
Les représentations Objets

Introduction

- La perception et la modélisation du monde réel est un problème philosophique central qui n'a toujours pas de réponse précise.
 - L'objectif de l'IA est de représenter ce monde réel de façon suffisamment fidèle, pour que des programmes puissent agir sur cette représentation et arriver à des conclusions semblables à celle d'un opérateur humain.
 - Dans cette représentation, la notion d'objet semble essentielle.
-

Introduction

- Le monde est peuplé d'objets réels ou abstraits.
 - Ces objets sont perçus par nos sens et d'une certaine façon qualifiés.
 - ⇒ Ils ont donc des qualités (propriétés, attributs).
 - ⇒ D'où l'approche linguistique qui définit les termes en fonction les uns des autres.
-

Rappels : lacunes des réseaux sémantiques

- Imprécision sur le sens des liens et des nœuds:
 - Virtuellement tout peut être représenté par des noeuds et des liens, notamment des faits, des pointeurs, le sens de phrases, des propositions, des actions, des événements, des propriétés, des assertions, des prédicats, des objets, des classes, des ensembles, des relations sémantiques, des relations linguistiques, des relations conceptuelles et arbitraires, entre autres choses.
 - ⇒ Ces différents niveaux de représentation sont mélangés, donc confusion de notation et difficulté d'expliquer le fonctionnement des interpréteurs.
-

Différentes approches

- Du fait des lacunes des réseaux sémantiques différents travaux ont vu le jour :
 - Les Frames: M. Minsky [75] a proposé un autre modèle appelé "frames". Ces "frames" sont des granules de connaissances plus importantes que les nœuds d'un réseau sémantique.
 - Les Logiques de Description: Brachman et Schmolze [77] ont proposé un autre modèle à mi-chemin entre les réseaux sémantiques et les Frames de Minsky dont le premier représentant s'appelle KL-ONE.
 - Les Graphes Conceptuels: Sowa [84].
-

Points communs et différences

- Réification des objets du monde réels
 - Un objet est défini par un ensemble fini de propriétés (plus formellement des prédicats n-aires) qui peuvent être des couples attributs / valeurs ou des relations entre objets - un attribut ou une relation donné peut posséder plusieurs valeurs. Il existe généralement deux catégories d'objets :
 - les objets individuels ou individus ou encore instances, exemple : Médor le chien, Félix le chat
 - Les objets abstraits ou classes, exemple : le chien ou le chat
-

Points communs et différences

- Les objets sont structurés dans une hiérarchie, encore appelée graphe d'héritage, par des liens "is-a" ou "kind-of" qui détermine une relation de généralisation/spécialisation entre les objets abstraits et les liens "member-of" la relation entre objets individuels et objets abstraits.
-

Notations

- Soit $G = (X, H)$ le graphe d'héritage avec X l'ensemble des objets, où :
 - $X = Y \cup Z$ avec Y l'ensemble des « classes » et Z l'ensemble des « instances »,
 - Soit H l'ensemble des arcs, où :
 - $H = H1 \cup H2$ avec :
 - $H1$, l'ensemble des liens is-a entre classes
 - $H2$, l'ensemble des liens member-of
 - \leq_H est l'ordre induit par le graphe G avec
 - ▽ $x, y \in X$ tels que si $x \leq_H y$ alors y est un ancêtre de x et x est un descendant de y .
 - Le graphe G possède toujours une racine unique.
-

Points communs et différences

- A un niveau donné de la hiérarchie, un objet ne possède - déclaré dans l'objet considéré - que les propriétés qui lui sont spécifiques et qui ne sont, en général, pas présentes chez ces ancêtres.
 - Toute propriété non présente dans un objet est recherchée chez ses ancêtres par un parcours de sa hiérarchie.
 - Ce mécanisme de déduction s'appelle mécanisme d'héritage. Le graphe d'héritage peut être :
 - Un arbre, l'héritage est simple dans ce cas
 - Un graphe orienté sans circuit - ou un treillis -, l'héritage est multiple dans ce cas : plusieurs ancêtres ou objets plus abstraits, pour un objet donné.
 - ⇒ En cas d'héritage multiple, il peut y avoir des conflits d'héritage : deux propriétés (couples attribut valeur) de même nom (même attribut) sont héritables et possèdent des valeurs différentes.
-

Points communs et différences

- Dans ce type de représentation, les objets sont définis en extension ou intension.
 - Définition en extension :
 - L'extension propre d'une classe est définie par un ensemble d'individus. L'extension au sens large d'une classe est définie par un ensemble d'instances et de classes plus spécifiques.
 - Définition en intension ou compréhension :
 - Un objet est défini en intension par l'ensemble des propriétés qu'il possède ou hérite.
 - Pour un objet abstrait, il s'agit, dans certains cas, des propriétés communes à tous ses membres (ceux appartenant à son extension).
-

Points communs et différences

- Rapports entre Extensions

- Soient $\text{Extp}(x)$ l'extension propre de l'objet x et $\text{Ext}(x)$ l'extension au sens large de l'objet x , alors

$$\forall x, y \in Y \text{ tels que } x \leq_H y \Rightarrow \text{Ext}(x) \subseteq \text{Ext}(y)$$

$$\forall y \in Y \text{ et } \forall z \in Z, \text{ tels que } z \leq_H y \Rightarrow z \in \text{Extp}(y)$$

- Rapports entre Intensions, dans certains cas

- Soient z un objet, $\text{Int}(z)$ l'intension de z ou ensemble de propriétés définissant z , c'est-à-dire celles déclarées dans z et celles héritées de ces ancêtres, alors

$$\forall x, y \in X \text{ tels que } x \leq_H y \Rightarrow \text{Int}(y) \subseteq \text{Int}(x)$$

- ⇒ Un objet donné possède toutes les propriétés déclarées chez ses ancêtres (raisonnement monotone).
 - ⇒ Cette relation n'est pas vérifiée par tous les systèmes (raisonnement non monotone).
-

Points communs et différences

- L'approche des Frames de Minsky et celle des concepts ou objets conceptuels structurés de Brachman et Schmolze peuvent se comparer grâce aux rapports qu'entretiennent l'intension et l'extension dans ces deux modèles.
-

Modèle Objet conceptuel structuré

- Le modèle classe/instance ou d'objet conceptuel structuré est conforme au modèle classique de la catégorisation ou modèle des CNS :
 - Les catégories sont constituées sur la base de propriétés communes.
 - Chaque propriété prise unitairement est nécessaire et elles sont globalement suffisantes pour décider de l'appartenance à la catégorie.
 - Un exemplaire appartient à la catégorie si et seulement s'il possède l'ensemble des propriétés de la catégorie.
 - Organisation intra-catégorielle : tous les exemplaires d'une catégorie ont un statut identique.
 - Organisation inter-catégorielle : les catégories sont organisées dans une taxinomie par une relation d'inclusion stricte
-

Théorie du Prototype

- Les expériences en psychologie cognitive sur les phénomènes de catégorisation ont mis à jour une structure intra-catégorielle, non plus basée sur l'équivalence des exemplaires de la catégorie, mais sur une distribution de ceux-ci selon un gradient de typicalité.
 - Les exemplaires se distribuent des plus représentatifs aux moins représentatifs.
 - La non équivalence du statut des exemplaires dans une catégorie est un fait empirique relatif aux jugements que les gens émettent sur l'appartenance à une catégorie.
 - ⇒ Rosch a proposé un modèle d'organisation hiérarchique des catégories appelé Théorie du Prototype : certains membres de la catégorie, appelés atypiques, ne partagent pas avec le prototype toutes ses propriétés.
-

Points communs et différences

- Dans le cas d'un modèle CNS, l'intension et l'extension d'objets hiérarchiquement dépendant entretiennent les relations suivantes :
 - $\forall x, y \in X$ tels que $x \leq_H y$
 $\Rightarrow \{ ((\text{Ext}(x) \subseteq \text{Ext}(y)) \rightarrow \text{Int}(y) \subseteq \text{Int}(x)) \}$
 - Dans le cas d'un modèle de prototype, il n'y a plus d'équivalence entre extension et intension.
-

Mécanismes d'inférence

- L'héritage

- C'est un mécanisme de partage de propriétés entre des entités structurées dans une hiérarchie qui induit un raisonnement monotone ou non monotone selon les systèmes.

- Le filtrage

- recherche d'un ensemble d'objets qui satisfont à certains critères donnés.
 - Le filtrage est souvent basé sur une logique à trois valeurs, vrai, faux, et inconnu (en monde ouvert)
 - Un objet ne vérifiant pas les conditions du filtre n'est pas rejeté s'il n'est pas contradictoire avec le filtre
 - Objets répartis en trois classes sûrs, possibles et impossibles
-

Mécanismes d'inférence

- La classification: la classification est l'opération qui permet de placer un objet x dans un graphe d'héritage. Deux cas sont à considérer :
 - Une instance x : il s'agit de trouver tous les objets A , ancêtres de x , qui sont les plus spécifiques. Ces derniers deviendront donc les ancêtres immédiats de x (reliés à x par un seul lien).
 - Une classe x : il s'agit de trouver tous les objets A , ancêtres de x , qui sont les plus spécifiques et tous les objets B , descendants de x , qui sont les plus généraux. Les objets A seront les ancêtres immédiats de x et les objets B les descendants immédiats de x .
-

Points communs et différences

■ Les Frames

- ❑ Plutôt prototype, héritage multiple avec conflits + filtrage + réflexes

■ Logique de Description

- ❑ CNS, héritage multiple sans conflit, classification (subsomption)

■ Graphes Conceptuels

- ❑ Plutôt CNS, héritage multiple sans conflit, {objet = sous-graphe}, objets organisés dans un treillis, classification par appariement de graphes
-

Les Frames

- Minsky propose en une approche basée sur la notion de frame
 - **But** : Résoudre un certain nombre de problèmes en intelligence artificielle, notamment la vision par ordinateur, l'analyse de discours et la compréhension du langage.
 - **Principe** : face à une nouvelle situation, on sélectionne en mémoire une structure de données appelée *frame* qui est la plus proche possible de la situation courante. Cette structure doit subir quelques modifications pour s'adapter à la réalité puis est mémorisée dans le système appelé système de *frames* et ainsi reconnaître ou identifier certains objets.
-

Les Frames

- Un frame est une structure de données qui représente une situation caractéristique ou prototypique
 - Elle se compose de plusieurs éléments, appelés slots ou éléments terminaux (sur différents niveaux).
 - Un frame est une sorte de réseau de relations et de noeuds.
 - Chaque slot peut être lui-même un frame.
 - Chaque slot peut prendre une valeur par défaut qui est remplacée afin de mieux cadrer avec la réalité et servir de variable.
-

Les Frames

- Un frame est une structure à trois niveaux qui sont emboîtés : le triplet (frame, slot, facette)
 - (Frames
 (slot1 (facette_value)
 (facette_value)
 ...)
 (slotn (facette_value)
 (facette_value)))
 - Un slot décrit les différentes propriétés d'un frame
 - Une facette est une modalité descriptive ou comportementale d'un slot. A une facette est toujours associée une valeur.
-

Les Frames

- Facettes déclaratives :

- Facettes de typage :

- \$un, \$liste-de, \$domaine, \$sauf, \$intervalle, \$a-verifier, \$card-min, \$card-max

- Facettes de valeurs :

- \$valeur, \$default, etc. ...

- Facettes procédurales :

- \$si-besoin, \$si-possible, \$si-ajout, \$si-enlève, \$si-succes \$si-echec \$avant-ajout \$apres-ajout \$avant-mod \$apres-mod \$avant-sup \$apres-sup \$sib-filtre etc. ...
-

Les Frames

- (date
 sorte-de \$valeur objet
 jour \$un entier
 \$intervalle [1 31]
 mois \$un chaîne
 \$domaine "janvier" "février" "mars" "avril"
 "mai" "juin" "juillet" "août"
 "septembre" "octobre"
 "novembre" "décembre"
 année \$un entier)
 - (date-23
 est-un \$valeur date
 jour \$valeur 15
 mois \$valeur "avril"
 année \$valeur 1987)
-

Les Frames

- (division-euclidienne
 - sorte-de \$valeur méthode
 - dividende \$un entier
 - diviseur \$un entier
 - \$sauf 0
 - reste \$un entier)
 - Lors de la lecture d'un slot, la valeur peut être obtenue par la facette \$valeur, \$default ou \$si-besoin.
 - Mais avec le mécanisme d'héritage, comment se comportent ces trois facettes si elles sont présentes pour un slot donné dans tous les frames plus généraux que celui considéré ?
-

Les Frames

■ La lecture

- Lecture d'un slot peut entraîner le parcours de la hiérarchie considérée.
 - Trois stratégies principales : en I, en Z et en N.
 - En I, seules les facettes \$valeur sont consultées dans la hiérarchie de l'objet.
 - En Z, les facettes \$valeur, \$default et \$si-besoin sont consultées dans cet ordre à chaque niveau hiérarchique du bas vers le haut.
 - En N, les facettes \$valeur sont consultées sur toute la hiérarchie (du bas vers le haut) puis on fait de même avec les facettes \$default puis \$si-besoin.
-

Les Frames

■ Ecriture

- ❑ Vérifications à priori dans l'ordre descendant de la hiérarchie (examen des facettes de typage et déclenchement des réflexes "si-possible"),
 - ❑ Mise en place de la valeur et enfin propagations des modifications, déclenchement des réflexes si-ajout dans l'ordre descendant de la hiérarchie.
-

Les Frames

- Suppression
 - il y a d'abord l'effacement de la valeur puis propagation des réflexes "si-enlève" dans l'ordre ascendant de la hiérarchie.



Différents modèles de frames

- On peut scinder la famille des frames, par exemple, en deux sous familles :
 - Une sous famille dérivée du modèle classes/instances
 - Une sous famille inspirée de la théorie du prototype
-

Différents modèles de frames : Modèle Classes/Instances

(meuble			
sorte-de	\$valeur		objet)
(table			
sorte-de	\$valeur		meuble
forme	\$un		chaîne
	\$domaine		"ronde" "ovale"
			"rectangulaire" "carrée"
			"triangulaire"
nbr-pieds	\$un		entier
	\$intervalle		[1 15]
composé-de	\$liste-de		\$un
			plateau
			\$liste-de pieds)
(plateau			
sorte-de	\$valeur		objet)
(pieds			
sorte-de	\$valeur		objet)

⇒ Un slot muni d'un domaine de valeurs est équivalent à une disjonction de couples (attribut/valeur) – ou de propriétés.

Différents modèles de frames : Modèle Prototype

```
( pieds
  sorte-de          $valeur          objet )
( meuble
  sorte-de          $valeur          objet )
( table
  sorte-de          $valeur          meuble
  forme             $un              chaîne
                  $défaut            "rectangulaire"
  nbr-pieds         $un              entier
                  $défaut            4
  composé-de       $liste-de        $un          plateau
                  $liste-de        pieds )
( plateau
  sorte-de          $valeur          objet )
( pieds
  sorte-de          $valeur          objet )
```

⇒ Un sous objet de table n'aura pas obligatoirement les mêmes propriétés puisque les valeurs des attributs sont par défaut.

LISP : Fonctions d'entrée-sortie

- De nombreuses macros ou fonctions permettent de lire (*read*) et écrire (*print*) sur des flots *ouverts*.
 - Les fonctions d'écriture telles que *print* écrivent par défaut un objet sur la sortie standard (flot prédéfini et toujours ouvert, référencé par la variable globale **standard-output**). Un argument mot-clé permet d'indiquer un autre flot.
 - Les fonctions de lecture lisent par défaut sur l'entrée standard **standard-input** et acceptent un argument mot-clé qui change le comportement par défaut.
-

LISP : Fonctions d'entrée-sortie

- Lecture : la fonction *read*

- La fonction *read* lit une *expression lisp* et la retourne.

```
>(read)
```

```
Pomme
```

```
Pomme
```

```
>
```

- **Attention** : La fonction *read* attend indéfiniment qu'une expression lisp complète soit tapée pour ensuite la retourner.
-

LISP : Fonctions d'entrée-sortie

■ Ecriture

- Il existe plusieurs formats d'écriture : *print*, *princ*. La fonction *format* est la fonction générale d'écriture qui permet de formater du texte :

(format destination contrôle param-1 param-2 ...)

- 1er argument obligatoire : *destination*. Il spécifie où écrire (fenêtre, fichier ...)
 - 2ème argument : chaîne de caractères de contrôle pour spécifier quoi écrire et comment écrire les arguments optionnels (param-1, param-2 ...)
-

LISP : Fonctions d'entrée-sortie

■ Ecriture

□ Différentes valeurs de destination :

- T : écrire dans la fenêtre de l'interacteur

```
>(format t " Introduire un nombre : ")
```

```
Introduire un nombre :
```

```
Nil
```

```
>
```

- Nil : écrire dans une chaîne de caractères qui est retournée comme valeur de l'appel *format*

```
>(format nil " Introduire un nombre : ")
```

```
"Introduire un nombre :"
```

```
>
```

LISP : Fonctions d'entrée-sortie

■ Ecriture

- Différentes directives à insérer dans la chaîne de contrôle
 - ~% : impose un saut à la ligne.
 - ~& : impose un saut de ligne, sauf si le curseur est déjà en début de ligne.
 - ~a indique une position à remplir par un des arguments optionnels dans l'ordre d'apparition.
 - ~{str~} : représente une construction itérative. L'argument doit être une liste, laquelle est utilisée comme un ensemble d'arguments comme pour un appel récursif de format. La chaîne str est utilisée comme contrôle. Chaque itération absorbe autant d'éléments de la liste qu'il est besoin d'arguments.
-

LISP : Fonctions d'entrée-sortie

- Ecriture

```
>(format nil "~a plus ~a égale ~a" 2 3 5)
```

```
"2 plus 3 égale 5"
```

```
>(format nil " The winners are : ~{ ~a ~}." '(fred  
  harry jill))
```

```
"The winners are : fred harry jill."
```

```
>
```

LISP : les fichiers

- La fonction *Load*

Load *filename* &key :verbose :print :if-does-not-exist

- charge le fichier de nom *filename* dans un environnement lisp.

- La fonction *Open*

Open *filename* &key :direction :element type :if-exist :if-does-not-exist :external-format

- La fonction *open* permet d'ouvrir un flot sur un fichier. Le premier paramètre obligatoire correspond au chemin d'accès au fichier. Viennent ensuite des paramètres mots-clés, en particulier *:direction* qui indique le sens du flot, entrée ou sortie (en Anglais : *input* ou *output*). .
-

LISP : les fichiers

:direction, cet argument indique si le flot doit être en input, output ou les deux

- ❑ **:input**, valeur par défaut,
- ❑ **:output**
- ❑ **:io**

:element-type, cet argument précise l'unité de transaction sur le flot

- ❑ **string-char**, unité de transaction par défaut. Les fonctions read-char et/ou write-char peuvent être utilisées sur le flot.
 - ❑ **Character**, . Les fonctions read-char et/ou write-char peuvent être utilisées sur le flot.
 - ❑ **Signed-byte, bit, etc.**
-

LISP : les fichiers

- **:if-exists**, cet argument spécifie l'action à effectuer lorsque la direction est output ou io et que le fichier spécifié existe déjà.
 - **:error**, action par défaut
 - **:new-version**, crée une nouvelle version du fichier avec un numéro de version plus élevé
 - **:rename**, renomme le fichier existant et crée le nouveau fichier avec le nom précisé.
 - **:rename-and-delete**
 - **:overwrite**, utilise le fichier existant. Les opérations d'écriture vont modifier le fichier. Si la direction est io, le pointeur de fichier est positionné en début de fichier. Ce mode est plus utilisé quand la fonction file-position peut être utilisée sur le flot.
 - **:append**, utilise le fichier existant, les opérations d'écriture vont modifier le fichier. Le pointeur de fichier est positionné à la fin du fichier.
-

LISP : les fichiers

- **:if-does-not-exist**, cet argument spécifie l'action à effectuer si le fichier spécifié n'existe pas
 - **:error**
 - **:create**
 - **nil**, le flot est initialisé à nil
-

LISP : les fichiers

- La macro *with-open-stream* exécute un certain nombre d'opérations sur un flot ouvert, renvoie une valeur puis ferme le flot. Elle a deux paramètres. Le premier est un nom de variable qui prendra comme valeur le flot correspondant au deuxième paramètre. Le corps est une suite d'expressions formant un *progn* implicite.
 - La macro *with-open-file* est similaire mais au préalable elle ouvre un flot sur un fichier avec *open*.
-

LISP : Les fichiers

- ```
>(defparameter *flot-sortie* (open "fichier" :direction :output))
FLOT-SORTIE
> (print '(1 2 3) *flot-sortie*)
(1 2 3)
>(print '(A B) *flot-sortie*)
(A B)
> (close *flot-sortie*)
T
> (with-open-file (flot-entree (open "fichier" :direction :input))
 (defparameter *expr1* (read flot-entree))
 (defparameter *expr2* (read flot-entree)))

EXPR2
> *expr1*
(1 2 3)
>*expr2*
(A B)
>
```
-

---

## LISP : les fichiers

- La fonction *close* ferme un flot ouvert. La connexion entre le flot et le fichier associé est terminée mais l'objet flot existe toujours en tant que flot non ouvert.
  - **Delete-file** *file* détruit le fichier spécifié.
    - Le *file* doit être une chaîne, un chemin ou un flot.
-