# Représentation des Connaissances : Introduction aux Frames

Bernard ESPINASSE
Professeur à l'Université d'Aix-Marseille

2008

**Plan**

- Les Frames et Scripts : l'approche objet en IA
- Notion de Frame (Schéma) : composant, attribut, aspect
- Héritage et inférences dans les Frames
- Introduction aux Scripts



## Les Frames, objets ...

**Faiblesses des Réseaux Sémantiques :** représentation **déclarative** de la connaissance (problèmes dans l'interprétation)

→ de nouveaux outils ont été développés : **représentations « orientées objet » :**

**Frames (schémas), Scripts, langages objets**

• représentations des connaissances **MIXTE :**

**déclaratives** et **procédurales**

afin de :

• mieux représenter des **connaissances par nature structurées**

• d'en **faciliter une part d'interprétation**



## L'approche objet en IA

**objet 1 — *relation* —> objet 2**

boite — *est_sur* —> table

boite — *est_de_couleur* —> rouge

on peut regarder ces descriptions de 2 façons différentes :

• **en privilégiant les relations :**

→ passer par la relation pour accéder aux objets qu'elle relie :

***est_sur* (boite, table)**

réseaux sémantiques, logique, ...

• **en privilégiant les objets :**

→ accéder à un objet pour vérifier s'il possède (ou non) une relation avec un autre objet

• **boite : *est_sur* : table**

• **table : *est_sous* : boite**

frames (schémas), objets,...



## L'approche objet en IA

• **de façon générale, en informatique traditionnelle :**

• **les programmes = éléments actifs**

*→ exécutent les opérations sur des ensembles de données*

• **les données = éléments passifs**

*→ subissent les opérations des programmes*

*(d'où : programmer = définir les opérations qui agissent sur de données)*

———

• **avec les langages orientés objets : renversement !!!!**

• **les objets (données) = éléments actifs**

• **les objets sont caractérisés par les opérations qu'ils connaissent**

## Notion de Frame (Schéma)

On doit la notion de Frame (schéma) à Minsky (1975)

• **Frame = structure de données générale**

• permettant de **décrire la connaissance** que l'on a sur des **objets** (concrets ou abstraits)

On peut rattacher **2 types d'informations** à un Frame :

→ **des informations relatives à la description des objets : partie déclarative**

- propriétés
- relations d'états
- relations entre les objets, …

→ **des informations relatives à la manipulation des objets : partie procédurale**

c.a.d. les opérations (procédures & méthodes) connues par l'objet :

- définition du contexte d'activation de l'objet
- comment calculer une propriété
- action à exécuter dans un contexte donné, …



## Notion de Frame

• **2 types de Frames :**

• **les frames « prototypes » :**

→ représentent en **intention** une **classe d'objets**
→ description du **contexte** attaché à cette classe

• **les frames « instances » :**

→ réalisations **particulières** d'une classe donnée
→ description des **individus** d'une classe

- oiseau → classe : frame **prototype**
- canari → classe : frame **prototype**
- titi → individu : frame **instance**

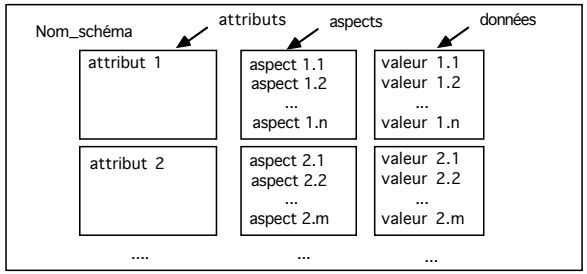


## Composants d'un Frame : attributs et aspects (1)

• **attribut** (slot) = information d'un frame permettant d'introduire :

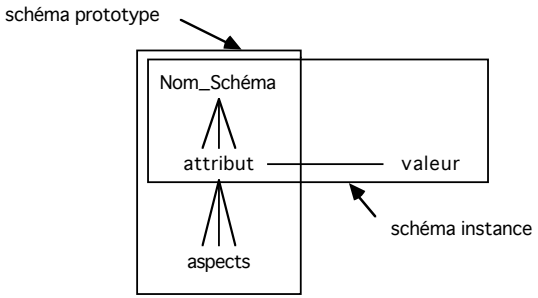
- des **propriétés** (1 ou n) **décrivant le frame** :
  - dans frame prototype : introduction de **valeurs permises**
  - dans frame instance : introduction de **valeurs effectives**
- des **relations** ( un-aire ou n-aire) **entre frames**
- un attribut est **structuré** par des **aspects**

• **aspect** = chaque aspect introduit une valeur élément de description de l'attribut






## Composants d'un Frame : attributs et aspects (2)




- chaque attribut d'une instance doit avoir l'**aspect "valeur"**,
- le frame d'instance ne diffère du frame prototype correspondant que par la donnée supplémentaire de la valeur de l'attribut,
- une instance **hérite** directement du frame de sa classe
- une instance peut être **partielle** ou **complète**,
- une instance de frame est liée à son frame prototype par l'attribut "**est_un**"
- 2 frames prototypes sont liés par l'attribut "**sorte_de**"

## Attribut d'un Frame

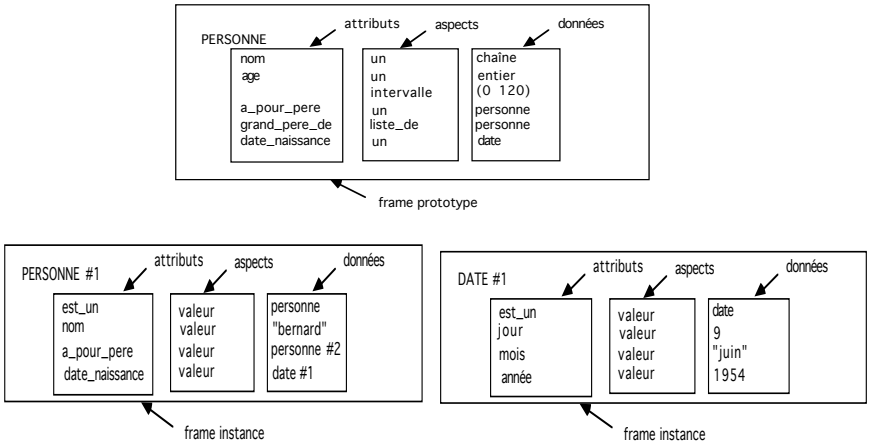
- **un attribut peut :**
  - permettre **d'introduire des valeurs effectives** (frame instance),
  - être un **pointeur sur un frame** plus **général**, plus **spécifique** voire alternatif (vers frame prototype),
  - **définir une valeur par défaut**, ou définir un ensemble de valeurs permises (frame prototype),
  - **constituer un démon**, sorte de procédure qui peut être activée chaque fois que l'attribut doit être modifié,
  - être un **pointeur sur un autre frame** (récursivité de frame), ce qui permet de représenter des objets composés

- **attributs et héritage :**
  - si la valeur d'un attribut est défini au niveau d'un prototype, tous les frames (prototypes et instances) plus spécifiques en héritent
  - les instances héritent des méthodes définies au niveau prototype



## Attribut d'un Frame

Exemple de frame (tiré de SHIRKA de RECHENMANN) :






## Les aspects d'attributs de Frame

- **un attribut est structurés par des aspects**
- chaque aspect introduit **une dimension décrivant l'attribut**
- l'ensemble des aspects est fixé par le langage, c'est ce qui **définit la sémantique de la représentation**

- **plusieurs types d'aspects** peuvent être rencontrés :

  - **Aspect de typage**
  - **Aspect d'obtention de valeur :**
  - **Aspect réflexe**
  - **Aspect de contrôle**
  - **Aspect de communication homme-machine**
  - **...**



## Les aspects d'attributs de Frame

- **Aspect de typage** : permet de définir le type de valeur des attributs d'un frame (valeurs permises) :
  - type simple: entier, bouléen, chaîne de caractères, ...
  - type composé: un autre frame de telle nature, frame "personne" dans l'exemple précédent.

  peut aussi donner une indication de **restriction de typage** :

  - sur un intervalle de valeur; aspect "**intervalle**",
  - une nono-valuation ou multi-valuation; aspect "**un**" et "**liste_de**",
  - l'aspect "**à_vérifier**" introduisant une procédure à satisfaire pour toute valeur,

- **Aspect d'obtention de valeur** : permet de spécifier des procédures d'obtention d'une valeur :
  - aspect "**valeur**": introduit la valeur de l'attribut
  - aspect "**si_besoin**": introduit une méthode de calcul de la valeur
  - aspect "**défaut**": introduit une valeur par défaut (traitement des exceptions).

## Les aspects d'attributs de Frame

**Aspect reflexe** :

permet d'introduire des procédures déclanchées en cas d'ajout, de modification, de suppression d'informations, ceci notament pour des soucis de maintient de cohérence :

aspect "**si_modif**", "**si_ajout**", "**si_supprime**",

**Aspect de contrôle :**

permet d'introduire des procédures définissant des actions à déclancher en cas d'échec ou de succés d'obtention de valeur, pouvant conduire à une propagation dans d'autres frames. Par exemple:

aspect "**si_succés**" , "**si_échec**", ...

**Aspect de communication Homme-machine :**

permet d'introduire des procédures permettant de passer d'une vision interne à une vision externe des valeurs ou réciproquement :

aspect "**lire**", "**écrire**", ...



## Héritage dans les frames : héritage simple

**Les frames s'inscrivent dans une structure de demi-treillis** :

- un frame peut **dominer** un ensemble de frames plus spécifiques et
- un frame peut **être dominé** par un ou plusieurs frames plus généraux

**2 situations d'héritage simple :**

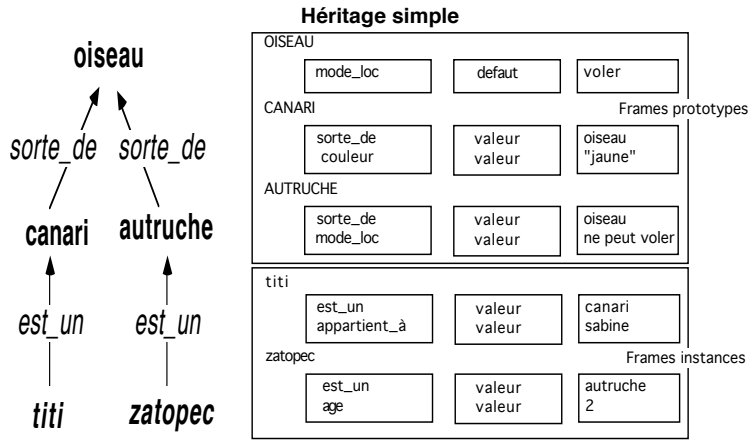

- d'un frame instance de son frame prototype (relié par un attribut "est_un")
- d'un frame prototype d'un autre frame prototype (relié par un attribut "sorte_de"

**d'une façon générale :**

- si la valeur de l'attribut est défini au niveau d'un **prototype**, **tous les frames prototypes et instances plus spécifiques en héritent**
- il y a aussi héritage des **procédures** ou **méthodes** définies au niveau des prototypes au niveau des instances.



## Héritage dans les frames



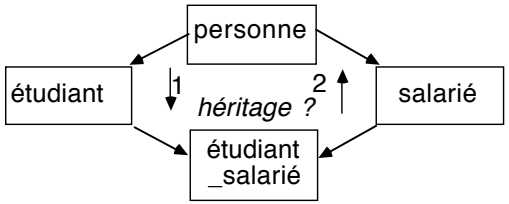

**Héritage simple**

| | | | |
|---|---|---|---|
| OISEAU | | | |
| | mode_loc | defaut | voler |
| CANARI | | | Frames prototypes |
| | sorte_de couleur | valeur valeur | oiseau "jaune" |
| AUTRUCHE | | | |
| | sorte_de mode_loc | valeur valeur | oiseau ne peut voler |
| titi | | | |
| | est_un appartient_à | valeur valeur | canari sabine |
| zatopec | | | Frames instances |
| | est_un age | valeur valeur | autruche 2 |



## Héritage dans les frames : héritage multiples

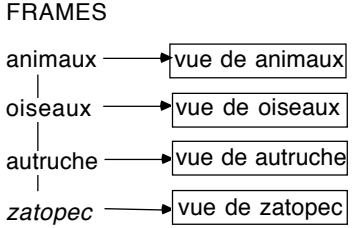
- **lorsqu'un schéma est dominé par plusieurs autres frames**
- **définir l'ordre d'application des héritages**, par exemple par une liste de précédence ou une énumération des frames à héritage :




- sens de parcours possible pour l'héritage :
  1) étudiant_salarié
     étudiant
     personne
     salairé
  2) étudiant_salarié
     salairé
     personne

## Inférence d'instanciation dans les frames

- **l'instanciation de frames** de type instance constitue **l'inférence fondamentale** dans les systèmes de frames
- consiste à "**construire**" ou "**compléter**" une instance de frame = **instancier par des valeurs les attributs définis dans les frames prototypes** auxquels est associé l'instance de frame considérée
- cette obtention de valeurs peut être **directe**; valeurs d'attributs propres à l'instance, ou par héritage.
- Si l'on appelle "**vue**" l'ensemble des attributs que possède un frame
- une **instance de ce frame sera l'union de toutes les vues propres et héritées** :

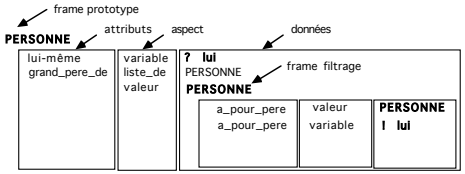





## Inférence par filtrage dans les frames

- intervient quand **l'aspect valeur d'un frame prototype renvoie sur un autre frame**
- ce dernier frame intervient alors comme un **filtre** :
  - **décrivant les instances possibles** : valeurs que cet attribut pourra prendre
  - **contenant généralement des conditions supplémentaires à vérifier** sur les valeurs de ce frame restreignant encore les instanciations possibles.

Exemple :



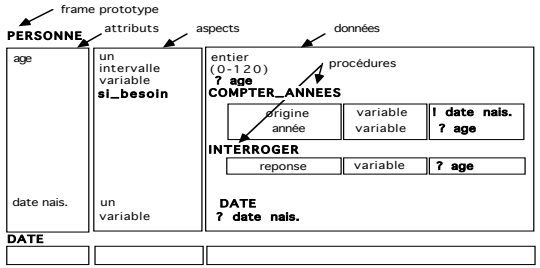

- dans "**! X**", préfixe "! ..." valeur de la variable X doit être instanciée à l'appel du filtre (**variable d'entrée**)
- dans "**? X**", préfixe "? ..." valeur de la variable X peut être inconnu à l'appel du filtre (**variable de sortie**)

Dans cet exemple:
- l'attribut "**lui_même**" du frame prototype principal représente le frame,
- l'aspect "variable" de cet attribut nomme une variable "**?lui**" qui représente l'attribut en question,
- à l'instanciation, dès que l'attribut reçoit une valeur, cette valeur est affecté à la variable,
- le déclenchement du filtre peut conduire à un retour arrière.



## Inférence procédurale dans les frames

- **obtenue par activation des procédures**, **méthodes** associées aux attributs afin d'en obtenir des valeurs d'instanciation
- ces procédures sont elles-mêmes décrites par des frames prototypes, dont les attributs sont leurs valeurs d'entrée et de sortie
- c'est l'aspect "**si-besoin**" qui distingue un frame de filtrage d'une telle procédure :




Exemple :
- l'âge est calculé à partir de la date de naissance "date nais." par la procédure COMPTER_ANNEES.
- Si cette date de naissance n'est pas connue, on interroge l'utilisateur pour obtenir l'âge avec la procédure INTERROGER.



## Quelques implémentations de frames (1)

De nombreuses implémentations de frames ont été réalisées en langage LISP ou Prolog

- **FRL :** "Frame Représentation Language" [ROBERTS & GOLDSTEIN 1977] (développé au MIT : un précurseur, assez rudimentaire)
  - les frames entre eux sont organisés **hiérarchiquement**
  - une base de connaissances en FRL = {frames} dont les attributs peuvent avoir les aspects suivants: valeurs limites, valeurs par défaut, contraintes et procédures (développées en LISP) pouvant être déclanchées quand une valeur sera dépassée ou non dépassée, voire utilisée par un autre attribut.

- **KRL :** (BOBROW & WINOGRAD - Xerox PARC)
  - s'inspire directement des idées de M.MINSKY, est plus ambitieux que le précédent.
  - il dispose d'opération d'unification de frames qui peuvent être contrôlées par le concepteur de la base de connaissances
  - il possède aussi la notion de "contexte de croyance" pouvant servir à définir un "mécanisme d'attention" (attention focusing mechanism)
  - il inclut de la "self Knowledge" dans le base de connaissance en fournissant des descriptions de descriptions.

## Quelques implémentations de frames (2)

- **les "unit" de KRL:**
  - toute entité conceptuelle à représenter est décrite par un frame appelé "**unit**" appartenant à une "catégories" (7 catégories dans KRL) définissant son niveau d'abstraction: objet générique, spécialisé ou élémentaire
  - une unit est définie par un ensemble d'attributs, décrits par de descripteurs.
  - les concepts généraux sont décrits par des units appartenant aux catégories "**basic**" et "**abstract**" :
    - les catégories "**basic**" définissent des particules de connaissances en arbre d'héritage simple dont elles sont les racines, ces particules peuvent être instanciées

      EX : définition de l'unit basic "CHAT":

      ```
      Chat UNIT Basic
      <SELF>
      <nom (a string)>
      <race (a Race) Persan; DEFAULT) >
      <age (an integer)>
      <propriétaire (a string)>]
      ```

    - les catégories "**abstract**" définissent des particules de connaissances plus générales qui ne peuvent être spécialisées.



## Conclusion sur les Frames

**Forces :**

- Très **séduisant** et **pratique** pour **exprimer la connaissance**
- Utilisé pour le **développement d'ontologies** (Protégé - Frame)
- Dans de nombreux **systèmes experts**
- Extension du concept de frame (R.C.Schank et R.Abelson), **scripts** = séquence d'évènements dans un contexte particulier, associés au concept de sénario
- …

**Faiblesses :**

- **Pas d'équivalence avec la logique** (ex : quantifieurs, disjonctions, …)
- **Pas d'expression de connaissances incertaines**, **imprécises**, **hypothétiques**
- Certaines **inférences** sont favorisés, d'autres sont **difficiles à réaliser**
- …