Introduction à la Logique de Description (résumé)

Nhan LE THANH, Labo I3S
Cours B7, Master 2 PMLT
Ecole doctorale STIC, UNSA

Formalisme de base de Logiques de Description

Historique et caractéristiques

- Une famille de formalismes de représentation de connaissances basés sur la logique
  - Descendants de réseaux sémantique and KL-ONE
  - Description du domaine d'application en terme de concepts (classes), rôles (propriétés, relations) and individus (objets)
- Caractéristiques :
  - Sémantique formelle (modèle théorique)
    - fragments décidables de la LPO (sous ensemble de classe C_2)
    - Lien étroit aux Logiques Modales Propositionnelles & Logiques Dynamiques
  - Production des services d'inférence
    - Procédure de décision pour des problèmes clés (satisfiabilité, subsumption, etc)
    - Systèmes implantés hautement optimisés

Historique et caractéristiques : évolution

Phase 1 (78-92):
- systèms Incomplets (Classic, Loom,)
- Basé sur les algorithmes structuraux

Phase 2 (93-98):
- Développement de algorithmes tableau et étude de la complexité
- Systèmes basés sur les algorithme Tableau avec la complexité Pspace (Kris, Crack)
- Développement des techniques d'optimisation

Phase 3 (98-03):
- Développement des algorithmes tableau pour les LDs très expressifs
- Développement des systèmes à tableau hautement optimisés pour une complexité ExpTime logics (e.g., FaCT, DLP, Racer)
- Preuve de relations avec la logique modale et les fragments décidables de la LPO

Phase 4 (actuel) :
- Applications Mainstream et tools
  - Base de données
    - Consistance de schémas conceptuels (EER, UML etc.)
    - Intégration de Schémas conceptuels
    - Requêtes (Query) de subsumption (sur un schéma conceptuel)
  - Ontologies, e-Science et Sémantique Web/Grid
    - Ingénierie d'Ontologie (schema design, maintenance, intégration)
    - Raisonnement avec annotation basées sur ontologies (données)
- Maturité des implémentations
  - Projets de recherche
    - FaCT, FaCT++, Racer, Pellet, …
  - Systèmes commerciaux
    - Cerebra par Network Inference
    - RacerPro par ?

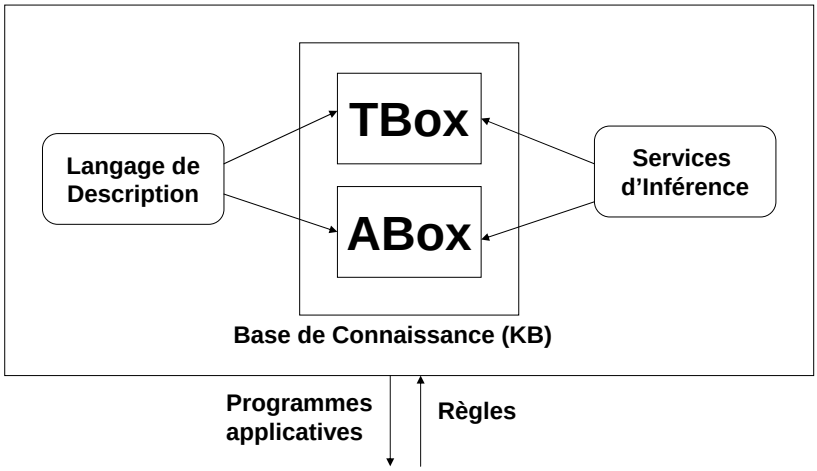
Eléments basiques

- Les (noms de) Concept sont équivalents aux prédicats unaires
  - En général, les concepts sont équivalents aux formules avec une variable libre
- Les (noms de) Rôle sont équivalents aux prédicats binaires
  - En général, les rôles sont équivalents aux formules avec deux variables libres
- Les (noms d') Individus sont équivalents aux constantes
- Les Constructeurs sont restreints pour que :
  - Le langage reste décidable et, si possible, avec une complexité faible
  - L'utilisation des variables n'est pas explicite
    - Forme de restriction sur co-domaine comme $\exists r.C$ et $\forall r.C$
  - Les caractéristiques telles que calcul peuvent être exprimées succinctement

Grands principes

- Les éléments du monde réel sont représentés par des *concepts*, des *rôles* et des *individus*.
- Adéquation syntaxe – sémantique
  - les concepts et rôles possèdent une *description structurée* à laquelle est associée une *sémantique*
  - les manipulations opérées sur les concepts et les rôles sont réalisées en accord avec la sémantique
- Deux types de connaissances sont prises en compte :
  - les concepts avec leurs composants, et
  - les *faits* ou *assertions*, où interviennent les concepts et les instances de concepts
- La relation de subsomption organise concepts et rôles en hiérarchies
- Raisonnement : *classification* et *instanciation*

Structure générale de système LD



Base de connaissances

- Une base de connaissances $\mathcal{K}$ est une paire $\langle \mathcal{T}, \mathcal{A} \rangle$

  où
  - $\mathcal{T}$est un ensemble d'axiomes "terminologiques" (TBox)
  - $\mathcal{A}$est un ensemble d'axiomes "assertionnels" (ABox)
- Les axiomes de TBox sont sous forme :
  A, $C \sqsubseteq D$, $C \equiv D$, $R \sqsubseteq S$, $R \equiv S$ and $R^+ \sqsubseteq R$

  où A est un concept atomique, C, D sont des concepts, R, S sont des rôles, et R^+ l'ensemble de rôles transitifs
- Les axiomes de Abox sont sous forme :
  $x:D$, $\langle x,y \rangle:R$

  où x,y sont des(noms d') individus, D est un concept et R est un rôle

Exemple : syntaxe d'une LD

La syntaxe d'une logique de description			
$C \sqsubseteq A$	Subsomption de concepts	$\forall r.C$	Restriction universelle
$\bot$	Concept absurde	$\exists r.C$	Restriction existentielle
$\top$	Concept universel	$(\geq n\ r)$	Restriction supérieure de cardinalité
$C \sqcap D$	Conjonction de concepts	$(\leq n\ r)$	Restriction inférieure de cardinalité
$C \sqcup D$	Disjonction de concepts	$r \sqsubseteq a$	Subsomption de rôles
$\neg C$	Négation de concept	$r \wedge s$	Composition de rôles
- C et D sont des expressions de concepts, r et s sont des expressions de rôles - A est un concept primitif et a est un rôle primitif - n est un entier non nul			

Exemple : Base de connaissance

- Exemple : Base de connaissances
  - Un **Homme** est une **Personne**.
  - Une **Femme** est une **Personne**.
  - Aucune **Femme** n'est un **Homme** et vice-versa.
  - Une **Equipe** est (définie comme) un **Ensemble** ayant au moins 2 **membres** qui sont tous des **Personnes**.
  - Une **Petite-équipe** est (définie comme) une **Equipe** ayant au plus 5 **membres**.
  - Une **Equipe-moderne** est (définie comme) une **Equipe** ayant au moins 4 **membres**, ayant au moins 1 **chef**, et dont tous les **chefs** sont des **Femmes**

- La base de connaissances
  - Concepts primitifs : $Personne,\ Ensemble$
  - Rôles primitifs : $membre$
  - $Femme \sqsubseteq Personne$
  - $Homme \sqsubseteq Personne \sqcap \neg Femme$
  - $chef \sqsubseteq membre$
  - $Equipe \sqsubseteq Ensemble \sqcap (\forall membre.Personne) \sqcap (\geq 2\ membre)$
  - $Petite\text{-}équipe = Equipe \sqcap (\leq 5\ membre)$
  - $Equipe\text{-}moderne = Equipe \sqcap (\geq 4\ membre) \sqcap \exists chef \sqcap (\forall chef.Femme)$

Famille de langages LD : langage $\mathcal{FLE}$

Un premier langage LD simple ($\mathcal{FLE}$) avec 3 constructeurs

Constructeur	Syntaxe	Exemple
Concept universel	$\top$	
Conjonction	$A \sqcap B$	Personne $\sqcap$ Jeune
Quantification Universelle	$\forall R.C$	$\forall$enfant.Masculin
Quantification Existentielle	$\exists R.\top$	$\exists$enfant.$\top$

Famille des langages LD: les $\mathcal{AL}$

$\mathcal{AL} = \{\top, \bot, \neg A, C \sqcap D, \forall r.C, \exists r\}$

- $\mathcal{ALC} = \mathcal{AL} \cup \{\neg C\}$ (négation de concepts définis) (équivalent à la classe de logique modal $K_{(m)}$)
- $\mathcal{ALU} = \mathcal{AL} \cup \{C \sqcup D\}$ (disjonction de concepts)
- $\mathcal{ALE} = \mathcal{AL} \cup \{\exists r.C\}$ (quantification existentielle typée)
- $\mathcal{ALN} = \mathcal{AL} \cup \{\geq n\ r, \leq n\ r\}$ (cardinalité de rôles - remarque $\exists r \equiv (\geq 1\ r)$)
- $\mathcal{ALR} = \mathcal{AL} \cup \{r1 \sqcup r2\}$ (conjonction de rôles)

Famille des langages LD: $\mathcal{SHOIN}$ et $\mathcal{SHIQ}$ (OWL)

Sigle	Nom	Syntaxe	Exemple
$\mathcal{S}$ ($\mathcal{ALC}$ et R+)	Conjonction	$C \sqcap D$	personne ⊓ jeune
	Disjonction	$C \sqcup D$	vieux ⊔ jeune
	Négation	$\neg C$	¬ (personne ⊓ jeune)
	Q.U.	$\forall R.C$	∀enfant.male
	Q.E.	$\exists R.C$	∃ enfant.male
	Rôle transitif (R+)	R.R	père.père
$\mathcal{H}$	Rôle hiérarchique	$R \sqsubseteq S$	mère ⊑ parent
$\mathcal{O}$	Concepts nominaux	{x}	{Terre, Mars, Venus}
$\mathcal{I}$	Inverse de rôle	R^-	enfant⇔parent
$\mathcal{N}$	Restriction de nombre	$\geq nR$	≥2enfant
$\mathcal{Q}$	RN qualifiée	$\geq nR.C$	≥2enfant.male

Complexité des langages (sans TBox)

Expressivité		⊨ C ⊑ D	⊨C(a)
$\mathcal{FLE}$	C ⊓D $\forall R.C$ $\exists R$	P	P
$\mathcal{AL}$	$\neg A$	P	P
$\mathcal{ALE}$	$\exists R.C$	NP	PSPACE
$\mathcal{ALC}$ (~ modal $K_{(m)}$)	$\neg C$	PSPACE	
$\mathcal{ALCO}$	{a1, ...}	PSPACE	
$\mathcal{SHOIN(D)}$		EXPTIME	
$\mathcal{SHIQ}$		EXPTIME	
NLOGSPACE ⊑ P ⊑ NP ⊑ PSPACE = NPSPACE ⊑ EXPTIME ⊑ NEXPTIME ⊑ EXPSPACE			

Complexité des langages (selon TBox)

Expressivité	⊨ C ⊑ D	⊨C(a)
$\mathcal{FLE}$	P	P
$\mathcal{AL}$	EXPTIME	EXPTIME
$\mathcal{ALE}$	PSPACE	PSPACE
$\mathcal{ALC}$ *(avec domaine concret et TBox cyclique)*	NEXPTIME	
$\mathcal{ALC}$ *(avec R⊓S, R⊔S, ¬R et TBox vide)*	NEXPTIME	
$\mathcal{SHOIN(D)}$	NEXPTIME	
$\mathcal{SHIQ}$	NEXPTIME	
NLOGSPACE ⊑ P ⊑ NP ⊑ PSPACE = NPSPACE ⊑ EXPTIME ⊑ NEXPTIME ⊑ EXPSPACE		

Sémantique de LD

- La sémantique de LD est définie par les interprétations
- Une interprétation $\mathcal{I}$ est définie comme $\mathcal{I}$= $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, où
  - $\Delta^{\mathcal{I}}$ est le domaine (un ensemble vide)
  - $\cdot^{\mathcal{I}}$ est une fonction d'interprétation qui fait correspondre:
    - Concept (classe) A → Sous ensemble $A^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$
    - Rôle (propriété) R → relation binaire $R^{\mathcal{I}}$ sur $\Delta^{\mathcal{I}}$**x** $\Delta^{\mathcal{I}}$
    - Individus i → $i^{\mathcal{I}}$ élément de $\Delta^{\mathcal{I}}$

Sémantique des expression LD

- La fonction d'interprétation $\cdot^{\mathcal{I}}$ sur les expressions de concept (ou de rôle) de la manière suivante :

$$
\begin{array}{lcl}
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
\{x\}^{\mathcal{I}} & = & \{x^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} & = & \{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} & = & \{x \mid \forall y.(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\} \\
(\leqslant nR)^{\mathcal{I}} & = & \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \leqslant n\} \\
(\geqslant nR)^{\mathcal{I}} & = & \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \geqslant n\} \\
(R^{-})^{\mathcal{I}} & = & \{(x, y) \mid (y, x) \in R^{\mathcal{I}}\}
\end{array}
$$

Sémantique de la base de connaissances

- On dit qu'une interprétation $\mathcal{I}$ satisfait (modèles) un axiome TBox C (noté $\mathcal{I} \models$ C) de la manière suivante :
  $\mathcal{I} \models C \sqsubseteq D$ ssi $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ $\quad$ $\mathcal{I} \models C \equiv D$ ssi $C^{\mathcal{I}} = D^{\mathcal{I}}$
  $\mathcal{I} \models R \sqsubseteq S$ ssi $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ $\quad$ $\mathcal{I} \models R \equiv S$ ssi $R^{\mathcal{I}} = S^{\mathcal{I}}$
  $\mathcal{I} \models R^{+} \sqsubseteq R$ ssi $(R^{\mathcal{I}})^{+} \subseteq R^{\mathcal{I}}$ $\quad$ $\mathcal{I} \models R^{+} \sqsubseteq R$ ssi $(R^{\mathcal{I}})^{+} \subseteq R^{\mathcal{I}}$
- Une interprétation $\mathcal{I}$ satisfait une Tbox $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$) ssi $\mathcal{I}$ satisfait tout axiome C dans $\mathcal{T}$
- Une interprétation $\mathcal{I}$ satisfait (produit des modèles) un axiome Abox a ($\mathcal{I} \models$ a) de la manière suivante :
  $\mathcal{I} \models x{:}D$ ssi $x^{\mathcal{I}} \in D^{\mathcal{I}}$ $\quad$ $\mathcal{I} \models \langle x,y\rangle{:}R$ ssi$(x^{\mathcal{I}},y^{\mathcal{I}}) \in R^{\mathcal{I}}$
- Une interprétation $\mathcal{I}$ satisfait une Abox $\mathcal{A}$ ($\mathcal{I} \models \mathcal{A}$) ssi $\mathcal{I}$ satisfait tout axiome a in $\mathcal{A}$
- Une interprétation $\mathcal{I}$ satisfait une BC $\mathcal{K}$ ($\mathcal{I} \models \mathcal{K}$) ssi $\mathcal{I}$ satisfait à la fois $\mathcal{T}$et $\mathcal{A}$

II.2- Système formel LD : ABox – Interprétation sémantique

- Exemple :
  - Soit la base terminologique
  - $\sum$ = {enfant(pierre, marie), (∀enfant.(¬MUSICIEN))(Pierre) , FEMME (∃ enfant))(Marie)}
  - La base $\sum$ est satisfiable et l'interprétation $\mathcal{I}$ ci-après est un modèle pour $\sum$
    - $\triangle^{\mathcal{I}} = \{Pierre, Marie\}$
    - $Pierre^{\mathcal{I}} = Pierre$
    - $Marie^{\mathcal{I}} = Marie$
    - $Enfant^{\mathcal{I}} = \{(Pierre, Marie)\}$
    - $FEMME^{\mathcal{I}} = \{Marie\}$
    - $MUSICIEN^{\mathcal{I}} = \varnothing$

II.2- Système formel LD : Exemple

- *Niveau Terminologique*
- Concepts primitifs : Cours, Professeur, Etudiants, DEA et Doctorat
- Rôles primitifs : enseigne et diplôme.
- *Contraintes*
  - Tout enseignant d'un cours est soit professeur, soit titulaire d'un diplôme de DEA.

  (∃ enseigne.Cours) ⊑ (Etudiant ⊓ (∃ diplôme.DEA)) ⊔ Professeur
  - Tout professeur est titulaire d'un diplôme de Doctorat.

  Professeur ⊑ (∃ diplôme.Doctorat)
  - Pour obtenir un diplôme de Doctorat, il faut posséder un DEA.

  (∃ diplôme.Doctorat) ⊑ (∃ diplôme.DEA)
  - Les DEA et les Doctorats sont des diplômes distincts.

  Doctorat ⊓ DEA ⊑ ⊥
- *Niveau assertionnel*
  - Jean enseigne un "cours d'IA".

  enseigne(Jean,Cours-IA)
  - Jean a au plus un diplôme.

  (≤ 1diplôme) (Jean)
  - Un "cours d'IA" est un cours.

  Cours(Cours-IA)

Raisonnement dans les logiques de Description

Sémantique de la BC (rappel)

- Une interprétation $\mathcal{I}$ satisfait (est modèle) un axiome C ($\mathcal{I} \models$ C):
  - $\mathcal{I} \models C \sqsubseteq D$ ssi $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ $\mathcal{I} \models C \equiv D$ ssi $C^{\mathcal{I}} = D^{\mathcal{I}}$
  - $\mathcal{I} \models R \sqsubseteq S$ ssi $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ $\mathcal{I} \models R \equiv S$ ssi $R^{\mathcal{I}} = S^{\mathcal{I}}$
  - $\mathcal{I} \models x \in D$ ssi $x^{\mathcal{I}} \in D^{\mathcal{I}}$
  - $\mathcal{I} \models \langle x,y \rangle \in R$ ssi $(x^{\mathcal{I}},y^{\mathcal{I}}) \in R^{\mathcal{I}}$
  - $\mathcal{I} \models R^+ \sqsubseteq R$ ssi $(R^{\mathcal{I}})^+ \subseteq R^{\mathcal{I}}$
  - $\mathcal{I} \models R^+ \sqsubseteq R$ ssi $(R^{\mathcal{I}})^+ \subseteq R^{\mathcal{I}}$
- $\mathcal{I}$ satisfait une TBox $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$) ssi $\mathcal{I}$ satisfait tout axiome C de $\mathcal{T}$
- $\mathcal{I}$ satisfies une ABox $\mathcal{A}$ ($\mathcal{I} \models \mathcal{A}$) ssi $\mathcal{I}$ satisfait tout axiome a de $\mathcal{A}$
- $\mathcal{I}$ satisfait une KB $\mathcal{K}$ ($\mathcal{I} \models \mathcal{K}$) ssi $\mathcal{I}$ satisfait à la fois $\mathcal{T}$ et $\mathcal{A}$

Les principaux tests d'inférence

- La connaissance est-elle correcte (intuitions capturées) ?
  - Est-ce que C est subsompmé par D selon $\mathcal{T}$? ($C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ pour *tout* modèle $\mathcal{I}$ de $\mathcal{T}$)
- La connaissance est-elle minimalement redondante (pas de synonymes inattendus)
  - Est-ce que C est équivalent à D selon $\mathcal{T}$? ($C^{\mathcal{I}} = D^{\mathcal{I}}$ dans *tout* modèle $\mathcal{I}$ de $\mathcal{T}$)
- La connaissance a-elle plein sens (meaningful) (classes doivent être non vide)
  - Est-ce que C est satisfiable selon $\mathcal{T}$? ($C^{\mathcal{I}} \neq \emptyset$ dans *un* modèle $\mathcal{I}$ de $\mathcal{T}$)
- La connaissance est-elle consultable (Querying knowledge)
  - Est-ce que x est une instance de C selon $\mathcal{T}$? ($x^{\mathcal{I}} \in C^{\mathcal{I}}$ dans tout modèle $\mathcal{I}$ de $\mathcal{T}$)
  - Est-ce que $\langle x,y \rangle$ est une instance de R selon $\mathcal{T}$? ($(x^{\mathcal{I}},y^{\mathcal{I}}) \in R^{\mathcal{I}}$ dans *tout* modèle $\mathcal{I}$ de $\mathcal{T}$)
- Les connaissances sont-elles incompatibles :
  - Deux concepts C et D sont-ils incompatibles selon $\mathcal{T}$? ($C^{\mathcal{I}} \sqcap D^{\mathcal{I}} = \emptyset$ dans tout modèle $\mathcal{I}$ de $\mathcal{T}$)

Réduction de tests

- On peut réduire les tests de base :
  - Au test de satisfaisabilité :
    - C est subsumé par D si seulement si $(C \sqcap \neg D)$ n'est pas satisfaisable
    - C et D sont équivalents si seulement si $(C \sqcap \neg D) \sqcup (D \sqcap \neg C)$ n'est pas satisfaisable
    - C et D sont incompatibles si seulement si $(C \sqcap D)$ n'est pas satisfaisable
  - Ou au test de subsomption :
    - C n'est pas satisfaisable si seulement si C est subsumé par $\bot$
    - C et D sont équivalents si seulement si C est subsumé par D et D est subsumé par C
    - C et D sont incompatibles si seulement si $(C \sqcap D)$ est subsumé par $\bot$

Propriétés de transformation

- Les constructeurs ⊓ et ⊔ obéissent aux règles suivantes
  - Idempotence: C ⊓ C ≡ C et C ⊔ C ≡ C
  - Commutativité: C ⊓ D = D ⊓ C et C ⊔ D = D ⊔ C
  - Associativité : C ⊓ (D ⊓ E) = (C ⊓ D) ⊓ E et C ⊔ (D ⊔ E) = (C ⊔ D) ⊔ E
  - Si C ⊑ D et C ⊑ E alors C ⊑ D ⊓ C
  - Si C ⊑ D et E ⊑ D alors C ⊔ E ⊑ D
  - Si C ⊑ D alors C ⊓ X ⊑ D pour tout description X
  - Si C ⊑ D alors C ⊑ D ⊔ X pour tout description X

III.1- Raisonnement LD : deux types de raisonnement

- Les *algorithmes de type normalisation–comparaison* (algorithmes NC) :
  - un processus de normalisation produit les *formes normales* des concepts définis qui sont ensuite
  - effectivement comparées à l'aide de règles de comparaison
- La *méthode des tableaux sémantiques* : la question *est-ce que D subsume C* (C ⊑ D) est transformée en *est-ce que (C* ⊓ ¬D) *est non satisfiable*.
- Note : le langage de description des concepts doit être muni de la négation des concepts définis

Algorithme de tableau basique(sans TBox)

- Utilisé pour le test de satisfiabilité (consistance) d'une expression de concepts C
- Essai de construire un arbre de modèles hypothétiques dont la racine est ¬C (sous forme normale : négation devant concepts atomiques)
- Décomposer syntaxiquement et successivement de C
  - En appliquant des règles d'expansion de tableau
  - En évaluant des contraintes sur les éléments du modèle
- Les règles d'expansion de tableau correspondent aux constructeurs dans le langage LD utilisés ($\sqcap$, $\sqcup$, $\forall$, $\exists$, $\geq$, $\leq$)
  - Certaines règles sont non déterministes (par exemple : $\sqcup$, 6)
  - En pratique, cela explose l'espace de recherche
- Arrêter quand aucune règle soit applicable ou motifs contradictoires (clash) par exemple (A(x), ¬A(x))
- Le contrôle de cycle (blocking) peut-être nécessaire pour la terminaison
- C sera satisfiable ssi les règles peuvent-être appliquées telles que l'arbre construit ne contenant que des feuilles avec motifs contradictoires (clash)

Algorithme de tableau avec TBox

- Satisfiabilité selon une TBox $\mathcal{T}$
  - Pour chaque axiome C $\sqsubseteq$ D $\in \mathcal{T}$, ajouter ¬C $\sqcup$ D au système de contraintes de départ
- Les langages LD plus expressif
  - Les techniques de base peuvent être extensives pour
    - Inclusion de Rôle (hiérarchie de rôles)
    - Restriction de Nombre
    - Inverse de rôles
    - domaines concrets/types de données
    - ABox, etc.
  - Introduction des règles d'expansion avec la stratégie de contrôle de cycle plus sophistiquées (blocking strategy)
  - Forêt à la place d'arbre de tableaux (pour les ABox)
    - Les noeuds racines correspondent aux individus dans ABox

Propriétés algorithmique

- *Procédure de décision :* un algorithme est appelé une procédure de décision si et seulement si il vérifie 3 propriétés suivantes :
  - ***Arrêt*** : l'algorithme doit donner le résultat à un temps fini
  - ***Correction*** : les inférences produites sont en accord avec la sémantique associée, autrement dit, ce qui est vrai sur le plan syntaxique l'est sur le plan sémantique.
  - ***Complétude*** : toutes les formules valides — vraies sur le plan sémantique — peuvent être démontrées sur le plan syntaxique.
- *Complexité d'un algorithme* : Un algorithme doit appartient à une classe de complexité qui qualifie son « coût » en temps et en espace de mémoire
  - NLOGSPACE $\sqsubseteq$ P $\sqsubseteq$ NP $\sqsubseteq$ PSPACE $\sqsubseteq$ EXPTIME $\sqsubseteq$ NEXPTIME $\sqsubseteq$ EXPSPACE
  - P $\neq$ PSPACE, PSPACE = NPSPACE, PSPACE $\neq$ EXPSPACE

Propriétés des algorithmes de tableau

- Théorème :
  - Les algorithmes de tableaux sont des procédures de décision pour la satisfiabilité de concepts
  - i.e., l'algorithme retourne "SAT" ssi le concept est satisfiable
  - Preuve (schématique)
    - Terminaison
      - La largeur (nombre de règles applicables par nœud) et la profondeur (avec la stratégie de blocking) de l'arbre de tableaux sont bornées
    - Correct
      - A partir d'un arbre de tableaux d'une formule satisfiable on peut contruire un un modèle pour cette formule
    - Complet
      - A partir d'un modèle d'une formule satisfiable, on peut diriger le processus d'application de règles d'expansion tel que arbre de tableaux obtenu contenant un tableau correspondant à ce modèle

Optimisation d'implantation

- Une implémentation naïve peut conduire à une situation non-terminaison
  - Un arbre avec 10 GCIs (Général Concept Inclusion) × 10 nœuds racines (assertions) peut conduire à $\rightarrow 2^{100}$ expansions possibles
- Les derniers systèmes incluent plusieurs optimisations
- classification optimisée (calcul d'un ordre partiel entre concepts)
  - Règles traversantes accélérées (explorer d'information de tests précédents)
  - Utiliser l'information structurale pour définir l'ordre de classification
- Tests satisfiabilité/subsumption optimisés
  - Normalisation et simplification de concepts
  - Absorption (simplification) des axiomes
  - Dépendance de retours (backtracking) directs
  - Préservation du résultat de satisfiabilité et modèles partiels
  - Ordonner heuristiquement des expansions propositionnelles and modales

Ontologies et Logiques de description

Ontologie et classification

- Origine de l'Ontologie :
  - Une branche de philosophie qui cherche un accord avec la nature et l'organisation de la réalité
  - Science de l'Être (Aristote, Métaphysiques, IV, 1)
    - Recherche de la réponse aux questions:
      - *Qu'est-ce que c'est les caractéristiques de l'Être?*
      - *Eventuellement, Qu'est-ce que c'est l'Être ?*
  - Comment les «choses » sont-elles classifiées ?
    - La classification a été étudiée depuis très long temps dans l'histoire

Ontologie en Informatique

- Une ontologie est un produit d'ingénierie qui consiste en :
  - un vocabulaire utilisé pour décrire (une vue particulière de) un certain domaine
  - une spécification explicite des sens attendus du vocabulaire.
    - Presque toujours inclure comment les concepts sont-ils classifiés
  - Capture des contraintes, des connaissances additionnelles sur le domaine
- Idéalement une ontologie permet de :
  - Capturer des compréhensions partagées d'un domaine
  - Produire un modèle formel du domaine manipulable par ordinateur

Où utiliserons les ontologies

- e-Science, par exemple, Bioinformatiques
  - L'Ontologie de Gènes (G.O)
  - L'Ontologie de Protéines (MGED)
- Médecine
  - Terminologies (ontologie UMLS)
- Databases
  - Intégration
  - Requêtes d'interrogation
- Interfaces d'utilisateur
- Linguistiques
- Le Sémantique Web

Ontologie et Logiques de description

- OWL est un standard de langage d'ontologie de W3C basé sur les logiques de description
  - Les axiomes et les constructeurs d'OWL sont restreints pour que le raisonnement soit décidable
- La Sémantique Web s'organise dans une architecture en couche
  - XML produit la couche de transport syntaxique
  - RDF(S) produit un language relationnel basique et des primitives ontologiques simples
  - OWL produit un langage d'ontologie puissant mais restant décidable
  - Autres couches (par exemple SWRL – Semantic Web Rule Language) seront des extensions de OWL
    - La plupart seront indécidables
- Besoin des "expériences d'implémentation"

Concepts (classes) et constructeurs

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

- C est un concept (classe); P est un rôle (propriété); x est un nom d'individu
- Types de données XMLS sont des classes dans ∀P.C and ∃P.C
  - Forme de restriction de DL avec domaines concrets

Exemple dans la syntaxe RDFS

Exemple : Personne ⊓ ∀aEnfant.(Docteur ⊔∃aEnfant.Docteur):

```
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Personne"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#aEnfant"/>
        <owl:toClass>
          <owl:unionOf rdf:parseType=" collection">
            <owl:Class rdf:about="#Docteur"/>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#aEnfant"/>
              <owl:hasClass rdf:resource="#Docteur"/>
            </owl:Restriction>
          </owl:unionOf>
        </owl:toClass>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Ontologie et Base de connaissances LD

- Une Ontologie OWL est équivalente à une Base de Connaissances de LD (BCLD)
- Une ontologie OWL consiste en un ensemble d'axiomes et faits
  - *Note: une ontologie comprend usuellement seulement des axiomes dans la TBox (schéma)---OWL est donc non-standard pour cette catégorie*
- Rappel : une BCLD $\mathcal{K}$ est la paire $\langle \mathcal{T}, \mathcal{A} \rangle$ où
  - $\mathcal{T}$est l'ensemble d'axiomes "terminologiques" (la TBox)
  - $\mathcal{A}$est l'ensemble d'axiomes "assertionnels" (l'ABox)

Ontologie / Axiomes TBox

OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor

- Une équivalence à la logique PO/Modale
  - p.e. DL: $\mathbf{C \sqsubseteq D}$ LPO: $\forall \mathbf{x}.\mathbf{C}(\mathbf{x}) \rightarrow \mathbf{D}(\mathbf{x})$ LM: $\mathbf{C} \rightarrow \mathbf{D}$
- Deux catégories différentes des axiomes TBox :
  - "Définitions" C $\sqsubseteq$ D ou C $\equiv$ D où C est un nom de concept
  - Axiomes d'Inclusion générale de concepts (GCIs) où C peut-être complexe

Faits d'Ontologie / Axiomes ABox

OWL Syntax	DL Syntax	Example
type	$a : C$	John : Happy-Father
property	$\langle a, b \rangle : R$	$\langle$John, Mary$\rangle$: has-child

- *Note: dans une utilisation nominale (par exemple, dans SHOIN), on peut réduire les axiomes d'ABox aux axiomes d'inclusion de concept*
  - a : C est équivalent à {a} $\sqsubseteq$ C
  - (a,b) : R est équivalent to {a} $\sqsubseteq \exists$R.{b}

Conclusion et perspectives

Complexité et Expressivité

- Voie de consensus
  - Plusieurs LD :
    - Plusieurs constructeurs
    - Chaque ensemble de constructeurs = une LD particulière
    - niveaux d'expressivité différents (description plus ou moins fine du domaine)
  - Compromis :
    - Plus grande est l'expressivité
    - plus complexes sont les raisonnements (voire indécidables)

Large ouverture de LD vers les autres disciplines

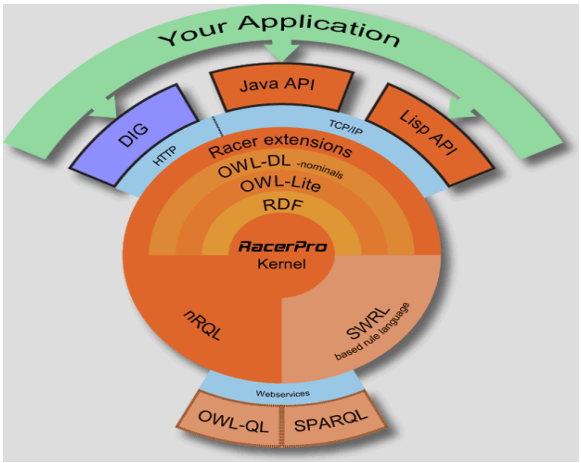
- Logiques de descriptions et logique modale.
- Logiques de descriptions et systèmes de représentation des connaissances par objets (RCO).
- Logiques de descriptions et bases de données.
- Logiques de descriptions et traitement du langage naturel
- Logique de description et calcul distribué

Implémentation de Systèmes

- Les précurseurs DL : KL-ONE (77-79), CLASSIC et LOOM (89-92)
- Les références actuelles de moteurs DL :
  - FaCT++ (00-04) : Fast Classification of Terminologies
    - Université Manchester (ALC – SHIQ - OWL DL), C++
    - Issue commerciale avec CELEBRA (Network Inference)
  - RACER (01-04) : **R**enamed **A**Box and **C**oncept **E**xpression **R**easoner
    - TBox et ABox (SHIQ (ALCQHIR+) – OWL DL)
    - Combiner DL et Algèbre Relationnelle
    - Issue commerciale avec RACERPro
  - PELLET (03-05) :
    - Raisonnement avec Abox (SHION(D))
    - Java, Open Source
- Moteurs RDF :
  - Modèle persistant : JENA2 (HP), SNOBASE (IBM)
  - Modèle volatile (en mémoire) : SESAM, METALOG, CORES, …

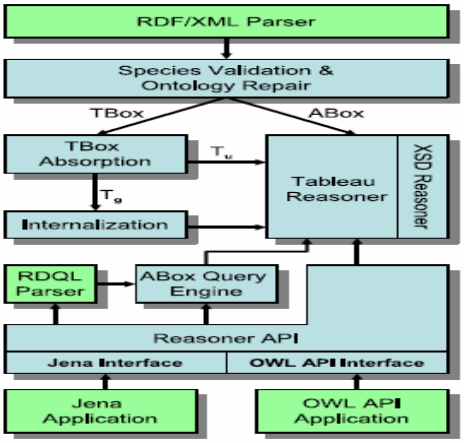
Implémentation de systèmes

- Architecture de PELLET



Implantation de systèmes

- Architecture de PELLET



Axes de recherche actuels

- Extension des Logique de description
  - Rôles complexes, domaines finis, domaines concrets, clés, e-connection, DDL, Composition, …
  - Future extensions de OWL (p.e, avec “rules”)
- Intégration de logique/systèmes
  - Par exemple, « Answer Set Programming »
- Techniques de raisonnement alternatives :
  - Algorithmes basés sur les automates
  - Translation dans datalog
  - Algorithmes basés sur les(pour un sous ensemble de langages $\mathcal{ALC}$

Axes de recherche actuels

- Scalabilité
  - Très grandes ontologies
  - Très grand nombre d'individus
- Autres taches de raisonnement (inférences non-standard)
  - Matching, LCS (Lest Common Subsumer), explication, requêtes, …
- Implémentation de tools et d'Infrastructure
  - langages plus expressifs (tels que $\mathcal{SHOIN}$et$\mathcal{SOIQ}$)
  - Techniques algorithmiques nouvelles
  - Tools pour le support ontologique : Editeur, visualiseur, etc.

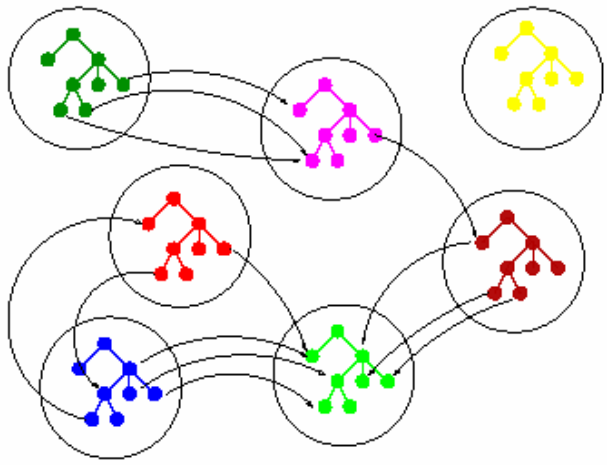
VI. Références

- [Baa03] F. Baader, D. Calvanese, D. McGuiness, D. Nardi and P.F. Patel-Schneider, « *The Description Logic Handbook: Theory, Implementation and Applications* » Cambridge University Press, 2003
- [Bor03] Borgida A., Serafini L. "Distributed Description Logics : Assimilating Information from Peer Sources" Journal of Data Semantics (1). 2003, pp.153-184
- [Don97] F.M. Donini, M. Lenzerini, D. Nardi, W. Nutt "*The Complexity of Concept Languages*" Information and Computation, 134, pp 1-58, 1997
- [Nap97] Amedeo Napoli, « *Une introduction aux logiques de description*" Rapport de recherche INRIA n°3314, décembre, 1997

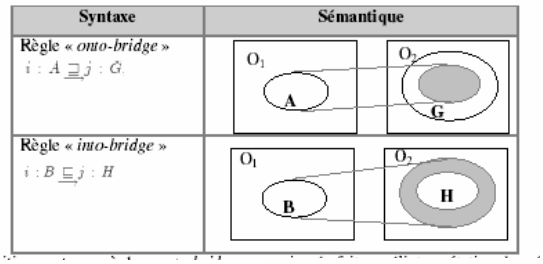
DDL et Drago

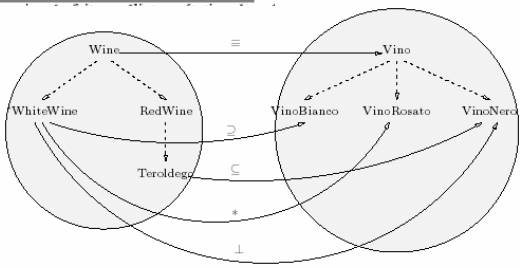
- Objectif :
  - Les principes de DDL
  - Le mécanisme d'inférence DDL
  - Modèle C-OWL
  - Architecture et fonctionnement du Drago
  - Vos remarques et/ou critiques
  - Autres approches surtout e-connection

DDL et Drago



DDL et Drago



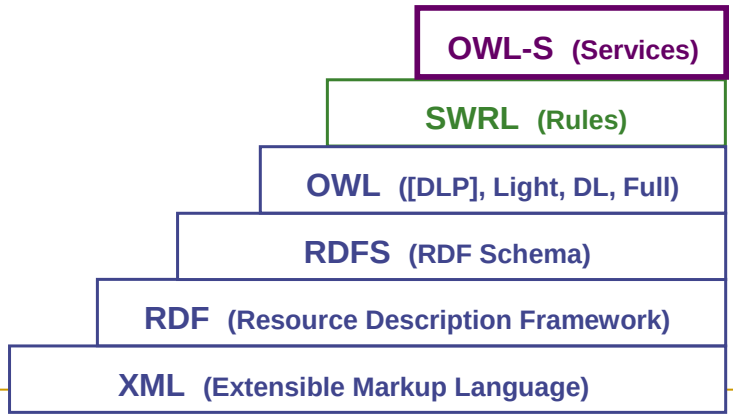


WSDL-S et METEOR-S

- Objectif :
  - Principes de représentation de la sémantique (RS) des web services
  - Le mécanisme de RS dans WSDL-S
  - Comparaison avec la solution OWL-S (DAML-S)
  - Etude des fonctionnalités de METEOR-S
  - Vos remarques et critiques
  - Identifier des différents problèmes à résoudre de RS dans les WS

WSDL-S et METEOR-S

OWL-S: an ontology expressed in OWL and related languages



WSDL-S et METEOR-S