼ interprétation ?
 - **a + b = c**
 - isomorphisme, comportement système symbolique et domaine d'observation
 - **autre interprétation a=c-b** (non unicité)
- ▼ axiome : **x p u e x**
 - production **uu p u e uu**
 - interprétation

- ❖ langage = syntaxe + sémantique
 - ▼ syntaxe = règles selon lesquelles les éléments d'un langage sont assemblés
 - ▼ sémantique = règles selon lesquelles les expressions syntaxiques se voient assignées un sens
- ❖ validité formelle \neq vérité factuelle
 - ▼ Tout homme est un automate donc Quelque homme est un automate
 - ▼ c'est une inférence valide
 - ▼ ce n'est pas une vérité
- ❖ Interprétation
 - ▼ isomorphisme avec un modèle
 - ▼ GIGO: garbage in garbage out

« « » »

5

- ❖ Système symbolique
 - ▼ symboles logiques \supset, \mathbf{f} plus $\mathbf{a, b, c, \dots}$ plus $(,)$
 - ▼ formule $\rightarrow \mathbf{f} \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{c} \mid \dots \mid (\text{formule} \supset \text{formule})$
 - ▼ ex: $(\mathbf{a} \supset (\mathbf{b} \supset \mathbf{c}))$
- ❖ Axiomes:
 - ▼ $(\mathbf{a} \supset (\mathbf{b} \supset \mathbf{a}))$
 - ▼ $((\mathbf{c} \supset (\mathbf{a} \supset \mathbf{b})) \supset ((\mathbf{c} \supset \mathbf{a}) \supset (\mathbf{c} \supset \mathbf{b})))$
 - ▼ $((\mathbf{a} \supset \mathbf{f}) \supset \mathbf{f}) \supset \mathbf{a}$
- ❖ Ajouts (notations / abréviations " formules $\mathbf{x, y}$)
 - ▼ $\mathbf{v} := (\mathbf{f} \supset \mathbf{f})$
 - ▼ $\neg \mathbf{x}$ désigne $(\mathbf{x} \supset \mathbf{f})$
 - ▼ $\mathbf{x} \vee \mathbf{y}$ désigne $((\mathbf{x} \supset \mathbf{y}) \supset \mathbf{y})$
 - ▼ $\mathbf{x} \wedge \mathbf{y}$ désigne $((\mathbf{x} \supset (\mathbf{y} \supset \mathbf{f})) \supset \mathbf{f})$
 - ▼ $\mathbf{x} \equiv \mathbf{y}$ désigne $((\mathbf{x} \supset \mathbf{y}) \supset ((\mathbf{y} \supset \mathbf{x}) \supset \mathbf{f})) \supset \mathbf{f}$

« « » »

6

❖ Inférences:

- ▼ I_1 : Si formule x contenant a_i et application $a_i \rightarrow z_i$ (z_i formule) alors formule x avec z_i à la place de a_i
ex: $x = (a \supset a)$ alors $((a \supset b) \supset (a \supset b))$ inférée
- ▼ I_2 : si x, y et z sont des formules et y à la forme $(x \supset z)$ alors z est inférée

$$\frac{p, p \rightarrow q}{q}$$

❖ Exemple de théorèmes prouvables

- ▼ $(p \supset p), (q \supset q) \dots$

❖ Regardons $(p \vee p)$

- ▼ soit le nouvel axiome $(p \vee p)$
- ▼ $(p \vee p)$ **la notation $x \vee y$ désigne $((x \supset y) \supset y)$**
- ▼ donc on a aussi **$((p \supset p) \supset p)$**
- ▼ $(p \supset p)$ théorème + théorème **$((p \supset p) \supset p)$ + Règle d'Inférence $I_2 \rightarrow p$ est inféré**
- ▼ si $(p \vee p)$ est vérifié alors p l'est aussi

« « « « » » » »

7

❖ Règles de valuation

- ▼ $V = \{\text{vrai, faux}\}$ ensemble des valeurs de vérité
- ▼ modèle : fonction v
 - $v: s \in (a, b, c, \dots) \rightarrow v(s) \in \{\text{vrai, faux}\}$
 - $v: f \rightarrow v(f) = \text{faux}$
 - $v: (p \supset q) \rightarrow \text{vrai}$ sauf si $v(p) = \text{vrai}$ et $v(q) = \text{faux}$

Entrée		Sortie				
p	q	$p \cup q$	$p \cap q$	$\emptyset p$	$p \rightarrow q$	$p \leftrightarrow q$
T	T	T	T	F	T	T
T	F	F	T	F	F	F
F	T	F	T	T	T	F
F	F	F	F	T	T	T

▼ Interprétation

- $x \supset y$: **l'implication** " si x alors y "
- $\neg x$: **la négation** " pas x "
- $x \vee y$: **disjonction** " au moins l'un des 2 "
- $x \wedge y$: **conjonction** " les 2 "
- $x \equiv y$: **équivalence** " si et seulement si "

« « « « » » » »

8

- ❖ Logique des propositions
 - ▼ logique travaillant sur un énoncé complet
 - ▼ Interprétation ex. $I(a) = \text{"Il y a des nuages"}$
 - ▼ Système déductif:
 - $I(a) = \text{"Il pleut"} ; I(b) = \text{"Il fait beau"} ; I(c) = \text{"je cours"}$
 - $a \equiv \neg b ; \neg b \supset \neg c$
 - rapidement avec table de vérité : $a \supset \neg c$
- ❖ Qualités de la logique des propositions
 - ▼ Def. **tautologie** : formule toujours vraie ex: $(a \supset a)$
 - ▼ **logique correcte** : toutes les formules qui sont des théorèmes sont des tautologies
 - ▼ **logique complète** : toutes les formules qui sont des tautologies sont des théorèmes
 - ▼ **logique décidable** : il existe un procédé de calcul qui pour toute formule indique en un temps fini si elle est un théorème de cette logique

« « » »

9

- ❖ Un philosophe stoïcien, Chrysippe, logique stoïcienne une logique des propositions : analyse les raisonnements sans entrer dans la structure interne de leurs propositions
- ❖ modus ponens
 - ▼ axiome (*i.e.* indémontrable) stoïcien : Si le premier, alors le second. Or le premier. Donc le second
$$\frac{p, p \rightarrow q}{q}$$
- ❖ Problème de la logique des propositions pour les ontologies :
 - ▼ pas de possibilité de faire la différence entre individus et classes d'individus.
 - ▼ pas de possibilité de parler des relations entre individus.

« « » »

10

❖ la logique commence avec Aristote, (IV^e siècle A.C.) : écrits édités sous le nom d'Organon

- ▼ Organon = Instrument
i.e. logique = instrument du savoir
- ▼ assurer des raisonnements correctes sur les propositions et en particulier les catégories
- ▼ structure d'une proposition catégorique:
(Quantificateur) Sujet-(Copule)-Prédictat
ex: Tous les hommes sont mortels
- ▼ annoncé déclaratif vrai ou faux
(≠ interrogatif ou impératif)
- ▼ quantité : universelle ou particulière
- ▼ copule/qualité : affirmative ou négative



« « » »

11

❖ Terme (concept)

- ▼ extension (l'ensemble des objets qu'il désigne)
- ▼ compréhension (les caractères qu'on énonce quand on définit le terme)

❖ Exemple de "homme"

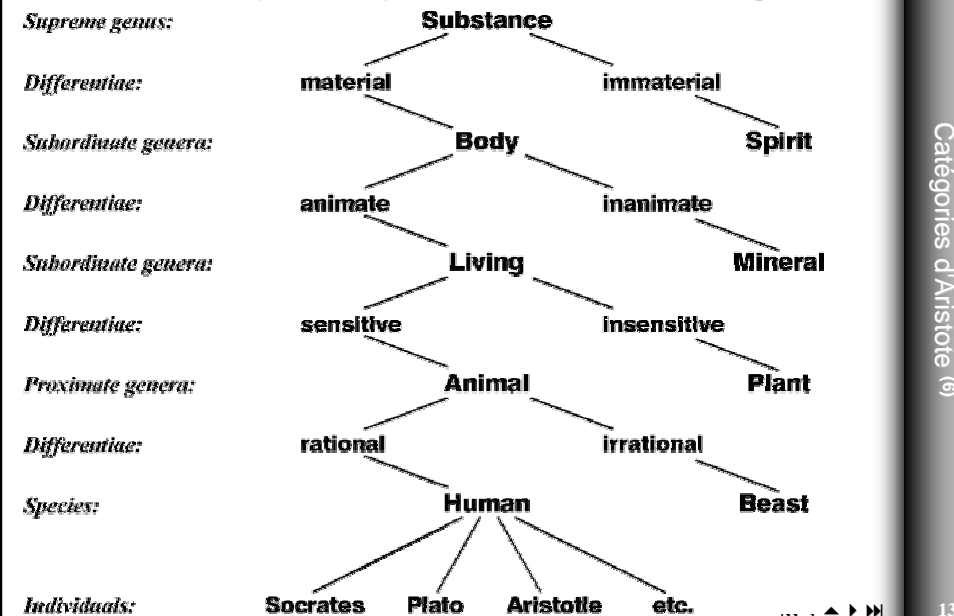
- ▼ extension : Pierre, Paul, Jacques...
- ▼ compréhension : être vivant, bipède, omnivore...
- ▼ les propositions universelles considèrent toute l'extension du sujet, les particulières seulement une partie

❖ La logique d'Aristote est une logique des termes, des classes ou des prédicats

« « » »

12

❖ Catégories organisée selon leurs *genus* et *differentiae* pour représenter une ontologie



❖ "raisonner c'est inférer" Aristote

- ▼ Inférer consiste à tirer d'une ou de plusieurs propositions données et connues comme vraies ou comme fausses (**prémisses**) une ou plusieurs propositions nouvelles jugées vraies ou fausses (**conclusions**) en fonction de la relation logique que l'on a établie entre elles et les prémisses
- ▼ Inférences immédiates : une seule prémisse
- ▼ les inférences médiate : plusieurs prémisses en particulier le syllogisme: deux prémisses

Tous les hommes sont mortels	(Majeure)	prémisses
Tous les Grecs sont des hommes	(Mineure)	
Tous les Grecs sont mortels		conclusion

- ❖ Organon : Instrument... développer une logique formelle pour raisonner sur les propositions catégoriques
- ❖ 4 types de proposition catégorique:
 - ▼ **A**: L'universelle affirmative **Tout A est B**
ex. Tout homme est mortel
 - ▼ **E** : L'universelle négative **Aucun A n'est B**
ex. Aucun homme n'est mortel
 - ▼ **I** : La particulière affirmative **Quelque A est B**
ex. Quelque homme est mortel
 - ▼ **O** : La particulière négative **Quelque A n'est pas B**
ex. Quelque homme n'est pas mortel
 - ▼ **A, E, I, O** : **A**ffirmo et **nEgO** (j'affirme, je nie)
4 praedicatum ... predicats

❖ Exemples de syllogismes

Barbara A: Every animal is material. A: Every human is an animal. A: Every human is material.	Celarent E: No spirit is a body. A: Every human is a body. E: No spirit is a human.
Darii A: Every beast is irrational. I: Some animal is a beast. I: Some animal is irrational.	Ferio E: No plant is rational. I: Some body is a plant. O: Some body is not rational.

base des règles d'héritage

vérification de contraintes

base des logiques de description

- ❖ Il en existe 64, les 15 plus intéressants sont obtenus par règles de conversion

- ❖ Dans les logiques contemporaines c'est la logique des prédicats / logique du 1^{er} ordre:
 - ▼ symboles logiques \supset, \forall quantificateur universel "pour tout..."
 - ▼ variables x, y, \dots
 - ▼ fonctions g, h, \dots
 - ▼ prédicats f, p, q, \dots } arité : nombre de paramètres
 - ▼ $(,)$
 - ▼ logique plus fine, qui inclut la logique propositions
- ❖ Réécritures possibles :
 - ▼ $pred_i, fonct_i$ symboles prédicats / formules d'arité i
 - ▼ $formule \rightarrow pred_0 \mid pred_1(terme) \mid pred_2(terme, terme) \mid \dots \mid (formule \supset formule) \mid (\forall var) (formule)$
 - ▼ $terme \rightarrow fonct_0 \mid fonct_1(terme) \mid fonct_2(terme, terme) \mid \dots \mid var$
 - ▼ $var \rightarrow x \mid y \mid \dots$

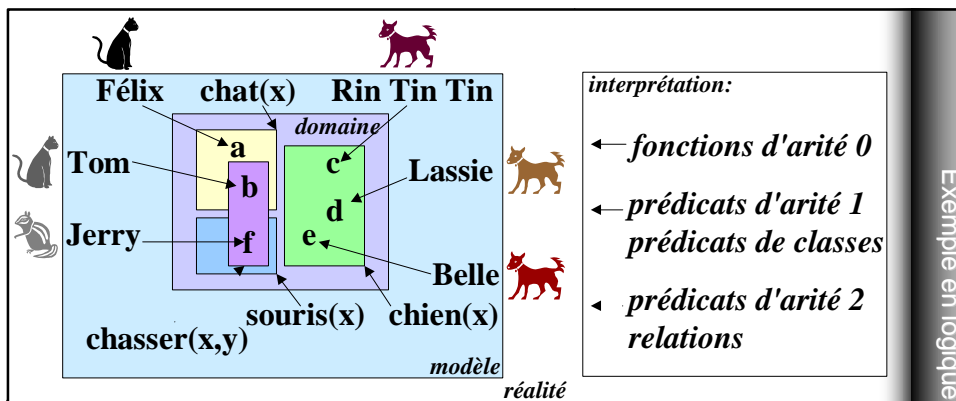
- ❖ Exemple :
 - ▼ $(\forall x) (p(x) \supset ((\forall y) q(x,y)))$
- ❖ Ajouts (notations)
 - ▼ $\neg x$ négation
 - ▼ $x \vee y$ disjonction
 - ▼ $x \wedge y$ conjonction
 - ▼ $x \equiv y$ équivalence
 - ▼ \exists quantificateur existentiel "il existe au moins un..."
 $(\exists x)(f)$ désigne la formule $\neg(\forall x)(\neg f)$

❖ $V = \{\text{vrai}, \text{faux}\}$ et Modèle:

- ▼ Domaine D : ensemble non vide absolument quelconque
- ▼ Interprétation I :
 - $I: \text{fonct}_n \rightarrow [f_n : D_n \rightarrow D]$ ($\text{fonct}_0 \rightarrow \text{constante}$)
 - $I: \text{préd}_n \rightarrow [p_n : D_n \rightarrow V]$
 - $\text{Préd}_0 \rightarrow \text{propositions}$
 - $\text{Préd}_1 \rightarrow \text{sous ensembles de } D \dots \text{prédicats de classes}$
 - $I: f \rightarrow \text{faux}$
- ▼ Assignment $A : \{x, y, \dots\} \rightarrow D$

❖ Evaluation

- ▼ $\text{pred}_n(\text{terme1}, \text{terme2}, \dots) \rightarrow p_n(d1, d2, \dots)$
- ▼ $(p \supset q) \rightarrow \text{vrai}$ sauf si $v(p)=\text{vrai}$ et $v(q)=\text{faux}$
- ▼ $(\exists x)(a) \rightarrow \text{faux}$ s'il existe au moins un x tel que a est faux ; vrai sinon

❖ Le truc génial $(\forall x) (\text{chat}(x) \supset \text{félin}(x))$:

classe 2
individu
classe 1

Connaissance	Logiques propositions	Logiques prédicats
Tom est un chat	$p_1 = \text{"Tom est un chat"}$	$\text{chat}(\text{Tom})$
Félix est un chat	$p_2 = \text{"Félix est un chat"}$	$\text{chat}(\text{Félix})$
les chats sont des félin	<i>impossible</i>	$(\forall x) (\text{chat}(x) \supset \text{félin}(x))$

❖ Exemples de modélisation:

- ▼ les livres sont des objets:
- ▼ $(\forall x) (\text{livre}(x) \supset \text{objet}(x))$
- ▼ les livres sont écrits par des auteurs, les auteurs sont des personnes:
- ▼ $(\forall x) (\text{livre}(x) \supset ((\exists y) (\text{auteur}(y) \wedge \text{a_écrit}(y,x))))$
 $(\forall x) (\text{auteur}(x) \supset \text{personne}(x))$
- ▼ celui qui écrit un livre est un auteur:
- ▼ $(\forall x) (\forall y) (\text{a_écrit}(y,x) \wedge \text{livre}(x) \supset \text{auteur}(y))$
- ▼ un livre est édité par une entreprise d'édition:
- ▼ $(\forall x) (\text{livre}(x) \supset ((\exists y) (\text{éditeur}(y) \wedge \text{a_édité}(x,y))))$
 $(\forall x) (\text{éditeur}(x) \supset \text{entreprise}(x))$
- ▼ $(\forall x) (\text{livre}(x) \supset ((\exists y) (\text{entreprise}(y) \wedge ((\exists z)$
 $(\text{a_pour_activité}(y,z) \wedge \text{édition}(z)))))$

❖ \neq ntes modélisations et leur portée

❖ Exercice:

- ▼ une entreprise est une organisation
- ▼ $(\forall x) (\text{entreprise}(x) \supset \text{organisation}(x))$
- ▼ une organisation contient des membres qui sont des personnes
- ▼ $(\forall x) (\text{organisation}(x) \supset ((\exists y) \text{membre}(y)$
 $\wedge \text{contient}(x,y))$
 $(\forall x) (\text{membre}(x) \supset \text{personne}(x))$
- ▼ un membre qui appartient à une organisation est contenu dans cette organisation (relation inverse)
- ▼ $(\forall x)(\forall y) ((\text{organisation}(x) \wedge \text{membre}(y) \supset$
 $(\text{appartient}(y,x) \equiv \text{contient}(x,y)))$
- ▼ la relation contient est transitive
- ▼ $(\forall x)(\forall y)(\forall z) (\text{contient}(x,y) \wedge \text{contient}(y,z) \supset$
 $\text{contient}(x,z))$

❖ Prix à payer:

- ▼ **semi-décidable**: Il n'existe pas d'algorithme capable de déterminer pour toute formule, en un temps fini, qu'elle n'est pas démontrable
- ▼ **décidable** si uniquement des prédicats d'arité < 2
- ▼ **faire des choix** pour simplifier les langages tout en gardant l'expressivité dont on a besoin

❖ Logique du premier ordre (first-order logic FOL) développée par Frege et Peirce

❖ Limitations:

- ▼ pas de propriétés sur les relations
 ex: $(R) \text{ (transitive}(R) \equiv (\forall x)(\forall y)(\forall z) (R(x,y) \wedge R(y,z) \supset R(x,z))$ **impossible**

❖ Extensions logiques non classiques

- ▼ logiques d'ordre supérieur & logiques modales (nouveaux opérateurs et expressivité supérieure)
- ▼ logiques multi-valuées & logiques des possibilités / logique floues & proba
- ▼ Ordres supérieur : augmenter l'abstraction, améliorer la lisibilité, simplifier les représentations et améliorer l'efficacité des méthodes de calcul
- ▼ axiomes: interprétés en permanence
- ▼ règles d'inférence : compilables

❖ Les réseaux sémantiques

- ▼ le terme vient de Quillian 1968 dans sa thèse
- ▼ réseau d'association de concept : // proximité sémantique et proximité spatiale
- ▼ "capturer l'aspect objectif du sens, les propriétés des choses"
- ▼ "les concepts sont définis par leur position dans le réseau sémantique"
- ▼ nous savons où se trouve Paris car nous le positionnons par rapport aux lieux qui nous sont connus:
 - **au nord** de St Etienne à environ 500 km
 - **sur** la Seine au Sud-Est de Rouen...
- ▼ nous savons ce qu'est un concept car nous le positionnons par rapport aux autres ex: Lion
 - **fait partie** des félins et **de couleur** jaune
 - **beaucoup plus gros** qu'un chat

❖ Mind Map: prise de note structurée rapide



❖ **Faites un réseau d'association (simple)**

agrafeuse
vis
marteau
tournevis
clou
scie
planche
perceuse
<au choix>
<au choix>
[] — []

❖ **Sémantique des liens ? (étiquettes !)**

❖ **Réseaux sémantiques (Quillian en 1968)**

- ▼ graphe orienté, nœuds et arcs étiquetés

❖ **Propagation de marqueurs (Fhalman en 1979)**

- ▼ parcours dans le sens des arcs
- ▼ accessibilité, transitivité et clôture

❖ **Limité**

- ▼ pas d'étiquette *a priori*, pas de différence entre les individus et les intensions
- ▼ i.e. pas de typage ni subsomption
- ▼ // logique des prédicats
- ▼ pas de signification claire des arcs
donc pas de sémantique claire

❖ Le dernier réseau est un tricheur... pourquoi ?

Exemples de problèmes

❖ **Danger** : sémantique des liens inconnue, conventions de positionnement implicite

❖ **Formule logique:**

$$(\text{possède}(a,b) \wedge \text{personne}(a) \wedge \text{chien}(b) \wedge \text{nom}(b, \text{"Rex"}) \wedge \text{prénom}(a, \text{"Jim"}) \wedge \text{nom}(a, \text{"Mini"}))$$

❖ Le personne qui s'appelle Jim Mini possède un chien qui s'appelle Rex.

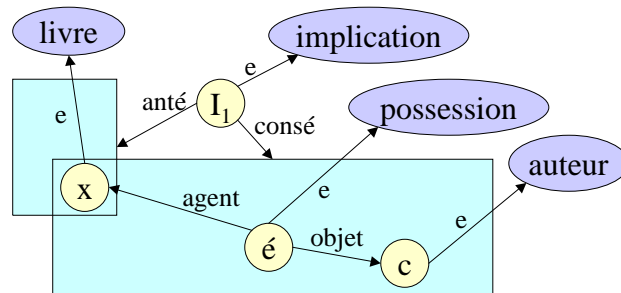
❖ **Graphe :**

❖ La personne qui s'appelle Tom Cat possède une voiture appelée Clio

❖ Meilleure lisibilité: visualisation immédiate des coréférences ... mais...

L'argument de lisibilité

- ❖ Contre exemple:
 $(\forall x) (\text{livre}(x) \supset ((\exists y) (\text{auteur}(y) \wedge \text{a_écrit}(y,x))))$
- ❖ Réseau partitionné (Hendrix en 1978) :

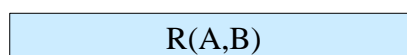
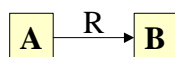


- ❖ Dépend:
 - ▼ type de réseau & type de connaissances
 - ▼ taille réseau considéré
- ❖ Aller au delà de la lisibilité : exploiter les particularité de cette représentation

- ❖ D'un point de vue implantation : les arcs et relations sont mémorisés comme des citoyens de première classe
 - ▼ suivre les arcs du regard ; la lecture du réseau est une propriété que n'ont pas les formules
 - ▼ suivre les arcs par algorithme
 - ▼ s'appuyer sur une structure efficace (représentation)
 - ▼ algo de propagation
- ❖ Notations graphiques permet d'appréhender de façon plus intuitive
 - ▼ solutions partielles dédiées
- ❖ Langages formels obligatoires pour manipulation et échanges

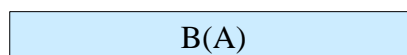
- ❖ Shapiro : différents types de relations
- ❖ Hendrix 78: quantification
- ❖ Wood 75: nécessité signification des nœuds et des arcs
- ❖ Rationalisation schémas et réseaux sémantiques [Brachman, 77, 79, 83, 85]
 - ▾ dichotomie individus / ensemble d'individus
 - ▾ propriétés descriptives / propriétés typiques
 - ▾ définition (ssi) / description (nécessaire) / prototype

- ❖ un arc "signifie" que deux nœuds sont "en relation" // prédicat binaire

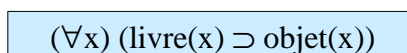


- ❖ certains nœuds sont des classes d'autres des individus

- ▾ lien d'appartenance d'un individu à une classe: "A Kind Of" ou AKO



- ▾ lien de subsomption / spécialisation entre classes: "is a" ou IS-A:

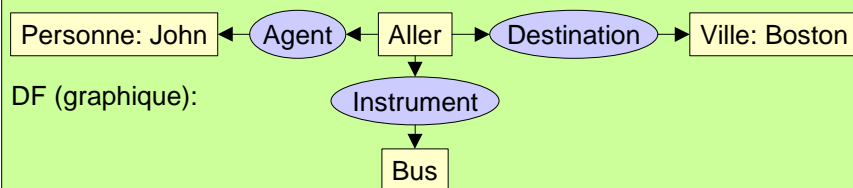


- ▾ nœuds de type 'classe' et de type 'individu'

- ❖ **Graphes Conceptuels de Sowa** ⁽¹⁹⁾ ⁽²⁰⁾
 - ▼ Notation graphique pour la logique
 - ▼ basée sur les graphes existentiels de Peirce
 - ▼ augmentée par des résultats des réseaux sémantiques
 - ▼ motivations :
 - aide à l'analyse du langage naturel
 - présentation agréable de la logique à l'humain
- ❖ **Sémantique formelle des GC** (Chein et Mugnier, 1992)
- ❖ **Différents niveaux dans le formalisme:**
 - ▼ Graphes simples
 - ▼ Graphes imbriqués
 - ▼ Graphe de règles

- ❖ **Plusieurs représentations:**
 - ▼ pour un affichage DF (Display Form) graphique
 - ▼ pour une lecture linéaire: LF (Linear Form)
 - ▼ pour la transmission entre machines: CGIF (Conceptual Graph Interchange Format)

Langue naturelle: "John va à Boston par le bus"



LF (pas graphique mais lisible): [Aller] -
 (Agent) ->[Personne: John]
 (Destination) ->[Ville: Boston]
 (Instrument) ->[Bus].

CGIF: [Aller *x] (Agent ?x [Person: John]) (Destination ?x [Ville: Boston])
 (Instrument ?x [Bus])

❖ Un GC est un graphe bipartite orienté

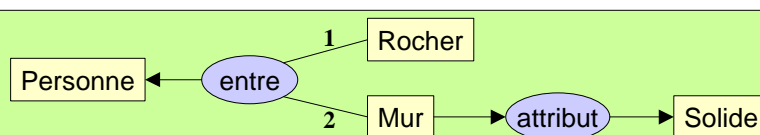
- ▼ bipartite : deux types de nœuds
 - nœuds de type concept
 - nœuds de type de relation
- ▼ Les nœuds sont liés par des arcs orientés
- ▼ Un arc lie toujours un nœud concept et un nœud relation
- ▼ un nœud concept peut être isolé (tout seul...snif)

❖ Les concepts :

- ▼ un type, ex. [**Personne**: John]
- ▼ un référent, ex. [Personne: **John**]
- ▼ cas du [Bus] : $\exists x$ [Bus: x] tel que... (quantificateur existentiel)

❖ Les relations :

- ▼ valence: entier n donnant le nombre d'arc monadique, dyadique, triadique... n-adique
- ▼ signature: ensemble de n types de concepts
- ▼ la valence et la signature sont fixés par le type de la relation



Valence de 3

Signature = (Entité,Entité,Entité)

[Personne] <-(Entre)-

<-1- [Rocher]

<-2- [Mur] -> (Attribut) -> [Solide]

❖ Nature des Relations

- ▼ fonction avec un argument ou plus et dont le domaine de valeur est {vrai, faux}
- ▼ soit synonyme de prédicat soit
 - relation : prédicat binaire ou plus
 - propriété : prédicat unaire
- ▼ Citoyennes de première classe au même titre que les concepts
- ▼ Algorithmes et inférences selon leur type

Type	Axiome	Exemple
Réflexive	$(\forall x) x @ x$	x est aussi vieux que y
Irréflexive	$(\forall x) \text{non } (x @ x)$	x est le frère de y
Symétrique	$(\forall x, y) x @ y \text{ implique } y @ x$	x est marié avec y
Asymétrique	$(\forall x, y) x @ y \text{ implique non } (y @ x)$	x est le père de y
Antisymétrique	$(\forall x, y) x @ y \text{ et } y @ x \text{ implique } x = y$	x était là à la naissance de y
Transitive	$(\forall x, y) x @ y \text{ and } y @ z \text{ implique } x @ z$	x est un ancêtre de y

subsumption

39

❖ λ - Expression

- ▼ lambda calculus, (Alonzo Church 1941)
 $f(x) = 2x^2 + 3x - 2$ $f = \lambda x (2x^2 + 3x - 2)$
- ▼ intension Vs extension : égalité et définition
- ▼ faire des définitions formelles (interchangeable)
- ▼ une lambda expression n-adique est définie par un graphe (appelé le corps de l'expression) contenant n paramètres formels $\lambda_1, \lambda_2, \dots, \lambda_n$
- ▼ signature de l'expression = ensemble (t_1, t_2, \dots, t_n) où t_i est le type de concept du paramètre λ_i

"quelqu'un habite à un endroit"

Personne: λ_1 ← Agent ← Habiter → Emplacement → Lieu: λ_2

$(\lambda (\text{Personne} * x, \text{Lieu} * y) [\text{Habiter} * z] (\text{Agent} ?z ?x) (\text{Emplacement} ?z ?y))$

40

❖ Types de concept

- ▼ Les types de concepts sont **primitifs** ou **définis**
 - primitif: juste un **étiquette** et un positionnement dans la hiérarchie
 - défini: la définition est donnée par une **λ -expression**
- ▼ Deux types prédéfinis:
 - Le type universel noté **T** : type de 'toute chose'
 - Le type absurde noté **\perp** : type impossible
- ▼ hiérarchie: ordre partiel donné par la relation de subsomption (spécialisation des types)
 - \geq super-type de $>$ strictement super-type de
 - \leq sous-type de $<$ strictement sous-type de

Personne $<$ EntitéVivante
 Directeur = [Personne: λ] \rightarrow (Diriger) \rightarrow [Groupe]

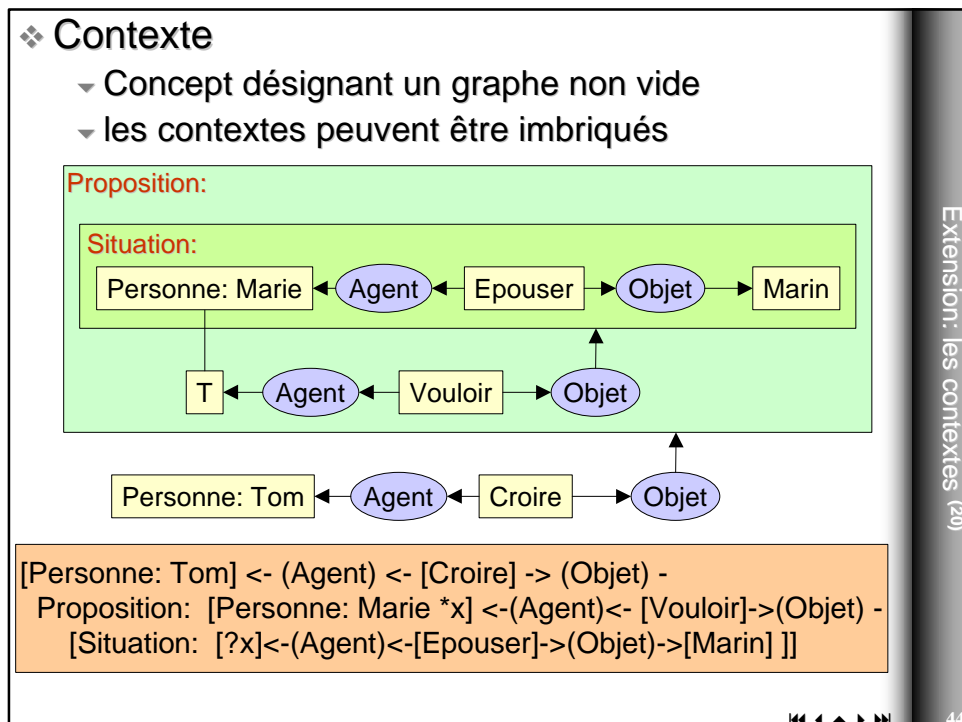
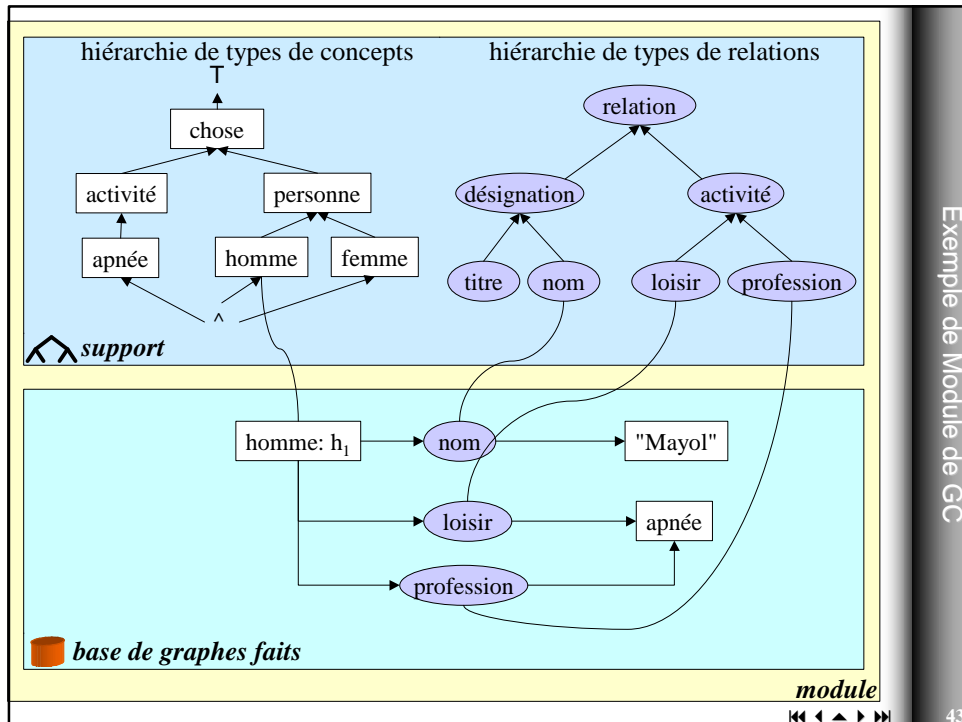
❖ Types de relation

- ▼ hiérarchie: ordre partiel donné par la relation de subsomption (spécialisation des types)
- ▼ Les types de relations sont **primitifs** ou **définis**
- ▼ valence de la relation (entier positif n)
- ▼ signature (types compatibles)

nom $<$ désignation

X vit à Y = "personne x habite à l'endroit Y"

[Relation: VivreA] \rightarrow (Def) \rightarrow [LambdaExpression: [Personne: λ_1] $<-$ (Agent) $<-$ [Habiter] \rightarrow (Emplacement) \rightarrow [Lieu: λ_2]]



❖ Exemple Inférence: projection

❖ Equivalence de 2 graphes

- ▾ algo exponentiel dans le cas général
- ▾ restrictions graphes acycliques : polynomial

Projection et algorithmique (20)

45

❖ Règles dans les graphes [Salvat Mugnier 1996]

IF

THEN

❖ Possibilité d'utiliser cela à la place des définitions (nécessairement et pas d'existentiel)

IF

THEN

Extension: les règles (20)

46

❖ Limites :

- ▼ problème de disjonction
- ▼ problème de négation
- ▼ problème d'imbrication de quantificateurs

❖ Plates-formes

- ▼ CoGiTo, CoGITaNT
- ▼ Notio (API Java)
- ▼ CharGer (editeur CG)
- ▼ WebKB - (CG et recherche info)
- ▼ CG Mars Lander (Question Réponse)
- ▼ Prolog+CG - (object-oriented extension of PROLOG, based on CG implémenté en JAVA)
- ▼ Project Peirce - A collaborative project for developing a CG workbench (see also ftp site).
- ▼ Synergy

❖ Autres réseaux

- ▼ réseaux connexionistes & réseaux de neurones
- ▼ pétri
- ▼ ...

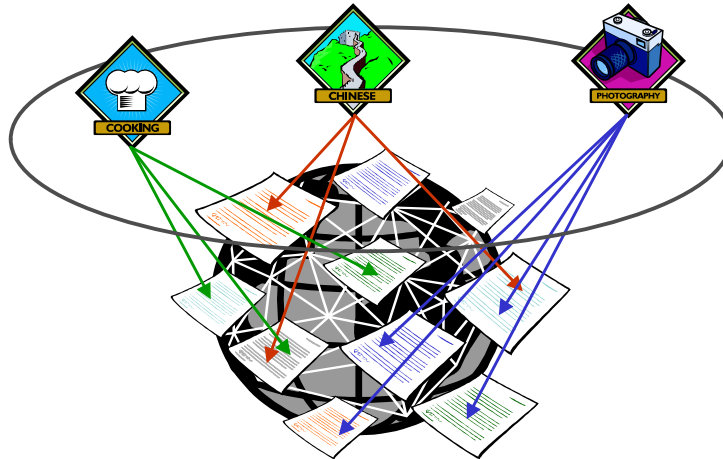
❖ Autre Communauté les documentalistes:

❖ Topic Maps

- ▼ index pour les documents électroniques
- ▼ glossaires, thesaurus, références, catalogues
- ▼ fusion d'index
- ▼ liens n-aires
- ▼ de multiples topic maps et les relier

❖ Liens avec SGML et maintenant ... XML

- ❖ Un 'topic' réifie un sujet sous forme d'un lien multiple pointant sur les occurrences de ce sujet dans la masse documentaire



- ❖ Les topics sont définis extérieurement aux documents et définissent une "carte des sujets"

- ❖ Groupes d'Occurrences de Topics : les rôles
 - ▼ mention: le sujet est mentionné
 - ▼ définition: le sujet est défini
 - ▼ graphique Vs. texte, principale Vs. ordinaire
- ❖ Classes de topics: occurrence = instance d'une classe
- ❖ Nom du Topic
 - ▼ nom de base / désignation
 - ▼ nom d'affichage: peut être le même pour plusieurs topics ; peut être graphique.
 - ▼ nom de tri: pour pb. chiffres romains, le 'll' qui vient après 'lo' en Espagnol, graphiques...

❖ **Noms multiples:**

- ▼ aucun nom: lien entre des occurrences pour lesquelles on a détecté une similarité du sujet sans expliciter exactement ce sujet
- ▼ un nom : cas commun
- ▼ plusieurs nom :
 - "voiture", "automobile"... entrée multiples dans un index
 - accès multilingue: "museum", "musée", "museo", ...

❖ **Association de topics:**

- ▼ exemple : inclusion ex. arts \supset musique
champs lexicaux, "voir aussi...", etc.
- ▼ relations libres entre topics
- ▼ indépendant des documents et occurrences
- ▼ **très proche de la notion d'ontologie !**

51

❖ **Caractéristiques d'un topic**

- ▼ caractéristique = noms + occurrences + rôles
- ▼ une caractéristique est associée à un topic
- ▼ une caractéristique a une portée (scope) définissant son domaine de validité pour un topic.
- ▼ permet différents points de vue
- ▼ Un élément de ce domaine de validité est appelé un thème
- ▼ exemple : définition dans le thème débutant
mention dans le thème expert

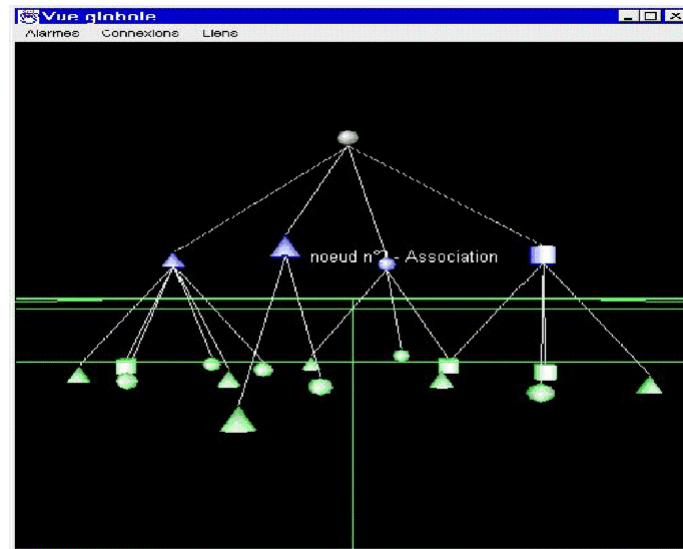
❖ **Facette:**

- ▼ filtres sur Topic Maps
- ▼ exemple : Topic Map multilingue | Facette Français

❖ **Documenter les Topic Maps pour les fusionner**

52

❖ Tradition de la visualisation

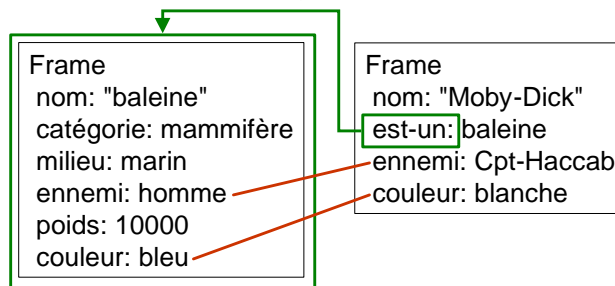


❖ Schémas ou Frames Minsky 1975

- ▼ représentation de connaissances acquises par la vision
- ▼ schéma : ensemble d'attributs
- ▼ attributs: valeurs de facettes
- ▼ ex: rôle, dénominations, position, composition...
- ▼ facette // point vue en vision
- ▼ mise en correspondance : comparaison de frames (un plus complet que l'autre ou 2 complétables de façon compatible)
- ▼ inspiré de théories psychologiques
 - nous consultons des structures représentant des situations auxquelles nous avons déjà été confrontés
 - nous choisissons la plus proche
 - nous l'adaptions pour qu'elle corresponde

❖ **Frame** : structure de donnée qui représente un objet typique ou une situation stéréotypée

- ▼ Typicalité : valeur probable, valeur par défauts
- ▼ Description différentielle:
 - nouveau frame en exprimant ses différences par rapport à un frame existant
 - hiérarchie mélangeant tous les types d'entités, catégories générales (de l'animal à "paf le chien")
 - langages d'acteur (attributs et comportements), création par clonage et extension



❖ **Absence de classes:**

- ▼ pas de modélisation préalable tout au même niveau
- ▼ problème pour le partage de caractéristiques au sein d'une famille (réintroduction de prototypes centraux)

❖ **Différentiation par rapport à un objet :**

- ▼ façon originale de représenter la connaissance: étape préalable de modélisation ?
- ▼ gestion d'un réseau de dépendance entre objets

❖ **NB(1):** la liberté d'expressivité n'est pas toujours un avantage, les contraintes aident parfois à faire des choix.

❖ **NB(2):** peut être utilisé pour des représentations intermédiaires

❖ Brachman 77...85

- ▼ rationaliser les frames
- ▼ différence individu / ensemble individus
- ▼ propriétés descriptives / propriétés typiques
- ▼ définition (ssi) / description (nécessaires) / prototype (valeurs typiques)

❖ Post-frame

- ▼ classes
- ▼ lien spécialisation
- ▼ rationaliser les facettes: restriction du nombre de facettes et parfois extensibles
- ▼ envoie de messages

❖ KNOWLEDGE CRAFT, ART, KE, Nexpert Objet, Shirka, Objlog, Kool, Smeci, Yafoo,

57

❖ Modélisation par Prototype

- ▼ se référer à un objet typique
- ▼ membre moyen, représentant idéal, l'exemple : le prototype
- ▼ vient des langages de frames et acteurs
- ▼ connaissance par défaut : la plus probable
- ▼ les autres objets en sont des spécialisations
- ▼ pas de distinction classe / instance
- ▼ tout objet peut être spécialisé, instancié
- ▼ objet créé :ex nihilo, clonage, extension
- ▼ éviter modélisation préalable, lourde et abstraite

❖ Exemples de langages: Self, Kevo, Exemplars Agora, Garnet, Moonstrap, Cecil, Omega, Newton-Script

❖ Pb: Modèle pas clair, sémantique non unifiée

58

- ❖ **Modélisation Objet: objets informatiques représentant objets réels**
 - ▼ stocker la connaissances autour de la notion d'objet
 - ▼ services inférentiels pour compléter l'info disponible
- ❖ **Notions de base:**
 - ▼ classes: objets instanciables
 - ▼ lien de spécialisation: sorte-de, a-kind-of, AKO
 - ▼ classes structurées en une hiérarchie
 - ▼ sémantique = inclusion ensembliste
 - ▼ instances: objets terminaux
 - ▼ lien d'instanciation: est-un, isa
- ❖ **Triptyque**
 - ▼ objet (object), attribut (slot), facette (facet)
 - ▼ chaque attribut a une liste de facettes
 - ▼ chaque facette a une valeur ou un démon

- ❖ **Spécialisation**
 - ▼ organisation des classes
 - ▼ spécialisation d'un ou plusieurs ancêtres
 - ▼ inclusion ensembliste : les instances d'une sous-classe appartiennent aux instances des classes ancêtres
 - ▼ plus les classes sont spécifiques, plus les contraintes sur les attributs sont précises: les contraintes sur une classes sont dans l'intersection des contraintes des classes ancêtres
- ❖ **Héritage**
 - ▼ hériter les attributs et leurs facettes
 - ▼ enrichir:
 - nouveaux attributs
 - nouvelles facettes sur les attributs hérités
 - spécialiser les facettes héritées en restreignant

❖ Facette

- ▼ précise la nature et le comportement d'un attributs
- ▼ facettes déclaratives / facettes procédurales
- ▼ nature de l'information et utilisation
- ▼ facettes prédéfinies
 - type: entier, réel, chaîne, objet, liste ou ensemble
 - domaine: énumération, intervalle, exception, restriction de types
 - cardinalité: nombres de valeurs admises
 - valeur
 - valeur par défaut
 - démon / réflexe

❖ certains langages permettent d'ajouter des facettes

Facettes (7)

« « » »

61

```
{
Véhicule
  sorte-de
    $valeur Objet ;
moteur
  $un Moteur ;
couleur
  $un symbole
  $domaine (bleu vert rouge noir) ;
année
  $un entier
  $intervalle [1900, 2002]
portes
  $liste-de Porte
}
```

Exemple Véhicule (7)

« « » »

62

Exemple Moteur (?)

```

{
Moteur
  sorte-de
    $valeur Objet ;
  puissance
    $un entier
    $intervalle [20, 500] ;
  carburant
    $un symbole
    $domaine (essence diesel GPL)
    $défaut diesel
    $si-besoin calcul-carburant
}

```

63

Hiérarchie de subsumption (?)

❖ **Structuration hiérarchique**

- ▼ héritage de la structure, des valeurs et des démons selon le type de lien

<pre> graph BT objets[les objets] tangibles[les tangibles] mobiles[les mobiles] artefacts[les artefacts] vehicules[les véhicules] objets --> tangibles objets --> mobiles tangibles --> artefacts mobiles --> vehicules </pre>	<pre> graph BT objets[les objets] tangibles[les tangibles] mobiles[les mobiles] artefacts[les artefacts] vehicules[les véhicules] objets --> tangibles objets --> mobiles tangibles --> artefacts vehicules --> artefacts vehicules --> mobiles </pre>
héritage simple	héritage multiple

- ▼ héritage multiple: définition par fusion (cf. supra)
≠ définition par composition (moteur + roues + ...)

64

```

class Véhicule
  slot moteur : Moteur
class Voiture AKO Véhicule
class VoitureSportive AKO Voiture
  slot moteur : MoteurEssence
class Utilitaire AKO Véhicule
  slot moteur : MoteurDiesel
class VoitureElectrique AKO Voiture
  slot moteur : MoteurElectrique

class Moteur
class MoteurThermique AKO Moteur
class MoteurDiesel AKO MoteurThermique
class MoteurEssence AKO MoteurThermique
class Moteur Electrique AKO Moteur

```

Exemple hiérarchie (?)

65

❖ Mécanismes d'inférence

- ▼ attachement procédural: démon, réflexe
- ▼ contrainte
- ▼ filtrage
- ▼ mutation
- ▼ classification

❖ Démon / réflexe

- ▼ comportement implanté sous forme d'une fonction ou d'une procédure
- ▼ déclenché automatiquement à la suite d'un accès ou d'un événement
 - accès: lecture ou écriture
 - événement: création, destruction, etc.

Mécanismes d'inférence et Démon/Réflexes (?)

66

❖ Types de démons

- ▼ si-besoin: calcule une valeur en cas de lecture s'il n'en possède pas déjà une
- ▼ si-possible: teste si la valeur affectée à un attribut vérifie une contrainte liée à l'objet
- ▼ si-modifie: en cas de modification de la valeur
- ▼ si-ajout, si-enleve: pour les attributs de type liste
- ▼ si-crée, si-détruit: à la création/destruction de l'objet
- ▼ ...

❖ Exemples

- ▼ voiture : si-crée : créer moteurs, portes, etc.
- ▼ *idem* pour si-détruit
- ▼ voiture.puissance.si-besoin = self.moteur.puissance

```

class Moteur
  puissance : fix

class Voiture
  puissance : fix
  si-besoin : getPower
  moteur: object Moteur

function getPower(object slot)
  return objet.moteur.puissance

```

❖ Accès aux attributs

- ▼ ordre : \$valeur → \$si-besoin → \$défaut
- ▼ Si la valeur n'est disponible par aucune de ces facettes:
 - parcours des ancêtres pour trouver par héritage
 - en Z: pour chaque objet on regarde les 3 facettes
 - en N: pour chaque facette on regarde les ancêtres

❖ Démons et raisonnement

- ▼ chaînage arrière: dirigé par les buts, "quelle est la valeur de ...?" si-besoin
- ▼ chaînage avant: dirigé par les données, "on dit que... alors..." si-modifie / si-ajout

⏪ ⏩ ⏴ ⏵

69

❖ Contrainte sur une valeur d'attribut

```
class Employé
  attributs:
    projets: Projet ;
    chef: Employé ;
  contraintes:
    chef=projet.chef ;
```

❖ Filtrage: accès par appariement à un filtre

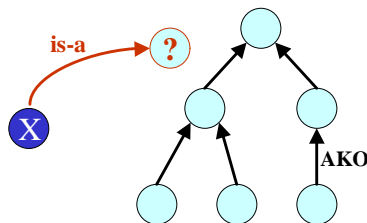
- ▼ Dans Shirka: \$valeur → \$sib-filtre → \$défaut

```
{ Véhicule
  sorte-de = Objet ;
  puissance
    $un entier
    $var-nom P ;
  moteurs-possibles
    $liste-de Moteur
    $sib-filtre
      { Moteur
        lui-même $var -> moteurs-possibles ;
        puissance $var <- P ; }
}
```

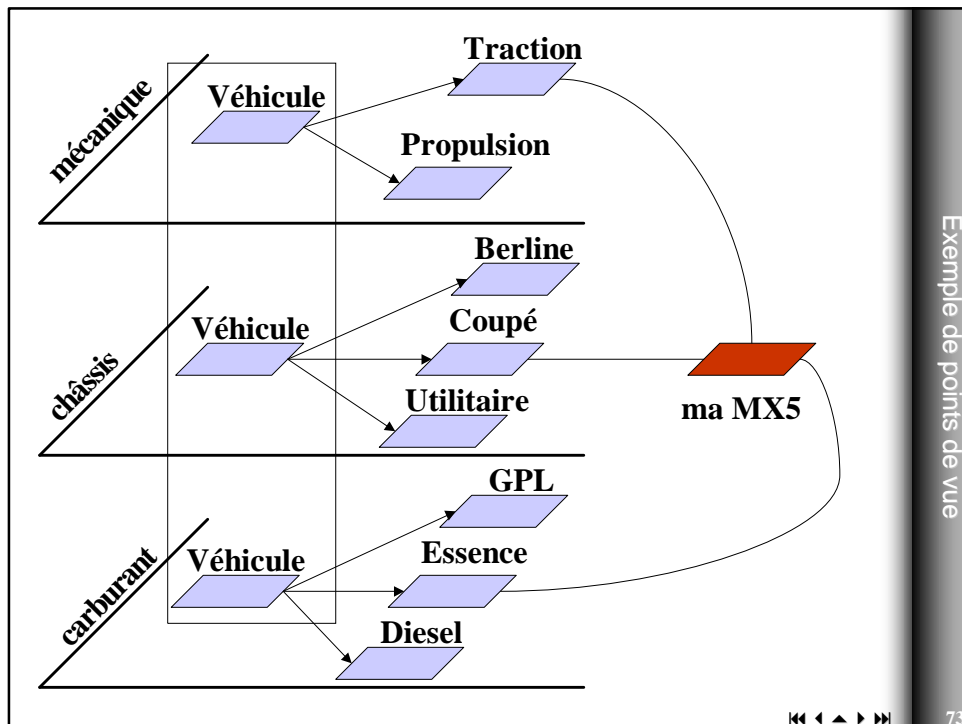
⏪ ⏩ ⏴ ⏵

70

- ❖ Mutation : changer la classe d'un objet
 - ▼ se fait dans le sens de la spécialisation: ajout et précision/restriction
 - ▼ mutation récursives
 - ex: muter un véhicule amène à muter son moteur
 - ▼ échec de la mutation: une contrainte est violée
- ❖ Classification (!!! identification) : trouver la classe la plus spécifique d'un objet



- ❖ Points de vue (ex. TROEPS)
- ❖ Concept:
 - ▼ structure objets: ensemble attributs typés
 - ▼ identité: sous ensemble attributs donnant un clef
 - nom, prénom, naissance, salaire, projet, diplôme
 - nom + prénom + date de naissance
 - ▼ concept vu sous plusieurs points de vue
- ❖ Un point de vue : une taxonomie
 - ▼ classe racine + sous classes
 - ▼ pas de multi-généralisation
 - ▼ objet - une classe / point de vue
 - ▼ passerelles entre classes (inclusion)
 - ex.: directeur et intéressés aux bénéfices



❖ Exemple de comparatif:

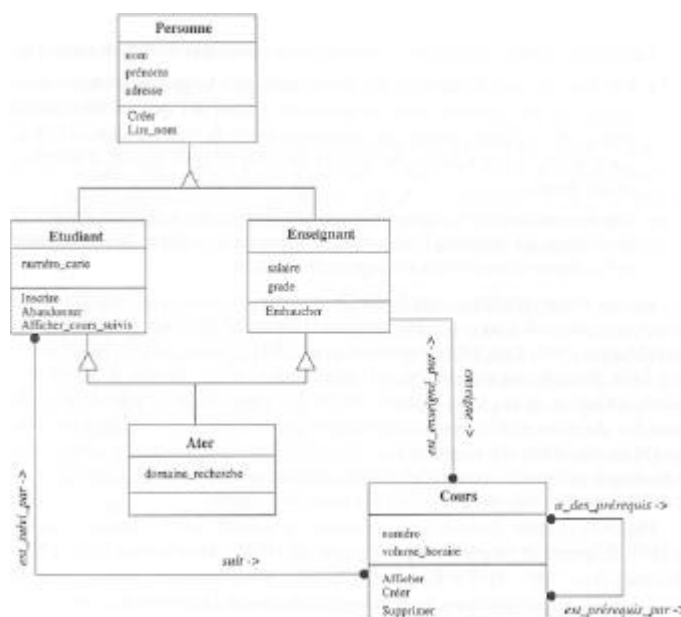
	message	classe/instance	metamodele	réflexes	multi-spec.	multi-inst.	types	attach. proc.	filtres	variables	défaut	héritage lat.	symétrique	contraintes	classification	categorisation	règles	points de vue	commercial
LPO	•	•	•	○	•						○								•
LD		•			•	•			•	•		○	○	+	•	•	+		
BDO	•	•		+	•		•		•								+	○	•
SRL2				•	•				•	•	•	•	•						
KRL		•					•		•	•		○	○	○					
FRL		•		•				•		•	•	•	•	○					
ART		○	•		•	•		•			•		•				•		•
KEE	•	•			•		•	•			•						•		•
SMECI	•	•		•			•	•				•	•				•		•
KOOL	•	•	•	•			•	•			•	•	•				•		•
YAFOOL	•	○	○	•	•		•	•	○	•	•	•	•	+			+		○
SHIRKA		•	•	•	•		•	•	•	•	•	•		○	•	+	+		
OBJLOG	•	•	○	•	•		•	•		•		•		○	•		+	•	•
FROME	•	•		•	•		•	•	•		•			○	•		+	•	•
TROEPS		•				•	•	•	•	•	•	○	○	•	•	+		•	•

❖ Bases de Données Objet

- ▼ Problématique des SGBD: concevoir et implanter un système de gestion de données basé sur un schéma conceptuel unique pouvant supporter plusieurs applications
- ▼ SGBDO: améliorer expressivité schéma de BD en intégrant des modèles objet. Allier:
 - efficacité et passage à l'échelle des SGBD (40 ans)
 - expressivité et abstraction des modèles objets
- ▼ Langages normalisés:
 - langage de description de données, ex. ODL
 - langage de manipulations de données, ex. OQL

❖ Les SGBD peuvent fournir des fondations robustes et efficaces pour les SBC

❖ Modèle OMT



❖ Schéma de base O2

```

Schema Gestion.Universitaire
Class Personne
  tuple (nom: string,
        prenom: list(string),
        adresse: tuple(rue: string, num: integer, ville: string))
  Method Créer (n: string): Personne,
    Lire_nom: string
End Personne
Class Etudiant inherit Personne
  tuple (numero_carte: integer,
        cours_suisvis: set(Cours))
  Method Inscrire (c: Cours),
    Abandonner (c: Cours)
End Etudiant
Class Enseignant inherit Personne
  tuple (grade: string,
        salaire: real,
        cours_enseignés: set(Cours))
  Method Embaucher()
End Enseignant
Class Ater inherit Enseignant, Etudiant
  tuple (domaine_recherche: string)
End Ater
Class Cours
  tuple (numero: string,
        volume_horaire: integer,
        inscrites: set(Etudiant),
        enseignant: Enseignant,
        ...)
  Method Créer(num: string): Cours,
    Supprimer,
    Afficher,
    ...
End Cours

```

❖ Langage d'interrogation

▼ Object Query Language

select objets/attributs
from objets
where conditions

▼ Exemples:

select *
from V Vehicule

select V
from V Vehicule, M MoteurElectrique
where V.moteur = M

❖ Systèmes à objets

- ▼ FRL: MIT 70
- ▼ RLL: Lenat 80
- ▼ SRL: Fox 78-85
- ▼ KRL: Xerox
- ▼ Units, KL-one
- ▼ Shirka, Smeci Yafool & Y3
- ▼ Troeps, Arome, Frome

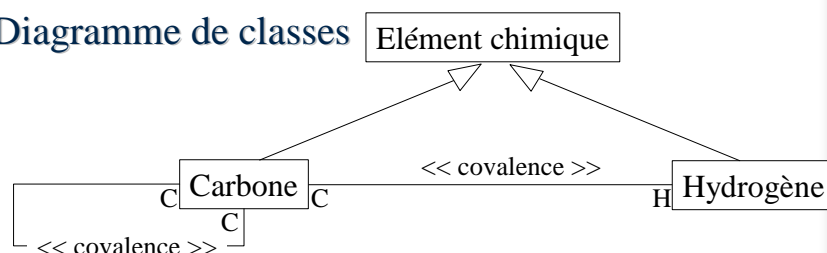
❖ SGBDO :

- ▼ O₂, Ontos, ITASCA, Gemstone, Objectstore, Versant, Matisse, Objectivity/DB, etc.

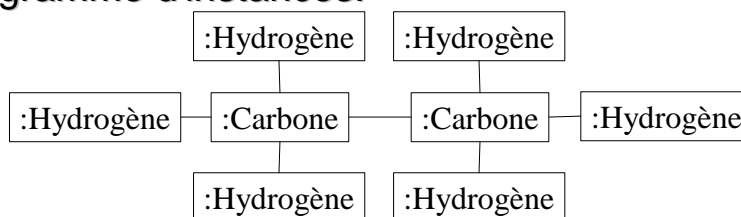
❖ UML

- ▼ *Notation Guide* – syntaxe graphique de UML
- ▼ *UML Semantics* – sémantique associée à la syntaxe

❖ Diagramme de classes



❖ Diagramme d'instances:



une représentation du méthane

- ❖ KL-ONE [Brachman et Shmolze, 85]
- ❖ Logiques de descriptions / logiques terminologiques:
 - ▼ logiques des prédicats
 - ▼ réseaux sémantiques
 - ▼ langages de frames
- ❖ Logiques de description
 - ▼ description: concept / rôle / individu
- ❖ Deux niveaux:
 - ▼ niveau terminologique: représentation et manipulation des concepts et des rôles (TBox)
 - subsumption: hiérarchies de concepts et de rôles
 - ▼ niveau factuel / assertionnel: description et manipulation des individus (ABox)

- ❖ Concept / Individu:
 - ▼ Un concept correspond à une entité générique d'un domaine d'application
 - ▼ Un individu correspond à une entité particulière, instance d'un concept
 - ▼ Un concept représente un ensemble d'individus
- ❖ Un rôle représente une relation binaire entre individus
- ❖ Par rapport aux logiques : pas de variables

$C, D \rightarrow$	$A \mid$	
	$\text{Top} \mid$	$T \mid$
	$\text{Bottom} \mid$	$\wedge \mid$
	$(\text{and } C \ D) \mid$	$C \cap D \mid$
	$(\text{not } A) \mid$	$\neg A \mid$
	$(\text{all } r \ C) \mid$	$\forall r. C \mid$
	$(\text{some } r) \mid$	\exists

syntaxe lipsienne

syntaxe allemande

❖ Description d'un concept ou d'un rôle

- ▼ primitifs ou définis (ayant une définition)
- ▼ définition : description structurée utilisant des constructeurs pour donner:
 - les rôles associés au concept
 - les restrictions des rôles (co-domaine, cardinalité) valeurs de base / concepts

❖ Définitions de descriptions

- ▼ équations terminologiques
- ▼ pas de circuit terminologique

❖ Constructeurs

- ▼ le et and \cap permet de définir une conjonction d'expressions conceptuelles
- ▼ le non not \neg correspond à la négation et ne porte que sur les concepts primitifs
- ▼ la quantification universelle tout all \forall permet de préciser le co-domaine d'un rôle $\forall r.C$
- ▼ la quantification existentielle non typée some certains \exists permet d'affirmer l'existence d'au moins un couple d'individus $(\exists r)$ en relation par r

<p>Personne \leq Top</p> <p>Ensemble \leq Top</p> <p>Homme \leq Personne</p> <p>Femme \leq (<u>and</u> Personne (<u>not</u> Homme))</p> <p>membre \leq toprole</p> <p>chef \leq membre</p>	primitifs
<p>Equipe \equiv (<u>and</u> Ensemble (all membre Personne) (atleast 2 membre))</p> <p>Petite-équipe \equiv (<u>and</u> Equipe (atmost 5 membre))</p> <p>Equipe-moderne \equiv (<u>and</u> Equipe (atmost 4 membre) (atleast 1 chef) (all chef femme))</p>	définis

Exemple (22)

85

❖ Les différents types de langages de DL

- ▼ \mathcal{AL} langage minimum
 - $\mathcal{AL} = \{T, \wedge, \neg A, C \cap D, \forall r.C, \exists r\}$ A concept primitif, C et D concepts définis, r rôle
 - expressions conceptuelles
 - concept Top (T) inclut tous les individus
 - concept Bottom (\wedge) est vide
- ▼ \mathcal{FL} et \mathcal{FL}^- Brachman
 - $\mathcal{FL} = \{C \cap D, \forall r.C, \exists r, r|C\}$ où $r|C$ se note aussi (restrict r C) et introduit une contrainte sur le co-domaine du rôle r.
 - $\mathcal{FL}^- = \{C \cap D, \forall r.C, \exists r\}$
- ▼ $\mathcal{PL1}$ et $\mathcal{PL2}$

Les familles de LD (17)(22)

86

- ❖ $\mathcal{AL} = \{T, \wedge, \neg, A, C \cap D, \forall r.C, \exists r\}$
 - ▼ $\mathcal{ALC} = \mathcal{AL} \cup \{\neg C\}$ négation des concepts définis
 - ▼ $\mathcal{ALU} = \mathcal{AL} \cup \{C \cup D\}$ disjonction des concepts
 $\wedge \equiv C \cup \neg C$ et $C \cup D \equiv \neg(\neg C \cap \neg D)$
 - ▼ $\mathcal{ALE} = \mathcal{AL} \cup \{\exists r.C\}$ ou c-some
qualification existentielle typée
 $\exists r \equiv \exists r.T$ et $\exists r.C \equiv \neg(\forall r. \neg C)$
 - ▼ $\mathcal{ALN} = \mathcal{AL} \cup \{\geq n r, \leq n r\}$ ou atleast et atmost
cardinalité: nombre de valeurs min. et max.
 - ▼ $\mathcal{ALL} = \mathcal{AL} \cup \{r_1 \cap r_2\}$ ou (and $r_1 r_2$)
conjonction des rôle
- ❖ Toute composition possible et \neq^{nce} expressivité d'autres constructeurs existent
- ❖ Cas de $\mathcal{ALCNR} = \mathcal{ALCUNR}$ langage total car
 $C \cup D \equiv \neg(\neg C \cap \neg D)$ $\exists r.C \equiv \neg(\forall r. \neg C)$

- ❖ Subsorption :
- ▼ Un concept D est subsumé par un concept C
 $(D \sqsubseteq C)$ ssi $D^I \subseteq C^I$ pour toute interprétation I.
- ▼ **Attention** : dualité extension/intension de la subsorption. Exemple
 - intension voiture \supset intension véhicule
 - extension véhicule \supset extension voiture
 - ici notation ensembliste
- ▼ C est le subsumant et D est le subsumé
- ▼ La relation est transitive et réflexive et antisymétrique
- ▼ Le haut de la hiérarchie est T et le bas est \wedge

❖ **niveau factuel:**

```

Equipe-moderne(TRIO)
Homme(OLIVIER)
Personne(ROSE)
membre(TRIO, FABIEN)
membre(TRIO, OLIVIER)
chef(ACACIA,ROSE)
(atmost 3 membre) (TRIO)

```

❖ **Inférences:**

- ▼ D subsumé par C ($D \subseteq C$) alors $D(x) \Rightarrow C(x)$
TRIO est une petite équipe
- ▼ FABIEN est une personne
ROSE est une Femme

❖ **Interprétation $I=(\Delta_I, .^I)$ // logiques classiques**

- ▼ domaine d'interprétation Δ_I
- ▼ $.^I$ fonction d'interprétation qui fait correspondre:
 - à un concept, un sous-ensemble de Δ_I
 - à un rôle, un sous-ensembles de $\Delta_I \times \Delta_I$
- ▼ Avec les contraintes suivantes:

$$\begin{aligned}
 T^I &= \Delta_I & \perp^I &= \emptyset \\
 (C \cap D)^I &= C^I \cap D^I & (C \cup D)^I &= C^I \cup D^I \\
 (\neg C)^I &= \Delta_I - C^I \\
 (\forall r.C)^I &= \{ x \in \Delta_I / \forall y: (x,y) \in r^I \rightarrow y \in C^I \} \\
 (\exists r.C)^I &= \{ x \in \Delta_I / \exists y: (x,y) \in r^I \wedge y \in C^I \} \\
 (\geq n \ r)^I &= \{ x \in \Delta_I / |\{ y \in \Delta_I / (x,y) \in r^I \}| \geq n \} \\
 (\leq n \ r)^I &= \{ x \in \Delta_I / |\{ y \in \Delta_I / (x,y) \in r^I \}| \leq n \} \\
 (r_1 \cap \dots \cap r_n)^I &= r_1^I \cap \dots \cap r_n^I
 \end{aligned}$$

❖ **Définitions:**

- ▼ Un concept C est satisfiable ou cohérent ssi il existe une interprétation I telle que $C^I \neq \emptyset$
- ▼ Deux concepts C et D sont équivalents $C \equiv D$ ssi $C^I = D^I$ pour toute interprétation I
- ▼ Deux concepts C et D sont incompatibles ou disjoints ssi $C^I \cap D^I = \emptyset$ toute interprétation I

❖ **Exemples:**

- ▼ Satisfiable / cohérente:
(and Femme (all enfant (and Musicien Homme)))
- ▼ Non satisfiable / incohérente:
(some r (and A (not A)))

❖ La preuve se fait sur les descriptions formelles et non pas par rapport à leur interprétation

91

❖ Base terminologique:

$\Sigma = \{ \text{enfant (PIERRE, MARIE)}$
 $(\text{all enfant (not Musicien)})(\text{PIERRE})$
 $(\text{and Femme (some enfant)})(\text{MARIE}) \}$

❖ Inférence:

$\Sigma \models (\text{not Musicien})(\text{MARIE})$

❖ Interprétation possible:

$\Delta_I = \{ \text{PIERRE, MARIE} \}$
 $\text{PIERRE}^I = \text{PIERRE}$
 $\text{MARIE}^I = \text{MARIE}$
 $\text{enfant}^I = \{ (\text{PIERRE, MARIE}) \}$
 $\text{Femme}^I = \{ \text{MARIE} \}$
 $\text{Musicien}^I = \emptyset$

92

- ❖ Base terminologique:
 - $\Sigma = \{ \text{efg}(\text{XYZ}, \text{ABC})$
 - $(\text{all efg}(\text{not Mno}))(\text{XYZ})$
 - $(\text{and Hjk}(\text{some efg}))(\text{ABC}) \}$
- ❖ Inférence:
 - $\Sigma \models (\text{not Musicien})(\text{ABC})$
- ❖ Interprétation possible:
 - $\Delta_I = \{ \text{PIERRE}, \text{MARIE} \}$
 - $\text{XYZ}' = \text{PIERRE}$
 - $\text{ABC}' = \text{MARIE}$
 - $\text{efg}' = \{ (\text{PIERRE}, \text{MARIE}) \}$
 - $\text{Hjk}' = \{ \text{MARIE} \}$
 - $\text{Mno}' = \emptyset$

- ❖ Opérations:
 - ▼ Test de subsomption: vérifier qu'un concept en subsume un autre (utile pour la classification)
 - ▼ Classification : placer un concept ou un rôle dans sa hiérarchie. Assistance à la construction et l'évolution des hiérarchies.
 - ▼ Test de satisfiabilité: vérifier qu'un concept admet des instances (utile pour vérifier la cohérence)
 - ▼ Identification ou test à l'instanciation : retrouver les concepts les plus spécifiques dont un individu est susceptible d'être une instance
- ❖ Beaucoup de travaux sur les complexité algorithmiques // différentes familles de langages

❖ Brachman

- ▼ KL-One TBox / ABox

❖ Outils:

- ▼ LOOM
- ▼ Classic
- ▼ Back
- ▼ Kris
- ▼ K-Rep
- ▼ etc.

❖ Comparatif inter-langages de Corcho & Gomez-Perez

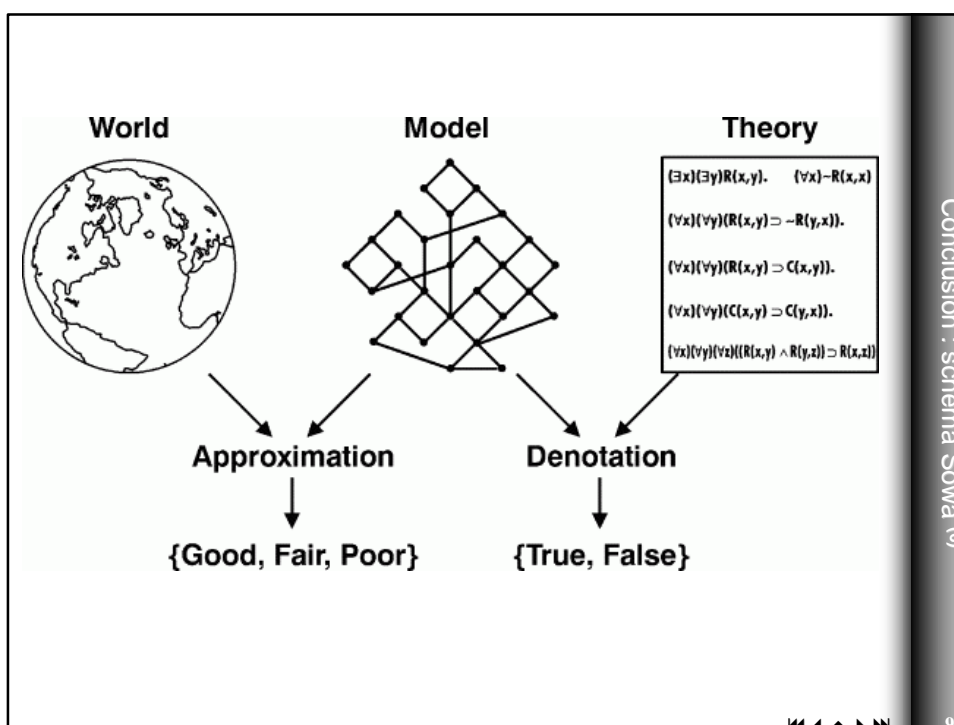
	Ontol ⁵	OKBC	OCML	LOOM	FLogic	XOL	SHOE	RDF(S)	OIL
Concepts	+	+	+	+	+	+	+	+	+
Relations	+	+/-	+	+	+/-	-	+	+	+
Functions	+	+/-	+	+	+/-	-	-	-	+
Procedures	+	+	+	+	-	-	-	-	-
Instances	+	+	+	+	+	+	+	+	-
Axioms	+	+/-	+	+	+	-	-	-	+
Production rules	-	-	+	+	-	-	-	-	+/-

Table 1. Definition of the main elements of domain knowledge.

CONCEPTS	Ontol	OKBC	OCML	LOOM	FLogic	XOL	SHOE	RDF(S)	OIL
METACLASSES	+	+	+	+	+	+	-	+	-
ATTRIBUTES									
Template (instance attrs)	+	+	+	+	+	+	+	+	+
Own (class attrs.)	+	+	+	+	+	+	-	+	+/-
Polymorphic	+	+	+	+	+	-	-	-	+
Local scope	+	+	+	+	+	+	+	+	+
FACETS									
Default slot value	-	+	+	+	+	+	-	-	-
Type constraint	+	+	+	+	+	+	+	+	+
Cardinality constraints	+	+	+	+	+/-	+	-	-	+
Documentation	+	+	+	+	-	+	+	-	+
Procedural knowledge	-	-	+	+	-	-	-	-	-
Adding new facets	+	+	+	+	-	-	-	-	-

Table 2. Definition of concepts.

- ❖ Délimiter l'envergure de l'ontologie
 - ▾ portée du recueil et de la formalisation
 - ▾ expressivité nécessaire pour la formalisation
 - objets : facettes, réflexes, prototypes, valeur défaut
 - graphes : contextes, relations n-aires , lambda
 - logiques de description : classification,
 - ▾ mais aussi monotonie, complétude, décidabilité...
- ❖ Langages hybrides: logiques & objet, objets & contraintes, graphes & acteurs, graphes & règles...
- ❖ Lien avec les BDs





❖ **Palette du peintre**

- ▼ choisir la bonne couleur
- ▼ beaucoup de couleurs
beaucoup de nuances
- ▼ trop de couleurs
complexité des choix

❖ **Approche modulaire?**

- ▼ méthodologies de choix
- ▼ langages en couches
ou modules
- ▼ bibliothèques et critères
de comparaisons
- ▼ passerelles entre
langages et langages
pivots