Langage à objets et logique de descriptions : Schéma d'intégration

Roland Ducournau — LIRMM – Université Montpellier 2

Gabriel Pavillet — C3T – Montpellier

Plan

1. Langage à objets et logique de descriptions
2. Objectif de l'intégration
3. Schéma de l'intégration
4. Comparaison
5. Conclusion, Perspectives

1. Langage à objets et logique de descriptions

2 paradigmes opposés :

- *programmation* : procédural, sémantique opérationnelle, non monotone (changements d'états) ;
- *représentation des connaissances* : déclaratif, sémantique logique intuitive, monotone.

1 modèle commun :
le modèle objet : classes, instances et attributs.

1.1. Le modèle objet standard

1. des *classes*,
2. chaque classe décrit des *propriétés* (attributs et méthodes),
3. des *objets*, créés par instanciation des classes,
4. une relation de *spécialisation* entre classes, qui supporte un mécanisme d'héritage des propriétés,
5. une abstraction procédurale : l'*envoi de message* ;
6. (éventuellement) la *réflexivité* : les classes sont des objets comme les autres.

L'ensemble est muni d'une sémantique opérationnelle, basée sur la spécification des mécanismes d'héritage et d'envoi de message.

1.2. Les logiques de descriptions

1. 3 catégories de bases : *concepts*, *individus* et *rôles*, décrits par ;
   - un langage de *termes*, constitué de *constructeurs* ;
   - des *axiomes terminologiques* pour introduire de nouveaux concepts et rôles, la *TBox* : $C \doteq D$ ou $C \dot{\leq} D$;
   - un langage d'*assertions* pour introduire des individus, la *ABox* : $a : C$ ou aRb .
2. une sémantique formelle,
   - constructive sur le langage de termes
   - contrainte par les axiomes terminologiques et les assertions.
3. un système de déduction, avec 2 problèmes :
   - la *subsomption* d'un concept par un autre ;
   - l'*appartenance* d'un individu à un concept.

   et une problématique générale : la *classification*, de classes ou d'instances.

Syntaxe et sémantique du langage de termes de la famille $\mathcal{AL}$

syntaxe	sémantique	
$C, D \longrightarrow$		*concepts*
A	$A^I \subseteq \Delta$	*concept atomique*
$\top$	Δ	*top*
$\bot$	$\emptyset$	*bottom*
$\neg C$	$\Delta - C^I$	*négation*
$C \sqcap D$	$C^I \cap D^I$	*conjonction*
$C \sqcup D$	$C^I \cup D^I$	*disjonction*
$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^I \Rightarrow d_2 \in C^I\}$	*universelle*
$\exists R.C$	$\{d_1 \mid \exists d_2 \in C^I : (d_1, d_2) \in R^I\}$	*existentielle*
$\geq nR.C$	$\{d_1 \mid \lvert\{d_2 \in C^I \mid (d_1, d_2) \in R^I\}\rvert \geq n\}$	*cardinalité min*
$\leq nR.C$	$\{d_1 \mid \lvert\{d_2 \in C^I \mid (d_1, d_2) \in R^I\}\rvert \leq n\}$	*cardinalité max*

1.3. Exemple

```
Personne ≤̇
        and(forall(mère,Personne),
            forall(père,Personne),
            exists(père),
            exists(mère))

Parent   ≐
        and(Personne,
            forall(enfants,Personne),
            exists(enfants))
```

```
class Personne
    mère : Personne
    père : Personne
    methods { ... }
end Personne

class Parent ako Personne
    enfants : Personne
    methods { ... }
end Parent
```

1.4. Correspondances

concept	classe	
individu	objet	(instance)
rôle	attribut	(relation/association)
subsomption	spécialisation	(héritage, sous-typage)
appartenance	instanciation	(vérification de type)
classification d'instances	migration d'instances	
classification de classes	réorganisation de hiérarchies	
construction	description	
déduction	programmation	

1.5. Sémantique commune

Sémantique « naturelle » (Aristote et Cie) de la spécialisation :
Socrate est un homme, les hommes sont mortels, donc Socrate est mortel

Si $B \prec A$:

- inclusion des extensions : $Ext(B) \subseteq Ext(A)$
  subsomption des logiques de descriptions ou du sous-typage
- inclusion des intensions : $Int(B) \supseteq Int(A)$
  sous-typage
- inclusion des domaines des attributs : $Dom(P, B) \subseteq Dom(P, A)$
  cf. covariance / contravariance du sous-typage

1.6. Différences dans le modèle

Dans les logiques de descriptions, 2 différences importantes dans les rapports entre :

- objets et classes : *multi-instanciation*
  $a : C$ et $a : D$, C et D sans sous-classe commune
- objets et attributs : pas d'*explication des attributs*
  aRb n'implique pas que R est décrit dans les classes de a
  (dans les objets, c'est une erreur de type)

Pas de sémantique pour la réflexivité.

2. Objectif de l'intégration

- pouvoir programmer et raisonner sur les mêmes objets ;
- 3 possibilités :
  1. intégration symétrique (les 2 systèmes sont co-extensifs),
  2. intégration de la programmation dans les logiques de descriptions
     Möller dans OOPSLA'96 ;
  3. **intégration de la logique de descriptions dans le langage à objets** ;
     notre choix
- dans tous les cas, alignement sur l'une ou l'autre des sémantiques pour
  - l'instanciation (mono ou multi)
  - l'explication des attributs (avec ou sans)

3. Schéma d'intégration

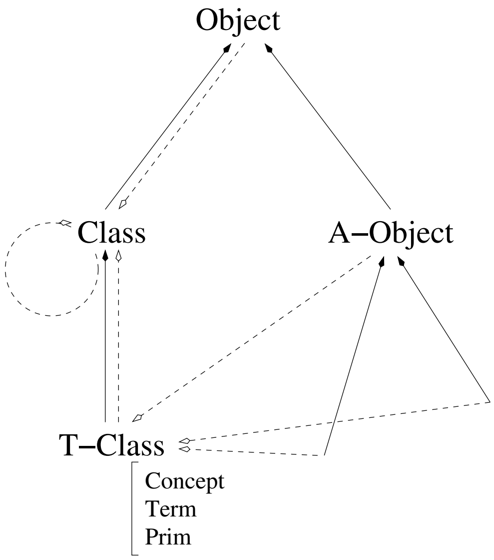
- choix d'un langage à objets et d'une logique de descriptions
  préexistants et raisonnablement quelconques ;
- conservation de la sémantique spécifique aux langage à objets :
  mono-instanciation et *explication des attributs* ;
- mise en correspondance terme à terme des éléments des 2 approches :
  1. classes et concepts,
  2. objets et individus,
  3. attributs et rôles ;
- conservation de la programmation par objets habituelle sur les objets-individus.

3.1. Classes et concepts

Principe

- associer des concepts à des classes :
  - ⇒ définir des classes particulières,
  - ⇒ besoin de méta-classes et de réflexivité ;
- faire correspondre spécialisation et subsomption :
  - ⇒ héritage de ces concepts
  - ⇒ distinguer *Terme* différentiel et *Concept* cumulé ;
- différencier concepts définis et primitifs ;
- imposer la mono-instanciation sur les concepts associés aux classes.

⇒ **Nécessite un langage objet réflexif**



`A-Object` : classe de tous les individus

`T-Class` : classe de tous les concepts

`A-Object` et ses sous-classes
sont instances de `T-Class`

- Concept associé à une classe :
  - $Term(C)$ terme différentiel explicité dans la classe ;
  - $Concept(C)$ terme calculé

$$
\begin{aligned}
Concept(\texttt{A-Object}) &= \top \\
Concept(C) &= Term(C) \sqcap \big(\bigsqcap_{C' \in Super(C)} Concept(C') \big)
\end{aligned}
$$

- Concept primitif : composant primitif à la Nebel

$$Concept(C) = Prim(C) \sqcap Term(C) \sqcap \big(\bigsqcap_{C' \in Super(C)} Concept(C') \big)$$

Mais la mono-instanciation n'est pas traitée.

Mono-instanciation en héritage simple :

- 2 classes incomparables sont incompatibles :
- solution : rajouter à la conjonction de $Concept(C)$ la négation des concepts de ses classes sœurs.

$$Concept(C) = Term(C) \sqcap Concept(Super(C)) \sqcap \big(\bigsqcap_{C' \in Sub(Super(C))} \neg Concept(C') \big)$$

$\Rightarrow$ **Nécessite une logique avec la négation généralisée** ($\mathcal{ALC}$)

Mono-instanciation en héritage multiple :

- 2 classes incomparables ne sont pas incompatibles,
- leurs instances communes doivent être instances de l'une de leurs sous-classes communes.
- solution : rajouter des axiomes terminologiques

$$C \sqcap C' \dot{\leq} \bigsqcup_{D \in Inf(C,C')} D$$

  pour tout C, C', où $Inf(C, C')$ est l'ensemble des minorants maximaux de C et C'.
- mais il est nécessaire d'avoir *fermé* la hiérarchie : plus de définition de classes possible.

⇒ **Nécessite une logique avec *TBox* libre :**
avec axiomes de subsomption généralisés, $C \dot{\leq} D$, où C est un terme quelconque.

3.2. Objets et individus

Principe : faire correspondre assertions et instanciation,
classification d'instance et migration d'instances :

- `make-instance(C,a)` et `change-class(C,a)` (suivant que a existe ou pas) sont équivalents à l'assertion $a : C$;
- `change-class(C,a)` n'est effectif que si C est une sous-classe de la classe courante de a :
  ⇒ possibilité de multi-instanciation provisoire ;
- l'inconsistance entraîne la levée d'une exception et le retrait de la dernière assertion.

⇒ **Nécessite un langage avec migration d'instances**,
au moins de façon monotone (de la classe à la sous-classe).

3.3. Attributs et rôles

Deux points à régler :

- problème sémantique : l'explication des attributs ;
- problème d'implémentation : comment implémenter les rôles ?
  physiquement ou logiquement ?

Seule restriction : les accès en écriture sont monotones

`add-slot-value` mais pas `set-slot-value`.

Explication des attributs :

- calculer $Def(R)$, l'ensemble des concepts C qui introduisent le rôle R :
  $\forall R.C$ ou $\exists R.C$ dans $Term(C)$
- *fermer* la hiérarchie ;
- rajouter les axiomes terminologiques :

$$\exists R.\top \dot{\leq} \bigsqcup_{D \in Def(R)} D$$

⇒ **Nécessite une logique avec *TBox* libre et disjonction**

Implémentation des rôles :

- physiquement, comme un attribut :
  - ⇒ générer automatiquement les attributs,
  - ⇒ propager les assertions ;
- logiquement, par des accesseurs qui génèrent des requêtes et des assertions :
  - ⇒ générer automatiquement les accesseurs ;
- mixte : physiquement pour les rôles mono-valués, logiquement pour les multi-valués.

3.4. Spécification

- langages de programmation
  - réflexivité
  - migration d'instances
- logique de descriptions
  - négation généralisée
  - *TBox* libre

4. Comparaisons

- toute la tradition des langages de *frames*, avec mélange de programmation et de représentation des connaissances, mais sans sémantique propre ;
- 2 travaux directement comparables
  - Rathke, Workshop IJCAI 1993 : démonstration du MOP CLOS pour décrire des concepts (pas de *ABox*) ;
  - Möller, OOPSLA 1996 : utilisation du MOP CLOS pour appliquer les fonctions génériques aux concepts et individus de CLASSIC ; (intégration dans l'autre sens) ;
- $\mathcal{ALUNI}$: tentative d'unification de plusieurs modèles objet, sur la base des logiques de descriptions :
  application erronée aux *frames* (KEE) sur la base d'un modèle de purs prototypes, avec la sémantique logique de Hayes (pas de *ABox*) ;
- OIL : synthèse récente entre *frames* et logiques de descriptions pour la définition d'ontologies.

5. Conclusion et perspectives

- réalisation en cours en CLAIRE et FACT
- performances attendues :
  - la déduction dans les logiques de descriptions est NP-difficile ;
  - l'intégration n'améliore rien ;
  - les axiomes pour la mono-instanciation sont équivalents au complété de MacNeille (exponentiel !)
- validation de l'approche :
  - convergence significative avec Möller 96 ou OIL,
  - application significative nécessaire ;
- améliorer l'intégration en transformant les axiomes de subsomption en règles de déduction supplémentaires.