BioPAX Validator

Igor Rodchenkov
University of Toronto

Introducing a new BioPAX software
BioPAX Workshop 2009 NYC

http://p.sf.net/biopax/validator

Why?

- We Are Not in the Ideal Semantic Web
- BioPAX Data Model (no XML Schema)
- Controlled Vocabularies
- Open World Assumption
- Best Practices and Normalization
  - *Chemical semantics*
  - *Xrefs Applicability*
  - *Sub-properties*
  - *Duplicate and Dangling Elements*
  - *Text and Numbers*

Goal: Easy and Consistent BioPAX

With The BioPAX Validator One Can:

- Check Syntax and Semantics
- Get Errors and Warnings (in Multiple Languages)
- Dynamically Control Rule's Action
- Loosely Integrate to Other BioPAX Software
- Auto-fix Errors and Normalize
- Get Examples and Comments

http://p.sf.net/biopax/validator

Requirements and Wishes

- User and System Requirements: http://p.sf.net/biopax/vrfc
- Validation Rules: http://p.sf.net/biopax/rules
  - Domain, Range and Cardinality
  - Proper Use of Sub-properties
  - Controlled Vocabulary
  - Syntax
  - Xrefs
  - Topology and Structure
  - Fix/Normalization

http://p.sf.net/biopax/validator

Design and Implementation

(1) Java 5, Spring Framework 2.5.6 (AOP, MVC,..), Apache Tomcat, and popular open source modules.

(2) Paxtools *(Excellent API!)*

(3) Ontology Manager (OM), MIRIAM, and OLS (or local ontologies

- *Java 5 and above Allows for Generics and Annotations*
- *Spring – to glue things together and make it configurable*
- *Spring AOP and LTW – to Intercept any Java Method (used for error reporting)!*
- *MIRIAM and OM – to Validate Xrefs and CV Terms*

http://p.sf.net/biopax/validator

(Live Demo)

- Using the Validator (Web Application)
  - Upload and Validate a Few Files
  - Rules Information and Dynamic Control
- Using WonderWeb OWL Validator
- Using Protégé

http://p.sf.net/biopax/validator

(Skip demo data)

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.biopax.org/examples/myExample#"
xmlns:bp="http://www.biopax.org/release/biopax-level3.owl#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#" xml:base="http://www.biopax.org/examples/myExample">
<owl:Ontology rdf:about="">
   <owl:imports rdf:resource="http://www.biopax.org/release/biopax-level3.owl" />
</owl:Ontology>
<!-- using illegal BioPAX L3 type (it's from L2; BioPAX problem, but, in fact, not an OWL error)-->
<bp:unificationXref rdf:ID="xrefL2">
   <bp:db rdf:datatype="http://www.w3.org/2001/XMLSchema#string">KEGG</bp:db>
<bp:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
This example is to illustrate using a wrong class name (from L2)
</bp:comment>
</bp:unificationXref>
   <!-- using non-existing property (OWL error) -->
   <bp:UnificationXref rdf:ID="xrefL3">
      <bp:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
This example is to illustrate using a wrong property name (this must be bp:id)
</bp:comment>
   <bp:ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">C00002</bp:ID>
<!-- validator must warn about using 'NIL', 'NULL', etc. for data properties-->
      <bp:db rdf:datatype="http://www.w3.org/2001/XMLSchema#string">NIL</bp:db>
      </bp:UnificationXref>
   <!-- missing ID (RDF error) -->
   <bp:Protein/>
</rdf:RDF>
```

Example 1: Cardinality and Range Rule

```
// BiochemicalPathwayStep.stepProcess range is Control.
@Component
public class BiochemicalPathwayStepProcessOnlyControlCRRule
 extends CardinalityAndRangeRule<BiochemicalPathwayStep> {
  public BiochemicalPathwayStepProcessOnlyControlCRRule() {
    super(BiochemicalPathwayStep.class, "stepProcess",
                                        0, 0, Control.class);
  }
}
```

Example 2: “ID” Rule

```
@Component
public class BiopaxElementIdRule
             extends AbstractRule<BioPAXElement> {

  public boolean canCheck(Object thing) {
    return (thing instanceof BioPAXElement);
  }

  public void check(BioPAXElement thing) {
    if(thing.getRDFId() != null) {
      String id = BiopaxValidatorUtils.getLocalId(thing);
      if(!NCName.isValid(id))
        error(thing, "invalid.rdf.id", id);
    } else error(thing, "invalid.rdf.id", "null value");
  }

  @Override
  protected void fix(BioPAXElement t, Object... values){}
}
```

Strengths

- Up-to-Date and Ongoing
- Customized for “BioPAX OWL”
- … incl. CV Rules (using latest ontologies)
- Simple API (only a few abstract classes to extend from)
- Greedy (looking for more cases)
- Traps *Expected* and *Unknown* Errors
- Open Source (finally...)

http://p.sf.net/biopax/validator

Next Steps

- Finalize and Tune (and add L2 rules, tests, and descriptions)
- Implement “Fix” Methods
- Enable the Max. Errors Limit and Fail on Severe Errors
- *Version 2.0: Migrate to OSGI*
- *Multi-parallel Multi-thread Validation*
- *A BioPAX Editor Backed by the Validator*

http://p.sf.net/biopax/validator

Thanks

- Special thanks to To Gary Bader and Emek Demir!
- Pathway Commons Team!
- Sylva Donaldson, Kumaran Kandasamy, Sasha Tkachev et al. for Valuable Feedbacks!

Java, Spring, Apache, and many open source projects - for excellent and sharp tools; to Google, Sourceforge, Apple et al. – for I can do everywhere!

BioPAX NYC'09 Workshop

- *Computational Biology Center Memorial Sloan Kettering Cancer Center 1275 York Avenue, Box #460, New York, NY 10065. November 11 – 13, 2009.*
- *Phone: +1 646 888 2602 or +1 646 888 2606*
- *Fax: +1 646 422 0717*
- *E-mail: workshop2009@biopax.org*
- *Wed-Thu: at 69th street entrance of the Zuckerman Building a little before 9am*
- *Fri: at security at the Rockefeller Research Laboratories building on 67th Street*

http://p.sf.net/biopax/validator