Ontology-based information management: languages and reasoning

Maurizio Lenzerini
DASI-lab, Data and Service Integration Laboratory
Dipartimento di Informatica e Sistemistica "A. Ruberti"
Università di Roma "La Sapienza"

Free University of Bozen-Bolzano – April 11, 2007

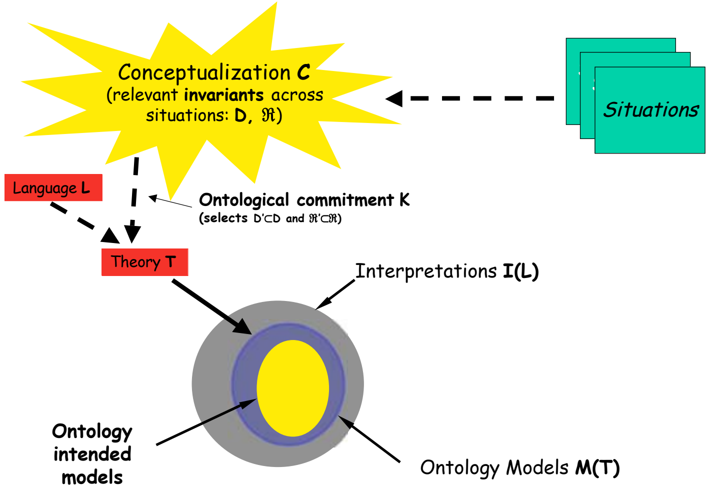
Goal and summary

- Goal
  introduction to ontologies, and in particular to their usage in designing advanced mechanisms for data (and service) access
- Summary
  1. Introduction to ontologies
  2. Languages for ontologies
  3. Automated reasoning on ontologies
  4. Ontologies for data access and integration

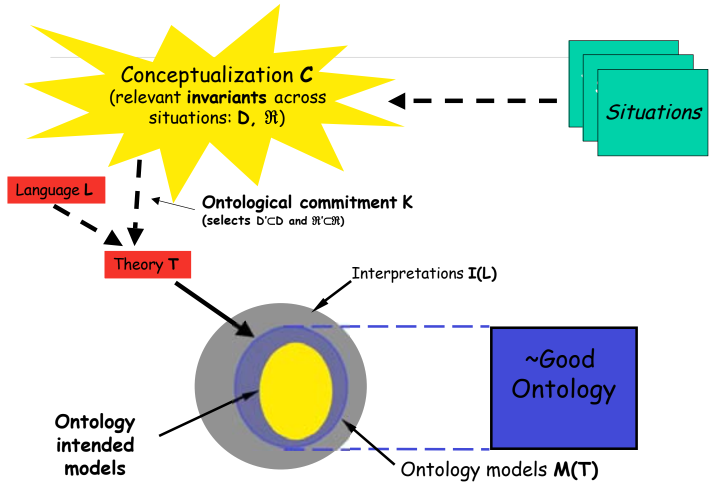
Part 1

Introduction to ontologies

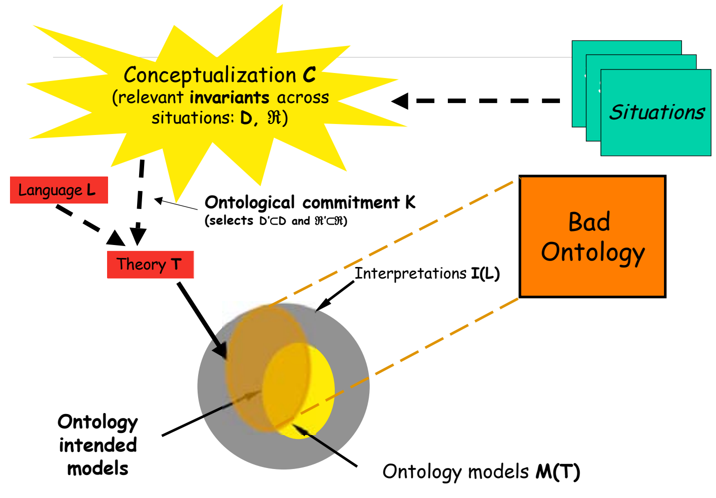
Ontologies and intended meaning (cfr. Guarino)



Ontologies and intended meaning (cfr. Guarino)



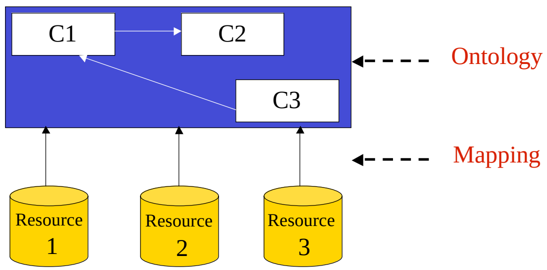
Ontologies and intended meaning (cfr. Guarino)



Ontologies: specification

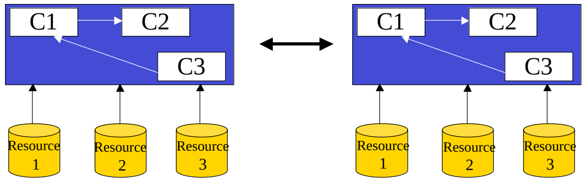
- An ontology is a representation schema describing a formal conceptualization of a domain of interest
- The specification covers three different levels:
  - Meta-level: set of categories used for modeling
  - Intensional level: set of elements (instances of categories) and rules describing the conceptual structure of the domain of interest
  - Extensional level: set of instances of elements in the intensional level, obeying the corresponding rules

Ontology: the keystone of the system



The use of all resources (data, services, etc.) is through the conceptualization of the domain

Ontology: the basis of cooperation



Cooperation is at the level of the conceptualization

Three challenges

- Languages
- Methodologies
- Tools (based on automated reasoning)

for the definition and the management of ontologies

Part 2

Languages for ontologies

Ontology languages

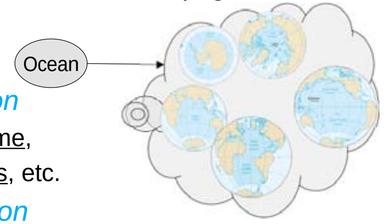
- The aspects of the domain of interest that can be modeled by an ontology language can be classified into:
  - Static (supported by virtually all languages)
  - Dynamic (supported by some languages)
- We concentrate essentially on the static aspects

Ontology languages

- An ontology language for expressing the intensional level usually includes mechanisms for describing:
  - Concepts
  - Properties of concepts
  - Relationships between concepts, and their properties
  - Axioms
  - Individuals and facts about individuals
  - Queries
- Ontologies are typically rendered as diagrams (e.g., Semantic Networks, Entity-Relationship schemas, UML class diagrams, specialized diagrams)

Concepts

- A *concept* is an element of the ontology that denotes a collection of instances (e.g., the set of "oceans")



- *Intensional definition*
  - Specification of name, properties, relations, etc.
- *Extensional definition*
  - Specification of the instances

Properties

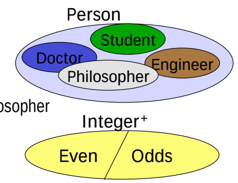
- A *property* qualifies an element (e.g., a concept) of an ontology
- Property definition (intensional and extensional)
  - Name
  - Type
    - Atomic (integer, real, string, …)
      e.g., "eye-color" → {blu, brown, green, grey}
    - Structured (date,sets,lists…)
      e.g., "date" → day/month/year

Relationships

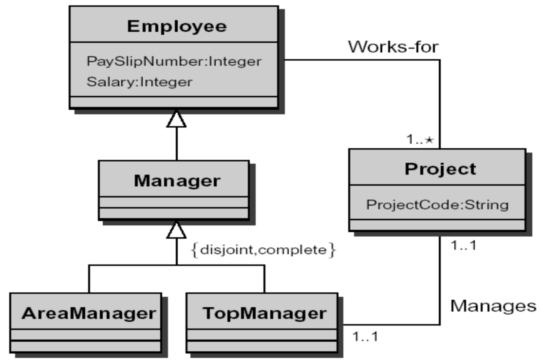
- A *relationship* expresses an association among concepts
- *Intensional definition*
  - Specification of involved concepts (example: workFor is defined on Employee and Company)
- *Extensional definition*
  - Specification of the occurrences, called facts (worksFor(Fulvio,IASI))

Axioms

- An *axiom* is a logical formula that expresses at the intensional level a condition that must be satisified by the elements at the extensional level
- Constructs in logical fomulae (example):
  - Union (and disjoint union)
    $Person \supseteq Student \cup Doctor \cup Engineer \cup Philosopher$
  - Intersection
    $Integer^{+} = Even \cup Odds$, $Even \cap Odds = \emptyset$
  - Restrictions (on cardinality)
  - Negation
  - Disjointness

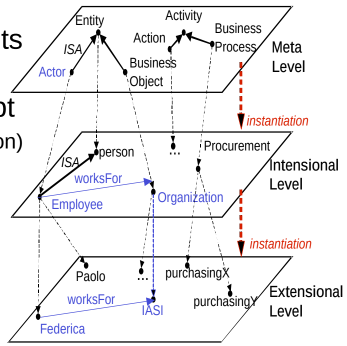


Example (thanks to Enrico Franconi)



Individuals and facts

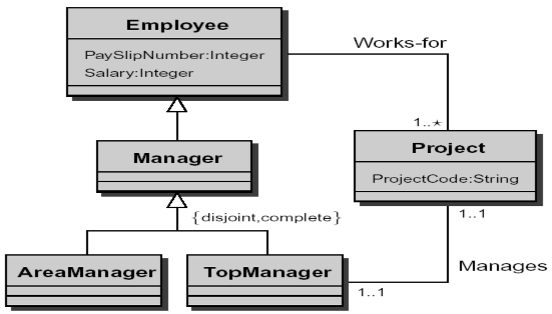
- An *instance* represents an individual in the extension of a concept
  - instance(IASI,Organization)
- A *fact* represents a relationship holding between instances
  - worksFor(Federica,IASI)



Queries

- An ontology language may also include constructs for expressing queries
  - Queries: expressions at the intensional level denoting collections of individuals satisfying a given condition
  - Meta-queries: expressions at the meta level denoting collections of elements satisfying a given condition
- The constructs for queries may be different from the constructs forming concepts and relationships

Example of query



{ (x.Salary, y.ProjectCode) | Manages(x,y) ∧ ¬Works-for(x,y) }

Some hystory (1)

- Semantic networks and frame-based systems [Brachman: What's in a Concept: Structural Foundations for Semantic Networks. International Journal of Man-Machine Studies 9(2), 1977]
- Object-oriented programming languages [SIMULA, 1967]
- Conceptual and semantic data models [Chen 1976], [J.M. Smith, D.C.P. Smith, "Database Abstractions: Aggregation and Generalization," ACM TODS 2, 1977]
- Description Logics [Brachman, H. J. Levesque: The Tractability of Subsumption in Frame-Based Description Languages. AAAI 1984]

Some hystory (2)

- 1984: The Cyc project [D. Lenat 1984]
- 1985: The KL-ONE system [Brachman et al 1985]
- 1987-1995: Formal study on expressivity and complexity of Description Logic languages [F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt: The Complexity of Concept Languages. Information and Computation 134(1), 1997]
- 1993: Formal Ontology meets Computer Science [Formal Ontology in Conceptual Analysis and Knowledge Representation, Guarino & Poli (Eds), Kluwer, 1993]
- 1994: UML [Unified Modeling Language – www.uml.org]

Some hystory (3)

- 1995-2000: Formal study on expressivity and complexity of Description Logic–based ontologies [P.h.D Theses of Giuseppe De Giacomo and Diego Calvanese - DIS-UNIROMA1]
- 2000 - : Ontology Management Systems [Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003 – www.dl.kr.org]
- 2000 - : Ontologies for the Semantic Web [http://www.w3.org/TR/owl-features/] [http://www.w3.org/TR/rdf-primer/]

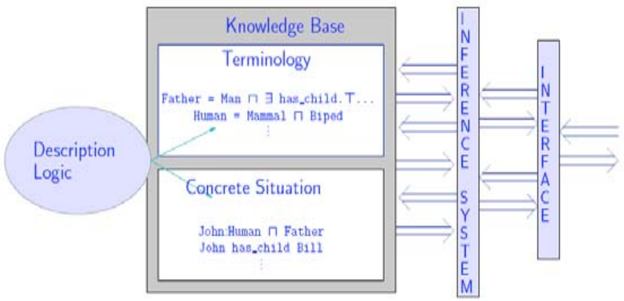
Comparison with other formalisms

- Ontology languages vs knowledge representation languages
  - Ontologies are knowledge representation schemas
- Ontology vs logic
  - Logic is a (the) tool for assigning semantics to ontology languages
- Ontology languages vs conceptual database models
  - Conceptual schema are special ontologies, suited for conceptualizing a single logical model (database)
- Ontology languages vs programming languages
  - Class definitions are special ontologies, suited for conceptualizing a single structure for computation

Classification of languages

- **Graph-based**
  - Semantic networks
  - Conceptual graphs
  - UML
- **Frame based**
  - Frame Systems
  - OKBC, XOL
- **Logic based**
  - Description Logics (e.g., DLR, DL-lite, OWL)
  - Rules (e.g., RuleML, LP/Prolog)
  - First Order Logic (e.g., KIF)
  - Non-classical logics (e.g., Nonmonotonic, probabilistic)

Description Logics



Description Logics

We start with alphabets for concepts, roles, and individuals. Syntactically, concepts and roles are either atomic (i.e., denoted by a name), or non-atomic, i.e. built out using the constructors of a given description language $\mathcal{L}$.

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of

- a nonempty set $\Delta^\mathcal{I}$, the domain of $\mathcal{I}$
- a function $\cdot^\mathcal{I}$, the interpretation function of $\mathcal{I}$, that maps
  - every individual to an element of $\Delta^\mathcal{I}$
  - every concept to a subset of $\Delta^\mathcal{I}$
  - every role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$

in such a way that suitable equations are satisfied.

Description Logics

A Description Logic - mainly characterised by a **set of constructors** that allow to build complex **concepts** and **roles** from atomic ones,

concepts correspond to classes / are interpreted as **sets** of objects,

roles correspond to relations / are interpreted as **binary relations** on objects,

Example: Happy Father in the DL $\mathcal{ALC}$

```
Man ⊓ (∃has-child.Blue) ⊓
      (∃has-child.Green) ⊓
      (∀has-child.Happy ⊔ Rich)
```


TBox e ABox

An $\mathcal{L}$-Tbox $\mathcal{T}$ is a set of statements (**inclusion assertions**) of the form:

$C \sqsubseteq D$ $R \sqsubseteq Q$

An $\mathcal{L}$-ABox Σ is a set of statements (**membership assertions**) of the forms (a, b are individuals, and we have $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$):

$C(a)$ $R(a, b)$

- $C \sqsubseteq D$ is satisfied by $\mathcal{I}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $R \sqsubseteq Q$ is satisfied by $\mathcal{I}$ if $R^{\mathcal{I}} \subseteq Q^{\mathcal{I}}$
- $C(a)$ is satisfied by $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $R(a, b)$ is satisfied by $\mathcal{I}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Knowledge base (Ontology)

An $\mathcal{L}$-knowledge base is a pair $\langle \mathcal{T}, \Sigma \rangle$, where $\mathcal{T}$ is an $\mathcal{L}$-Tbox, and Σ is an $\mathcal{L}$-ABox.

An interpretation $\mathcal{I}$ is a model of $\mathcal{K} = \langle \mathcal{T}, \Sigma \rangle$ if it satisfies all assertions of $\mathcal{T}$ and all assertions of Σ. $\mathcal{K}$ is said to be satisfiable if it admits a model.

$\mathcal{K}$ logically implies an assertion α (written $\mathcal{K} \models \alpha$) if α is satisfied by every model of $\mathcal{K}$. C is subsumed by D in $\mathcal{K}$, if $\mathcal{K} \models C \sqsubseteq D$.

open world assumption

Example

Note: $\{ C \sqsubseteq D,\ D \sqsubseteq C \}$ is written simply as $C = D$

TBox $\mathcal{T}$:

$$\exists(\text{child}^-)^*.\exists\text{live}.\text{SouthOfPo} \sqsubseteq \neg\text{RealPadano}$$

$$\text{RealPadano} = \text{Italian} \sqcap (\forall\text{child}^*.\text{RealPadano}) \sqcap (\forall\text{friend}^*.\text{RealPadano})$$

ABox Σ:

RealPadano(Umberto),
child(Umberto,Aldo),
¬ RealPadano(Gianfranco)

OWL Ontology Web Language

OWL concept constructors:

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

OWL Ontology Web Language

Types of axioms:

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	$\top \sqsubseteq \leqslant 1P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

Part 3

Automated reasoning on ontologies

Reasoning over ontologies

- Given an ontology, *additional* properties can be inferred, by
  - **Logical reasoning**
  - Meta-level querying *(typically, also based on logical reasoning)*
- Different goals of reasoning
  - Verification
  - Validation
  - Analysis
  - Querying

Types of logical reasoning

- Classification based on semantic property
  - **Classical**
  - Non-classical (e.g., non-monotonic reasoning, common-sense reasoning, etc.)
- Classification based on the type of desired conclusions
  - **Deduction**
  - Induction
  - Abduction

Logical reasoning: deduction

Let T and A be the intensional level and the extensional level of an ontology, respectively.

Deduction

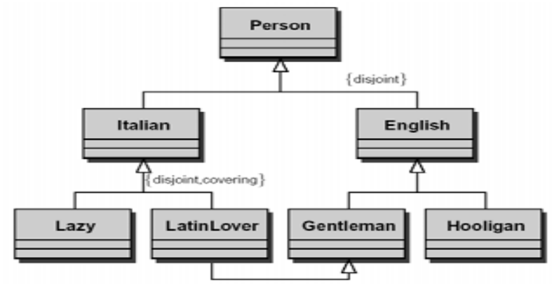
P is a deductive conclusion from T and A ($T,A \models P$) if it holds in every model of T and A, i.e., in every interpretation satisfying both T and A

Reasoning over ontologies

Examples of classical deductive logical reasoning:

- Concept consistency: A concept is consistent if it does not always (i.e.m in every model of the ontology) denote the empty set, inconsistent otherwise
- Concept subsumption: A concept is a subconcept (or, is subsumed by) of another concept if the former always denotes a subset of the set denoted by the latter
- Equivalence: Two concepts are equivalent if they always denote the same set
- Query answering: a tuple of objects is an answer to a query if it satisfies the query in every model of the ontology. Note that query answering over ontologies is different from query evaluation in database.

Example of consistency



Q: Is LatinLover consistent?

A: no!

Examples taken from Enrico Franconi

Automated reasoning

Not FOL, but DL!

Most ontology languages are designed to admit automated reasoning procedures for the basic reasoning tasks that are:

- Sound (return only correct answers)
- Complete (return all correct answers)
- Terminating (stop producing their results in finite amount of time)

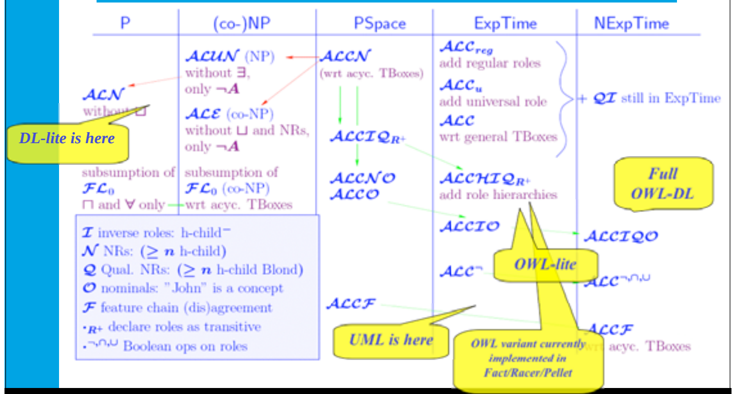
Computational complexity:
form PTIME to EXPTIME (or even NEXPTIME), depending on expressivity of the language

Automated reasoning techniques

- Tableaux: this is the most mature technique, used in systems such FACT/Racer/Pellet
- Automata on infinite tree: the most powerful technique, but not implemented
- Structural analysis: simple, but works only for the weakest languages
- Other: e.g., specialized chase-based techniques, used for example in QuOnto

In fact, of all basic logical reasoning tasks

Complexity of concept consistency



Part 4

Ontologies for data access and integration

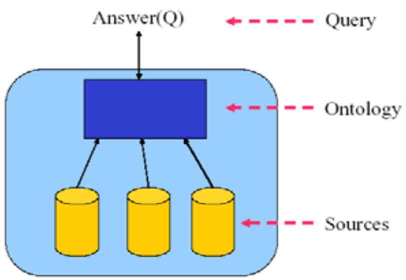
Scenarios of use of ontologies

Integration and cooperation
- One of the challenges of ICT
- Big market (7.5 billion $ – Aberdeen Group)
- Two facets:
  - Integration: intra-organization (e.g., EIS)
  - Cooperation: inter-organization
- Objects of interest
  - Data
  - Services

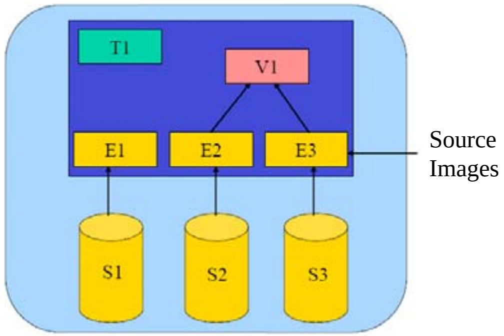
Ontologies for data access and integration
1. Ontologies for mediator-based integration
2. Ontology as conceptual tool for interoperability

Mediator-based data and service integration

- One ontology
- Mapping between the data sources and the ontology
- Queries over the ontology



Data federation (i.e., DB2 information integrator)



Tools for data federation

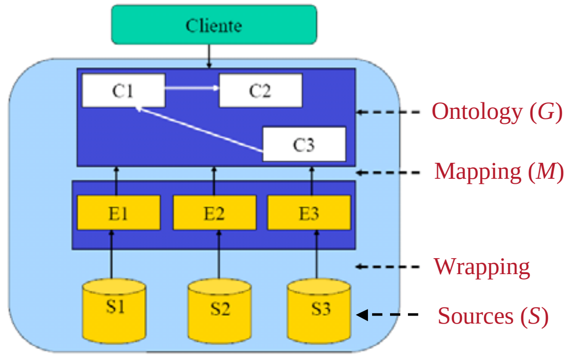
- Advantages
  - Physical transparency
  - Heterogeinity
  - Source autonomy
  - Efficient distributed query answering
- Disadvantages
  - No conceptual trasparency: the global schema is not independent from the sources

Conceptual transparency

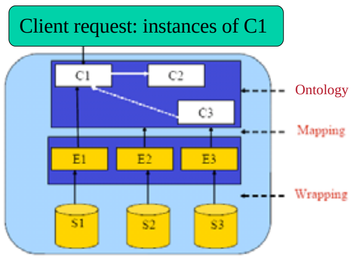
- Global schema independent from the sources
- Global schema described with a rich formalism
- Mappings are crucial for linking the sources to the ontology

- Important in several areas
  - Programming languages
  - Software engineering
  - Databases
  - Knowledge representation

Conceptual transparency



Why conceptual transparency failed



Suppose the ontology logically implies C3 isa C1. If the system is not able to compute this subsumption relationship, it will not access C3 when answering the query, thus missing the answers from C1. Reasoning on the ontology is crucial!

Mapping between sources and ontology

How is the mapping $\mathcal{M}$ between $\mathcal{S}$ and $\mathcal{G}$ specified?

- Are the sources defined in terms of the ontology?
  Approach called local-as-view, or LAV
- Are the extensions of the ontology elements defined in terms of the sources?
  Approach called global-as-view, or GAV
- A mixed approach?
  Approach called GLAV

Mapping: example

Ontology: Movie($Title, Year, Critique*$)

directs(Movie, Director)

European isa Director

Source 1: $r_1(Title, Year, Director)$ since 1960, european directors

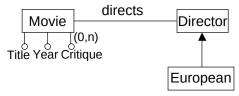
Source 2: $r_2(Title, Critique)$ since 1990

GAV mapping

Ontology: Movie($Title, Year, Critique*$)

directs(Movie, Director)

European isa Director



Movie$(id, T, Y, C*)$ $\rightsquigarrow$ $\{ (mk(T), T, Y, C) \mid r_1(T, Y, D) \wedge r_2(T, C) \}$

European(id) $\rightsquigarrow$ $\{ (mk(D)) \mid r_1(T, Y, D) \}$

directs(idM, idD) $\rightsquigarrow$ $\{ (mk(T), mk(D)) \mid r_1(T, Y, D) \}$

QuOnto: ontology-based data integration

- QuOnto is a DL reasoner jointly developed by University of Rome "La Sapienza" (G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati) and the Free University of Bozen-Bolzano (D. Calvanese)
- It allows modeling, managing, and querying a data integration system whose global schema is an ontology expressed in a Description Logic (DL-lite) resulting from the research carried out in the last 20 years
- The language for specifying the ontology is an optimal compromise between expressive power and complexity of query answering

DL-lite

- Concepts constructs:

$B ::= A \mid \exists R \mid \exists R^-$ *basic concepts*

$C ::= B \mid \neg B \mid C_1 \sqcap C_2$ *general concepts*

- TBox assertions:

$B \sqsubseteq C$ *inclusion assertions*

$(funct\ R)$ $(funct\ R^-)$ *functionality assertions*

- ABox assertions:

$B(o_1)$ $R(o_1, o_2)$ *with o_1, o_2 constants*

Constructs of DL-lite

- ISA between classes

  $A_1 \sqsubseteq A_2$

- class disjointness

  $A_1 \sqsubseteq \neg A_2$

- role typing

  $\exists R \sqsubseteq A_2$ $\exists R^- \sqsubseteq A_1$

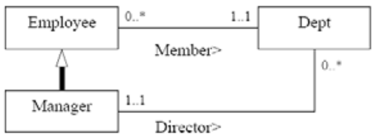
- mandatory participation

  $A_1 \sqsubseteq \exists R$ $A_2 \sqsubseteq \exists R^-$

- functionality of roles

  $(\mathit{funct}\ R)$ $(\mathit{funct}\ R^-)$

Example



$\exists Member \sqsubseteq Employee$
$\exists Member^- \sqsubseteq Dept$
$Employee \sqsubseteq \exists Member$
$(\mathit{funct}\ Member)$
$\exists Director \sqsubseteq Manager$
$\exists Director^- \sqsubseteq Dept$
$Dept \sqsubseteq \exists Director^-$
$(\mathit{funct}\ Director^-)$
$Manager \sqsubseteq Employee$

A note on mappings

- In Ontology-based integration we have to deal with the "impedence mismatch" problem
  - Sources store data, while instances of concepts and relations in the ontologies are objects
- The solution is to define a mapping language that allows specifying how to transform data into objects
- Basic idea: use "Skolem functions" in the head of the mapping
- Semantics: objects are denoted by "terms" (of exaclty one level of nesting), and different terms are different objects (unique name assumption on terms)

Mappings: example

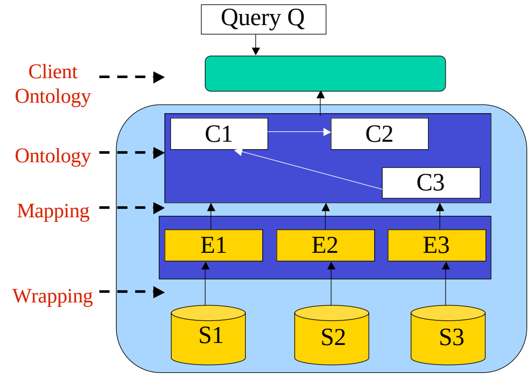
Three sources on students:
- s1 uses code for identifying students
- s3 uses number for identifying
- s2 stores (incomplete) correspondences between code and number

Student(sbc(code)) :- s1(code,dob,addr,city)
Student(sbc(code)) :- s2(number,code)
Student(sbn(number)) :- s3(number,addr,city), not s2(number,code)
LivesIn(sbc(code),c(city)) :-s1(code,dob,addr,city), city is not null
LivesIn(sbc(code),c(city3)) :- s1(code,dob,addr,city1), city1 is null,
s2(number,code), s3(number,addr,city3), city3 is not null
LivesIn(sbc(code),c(city3)) :- not s1(code,dob,addr,city1),
s2(number,code), s3(number,addr,city3), city3 is not null
LivesIn(sbn(number),c(city)) :- s3(number,address,city), city is not null,
not s2(number,code)

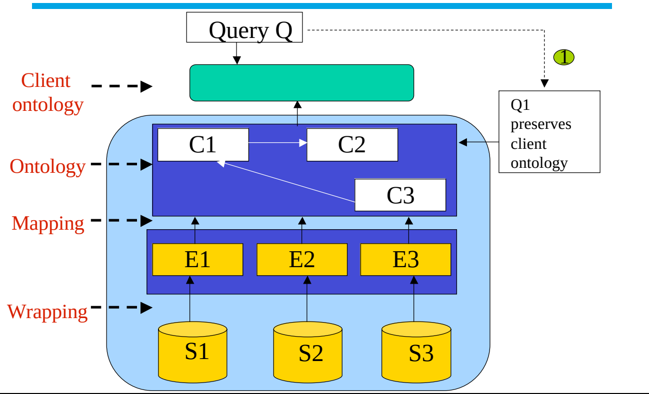
Complexity analysis

	Cl	Cr	$\mathcal{F}$	$\mathcal{R}$	Data complexity of query answering
1	*DL-Lite*$_{\mathcal{F}}$		√	–	in LOGSPACE
2	*DL-Lite*$_{\mathcal{R}}$		–	√	in LOGSPACE
3	*DLR-Lite*$_{\mathcal{F}}$		√	–	in LOGSPACE
4	*DLR-Lite*$_{\mathcal{R}}$		–	√	in LOGSPACE
5	$A \mid \exists P.A$	A	–	–	NLOGSPACE-hard
6	A	$A \mid \forall P.A$	–	–	NLOGSPACE-hard
7	A	$A \mid \exists P.A$	√	–	NLOGSPACE-hard
8	$A \mid \exists P.A \mid \exists P^{-}.A$	$A \mid \exists P$	–	–	PTIME-hard
9	A	$A \mid \exists P.A \mid \exists P^{-}.A$	√	–	PTIME-hard
10	$A \mid \exists P.A$	$A \mid \exists P.A$	√	–	PTIME-hard
11	$A \mid \exists P.A \mid A_1 \sqcap A_2$	A	–	–	PTIME-hard
12	$A \mid A_1 \sqcap A_2$	$A \mid \forall P.A$	–	–	PTIME-hard
13	$A \mid A_1 \sqcap A_2$	$A \mid \exists P.A$	√	–	PTIME-hard
14	$A \mid \neg A$	A	–	–	coNP-hard
15	A	$A \mid A_1 \sqcup A_2$	–	–	coNP-hard
16	$A \mid \forall P.A$	A	–	–	coNP-hard

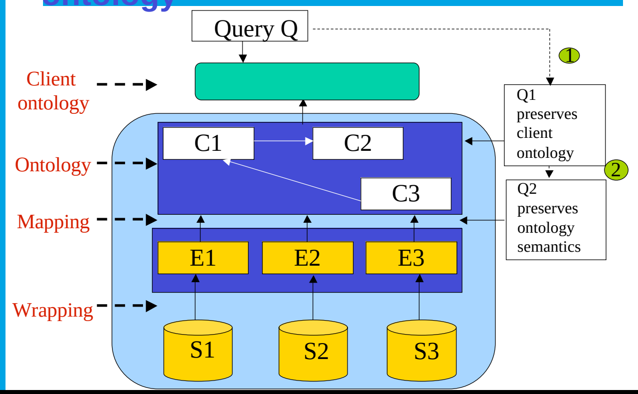
How QuOnto answers queries



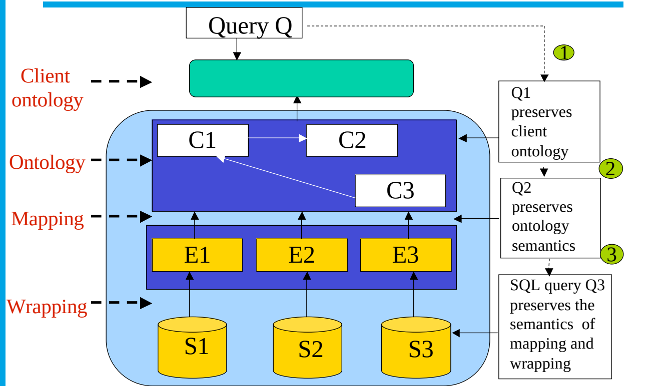
Phase 1: reformulation wrt client ontology



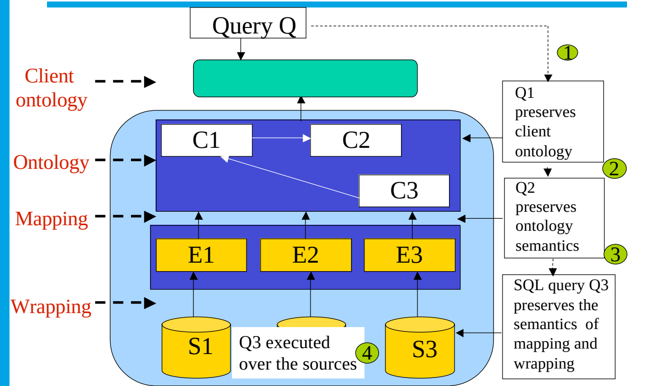
Phase 2: reformulation over the ontology



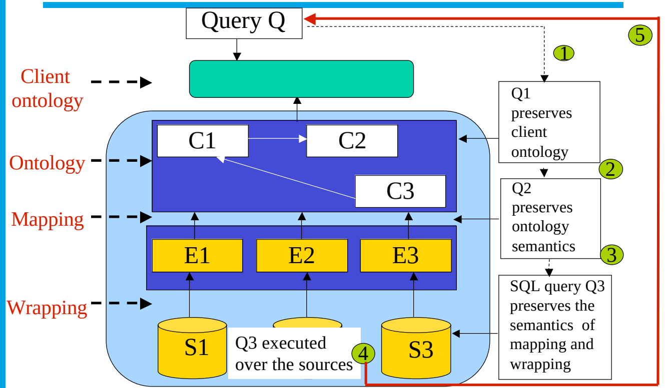
Phase 3: reformulation over the sources



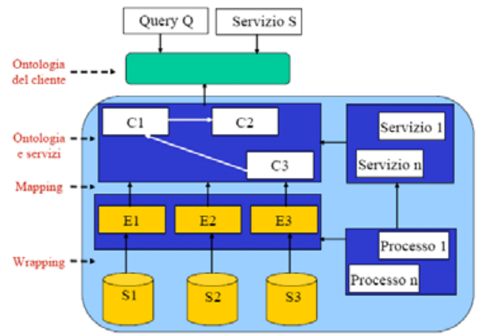
Phase 4: SQL query evaluation over the sources



Phase 5: returning the answer

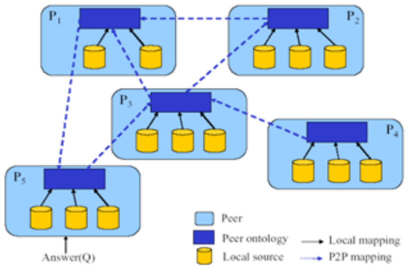


Extending QuOnto towards service integration



Ontologies for interoperability

- Several peers, each with one ontology
- P2P mappings
- Each query over one peer



Taken from
[Calvanese&al PODS04]

Some references

- Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider, "The Description Logic Handbook", Cambridge University Press, 2004
- Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, "DL-Lite: Tractable description logics for ontologies", Proc. of the 20th Nat. Conf. on Artificial Intelligence, AAAI-2005
- Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati, "Tailoring OWL for data intensive ontologies", Proc. of the Workshop on OWL: Experiences and Directions, 2005
- Pagina web su Description Logics - http://dl.kr.org/