Meeting Minutes 15 February 2002
CellML Ontologies: How do we handle them?

Author:
Autumn Cuellar (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Warren Hedley (San Diego Supercomputer Center)
Poul Nielsen (Bioengineering Institute, University of Auckland)

1 Introduction

These meeting minutes cover discussions on how to handle different CellML ontologies. What do we mean by CellML ontologies? What existing languages were designed to define new ontologies? What are the limitations of these languages, i.e. how do they fulfill our needs and where do they fall short? Where do we go from here?

2 CellML ontologies: What are they?

Ontologies, as defined in the computer science world, are a fancy name for vocabularies. XML was designed to be extensible, but that extensibility can cause obvious problems since each group can define their own elements and attributes. For instance, one group may define an element **<creator>** that contains the name of the creator of the document while another group may define the element **<author>** for the same purpose. By the same token, two groups may use the same element name to convey two completely different meanings. There's been a move to find a way to define these different vocabularies to assist in translating between the languages.

When referred to in CellML, ontologies can have two different meanings. The first refers to CellML's individual role in the Physiome Markup Languages. CellML will eventually be combined with AnatML and FieldML (and possibly others) to be an entire XML dictionary which can be used to describe vast biological processes from the level of the body as a whole down to individual molecular interactions. CellML and its elements will need to be tied to the other Physiome Markup Languages. In a certain respect, CellML is a single ontology of a greater physiome language.

The second meaning of ontology used in CellML is of specific vocabularies of "things" at the cellular level that can be used over and over again by modellers. These can be cell compartments (mitochondria, cytoplasm, nucleus, etc.), proteins, kinetics mathematics, oft-used metabolic pathways, or anything else used repeatedly to describe what happens in the body at the cellular level.

These two types of ontologies have different requirements, therefore, I do not think we can approach them in the same way. Implementation of the first will consist of describing relationships between the MLs. Implementation of the second will depend on our capabilities to import or include the definitions of the cellular components into the CellML description of a model.

The following sections analyse the ontology definition languages that already exist and how appropriate each is for use in CellML.

3 Existing Ontology Languages

The World Wide Web Consortium[1], the organisation responsible for bringing us XML, seems not to have thought a whole lot about the ontology problem until recently when they became concerned with the

[1] http://www.w3.org/

semantic web activity[2]. To that end the RDF Schema Specification[3] was written which defines classes, simple relationships between the classes, and properties of the classes. Realizing the limitations of the RDF Schema language, the W3C accepted the submission of the DAML+OIL specification, which builds on both the RDF and XML Schema languages.

Before the RDF Schema Specification, a few ontology languages were written. I'll cover these briefly. All are fairly similar. Most of them, however, seem to have been abandoned since the W3C took over the ontology project.

All of the examples in this section use the **<reaction>** element of CellML and its children to demonstrate how a separate ontology could be described.

3.1 SHOE: Simple HTML Ontology Extensions

SHOE[4], an acronym for Simple HTML Ontology Extensions was originally designed for HTML (before XML was in widespread use). Minor changes were made to apply the language to XML, and the changes were encoded in a DTD.

Figure 1 shows the use of the SHOE ontology definition in CellML. Elements are defined in a **<def-category>** element, and the **isa** attribute explains the relationship between the current element and others. For instance, the category "variable_ref" is a child of the "reaction" category. Attributes are described using the **<def-relation>** element. A **<def-arg>** element with a **pos** of "1" should have a **type** value of the element (or category) that the attribute belongs to. A **<def-arg>** element with a **pos** of "2" should have a **type** value that gives a hint to what kind of value the attribute should have. For example, the "reversible" attribute should have a value of ".TRUTH" type: true or false (yes or no).

3.2 OML & XOL: Ontology Markup Language and the Ontology Exchange Language

OML[5], the Ontology Markup Language, was designed to complement CKML, the Conceptual Knowledge Markup Language. For this reason, the semantics are based on conceptual graphs. As a result, the folks who created XOL[6] were unhappy with it and decided to write their own ontology description language based on OML.

Figure 2 shows how XOL would be used to define the CellML reaction ontology. XOL describes ontologies in terms of classes, sub-classes, and slots. Slots contain a **<domain>** element which specifies the class the slot describes and a **<slot-value-type>** element which gives the type the slot value should be. The role slot shows that one can be specific in limiting the values of the **role** attribute. This is not possible in SHOE. However, XOL does not allow users to build on other documents. In other words, there would be no way to connect the reaction ontology to the overall CellML ontology.

3.3 RDF Schema

The RDF Schema[7], written in conjunction with the Resource Description Framework (RDF) Model and Syntax Specification[8], was meant to convey the relationships between resources and properties. Unlike DTDs and XML Schemas, RDF Schema does not put many constraints on the relationships.

[2] http://www.w3.org/2001/sw/
[3] http://www.w3.org/TR/2000/CR-rdf-schema-20000327/
[4] http://www.cs.umd.edu/projects/plus/SHOE/index.html
[5] http://www.ontologos.org/OML/OML%200.3.htm
[6] http://www.oasis-open.org/cover/xol-03.html
[7] http://www.w3.org/TR/2000/CR-rdf-schema-20000327/
[8] http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

```
<shoe
    version="1.0"
    xmlns="http://www.cs.umd.edu/projects/plus/SHOE/">
  <ontology id="reaction-ontology" version="1.0">
    <use-ontology id="base-ontology" prefix="base" url="http://www.cs.umd.edu/projects/plus/S
    <def-category isa="base.SHOEEntity" name="reaction" />
    <def-category isa="reaction" name="variable_ref" />
    <def-category isa="variable_ref" name="role" />
    <def-relation name="reversible">
      <def-arg pos="1" type="reaction" />
      <def-arg pos="2" type=".TRUTH" />
    </def-relation>
    <def-relation name="variable">
      <def-arg pos="1" type="variable_ref" />
      <def-arg pos="2" type=".STRING" />
    </def-relation>
    <def-relation name="role">
      <def-arg pos="1" type="role" />
      <def-arg pos="2" type=".STRING" />
    </def-relation>
  </ontology>
</shoe>
```

FIGURE 1: CellML's reaction ontology defined with SHOE.

In Figure 3 the RDF Schema language is used to define CellML's reaction ontology. As in XOL, RDF Schema describes ontologies in classes and sub-classes, but it uses the term property instead of slot for attributes. RDF Schema allows one to reference terms of other ontologies to describe the current ontology with the use of XML namespaces. In this example, the MathML **<math>** element becomes a **<subClassOf>** CellML's **<Role>** element. However, the use of the **ID** property makes it impossible to give elements and attributes the same name (which CellML does frequently), and, because it was not intended to put constraints on the language (except for the domain and range properties), RDF Schema is perhaps too limiting for the purposes of CellML ontologies.

3.4 DAML+OIL: DARPA Agent Markup Language and Ontology Interchange Language

DAML+OIL[9] is so named because it stems from two different languages: the DARPA Agent Markup Language and the Ontology Interchange Language. OIL was based on XOL, and DAML+OIL combines the OIL ontology base with RDF Schema and XML Schema to make a slightly more flexible ontology description language.

Figure 4 shows DAML+OIL's capabilities in defining the CellML reaction ontology. DAML+OIL does allow you to put greater restrictions on the classes, but the **role** and **reversible** attributes, which were formerly (and properly) properties, must become classes to limit their values. The classes can then have instances of the classes, i.e. "reactant" becomes an instance of the **<role>** class. Unfortunately, we'd have to alter the CellML language to implement these restrictions properly.

[9] http://www.w3.org/Submission/2001/12/

```
<module>
  <name>reaction</name>

  <class>
    <name>reaction</name>
    <documentation>The general reaction class.</documentation>
  </class>

  <class>
    <name>variable_ref</name>
    <subclass-of>reaction</subclass-of>
  </class>

  <class>
    <name>role</name>
    <subclass-of>variable_ref</subclass-of>
  </class>

  <slot>
    <name>reversible</name>
    <documentation>
      Describes whether a reaction is reversible or not.
    </documentation>
    <domain>reaction</domain>
    <slot-value-type>(set-of yes no)</slot-value-type>
  </slot>
  <slot>
    <name>variable</name>
    <slot-value-type>variable</slot-value-type>
  </slot>
  <slot>
    <name>role</name>
    <domain>role</domain>
    <slot-value-type>
      (set-of reactant catalyst activator inhibitor modifier
      rate)
    </slot-value-type>
  </slot>
</module>
```

FIGURE 2: CellML's reaction ontology defined with XOL.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdfs:Class rdf:ID="reaction">
    <rdfs:label>reaction</rdfs:label>
    <rdfs:comment>The reaction class.</rdfs:comment>
  </rdfs:Class>
  <rdfs:Class rdf:ID="variable_ref">
    <rdfs:subClassOf rdf:resource="#reaction" />
  </rdfs:Class>
  <rdfs:Class rdf:ID="Role">
    <rdfs:subClassOf rdf:resource="#variable_ref" />
  </rdfs:Class>
  <rdfs:Class rdf:about="http://www.w3.org/1998/Math/MathML/math">
    <rdfs:subClassOf rdf:resource="#Role" />
  </rdfs:Class>
  <rdf:Property rdf:ID="reversible">
    <rdfs:label>reversible</rdfs:label>
    <rdfs:comment>
    Describes whether a reaction is reversible.
    </rdfs:comment>
    <rdfs:domain rdf:resource="#reaction" />
  </rdf:Property>
  <rdf:Property rdf:ID="variable">
    <rdfs:domain rdf:resource="#variable_ref" />
    <rdfs:range rdf:resource="#Variable" />
  </rdf:Property>
  <rdf:Property rdf:ID="role">
    <rdfs:domain rdf:resource="#Role" />
  </rdf:Property>
</rdf:RDF>
```

FIGURE 3: CellML's reaction ontology defi ned with RDF Schema.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/02/22-rdf-syntax-ns#"
    xmlns:daml="http://www.w3.org/2001/10/daml+oil#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <daml:Ontology rdf:about=" ">
    <daml:imports rdf:resource="http://www.w3.org/2001/10/daml+oil" />
  </daml:Ontology>
  <daml:Class rdf:ID="reaction">
    <rdfs:label>reaction</rdfs:label>
    <rdfs:comment>The reaction class.</rdfs:comment>
  </daml:Class>
  <daml:Class rdf:ID="variable_ref">
    <rdfs:subClassOf rdf:resource="#reaction" />
  </daml:Class>
  <daml:Class rdf:ID="Role">
    <rdfs:subClassOf rdf:resource="#variable_ref" />
  </daml:Class>
  <daml:Class rdf:about="http://www.w3.org/1998/Math/MathML/math">
    <rdfs:subClassOf rdf:resource="#role" />
  </daml:Class>
  <daml:Class rdf:ID="reversible">
    <rdfs:label>reversible</rdfs:label>
    <rdfs:comment>
    Describes whether a reaction is reversible.
    </rdfs:comment>
    <rdfs:subClassOf rdf:resource="#reaction" />
  <daml:oneOf rdf:parseType="daml:collection">
    <reversible rdf:about="#Yes" />
    <reversible rdf:about="#No" />
   </daml:oneOf>
  </daml:Class>
  <rdf:Property rdf:ID="variable">
    <rdfs:domain rdf:resource="#variable_ref" />
    <rdfs:range rdf:resource="#Variable" />
  </rdf:Property>
  <daml:Class rdf:ID="role">
    <rdfs:subClassOf rdf:resource="#Role" />
  <daml:oneOf rdf:parseType="daml:collection">
      <role rdf:ID="reactant" />
    <role rdf:ID="product" />
  </daml:oneOf>
  </daml:Class>
</rdf:RDF>
```

FIGURE 4: CellML's reaction ontology defi ned with DAML+OIL.

4 XML Schema

XML Schema[10] and DTDs can both be thought of as low-level ontology languages. As David Bullivant pointed out, the difference between these languages and the ones listed above is that the ones above are intended for global (literally) exchanges of XML languages, whereas XML Schema and DTDs are very much specific to one language.

Figure 5 shows a clip of the definition of the CellML reaction ontology yet again, this time using the XML Schema language for definition. Unfortunately, there's a bit too much here to explain the XML Schema fully (see the W3C's XML Schema site for more information), but the interesting thing to note here is the way that this schema is tied to others. The MathML schema can be imported so that when the role element references the math element, any XML Schema-capable processing software will be able to understand what the **<mathml:math>** element is. To show that the **<reaction>** element is a child element of the CellML **<component>** element, the XML Schema **<redefine>** element is used. These are useful mechanisms, but like I've said, the XML Schema is highly language specific. One has to have control of the indicated namespaces to make clear these types of relationships. For instance, if one were to add another element that was essentially a subclass of the MathML **<math>** element, there'd be no way to indicate this with the XML Schema.

5 Conclusions

I have to keep reminding myself that XML is still a relatively new language. It's really taken off in the last few years, but there's still a whole lot to be done to make it what it could be. The CellML development team is used to adapting what's already out there for our own uses (reasonably enough: it makes things so much easier when the tools are already out there). But in this case, I don't know how useful any of the above languages are for what we intend with our ontologies.

I'm still convinced that XML Schema is simply a glorified DTD. It is often used in validation of documents, but it's not even that effective as a validator. There are many rules in CellML that cannot be conveyed with XML Schema. The fact that XML Schema cannot handle global relationship descriptions means little to CellML if we're talking about tying together the Physiome Markup Languages in the way described above. However, I should point out that it is unlikely that all the Physiome languages will be under the same namespace. This simple fact may hinder the suggestion of relationships between classes of elements. Then again, I'm not sure it's even necessary to link all the separate Physiome Languages as ontologies of the same language. Thoughts?

It seems likely that we will start with defining cellular component ontologies. The CellML development team agrees that it will be useful to create a standard dictionary in XML that can be *understood* by other XML documents. At the moment, with CellML Metadata, one can refer to databases, but the information in the databases doesn't mean anything to an XML processor. Theoretically, if there were a standard dictionary encoded in XML, a CellML document could refer to that dictionary and extract only the exact information it needs in a useful manner much in the way a web link can link to a section of the document. The dictionary could build off the work done by other organisations such as the GO Consortium[11]. RDF Schema will suffice (for the time being) in providing descriptions of the components that are repeatedly used by cell modellers. What's important enough to go into the first dictionaries will be a matter of debate.

[10] http://www.w3.org/XML/Schema
[11] http://www.geneontology.org/

```
<xsd:schema
    targetNamespace="http://www.cellml.org/cellml/1.0#"
    xmlns:cellml="http://www.cellml.org/cellml/1.0#"
    xmlns:mathml="http://www.w3.org/1998/Math/MathML"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      CellML 1.0 Reaction Ontology Schema for cellml.org.
      Copyright 2002 Bioengineering Institute, University of Auckland.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:include schemaLocation="./cellml_1_0.xsd" />
  <xsd:import namespace="http://www.w3.org/1998/Math/MathML" />

  <xsd:redefine schemaLocation="./cellml_1_0.xsd">
    <xsd:complexType name="component_type">
      <xsd:complexContent>
        <xsd:extension base="cellml:component_type">
          <xsd:choice maxOccurs="unbounded" minOccurs="0">
            <xsd:element maxOccurs="unbounded" minOccurs="0" ref="cellml:reaction" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:redefine>

  <xsd:element name="reaction" type="cellml:reaction_type">
    <xsd:keyref name="rxn_variables" refer="cellml:variable_name">
      <xsd:selector xpath="./cellml:variable_ref" />
      <xsd:field xpath="@cellml:variable" />
    </xsd:keyref>

    <xsd:unique name="rxn_variable">
      <xsd:selector xpath="./variable_ref" />
      <xsd:field xpath="@cellml:variable" />
    </xsd:unique>
  </xsd:element>

  <xsd:element name="variable_ref" type="cellml:variable_ref_type" />

  <xsd:element name="role" type="cellml:role_type" />

  <xsd:complexType name="reaction_type">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element maxOccurs="unbounded" ref="cellml:variable_ref" />
    </xsd:choice>
    <xsd:attribute default="yes" name="reversible" type="cellml:yes_no" use="optional" />
  </xsd:complexType>

  <xsd:complexType name="variable_ref_type">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element maxOccurs="unbounded" ref="cellml:role" />
    </xsd:choice>
    <xsd:attribute name="variable" type="xsd:string" use="required" />
  </xsd:complexType>

  <xsd:complexType name="role_type">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element maxOccurs="unbounded" minOccurs="0" ref="mathml:math" />
    </xsd:choice>
    <xsd:attribute name="role" type="cellml:species_role" use="required" />
  </xsd:complexType>

  <xsd:simpleType name="species_role">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="reactant" />
      <xsd:enumeration value="product" />
      <xsd:enumeration value="activator" />
      <xsd:enumeration value="catalyst" />
      <xsd:enumeration value="inhibitor" />
      <xsd:enumeration value="modifier" />
      <xsd:enumeration value="rate" />
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

FIGURE 5: CellML's reaction ontology defi ned with XML Schema.