An Algebra of Lightweight Ontologies

Marco A. Casanova

Departamento de Informática, PUC-Rio
Topics

• Introduction
• A Formal Framework
• Operations over Ontologies
• Implementation of the Operations
• Conclusions
Introduction

• **Problem addressed**
  – How to design databases to be published on the Web
    (so that software agents can understand the data)

• **Contexts**
  – Databases in a traditional Web environment
  – Triplesets in the Linked Data Cloud
Introduction

Web interfaces expose data to (human) users

Data semantics is up to the (human) users
Introduction

Web services expose data to applications. Applications have to figure out the meaning of the data.
Introduction

Mediators help expose data to applications

Data semantics is up to the Mediator
(Digression)

• Question

What is the meaning of “meaning”?
(Digression)

• Question

What is the meaning of “meaning”?

• Answer

“Terms have the same extension when true of the same things”

(Digression)
(Digression)
(Digression)
(Digression)

Rabbit  Lapin  Coelho
(Digression)

Rabbit

Lapin

Coelho
• Question

How an application figures out the meaning of the data?
(Digression)

• Question

How an application figures out the meaning of the data?

• Suggested Answer

“By matching its own conceptual schema with the database external schema”
(Digression)

• Problem

**Schema matching is an intractable problem!**
(Digression)

- Problem
  
  Schema matching is an intractable problem!

- Suggested Solution
  
  Ontologies!
Introduction

• Mediators

Mediated Schema

Mapping

External schema

Mapping

Conceptual schema

Database

Other
Introduction

• Suggested Solution

– An external schema should be a combination of fragments of one or more domain ontologies

Bruegel, Pieter. The Tower of Babel. c 1563
Oil on oak panel. 114 x 155 cm
Kunsthistorisches Museum Wien, Vienna
Introduction

• Suggested Solution
  – An external schema should be a combination of fragments of one or more domain ontologies so that matching the application schema with the external schema should become a *non-problem* (or it will remain an intractable problem)

Bruegel, Pieter. The Tower of Babel. c 1563
Oil on panel. 60 × 74.5 cm
Museum Boijmans Van Beuningen, Rotterdam, Netherlands
Introduction

• Suggested Solution
  – An external schema should be a combination of fragments of one or more domain ontologies so that matching the application schema with the external schema should become a non-problem (or it will remain an intractable problem)

  – “Standards for everything”
    • Domain ontologies
    • Object Ids
    • ...

van Valckenborch, Lucas. The Tower of Babel. 1568
Oil on panel. 41 × 56 cm
Galerie de Jonckheere, Paris, France
Introduction

• Problem
  – What is the meaning of “a combination of fragments of one or more domain ontologies”?
    (So that matching the application schema with the external schema becomes a non-problem)

van Valckenborch I, Marten. The Building of the Tower of Babel. 1595

Topics

• Introduction

• A Formal Framework
  – A Simple Example
  – Constraints
  – Constraint Graphs

• Operations over Ontologies

• Implementation of the Operations

• Conclusions
A Simple Example
A Formal Framework

• **Example:**
  – Domain ontology:
    • The Music Ontology
  – External schema:
    • a fragment of the Music Ontology
    • includes solo artists, music groups and (record) labels
A Simple Example
A Formal Framework

![Diagram showing a formal framework with classes and relationships including owl:disjointWith, mo:member_of, mo:MusicArtist, mo:SoloMusicArtist, mo:MusicGroup, mo:Label, and foaf:Agent, foaf:Person, foaf:Organization, and mo:CorporateBody classes.](image-url)
A Simple Example
A Formal Framework
A Simple Example
A Formal Framework

mo:SoloMusicArtist  mo:MusicGroup  mo:Label

Mapping  “Triplification”
A Simple Example
A Formal Framework

(uri1, mo:SoloMusicArtist, “Janis Joplin”)
(uri2, mo:MusicGroup, “Big Brother and the Holding Company”)
(uri3, mo:Label, “Columbia”)
Done!
A Simple Example
A Formal Framework
A Simple Example
A Formal Framework

Diagram:
- `foaf:Agent`
- `mo:Member_of`
- `mo:MusicArtist`
- `mo:SoloMusicArtist`
- `mo:MusicGroup`
- `mo:CorporateBody`
- `mo:Label`

Relationships:
- `owl:disjointWith`

31
What is the point here?

- the domain ontology constraints play an essential role when designing the external schema
  - they carry the semantics of the terms of the domain ontology vocabulary
  - the design process cannot be reduced to merely copying the constraints from the domain ontology to the external schema
  - the design process must take into account constraints derived from those of the domain ontology
(Intermezzo)

• Design process:
  – Designer *constructs* the vocabulary of the external schema by selecting classes and properties from the vocabulary of the domain ontology
  – “Black Box” *constructs* the constraints of the external schema as the set of all constraints that are derivable from the constraints of the domain ontology and that apply to the vocabulary of the external schema
## Constraints

### A Formal Framework

- **Lightweight constraints**

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Formalization</th>
<th>Unabbreviated form</th>
<th>Informal semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Constraint</td>
<td>$\exists P \subseteq D$</td>
<td>$(\geq 1 \ P) \subseteq D$</td>
<td>property $P$ has class $D$ as domain, that is, if $(a,b)$ is a pair in $P$, then $a$ is an individual in $D$</td>
</tr>
<tr>
<td>Range Constraint</td>
<td>$\exists P^- \subseteq R$</td>
<td>$(\geq 1 \ P^-) \subseteq R$</td>
<td>property $P$ has class $R$ as range, that is, if $(a,b)$ is a pair in $P$, then $b$ is an individual in $R$</td>
</tr>
<tr>
<td>minCardinality Constraint</td>
<td>$C \subseteq (\geq k \ P)$ or $C \subseteq (\geq k \ P^-)$</td>
<td></td>
<td>property $P$ or its inverse $P^-$ maps each individual in class $C$ to at least $k$ distinct individuals</td>
</tr>
<tr>
<td>maxCardinality Constraint</td>
<td>$C \subseteq (\leq k \ P)$ or $C \subseteq (\leq k \ P^-)$</td>
<td>$C \subseteq \neg(\geq k+1 \ P)$ or $C \subseteq \neg(\geq k+1 \ P^-)$</td>
<td>property $P$ or its inverse $P^-$ maps each individual in class $C$ to at most $k$ distinct individuals</td>
</tr>
<tr>
<td>Subset Constraint</td>
<td>$E \subseteq F$</td>
<td></td>
<td>each individual in $E$ is also in $F$, that is, class $E$ denotes a subset of class $F$</td>
</tr>
<tr>
<td>Disjointness Constraint</td>
<td>$E \mid F$</td>
<td>$E \subseteq \neg F$</td>
<td>no individual is in both $E$ and $F$, that is, classes $E$ and $F$ are disjoint</td>
</tr>
</tbody>
</table>
A \cap B = \emptyset \iff A \subseteq \neg B \iff B \subseteq \neg A
A Decision Procedure
A Formal Framework

• The IMPLIES procedure
  – A sound and complete procedure to test logical implication for lightweight constraints

```
IMPLIES(Σ, e ⊆ f)
input: a set Σ of unabbreviated inclusions and an unabbreviated inclusion e ⊆ f
output: “YES - Σ logically implies e ⊆ f”
         “NO - Σ does not logically imply e ⊆ f”
begin
  Construct G(Σ, {e,f}), the constraint graph for Σ and {e,f};
  if the node of G(Σ, {e,f}) labeled with e is a ⊥-node, or
    the node of G(Σ, {e,f}) labeled with f is a ⊤-node, or
    there is a path in G(Σ,{e,f}) from the node labeled with e
    to the node labeled with f,
  then return “YES - Σ logically implies e ⊆ f”;
  else return “NO - Σ does not logically imply e ⊆ f”;
end
```
A Decision Procedure
A Formal Framework

\[ A \cap B = \emptyset \iff A \subseteq \neg B \iff B \subseteq \neg A \]
A Decision Procedure
A Formal Framework

- **Constraint graphs**
  - inspired on a 2-SAT solver

- **Completeness proof**
  - Constructs a Herbrand model
    - Use constants to represent classes
    - Use function symbols to represent number restrictions
  - Depends heavily on the fact that the left-hand side of a lightweight inclusion is a positive expression
  - *(Seems to carry on to DL-lite core with arbitrary number restrictions)*
Topics

- Introduction
- A Formal Framework
- Operations over Ontologies
  - Definition of the operations
  - Computing the operations
  - Optimization
- Implementation of the Operations
- Conclusions
Operations over Ontologies

• Questions
  – How to design an external schema?
  – How to compare two ontologies?
  – How to design a mediated ontology?

• Answer
  – Operations over ontologies

van Valckenborch I, Marten. The Building of the Tower of Babel. 1595

Operations over Ontologies

• Operations

  – Create new ontologies, including their constraints, out of other ontologies

  – Treat an ontology $O=(V, \Sigma)$ as a theory, i.e., a set of constraints $\tau[\Sigma]$
Operations over Ontologies

• Useful operations:
  – *Projection*
    • The *projection* of $O_1 = (V_1, \Sigma_1)$ over $W$, denoted $\pi[W](O_1)$, returns the ontology $O_P = (V_P, \Sigma_P)$, where $V_P = W$ and $\Sigma_P$ is the set of constraints in $\tau[\Sigma_1]$ that use only classes and properties in $W$

  – *Deprecation*
    • The *deprecation* of $\Psi$ from $O_1 = (V_1, \Sigma_1)$, denoted $\delta[\Psi](O_1)$, returns the ontology $O_D = (V_D, \Sigma_D)$, where $V_D = V_1$ and $\Sigma_D = \Sigma_1 - \Psi$
Operations over Ontologies

- **Union**
  - The *union* of \( O_1 = (V_1, \Sigma_1) \) and \( O_2 = (V_2, \Sigma_2) \), denoted \( O_1 \cup O_2 \), returns the ontology \( O_U = (V_U, \Sigma_U) \), where \( V_U = V_1 \cup V_2 \) and \( \Sigma_U = \Sigma_1 \cup \Sigma_2 \)

- **Intersection**
  - The *intersection* of \( O_1 = (V_1, \Sigma_1) \) and \( O_2 = (V_2, \Sigma_2) \), denoted \( O_1 \cap O_2 \), returns the ontology \( O_N = (V_N, \Sigma_N) \), where \( V_N = V_1 \cap V_2 \) and \( \Sigma_N = \tau[\Sigma_1] \cap \tau[\Sigma_2] \)

- **Difference**
  - The *difference* of \( O_1 = (V_1, \Sigma_1) \) and \( O_2 = (V_2, \Sigma_2) \), denoted \( O_1 - O_2 \), returns the ontology \( O_F = (V_F, \Sigma_F) \), where \( V_F = V_1 \) and \( \Sigma_F = \tau[\Sigma_1] - \tau[\Sigma_2] \)
# Operations over Ontologies

<table>
<thead>
<tr>
<th>Question</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to design an external schema</td>
<td>Projection, Union, Deprecation</td>
</tr>
<tr>
<td>How to compare two ontologies</td>
<td>Intersection</td>
</tr>
<tr>
<td>How to design a mediated ontology</td>
<td>Intersection</td>
</tr>
</tbody>
</table>
Operations over Ontologies

- Example: a fragment of the Music Ontology
\[ O = \pi[W](MO) \quad \text{where} \quad W = \{ \text{mo:SoloMusicArtist, mo:MusicGroup, mo:Label} \} \]

\[ O = (V,S) \quad \text{where} \quad V = W \quad \text{and} \quad S = \{ \text{mo:SoloMusicArtist} \sqsubseteq \neg \text{mo:Label} \} \]
$O_1 = \pi[W](FOAF)$ where $W = \{ \text{foaf:person}, \text{foaf:Agent}, \text{foaf:Organization} \}$
$O_2 = (V_2, S_2)$, where $V_2 = \{ \text{mo:Group, mo:MusicArtist, mo:CorporateBody, mo:SoloMusicArtist, mo:MusicGroup, mo:Label} \}$

$S_2 = \{ \text{mo:SoloMusicArtist} \sqsubseteq \text{mo:MusicArtist, mo:MusicGroup} \sqsubseteq \text{mo:MusicArtist, \ldots} \}$

$O_1 = \pi[W](\text{FOAF})$ where $W = \{ \text{foaf:person, foaf:Agent, foaf:Organization} \}$
\[ O_3 = \sigma[F](\pi[W](FOAF) \cup O_2), \text{ where } F = \{\text{mo:SoloMusicArtist} \sqsubseteq \text{foaf:person}, \text{mo:MusicArtist} \sqsubseteq \text{foaf:Agent}, \text{mo:Group} \sqsubseteq \text{foaf:Agent}, \text{mo:CorporateBody} \sqsubseteq \text{foaf:Organization}, \exists \text{mo:member_of} \sqsubseteq \text{foaf:person}, \exists \text{mo:member_of} \sqsubseteq \text{mo:Group} \} \]

\[ O_2 = (V_2, S_2), \text{ where } V_2 = \{ \text{mo:Group}, \text{mo:MusicArtist}, \text{mo:CorporateBody}, \text{mo:SoloMusicArtist}, \text{mo:MusicGroup}, \text{mo:Label} \} \]
\[ S_2 = \{\text{mo:SoloMusicArtist} \sqsubseteq \text{mo:MusicArtist}, \text{mo:MusicGroup} \sqsubseteq \text{mo:MusicArtist}, \ldots \} \]

\[ O_1 = \pi[W](FOAF) \text{ where } W = \{ \text{foaf:person}, \text{foaf:Agent}, \text{foaf:Organization} \} \]
\[ O_3 = \sigma[F](\pi[W](FOAF) \cup O_2) \]

\[ W = \{ \text{foaf:person, foaf:Agent, foaf:Organization} \} \]

\[ O_2 = (V_2, S_2), \text{ where} \]
\[ V_2 = \{ \text{mo:Group, mo:MusicArtist, mo:CorporateBody, mo:SoloMusicArtist, mo:MusicGroup, mo:Label} \} \]
\[ S_2 = \{ \text{mo:SoloMusicArtist} \sqsubseteq \text{mo:MusicArtist}, \text{mo:MusicGroup} \sqsubseteq \text{mo:MusicArtist}, \ldots \} \]

\[ F = \{ \text{mo:SoloMusicArtist} \sqsubseteq \text{foaf:person}, \text{mo:MusicArtist} \sqsubseteq \text{foaf:Agent}, \text{mo:Group} \sqsubseteq \text{foaf:Agent}, \text{mo:CorporateBody} \sqsubseteq \text{foaf:Organization}, \exists \text{mo:member_of} \sqsubseteq \text{foaf:person}, \exists \text{mo:member_of} - \sqsubseteq \text{mo:Group} \} \]
Operations over Ontologies

• Computing the operations
  – Union
    • (must check if the new set of constraints implies $e \sqsubseteq \perp$, for some $e$)
  – Projection, Intersection
    • implemented as variants of IMPLIES
    • use the transitive closure of the constraint graphs
  – Difference
    • (To be further investigated)
Operations over Ontologies

• Projection

Input: an ontology $O_1 = (V_1, S_1)$ and a vocabulary $W \subseteq V_1$

Output: an ontology $O_P = (W, S_P)$, where

1. Generate $G(S_1)$, the constraint graph of $S_1$.
2. Compute the transitive closure $G^*(S_1)$ of $G(S_1)$.
3. Mark all nodes of $G^*(S_1)$ that are labeled with expressions that use only symbols in $W$.
4. Generate a set of constraints $S_P$ that correspond to:
   a) Arcs of $G^*(S_1)$ connecting marked nodes; and
   b) Expressions (in $W$) that label the same marked node.
Operations over Ontologies

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Informal specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>(≥1 foaf:name) ⊆ foaf:Person</td>
<td>The domain of foaf:name is foaf:Person</td>
</tr>
<tr>
<td>(≥1 foaf:name⁻) ⊆ xsd:string</td>
<td>The range of foaf:name is xsd:string</td>
</tr>
<tr>
<td>(≥1 mo:member_of) ⊆ foaf:Person</td>
<td>The domain of mo:member_of is foaf:Person</td>
</tr>
<tr>
<td>(≥1 mo:member_of⁻) ⊆ foaf:Group</td>
<td>The range of mo:member_of is foaf:Group</td>
</tr>
<tr>
<td>mo:MusicArtist ⊆ foaf:Agent</td>
<td>mo:MusicArtist is a subset of foaf:Agent</td>
</tr>
<tr>
<td>foaf:Group ⊆ foaf:Agent</td>
<td>foaf:Group is a subset of foaf:Agent</td>
</tr>
<tr>
<td>foaf:Organization ⊆ foaf:Agent</td>
<td>foaf:Organization is a subset of foaf:Agent</td>
</tr>
<tr>
<td>mo:SoloMusicArtist ⊆ foaf:Person</td>
<td>mo:SoloMusicArtist is a subset of foaf:Person</td>
</tr>
<tr>
<td>mo:SoloMusicArtist ⊆ mo:MusicArtist</td>
<td>mo:SoloMusicArtist is a subset of mo:MusicArtist</td>
</tr>
<tr>
<td>mo:MusicGroup ⊆ mo:MusicArtist</td>
<td>mo:MusicGroup is a subset of mo:MusicArtist</td>
</tr>
<tr>
<td>mo:MusicGroup ⊆ foaf:Group</td>
<td>mo:MusicGroup is a subset of foaf:Group</td>
</tr>
<tr>
<td>mo:CorporateBody ⊆ foaf:Organization</td>
<td>mo:CorporateBody is a subset of foaf:Organization</td>
</tr>
<tr>
<td>mo:Label ⊆ mo:CorporateBody</td>
<td>mo:Label is a subset of mo:CorporateBody</td>
</tr>
<tr>
<td>foaf:Person ⊆ ¬ foaf:Organization</td>
<td>foaf:Person and foaf:Organization are disjoint</td>
</tr>
</tbody>
</table>
Implementation of the Operations

\[ A \cap B = \emptyset \quad \text{iff} \quad A \subseteq \neg B \quad \text{iff} \quad B \subseteq \neg A \]
Operations over Ontologies

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Informal specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>mo:Label $\subseteq \neg(\geq 1 \text{ foaf:name})$</td>
<td>$G(\Sigma_{APO})$ has a path from the node labeled with mo:Label to the node labeled with $\neg(\geq 1 \text{ foaf:name})$ (which indicates that a label has no name)</td>
</tr>
<tr>
<td>$(\geq 1 \text{ foaf:name}^-) \subseteq \text{xsd:string}$</td>
<td>The range of foaf:name is xsd:string</td>
</tr>
<tr>
<td>mo:SoloMusicArtist $\subseteq$ mo:MusicArtist</td>
<td>mo:SoloMusicArtist is a subset of mo:MusicArtist</td>
</tr>
<tr>
<td>mo:MusicGroup $\subseteq$ mo:MusicArtist</td>
<td>mo:MusicGroup is a subset of mo:MusicArtist</td>
</tr>
<tr>
<td>mo:SoloMusicArtist $\subseteq \neg$mo:Label</td>
<td>mo:SoloMusicArtist and mo:Label are disjoint</td>
</tr>
</tbody>
</table>
Operations over Ontologies

• Optimization
  – Problem:
    • The transitive closure $G^*(S_1)$ contains redundancies!
  – Solution:
    • The problem is equivalent to finding the minimum equivalent graph (MEG) of a graph $G$, defined as the graph $G'$ with the minimum set of edges such that the transitive closure of $G$ and $G'$ are equal
    • Finding the minimum equivalent graph has a polynomial solution for acyclic graphs and is NP-hard for strongly connected graphs
Topics

• Introduction
• A Formal Framework
• Operations over Ontologies
• Implementation of the Operations
• Conclusions
Implementation of the Operations

• **OntologyManagerTab**
  – Implemented as a Protégé Plugin
  – Works with (restricted) OWL ontologies
  – Offers a friendly user interface

• **Examples:**
  – A projection of the FOAF Ontology
  – The intersection of FOAF and the Music Ontology
Welcome to the Ontology Manager Tab.
Developed by Romulo de Carvalho Magalhaes

Loading: /Users/romulo/Ontologias/foafFull.rdf
Ontology successfully loaded as Ontology 1.
Welcome to the Ontology Manager Tab!
Developed by Romulo de Carvalho Magalhães

Loading : /Users/romulo/Ontologies/sofull.rdf
Ontology successfully loaded as Ontology 1
Loading : /Users/romulo/Ontologies/musicontology.owl
Ontology successfully loaded as Ontology 2
Running Intersection over : /Users/romulo/Ontologies/sofull.rdf and /Users/romulo/Ontologies/musicontology.owl
Intersection done
Topics

• Introduction
• A Formal Framework
• Operations over Ontologies
• Implementation of the Operations
• Conclusions
Conclusions

• Question…

  – How to design databases to be published on the Web
    (so that software agents can understand the data)?
Conclusions

- **Suggested answer…**
  
  An external schema should be a combination of fragments of one or more domain ontologies so that matching the application schema with the external schema should become a *non-problem* (or it will remain an intractable problem)
Conclusions

• New question…

– What is the meaning of “a combination of fragments of one or more domain ontologies”?

(So that matching the application schema with the external schema becomes a non-problem)
Conclusions

• Suggested answer…

  – Operations over ontologies that create new ontologies, including their constraints, out of other ontologies
Conclusions

• Implementation of the operations

  – Based on a structural proof procedure, i.e.,
    a procedure that explores the structure of sets of constraints

  – Works for lightweight constraints
Conclusions

• Complexity
  – “Easy”,
    when the domain ontology contains
    class hierarchies and cardinality constraints (and keys)
  – “Difficult, but still possible”,
    when the domain ontology contains
    class hierarchies, property hierarchies and
    cardinality constraints (and well-behaved keys)
  – “Difficult, but still possible – under certain assumptions”,
    when, in addition,
    property domains and ranges are defined by unions
Acknowledgements

• INCT for Web Science
  – UFC: Vânia Vidal, José Antonio Macedo, Ângela Pinheiro
  – PUC-Rio: Antônio Furtado, Rômulo Magalhães
Thank You!