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

Ontologies and intended meaning (cfr. Guarino)

Part 1
Introduction to ontologies
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 satisfied by the elements at the extensional level.

- Constructs in logical formulae (example):
  - Union (and disjoint union)
  - Intersection
  - Restrictions (on cardinality)
  - Negation
  - Disjointness

Example (thanks to Enrico Franconi)

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) \land \neg Works-for(x,y) \}\n
Some history (1)

- Semantic networks and frame-based systems
- Object-oriented programming languages [SIMULA, 1967]
- Description Logics [Brachman, H. J. Levesque: The Tractability of Subsumption in Frame-Based Description Languages. AAAI 1984]

Some history (2)

- 1984: The Cyc project [D. Lenat 1984]
- 1985: The KL-ONE system [Brachman et al 1985]

Some history (3)

- 2000 - : Ontologies for the Semantic Web [http://www.w3.org/TR/owl-features/]
  [http://www.w3.org/TR/rdf-primer/]

6
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 schemas 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

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

- a nonempty set $\Delta^I$, the domain of $I$
- a function $\mathcal{I}$, the interpretation function of $I$, that maps
  - every individual to an element of $\Delta^I$
  - every concept to a subset of $\Delta^I 
  - every role to a subset of $\Delta^I \times \Delta^I$
  in such a way that suitable equations are satisfied.
Description Logics

A Description Logic is mainly characterized 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 ALC

\[ \exists \text{has-child}.\exists \text{has-child} \cap \exists \text{has-child}.\text{Green} \cap \exists \text{has-child}.\text{Happy and Rich} \]

Knowledge base (Ontology)

An L

knowledge base is a pair \( \langle T, \Sigma \rangle \), where \( T \) is an L-Tbox and \( \Sigma \) is an L-ABox.

An interpretation \( I \) is a model of \( K = \langle T, \Sigma \rangle \) if it satisfies all assertions of \( T \) and all assertions of \( \Sigma \). \( K \) is said to be satisfiable if it admits a model.

\( K \) logically implies an assertion \( \alpha \) (written \( K \models \alpha \)) if \( \alpha \) is satisfied by every model of \( K \). \( C \) is subsumed by \( D \) in \( K \) if \( K \models C \subseteq D \).

Example

Note: \( C \subseteq D, D \subseteq C \) is written simply as \( C = D \)

\[ \exists \text{child} \cap \exists \text{live}, \exists \text{live, SouthOfPo} \subseteq \neg \text{RealPardo} \]

\[ \text{RealPardo} = \text{Italian} \cap (\exists \text{child} \neg \text{RealPardo}) \cap (\exists \text{friend} \neg \text{RealPardo}) \]

\[ \text{ABox} \Sigma: \]

- \( \text{RealPardo}(\text{Umberto}), \)
- \( \text{child} (\text{Umberto}, \text{Alina}), \)
- \( \neg \text{RealPardo} (\text{Gianfranco}) \)
OWL Ontology Web Language

OWL concept constructors:

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>Modal Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \cap \ldots \cap C_n$</td>
<td>Human $\cap$ Male</td>
<td>$C_1 \wedge \ldots \wedge C_n$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>Doctor $\cup$ Lawyer</td>
<td>$C_1 \vee \ldots \vee C_n$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\sim C$</td>
<td>Male</td>
<td>$\sim C$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>${x_1} \cup \ldots \cup {x_n}$</td>
<td>${\text{john}} \cup {\text{mary}}$</td>
<td>$x_1 \cup \ldots \cup x_n$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists \text{hasChild}.\text{Doctor}$</td>
<td>$(P)C$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq nP$</td>
<td>$\leq 1\text{hasChild}$</td>
<td>$(P)_n$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq nP$</td>
<td>$\geq 2\text{hasChild}$</td>
<td>$(P)_n$</td>
</tr>
</tbody>
</table>

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., 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

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

Complexity of concept consistency

In fact, of all basic logical reasoning tasks

Full OWL-DL is here

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
  - Heterogeneity
  - Source autonomy
  - Efficient distributed query answering

- Disadvantages
  - No conceptual transparency: 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

---

**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 $M$ between $S$ and $G$ 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: \( \text{Movie} \langle \text{Title, Year, Critics}\rangle \)
- \( \text{directs(Movie, Director)} \)
- \( \text{European(ISO Director)} \)

Source 1: \( r_1(\text{Title, Year, Director}) \) since 1960, european directors

Source 2: \( r_2(\text{Title, Critique}) \) since 1999

**GAV mapping**

Ontology: \( \text{Movie} \langle \text{Title, Year, Critics}\rangle \)
- \( \text{directs(Movie, Director)} \)
- \( \text{European(ISO Director)} \)

\[ \begin{align*}
\text{Movie}(id, T, Y, C) & \models \{ (mk(T), T, Y, C) \mid r_1(T, Y, D) \land r_2(T, C) \} \\
\text{European}(id) & \models \{ (mk(D)) \mid r_1(T, Y, D) \} \\
\text{directs}(idM, idD) & \models \{ (mk(T), mk(D)) \mid r_1(T, Y, D) \}
\end{align*} \]

**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:
  \[ B := A \mid \exists R^+ \mid \exists \forall C \mid A \mid B \mid \neg B \mid C \cap C \]
- TBox assertions:
  \[ B \sqsubseteq C \]
  \[ \text{funt}(R) \sqsubseteq \text{funt}(R^-) \]
- ABox assertions:
  \[ B(a_1) \sqsubseteq R(a_1, a_2) \quad \text{with } a_1, a_2 \text{ constants} \]
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 exactly one level of nesting), and different terms are different objects (unique name assumption on terms)
### Complexity analysis

<table>
<thead>
<tr>
<th>CI</th>
<th>C2</th>
<th>$F$</th>
<th>$R$</th>
<th>Data complexity of query answering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DL-LiteA</td>
<td>✓</td>
<td>✓</td>
<td>LOGSPACE</td>
</tr>
<tr>
<td>2</td>
<td>DL-LiteA</td>
<td>✓</td>
<td>✓</td>
<td>LOGSPACE</td>
</tr>
<tr>
<td>3</td>
<td>DL-LiteA</td>
<td>✓</td>
<td>✓</td>
<td>LOGSPACE</td>
</tr>
<tr>
<td>4</td>
<td>DL-LiteA</td>
<td>✓</td>
<td>✓</td>
<td>LOGSPACE</td>
</tr>
<tr>
<td>5</td>
<td>$A \cup B \rightarrow A$</td>
<td>✓</td>
<td>✓</td>
<td>LOGSPACE</td>
</tr>
<tr>
<td>6</td>
<td>$A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{NLLOGSPACE-hard}$</td>
</tr>
<tr>
<td>7</td>
<td>$A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{NLLOGSPACE-hard}$</td>
</tr>
<tr>
<td>8</td>
<td>$A \cup B \rightarrow A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{PTIME-hard}$</td>
</tr>
<tr>
<td>9</td>
<td>$A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{PTIME-hard}$</td>
</tr>
<tr>
<td>10</td>
<td>$A \cup B \rightarrow A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{PTIME-hard}$</td>
</tr>
<tr>
<td>11</td>
<td>$A \cup B \rightarrow A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{PTIME-hard}$</td>
</tr>
<tr>
<td>12</td>
<td>$A \cup B \rightarrow A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{PTIME-hard}$</td>
</tr>
<tr>
<td>13</td>
<td>$A \cup B \rightarrow A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{PTIME-hard}$</td>
</tr>
<tr>
<td>14</td>
<td>$A \cup B \rightarrow A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{coNP-hard}$</td>
</tr>
<tr>
<td>15</td>
<td>$A \cup B \rightarrow A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{coNP-hard}$</td>
</tr>
<tr>
<td>16</td>
<td>$A \cup B \rightarrow A$</td>
<td>✓</td>
<td>✓</td>
<td>$\text{coNP-hard}$</td>
</tr>
</tbody>
</table>

### How QuOnto answers queries

#### Phase 1: reformulation wrt client ontology

- **Client ontology**
- **Ontology**
- **Mapping**
- **Wrapping**

#### Phase 2: reformulation over the ontology

- **Client ontology**
- **Ontology**
- **Mapping**
- **Wrapping**
Phase 3: reformulation over the sources

- Query Q
- Client ontology
- Ontology
- Mapping
- Wrapping

C1 C2 C3
E1 E2 E3
S1 S2 S3

Q1 preserves client ontology
Q2 preserves ontology semantics
SQL query Q3 preserves the semantics of mapping and wrapping

Phase 4: SQL query evaluation over the sources

- Query Q
- Client ontology
- Ontology
- Mapping
- Wrapping

C1 C2 C3
E1 E2 E3
S1 S2 S3

Q1 preserves client ontology
Q2 preserves ontology semantics
SQL query Q3 preserves the semantics of mapping and wrapping
Q3 executed over the sources

Phase 5: returning the answer

- Query Q
- Client ontology
- Ontology
- Mapping
- Wrapping

C1 C2 C3
E1 E2 E3
S1 S2 S3

Q1 preserves client ontology
Q2 preserves ontology semantics
SQL query Q3 preserves the semantics of mapping and wrapping
Q3 executed over the sources

Extending QuOnto towards service integration
Ontologies for interoperability

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

Some references

- 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/