An Algebra of Lightweight Ontologies: Implementation and Applications

Marco A. Casanova

Departamento de Informática, PUC-Rio

NII Shonan Meeting: “Implicit and explicit semantics integration in proof based developments of discrete systems”
Shonan Village Center (SVC), Japan – November 22-25, 2016
Topics

- Introduction
- A Formal Framework
- Operations over Ontologies
- Implementation of the Operations
- Applications
- Conclusions
Introduction

• Question
  – How to design an export ontology to publish data on the Web so that an application *can understand the data*?
Introduction

• **Question rephrased**
  – How to design an export ontology (and an application ontology) so that *matching* the application ontology with the export ontology becomes *trivial*?
Introduction

• Suggested Answer

  – An export ontology
    (and an application ontology)
    should be "a combination of fragments
    of one or more well-known
    domain ontologies"

    so that matching the application ontology
    with the export ontology becomes trivial

  – “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

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

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

Introduction

• **Others Questions**
  – How to compare two ontologies?
  – How to compare two versions of the same ontology?
  – How to design a mediated ontology?
Topics

• Introduction
• A Formal Framework
• Operations over Ontologies
• Implementation of the Operations
• Applications
• Conclusions
# 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>$C \subseteq (\geq k P) \text{ or } C \subseteq (\geq k P^-)$</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>$C \subseteq (\leq k P) \text{ or } C \subseteq -(\leq k+1 P) \text{ or } C \subseteq -(\leq k+1 P^-)$</td>
<td></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>$E \upharpoonright F$</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 Formal Framework
A Decision Procedure

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

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

• **Completeness proof**
  – Constructs a Herbrand model
    • Uses constants to represent classes
    • Uses function symbols to represent number restrictions
  – Depends heavily on the fact that the left-hand side of a lightweight inclusion is a positive expression
Topics

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

• **Computing the operations**
  – Union
    • (must check if the new set of constraints implies $e \sqsubseteq \bot$, 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 Lightweight 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

$S_P$ is a set of constraints tautologically equivalent to the set of constraints in $\tau[S_1]$ that use only symbols in $W$.

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 Lightweight 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
• Applications
• Conclusions
Implementation of the Operations

• OntologyManagerTab
  – Implemented as a Protégé Plugin
  – Works with lightweight ontologies
  – Offers a friendly user interface

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

Loading: /Users/romulo/Ontologias/foafFull.rdf
Ontology successfully loaded as Ontology 1
Loading: /Users/romulo/Ontologias/musiconontology.owl
Ontology successfully loaded as Ontology 2
Running Intersection over /Users/romulo/Ontologias/foafFull.rdf and /Users/romulo/Ontologias/musiconontology.owl
Intersection done!
Topics

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

<table>
<thead>
<tr>
<th>Question</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to design an export ontology?</td>
<td>Projection, Union, Deprecation</td>
</tr>
<tr>
<td>How to compare two ontologies?</td>
<td>Intersection</td>
</tr>
<tr>
<td></td>
<td>Difference</td>
</tr>
<tr>
<td>How to design a mediated ontology?</td>
<td>Intersection</td>
</tr>
</tbody>
</table>
Applications
Design of an Export Ontology

• Example:
  – Goal:
    • Create a new ontology about music artists, solo artists, music groups and (record) labels
  – Strategy:
    • Design the new ontology as a projection of the Music Ontology
Applications
Design of an Export Ontology
Applications
Design of an Export Ontology

Diagram showing relationships between foaf:Person, foaf:Agent, mo:Group, mo:Organization, mo:MusicArtist, mo:SoloMusicArtist, mo:MusicGroup, and mo:Label.
Applications
Design of an Export Ontology

![Ontology Diagram]

- `foaf:Person` is disjointWith `foaf:Agent`.
- `mo:MusicArtist` memberOf `mo:Group`.
- `mo:Group` memberOf `mo:Organization`.
- `mo:SoloMusicArtist` and `mo:MusicGroup` are subclasses of `mo:MusicArtist`.
- `mo:CorporateBody` memberOf `mo:Label`.
Applications
Design of an Export Ontology
### Applications

Design of an Export Ontology

<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>
$A \cap B = \emptyset$ 
iff 
$A \subseteq \neg B$ 
iff 
$B \subseteq \neg A$

Applications
Design of an Export Ontology

\[ (\geq 1 \text{mo:member_of}) \]
\[ (\geq 1 \text{foaf:name}) \]
\[ \neg \text{mo:SoloMusicArtist} \]
\[ \neg \text{foaf:Person} \]
\[ \neg \text{mo:MusicArtist} \]
\[ \neg \text{foaf:Agent} \]
\[ \neg \text{mo:MusicGroup} \]
\[ \neg \text{foaf:Organization} \]
\[ \neg \text{foaf:Group} \]
\[ \neg \text{mo:CorporateBody} \]
\[ \neg \text{mo:Label} \]
\[ \neg (\geq 1 \text{foaf:name}) \]
\[ \neg \text{xsd:string} \]
\[ \text{mo:Label} \]
\[ \text{xsd:string} \]
\[ (\geq 1 \text{mo:member_of}) \]
\[ (\geq 1 \text{foaf:name}) \]
### Applications
**Design of an Export Ontology**

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

• Example:
  – Reconstruction of a fragment of the Music Ontology
The diagram illustrates a network of classes and properties related to music entities, specifically focusing on the disjoint relationship between `foaf:Person` and `foaf:Agent`. The diagram includes:

- **foaf:Person**
- **mo:MusicArtist**
- **mo:SoloMusicArtist**
- **mo:MusicGroup**
- **mo:Group**
- **foaf:Agent**
- **mo:Organization**
- **mo:CorporateBody**
- **mo:Label**

The relationships are indicated by arrows, with specific properties such as `owl:disjointWith` and `mo:member_of`.
\[ O_1 = \pi[W](\text{FOAF}) \text{ where } W = \{ \text{foaf:person, foaf:Agent, foaf:Organization} \} \]
\[ O_2 = (V_2, S_2), \text{ where } V_2 = \{ \text{mo:Group, mo:MusicArtist, mo:CorporateBody, mo:SoloMusicArtist, mo:MusicGroup, mo:Label} \} \]
\[ S_2 = \{ \text{mo:SoloMusicArtist} \sqsubseteq \text{mo:MusicArtist}, \text{mo:MusicGroup} \sqsubseteq \text{mo:MusicArtist}, \ldots \} \]
\[ O_1 = \pi[W](\text{FOAF}) \text{ where } W = \{ \text{foaf:person, foaf:Agent, foaf:Organization} \} \]
\[ O_3 = \sigma[F](\pi[W](\text{FOAF}) \cup O_2) \]

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

\[ O_2 = (V_2, S_2), \text{ where} \]

\[ V_2 = \{ \text{mo:Group, mo:MusicArtist, mo:CorporateBody, mo:SoloMusicArtist, mo:MusicGroup, mo:Label} \} \]

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

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

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

• Ontology Design
  – Design of an export ontology
    • a combination of fragments of one or more domain ontologies
      (So that matching the application ontology with the export ontology becomes trivial)

  – Operations over ontologies
    • create new ontologies, including their constraints, out of other ontologies
    • Implementation based on a structural proof procedure that works for lightweight constraints
Conclusions

• **Further Work**
  – Data Mashups as Default Theories
    • Maximally consistent data mashup $\approx$
      extension of the Default Theory
    • Computation of maximally consistent data mashups based on
      the *structural proof procedure, extended to assertions*
Example

Set $\Sigma$ of constraints:

$\sigma_1$: $(\geq 1 \ p) \subseteq D$

$\sigma_2$: $C \subseteq (\geq 1 \ p)$

$\sigma_3$: $C \subseteq \neg(\geq 2 \ p)$

$\sigma_4$: $A \subseteq C$

$\sigma_5$: $B \subseteq C$

$\sigma_6$: $A \subseteq \neg B$

Set $\Delta$ of simple defaults:

$\delta_1$. " $B(a) / B(a)$ "

$\delta_2$. " $p(a,b) / p(a,b)$ "

$\delta_3$. " $p(a,c) / p(a,c)$ "

Set $A$ of assertions obtained by firing all defaults:

1. $B(a)$ (obtained by firing default $\delta_1$)
2. $p(a,b)$ (obtained by firing default $\delta_2$)
3. $p(a,c)$ (obtained by firing default $\delta_3$)

Set $S$ of the maximal subsets of the consequents of the defaults in $\Delta$:

$S = \{ \{B(a), p(a,b)\}, \{B(a), p(a,c)\}, \{p(a,b), p(a,c)\} \}$
Thank You!

For this presentation and associated references, search for “marco antonio casanova puc-rio”