Querying data through ontologies

Marie-Christine ROUSSET
Université Joseph Fourier de Grenoble

Ontologies [Gruber 92]

- Description explicite de connaissances partagées entre différents acteurs (personnes, applications, agents)
- Représentation abstraite et simplifiée du monde réel avec un but (une application) précis(e)

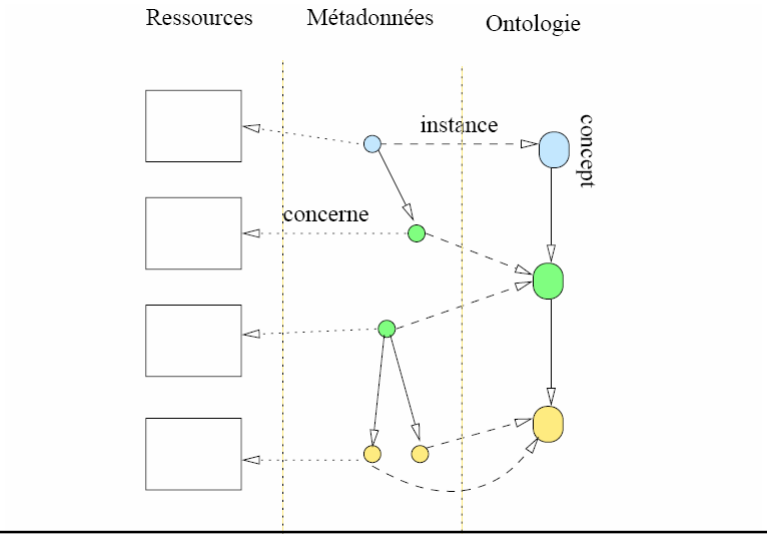
Exemples d'utilisation

Application	Conceptualisation	Utilisation
Pages Jaunes	domaines d'activités professionnelles	recherche de services
Amazon	types de loisirs	recherche de produits
LMD	types d'enseignements	orientation d'étudiants

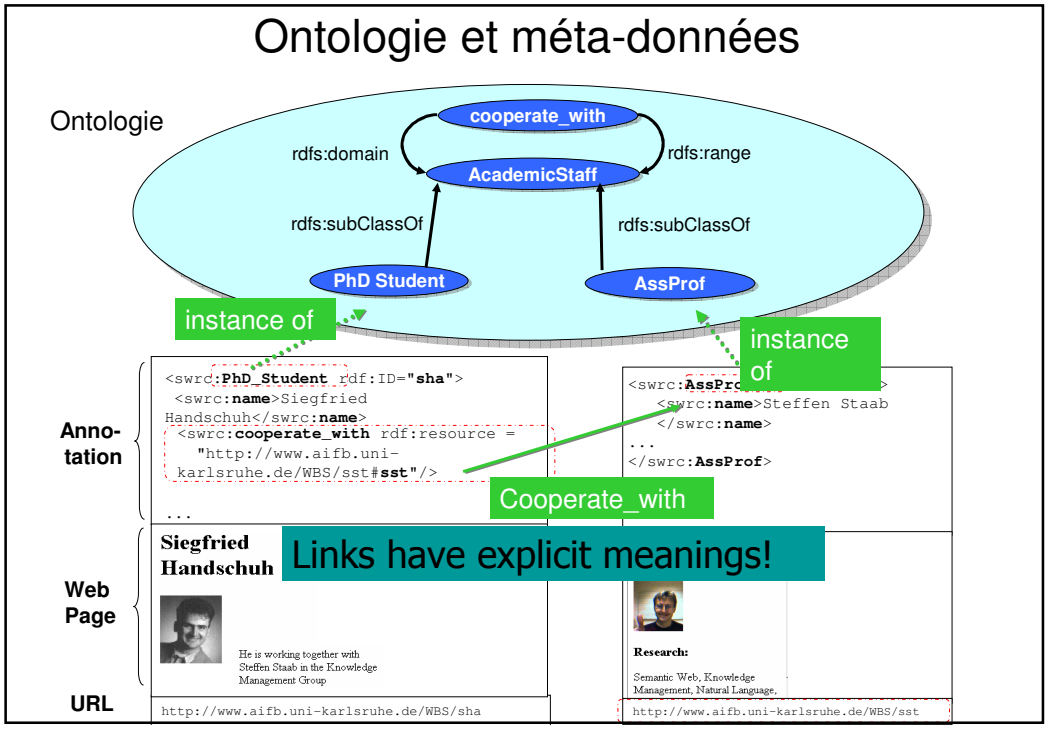
Ontologies (en bref)

- vocabulaires structurés
  - noms de concepts/classes
  - noms de propriétés
- définis à l'aide de langages formels
  - pour la définition et typage des concepts et des propriétés
  - permettant de faire des inférences fondées sur une sémantique logique
- peuvent être très simples … ou très complexes
  - universelles ou spécifiques d'un domaine
  - selon le niveau de finesse désiré pour la modélisation du domaine

Ontologies dans le Web sémantique : description de la sémantique de méta-données



Ontologie et méta-données



RDFS et OWL

- Standards de l'activité Web sémantique du W3C
  - Plusieurs syntaxes XML pour l'échange de méta-données RDF, de leurs schémas RDFS, d'ontologies en OWL
- RDFS et RDF
  - RDF (Resource Description Framework) : permet de définir des méta-données associées aux ressources du Web.
    - Une ressource peut être une page Web, mais aussi un concept (utilisé pour décrire une page Web) que l'on veut lui-même décrire.
  - RDFS: Définition du vocabulaire d'un domaine donné et des relations entre les objets de ce vocabulaire.
    - Avec une sémantique formelle logique associée
- OWL (Ontology Web Language)
  - Fondé sur les logiques de description (OWL-DL)

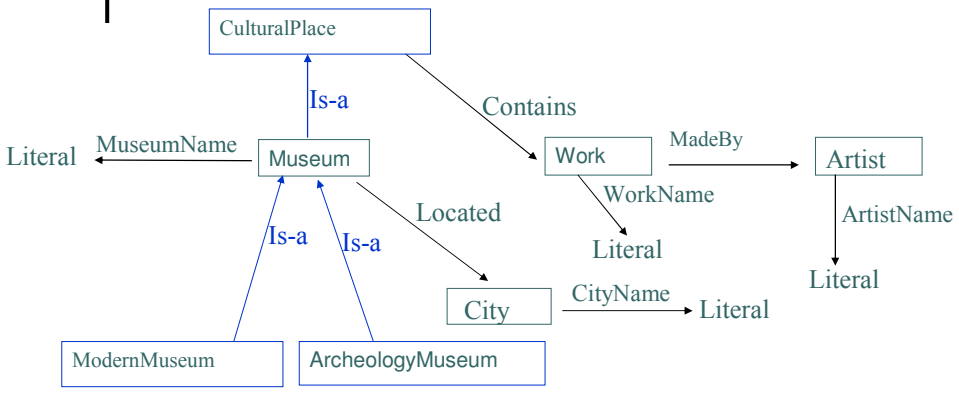
6

RDFS : pouvoir d'expression

Permet de définir
- des ***classes*** et une ***hiérarchie de spécialisation*** sur les classes.
- qu'une ressource RDF peut être une instance d'une classe RDFS (***rdf:type***).
- des ***propriétés*** et une ***hiérarchie de spécialisation*** sur les propriétés.
- des ***restrictions*** sur la valeur d'une propriété (***range***) et sur le type de ressource décrit par la propriété (***domain***).

7

RDFS : illustration



8

RDFS : sémantique formelle

- Fragment de la logique du premier ordre

Constructor	FOL axiomatization
Class inclusion	$\forall X(C_1(X) \Rightarrow C_2(X))$
Property inclusion	$\forall X \forall Y(P_1(X,Y) \Rightarrow P_2(X,Y))$
Domain typing of a property	$\forall X \forall Y(P(X,Y) \Rightarrow C(X))$
Range typing of property	$\forall X \forall Y(P(X,Y) \Rightarrow C(Y))$

- Sur lequel on peut greffer un langage de requêtes relationnel standard

Q(X): Museum(X) ∧ Contains(X,Y)∧ MadeBy(Y,Z) ∧ ArtistName(Z, »Picasso »)

9

RDFS : pouvoir d'expression limité

- Classes et propriétés atomiques
  - On ne peut pas définir la classe Personne comme l'union des classes Homme et Femme
- Contraintes limitées aux axiomes d'inclusion
  - pas d'axiomes de disjonction entre classes
    - on ne peut pas exprimer que Homme et Femme sont deux classes disjointes
  - pas de contrainte d'existence/cardinalité/unicité de valeurs
    - On ne peut pas exprimer que toute Personne a une mère ni que toute Personne a exactement deux parents
  - pas de propriété sur les propriétés
    - On ne peut pas dire que la propriété estPlusGrandQue est transitive, que le propriété estPèreDe est fonctionnelle, que la propriété estParentDe a pour inverse la propriété estEnfantDe

10

OWL: Ontology Web language

- Etend les standards existants du Web
  - Tels que XML, RDF, RDFS
- Fondé sur les Logiques de Description
  - issues de nombreux travaux en Représentation de Connaissances
  - Sémantique formelle logique: fragments décidables de la logique du premier ordre avec égalité
    - Algorithmes de raisonnement avec une étude approfondie de leur complexité
    - Des systèmes implémentés (RACER, PELLET)

11

Les constructeurs de classes de OWL

Constructor	DL Syntax	Example	Modal Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1 \wedge \ldots \wedge C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1 \vee \ldots \vee C_n$
complementOf	$\neg C$	¬Male	$\neg C$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x_1 \vee \ldots \vee x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$[P]C$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\langle P \rangle C$
maxCardinality	$\leqslant nP$	⩽1hasChild	$[P]_{n+1}$
minCardinality	$\geqslant nP$	⩾2hasChild	$\langle P \rangle_n$

- OWL fait la distinction entre :
  - des propriétés abstraites (e.g. “friend” or “father”)
  - des propriétés concrètes (e.g. “age” or “weight”)
    - Les datatypes de XML Schema peuvent être utilisés comme propriétés concrètes
- Les constructeurs de classes peuvent être imbriqués
  - Person∩ ∃ haschild (∀haschild.Doctor)

12

Les contraintes qui peuvent être déclarées en OWL

Axiom	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
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg$ {peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant 1$hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant 1$hasSSN$^-$

13

Problèmes d'inférence étudiés en LD

- Test de subsomption
  - Entre 2 expressions de concepts
  - Entre 2 expressions de concepts modulo un ensemble de contraintes (GCIs) définies dans une Tbox T:

    T |= C1 $\subseteq$ C2 ?
- Test d'appartenance d'une instance à un concept
  - Étant donnés un concept C, une constante a, une Tbox T, une Abox A (un ensemble de faits de la forme A(b) et P(b,c))

    T $\cup$ A |= C(a) ?
- Nombreux résultats de décidabilité et de complexité en fonction des constructeurs et des axiomes autorisés dans T

14

Quelques résultats de complexité

Constructeurs	Complexité de la subsomption
ALN (∩, forall, atleast, atmost)	P
ALE (∩, forall, exists)	NP-complet
ALNE (∩, forall, atleast, atmost, exists)	NP-complet
ALN + conjonction de rôles	co-NP-hard
ALC (∩, forall, not)	Pspace-complet
ALCNR	Pspace-complet

15

Requêtes

- Formule ouverte de la logique du premier ordre

  $q(\underline{x}): \exists\, \underline{y}\ \Phi(\underline{x},\underline{y})$

  - $\underline{x}$ est un vecteur de variables libres (distinguées)
  - $\Phi(\underline{x},\underline{y})$ est une formule ayant comme variables libres celles de $\underline{x}$ et $\underline{y}$ (pouvant aussi contenir des variables liées et des constantes)
- Exemple

$q(X) : \exists A,C\ Flight(X) \wedge ArrivalAirport(X,A) \wedge Located(A,C) \wedge Capital(C)$

Notation Datalog :

$q(X) \leftarrow Flight(X) \wedge ArrivalAirport(X,A) \wedge Located(A,C) \wedge Capital(C)$

16

Réponses à une requête

- Relativement à une base de connaissances K

  $Ans(q,K) = \{\underline{a} \mid K \models q(\underline{a})\}$

  - K est un ensemble de formules fermées
    - Une base de données (un ensemble de faits de la forme R(a1, …,an) où R est une relation du schéma) + éventuellement des constraintes sur ces relations
    - Abox ∪ Tbox (une BC en Logique de Description)
  - $\underline{a}$ est un vecteur de constantes apparaissant dans K
  - $q(\underline{a})$: la formule $\exists \underline{y}\ \Phi(\underline{x},\underline{y})$ où les variables de $\underline{x}$ sont remplacées par les constantes de $\underline{a}$

17

Evaluation d'une requête

- Un problème de raisonnement

  $q(\underline{x})$: $\exists \underline{y}\ \Phi(\underline{x},\underline{y})$

  $Ans(q,K) = \{\underline{a} \mid K \models q(\underline{a})\}$
- Cas particulier décidable

  K = instance d'une BD relationnelle

  q : une requête conjonctive

  $q(X) : \exists A,C\ Flight(X) \wedge ArrivalAirport(X,A) \wedge Located(A,C) \wedge Capital(C)$

  - pôlynomial en fonction de la taille des données, NP-complet en fonction de la taille de la requête
  - des algorithmes optimisés pour calculer efficacement Ans(q,K)

18

Test d'inclusion de requêtes

- Vérifier si Ans(q1, I(BD)) $\subseteq$ Ans(q2, I(BD)) pour tout I(BD)

  q1($\underline{x}$): $\exists$ $\underline{y1}$ $\Phi 1(\underline{x},\underline{y1})$

  q2($\underline{x}$): $\exists$ $\underline{y2}$ $\Phi 2(\underline{x},\underline{y2})$
- Un problème de raisonnement

  $\exists$ $\underline{y1}$ $\Phi 1(\underline{x},\underline{y})$ |= $\exists$ $\underline{y2}$ $\Phi 2(\underline{x},\underline{y2})$?
  - Notation BD: q1$\subseteq$ q2 ?

19

Cas particulier: inclusion de requêtes conjonctives

- NP-complet
- Algorithme illustré sur un exemple :

  q1(X): R(X,Y), R(Y,Z), R(Z,Z)

  q2(X): R(X,Y), R(Y,Z1), R(Y,Z2)
  - **q1 vu comme une instance de BD par « gel » de ses variables : X $\rightarrow$a1, Y $\rightarrow$a2, Z $\rightarrow$a3**
  - **évaluation de q2 sur cette « instance » de BD**
    - **si l'ensemble des réponses est non vide: OUI**
    - **sinon: NON (q1 est non incluses dans q2)**
  - Sur l'exemple: réponses= {a1,a2,a3} , donc q1$\subseteq$ q2

20

Autre cas particulier décidable

- Les requêtes sont exprimables en logique de description
  - Inclusion de requêtes = subsomption entre deux descriptions de concepts
  - Prise en compte d'axiomes sur le schéma
- Calcul des réponses à une requête
  - Problème de décision associé : test d'appartenance d'une instance à un concept $T \cup A \models C(a)$?
  - Mais pas ou peu d'algorithmes d'évaluation de requêtes

21

Algorithme d'évaluation naif

- Si la requête est unaire et exprimable en LD

  q(X) : $\exists A,C$ Flight(X)$\wedge$ ArrivalAirport(X,A) $\wedge$ Located(A,C) $\wedge$ Capital(C)

  $\equiv$ Flight$\cap$ $\exists$ArrivalAirport. ($\exists$Located.Capital) (X)
  - $a \in$ Ans(D, A $\cup$T) $\Leftrightarrow$ A $\cup$T $\cup\{\neg D(a)\}$ est insatisfiable
  - nécessite le pouvoir d'expression d'au moins ALC même si A, T et D sont exprimés en AL(E)
    - Test d'insatisfiabilité est P-Space complet,
    - des algorithmes de tableaux de complexité exponentielle
  - Algorithme d'évaluation naif
    - Pour chaque constante a de la Abox, tester si A $\cup$T $\cup\{\neg D(a)\}$ est insatisfiable
      - en théorie, ne rajoute que la taille de la Abox dans l'ordre de grandeur de complexité, déjà exponentiel
      - en pratique, irréalisable si la Abox est stockée dans une BD (avec des millions de tuples)

22

Décidable mais non « tractable »

- Si le problème de décision « a est une réponse ? » n'est pas polynômial en fonction de la taille des données (de la Abox)
- La réduction de ce problème à l'insatisfiabilité de A ∪T ∪{¬D(a)} est une impasse pour trouver des fragments « tractables » :
  - si l'ontologie est exprimée en AL ou une LD plus expressive
  - et si la requête contient une conjonction car la négation de la requête ajoute nécessite le pouvoir d'expression d'au moins ALC pour laquelle le test d'insatisfiabilité est dans PSPACE
- Nouvelle approche requise :
  - qui englobe si possible le cas des requêtes conjonctives quelconques (non seulement celles qui sont exprimables en LD) au dessus d'ontologies (qui doivent nécessairement être peu expressive)

Marge de manœuvre limitée (illustrée sur un exemple)

Qui montre la difficulté d'interroger une Abox comme un SGBD …

à cause de la connaissance exprimée dans la Tbox (l'ontologie)

C	D	D'	R	S
a e	b f	d k	a , b a , c a , d	b,c c,d

A

D' ≡ ¬ D

T

q(X) ← C(X),R(X,Y),D(Y),S(Y,Z), D'(Z)

Ans(q, A) = ∅

Ans(q, A ∪T) = {a}

Nécessite un raisonnement par cas sur les constantes: ici sur la constante c

- soit D(c) est vrai (et {X/a , Y/c , Z/d} satisfait la requête)
- soit D(c) est faux et donc D' (c) est vrai (et {X/a , Y/b , Z/c} satisfait la requête)

Calcul des réponses relativement à une BD + contraintes

- Monde Fermé
  - Les contraintes servent à vérifier la cohérence mais aussi la complétude de la BD

  Professor ⊆ ∃TeachesTo

  Contrainte référentielle : toutes les constantes de la table **Professor** doivent apparaître dans la première colonne de la table **TeachesTo**
  - Elles n'interviennent pas dans le calcul des réponses
- Monde Ouvert
  - Les contraintes sont des connaissances supplémentaires sur les faits déclarés dans la BD
  - Peuvent intervenir dans le raisonnement sous-jacent au calcul des réponses à la requête

25

Exemple

K:

Professor
Mary
Jim
Tom

TeachesTo	
Mary	Bill
John	Ann

+

Professor ⊆ ∃TeachesTo

q(x): TeachesTo(x,y)

Ans(q,K) = {Mary, John, Jim, Tom}

26

Calcul de réponses en Monde Ouvert

- Problème en général aussi difficile que raisonner en FOL ou dans des fragments de FOL
  - Pour chaque tuple **a** de constantes, **K |= q(a)** ?
- Fragments décidables pour lesquels Ans(q,K) est polynômial en fonction des données
  - BD déductives: K = BD + règles
  - DL-Lite: K = Abox + Tbox
- Approche commune:
  - reformulation de la requête en une disjonction de requêtes évaluables directement sur une BD

27

La famille DL-Lite

- Restriction sur les constructeurs et les contraintes autorisées pour une complexité polynômiale du calcul des réponses à une requête conjonctive
- Approche suivie
  - Trouver des logiques de description (ou des langages logiques) pour lesquels le calcul de réponses est réductible à l'évaluation de requêtes sur une BD relationnelle simple
  - Condition nécessaire: la reconnaissance qu'un tuple est une réponse est LOGSPACE (et donc P)

28

DL-Lite : constructeurs et axiomes

- DL-Lite$_{core}$:
  un ensemble d'axiomes d'inclusion $B \subseteq C$ où B et C sont des expressions de concept sur des concepts atomiques A et des rôles atomiques P de la forme
  $B \rightarrow A \mid \exists R \qquad R \rightarrow P \mid P^- \qquad C \rightarrow B \mid \neg B$
- DL-Lite$_R$:
  DL-Lite$_{core}$ + axiomes d'inclusion de rôles $R \subseteq E$ où E est de la forme R ou $\neg R$
- DL-Lite$_F$:
  DL-Lite$_{core}$ + axiomes de fonctionnalité : (funct R)

29

Expressivité de DL-Lite

- Capture les principales contraintes utilisées en bases de données, génie logiciel et RDFS
  - Relation ISA : $A1 \subseteq A2$
  - Disjonction : $A1 \subseteq \neg A2$
  - typage (RDFS):
    - $\exists P \subseteq A1$ (le premier attribut de P est typé par A1)
    - $\exists P^- \subseteq A2$ (le second attribut de P est typé par A2)
  - Propriétés obligatoires ou interdites :
    - $A \subseteq \exists P \qquad A \subseteq \exists P^- \qquad A \subseteq \neg\exists P \qquad A \subseteq \neg\exists P^-$
  - Restriction de fonctionnalité pour les relations binaires (les rôles)

30

Exemple

$\text{Professor} \subseteq \exists \text{TeachesTo}$
$\text{Student} \subseteq \exists \text{HasTutor}$
$\exists \text{TeachesTo}^- \subseteq \text{Student}$
$\exists \text{HasTutor}^- \subseteq \text{Professor}$
$\text{Professor} \subseteq \neg \text{Student}$

$\text{HasTutor}^- \subseteq \text{TeachesTo}$ $\quad$ DL-Lite_R
(func HasTutor) $\quad$ DL-Lite_F

31

Requêtes au dessus de DL-Lite

Abox A :
Professor(Jim) HasTutor(John, Mary) TeachesTo(John, Bill)
Tbox T :
$\text{Professor} \subseteq \exists \text{TeachesTo}$
$\text{Student} \subseteq \exists \text{HasTutor}$
$\exists \text{TeachesTo}^- \subseteq \text{Student}$
$\exists \text{HasTutor}^- \subseteq \text{Professor}$
$\text{Professor} \subseteq \neg \text{Student}$
Requêtes conjonctives sur des concepts et rôles atomiques :

$q0(x) \leftarrow \text{TeachesTo}(x,y) \wedge \text{HasTutor}(y,z)$

32

Reformulation de requêtes

- Algorithme de reformulation : illustration
  - q1(x) ← TeachesTo(x,y)∧Student(y)
  - q2(x) ← TeachesTo(x,y)∧TeachesTo(z',y)
  - q3(x) ← TeachesTo(x,y')
  - q4(x) ← Professor(x)
  - q5(x) ← HasTutor(u,x)
  - Pour chaque i: **qi,T |= q0**
  - **Ans(q0,T ∪ A) = $\cup_i$ Ans(qi,A)**
- Algorithme correct si T∪A est satisfiable

33

Illustration sur l'exemple

Abox A :

Professor(Jim) HasTutor(John, Mary) TeachesTo(John, Bill)

Requête: q0(x) ← TeachesTo(x,y)∧HasTutor(y,z)

Reformulations:

q1(x) ← TeachesTo(x,y)∧Student(y)
q2(x) ← TeachesTo(x,y)∧TeachesTo(z',y)
q3(x) ← TeachesTo(x,y')
q4(x) ← Professor(x)
q5(x) ← HasTutor(u,x)

Ans(q,A∪T) = {Mary, Jim, John}

34

Test de satisfiabilité

- T ∪ A peut être insatisfiable

Abox A :

Professor(Jim) HasTutor(John, Mary) TeachesTo(John, Bill)

Tbox T :

Professor ⊆ ∃TeachesTo

Student ⊆ ∃HasTutor

$\exists TeachesTo^{-} \subseteq$ Student

$\exists HasTutor^{-} \subseteq$ Professor

Professor ⊆ ¬Student **+ ∃TeachesTo ⊆ ¬ Student**

∃HasTutor ⊆ Student

35

Test de satisfiabilité dans DL-Lite

- On sature les inclusions négatives et on les traduit en des requêtes conjonctives booléennes qui sont évaluées sur la Abox vue comme une BD
  - vrai ssi A ∪ T insatisfiable

Abox A :

Professor(Jim) HasTutor(John, Mary) TeachesTo(John, Bill)

Tbox T :

∃TeachesTo ⊆ ¬ Student

∃HasTutor ⊆ Student

∃TeachesTo ⊆ ¬ ∃HasTutor

Q: TeachesTo(X,Y)∧ HasTutor(X,Y')

36

Résumé

- Répondre (de façon complète) à des requêtes par récriture n'est « tractable » que pour très peu de Logiques de Description
- La reconnaissance d'instance (et donc le calcul des réponses à une requête) est coNP-hard en complexité des données pour des extensions mineures of DL-Lite$_{core}$ telles que :
  - DL-Lite$_{FR}$
  - $\forall$R.A ou $\neg$A apparaissent en partie gauche d'un axiome d'inclusion
  - $\forall$R.A ou $\neg$A apparaissent dans la requête
  - Union de concepts en partie gauche d'un axiome d'inclusion

37

Impact sur l'interrogation de RDFS

- SPARQL un langage de requêtes sur des données/graphes RDF de type relationnel
- Problème encore ouvert: extension à RDFS
  - prendre en compte les axiomes d'inclusion de classes et propriétés, et de typage des propriétés
- RDFS est un fragment de DL-Lite$_{R}$
  - Reformulation d'une requête SPARQL en une union de requêtes SPARQL évaluables directement sur le graphe RDF
  - On peut étendre RDFS jusqu'à DL-Lite$_{R}$ et garder cette propriété de répondre à des requêtes par reformulation

38

Références

- Logic-based techniques in data integration. A. Halevy, in Logic Based Artificial Intelligence, Ed. Jack Minker, Kluwer Publishers, 2000.
- Query Containment for Data Integration Systems, T. Millstein, A.Levy, M. Friedman, Proceedings PODS 2000
- Decidable reasoning in terminological knowledge representation systems,
  Buccheit M., Donini, F., Schaerf A.,
  Journal of Artificial Intelligence Research, Volume 1, 1993
- Reasoning in Description Logics,
  Donini F., Lenzerini M., Nardi D., Schaerf A.,
  Principles of Artificial Intelligence, G. Brewka editor, Springer Verlag, 1995
- Transparents de Bernd Amann, cours BDIA-M2 de l'Université Paris6
- D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, DL-Lite: Tractable Description Logics for Ontologies, Proceedings of AAAI 2005.
- D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Data Complexity of Query Answering in Description Logics, Proceedings of KR 2006