Tools for Molecular Interaction Maps and a Graphical Notation Validation Framework for PathVisio

COMBINE 2011

9/6/2011

Augustin Luna

augustin@mail.nih.gov

Bioinformatics Dept., Boston University, Boston, MA, USA

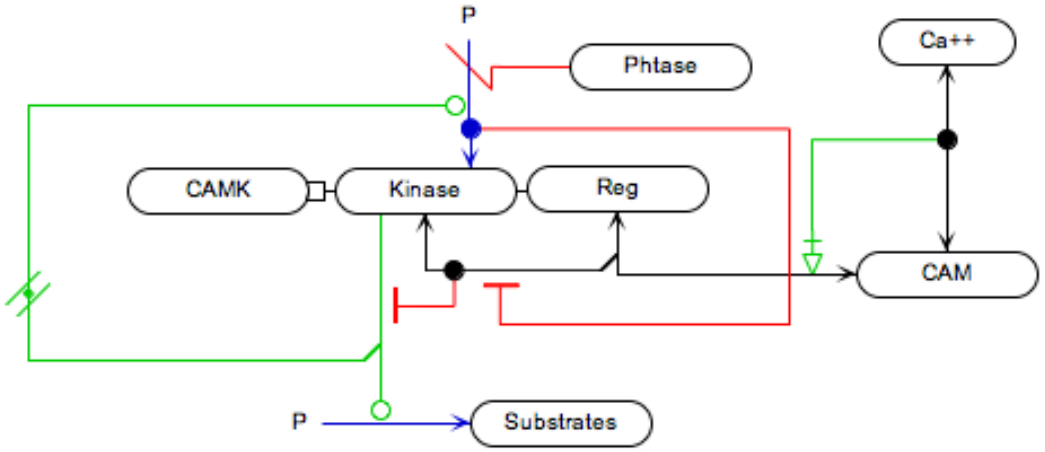
Laboratory of Molecular Pharmacology, National Cancer Institute, Bethesda, MD, USA

Overview

- Overview of Molecular Interaction Map (MIM) software
  - Update specification, format, API, and diagram editor
  - Network analysis
- Validation framework (Kumar Chandan/GSOC)
  - Validator plugin
  - Features
  - Groovy and Schematron ruleset support
- Issues in validator development
- Extending Schematron
- Future work

Molecular Interaction Maps (MIMs)

- A graphical notation for bioregulatory networks created by Kurt Kohn in 1999
- Creation of large diagrams that remain intuitive for use by biologists by minimizing redundant elements
- Precursor to SBGN-ER
- Tools for biologists and programmers



Luna A. et al., 2011

Current MIM Tools

```
23      <mimVis:EntityGlyph visId="f1327" pos
24      <mimVis:EntityGlyph visId="e1fe7" pos
25      <mimVis:EntityGlyph visId="d73bd" cer
="20.000000000000018" color="000000" type
26          <mimVis:GenericProperty key="Numb
27          <mimVis:GenericProperty key="Shap
28      </mimVis:EntityGlyph>
29      <mimVis:EntityGlyph visId="e4900" cer
="20.0" color="000000" type="SimplePhysic
30          <mimVis:GenericProperty key="Numb
31          <mimVis:GenericProperty key="Shap
32      </mimVis:EntityGlyph>
33      <mimVis:EntityGlyph visId="c0757" cer
"20.0" color="000000" type="SimplePhysica
34          <mimVis:GenericProperty key="Numb
35          <mimVis:GenericProperty key="Shap
36      </mimVis:EntityGlyph>
37      <mimVis:EntityGlyph visId="a482c" cer
type="SimplePhysicalEntity" displayName="
38          <mimVis:GenericProperty key="Numb
39          <mimVis:GenericProperty key="Shap
40      </mimVis:EntityGlyph>
41      <mimVis:EntityGlyph visId="d8200" cer
```

Machine-Readable Format

```
1  package gov.nih.nci.lmp.mim.mimExample;
2
3  import java.io.File;
4  import java.io.FileOutputStream;
5  import java.io.OutputStream;
6  import java.util.HashMap;
7
8  import org.apache.xmlbeans.XmlOptions;
9
10 import gov.nih.nci.lmp.mim.mimVisLevel1.*
11
12 /**
13  * An class that shows an toy example for
14  */
15 public class CaCamExample {
16
17     /** The MIM-Vis doc. */
18     public DiagramDocument visDoc;
19
20     /**
21      * Create a DiagramDocument from scra
22      * physical entity (SPE), Ca++, bindi
23      */
24     public void createCaCam() {
25         visDoc = DiagramDocument.Factory.
26
27         // Create a new diagram to go wit
28         DiagramType dia = visDoc.addNewDi
29         dia.setWidth(84.7);
30         dia.setHeight(146.3);
31
32         // Create a new entity and add it
33         EntityGlyphType ca = dia.addNewEn
34         ca.setVisId("b4357");
35         ca.setCenterX(47.0);
36         ca.setCenterY(38.0);
37         ca.setWidth(60.0);
38         ca.setHeight(20.0);
39         ca.setColor("000000");
40         // This shows the way to access e
41         ca.setType(EntityGlyphType.Type.S
```

Programming Interface

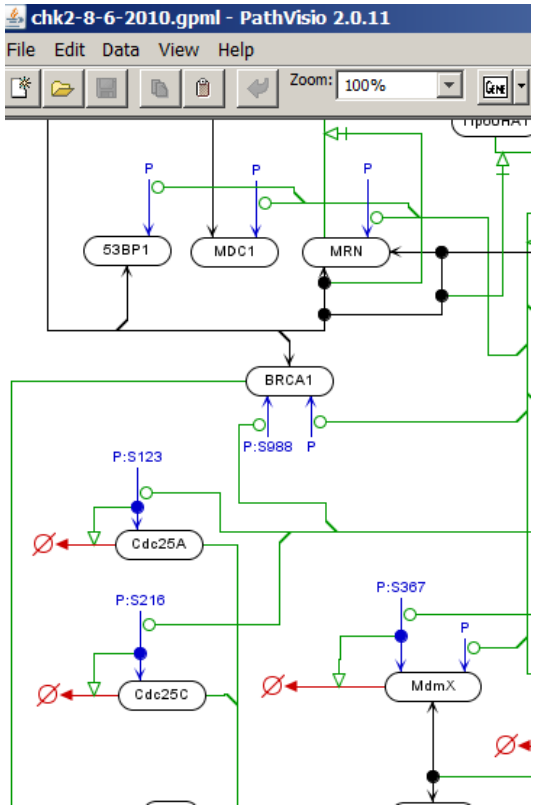


Diagram Editor

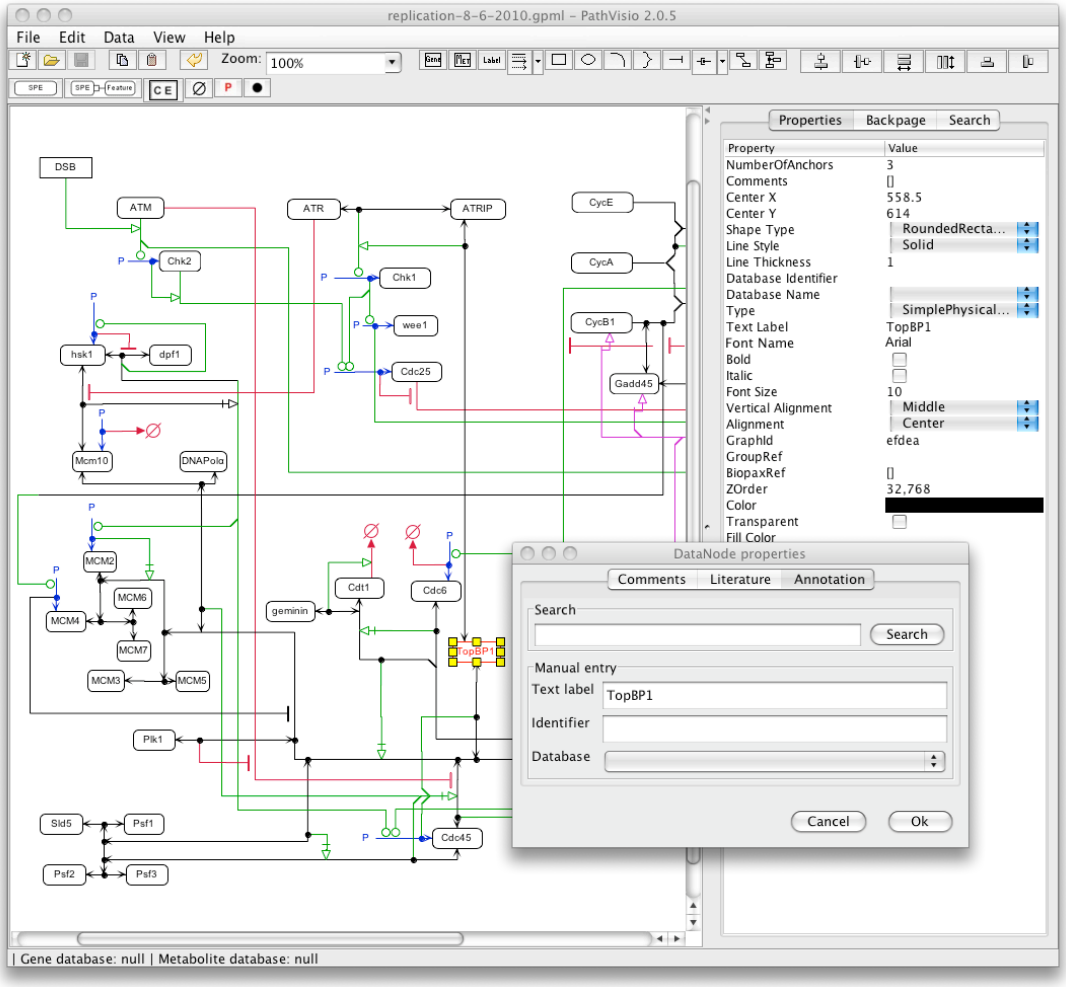
Luna A. et al. (BMC Bioinformatics) 2011; Luna A. et al. (Bioinformatics) 2011

Current MIM Tools

- Updated MIM specification to make MIM “machine-friendly”
- MIM format similar on GPML
- API being generated with XMLBeans for Java
- Syntactic validation using Schematron
  - Makes assertions about the presence or absence of patterns in XML documents using XPath
  - Processed into XSLT

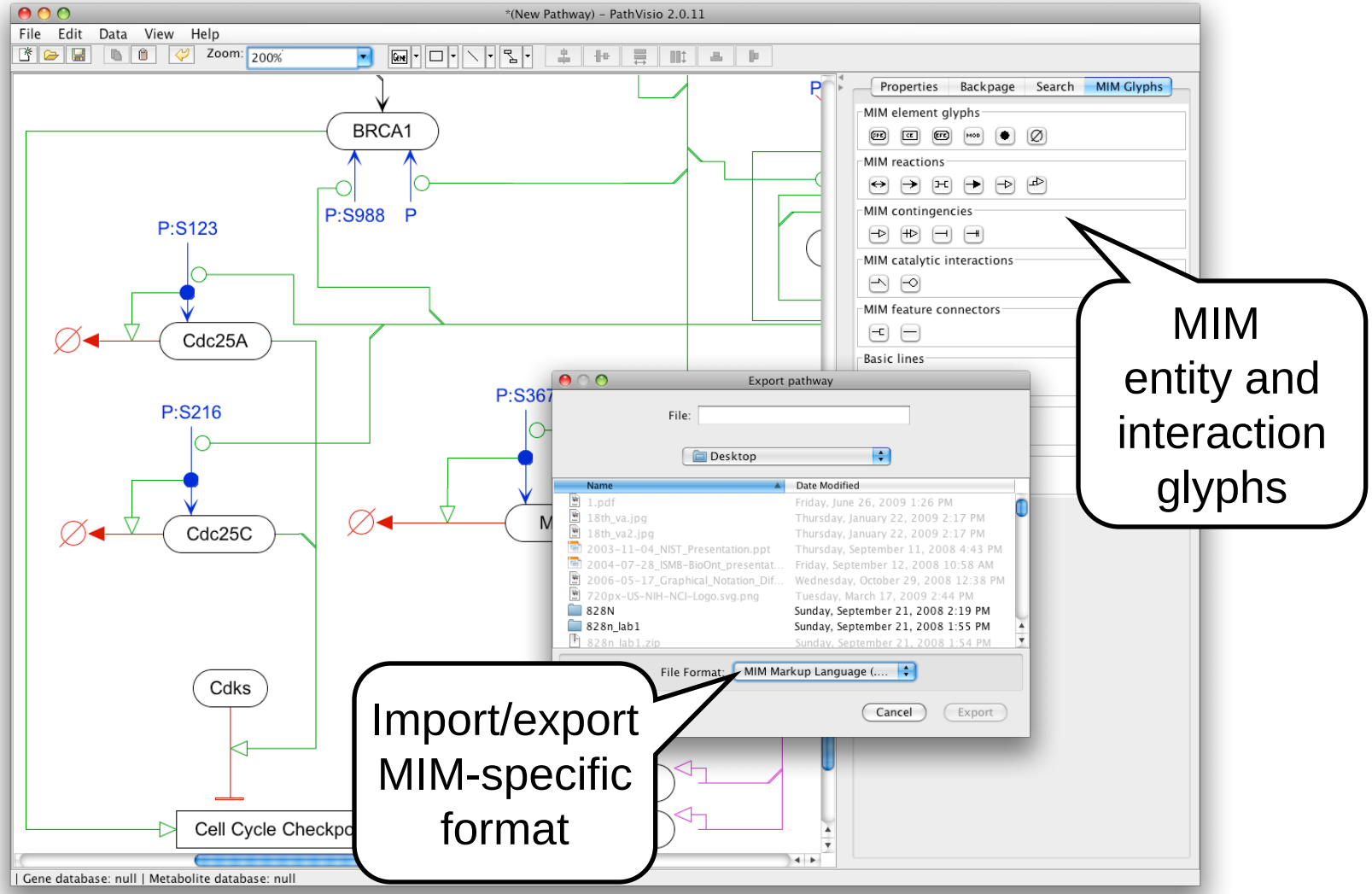
Choosing PathVisio for MIM Diagrams

- Open-source
- Java
- Plugins
  - Cytoscape
  - BioPAX
  - SBGNML
- WikiPathways

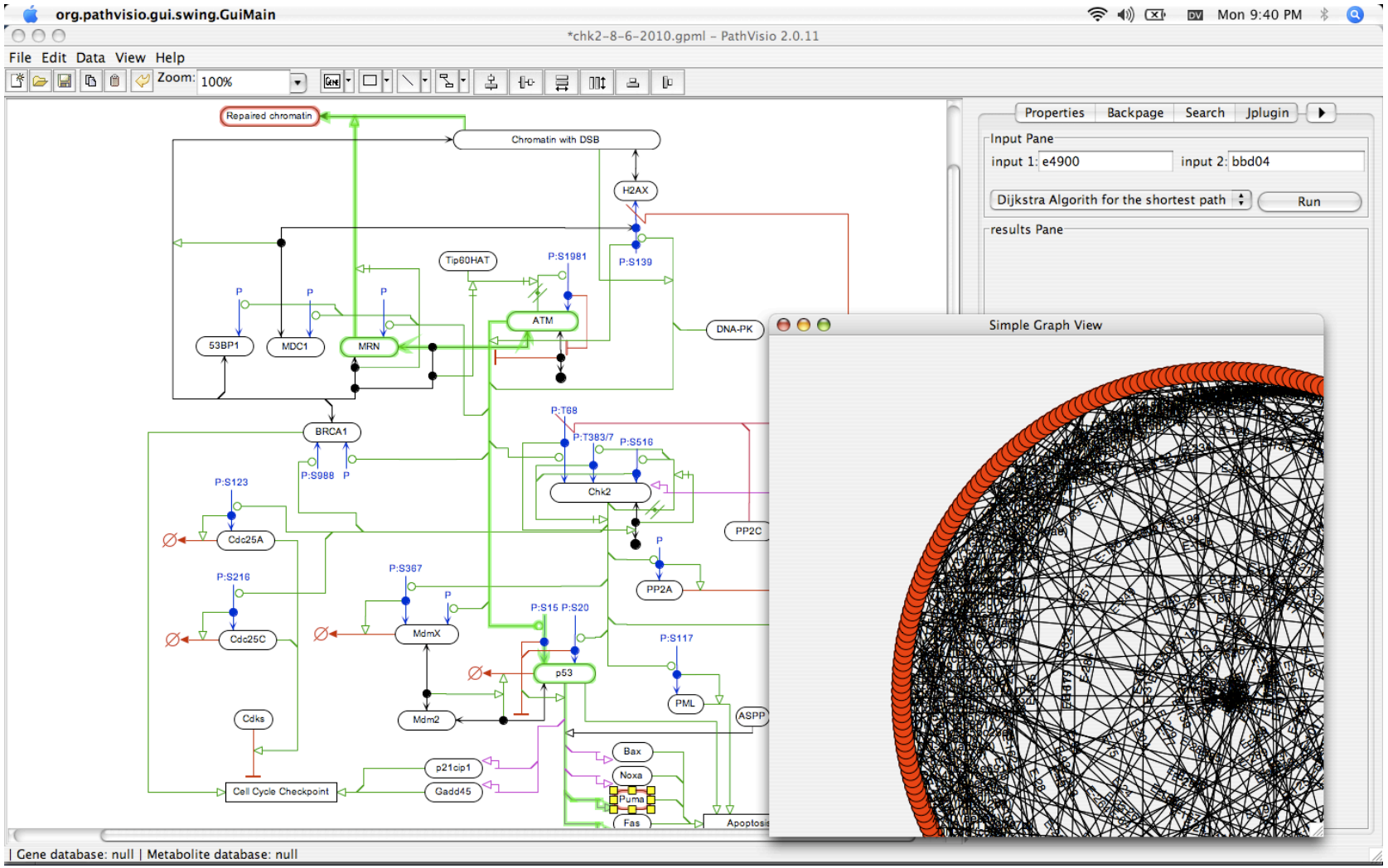


van Iersel, M. et al. 2008

PathVisio-MIM Plugin

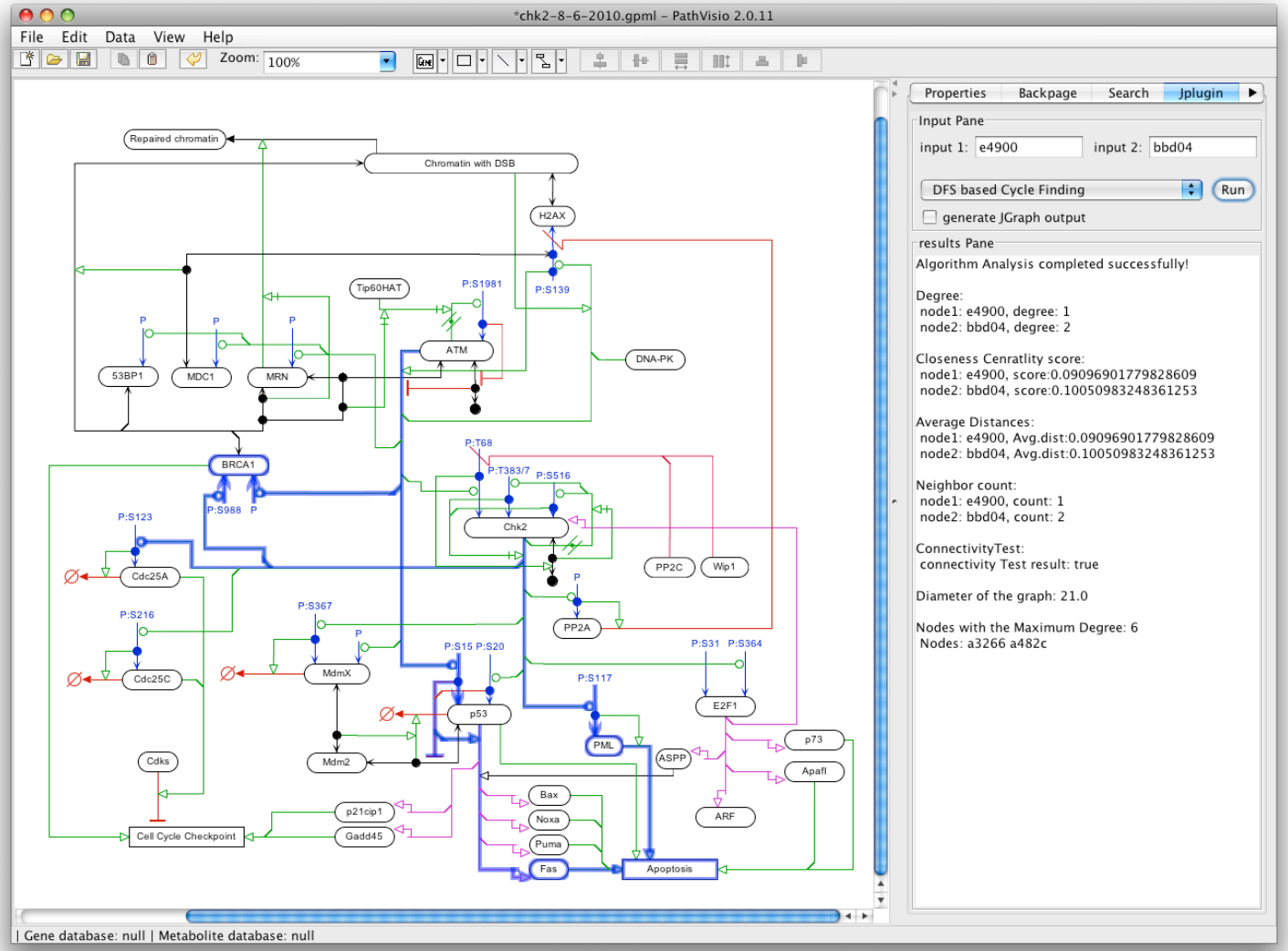


Network Analysis: Shortest Path



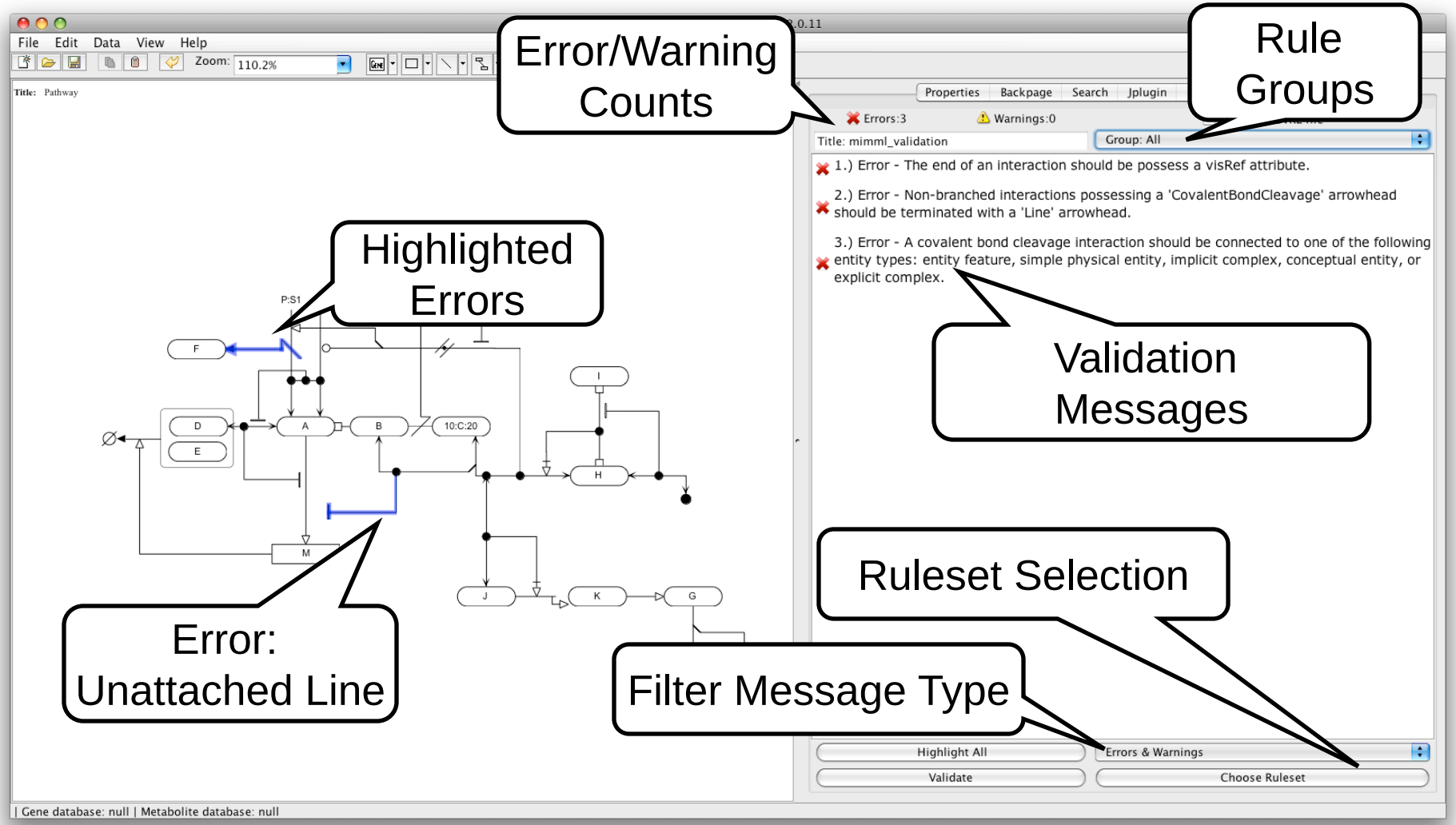
Chandan, K. and Luna, A. (unpublished)

Feedback Cycle Finding



Chandan, K. and Luna, A. (unpublished)

PathVisio Validator Plugin

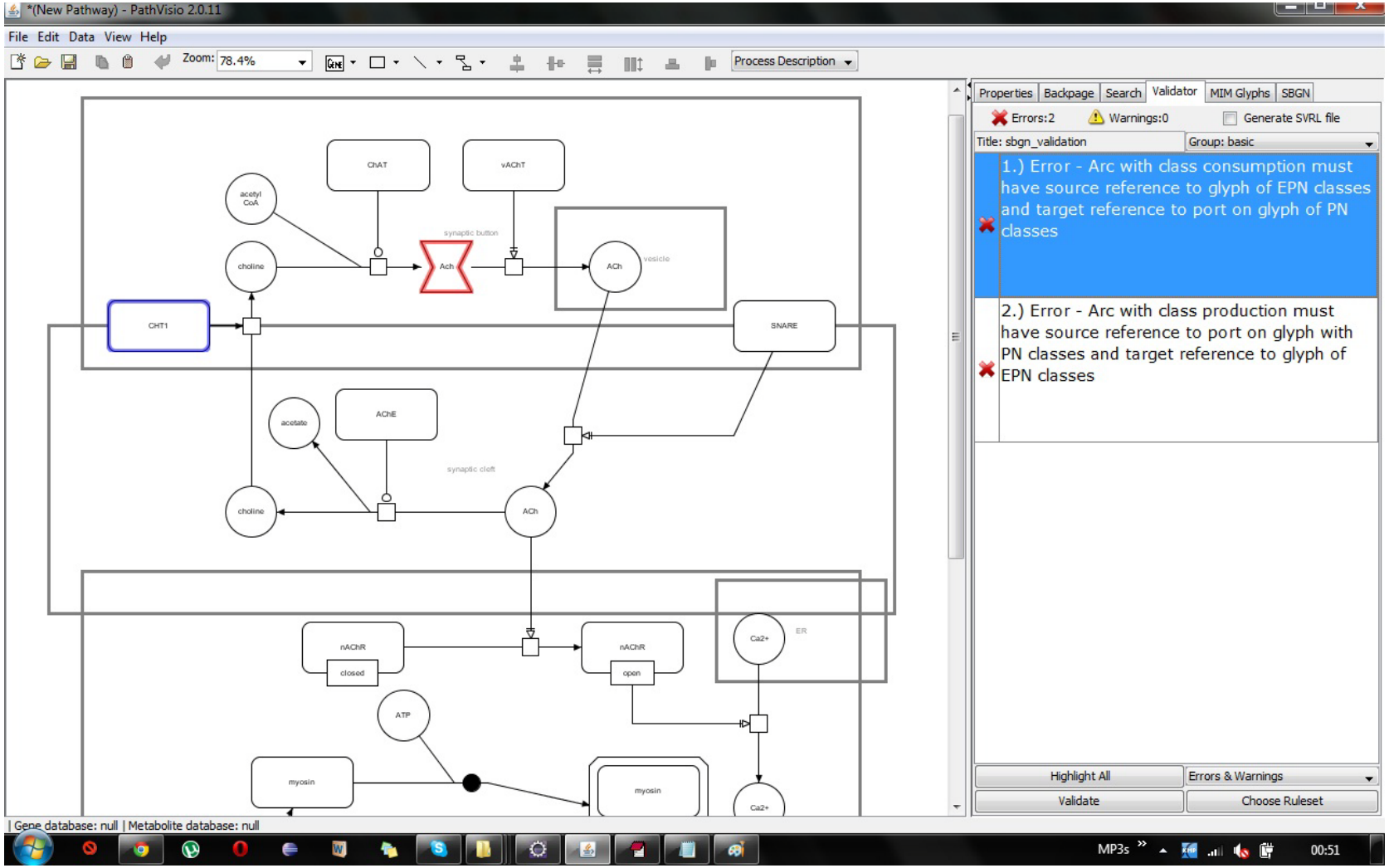


Chandan K. et al. (submitted)

PathVisio Validator Plugin Features

- Current Rulesets: MIM, WikiPathways, SBGN expected
- Selective rules
  - Validation on rule subsets
  - Users can filter out errors or warnings
  - Ignore diagram element or message instance
  - Ignore in session or permanently
- Groovy and Schematron support
  - Groovy:
    - Easier to write for Java developers
    - Tied to internal workings of PathVisio
  - Schematron:
    - Reusable in software with access to an XSLT processor
    - Standardized validation report language

SBGN Validation Support in PathVisio



PathVisio Validator Plugin Features

- Current Rulesets: MIM, WikiPathways, SBGN expected
- Selective rules
  - Validation on rule subsets
  - Users can filter out errors or warnings
  - Ignore diagram element or message instance
  - Ignore in session or permanently
- Groovy and Schematron support
  - Groovy:
    - Easier to write for Java developers
    - Tied to internal workings of PathVisio
  - Schematron:
    - Reusable in software with access to an XSLT processor
    - Standardized validation report language

Comparison Schematron and Groovy: Unattached Line Check

Groovy

```
//The rule below checks for unattached lines by checking each of the "Line" element's first and last "GraphRef" attributes.
ArrayList<String[ ]> ruleUnattachedLines(Pathway pw) {
    String[ ] result = null; // the intermediate result to be added to the final result
    ArrayList<String[ ]> totalResultForThisRule = null; // the final resullt which is sent to the Validator
    for(PathwayElement pwe: pw.getDataObjects()){
        if(pwe.getObjectType()==ObjectType.LINE &&  // check if the Pathway element is of the type : "Line"
                ( pwe.getStartGraphRef()==null | pwe.getEndGraphRef()==null // check its "GraphRef" values
                    | pwe.getEndGraphRef().equals("") | pwe.getStartGraphRef().equals("") ) ){
            if(totalResultForThisRule==null){
                totalResultForThisRule=new ArrayList<String[ ]>();
            }
            //result[0]: "error" / "warning"; result[1]: diagnostic message; result[2]: element's Graph-Id
                result = ["error", "Lines should be attached at both ends.", pwe.getGraphId()];
                totalResultForThisRule.add(result);
        }
    }
    if(totalResultForThisRule==null) System.out.println("All the lines are attached");
    return totalResultForThisRule; // this final result is passed to the Validator
}
```

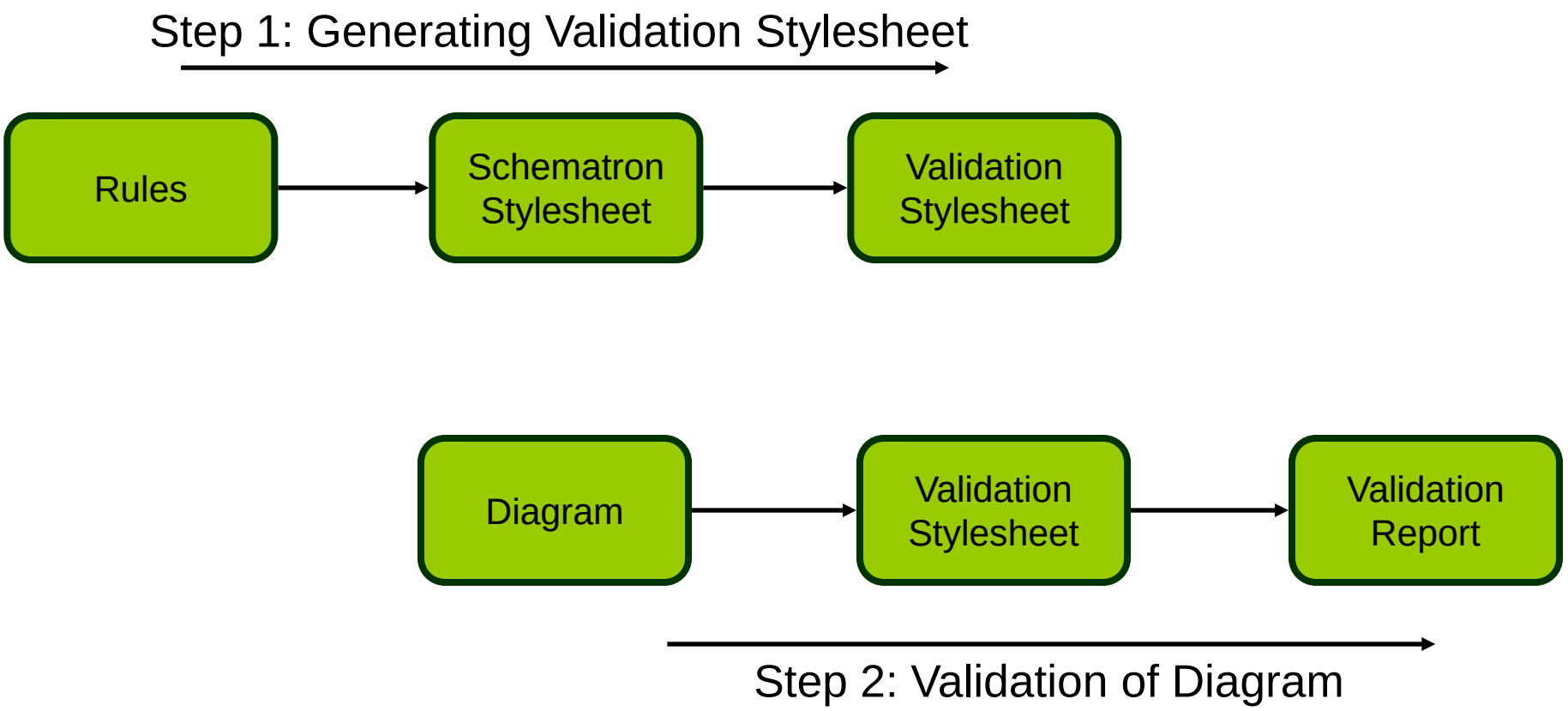
Schematron

```
<!-- Check that all lines are attached at both ends -->
  <iso:pattern name="check-unattached-lines" id="check-unattached-lines">
    <!-- Check "Line" elements -->
    <iso:rule context="gpml:Line" role="error">
      <!-- Get the unique identifier for the line -->
      <iso:let name="graph-id" value="@GraphId"/>

      <!-- Get the GraphRefs (these indicate the GraphIds of the objects the line is connected to) of the first and last points for a line -->
      <iso:let name="start-graph-ref" value="gpml:Graphics/gpml:Point[position()=1]/@GraphRef"/>
      <iso:let name="end-graph-ref" value="gpml:Graphics/gpml:Point[last()]/@GraphRef"/>

      <!-- Assert that the GraphRef attributes on the first and last points are present and not empty, otherwise output error message and
      identifier -->
      <iso:assert test="$start-graph-ref and $end-graph-ref and not($start-graph-ref='') and not($end-graph-ref='')"
      diagnostics="graph-id">Lines should not be unattached.</iso:assert>
    </iso:rule>
  </iso:pattern>
```

Validation Mechanism Using Schematron



Validation Process

- Groovy
  1. Validate diagram
  2. Display messages
- Schematron
  1. Generate validation stylesheet
  2. Export diagram
  3. Transform diagram to validate
  4. Parse validation report
  5. Display messages

Schematron Rule and Validation Report

```
<iso:pattern name="check-int-visref" id="check-int-visref">
<iso:rule context="mimVis:InteractionGlyph">
<iso:let name="vis-id" value="@visId"/>
<iso:assert test="mimVis:Point[1]/@visRef" diagnostics="vis-id">The start of an interaction should possess a visRef attribute.</iso:assert>
<iso:assert test="mimVis:Point[last()]/@visRef" diagnostics="vis-id">The end of an interaction should be possess a visRef attribute.</iso:assert>
</iso:rule>
</iso:pattern>
```

Validation Schematron File

```
<svrl:active-pattern id="check-int-visref" />
<svrl:failed-assert
    test="mimVis:Point[1]/@visRef"
  location="...">
<svrl:text>The start of an interaction should possess a visRef attribute.</svrl:text>
<svrl:diagnostic-reference diagnostic="vis-id">idcd72516b</svrl:diagnostic-reference>
</svrl:failed-assert>
</svrl:active-pattern>
```

SVRL Validation Output

Issues Encountered

- Validating files with non-MIM glyphs
  - Exporter could not function because of enumerated types in XML schema
  - Resolution: Moved enumeration validation to Schematron
- Issues incorporating Saxon XSLT processor
  - As of Saxon 9.2, some useful features (e.g. extensions) were removed from the free version
  - Resolution: Used Saxon-B (Saxon 9.1)

Schematron Extensions with Saxon XSLT Processer

Schematron File

```
<iso:schema
  xmlns:iso="http://purl.oclc.org/dsdl/schematron"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <iso:ns prefix="sbgn" uri="http://sbgn.org/libsbgn/0.2"/>
  <iso:ns prefix="sbgn-ext" uri="java:org.sbgn.SaxonTools.Extensions"/>
  <iso:pattern name="saxon-test" id="00001"><iso:rule context="sbgn:arc">
    <iso:assert
      test="not(sbgn-ext:return-true())">Saxon extension worked.</iso:assert>
  </iso:rule></iso:pattern>
</iso:schema>
```

Java Extension

```
package org.sbgn.SaxonTools;
public class Extensions {
    public static Boolean returnTrue() {
        Boolean b = new Boolean("true");
        return b;
    }
}
```

Next Steps

- Molecular Interaction Maps (MIMs)
  - Continued work network analysis components for MIMs
- Validation
  - Extend to other notations:
    - SBGN ruleset
    - KEGG ruleset?
  - Incorporate validator into WikiPathways (http://wikipathways.org/)
  - Continuous background validation
  - COMBINE validation report language (combinevrl)
- Software Links
  - MIM software: http://discover.nci.nih.gov/mim
  - Validator plugin: http://pathvisio.org/wiki/PathwayValidatorHelp

Acknowledgements

- National Cancer Institute (MIM)
  - Kurt Kohn
  - Mirit Aladjem
  - Margot Sunshine
  - Sohyoung Kim
  - Lucas Chang
- U of Maastricht (Pathvisio)
  - Martjin van Iersel
  - Pathvisio developers
- Google Summer of Code/KMIT
  - Kumar Chandan
- Bogazici University and NCI-Frederick
  - Itir Karac
  - Mine Edes
  - Ruth Nussinov
  - Can Ozturan
- BioPAX Community
- SBGN Community