



Systemes d'information logiques

Sébastien Ferré, Olivier Ridoux
Projet Lande

mars 2000

Systemes d'information logiques (SIL)

- système d'information :
 - naviguer (avec flexibilité)
 - interroger (avec interactivité)
 - organiser (automatiquement)
 - mettre à jour (avec cohérence)
- logique (arbitraire) :
 - diversité des domaines d'application
 - expressivité (mais décidabilité)

Naviguer avec flexibilité

- **naviguer** : aller de place en place en suivant des **liens**
- **place** : sélection d'un sous-ensemble d'objets
 - *système hiérarchique* : place = répertoire, lien = sous-répertoire
⇒ rigide, organisation souvent manuelle
 - *système booléen* : pas de navigation
⇒ manque d'interactivité
 - *SIL* : place \leftarrow formule logique, lien = formule logique
⇒ flexible, organisation automatique

Interroger avec interactivité

- **requête** : propriété des informations cherchées
- **réponse** : les informations satisfaisant la requête (rappel, précision)
 - *système hiérarchique* : requête = chemin d'accès ?
⇒ besoin d'érudition ou de chance, rappel faible [GMA 93]
 - *système booléen* : requête = proposition logique (mots-clés)
⇒ difficulté de contrôler le rappel et la précision [GMA 93]
 - *SIL* : requête \leftarrow formule, réponse = objets + formules (liens)
⇒ interactif : contrôle du rappel et de la précision

Mettre à jour avec cohérence

- déplacer un objet d'une place à une autre
- maintenir la cohérence entre contenu et place d'un objet
- *systemes hybrides* : hiérarchique et booléen
 - ⇒ certaines mises à jour sont impossibles (SFS) ou incohérentes (HAC)
- *SIL* :
 - analyse de concepts formels [Wille 82] : unification complète et cohérente des places et des requêtes → concepts
 - contexte formel : propriétés extrinsèques et intrinsèques

Travail en cours et à venir

- Analyse de concepts logique : généralisation de l'analyse de concepts à une logique arbitraire
- Shell conceptuel
 - prototype générique
 - interface du shell UNIX (cd, ls, ...)
- Enrichir le modèle : objets structurés, terminologie
- Implémenter au niveau système pour être visible depuis les applications (ex : éditeurs, compilateurs, etc.)
- Etude d'applications :
 - génie logiciel : gestion de versions, architecture logicielle
 - grand public : guides, catalogues, moteurs de recherche

Plan

- Analyse de Concepts Logique (ACL)
- Shell conceptuel : réalisation et expérimentation
- Conclusion et perspectives

Comparaison de 3 méthodes de recherche d'informations [GMA93]

	système hiérarchique	système booléen	analyse de concepts (AC)
navigation	●		●
interrogation		●	●
maintenance navigation	manuel	—	automatique
temps recherche	3.90	3.55	3.95
taux rappel	70.5%	80.9%	79.5%

système hiérarchique : taux de rappel faible

système booléen : absence de navigation

⇒ l'AC combine interrogation et navigation avec des performances semblables au système booléen.

Analyse de concepts (1/2)

- **contexte formel** : $(\mathcal{O}, \mathcal{A}, I)$ avec \mathcal{O} : **objets**, \mathcal{A} : **attributs**,
 $I \subseteq \mathcal{O} \times \mathcal{A}$: **table** de correspondance entre objets et attributs
ex : $\mathcal{O} \rightarrow$ **procédures**, $\mathcal{A} \rightarrow$ **variables globales**,

	a1	a2	a3	a4	a5	a6	a7	a8
o1	•	•						
o2			•	•	•			
o3			•	•		•	•	•
o4			•	•	•	•	•	•

- **concept** : (extension, intention) = (O, A)

tq $\sigma(O) = A$ et $\tau(A) = O$

$$\sigma(O) = \{a \in \mathcal{A} \mid \forall o \in O : (o, a) \in I\} \quad \forall O \subseteq \mathcal{O}$$

$$\tau(A) = \{o \in \mathcal{O} \mid \forall a \in A : (o, a) \in I\} \quad \forall A \subseteq \mathcal{A}$$

Analyse de concepts (2/2)

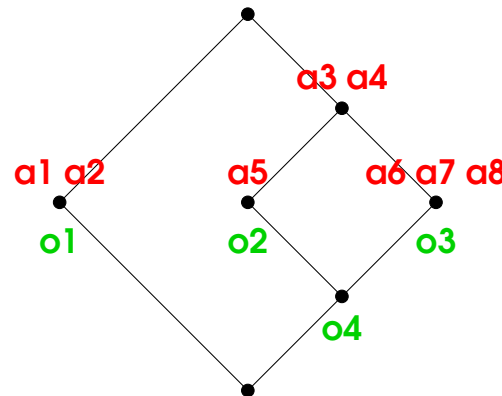
- ordre sur les concepts

$$(O_1, A_1) \leq (O_2, A_2) \iff O_1 \subseteq O_2 \iff A_1 \supseteq A_2$$

→ **treillis complet** sur l'ensemble des concepts

+ **étiquetage** pour aider l'interprétation

ex :



- le treillis permet de retrouver la table d'origine I
- le treillis et l'étiquetage peuvent être construits automatiquement

Généralisation logique de l'AC (1/2)

- dans AC : propriété = ensemble d'attributs
→ pouvoir d'expression faible
- types de propriétés différents suivant l'application

ex :

- catalogues, annuaires
- bibliothèques (ouvrages, logiciel)

⇒ remplacement des **ensembles d'attributs** par les **formules** d'une logique *presque* arbitraire $\langle \mathcal{L} ; \dot{\vee}, \dot{\wedge}, \dot{=} \rangle$ pour représenter les propriétés des objets.

↪ **Analyse de Concepts Logique (ACL)**

Généralisation logique de l'AC (2/2)

- **contexte formel** : $(\mathcal{O}, \mathcal{L}, i)$
 $i \in \mathcal{O} \rightarrow \mathcal{L}$: description des objets par une formule
- **concept** : $(extension, intension) = (O, f)$
 tq $\sigma(O) \doteq f$ et $\tau(f) = O$

$$\sigma(O) \doteq \bigvee_{o \in O} i(o) \quad \forall O \subseteq \mathcal{O}$$

$$\tau(f) = \{o \in \mathcal{O} \mid i(o) \models f\} \quad \forall f \in \mathcal{F}$$

- on retrouve les résultats classiques de l'AC
 - $\langle \mathcal{C} ; \leq^c \rangle$ forme un **treillis** complet
 - **étiquetage** du treillis par les objets et les formules

Logique contextualisée

- $f \dot{\models} g$: déduction **logique**
- $f \dot{\models}^K g$: déduction **contextualisée**
définition : $\tau(f) \subseteq \tau(g)$
ex : *document* $\dot{\models}^K$ *LaTeX*
- $f \dot{\models} g \implies f \dot{\models}^K g$: la déduction contextualisée *étend* la déduction logique

Propriété remarquable : $\langle \mathcal{L} / \dot{=}^{\kappa} ; \dot{\models}^K \rangle$ est isomorphe à $\langle \mathcal{C} ; \leq^c \rangle$
 par la fonction d'étiquetage

Retour à l'AC standard

$\mathcal{L} \rightsquigarrow$ logique à attributs

Soit \mathcal{A} un ensemble d'attributs.

logique générale	\mathcal{L}	$\dot{\vee}$	$\dot{\wedge}$	$\dot{\vDash}$
logique à attributs	$\mathcal{P}(\mathcal{A})$	\cap	\cup	\supseteq

Pour la logique à attributs, ACL est équivalente à AC.

Remarque : l'ACL a aussi été instanciée pour des logiques plus riches : des ensembles de termes du 1er ordre, une logique de description.

Réalisation d'un shell conceptuel

Quoi ?

- prototype de SIL **générique**
- en **ligne de commande**

Pourquoi ?

- expérimenter notre modèle de SIL
- concevoir et tester des algorithmes

Spécification informelle du shell conceptuel

On utilise les mêmes commandes que celles d'UNIX.

shell UNIX	shell conceptuel
fichier	objet
path/chemin	formule logique
répertoire	concept \leftarrow formule
racine	concept $\top^c \leftarrow$ formule $\dot{\top}$
répertoire de travail	concept de travail

Dans les commandes, les arguments sont des formules et désignent soit :

- un objet
- un ensemble d'objets

Opérations élémentaires

Logique : $\dot{\models}$ (déduction), $\dot{\wedge}$ (conjonction), $\dot{\Leftarrow}$ (complément relatif)

Contexte : i_e (description extrinsèque), c (contenu), α (analyseur)

Interrogation : **extension** d'une requête (τ)

- $\tau(q) = \{o \in \mathcal{O} \mid i(o) \dot{\models} q\}$

Navigation : **objet local** (t) et **liens de navigation** (Inc)

- $t(q) \in \mathcal{O}$, tq concept de $t(q) =$ concept de q
- $Inc(q) = [\{x \in X \mid \emptyset \subsetneq \tau(q \dot{\wedge} x) \subsetneq \tau(q)\}]$
 \Rightarrow complétude navigationnelle

Spécification formelle du shell conceptuel

commande	sémantique
<code>pwd</code>	$out(wq())$
<code>cd ..</code>	$pop()$
<code>cd to</code>	$push(to)$
<code>ls f</code>	$out(\mathbf{t}(f)); out(\mathbf{Inc}(f))$
<code>ls -r f</code>	$out(\mathbf{\tau}(f))$
<code>mkfile f c</code>	$o := new_o(); c(o) := c; i_e(o) := extr(f)$
<code>rm f</code>	$del_o(\mathbf{t}(f))$
<code>rm -r f</code>	forall $o \in \mathbf{\tau}(f)$ do $del_o(o)$
<code>mv from to</code>	$o := \mathbf{t}(from); i_e(o) := (i_e(o) \leftarrow extr(from)) \dot{\wedge} extr(to)$
<code>mv -r from to</code>	forall $o \in \mathbf{\tau}(from)$ do $i_e(o) := (i_e(o) \leftarrow extr(from)) \dot{\wedge} extr(to)$

Expérimentation

Choix du contexte formel :

- les objets : des plats vietnamiens
- la logique : la logique à attributs
- description des objets :
 - ingrédients du plat
 - menus composés avec le plat
 - nom(s) vietnamien(s) du plat

Expérimentation : interrogation et navigation

- interrogation

```
% ls -r /glutamate  $\rightsquigarrow$  40 réponses
```

```
% ls -r /glutamate/epinard  $\rightsquigarrow$  0 réponse
```

- navigation

```
% cd glutamate/legume  $\rightsquigarrow$  17 objets
```

```
% ls .  $\rightsquigarrow$  liens de navigation
```

```
% cd chou_vert  $\rightsquigarrow$  objet local  $o$  (Nouilles garnies)
```

```
 $i(o) =$ 
```

```
/Pho_xao/plat_complet/pate_riz/chou_vert/boeuf/crevette/glutamate
```

Expérimentation : mises-à-jour

- constitution d'un menu
`% mv . menu_perso`
- consultation d'un menu
`% ls -r /menu_perso`
- création de nouvelles recettes à partir d'anciennes
`% cp -r /volaille /boeuf`

Conclusion

Avantages de notre SIL :

- concepts ← formules : *vraies requêtes* et *vrais places*
 - ⇒ combinaison étroite de d'interrogation et de navigation
 - ⇒ mises-à-jour cohérentes
- consultations et mises-à-jour avec le même langage de commandes
- accès par des *propriétés* plutôt que par des *adresses*
- traitement par ensemble avec l'option -r
- *précision* des descriptions et *concision* des requêtes
- organisation automatique (stockage, navigation)

Perspectives théoriques

- relations entre les objets

ex : relations entre plats et ingrédients
plats et menus

- structurer les objets
- rechercher des parties de documents (Web, XML)

- points de vue

ex : `ls -v ingredients`, `ls -v legumes`

- filtrer la navigation
- faire abstraction de certaines propriétés

Perspectives applicatives

- instantiation du SIL
 - ex : isomorphismes de types [Di Cosmo] pour la recherche de composants logiciels (sujet de DEA en cours)
- implémentation au niveau système de fichiers
 - visibilité depuis toutes les applications
 - remarque : prototype réalisé par un projet de maîtrise
- domaines d'applications
 - génie logiciel (ex : gestion de versions et de configurations)
 - outils informatiques (ex : menus, moteurs de recherche)
 - grand public (ex : catalogue, bibliothèque)

Génie logiciel

- Réutilisation (navigation,interrogation)
 - isomorphismes de types et autres abstractions
- Gestion de configuration (MàJ,navigation,interrogation)
 - descripteurs de configuration (taxinomie)
- Gestion de version
- Architecture logicielle (Màj,navigation,interrogation,vues,relations)
 - architecture “dynamique” : processus, connecteurs, déploiement
 - architecture “statique” : développement, maintenance, documentation, traçabilité

Expérimentation : entrée des données

```
[2] mkdir abalone
[3] mkdir agar_agar
[4] mkdir alcool_riz
[...]
[92] cd entree
[93] mkfile vermicelle/germe_soja/champignon/crevette
    /glutamate/galette_riz/Nem/Cha_gio
    "Paté impérial à la vietnamienne"
[94] mkfile farine_riz/ciboule/safran/porc/champignon
    /crevette/germe_soja/Banh_xeo_Hue
    "Crêpe de Hué farcie"
```

Expérimentation : interrogation

```
[350] pwd
```

```
/
```

```
[351] ls -r glutamate
```

```
...
```

```
40 object(s)
```

```
[352] ls -r glutamate/epinard
```

```
0 object(s)
```

```
[...]
```

```
[356] cd glutamate/legume
```

```
[357] ls -r
```

```
...
```

```
17 object(s)
```

Expérimentation : navigation

```
[358] ls .
```

```
1 curry          2 concombre      5 fruit_mer
1 safran         2 celeri        5 menu
1 chou_sale     2 poisson       6 ciboule
1 carotte       2 champignon    6 coriandre
1 asperge       3 menthe        8 nuoc_mam
1 chou_blanc    3 soja          11 plat
1 chou_vert     3 pousse_bambou 12 cereale
1 petit_pois    3 tomate        14 viande
```

```
[359] cd chou_vert
```

```
[360] edit .
```

```
48 Pho_xao/boeuf/chou_vert/crevette/glutamate/pate_riz/plat_complet
```

```
Nouilles garnies à la Hanoienne
```

```
[368] cd -glutamate
```