Developing markup metaschemas to support interoperation among resources with different markup schemas

Gary Simons
SIL International

ACH/ALLC Joint Conference
29 May to 2 June 2003, Athens, GA

The Context

- The EMELD project
  - Electronic Metastructures for Endangered Language Data
  - Five year grant from NSF
  - Eastern Michigan, Wayne State, Arizona, LDC (Penn), Endangered Language Fund, SIL
- A major objective
  - The "formulation and promulgation of best practice in linguistic markup of texts and lexicon"
Problem Statement

- Three points of community consensus:
  - XML markup provides the best format for the interchange and archiving of EL data.
  - No single system of XML markup can be imposed on all language resources.
  - Linguists need to be able to perform queries across multiple resources.

- The problem
  - How do we interoperate when resources use different markup schemas?

The Basic Strategy

1. Develop community consensus on a shared ontology of linguistic concepts.
2. Define the semantics of a markup schema in terms of the shared linguistic concepts.
3. Map individual language resources onto their semantic interpretation.
4. Perform queries across resources over these semantic interpretations.
Overview of Paper

- Explain and illustrate the four steps of the basic strategy
- The sample application is from the domain of lexicography
- The sample language resources were three dictionaries with TEI-based markup:
  - Sikaiana of Solomon Islands (Donner, Simons)
  - Limbu of Nepal (Michailovsky)
  - Sindarin of Middle-earth (Tolkien, Willis)

1. Developing an Ontology

- Finding the GOLD
  - General Ontology for Linguistic Description
  - Langendoen, Lewis, and Farrar (U. of Arizona)
- Building on the W3C’s Semantic Web activity
  - Uses the RDF (Resource Description Framework) approach to semantic representation
  - Represents each concept by a URI
  - Defines formal properties of concepts with RDF Schema and OWL (Web Ontology Language)
The RDF Approach to Semantics

- Meaning is represented as a set of statements.
- Statement = < subject, predicate, object >
  - The subject is a URI representing a resource.
  - The predicate is a URI representing a property.
  - The object may be another resource or it may be a literal value.
- A set of statements forms a directed graph.
- Basis for interoperation: graphs for individual resources can be merged into one large graph.

Basics of RDF Schema

- The semantic schema formally defines the concepts (resource classes and properties) that are permitted in a semantic representation.
- rdfs:Class and rdf:Property are built-in resources.
- rdf:type is a property to identify the class of which a particular resource is an instance.
- rdfs:domain and rdfs:range are properties that constrain the subjects and objects of properties.
- rdfs:subClassOf and rdfs:subPropertyOf are properties that define is-a-kind-of hierarchies.
Example (in N3 notation)

@prefix gold: <http://www.emeld.org/GOLD-ns#>.
gold:LexicalItem  a rdfs:Class .
gold:form        a rdf:Property;
                 rdfs:domain  gold:LexicalItem;
                 rdfs:range   gold:LinguisticForm .
gold:variantForm a rdf:Property;
                 rdfs:subPropertyOf gold:form;
                 rdfs:domain  gold:LexicalItem;
                 rdfs:range   gold:LinguisticForm .
gold:meaning     a rdf:Property;
                 rdfs:domain  gold:LexicalItem;
                 rdfs:range   gold:LexicalSense .

2. Defining the Semantics of Markup

- markup schema
  - A formal definition (as with XML DTD or XML Schema) of the permitted vocabulary and syntax of markup for a class of source documents.

- semantic schema
  - A formal definition (as with RDF Schema or OWL) of the concepts in a particular domain.

- metaschema
  - A formal definition of how the elements and attributes of a markup schema are interpreted in terms of the concepts of a semantic schema.
A Metaschema Language

```xml
<!ELEMENT metaschema (interpret | ignore)+ >
<!ELEMENT interpret (resource | literal | property)* >
<!ATTLIST interpret markup CDATA #REQUIRED>
<!ELEMENT resource (literal | property | embed)>
<!ATTLIST resource concept CDATA #REQUIRED>
<!ELEMENT literal (text-content)* >
<!ATTLIST literal concept CDATA #REQUIRED>
<!ELEMENT property (resource | resourceRef | embed)>
<!ATTLIST property concept CDATA #REQUIRED>

... 
```

For example

- **Source document:**
  
  ```xml
  <entry id="aba"> <!-- Content --> </entry>
  ```

- **Metaschema directive:**
  
  ```xml
  <interpret markup="entry">
    <resource concept="gold:LexicalItem"/>
  </interpret>
  ```

- **Interpretation of document:**
  
  ```xml
  <gold:LexicalItem rdf:about="#element(aba)">
    <!-- Interpretation of content -->
  </gold:LexicalItem>
  ```
Example 2

- **Source document:**
  
  ```xml
  <form type="variant"><!-- Content --></form>
  ```

- **Metaschema directive:**
  
  ```xml
  <interpret markup="form[@type='variant']">
    <property concept="gold:variantForm">
      <resource concept="gold:LinguisticForm"/>
    </property></interpret>
  ```

- **Interpretation of document:**
  
  ```xml
  <gold:variantForm>
    <gold:LinguisticForm>
      <!-- Interpretation of content -->
    </gold:LinguisticForm>
  </gold:variantForm>
  ```

Example 3

- **Source document:**
  
  ```xml
  <orth>abba</orth>
  ```

- **Metaschema directive:**
  
  ```xml
  <interpret markup="orth">
    <literal concept="gold:spelling"/>
  </interpret>
  ```

- **Interpretation of document:**
  
  ```xml
  <gold:spelling>abba</gold:spelling>
  ```
More Features

- The full power of the XPath expression language is available to specify @markup.
- `<text-content>` allows literal values to be composed (with optional before and after labels) from multiple markup sources.
- `<embed>` allows explicit control of embedding:
  - partition of source child elements into separate substructures of the semantic interpretation
  - movement of source elements to a different spot in the semantic interpretation

3. Interpreting Individual Resources

```
+----------------+-------------------+-------------------+
<table>
<thead>
<tr>
<th>Metaschema</th>
<th>Source Document</th>
<th>Semantic Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>→</td>
<td>Document Interpreter</td>
<td></td>
</tr>
</tbody>
</table>
+----------------+-------------------+-------------------+
Implementation Strategy

- The document interpreter has been implemented in XSLT as a two-stage process:
  - Input: a metaschema document
    Stylesheet: the metaschema compiler (XSLT)
    Output: interpreter for that metaschema (XSLT)
  - Input: a source document
    Stylesheet: interpreter for the metaschema (XSLT)
    Output: the semantic interpretation (RDF/XML)

4. Querying Across Resources

- SD₁ → DI₁ → SI₁
- SD₂ → DI₂ → SI₂
- SD₃ → DI₃ → SI₃

Query Engine

Pooled Knowledge Store

MS₁
MS₂
MS₃

Query
Results
An Experimental Query Engine

- Uses *rdf_db: a simple RDF database in Prolog* with the open source SWI-Prolog
- Load each RDF/XML semantic interpretation file into the database with *rdf_load('filename').*
- This loader converts the RDF/XML into the equivalent *< Subject, Predicate, Object >* triples and asserts them into an RDF database.
- Use Prolog’s backward-chaining inference engine to answer queries.

For example

- Return the URI of all polysemous entries
  - ?- polysemous(X).
- Where:
  - lexicalItem(X) :- rdf(X, 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 'http://www.emeld.org/gold-ns#LexicalItem').
  - meaning(X,M) :- rdf(X, 'http://www.emeld.org/gold-ns#meaning', M).
  - polysemous(X) :- lexicalItem(X), meaning(X, M1), meaning(X, M2), M1 \neq M2.
Observations with Implications

<table>
<thead>
<tr>
<th>The three dictionaries were TEI-based, but there are significant differences in the metaschemas.</th>
<th>Using the same DTD is not enough to guarantee semantic interoperation of resources.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The exercise of mapping markup to semantics revealed aspects of markup that lacked a clear interpretation.</td>
<td>Ideally the metaschema would be created with the markup schema to ensure clear semantics for markup.</td>
</tr>
</tbody>
</table>

Conclusion

- A metaschema language for expressing the semantic interpretation of markup has been successfully defined and implemented:
  - The Semantic Web activity of the W3C proved a useful foundation for the approach to semantics.
  - An XSLT complier to produce an XSLT interpreter proved an easy way to implement it.
- Developing a service based on a complete semantic schema will be hard; but services with focused semantic schemas look feasible.