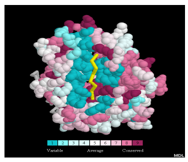
Logic-based techniques for Information Integration

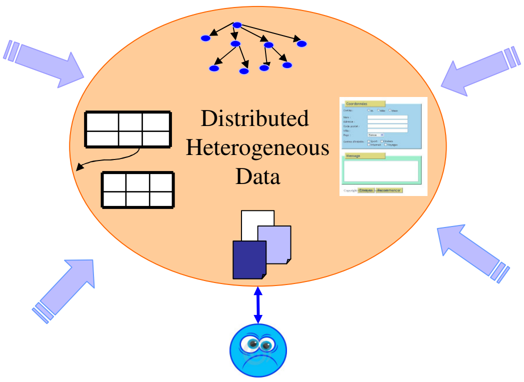
Marie-Christine ROUSSET
LIG
University of Grenoble

Information Integration

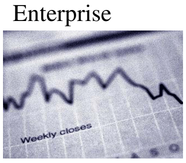
Web

Sciences





Enterprise



Administration

a difficult challenge !

2

Semantics: the glue between heterogeneous data sources

- Overview of some **challenges** and **existing solutions** for representing and exploiting the semantics
  - to describe and query heterogeneous pre-existing autonomous data sources
- Logic:
  - an appropriate formal background with associated automatic reasoning techniques

Focus on the use of logic for two primary challenges

(1) Describe and compare the content of pre-existing data sources

(2) Create single query interface over multiple and heterogeneous data sources

Illustration (Challenge1)

Source 1: ***Flights*** **with atmost one** ***Stop***

Source 2: **Direct** ***Flights*** **(without** ***Stop*****)**

Source 3: ***Flights*** **whose Stop(s) are in** ***AmericanCities*** **only**

Source 4: ***Flights*** **with atleast one** ***Stop*** **in an** ***AmericanCity***

Only possible comparison resulting from the description in English:

Source 2 and Source 4 are disjoint

Other comparisons grounded on the logical semantics:

Source 1 $\cap$ Source 4 $\subseteq$ Source3 (under completeness assumption of Source 3)

Source 2 $\subseteq$ Source3 (under completeness assumption of Source 3 and depending on the logical interpretation of « whose …only »)

Modeling in (description) logic

- Force to solve ambiguities and/or to set clear hypotheses by a set of formulas and axioms
- Gain automatic reasoning on those formulas and axioms

Source 1: **The or Some** ***Flights*** **with atmost one** ***Stop***

Source1 $\equiv$ Flight $\cap$ ($\leq$ 1 Stop) versus **Source1 $\subseteq$ Flight $\cap$ ($\leq$ 1 Stop)**

Source 2: ***Flights*** **without** ***Stop***

Source2 $\subseteq$ Flight $\cap$ ($\leq$ 0 Stop)

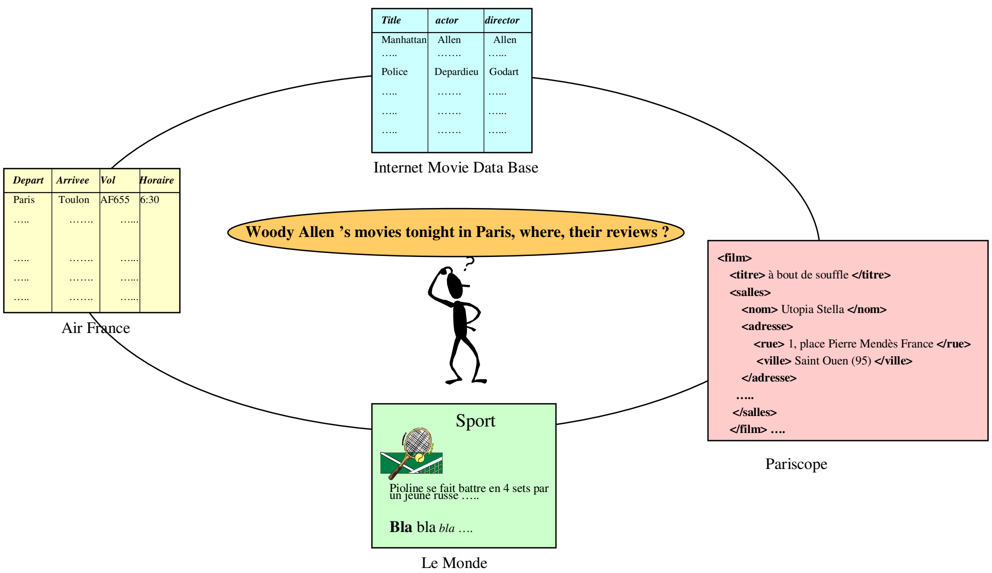
Source 3: **The** ***Flights*** **whose** ***Stops*** **are in** ***AmericanCities*** **only**

Source3 $\equiv$ Flight $\cap \forall$ Stop.AmericanCity

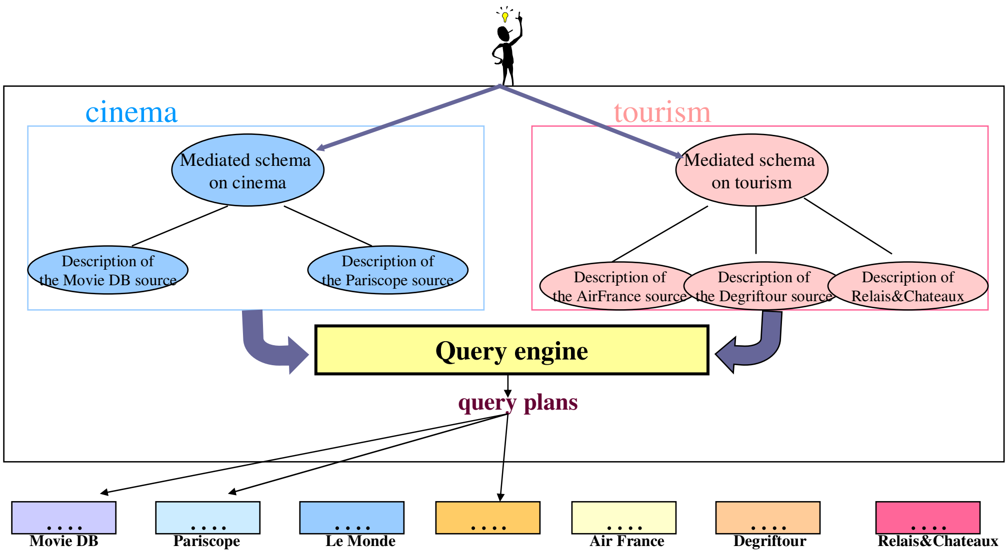
Source 4: ***Flights*** **with atleast one** ***Stop*** **in an** ***AmericanCity***

Source4 $\subseteq$ Flight $\cap \exists$ Stop.AmericanCity

Illustration (Challenge2)



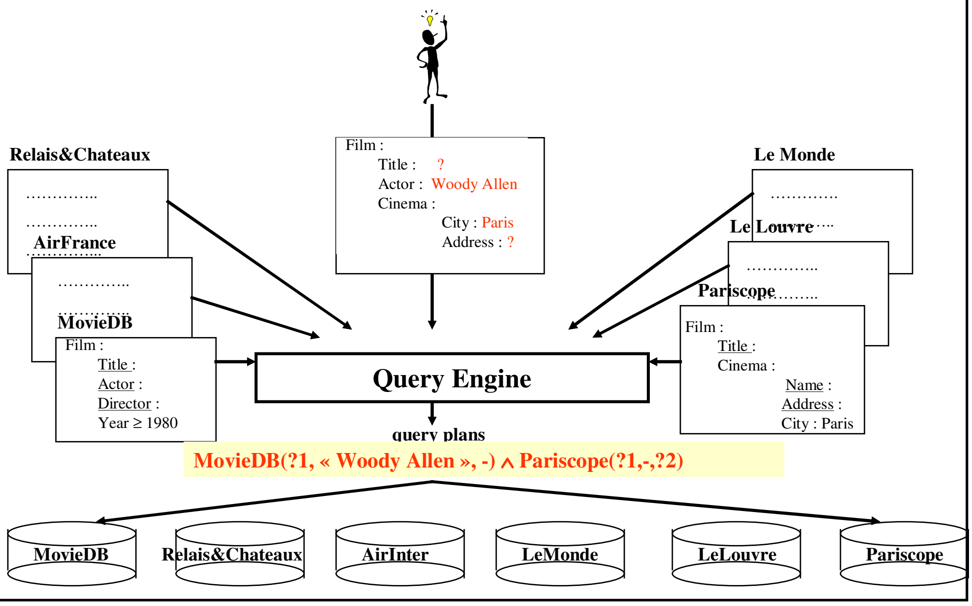
The mediator approach



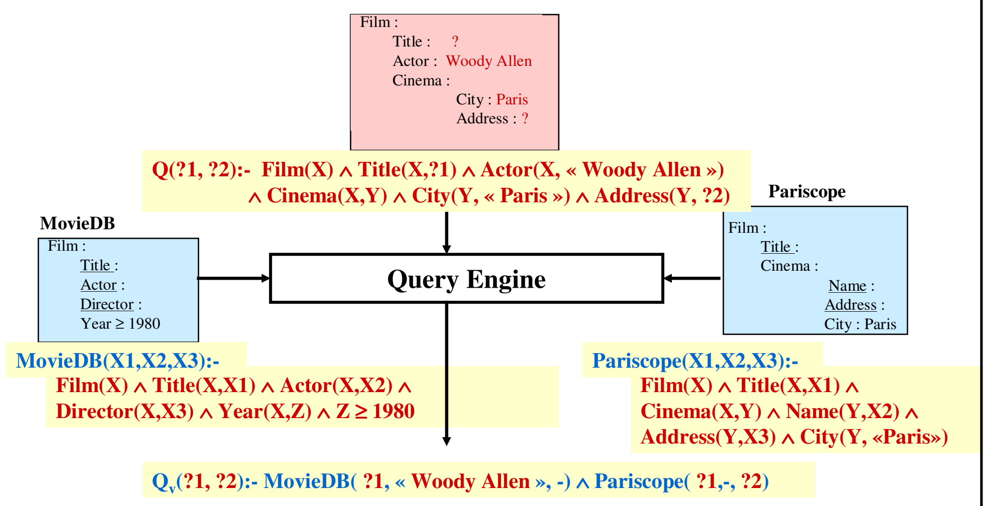
Modeling and algorithmic issues

- Define a mediated schema
  - Structured vocabulary serving as a query interface for users queries
- Model the content of the sources to integrate in terms of the mediated schema
- Reformulate et decompose the users queries in queries executable against the data sources
- Combine the answers of local queries to build the answers of the global queries

Reformulation of the query: illustration



The underlying machinery is logic-based



Plan

- Logical foundations of databases
  - for modeling schemas, constraints, queries, views
  - for query containment and rewriting
- Description logics in a nutshell for reasoning on data semantics
- Models and algorithms for (virtual) integration of heterogeneous data sources

Logic: a unifying framework for posing and solving database problems

Queries

- Open formula of first-order logic (FOL)

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

  – $\underline{x}$ is a vector of free (distinguished) variables
  – $\Phi(\underline{x},\underline{y})$ is a formula the free variables of which are those of $\underline{x}$ and $\underline{y}$ (with possibly constants and bound variables)
- Example (conjunctive query)

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

Datalog notation :

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

Logical semantics

- Answers to a query q relatively to a <u>KB</u> K

  Ans(q,K) = {<u>a</u> | K |= q(<u>a</u>)}

  – K is a set of closed formulas
    - A DB extension (a set of facts R(a1, …,an) where R is a relation of the schema) + possibly contraints
    - Abox $\cup$ Tbox (a KB expressed in Description Logic)

  – <u>a</u> is a vector of constants appearing in K

  – q(<u>a</u>): obtained from $\exists\ \underline{y}\ \Phi(\underline{x},\underline{y})$ by replacing variables of <u>x</u> with constants of <u>a</u>

  K |= q(<u>a</u>): the tuple <u>a</u> satisfies the query q in all the interpretations satisfying (models of) K

Interpretations of formulas in logic

- an interpretation I of φ:
  – A domain of interpretation Δ^I
  – A function of interpretation mapping
    - constants a in φ to elements a^I of Δ^I
    - (functions f and relations R) in φ to (functions f^I from Δ^I to Δ^I) to relations R^I on Δ^I
  – Rules of interpretation for interpreting any formula from the interpretation of its sub-formulas
    - Interpretation of a closed formula: true or false
    - Interpretation of an open formula with n free variables: an n-ary relation on Δ^I
- A model of a formula φ: an interpretation I such that
  – φ^I = true (if φ is closed)
  – $\varphi^I \neq \emptyset$ (if φ is open)

Rules of interpretation for quantifiers

- Let φ be a closed formula of the form $\forall x\ \psi$

  $[\forall x\ \psi(x)]^I$ = true iff for every $e \in \Delta^I$, $\psi^I(e)$ is true
- Let φ be a closed formula of the form $\exists x\ \psi$

  $[\exists\ x\ \psi(x)]^I$ = true iff there exists $e \in \Delta^I$, $\psi^I(e)$ is true
- Let $\varphi(x1,\ldots,xn)$ be a formula with n free variables I

  $[\varphi(x1,\ldots,xn)]^I = \{(e1,\ldots,en) \in \Delta^I \times \ldots \times \Delta^I / \varphi^I(e1,\ldots,en)$ is true$\}$

K |= q(<u>a</u>): particular case

K is a DB extension, q a positive formula

- **K can be viewed as an Herbrand model**
  - Δ^I = the set of all the constants in the DB extension
  - $a^I = a$ for every constant a
  - $R^I = R$ for every relation R
- K |= q(<u>a</u>) is reduced to evaluate q(<u>a</u>) in K
- **If q is a conjunctive query**
  - K |= q(<u>a</u>) is true iff there exists a mapping m from the constants and existential variables of q(<u>a</u>) to constants in K such that for every conjunct R(t1,…, tn) of q(<u>a</u>): R(m(t1), …, m(tn)) is in K

  Homomorphism theorem in [Chandra-Merlin 77]:

  Optimal implementation of conjunctive queries in relational database

  9th ACM symposium on Theory of Computing (STOC'77)

Reasoning problems in FOL

- Satisfiability checking of a formula or a set of formulas
  - existence of a model
- Logical entailment: $\varphi 1, \ldots, \varphi n \models \varphi$
  - Every model of $\varphi 1, \ldots, \varphi n$ is a model of φ
  - Can be reduced to satisfiability checking:

    $\varphi 1, \ldots, \varphi n, \neg\varphi$ is unsatisfiable
- Semi-decidable problems
  - There does not exist an algorithm to decide whether <u>any</u> formula is satisfiable or not (is entailed or not by a given set of formulas)
  - Infinite number of interpretations of a FOL formula

Query evaluation

- A reasoning problem

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

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

  K = DB extension

  q : conjunctive query

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

  - polynomial in the size of the data, NP-complete in the size of the query

    (results from the homomorphism theorem [Chandra-Merlin 77])
  - Optimized algorithms for efficient computation of Ans(q,K) in DBMSs

Query containment

- Let q1($\underline{x}$): $\exists\ \underline{y1}\ \Phi 1(\underline{x},\underline{y1})$ and q2($\underline{x}$): $\exists\ \underline{y2}\ \Phi 2(\underline{x},\underline{y2})$

q1$\subseteq$ q2 iff

Ans(q1, I(DB)) $\subseteq$ Ans(q2, I(DB)) for every I(DB)

- Another reasoning problem:

  $\exists\ \underline{y1}\ \Phi 1(\underline{x},\underline{y}) \models \exists\ \underline{y2}\ \Phi 2(\underline{x},\underline{y2})$?

Particular case: containment of conjonctive queries

- NP-complete problem
- Algorithm illustrated on an example :

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

  q2(X'): R(X',Y'), R(Y',Z'1), R(Y,Z'2)

  – **q1 viewed as a DB extension by freezing its variables : X , Y and Z considered as constants**

  – **evaluating q2 against this DB « extension »**

    - **If X is an answer : YES**
    - **If not: NO (q1 is not contained in q2)**

  – On the example: X is an answer and thus q1$\subseteq$ q2

No containment: example

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

q2(X'): R(X',Y'), R(Y',Z'1), R(Y',Z'2)

- **q2 $\not\subset$ q1**
- **Freezing the variables of q2:**

X' , Y' , Z'1 , Z'2 are distinct constants

- **Evaluation of q1 against this D « extension »**
  - Ans(q1, freeze(q2)) = $\emptyset$ thus: q2 $\not\subset$ q1

Query containment

- Central problem for the comparison of different data sources
  - A query : a formula that describes in a compact way the content of a data source
- Other decidable cases :
  - When the queries are expressible in Description Logic
  - Query containment = subsomption between two concept descriptions
  - Extended with constraints on the schema: inclusion statements between concept expressions

Description logics in a nutshell for reasoning on data semantics

Description Logics

- Logic-based representation of classes of objects using a set of constructors (having a FOL semantics)
  - Decidable fragments of FOL based on unary and binary predicates
    - Unary predicates: classes (called **concepts**)
    - Binary predicates: properties (called **roles**)
- Many decidability and complexity results for reasoning problems
- Implemented reasoners: RACER, PELLET

Description Logics by example

the description :

$Paper \cap (\exists\ Author\ PhDStudent) \cap (\exists\ Author\ (\neg\ PhDStudent))$

is subsumed by : $Paper \cap (\geq 2\ Author)$

is disjoint with: $Paper \cap (\forall\ Author\ PhDStudent)$

Description Logics by example

the query / the source content :

$Paper \cap (\exists\ Author\ PhDStudent) \cap (\exists\ Author\ (\neg\ PhDStudent))$

$\{x |\ \mathbf{Paper}(x) \wedge \exists y\ (\mathbf{Author}(x,y) \wedge \mathbf{PhDStudent}\ (y)) \wedge \exists z (\mathbf{Author}\ (x,z) \wedge \neg\ \mathbf{PhDStudent}(z))\}$

is contained in : $Paper \cap (\text{atleast } 2\ Author)$

$\{x |\ \mathbf{Paper}(x) \wedge \exists y\ \exists z\ (\mathbf{Author}(x,y) \wedge (\mathbf{Author}\ (x,z) \wedge (y \neq z)\}$

is disjoint with: $Paper \cap (\forall\ Author\ PhDStudent)$

$\{x |\ \mathbf{Paper}(x) \wedge (\forall y\ \ (\mathbf{Author}(x,y) \Rightarrow \mathbf{PhDStudent}\ (y))\ \}$

(restricted) negation

FOL semantics of the main constructors

$(C1 \cap C2)(X) \equiv C1(X) \wedge C2(X)$

$(C1 \cup C2)(X) \equiv C1(X) \vee C2(X)$

$(\forall\ R\ C)(X) \equiv \forall Y\ (R(X,Y) \Rightarrow C(Y))$

$(\exists\ R\ C)(X) \equiv \exists Y\ (R(X,Y) \wedge C(Y))$

$$(\geq n\ R)(X) \equiv \exists Y_1 \ldots Y_n\ (R(X, Y_1) \wedge \ldots \wedge R(X, Y_n) \wedge \bigwedge_{\{i,j/i \neq j\}} Y_i \neq Y_j$$

$$(\leq n\ R)(X) \equiv \forall\ Y_1 \ldots Y_{n+1}\ (R(X, Y_1) \wedge \ldots \wedge R(X, Y_{n+1}) \Rightarrow \bigvee_{\{i,j/i \neq j\}} Y_i = Y_j)$$

A Description Logic KB

Abox A : a set of facts

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

Tbox T : a set of General Concept Inclusions (CGI)

Professor $\subseteq \exists$TeachesTo

Student $\subseteq \exists$HasTutor

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

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

Professor $\subseteq \neg$Student

Reasoning problems

- Subsumption checking
  - Between two concept descriptions
  - Between two concept descriptions given a set of GCIs defined in a Tbox T:

    $T \models C1 \subseteq C2$?
- Membership checking of an instance to a concept
  - Given a concept C, a constant a, a Tbox T, an Abox A (a set of facts of the form A(b) and P(b,c))

    $T \cup A \models C(a)$?
- Many decidability and complexity results in function of the constructrors and the GCIs allowed in T

Some results of complexity

Constructors	**Complexity of subsumption checking**
ALN ($\cap$, $\forall$, $\geq$, $\leq$)	P
ALE ($\cap$, $\forall$, $\exists$)	NP-complet
ALNE ($\cap$, $\forall$, $\exists$, $\geq$, $\leq$)	NP-complet
ALN + conjunction of roles	co-NP-hard
ALC ($\cap$, $\forall$, $\neg$)	Pspace-complet
DL-Lite (GCIs with restriction on $\neg$, $\exists$)	**P**

Example of constraints expressible in DL-Lite

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

Expressivity of DL-Lite

- Captures the main constraints used in DB and Software Engineering
  - Relation ISA : A1 $\subseteq$ A2
  - Disjunction : A1 $\subseteq \neg$A2
  - typing :
    - $\exists$P $\subseteq$ A1 (the first attribute of P is typed by A1)
    - $\exists$P $^{-} \subseteq$ A2 (the second attribute of P is typed by A2)
  - Mandatory or forbidden properties :
    - A $\subseteq \exists$P   A $\subseteq \exists$P $^{-}$ A $\subseteq \neg\exists$P   A $\subseteq \neg\exists$P $^{-}$
- Extends RDFS and corresponds to a profile of OWL2
  - Languages for describing metadata and ontologies
  - W3C standards for the Semantic Web

DL used for reasoning on data

- Comparing different data sources described using DL
  - Inclusion, disjointness
- Checking query containment in presence of constraints on the schema
- Checking data consistency
- Checking that there exists an answer for a query without evaluating it
- Reformulating queries (generalization, specialization)
- DL reasoning is « open-world »: the data are incomplete
  - It is the case for Web data (in contrast with DBMS)

Close versus Open World

- Close World
  - Constraints are used to check consistency but also the completeness of the DB

  Professor $\subseteq$ **∃TeachesTo**

  Referential constraint: all the constants of the table **Professor** must appear in the table **TeachesTo**
  - Constraints are then not used to compute the answers
- Open World
  - Constraints are additional knowledge on the (incomplete) data declared in the DB
  - They can be used in the reasoning underlying the query evaluation

Example

K:

Professor		TeachesTo	
Mary		Mary	Bill
Jim		John	Ann
Tom			

+

Professor $\subseteq$ ∃TeachesTo

q(x): TeachesTo(x,y)

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

Answering queries in Open World

- Problem in general as hard as reasoning in FOL or in fragments of FOL
  - For each tuple **a** of constants, **K |= q(a)** ?
- Decidable fragments for which Ans(q,K) is polynomial in the size of the data
  - Deductive DB: K = DB + rules
  - DL-Lite: K = Abox + Tbox
- Common approach:
  - reformulation of the query in a union of queries that are directly executable against a DB

Illustration in DL-Lite

Abox A :

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

Tbox T :

Professor $\subseteq \exists$TeachesTo

Student $\subseteq \exists$HasTutor

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

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

Professor $\subseteq \neg$Student

Conjunctive queries on concepts and atomic roles :

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

Query reformulation

- Reformulation algorithm: illustration

  q1(x) $\leftarrow$ TeachesTo(x,y)$\wedge$Student(y)

  q2(x) $\leftarrow$ TeachesTo(x,y)$\wedge$TeachesTo(z',y)

  q3(x) $\leftarrow$ TeachesTo(x,y')

  q4(x) $\leftarrow$ Professor(x)

  q5(x) $\leftarrow$ HasTutor(u,x)

  – For each i: **qi,T |= q0**

  – **Ans(q0,T $\cup$ A) = $\cup_i$ Ans(qi,A)**
- Sound algorithm if T$\cup$A is satisfiable

Illustration

Abox A :

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

Query: $q0(x) \leftarrow TeachesTo(x,y) \wedge HasTutor(y,z)$

Reformulations:

$q1(x) \leftarrow TeachesTo(x,y) \wedge Student(y)$

$q2(x) \leftarrow TeachesTo(x,y) \wedge TeachesTo(z',y)$

$q3(x) \leftarrow TeachesTo(x,y')$

$q4(x) \leftarrow Professor(x)$

$q5(x) \leftarrow HasTutor(u,x)$

Ans(q,A∪T) = {Mary, Jim, John}

Satisfiability checking

- T ∪ A may be unsatisfiable

Abox A :

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

Tbox T :

$Professor \subseteq \exists TeachesTo$

$Student \subseteq \exists HasTutor$

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

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

$Professor \subseteq \neg Student$ **+** $\mathbf{\exists TeachesTo \subseteq \neg Student}$

$\mathbf{\exists HasTutor \subseteq Student}$

Satisfiability checking in DL-Lite

- Saturation of the Negative Inclusions (NIs) that are translated in boolean conjunctive queries evaluated against the Abox seen as a DBMS
  - True iff $A \cup T$ unsatisfiable

Abox A :

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

Tbox T :

$\exists$TeachesTo $\subseteq \neg$ Student

$\exists$HasTutor $\subseteq$ Student

$\exists$TeachesTo $\subseteq \neg$ $\exists$HasTutor

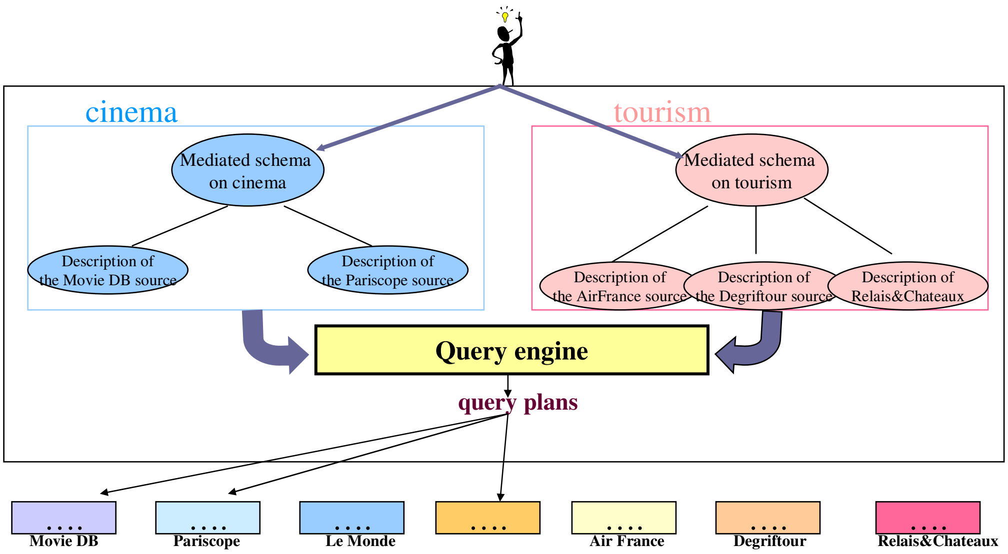
Q: TeachesTo(X,Y)$\wedge$ HasTutor(X,Y')

References

- 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
- 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

Models and algorithms for (virtual) integration of heterogeneous data sources

The mediator approach



Underlying principles

- Defining a mediated schema (also called a **global schema**) : serving as query interface for users
- Specifying **schema mappings** between the global schema and the schemas of the local data sources
  - Global-As-Views (GAV) approach: the global relations are defined as **views** over the local relations
  - Local-As-Views (LAV) approach: the local relations are defined as **views** over the global relations
- Rewriting the users queries (expressed using global relations) in terms of local relations => logical query plan

Views

- Named queries that can be re-used in other queries
  - Represents by a formula the answer set of a query or the content of a data source
- Example

Source1(X,Y1,Y2) : Flight(X) $\wedge$ DepartureAirport(X,Y1) $\wedge$ ArrivalAirport(X,Y2)
Source2(X,Y) : Place(X) $\wedge$ Located(X,Y) $\wedge$ Capital(Y)

- Can be materialized or virtual
  - Their extension is stored (in memory or in a cache) or computed on demand (by querying a data source)

Different semantics

of the correspondance $v(\underline{x}) : def(\underline{x},\underline{y})$ between the view and the query defining it:

- « exact» semantics

  Ext(v) = Ans(def,K)

  axiom : $\forall \underline{x}\ [v(\underline{x}) \Leftrightarrow \exists y\ def(\underline{x},\underline{y})]$ added to K
- « sound » semantics

  Ext(v) $\subseteq$ Ans(def,K)

  axiom : $\forall \underline{x}\ [v(\underline{x}) \Rightarrow \exists y\ def(\underline{x},\underline{y})]$ added to K
- « complete » semantics

  Ans(def,K) $\subseteq$ Ext(v)

  axiom : $\forall \underline{x}\ [\exists y\ def(\underline{x},\underline{y}) \Rightarrow v(\underline{x})]$ added to K

The Global-As-Views approach

Illustration on 4 existing data sources

- S1: a catalogue of teaching programs of (some) French universities

  S1.Catalogue(nomUniv, programme)
- S2: Erasmus students enrolled in courses of (some) European universities

  S2.Erasmus(student, course, univ)
- S3: Foreign students enrolled in programs of (some) French universities

  S3.CampusFrance(student, program, university)
- S4: the course content of (some) international master programs

  S4.Mundus(programTitle,course)

GAV modeling of a mediated schema

University (U) : S1.Catalogue(U,P) ∨ S2.Erasmus(N,C,U)
∨ S3.CampusFrance(N',P',U)

MasterStudent (N) : S2.Erasmus(N,C,U), S4.Mundus(P,C)
∨ S3.CampusFrance(N,P',U'),S4.Mundus(P',C')

MasterCourse (C): S4.Mundus(P,C)

MasterProgram(P): S4.Mundus(P,C)

EnrolledIn (N,P): S2.Erasmus(N,C,U), S4.Mundus(P,C)
∨ S3.CampusFrance(N,P,U'),S4.Mundus(P,C')

RegisteredTo(N,U): S3.CampusFrance(N,P,U),

Logical semantics of GAV mappings

MasterStudent (N) : S2.Erasmus(N,C,U), S4.Mundus(P,C)
∨ S3.CampusFrance(N,P',U'),S4.Mundus(P',C')

<u>Exact semantics:</u>

$\forall$N [($\exists$C$\exists$U $\exists$P (S2.Erasmus(N,C,U) $\wedge$ S4.Mundus(P,C))
$\vee$ ($\exists$C'$\exists$U' $\exists$P' (S3.CampusFrance(N,P',U'),S4.Mundus(P',C')))
$\Leftrightarrow$ MasterStudent (N)]

<u>Sound semantics:</u>

$\forall$N [($\exists$C$\exists$U $\exists$P (S2.Erasmus(N,C,U) $\wedge$ S4.Mundus(P,C))
$\vee$ ($\exists$C'$\exists$U' $\exists$P' (S3.CampusFrance(N,P',U'),S4.Mundus(P',C')))
$\Rightarrow$ MasterStudent (N)]

Query rewriting by unfolding

The two semantics express how to obtain tuples for the corresponding global relation

$\Rightarrow$The logical query plans are obtained by **unfolding** each atom of the query, i.e., by replacing each atom that can be matched with the head of atleast one view with the body of the corresponding view (possiblly splitted in **conjunctive** views)

MasterStudent (N) : S2.Erasmus(N,C,U), S4.Mundus(P,C)
MasterStudent (N) : S3.CampusFrance(N,P',U'),S4.Mundus(P',C')

Illustration

Query: **q(x): RegisteredTo(s,x), MasterStudent(s)**

Conjunctive views:

RegisteredTo(N,U): S3.CampusFrance(N,P,U)

MasterStudent (N) : S2.Erasmus(N,C,U), S4.Mundus(P,C)

MasterStudent (N) : S3.CampusFrance(N,P',U'),S4.Mundus(P',C')

2 rewritings by unfolding:

(existential variables in the view bodies are replaced by new variables)

u1(x):

S3.CampusFrance(s,v1,x), S2.Erasmus(s,v2,v3),S4.Mundus(v4,v2)

u2(x):

S3.CampusFrance(s,v5,x), S3.CampusFrance(s,v6,v7), S4.Mundus(v6,v8)

Illustration (ctd)

Simplification of u2(x):

S3.CampusFrance(s,v5,x), S3.CampusFrance(s,v6,v7), S4.Mundus(v6,v8)

by unifying the two first atoms into **S3.CampusFrance(s,v6,x)** with the substitution σ = {v5/v6, v7/x} where v5 and v7 are unbounded existential variables

$\Rightarrow$equivalent query expression

2 resulting logical query plans:

u1(x):

S3.CampusFrance(s,v1,x), S2.Erasmus(s,v2,v3),S4.Mundus(v4,v2)

u'2(x): **S3.CampusFrance(s,v6,x), S4.Mundus(v6,v8)**

Results and discussion

- The union U of the logical query plans obtained by unfolding the atoms of a query q using a set GV of GAV mappings is **complete :** for every instance I of the source relations, ans(q, GV $\cup$ I) = $\bigcup_{u \in U}$ans(u,I)
- The evaluation of some query plans may lead to redundant answers or to no answer at all
  - It can be known in advance (before their execution) if some additional knowledge is provided
  - Example: from the knowledge that the students found in S3. CampusFrance are non European Students, while those found in S2.Erasmus are European students, we can infere that the query plan u1 will return an empty set of answers

u1(x): **S3.CampusFrance(s,v1,x), S2.Erasmus(s,v2,v3),S4.Mundus(v4,v2)**

Main limitation of the GAV approach

- Adding or removing data sources requires to revise all the GAV mappings defining the global schema
  - when a new data source arrives, we must consider how it may be combined with all the existing data sources to produce tuples of any global relation

⇒ In the Local-As-Views (LAV) approach, the mediated schema is designed to remain stable even when data sources join or leave the integration system

The LAV approach

- The mediated schema is defined as a set of global relations in function of a given domain
- Example :

  Student(studentName),…, University(uniName)

  Program(title), MasterProgram(title), Course(code)

  EnrolledInProgram(studentName,title)

  EnrolledInCourse(studentName,code), PartOf(code,title)

  RegisteredTo(studentName, uniName)

  OfferedBy(title, uniName)

LAV mappings

S1.Catalogue(U,P):

FrenchUniversity(U), Program(P),
OfferedBy(P,U), OffereBy(P',U), MasterProgram(P')

S2.Erasmus(S,C,U):

Student(S), EnrolledInCourse(S,C), PartOf(C,P),
OfferedBy(P,U), EuropeanUniversity(U), RegisteredTo(S,U')
EuropeanUniversity(U'), U≠U'

S3. CampusFrance(S,P,U):

NonEuropeanStudent(S), EnrolledInProgram(S,P),
Program(P), Offeredby(P,U), FrenchUniversity(U),
RegisteredTo(S,U)

S4.Mundus(P,C):

MasterProgram(P), OfferedBy(P,U), OfferedBy(P,U'),
EuropeanUniversity(U), NonEuropeanUniversity(U),
PartOf(C,P)

Logical semantics of the LAV mappings

S1.Catalogue(U,P):

FrenchUniversity(U), Program(P),
OfferedBy(P,U), OffereBy(P',U), MasterProgram(P')

Exact semantics:

$\forall$U $\forall$P [S1.Catalogue(U,P)

$\Rightarrow \exists$P' (FrenchUniversity(U), Program(P),
OfferedBy(P,U), OffereBy(P',U), MasterProgram(P'))]

Sound semantics:

$\forall$U $\forall$P [S1.Catalogue(U,P)

$\Leftrightarrow \exists$P' (FrenchUniversity(U), Program(P),
OfferedBy(P,U), OffereBy(P',U), MasterProgram(P'))]

Discussion

- Allows a fine-grained description of the data sources, and a loose coupling between local and global relations
  - Important for robustness and flexibility
    - **Illustration:** if we are interested in Master students, we do not need to know in advance how to join the available data sources to obtain them like in the GAV approach ; we just define them as a global query

      MasterStudent(S):
      Student(S), EnrolledInProgram(S,P), MasterProgram(P)
- Price to pay flexibility and robustness: building the rewritings requires more work than the simple unfolding of the GAV approach
  - Several algorithms: Bucket, Minicon, Inverse-rules

The Bucket algorithm

- Input
  - A conjonctive query (with comparison predicates) over a global schema
  - A set of local relations defined as conjunctive views (with comparison predicates) « sound» semantics) over the global schema
- output : a set of conjunctive queries over the local relations
- Implemented in Information Manifold

A.Levy, A. Rajaraman, J.Ordille. Querying heterogeneous information sources using source descriptions. Proceedings of the Int.Conference on Very Large Data bases (VLDB 96)

Principle: two steps

- Create a « bucket» for each atom g of the query
  - Store each view atom with an atom in its definition being unifiable with g (without violating comparison predicates)
- Build the set of candidate rewritings
  - Take one view-atom in each bucket and take their conjunction
  - For each candidate rewriting, check if its expansion is contained in the query
    - If yes: return it in the output
    - If no : try to add some comparison predicates to satisfy the containment

Creation of the buckets: illustration

q(x): RegisteredTo(s,x), EnrolledInProgram(s,p), MasterProgram(p)

- **RegisteredTo(S,U')** is in the definition of **S2.Erasmus(S,C,U)** but mapping the existential variable U' in the view definition to the distinguished variable x in the query is not enough to infer **RegisteredTo(s,x)** from **S2.Erasmus(s,C,U)**
  S2.Erasmus(s,C,U) <u>is not added</u> to Bucket(RegisteredTo(s,x))
- **RegisteredTo(S,U)** in the definition of **S3.CampuFrance(S,P,U)** **has U as distinguished variable to which the distinguished variable x can be mapped**

Bucket(RegisteredTo(s,x)) = {S3.CampusFrance(s, v1,x)}

Combination of the buckets

q(x): RegisteredTo(s,x), EnrolledInProgram(s,p), MasterProgram(p)

Bucket(RegisteredTo(s,x)) = {S3.CampusFrance(s, v1,x)}
Bucket(EnrolledInProgram(s,p)) = {S3.CampusFrance(s, p,v2)}
Bucket(MasterProgram(p)) = {S1.Catalogue(v3,v4), S4.Mundus(p,v5)}

⇒ 2 **<u>candidate</u>** rewritings :

r1(x): S3.CampusFrance(s, v1,x), S3.CampusFrance(s, p,v2), S1.Catalogue(v3,v4)

r2(x): S3.CampusFrance(s, v1,x), S3.CampusFrance(s, p,v2), S4.Mundus(p,v5)

Complexity

The creation of buckets:

O(NxMxV)

N= size of the query, V= number of views, M = size of the views

⇒ N buckets containing each O(MxV) view atoms

⇒ The number of candidate rewritings : $O((MxV)^N)$

Verification of each candidate rewriting

q(x): RegisteredTo(s,x), EnrolledInProgram(s,p), MasterProgram(p)
r1(x): S3.CampusFrance(s, v1,x), S3.CampusFrance(s, p,v2), S1.Catalogue(v3,v4)

r1(x) is a **valid** rewriting
iff r1(x) together with the LAV mappings logically entail q(x)
iff the **expansion** of (r1(x)) is **contained** in q(x)

Verification by expansion and containment checking

q(x): RegisteredTo(s,x), EnrolledInProgram(s,p), MasterProgram(p)
r1(x): S3.CampusFrance(s, v1,x), S3.CampusFrance(s, p,v2), S1.Catalogue(v3,v4)

Expand(r1(x)): **NonEuropeanStudent(s), EnrolledInProgram(s,v1), Program(v1), Offeredby(v1,x), FrenchUniversity(x), RegisteredTo(s,x), EnrolledInProgram(s,p), Program(p), Offeredby(p,v2), FrenchUniversity(v2), RegisteredTo(s,v2), FrenchUniversity(v3), Program(v4), OfferedBy(v4,v3), OffereBy(v5,v3), MasterProgram(v5)**

Expand(r1(x)) is not contained in q(x) : r1 is not a valid rewriting

Minicon : optimization of Bucket

- Containment checking is avoided by a stricter verification of the atoms to add to the buckets
  - When the definition of a view V contains an atom g' such that : σ(g') = g
    - If an existential variable Y of g appears in other atoms g1, g2, …, gk of the query
    - If Y' = σ(Y) is also existential in the view definition
      - σ(V) is added to Bucket(g) only if g1, g2, …, gk are also covered by the definition of σ (V)

Illustration

V4(X) : cite(X,Y), cite(Y,X)

V5(X,Y) : sameTopic(X,Y)

V6(X,Y) : cite(X,Z) , cite(Z,Y) , sameTopic(X,Z)

Query : Q(U) : cite(U,V) , cite(V,U) , sameTopic(U,V)

<u>Bucket (cite(U,V)) ?</u>

- **V4(U) is not added** because sameTopic(U,V) is not covered by the definition of V4(U)

- **V6 ?**

σ(X)=U et σ(Z)=V

Covering of cite(V,U) by the definition of V6(U,Y) => σ(Y)=U

Covering of sameTopic(U,V) by the definition of σ (V6(X,Y)) ? yes

Bucket(cite(U,V)) = {V6(U,U)}

cover(V6(U,U)) = {cite(U,V), cite(V,U), sameTopic(U,V)}

=> r(U): V6(U,U) is a valid rewriting of q(U)

Advantages of Minicon

- The rewritings are directly obtained by taking the conjunction of the view-atoms in the « buckets » which have pairwise disjoint coverings
- Results
  - theoretical :
    - same worst-case complexity as Bucket (exponential in the size of the query)
  - experimental :
    - Scalable when there are many views

The Inverse-rules algorithm

- Principle:
  - The LAV mappings are splitted into GAV mappings (called inverse rules)

    independently of the query
    - Existential variables are replaced by **Skolem terms** in order to keep the binding of the different occurrences of existential variables
  - At query time, the rewritings are obtained by unfolding
    - **The unfolding operation is a little trickier because of the Skolem functions**

Illustration

V4(X) : cite(X,Y), cite(Y,X)

V5(X,Y) : sameTopic(X,Y)

V6(X,Y) : cite(X,Z) , cite(Z,Y) , sameTopic(X,Z)

Result of the Inverse-rules algorithm:

cite(X,f1(X)): V4(X)

cite(f1(X),X)): V4(X)

sameTopic(X,Y): V5(X,Y)

cite(X,f2(X,Y)): V6(X,Y)

cite(f2(X,Y),X)): V6(X,Y)

sameTopic(X,f2(X,Y)): V6(X,Y)

Query unfolding (illustration)

cite(X,f1(X)): V4(X)
cite(f1(X),X)): V4(X)
sameTopic(X,Y): V5(X,Y)
cite(X,f2(X,Y)): V6(X,Y)
cite(f2(X,Y),X)): V6(X,Y)
sameTopic(X,f2(X,Y)): V6(X,Y)

Q(U):

cite(U,V),cite(V,U),sameTopic(U,V)

σ={X/U, V/f1(U)}

Q'1(U):

V4(U),cite(f1(U),U),sameTopic(U,f1(U))

Q'2(U):

V4(U), V4(U),sameTopic(U,f1(U))

Q'3(U):

V4(U), V5(U,f1(U))

The evaluation of this query plan will produce no answer: there is no way to match V5(U,f1(U)) with a fact V5(a,b) in the data source

Query unfolding (illustration ctd)

cite(X,f1(X)): V4(X)
cite(f1(X),X)): V4(X)
sameTopic(X,Y): V5(X,Y)
cite(X,f2(X,Y)): V6(X,Y)
cite(f2(X,Y),X)): V6(X,Y)
sameTopic(X,f2(X,Y)): V6(X,Y)

Q(U):

cite(U,V),cite(V,U),sameTopic(U,V)

σ={X/U, V/f2(U,Y)}

Q''1(U): V6(U,Y),cite(f2(U,Y),U), sameTopic(U,f2(U,Y))

σ={X/U, Y/U}

Q''2(U):

V6(U,U),V6(U,U),sameTopic(U,f2(U,U))

Q''3(U): V6(U,U),V6(U,U), V6(U,U)

simplified in:

Q''4(U): V6(U,U)

=> a valid query plan

Summary

- When the queries and the views are (unions of) conjonctive queries over simple relational schemas, the number of (maximal) conjunctive rewritings is finite and there are several algorithms to compute them
- It is not necessary the case when constraints are added
  - to the mediated schema
  - to the views (to express constraints on their access)

DL-Lite (again)

- If the constraints on the schema are expressible in DL-Lite
  - Consistency checking of the views:
    - Saturation and translation of the NIs into boolean conjunctive queries
    - Application of MiniCon for computing the rewritings of those boolean queries into views
    - Evaluation of those rewritings against the view extensions
  - Rewriting of the query:
    - Reformulation of the query using the PI
    - Application of MiniCon for computing the rewritings of each reformulation
- The computation of all the answers is not possible when the schema constraints requires (slight) extensions
  - The instance recognition (and thus the tuple recognition problem) is NP-complete in data complexity for slight extensions of DL-Lite

References

- Querying heterogeneous information sources using source descriptions. A.Levy, A. Rajaraman, J.Ordille. Proceedings of the Int.Conference on Very Large Data bases (VLDB 96)
- Information Integration Using Logical Views, J. Ullman, Proceedings ICDT ’97
- 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
- Complexity of Answering Queries Using Materialized Views, S. Abiteboul, O. Duschka, Proceedings PODS 1998
- Theory of Answering Queries Using Views, A. Halevy, Proceedings of SIGMOD 2000
- The Use of CARIN Language and Algorithms for Information Integration: the PICSEL System, F. Goasdoué, V. Lattès, -C. Rousset. International Journal of Cooperative Systems, Vol 9, Number 4 (2000)
- Minicon: a scalable algorithm for answering queries using views. R. Pottinger, A. Halevy, VLDB Journal, Volume 10 (2-3), 2001.
- Query Rewriting and Answering under Constraints in Data Integration Systems, A. Cali, D. Lembo, and R. Rosati, Proceedings of IJCAI 2003
- 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