Arrays Package

Leandro H. Watanabe

University of Utah

Google Summer of Code

August 20, 2014

SBML Current Limitations

- Currently, mathematical operations in SBML are restricted to operations on scalar values.
- Regular structures cannot be represented efficiently.
- This motivated the development of the arrays package.

Extensions to MathML

- *constructors*:
  - **vector**
- *element referenced operator*:
  - **selector**

Vector Examples

Empty Vector

Infix Notation	MathML
{}	<vector> </vector>

Vector Examples

1-D Vector

Infix Notation	MathML
{1,2,3}	<vector> <cn type="integer">1</cn> <cn type="integer">2</cn> <cn type="integer">3</cn> </vector>

Vector Examples

2-D Vector

Infix Notation	MathML
{{0,X},{1, X+1}}	<vector> <vector> <cn type="integer">0</cn> <ci>X</ci> </vector> <vector> <cn type="integer">1</cn> <apply> <plus /> <ci>X</ci> <cn type="integer">1</cn> </apply> </vector> </vector>

Selector Examples

Selector for selecting an object from a 1-D array

Infix Notation	MathML

X[0]	`<apply>`
	`  <selector />`
	`  <ci>X</ci>`
	`  <cn type="integer">0</cn>`
	`</apply>`

Selector Examples

Selector for selecting an object from a 2-D array

Infix Notation	MathML

X[(n-1)-i][j]

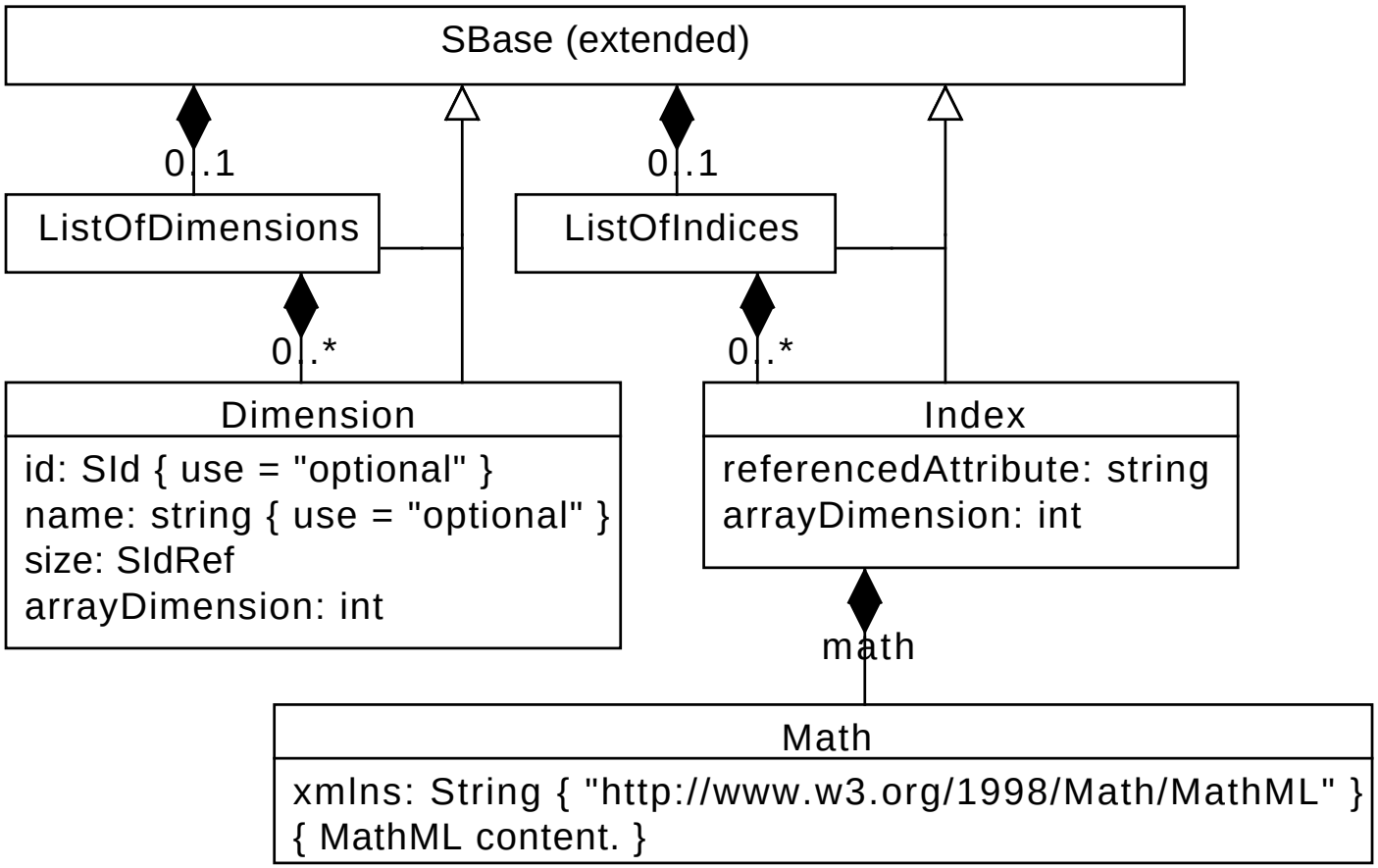
```
<apply>
   <selector />
   <ci>X</ci>
   <apply>
      <minus />
      <apply>
         <minus />
         <ci>n</ci>
         <cn type="integer">1</cn>
      </apply>
      <ci>i</ci>
   </apply>
   <ci>j</ci>
</apply>
```

Selector Examples

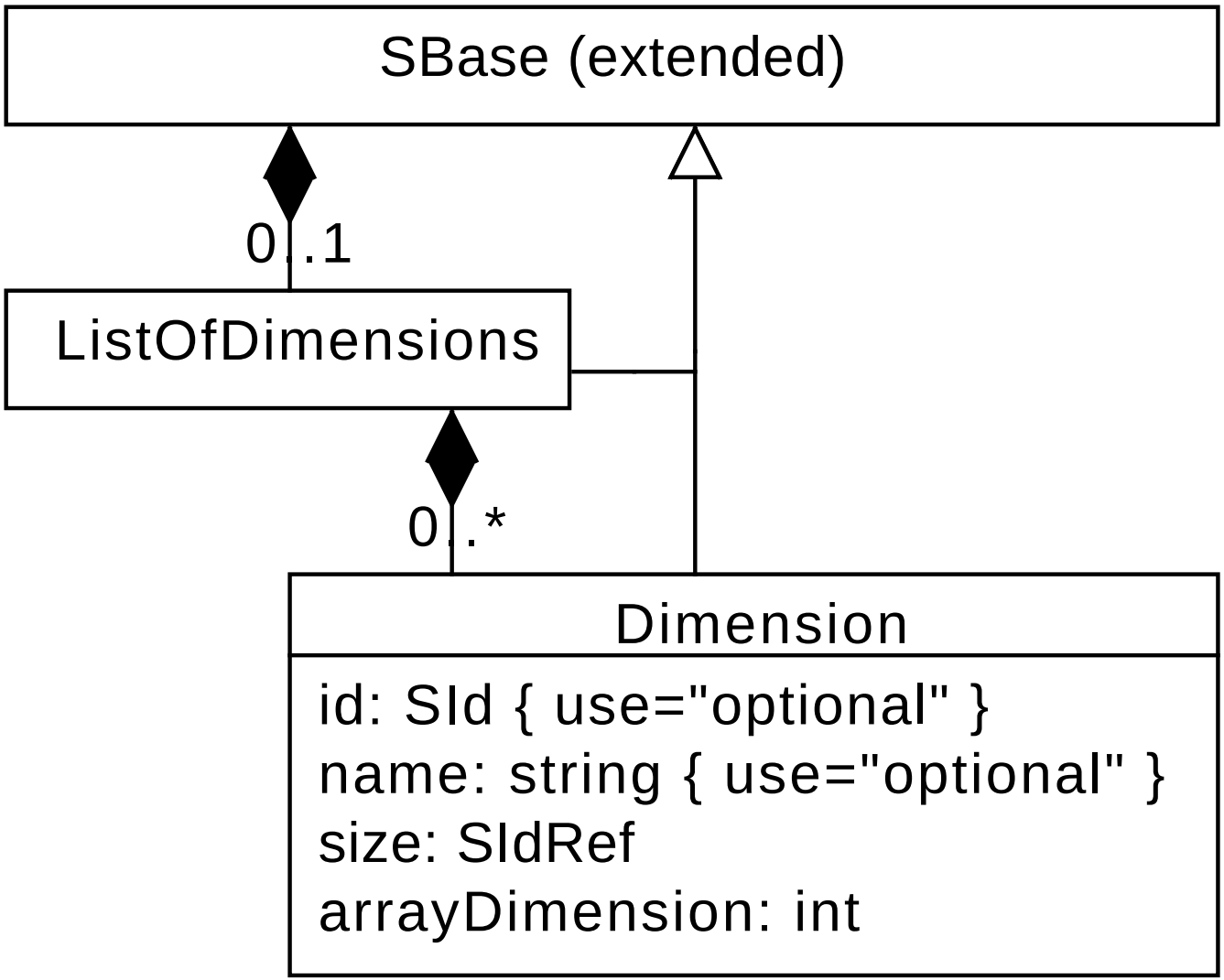
Selector for selecting an object from a vector

Infix Notation	MathML
{1,2,3}[0]	<apply> <selector /> <vector> <cn type="integer">1</cn> <cn type="integer">2</cn> <cn type="integer">3</cn> </vector> <cn type="integer">0</cn> </apply>

Arrays Extension



Dimension


SBase (extended)
0..1
ListOfDimensions
0..*
Dimension
id: SId { use="optional" }
name: string { use="optional" }
size: SIdRef
arrayDimension: int

Dimension Example

Example how you would create a 10x10 compartment array in SBML.

```
<parameter id="n" value="10"/>
<compartment id="C" ...>
    <arrays:listOfDimensions>
        <arrays:dimension id="d0" size="n" arrayDimension="0"/>
        <arrays:dimension id="d1" size="n" arrayDimension="1"/>
    </arrays:listOfDimensions>
</compartment>
```

Dimension Example

Create parameter to be used as the dimension size.

```
<parameter id="n" value="10"/>
<compartment id="C" ...>
    <arrays:listOfDimensions>
        <arrays:dimension id="d0" size="n" arrayDimension="0"/>
        <arrays:dimension id="d1" size="n" arrayDimension="1"/>
    </arrays:listOfDimensions>
</compartment>
```

Dimension Example

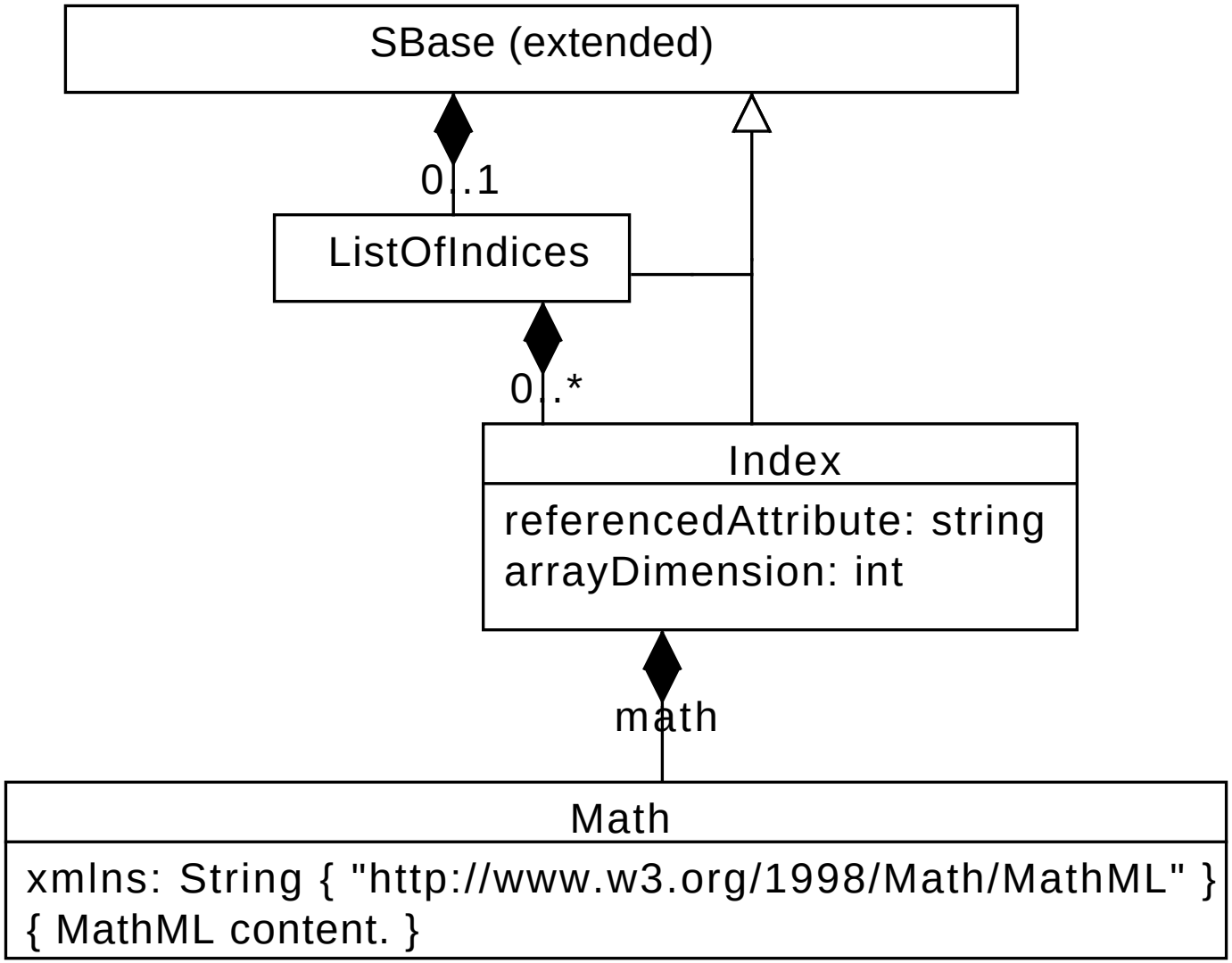
Create two dimension objects of size n.

```
<parameter id="n" value="10"/>
<compartment id="C" ...>
    <arrays:listOfDimensions>
        <arrays:dimension id="d0" size="n" arrayDimension="0"/>
        <arrays:dimension id="d1" size="n" arrayDimension="1"/>
    </arrays:listOfDimensions>
</compartment>
```

SBML Elements That Can Have List Of Dimensions

- In SBML Core, ListOf objects are not allowed to have a ListOfDimensions.
- All SBML objects defined that inherit from SBase are permitted to have a ListOfDimensions unless it is explicitly disallowed in the corresponding specification.
- In SBML Core, the elements that can have a ListOfDimensions are:
  - Compartments
  - Species, Parameters
  - Initial assignments
  - Rules
  - Constraints
  - Reactions
  - Species references
  - Events
  - Event assignments

Index


SBase (extended)
0..1
ListOfIndices
0..*
Index
referencedAttribute: string
arrayDimension: int
math
Math
xmlns: String { "http://www.w3.org/1998/Math/MathML" }
{ MathML content. }

Index Example

```
n = 10;
x[n], y[n];
for (d0=0; d0 < n; d0++) {
  y[(n-1) - d0] = x[d0];
}
```

Index Example

```
n = 10;
x[n], y[n];
for (d0=0; d0 < n; d0++) {
  y[(n-1) - d0] = x[d0];
}
```

Index Example (cont.)

Create parameter to be used as the size of the arrays.

```
<parameter id="n" value="10"/>
<parameter id="X" ...>
  <arrays:listOfDimensions>
    <arrays:dimension id="d0" size="n" arrayDimension="0"/>
  </arrays:listOfDimensions>
</parameter>
<parameter id="Y" ...>
  <arrays:listOfDimensions>
    <arrays:dimension id="d0" size="n" arrayDimension="0"/>
  </arrays:listOfDimensions>
</parameter>
```

Index Example

```
n = 10;
x[n], y[n];
for (d0=0; d0 < n; d0++) {
  y[(n-1) - d0] = x[d0];
}
```

Index Example (cont.)

Create parameter array X.

```
<parameter id="n" value="10"/>
<parameter id="X" ...>
    <arrays:listOfDimensions>
        <arrays:dimension id="d0" size="n" arrayDimension="0"/>
    </arrays:listOfDimensions>
</parameter>
<parameter id="Y" ...>
    <arrays:listOfDimensions>
        <arrays:dimension id="d0" size="n" arrayDimension="0"/>
    </arrays:listOfDimensions>
</parameter>
```

Index Example (cont.)

Create parameter array Y.

```
<parameter id="n" value="10"/>
<parameter id="X" ...>
    <arrays:listOfDimensions>
        <arrays:dimension id="d0" size="n" arrayDimension="0"/>
    </arrays:listOfDimensions>
</parameter>
<parameter id="Y" ...>
    <arrays:listOfDimensions>
        <arrays:dimension id="d0" size="n" arrayDimension="0"/>
    </arrays:listOfDimensions>
</parameter>
```

Index Example

```
n = 10;
x[n], y[n];
for (d0=0; d0 < n; d0++) {
  y[(n-1) - d0] = x[d0];
}
```

Index Example (cont.)

Create an array of assignment rules with size n.

```
<assignmentRule variable=Y>
    <arrays:listOfDimensions>
        <arrays:dimension id="d0" size="n" arrayDimension="0"/>
    </arrays:listOfDimension>
```

Index Example

```
n = 10;
x[n], y[n];
for (d0=0; d0 < n; d0++) {
  y[(n-1) - d0] = x[d0];
}
```

Index Example (cont.)

Assign a value to y.

```
<arrays:listOfIndices>
      <arrays:index referencedAttribute="variable"
        arrayDimension="0">
          <math>
              <apply>
                  <apply>
                      <minus/>
                      <apply>
                          <minus/>
                          <ci> n </ci>
                          <cn type="integer"> 1 </cn>
                      </apply>
                      <ci> d0 </ci>
                  </apply>
              </apply>
          </math>
      </arrays:index>
  </arrays:listOfIndices>
```

Index Example (cont.)

Assign a value to y at (n-1)-d0.

```
<arrays:listOfIndices>
        <arrays:index referencedAttribute="variable"
          arrayDimension="0">
            <math>
                <apply>
                    <apply>
                        <minus/>
                        <apply>
                            <minus/>
                            <ci> n </ci>
                            <cn type="integer"> 1 </cn>
                        </apply>
                        <ci> d0 </ci>
                    </apply>
                </apply>
            </math>
        </arrays:index>
    </arrays:listOfIndices>
```

Index Example (cont.)

Assign the value of x[d0] to y[(n-1)-d0].

```
    <math>
        <apply>
            <selector/>
            <ci>X</ci>
            <ci>d0</ci>
        </apply>
    </math>
</assignmentRule>
```

SBML Core Elements That Can Have List of Indices

- Only SBML objects that include defined attributes of type SIdRef are permitted to have a List of Indices.
- The following SBML Core objects can have a List of Indices:
  - Model - conversionFactor
  - Species - compartment, conversionFactor
  - Initial assignments - symbol
  - Rules - variable
  - Species references - species
  - Events assignments - variable

Arrays Validation

- A validation routine for the Arrays package is implemented in JSBML.
- The arrays specification defines a set of rules that models using the arrays extension must not violate.
- The validation routine checks if every validation rule is met.

Arrays Validation Rules Examples (MathML)

(Ragged)

{{1,2,3}, {4,5}}

(Ragged)

{{{1},{2},3}}

(OK)

{{1,2,3}, {4,5,6}}

(OK)

{{{1},{2},{3}}}

(Vectors do not match in size)

{1,2} + {3} - 1

(OK)

{1,2} + {3,4} - 1

Arrays Validation Rules Examples (MathML)

- n is a constant parameter of size 10 and X is an array of size n.

i is non-constant parameter	i is dimension id
(Not statically computable)	(OK)
X[n-1-i] 	X[n-1-i] 
i = 10	i = 10
(Out-of-bounds)	(Out-of-bounds)
X[n-1-i] 	X[i] 
i = 9	i = 0
(OK)	(OK)
X[n-1-i] 	X[i] 

Arrays Validation Rules Examples (SBase)

(Multiple Dimension objects with the same array dimension attribute)

```
<arrays:listOfDimensions
xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
    <arrays:dimension arrays:size="n" arrays:arrayDimension="0"/>
    <arrays:dimension arrays:size="n" arrays:arrayDimension="0"/>
</arrays:listOfDimensions>
```

(OK)

```
<arrays:listOfDimensions
    xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
    <arrays:dimension arrays:size="n" arrays:arrayDimension="0"/>
    <arrays:dimension arrays:size="n" arrays:arrayDimension="1"/>
</arrays:listOfDimensions>
```

Arrays Validation Rules Examples (SBase)

(Must have a Dimension object with array dimension of value 0)

```
<arrays:listOfDimensions
    xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
    <arrays:dimension arrays:size="n" arrays:arrayDimension="1"/>
</arrays:listOfDimensions>
```

(OK)

```
<arrays:listOfDimensions
    xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
    <arrays:dimension arrays:size="n" arrays:arrayDimension="0"/>
</arrays:listOfDimensions>
```

Arrays Validation Rules Examples (SBase)

(Cannot have multiple Index objects with the same array dimension value)

```
<arrays:listOfIndices
    xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
    <arrays:index arrays:referencedAttribute="species" arrays:arrayDimension="0">
        <math
            xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> 0 </ci>
        </math>
    </arrays:index>
    <arrays:index arrays:referencedAttribute="species" arrays:arrayDimension="0">
        <math
            xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> 0 </ci>
        </math>
    </arrays:index>
</arrays:listOfIndices>
```

Arrays Validation Rules Examples (SBase)

(OK)

```
<arrays:listOfIndices
    xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
    <arrays:index arrays:referencedAttribute="species" arrays:arrayDimension="0">
        <math
            xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> 0 </ci>
        </math>
    </arrays:index>
    <arrays:index arrays:referencedAttribute="species" arrays:arrayDimension="1">
        <math
            xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> 0 </ci>
        </math>
    </arrays:index>
</arrays:listOfIndices>
```

Arrays Validation Rules Examples (SBase)

(Needs an Index with array dimension of value 0)

```
<arrays:listOfIndices
    xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
    <arrays:index arrays:referencedAttribute="species" arrays:arrayDimension="1">
        <math
            xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> 0 </ci>
        </math>
    </arrays:index>
</arrays:listOfIndices>
```

(OK)

```
<arrays:listOfIndices
    xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
    <arrays:index arrays:referencedAttribute="species" arrays:arrayDimension="0">
        <math
            xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> 0 </ci>
        </math>
    </arrays:index>
</arrays:listOfIndices>
```

Arrays Validation Rules Examples (Dimension)

(Parameter size does not exist)

```
<listOfParameters>
    <parameter id="n" constant="true" value="10"/>
    <parameter id="X" value="1">
        <arrays:listOfDimensions
            xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
            <arrays:dimension arrays:id="d0" arrays:size="m" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
    </parameter>
</listOfParameters>
```

(Parameter size is not constant)

```
<listOfParameters>
    <parameter id="n" constant="false" value="10"/>
    <parameter id="X" value="1">
        <arrays:listOfDimensions
            xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
    </parameter>
</listOfParameters>
```

Arrays Validation Rules Examples (Dimension)

(Size cannot be negative)

```
<listOfParameters>
    <parameter id="n" constant="true" value="-1"/>
    <parameter id="X" value="1">
        <arrays:listOfDimensions
            xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
    </parameter>
</listOfParameters>
```

Arrays Validation Rules Examples (Dimension)

(Size must be a scalar Parameter)

```
<listOfParameters>
    <parameter id="n" constant="true" value="10"/>
    <arrays:listOfDimensions
        xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
        <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
    </arrays:listOfDimensions>
    <parameter id="X" value="1">
        <arrays:listOfDimensions
            xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
    </parameter>
</listOfParameters>
```

Arrays Validation Rules Examples (Dimension)

(OK)

```
<listOfParameters>
    <parameter id="n" constant="true" value="10"/>
    <parameter id="X" value="1">
        <arrays:listOfDimensions
            xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1">
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
    </parameter>
</listOfParameters>
```

Arrays Validation Rules Examples (Index)

(Referenced attribute is not valid)

```
<listOfParameters>
    <parameter id="n" constant="true" value="10"/>
    <parameter id="X" value="1">
        <arrays:listOfDimensions...>
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
    </parameter>
</listOfParameters>
<listOfInitialAssignments>
    <initialAssignment symbol="X">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> d0 </ci>
        </math>
        <arrays:listOfDimensions...>
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
        <arrays:listOfIndices...>
            <arrays:index arrays:referencedAttribute="variable" arrays:arrayDimension="0">
                <math...>
                    <ci> d0 </ci>
                </math>
            </arrays:index>
        </arrays:listOfIndices>
    </initialAssignment>
</listOfInitialAssignments>
```

Arrays Validation Rules Examples (Index)

(Index math is not statically computable)

```
<listOfParameters>
    <parameter id="n" constant="true" value="10"/>
    <parameter id="var" constant="false" value="10"/>
    <parameter id="X" value="1">
        <arrays:listOfDimensions...>
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
    </parameter>
</listOfParameters>
<listOfInitialAssignments>
    <initialAssignment symbol="X">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> d0 </ci>
        </math>
        <arrays:listOfDimensions...>
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
        <arrays:listOfIndices...>
            <arrays:index arrays:referencedAttribute="symbol" arrays:arrayDimension="0">
                <math...>
                    <ci> var </ci>
                </math>
            </arrays:index>
        </arrays:listOfIndices>
    </initialAssignment>
</listOfInitialAssignments>
```

Arrays Validation Rules Examples (Index)

(Index math has out-of-bounds problems)

```
<listOfParameters>
    <parameter id="n" constant="true" value="10"/>
    <parameter id="X" value="1">
        <arrays:listOfDimensions...>
            <arrays:dimension arrays:id="i" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
    </parameter>
</listOfParameters>
<listOfInitialAssignments>
    <initialAssignment symbol="X">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> d0 </ci>
        </math>
        <arrays:listOfDimensions...>
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
        <arrays:listOfIndices...>
            <arrays:index arrays:referencedAttribute="symbol" arrays:arrayDimension="0">
                <math...>
                    <ci> d0+1 </ci>
                </math>
            </arrays:index>
        </arrays:listOfIndices>
    </initialAssignment>
</listOfInitialAssignments>
```

Arrays Validation Rules Examples (Index)

(OK)

```
<listOfParameters>
    <parameter id="n" constant="true" value="10"/>
    <parameter id="X" value="1">
        <arrays:listOfDimensions...>
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
    </parameter>
</listOfParameters>
<listOfInitialAssignments>
    <initialAssignment symbol="X">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> d0 </ci>
        </math>
        <arrays:listOfDimensions...>
            <arrays:dimension arrays:id="d0" arrays:size="n" arrays:arrayDimension="0"/>
        </arrays:listOfDimensions>
        <arrays:listOfIndices...>
            <arrays:index arrays:referencedAttribute="symbol" arrays:arrayDimension="0">
                <math...>
                    <ci> d0 </ci>
                </math>
            </arrays:index>
        </arrays:listOfIndices>
    </initialAssignment>
</listOfInitialAssignments>
```

Arrays Flattening

- Arrays package is syntatic sugar for SBML models.
- An SBML document using Arrays constructs can be converted to a new SBML document with objects inlined.
- Assumption: The flatten routine takes a valid SBML document and produces a valid flattened SBML document.

Flattening Example

```
n = 5;
x[n], y[n];
for (d0=0; d0 < n; d0++) {
  y[(n-1) - d0] = x[d0];
}
```

Flattening Example (Expanding Rules)

```
n = 5;
x[n], y[n];
y[(n-1) - 0] = x[0];
y[(n-1) - 1] = x[1];
y[(n-1) - 2] = x[2];
y[(n-1) - 3] = x[3];
y[(n-1) - 4] = x[4];
```

Flattening Example (Expanding Parameters)

```
n = 5;
x_0, x_1, x_2, x_3, x_4;
y_0, y_1, y_2, y_3, y_4;
y[(n-1) - 0] = {x_0, x_1, x_2, x_3, x_4}[0];
y[(n-1) - 1] = {x_0, x_1, x_2, x_3, x_4}[1];
y[(n-1) - 2] = {x_0, x_1, x_2, x_3, x_4}[2];
y[(n-1) - 3] = {x_0, x_1, x_2, x_3, x_4}[3];
y[(n-1) - 4] = {x_0, x_1, x_2, x_3, x_4}[4];
```

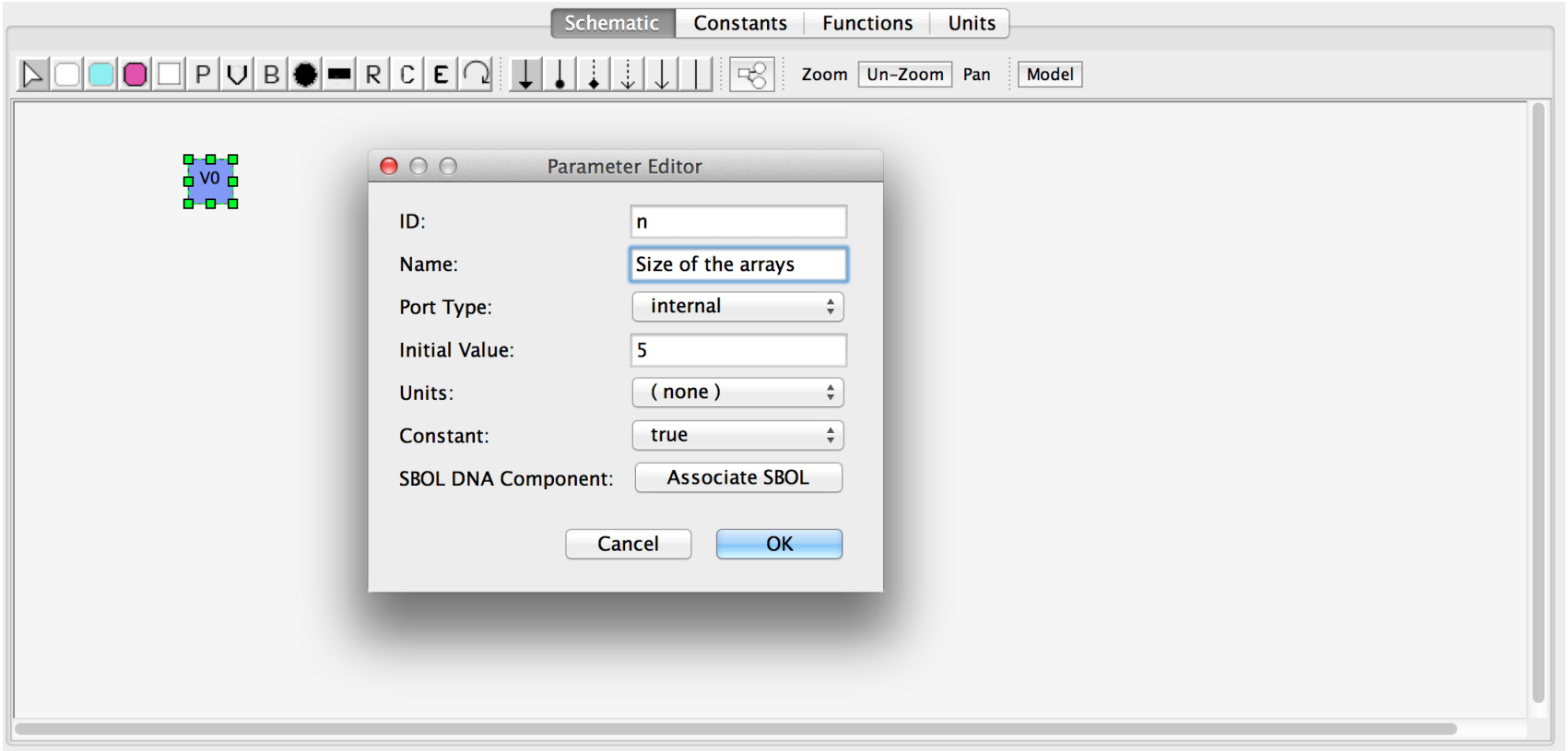
Flattening Example (Evaluating Indices)

```
n = 5;
x_0, x_1, x_2, x_3, x_4;
y_0, y_1, y_2, y_3, y_4;
y_4 = {x_0, x_1, x_2, x_3, x_4}[0];
y_3 = {x_0, x_1, x_2, x_3, x_4}[1];
y_2 = {x_0, x_1, x_2, x_3, x_4}[2];
y_1 = {x_0, x_1, x_2, x_3, x_4}[3];
y_0 = {x_0, x_1, x_2, x_3, x_4}[4];
```

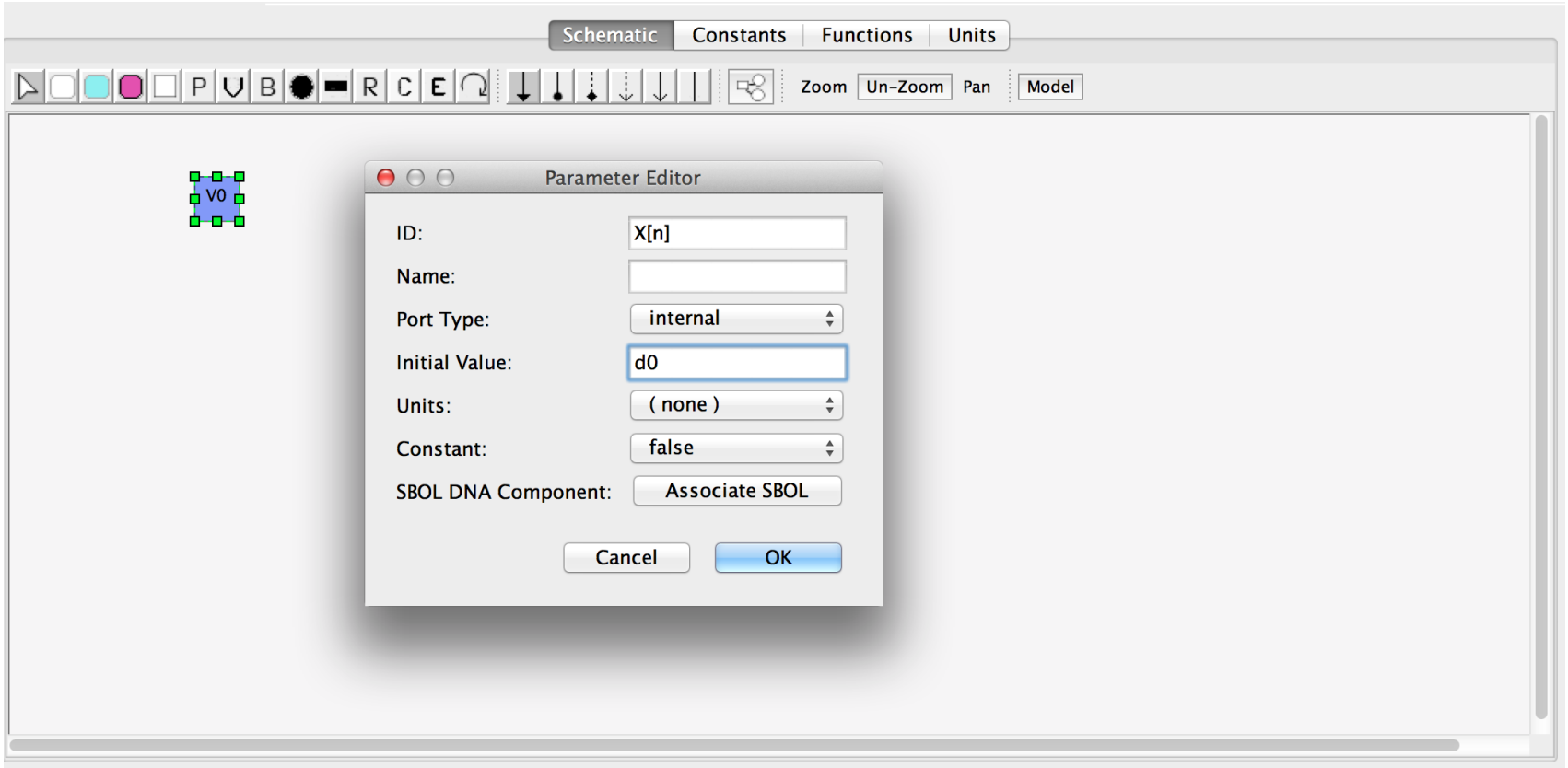
Flattening Example (Evaluating Vectors and Selectors)

```
n = 5;
x_0, x_1, x_2, x_3, x_4;
y_0, y_1, y_2, y_3, y_4;
y_4 = x_0;
y_3 = x_1;
y_2 = x_2;
y_1 = x_3;
y_0 = x_4;
```

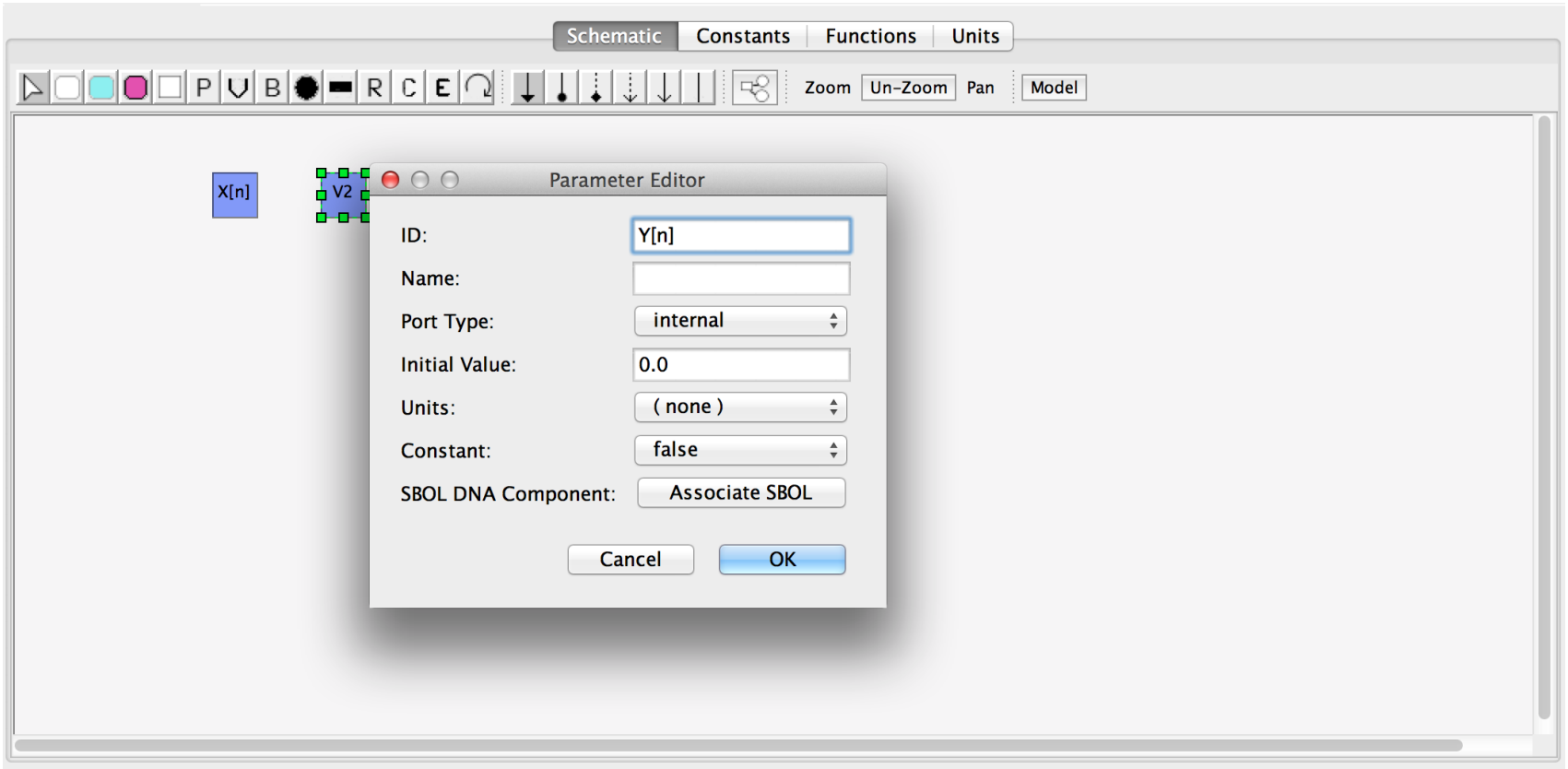
iBioSim and Arrays package (Creating Constant Parameter)



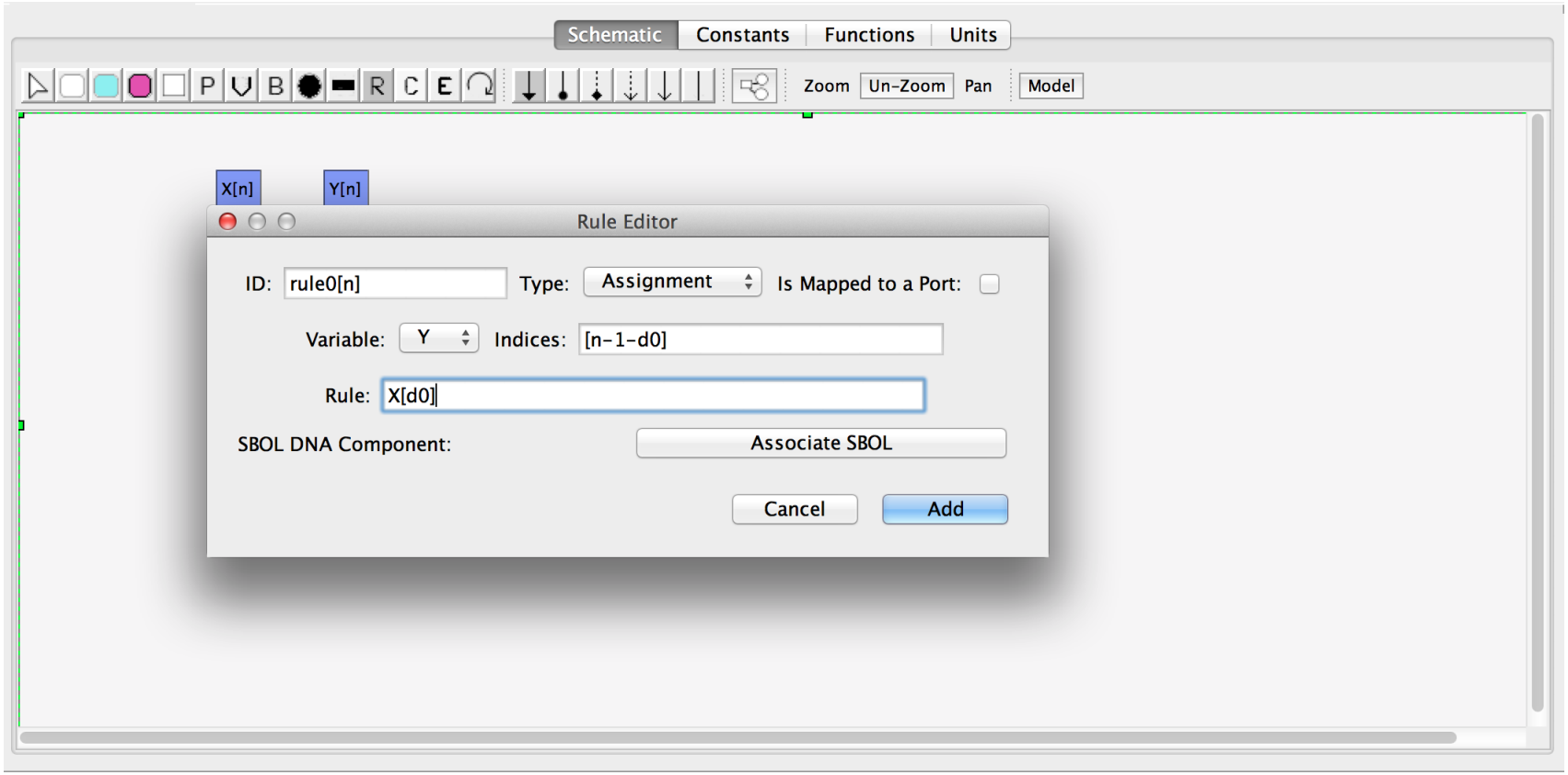
iBioSim and Arrays package (Creating Array X)



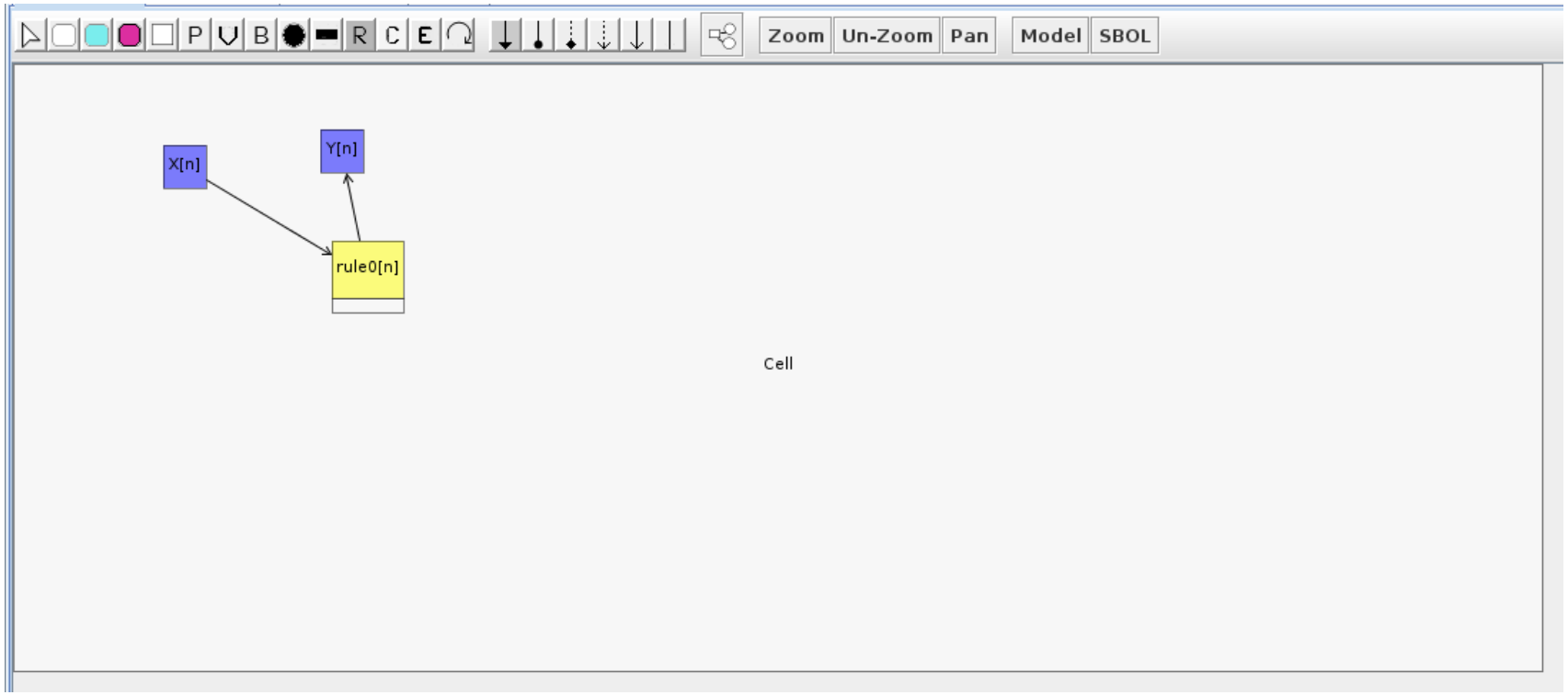
iBioSim and Arrays package (Creating Array Y)



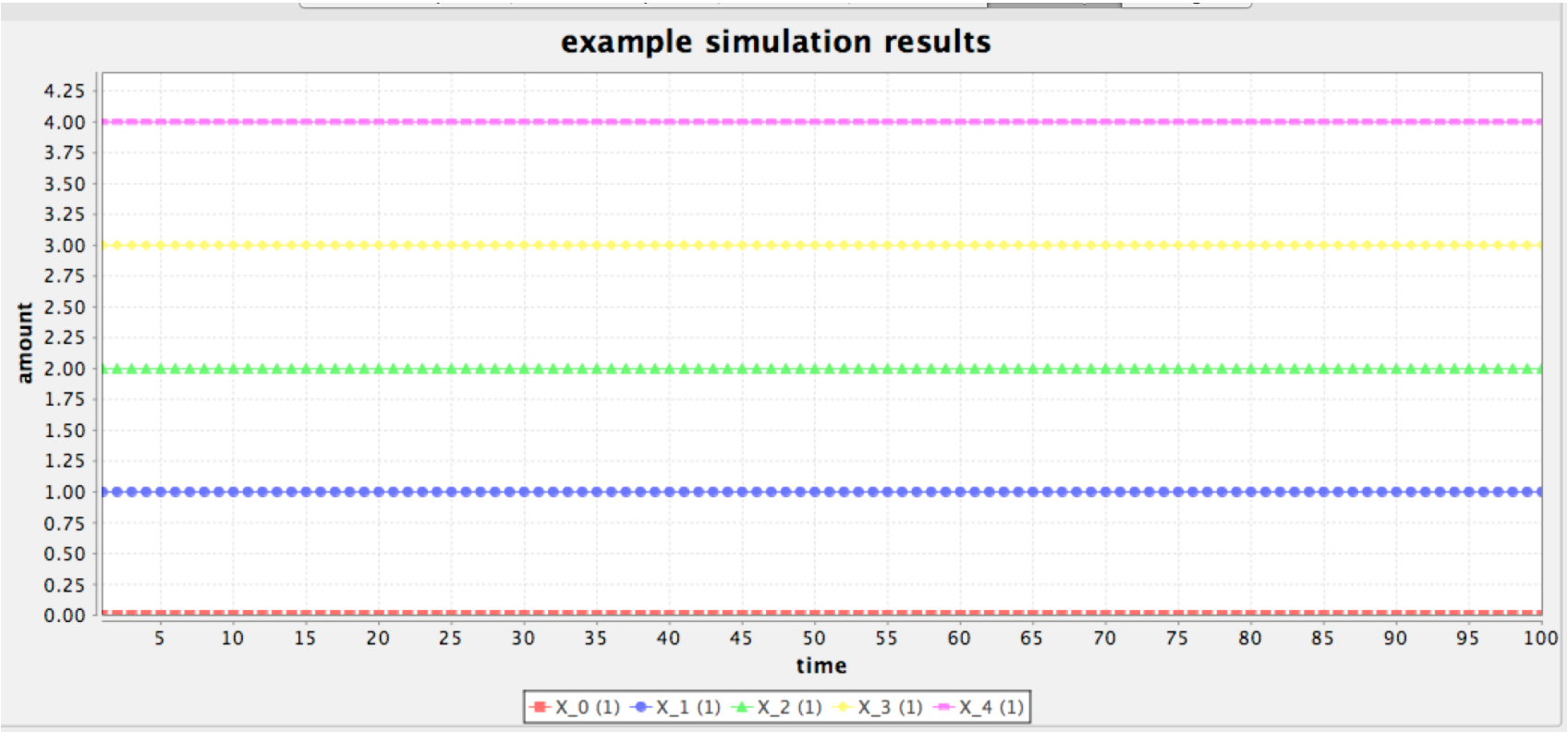
iBioSim and Arrays package (Creating Array of Rule)



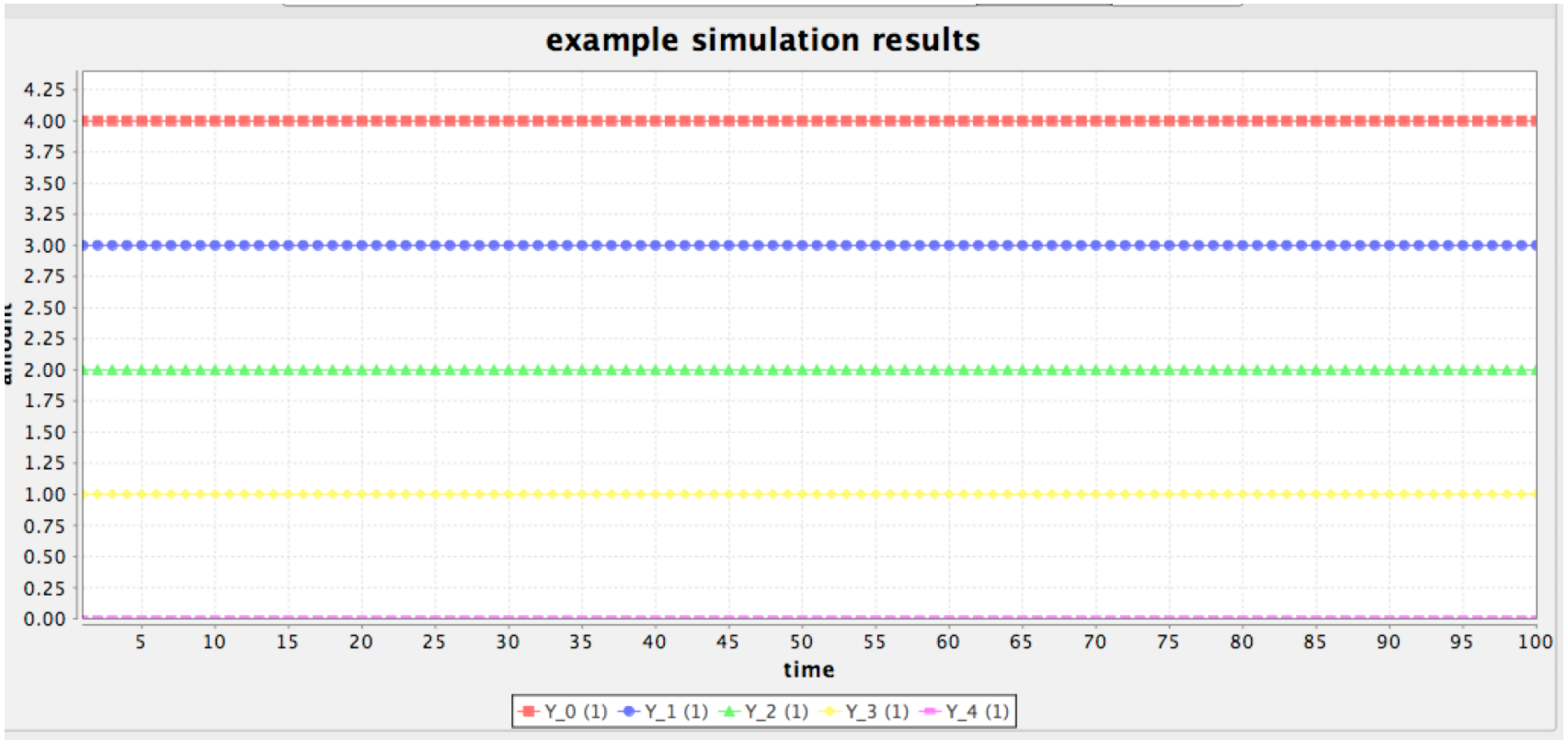
iBioSim and Arrays package (Full Model)



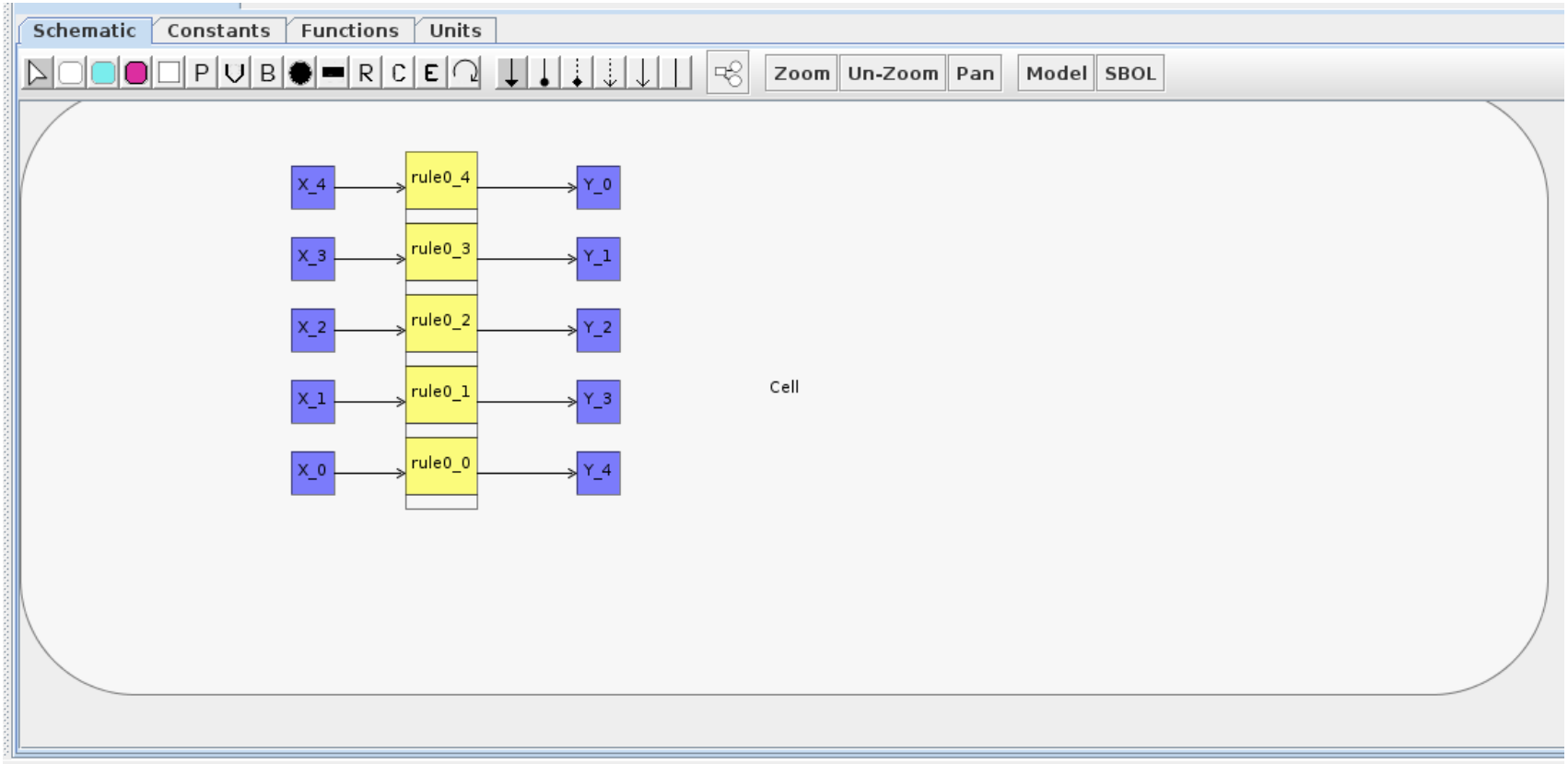
iBioSim Simulation Results



iBioSim Simulation Results (cont.)



Flattened Model



Challenges

- Arrays package is not as easy as it seems.
- Implemented a compiler for evaluating vector operations.
- Determining whether vector operations are valid is not easy.
- Event and reactions need a special case for validation and flattening.
- Trigger/Delay/Priority, if vector, need to match the parent's dimensions.
- The same applies for Kinetic Law and Reaction.
- Event's dimension ids can appear in the math of the children.
- The same applies for Reaction.
- Species References need implicit dimensions.

Discussion

- Should we allow arrays of size 0?
- Is having implicit dimensions on species references acceptable?
- Is it acceptable to have every dimension explicit other than for species references?
- Should we support more MathML operators?
- Should functions be allowed to have dimensions?

Summary

- JSBML has arrays support.
- Project has helped solidify the specification.
- JSBML has a routine that checks whether an arrayed model is valid.
- Tools can “simulate” arrayed models using the flattening routine.
- Need other tools to implement the Arrays package.

Future Work

- More testing.
- Extend flattening to work with other packages.
- Implement a simulator.
- Use cases.

Acknowledgments

Nico Rodriguez

Chris Myers

Sarah Keating

Andreas Dräger

Scott Glass

Google This work is supported by the Google Summer of Code Program.

O|B|F This work is organized by the Open Bioinformatics Foundation.