CSML 3.0

-Cell System Markup Language-

Basic Concept and Specification

19/Sep/2006

Release Candidate 1

This document edited by Euna Jeong, Ayumu Saito, and Masao Nagasaki

CSML Project Team:

Masao Nagasaki, Euna Jeong, Atsushi Doi, Ayumu Saito, and Satoru Miyano

CONTENTS

Preface

Notation

Notation in CSML3.0

(Local XML tag and attribute rule)

All tags and attributes start from lowercase letter. If a tag is a composed word where the words are joined without spaces, and each word from the second one is capitalized within the compound, e.g. `"simulationProperty"`. For abbreviated words composed of all uppercase letters e.g. `"ID"`, these words are left in uppercase, e.g. `"refID"`.

One attribute can take at most one value in XML. However, some attributes may take more than one argument. For the case, in CSML3.0, one special value is generated by joining set of arguments with semicolon, e.g. if a refID has two arguments e1 and e2, the value becomes `"e1;e2"`.

Notation in the Document

The word enclosed with angle brackets is XML tag, e.g. `<tag>`. For many cases, a double quoted word is an XML attribute, e.g. `"value"`. A word with Arial style is an XML value, e.g. mRNA.

1 Introduction

Currently, some XML formats are proposed to be a standard format for biopathways. However all formats provide only a partial solution for the storage and integration of biological data. The aim of CSML3.0 is to create a really usable XML format for visualizing, modeling and simulating biopathways.

For many cases, *in vivo/vitro* biological experimental results and *in silico* analyzed results are useful information for biopathway analysis. A successful application is Cytoscape, which can combine *in vivo/vitro* and *in silico* analyses into one graphical network. The core application supports a text-based and a GML formats. Plugins for importing XML format are developed, however, the functionality is limited. In addition, the application just visualizes the biopathway related data but dynamic simulation part is missing.

Other XML formats, SBML2.0 and CellML1.0 are proposed and developed for dynamic simulation. These formats have become popular for chemical reactions and many applications support them as data exchanging formats. However, these formats do not define any graphical elements, which causes a difficulty to be a powerful data exchange format among biopathway applications.

Here, CSML3.0 is proposed as an integrated/a unified data exchange format which covers widely used data formats and applications, e.g. CellML1.0, SBML2.0, BioPAX, and Cytoscape. In CSML1.9 and CSML2.0, the main focus was to support Hybrid Functional Petri net (HFPN) based visualization and simulation. CSML3.0 has focused on HFPNe architecture, extended HFPN with object notion, for more advanced biopathway representation. In short, objects that construct biopathways are treated as “generic entity” of HFPNe architecture and any relations among objects are treated as “generic process” on the HFPNe architecture.

To assist the dynamics of simulation from biological information, constraints of pathways can be descrbied from CSML3.0, e.g. first order logic, temporal logic and algebraic rule. These description will contribute to the model checking of the biopathway.

The advance is not restricted to the simulation architecture and its around. In the previous CSML1.9 and CSML2.0, just one view can be assigned to one model. Instead for a biopathway more than one views will enhance the understanding the nature of the pathway. Thus, multi view concepts are introduced in the CSML3.0. In addition, as in the Cytoscape interface, selecting subnetworks from a biopathway will also improve the understanding of the pathway. Related with this, filtering concept is also introduced.

For easy semantic mapping between CSML and other ontology, all CSML elements and attributes has the theoretical grounding to the OWL based ontology. Thus, CSML format can be said one of ontologies for biopathway. With the feature, all BioPAX Level2 Version1 ontology is encapsulated into CSML3.0.

2 Cell System Markup Language 3.0

2.1 XML Namespcae Identifier for the CSML3.0

The XML namespace identifier for the complete set of CSML3.0 modules, and the elements and attributes are contained within is: http://www.csml.org/csml/version3/.

2.2 Project

2.2.1 Project: <project>

```
<project xmlns:csml="http://www.csml.org/csml/version3"
     majorVersion="3" minorVersion="0" projectID="local"
projectVersionID="">
  <model/>
  <subModelSet/>
  <globalSimulatinProperty/>
  <globalViewProperty/>
  <globalAnimationProperty/>
  <chartSet/>
  <viewSet/>
  <globalBiologicalProperty/>
  <references/>
</project>
```

In CSML3.0, one XML file corresponds to one project. In the project, just one model can be stored with multi submodels and multi views. The model can be defined in the `<model>` tag (see Section 2.3). The submodels can be defined in the `<subModelSet>` tag (see Section 2.7.1). The multi views can be defined in the `<viewSet>` tag (see Section 2.9.1). The 2D plotting properties can be defined in the `<chartSet>` tag (see Section 2.10.1). The view can be changed with updated properties among simulation steps and simulation unrelated properties, e.g. biological properties, with `<globalAnimationProperty>` (see Section 2.11.1). The simulation properties for the project can be defined in `<globalSimulationProperty>` (see Section 2.4.4). The biological properties for the project, e.g. species, can be defined in `<globalBiologicalProperty>` (see Section 2.12). The all external references in the project, e.g. publication reference, protein sequence database reference, `<references>` (see Section 2.14.1).

The attributes "majorVersion" and "minorVersion" specify the CSML version of the document. For the CSML3.0 case, "majorVersion" and "minorVersion" is 3 and 0, respectively.

In order to store a project to database and share the project among users, CSML3.0 introduces "projectID" and "projectVersionID" attributes. The "projectID" will be the unique value among all projects in a database and "projectVersionID" will be used to obtain the latest project from the database. The values of "projectID" that starts from csmldb: is reserved. The value local of the "projectID" attribute is reserved for local usage of the project. For this case, "projectVersionID" is not necessary.

CSML Ontology: Base::Project

2.3 Model

2.3.1 Model:<model>

```
<model modelID="" modelVersionID="">
  <entitySet>
    <entity>*
  </entitySet>{0,1}
  <relationSet>
    <relation>*
  </relationSet>{0,1}
  <factSet>
    <fact>*
  </factSet>{0,1}
  <importModel/>*
  <modelSimulationProperty>{0,1}
  <loggedProperty/>{0,1}
  <biologicalProperty/>{0,1}
</model>
```

The `<model>` tag is used to model the main contents of biological pathway for a project. The `<model>` takes of six child tags `<entitySet>` (see Section 2.3.7), `<relationSet>` (see Section 2.3.8), `<factSet>` (see Section 2.3.5), `<importModel>` (see Section 2.3.10), `<modelSimulationProperty>` (see Section 2.4.5), `<loggedProperty>` (see Section 2.6.1), and `<biologicalProperty>` (see Section 2.12).

In order to store a model to database and share the model among users, CSML3.0 introduces "`modelID`" and "`modelVersionID`" attributes. The "`modelID`" will be the unique value among all models in a database and "`modelVersionID`" will be used to obtain the latest model from the database. The values of "`modelID`" that starts from csmldb: is reserved. The value local of

the "modelID" attribute is reserved for local usage of the project. For this case, "modelVersionID" is not necessary.

CSML Ontology: Base::Model

2.3.2 Entity: <entity>

```
<entity id="" name="" type="">

  <logProperty/>{0,1}

  <entitySimulationProperty/>{0,1}

  <viewProperty/>{0,1}

  <animationProperty/>{0,1}

  <biologicalProperty/>{0,1}

</entity>
```

A content of <entity> tag is an entity that defines any object. In CSML3.0, one entity may be a biological object, e.g. mRNA, protein, ATP or a compartmental object, e.g. cytoplasm and cell. It must be noted that all entities should not always be biologically related ones, for example, non biological entity that just means primitive entity of HFPNe is acceptable.

The <entity> can hold five child tags <logProperty>, <entitySimulationProperty>, <viewProperty>, <animationProperty>, and <biologicalProperty>, to set the logging condition (see Section 2.5.1), to set the dynamic simulation condition (see Section 2.4.1), the view settings property (see Section 2.8.1), the animation condition property (see Section 2.11.3), and the biological property (see Section 2.12), respectively.

The <entity> has three attributes "id", "name", and "type". The value of "id" must be a unique value in the <project>. The "name" should be assigned human understandable words.

The <entity> tag corresponds to the Entity class in CSML3.0 Ontology. In order to specify the subclass of the Entity class, the "type" attribute is explicitly used to specify the subclass. For example in biological components of species, the mRNA subclass of the Entity class has the type named mRNA. The detail is in Section 3.2 "Element Type in CSML3.0." The value of "type" that starts from "csml-element:entity:" is reserved. For physical components of species case, the subclass Cytoplasm of the Entity class has the type named Cytoplasm. The "type" attribute is necessary for importing from and exporting to other

ontology format, e.g. BioPAX. The entity type will be also important for automatic shape assignment to each entity as in Example 19. The subclasses of Compartment are necessary for the mapping from <compartment> tag in SBML2.0 and <component> tag in CellML1.0.

Example 1

```
<entity id="e1" name="Protein" type="Protein">
  <logProperty/>
  <entitySimulationProperty/>
  <viewProperty/>
  <animationProperty/>
  <biologicalProperty/>
</entity>
```

2.3.3 Process: <process>

```
<process id="" name="" type="">
  (<connectorList>{0,1}<connector/>+</connectorList>|<connector/>)*
  <logProperty/>{0,1}
  <processSimulationProperty/>{0,1}
  <viewProperty/>{0,1}
  <animationProperty/>{0,1}
  <biologicalProperty/>{0,1}
</process>
```

The <process> can hold the same tags of <entity> tag, <logProperty>, <viewProperty>, <animationProperty>, and <biologicalProperty>, to set the logging condition (see Section 2.5.1), the view settings property (see Section 2.8.1), the animation condition property (see Section 2.11.3), and the biological property (see Section 2.12), respectively. The <process>

holds the simulation related tag <processSimulationProperty>. The contents are different from <entitySimulationProperty>, so the name is different.

The <process> tag also has <connectorList>, <connector> as its child. In this document, <entity>, <process>, and <connector> are called *element tags*.

A content of <process> is a process that defines any interaction among element tags that are referenced in the <connector> child tag. The <process> tag corresponds to the Process class in CSML3.0 Ontology. The "type" attribute of <process> specifies the subclass of the Process class in CSML3.0 Ontology. For example, a Primitive class in Process has Primitive value as its attribute. In Section 3.2 "Element Type in CSML3.0" describing the mappling rule between XML tags and the subclasses of Process in CSML3.0 Ontology. The value of "id" must be the unique value in a <project>. The "name" attribute should be assigned human understandable words.

A content of <process> is a process among entities. From HFPNe's point of view, the <entity> notion corresponds to "generic entity" and the <process> notion corresponds to the "generic process", respectively. For example, phosphorylation, interaction, translocation are relationship among entities. In the Nagasaki's research paper describes the complete HFPNe definition and its concept [xx].

In CSML1.9 and CSML2.0, connectors in the process are treated as just one set. In CSML3.0, in order to introduce orders among connectors in a process, <connectorList> tag can be assigned as the parent of a set of <connector> tags. The simple usage is shown in Example 2 and complicated usage is in Example 4 and Example 5. The requirement of <connectorList> tag and the number of connector tags of a <process> will be restricted with the value of "type" attribute of the <process>.

Example 2

```
<process id="r1" name="interaction" type="biological">

   <connector/>

   <connector/>

</process>
```

2.3.4 Connector: <connector>

```
<connector id="" name="" refID="" type="">

  <logProperty/>{0,1}

  <connectorSimulationProperty/>{0,1}

  <viewProperty/>{0,1}

  <animationProperty/>{0,1}

  <biologicalProperty/>{0,1}

</connector>
```

A <connector> tag is used to specify one of entities for a process with <process> tag. The connector is the same to the connector element in HFPNe. The connector tags can hold the same tags of <entity> and <process> tags, <logProperty>, <viewProperty>, <animationProperty>, and <biologicalProperty>, to set the logging property (see Section 2.5.1), the view setting property (see Section 2.8.1), the animation condition property (see Section 2.11.3), and the biological property (see Section 2.12), respectively. The connector tags hold the simulation related tag <connectorSimulationProperty>. The contents are different from <entitySimulationProperty> and <processSimulationProperty>, so the name is different.

The connector tags has four attributes "id", "name", "type", and "refID". The "id", "name", and "type" are the same of <entity> and <process> tags. The value of "id" must be a unique value in the <project>. The "name" should be assigned human understandable words. The "type" attribute of <connector> specifies the subclass of the Connector class in CSML3.0 Ontology. For example, a Primitive class in Connector has Primitive value as its attribute. In Section 3.2 "Element Type in CSML3.0" describing the mappling rule between XML tags and the subclasses of Connector in CSML3.0 Ontology. The "refID" is special from other element tags and used to refer to an entity in the model. If the "refID" is in <connector> tag with value of type starts from Input, the referenced element is the source of the connector. If the "refID" is in <connector> tag with value of type Output, the referenced element is the target of the connector. The role of connector with the value of attribute process, association, inhibitor, output is the same usage of normal arc, test arc, inhibitor arc, and output arc in the Petri net, respectively.

Example 3

The model describes the phosphorylation relay from phosphorylated protein Ap to B. The reaction then generates protein A and phosphorylated protein Bp.

In the <process id="r1">, the value of the attribute "type" is biological. The type is defined to have two ordered input entities with type protein. In addition, the type is

defined to have two ordered output entities with type protein. The detailed biological information of process, here phosphorylation-relay is described in <biologicalProperty> tag (see Section XX).

```
…
<entity id=″e1″ name=″Ap″ type=″protein″>
<entity id=″e2″ name=″B″ type=″protein″>
<entity id=″e3″ name=″A″ type=″protein″>
<entity id=″e4″ name=″Bp″ type=″protein″>
<process id=″r1″ name=″phosphorylationRelay″ type=″biological″>
  <connectorList>
    <connector ref=″e1″ type=″input:process:biological″/>
    <connector ref=″e2″ type=″input:process:biological″/>
  </connectorList>
  <connectorList>
    <connector ref=″e3″ type=″output:process:biological″/>
    <connector ref=″e4″ type=″output:process:biological″/>
  </connectorList>
</process>
…
```

Example 4

The model describes the complicated usage of <connectorList> tag in a <process>.

```
…
<process id=″r2″ name=″complicated process″>
   <connectorList><connector type=″input:process:biological″/>
       <connector/></connectorList>
```

```
   <connector type=″input:process:biological″/>
   <connector type=″input:process:biological″/>
   <connector type=″output:process:biological″/>
   <connectorList>
        <connector type=″output:process:biological″/>
        <connector type=″output:process:biological″/>
   </connectorList>
   <connector/>
</process>
…
```

2.3.5 Fact: <fact>

```
<fact id=″″ name=″″ refID=″″ type=″″>
(<participantList>{0,1}<participant/>+</participantList>|<participant/
>)*
  <viewProperty/>{0,1}
  <animationProperty/>{0,1}
  <biologicalProperty/>{0,1}
</fact>
```

For modeling biopathways, information does not directly relate with dynamic simulation but states the behavior of the dynamic simulation will be important for checking the behavior of the status of biopathway. In addition, just view related information will also accelerate the understanding of biopathway for users. To represent these information, CSML3.0 introduces the `<fact>` tag. The tag corresponds to Fact class in CSML3.0 Ontology. The subclass has

simulation related information that does not effect the simulation behavior, view related information, and biological information.

The fact tag can hold the part of tags of <entity> and <process> tags, <viewProperty>, <animationProperty>, and <biologicalProperty>, to set the view setting property (see Section 2.8.1), the animation condition property (see Section 2.11.3), and the biological property (see Section 2.12), respectively. The fact tags does not hold any simulation related tag. The fact tag takes the special tags <participantList> and <participant>. The usage of tags are the similar to <connectorList> and <connector> in <process> tag.

The fact tag has four attributes "id", "name", "type", and "refID". The "id", "name", and "type" are the same of <entity> and <process> tags. The value of "id" must be a unique value in the <project>. The "name" should be assigned human understandable words. The "type" attribute of <fact> specifies the subclass of the Fact class in CSML3.0 Ontology. The "refID" is special from other element tags and used to refer to an element in the model. It must be noted that the reference targe of "refID" in the <connector> is restricted to an entity.

Thus, it is possible to represent the relationship among processes. For example, it is possible to specify the firing order between two processes.

2.3.6 Participant: <participant>

```
<participant id="" name="" refID="" type="">

  <viewProperty/>{0,1}

  <animationProperty/>{0,1}

  <biologicalProperty/>{0,1}

</participant>
```

A <participant> tag is used to specify one of elements for a fact with <fact> tag. The participant tags can hold partial tags of <entity> and <process>, <viewProperty>, <animationProperty>, and <biologicalProperty>, to set the view setting property (see Section 2.8.1), the animation condition property (see Section 2.11.3), and the biological property (see Section 2.12), respectively.

The connector tags has four attributes "id", "name", "type", and "refID". The "id", "name", and "type" are the same of <entity> and <process> tags. The value of "id" must be a unique value in the <project>. The "name" should be assigned human understandable words. The "type" attribute of <fact> specifies the subclass of the Connector class in CSML3.0

Ontology. For example, a Biological class in Fact has Biological value as its attribute. In Section 3.2 “Element Type in CSML3.0” describing the mappling rule between XML tags and the subclasses of Fact in CSML3.0 Ontology. The “refID” is special from other element tags and used to refer to an element in the model.

2.3.7 Set of Entity: <etnitySet>

```
<entitySet>

  <entity>*

</entitySet>
```

The <entitySet> tag takes set of <entity> tags as its child. The order of <entity> tags does not effect the model.

2.3.8 Set of Process: <processSet>

```
<processSet>

  <process>*

</processSet>
```

The <processSet> tag takes set of <process> tags as its child. The order of <process> tags does not effect the model.

2.3.9 Set of Fact: <factSet>

```
<factSet>

  <fact>*

</factSet>
```

The <factSet> tag takes set of <fact> tags as its child. The order of <fact> tags does not effect the model.

2.3.10 Import: <importModel>

```
<importModel xlink:href="" id="">

  <selection/>*

  <filter/>*

  <entityMap/>*

  <processMap/>*

  <factMap/>*

  <elementMerge/>*

  <viewProperty/>{0,1}

  <animationProperty/>{0,1}

  <biologicalProperty/>{0,1}

</importModel>*
```

The <importModel> tag is used to import a model in other CSML3.0 project file to the current model. The imported file can be specified with "xlink:href" attribute. The "id" attribute is used to separately treat elements in the imported model from the current model. In Example 5, each of import.gon and the current model has an entity with the same id e1, however, they can be separately treated with e1 and im1 e1.

The <importModel> tag can hold the same tags of <entity>, <process>, <fact>, and <connector> tags, <viewProperty>, <animationProperty>, and <biologicalProperty>, to set the view setting property (see Section 2.8.1), the animation condition property (see Section 2.11.3), and the biological property (see Section 2.12), respectively.

Example 5

The all elements in the <model> tag of external project file "import.gon" are imported to the project. It is possible to create a process between an entity e1 in the current model and an entity in the imported model im1 e1.

```
<project>

<model>
```

```
<entity id="e1" name="entity" type="protein"/>

<process id="r1" type="biological">

  <connector id="c1" name="input connector" refID="e1"
type="input:process:biological"/>

  <connector id="c2" name="output connector" refID="im1 e1"
type="output:process:biological"/>

</process>

<importModel xlink:href="import.gon" id="im1" />

</model>

</project>
```

For some cases, part of elements in the imported project will be used in the current project. This can be achieved with `<selection>` and `<filter>` tags as in Example 6. The usage of `<selection>` and `<filter>` is described in Section 2.7.3.

Example 6

The e1, e2, and e3 elements in the `<model>` tag of external project file import.gon are imported to the project.

```
<project>

<model>

<importModel id="im1" xlink:href="import.gon"/>

<selection>

   <element refID="e1;e2;e3"/>

</selection>

</model>

</project>
```

2.3.10.1 Map: <entityMap> <processMap>

```
<entityMap fromRefID="" toRefID=""/>

<processMap fromRefID="" toRefID=""/>

<factMap fromRefID="" toRefID=""/>
```

The above case, imported elements are treated as different elements from other elements in <entityList> and <processList> tags. In addition, the properties of imported elements cannot changed.

In CSML3.0, an imported entity, process, fact can be mapped to an entity, process, fact in the <entity>, <process>, <fact> with <entityMap>, <processMap>, <factMap> tag, respectively. With these tags, properties of imported elements can be changed in the model. In addition, the same entity between imported and editing model can be correctly treated as the same entity. The "fromRefID" references to the value of the "id" in the imported entity and the "toRefID" references to the value of the "id" in the <entity>. The same is true to the <processMap> and <factMap> tags. It must be noted that if a process is mapped with <processMap>, all connected entities should be also mapped with <entityMap>. For similarity, if a fact is mapped with <factMap>, all participants should be also mapped with <entityMap>, <processMap>, or <factMap>. The mapping of connector is not supported in CSML3.0 for avoiding complexity. Thus, if a <fact> contains a reference to a connector in a <participant> tag, the fact cannot map to any imported <fact>. For shorter notation of mapping entities, processs, and participant with <entityMap>, <processMap>, and <factMap> the value of "fromRefID" and "toRefID" can be joined with ";" as ex1;ex2 in Example 7.

Example 7

The e1 and e2 entities in the imported model are treated as ex1 and ex2 entities in the project, respectively. The properties of entities e1 and e2 can be changed in the child tags of <entity ID="ex1"> and <entity ID="ex2">.

```
<project>

<model>

   <entity ID="ex1"/>

   <entity ID="ex2"/>

<importModel xlink:href="import.gon"/>

<selection>
```

```
   <element refID=”e1;e2;e3;rx2”/>

</selection>

<entityMap fromRefID=”e1;e2” toRefID=”ex1;ex2”/>

<processMap fromRefID=”r1” toRefID=”rx2”/>

</model>

</project>
```

2.3.10.2 Merge: <elementMerge>

```
<elementMerge type=”merge:”>

   <parameter key=”” value=”” type=””/>*

</elementMerge>
```

For advanced mapping from importing model to editing model, the <elementMerge> can be used. The <elementMerge> has “type” attribute to specify the merge algorithm. The child tag <parameter> sets the options that are used for the merge operation. Thus, the number of <parameter> tags and their values of “key” attributes depend on the value of “type” in <elementMerge>. The behaviors of merging algorithms are defined in Section 3.14 “Merging Specification in CSML3.0.”

2.4 Simulation Property

In CSML3.0, all simulation related properties should be defined in the tags that ends with `simulationProperty`, e.g. `<globalSimulationProperty>`, `<entitySimulationProperty>`, `<processSimulationProperty>`, and `<connectorSimulationProperty>`. In CSML3.0, an element in the model has more than one view. In contrast, just zero or one simulation parameters can be assigned to each element. Still, user can create two models with different kinetics for the same biopathway network by using submodel concept in CSML3.0, i.e. create two elements with different kinetics in main model and by choosing the different element with different kinetic in each submodel. In order to modify the simulation parameters of the main model, the other CSML derived format, shortly mentioned in Section 4.1 "CSML Simulation Updating Specification", can be used. This format is separately developed as the CSML Updater XML.

2.4.1 Entity: <entitySimulationProperty>

```
<entity>

   <entitySimulationProperty>

     <variable/>

   </entitySimulationProperty>

</entity>
```

The `<entitySimulationProperty>` takes exactly one `<variable>` tag as its child and specifies the simulation property of entity.

2.4.1.1 Variable: <variable>

```
<variable id="" name="" type="">

       <parameter key="" value="" type=""/>*
```

```
      <script key="" language="" type=""/>*

    </variable>
```

The <variable> tag has "id", "name", and "type" attributes. The value of "id" should be the unique value in the <project>. The "name" should be assigned human understandable words. The "type" is used for specifying the variable type. The variable type is defined in Section 3.4 "Entity Object Type in CSML3.0". The <variable> tag has two child tags <parameter> and <script> for setting the parameters of variable. The number of parameters and their values of "key" attributes depend on the "type" of <variable>. The "type" attribute in the <parameter> specifies the type of parameter. The attribute is optional because the "type" will be apparent from the "type" of <variable> and the "key" of <parameter>. The value of "type" can be selected from the value in Section 3.3 "Primitive Type in CSML3.0". In CSML3.0, predefined keys are summarized in Section 3.5 "Variable Parameter in CSML3.0". The <script> tag is used to specify a complicated evaluatable script as the variable parameter. There may be multiple script tags for various parameters of the variable, e.g. initial value, minimum value. Therefore the <script> tag has the key attribute to denote the parameter for which the script is defined.

Example 8

```
<entity id="e1" name="mRNA" type="mRNA">

  <entitySimulationProperty>

    <variable id="v1" name="mRNA variable"
type="csml-object:mRNA<double>" >

      <parameter key="csml-variable:parameter:initialValue" value="0"/>

      <parameter key="csml-variable:parameter:evaluateScriptOnce"
value="true"/>

      <parameter key="csml-variable:parameter:minimumValue" value="0"/>

      <parameter key="csml-varaible:parameter:maximumValue"
value="infinite"/>

      <parameter key="csml-variable:parameter:unit"
value="dimensionless"/>

    </variable>
```

```
  </entitySimulationProperty>

</entity>

<entity id=”e2” name=”protein” type=”protein”>

  <entitySimulationProperty>

   <variable id=”v2” name=”protein variable”
type=”csml-object:protein<double>”>

     <parameter key=”csml-varaible:parameter:evaluateScriptOnce”
value=”true”/>

     <parameter key=”csml-variable:parameter:minimumValue” value=”0”/>

     <parameter key=”csml-varaible:parameter:maximumValue”
value=”infinite”/>

     <parameter key=”csml-variable:parameter:unit”
value=”dimensionless”/>

     <script key=”csml-variable:parameter:initialValue”
language=”pnuts”>Math.random()*10</script>

    </variable>

  </entitySimulationProperty>

</entity>
```

2.4.2 Process: <processSimulationProperty>

```
<process>

<processSimulationProperty>

   <variable/>*

   <priority/>{0,1}

   <firing/>{0,1}

   <delay/>{0,1}

   <kinetic/>{0,1}

</processSimulationProperty>

</process>
```

The <processSimulationProperty> takes HFPNe simulation related child tags <variable> (see Section 2.4.2.1), <priority> (see Section 2.4.2.2), <firing> (see Section 2.4.2.3), <delay> (see Section 2.4.2.4), and <kinetic> (see Section 2.4.2.5).

```
<variable id="v1" name="mRNA variable" type="csml-object:mRNA<Double>" />

   <parameter key="initialValue" value="0"/>

   <parameter key="minimumValue" value="10"/>

   <parameter key="maximumValue" value="infinite"/>

</variable>
```

2.4.2.1 Variable: <variable>

All attributes of <variable> tag in <processSimulationProperty> is the same to the <variable> tag in <entity>. The difference is the scope of the variable. The <variable> in the <processSimulationProperty> can be accessed just in the <processSimulationProperty> tag. In other words, the variable is like local variable for the

relation while <variable> in <entity> is global in a project. The number of <variable> can be plural, i.e. for one relation, more than one local variables can be defined.

2.4.2.2 Priority: <priority>

```
<priority order=""/>
```

The <priority> tag specifies the firing order among all fired relations. The value of attribute "order" is integer. The bigger the value of "order" is, the higher the priority is.

2.4.2.3 Firing: <firing>

```
<firing value="" firingStyle="csml-firingStyle:and|or|rule"
firingOnce="true|false" ruleValue="">

  <script type="value">

  </script>{0,1}

</firing>
```

The <firing> tag specifies the activity of the relation. If the evaluated result of the value of attribute "value" is true, the relation satisfies partial condition to be active. For complicated script description, e.g. script with multiple lines, <script type="value"> can be used. For full activity, all connectors must satisfy the condition that is defined with the attribute "firingStyle". The attribute can takes one of three values, "csml-firingStyle:and", "csml-firingStyle:or", "csml-firingStyle:rule". The value starts from "csml-firingStyle:" is reserved for future extension. If the value is "csml-firingStyle:and", for the full activity of relation, all connectors should be active status. If the value is "csml-firingStyle:or", for the full activity of relation, just one connector is enough to be active. If the value is "csml-firingStyle:rule", all connectors and the relation should satisfy the boolean operation that can be defined in the attribute "ruleValue", e.g. (c1 & c2) | (c3 & c4). The value of "ruleValue" is composed of the "id" attributes of connectors in the relation and &, |, (, and). The & is boolean "and operation". The | is boolean "or operation". The (and) set the evaluation order of the value. When the current value of "ruleValue" is evaluated, each "id" part is assigned with the current firing state of the connector of "id". For the above example, if c1=false, c2=false, c3=true, and c4=true, the evaluated result is true. The "firingOnce" attribute takes true or false values. If the default value is true, if the value is false, the relation is once in active state, the relation can not be active in future.

2.4.2.4 Delay: <delay>

```
<delay value="" delayStyle="nodelay|delay">
```

```
  <script type="value">

  </script>

</delay>
```

The <delay> tag specifies the delaying time of a relation after the full activity of the relation. The delaying time can be assigned with the value of the attribute "value". As <firing> case, <script type="value"> can be used for complicated script description. The attribute "delayStyle" has one of two values, nodelay and delay. If it is nodelay, the reaction acts with no delay.

2.4.2.5 Kinetic: <kinetic>

```
<kinetic calcStyle="speed|add|update|check" kineticStyle="" fast="">

  <parameter name="" value="" type="">

     <script type="value">

     </script>

  </parameter>*

</kinetic>
```

The <kinetic> tag specifies how to modify the values of entities that are connected to the relation for each reaction.

The attribute kineticStyle defines the kinetics of the relation. The value of the attribute always starts from csml-kineticStyle:, e.g. csml-kineticStyle:custom, csml-kineticStyle:mass. The defined value is Section 3.8 "Kinetic Style in CSML3.0". Depending on the value, the needed <parameter> tags are automatically defined, e.g. the number of child tag <parameter> and their values of "name" attributes are automatically specified. The "fast" attribute has a value, true or false. If the value is true, it means that the reaction speed is relatively faster than other reactions. In biopathway simulation, some reaction speed is much faster than other reaction, e.g. the speed of transcription is usually much slower than that of ion signaling via ion channel. It might be used for simulators for accurate simulation.

The attribute "calcStyle" takes one of four values, speed, add, update. The "calcStyle" is used to define how to use the evaluated result of kineticStyle. If the value is speed, the values of entities are updated with sampling interval * evaluated result. If the value is add,

the entities values are updated with evaluated result. If the value is update, the entities values are related with evaluated result.

Example 9

Translation reaction of mRNA "e1" to Protein "e2" with mass kinetics.

```
<entity id="e1" name="mRNA" type="mRNA">
  <entitySimulationProperty>
    <variable id="m1" type="double">
        <parameter key="initialValue" value="1"/>
        <parameter key="minimumValue" value="0"/>
        <parameter key="maximumValue" value="infinite"/>
        <parameter key="unit" value="dimensionless"/>
    </variable>
  </entitySimulationProperty>
</entity>
<entity id="e2" name="Protein" type="protein">
  <entitySimulationProperty>
    <variable id="m2" type="double">
        <parameter key="initialValue" value="0"/>
        <parameter key="minimumValue" value="0"/>
        <parameter key="maximumValue" value="infinite"/>
        <parameter key="unit" value="dimensionless"/>
    </variable>
  </entitySimulationProperty>
</entity>
```

```
<process id=”r1” name=”translation” type=”biological”>
  <connector refID=”e1” type=”input:association:biological”/>
  <connector refID=”e2” type=”output:process:biological”/>
  <processSimulationProperty>
     <priority order=”0”/>
     <firing value=”true” firingStyle=”and”/>
     <delay value=”0” delayStyle=”nodelay”/>
     <kinetic calcStyle=”speed” kineticStyle=”csml-kineticStyle:mass”
first=”true”>
        <parameter key=”coefficient1” value=”1”/>
        <parameter key=”coefficient2” value=”0.1”/>
     </kinetic>
  </processSimulationProperty>
</process>
```

Example 10

Translation reaction of mRNA e1 with variable type csml-object:Protein<Double> to Protein e2 with variable type csml-object:Protein<Double> with method translation.

```
<entity id="e1" name="mRNA" type="mRNA">
  <entitySimulationProperty>
    <variable id="m1" type="csml-object:mRNA<Double>">
        <parameter key="initialValue" value="1"/>
        <parameter key="minimumValue" value="0"/>
        <parameter key="maximumValue" value="infinite"/>
        <parameter key="unit" value="dimensionless"/>
    </variable>
  </entitySimulationProperty>
</entity>
<entity id="e2" name="Protein" type="protein">
  <entitySimulationProperty>
    <variable id="m2" type="csml-object:Protein<Double>">
        <parameter key="initialValue" value="0"/>
        <parameter key="minimumValue" value="0"/>
        <parameter key="maximumValue" value="infinite"/>
        <parameter key="unit" value="dimensionless"/>
    </variable>
  </entitySimulationProperty>
</entity>
<process id="r1" name="translation" type="biological">
```

```
  <connector refID=”e1” type=”input:association:biological”/>
  <connector refID=”e2” type=”output:process:biological”/>
  <processSimulationProperty>
     <priority order=”0”/>
     <firing value=”true” firingStyle=”and”/>
     <delay value=”0” delayStyle=”nodelay”/>
     <kinetic calcStyle=”csml-calcStyle:update”
kineticStyle=”csml-kineticStyle:method”>
        <parameter key=”method” value=”translation”/>
        <parameter key=”k1” value=”0.1”/>
     </kinetic>
  </processSimulationProperty>
</process>
```

2.4.3 Connector:<connectorSimulationProperty>

```
<connector>

<connectorSimulationProperty>

   <variable/>*

   <firing/>{0,1}

   <kinetic>

     <parameter name="" value=""/>*

   </kinetic>{0,1}

</connectorSimulationProperty>

</connector>
```

The <connectorSimulationProperty> takes HFPNe simulation related child tags, <variable>(see Section 2.4.3.1), <connectorFiring>(see Section 2.4.3.2), and <kinetic> (see Section 2.4.2.5).

2.4.3.1 Variable: <variable>

```
<variable id="" name="" type="" value=""/>
```

The usage is the same to <variable> tag in Section 2.4.2.1. As <variable> in the <reactionSimulationProperty> tag, the scope of <variable> is local for the <connector> section like that of <process>. The number of <variable> can be plural.

2.4.3.2 Connector Firing: <connectorFiring>

```
<connectorfiring
connectorFiringStyle="csml-connectorFiringStyle:threshold|nocheck|rule
">

  <parameter name="" value=""/>*
```

```
  <script>

  </script>{0,1}

</connectorfiring>
```

The <connectorFiring> section is used to determine the activity of the connector. The attribute "connectorFiringStyle" specifies how to check the firing state of the connector. The value is predefined in Section 3.14 and the predefined value always starts from csml-connectorFiringStyle:. Depending on the value, the number of child tag <parameter> and their name are automatically specified. If the "connectorFiringStyle" is csml-connectorFiringStyle:rule, <parameter> tags are skipped and the script in the <script> tag is evaluated for deciding whether the connector can fire or not.

2.4.4 Global: <globalSimulationProperty>

In many cases, model will be a dynamic simulatable one, the global options of simulation are defined in the <globalSimulationProperty> tag.

```
<project>

…

  <globalSimulationProperty>

    <unitSet/>*

    <functionSet/>*

  </globalSimulationProperty>

</project>
```

The <globalSimulationProperty> takes global HFPNe simulation related child tags <unitdefs> (see Section エラー! 参照元が見つかりません。) and <functionSet> (see Section 2.4.4.2).

2.4.4.1 Unit: <unitdefs><newUnit><unit>

The physical unit definition that is mainly related with dynamic simulation is available in the <unitSet> tag. The definition of unit is exactly the same to the <unitSet> tag in CSML1.9.

```
<unitSet>

   <units />*

</unitSet>
```

The <unitSet> element may contain zero or more <newUnit> elements that are the containers for the <unit> element. The <newUnit> tag may contain two attributes: symbol (the symbol of the defined unit) and base units (if present and true, this unit is a new base unit, not derived from any other units; if a unit is defined as a new base unit, inner unit elements are disallowed; assumed false if not present).

```
<newUnit>
```

```
   <unit>*

</newUnit>
```

The unit element defines each base element of a new physical unit measured in units element. The <unit> tag may contain five attributes: units (name of the unit being referred to), prefix (SI decimal scale prefix, if not present, unity scale (1.0) is assumed), exponent (unit dimension exponent, if not present, 1.0 is assumed), multiplier (a multiplier attribute can be used to pre-multiply the quantity to be converted by any real scale factor; for instance, a multiplier of 1.8 is used to define Fahrenheit in terms of Celsius; if not present, 1.0 is assumed), and offset (the offset attribute is used to represent the addition of a constant in the transformation between the current units and the base units; this should only be necessary for the definition of temperature scales; for instance, an offset attribute value of 32.0 is needed to define Fahrenheit in terms of Celsius; if not present, 0.0 is assumed; an offset attribute cannot appear in complex unit definitions - that is, having more than one unit element).

Example 11

```
<unitSet>

   <newUnit name="pascal" symbol="Pa">

      <unit units="newton"/>

      <unit units="metre" exponent="-2"/>

   </newUnit>

   <newUnit name="inch" symbol="in">

      <unit units="metre" multiplier="0.0254"/>

   </newUnit>

</unitSet>
```

2.4.4.2 Function: <function><functionSet>

```
<functionSet>

  <function>

    <script language="csml-scriptLanguage:mathml|pnuts">
```

```
    </script>

  </function>*

</functionSet>
```

The <functionSet> specifies global functions in the project. One <function> corresponds to one global function. The <function> tag has a <script> tag as a child with attribute "language", the value of "language" is in Section 3.15 "Script Language in CSML3.0." If a function is used in the <script> tag of other function, the referenced function should appears in advance.

2.4.5 Model: <modelSimulationProperty>

```
<model>

  ...

  <modelSimulationProperty>

     <parameter key="" value="" type=""/>

  </modelSimulationProperty>

  ...

</model>



<subModel>

  ...

  <modelSimulationProperty>

     <parameter key="" value="" type=""/>

  </modelSimulationProperty>{0,1}

  ...

</subModel>
```

In many cases, model will be a dynamic simulatable one, the simulation options of models are defined in the <modelSimulationProperty> tag. The tag contains <parameter> tag as its child. The "key" is the key of the parameter and "value" is the value of the key. The type of the key can be defined with "type". Some keys are already defined in CSML3.0. They are defined in Section 2.4.4 "Global Simulation Options in CSML3.0". These reserved keys always start from csml-simParameter:.

The <modelSimulationProperty> tag can be the child of <model> and <subModel> tags. All simulation options in <model> tag is inherited to <subModel> tag. If the same option is defined in both tags, the setting in the <subModel> is used.

Example 12

The same simulation option csml-simParameter:samplingInterval is defined in <model> and <subModel> tag. In the <subModel>, the value of sampling interval is overrided and 0.1 is used.

```
<model>

  <modelSimulationProperty>

    <parameter key="csml-simParameter:enhancedFiring" value="true"
type=":boolean"/>

    <parameter key="csml-simParameter:samplingInterval" value="0.1"
type="double"/>

    <parameter key="csml-simParamter:logUpdateInterval" value="1.0"
type="double"/>

    <parameter key="csml-simParamter:plotUpdateInterval" value="1.0"
type="double"/>

  </modelSimulationProperty>

</model>

<subModelSet>

  <subModel>

    <parameter key="csml-simParameter:samplingInterval" value="0.1"
type="double"/>

  </subModel>
```

```
</subModelSet>
```

2.5 Log Property

2.5.1 Log Property: <logProperty>

```
<entity>

  <logProperty>{0,1}

</entity>

<process>

  <logProperty>{0,1}

</process>

<connector>

  <logProperty>{0,1}

</connector>



<logProperty>

  <property/>+

</logProperty>
```

Elements, i.e. <entity>, <process>, and <connector> in the model can set the flag of logging and if the flag is on, then the status of element can be logged among simulation steps.

2.5.2 Property: <property>

```
<property key="csml-accessibleProperty:" value="true|false" type="" />+
```

The log targets of an element can be selected from accessibleProperty of the element in Section 3.7. The accessibleProperty is specified in the "key" attribute of <property> tag. The tag also has the attribute "value" with boolean value. If the value is "true", the attribute is logged.

2.6 Logged Property

2.6.1 Logged Property: <loggedProperty>

```
<model>

  <loggedProperty>{0,1}

</model>



<subModel>

  <loggedProperty/>{0,1}

</subModel>



<loggedProperty>

  <property/>*

  <refLogElement/>+

  <logDataList/>+

</loggedProperty>
```

CSML3.0 allows for the results of simulation to be logged for the model and submodels in a project. To deal with this, the `<model>` and `<subModel>` tags can contain a `<loggedProperty>` tag as its child. The `<loggedProperty>` tag consists of `<property>` (see Section 2.6.2), `<refLogelement>` (see Section 2.6.3), and `<logDataList>` tags (see Section 2.6.4).

2.6.2 Property: <property>

```
<property key=”csml-logProperty:” value=”” type=”” />
```

The <property> tag is used to set properties of the log, e.g. logged date , and logged status. The predefined keys for log in CSML3.0 always start from csml-logProperty:. The keys are summarized in Section 3.12 “Predefined Log Properties in CSML3.0”.

2.6.3 Reference Log Element: <refLogElement>

```
<refLogElement logID="" refID="" refProperty=""/>
```

The <refLogElement> tags specify which elements are logged on the chart. The “refID” attribute is used to specify the referenced element and “refProperty” is used to specify which property of the element is logged on the chart. The acceptable values are described in Table 7.

2.6.4 Log Data List: <logDataList>

```
<logDataList timePoint="">

  <logData/>+

</logDataList>
```

The logged data is stored in the <logDataList> tag. The tag has an attribute “timePoint” to inform how many logged data points are stored.

2.6.5 Log Data and Value: <logData> <logValue>

```
<logData timePoint="">

  <logValue refLogID="" value="">+

</logData>
```

One <logData> corresponds to one logged data of a time point that is specified with “timePoint” attribute of the tag.

<logValue> contains “refLogID” and “value” attributes. The “refLogID” refers to a one “logID” in the <refLogElement>. The “value” stores the value of referenced element at the time point of <logData>. For shorter notation, the “refLogID” and “value” can take multiple logged element with ; as in Example 13.

Example 13

Normal Notation of <logData>

```
<logData timePoint=”100”>

  <logValue refLogID=”1” value=”10”>

  <logValue refLogID=”2” value=”15”>

  <logValue refLogID=”3” value=”30”>

</logData>
```

Shorter Notation of <logData>

```
<logData timePoint=”100”>

  <logValue refLogID=”1;2;3” value=”10;15;30”>

</logData>
```

When a given value doesn't change from one time point to the next one, it can be omitted in the log as in Example 14.

Example 14

```
<loggedProperty>

  <property key=”csml-logProperty:filename” value=”sim12.cil”/>

  <property key=”csml-logProperty:status” value=”finished”/>

  <property key=”csml-logProperty:timeStamp” value=”Dec 2, 2006 2:06:39
PM”/>

  <property key=”csml-logProperty:server” value=”localhost”/>

  <refLogElement refID="e2" logID="1"
propertyType="csml-accessibleProperty:simulation:currentValue"/>

  <logDataList>

   <logData timePoint="0">

    <logValue logID="0" value="0.0"/>
```

```
   <logValue logID="1" value="10.0"/>
  </logData>
  ...
  <logData timePoint="1999000000"/>
  <logData timePoint="2000000000"/>
 </logDataList>
 <comments><comment type=”text”>test</comment></comments>
</loggedProperty>
```

2.7 SubModel Property

2.7.1 SubModelSet: <subModelSet>

```
<subModelSet>

   <subModel/>+

</subModelSet>
```

The <subModelSet> tag contains more than one <subModel> as its child. Each <subModel> corresponds to a subModel in the <model> tag, e.g. PN, GN, BN, PPI.

All processs should be defined in the <model> tag in advance. In other words, no new process can not add in the <subModel> section.

2.7.2 SubModel: <subModel>

```
<subModel modelID="" name="">

  <selection/>* | <filter/>*

  <loggedProperty/>

</subModel>
```

One <subModel> corresponds to one special model, e.g. Petri net model, gene network model, boolean network, and protein-protein interaction model. The tag has "modelID" and "name" attributes. All <subModel> should have a unique value as its "modelID". The "name" attribute should be human readable words. The <subModel> contains <selection> (see Section 2.7.3), <filter> (see Section 2.7.4), and <loggedProperty> (see Section 2.6.1) tags.

2.7.3 Selection: <selection>

```
<selection>

  <refElement refID="xx" includeChild="true|false" />*

 </selection>
```

The <subModel> contains more than one <selection> and <filter> tag as its child. In the <selection> tag, <refElement> tags are listed up to specify which elements are used in the <subModel>. A <selection> tag refers to an element in the <model> with "refID" attribute if the value of "includeChild" is flase. If the "includeChild" is true and the referenced element is <process>, all connected entities are also selected and all connected processs are also recursively selected into the model. The value of "refID" must be equal to the "id" attribute of the target <refElement> or a set of them that is joined with ";". The second notation is useful for reducing the file size of the CSML project.

2.7.4 Filter: <filter>

```
<filter type="xx">

  <parameter key="xx" value="yy"/>*

  <filter/>

  <selection/>

</filter>
```

<filter> tag can be used to select elements in the <model> with a customized condition, e.g. select all processs with interaction as their "refMoleculeRole" attribute of <biologicalProperty> tag and all entities connected with them. The <filter> can take <filter> and <select> tags as its child for recursive selections. In Section 3.14 "Filtering Specification in CSML3.0", the behavior of each filter is defined.

Example 15

The model1 and the model2 are the same submodels. The difference between them is that the model1 is defined with <select> tag and the model2 is with <filter> tag.

```
 <model>
```

```
   <entity id=”e1”/><entity id=”e2”/><entity id=”e3”/>
   <process id= ”r1” type=”csml-element :process:interaction”>
      <connector refID=”e1” type=”input:process:biological”/>
     <connector refID=”e2” type=”input:process:biological”/>
     <connector refID=”e3” type=”output:process:biological”/>
   </process>
   <biologicalProperty refMoleculeRoleID=”interaction”/>
 </model>

 <subModelSet>
  <subModel modelID=”model1” name=”Protein-Protein Interaction with
select”>
    <selection>
      <refElement refID=”e1;e2;e3;r1”>
    </selection>
  </subModel>
  <subModel modelID=”model2” name=”PPI with filter”>
    <filter type=”csml-filter:biologicalProperty”>
      <parameter key=”moleculeRoleID” value=”interaction”/>
      <parameter key=”selectionType” value=”match”/>
    </filter>
  </subModel>
 </subModelSet>
```

2.8 View Property

Each element has two types of view, "shape" and "position". In CSML3.0, they are explicitly defined with separate tags for a better reusability of multi-views in <ViewSet> tag (see Section 2.9.1). The view of each element can be specified with two manners. First is to directly set each element's view in the <viewProperty> of element tags (see Section 2.8.1). Second is to apply a set of elements in the model with the same view with global view settings in <globalViewProperty> (see Section 2.8.4).

2.8.1 View Property: <viewProperty>

```
<entity|process|connector|fact|importModel>

  <viewProperty>

     <position/>*

     <shape/>*

  </viewProperty>{0,1}

</entity|process|connector|fact|importModel>
```

CSML3.0 supports to define multi-view in <viewProperty> as child <entity>, <process>, <connector>, <fact>, and <importModel> tags. The <viewProperty> has two child tags, <position> (see Section 2.8.2) and <shape> (see Section 2.8.3). The order of <position> and <shape> tags in <viewProperty> is this order.

2.8.2 Position: <position>

```
<position positionID="" x="" y="" position="static|auto">
```

<position> tag has four attributes. "x" and "y" attributes specify the geometrical location in views. The tag can appear plural numbers in <viewProperty> tag. Among them, "positionID" attribute should be different each other. The "position" attribute takes two

values static and auto, The attribute is related with <layout> tag in <view> (see Section 2.9.2.1). If the value is static, the element with this position does not change with the defined layout algorithm. If the value is auto, the element with this position can change with the defined layout algorithm.

2.8.3 Shape: <shape>

```
<shape shapeID="" depth="">

  <svg:svg/>

  <cache/>

</shape>
```

The <shape> specifies the graphical shape of the element, e.g. background color, line path, size, except for the base geometrical position of the shape. The geometrical position is separately defined in the <position> tag. The <shape> tag can be plural for supporting multi-views, e.g. protein-protein interaction view and gene regulatory network view. The plural tag has an attribute "shapeID" and referenced from <view> tag with "refShapeID" as in Section 2.9. The "refShapeID" attribute value should be different among <shape> tags in the same <viewProperty>. The <shape> tag has <svg:svg> and <cache> tags. The <shape> tag has the "depth" attribute for setting the Z-order (overlap depth) among <shape>s. A figure with a lower depth is displayed over a figure with a higher depth value.

Example 16

```
<entity id="e1" name="mRNA">

 <viewProperty>

  <position x="10" y="30" positionID="position1"/>

  <position x="20" y="35" positionID="position2"/>

  <shape shapeID="shape1"/>

  <shape shapeID="shape2"/>

 </viewProperty>

</entity>
```

```
<entity id="e2" name="Protein">
  <viewProperty>
    <position x="50" y="50" positionID="position1"/>
    <position x="30" y="70" positionID="position2"/>
  </viewProperty>
</entity>
```

2.8.3.1 SVG: <svg:svg>

```
<svg:svg>
</svg:svg>
```

In CSML3.0, the shape is defined in the <svg:svg> tag the sub-elements of which are basically equal to the specification of standard scalable vector graphic format SVG1.1. The complete document will be obtained from http://www.w3.org/TR/SVG/. In CSML1.9, the <shape> and <position> definitions were original format, with the update, CSML3.0 can generates views with vector graphic format much easily. It is important for creating a high resolution view for large scale pathway with small file size, e.g. printing to plotter, and for exporting and remodeling in other major graphic application. For better usability, some special tags and attributes are introduced in CSML3.0. The detail is described in Section 3.1 "Special SVG Shape Tags and Attributes in CSML3.0".

Example 17

```
<project>
<model>
<entity id="e1" name="mRNA" type="csml-element:entity:biological:mRNA">
  <viewProperty>
    <shape shapeID="shape1">
      <svg:svg>
```

```
        <svg:ellipse style="stroke:#000000;stroke-width:1.0;fill:none;"
rx="50" ry="50"/>

      </svg:svg>

    </shape>

    <shape shapeID="shape2">

      <svg:svg>

        <svg:rect width="50" height="50"
style="stroke:#000000;stroke-width:2.0;fill:none;"
transform="matrix(1.0,0,1,2,-10)"/>

      </svg:svg>

    </shape>

    <shape shapeID="shape2">

      <svg:svg/>

    </shape>

  </viewProperty>

</entity>
```

2.8.3.2 Cache: <cache>

```
<cache width="" height="" xlink:href=""/>
```

The <cache> tag has three attributes "width", "height", and "xlink:href". The "xlink:href" refers to the raster image internally or externally (see Example 18). The "width" and "height" attribute sets the width and height of the image, respectively. The cache tag assume two main usages (i) rendering speed of the scalable graphic is usually slower than that of raster image. Thus, for many case, instead of using <svg> image, using <cache> image might be used in software. (ii) The scalable graphic image handling is much difficult than raster image handling. Thus, many application may not support <svg> format, for the case <cache> image can be used. It must note that the raster image is much difficult

to edit their shape except for width and height. Thus, both tags <svg> and <cache> are important for better usability.

Example 18

```
<project>

<model>

<entity id=”e1” name=”mRNA” type=”csml-element:entity:biological:mRNA”>

  <viewProperty>

    <position x=”10” y=”30” positionID=”position1” />

    <position x=”20” y=”35” positionID=”position2”/>

    <shape shapeID=”shape1”>

       <svg:svg/>

       <cache width=”30” height=”30” xlink:href=”test.png”/>

    </shape>

    <shape shapeID=”shape2”>

       <svg:svg/>

       <cache width=”50” height=”50”
xlink:href=”
dfkljsdljsdlsdld”/>

    </shape>

  </viewProperty>

</entity>
```

2.8.4 Global View Property: <globalViewProperty>

```
<project>

  <globalViewProperty>
```

```
    <globalShape>*

  </globalViewProperty>{0,1}

</project>
```

The <globalViewProperty> section defines global view styles - especially shape styles - of model and submodels in the child tag <globalShape>. In each view, subset of view styles can be applied to create custom view.

2.8.4.1 Global Shape: <globalShape>

```
<globalShape shapeID="" name="">

  <select/>*

  <filter/>*

  <svg:svg>

    svg shape tags

  </svg:svg>

</globalShape>
```

In one <globalShape> section, elements are selected in <select> and <filter> tags and for these elements, the same custom shape are specified in <svg:svg> with svg shape related tags, e.g. <rect>, <circle>, and <line>. For the selection of elements, the same tags <select> (see Section) and <filter> (see Section) in the <subModel> are used with the same manner.

The "shapeID" attribute should set a unique value among <globalShape> tags. The "name" should be human understandable words.

Example 19

```
<project>

<model>

<entity id="e1" name="mRNA" type=" mRNA">

  <viewProperty>
```

```
    <position x=”10” y=”30” positionID=”position1” />
    <position x=”20” y=”35” positionID=”position2”/>
    <shape shapeID=”shape1”/>
    <shape shapeID=”shape2”/>
  </viewProperty>
</entity>
<entity id=”e2” name=”Protein” type=”protein”>
  <viewProperty>
    <position x=”50” y=”50” positionID=”position1”/>
    <position x=”30” y=”70” positionID=”position2”/>
  </viewProperty>
</entity>
<entity id=”e3” name=”Protein” type=”Protein”>
<entity id=”e4” name=”mRNA” type=”mRNA”>
</model>
<globalViewProperty>
  <globalShape shapeID=”g:shape1”>
    <filter type=”csml-filter:elementType”>
      <parameter key=”elementType” value=”entity”/>
      <parameter key=”type” value=”protein”/>
      <parameter key=”selectionType” value=”match”/>
    </filter>
    <svg:svg>
     <ellipse rx=”40” ry=”50”/>
```

```
    </svg:svg>
  </globalShape>
  <globalShape shapeID=”g:shape2”>
    <filter type=”csml-filter:elementType”>
      <parameter key=”elementType” value=”entity”/>
      <parameter key=”type” value=”mRNA”>
      <parameter key=”selectionType” value=”match”/>
    </filter>
    <svg:svg>
       <rect width=”50” height=”50”/>
    </svg:svg>
  </globalShape>
</globalviewProperty>
</project>
```

2.9 View Model Property

2.9.1 ViewSet: <viewSet>

```
<viewSet>

  <view/>+

</viewSet>
```

In CSML3.0, user can specify more than one view for each subModel in <subModel> and <model>. The <viewSet> tag contains more than one <view> as its child.

2.9.2 View: <view>

```
<view viewID="" name="" refModelID="" refShapeID="" refPositionID=""
refAnimationID="" refChartID="">

  <layout/>{0,1}

  <refElement/>*

  <refChart/>*

</view>
```

One <view> corresponds to one special view, e.g. automatic layout, defined positioned layout, Petri net view, and gene network view. The tag has seven attributes. All <view> should have a unique value each other as its "viewID". The "name" attribute should be human understandable value. The "refModelID" attribute refers to the value of "modelID" attribute in one of subModels and specifies which model should be visualized. If the "refModelID" attribute is not defined, the main model is visualized. The "refShapeID" attribute refers to the value of "shapeID" attribute in <shape> tag and specifies which shape should be used for each element in the model. The value can specify more than one shapeIDs. If more than one shapeID is specified, the former shape is overrided with later one. For example, <view refShapeID="shape1;shape2"> means if both shapes are defined in an element, then "shape2" is used, if just "shape2" is defined in an element, then "shape2" is used.

Similarly, the "refPositionID" attribute refers to the value of "positionID" attribute in <position> tag and specifies which position should be used for each element in the model. The value can specify more than one positionIDs, e.g. <view refPositionID="pos1:pos2">.

The "refAnimationID" attribute refers to the value of "animationID" attribute in <animation> tag and specifies which animation should be used for each element that is described in Section 2.11. Similar to other values of attributes "refShapeID" and "refPositionID", the value of "refAnimationID" can reference more than one "animationID" with ;, e.g. <animation animationID="anim1;anim3;anim5"/>.

The <view> contains <layout>, <refElement>, and <refChart> as its child. The <layout> is used for applying automatic layout to the model. The <refElement> is used for overriding the shape, position, and animation of the referenced element (see Section 2.9.2.2). The <refChart> is used for overriding the shape, position, and animation of the referenced chart (see Section 2.10.3).

2.9.2.1 Layout: <layout>

```
<project … "xmlns:http://www.csml.org/layout/2_0">

…

<layout>

  <csml-layout:layout/>

</layout>
```

The <layout> tag takes <csml-layout> as its child. In CSML3.0, the layout settings can be defined in the <csml-layout> tag. The format is separately defined in the "CSML Layout XML" and shortly mentioned in Section 4.2.

2.9.2.2 Reference Element: <refElement>

```
<refElement refID="">

   <position x="" y="" position="">{0,1}

   <shape/>{0,1}

   <animation/>
```

```
</refElement>
```

The <refElement> tag is used for overriding the shape and position of the element that is referred with "refID" attribute. The value of "refID" attribute should be the value of "id" attribute in the referenced element.

For some cases, subset of model might not to be automatically moved with the layout option. To deal with this, the <position> tag has "position" attribute as in Section 2.8.2, the value can be "static" or "auto". If the value is "static", the referenced element does not automatically move. The default value of "position" attribute is "auto".

The <refElement> tag has three child tags <position>, <shape>, and <animation>. The <animation> tag is described in Section 2.8. Different from the <position> and <shape> tags in <viewProperty>, the tag just occurs zero or once. This is because one shape and position should be decided in one view. With the same reason, "positionID" and "shapeID" attribute is not defined as in the <position> (see Section 2.8.2) and <shape> (see Section 2.8.3) tag in the <view> tag, respectively. Similarly, different from the <animation> tag in <animationProperty> (see Section 2.11.3), the tag just occurs zero or once. With the same reason, "animationID" attribute is not defined as in the <animation> tag in the <view> tag.

Example 20

```
<subModelSet>

  <subModel modelID="model1">

    <element refID="e1"/>

    <element refID="e2"/>

  </subModel>

</subModelSet>



<viewSet>

  <view viewID="View1" name="sample view1" refModelID="model1"
refShapeID="shape1" refPositionID="position1"/>

  <view viewID="View2" name="sample view2" refModelID="model1"
refShapeID="shape1" refPositionID="position2"/>
```

```
  <view viewID=”View3” name=”sample view3” refModelID=”model1”
refShapeID=”shape2” refPositionID=”position1”/>

  <view viewID=”View4” name=”sample view4” refModelID=”model1”
refShapeID=”shape2” refPositionID=”position1”/>

    <refElement refID=”e1”>

      <position x=”50” y=”50”/>

      <shape/>

    </refElement>

  </view>

  <view viewID=”View5” name=”sample view5” refModelID=”model1”
refShapeID=”shape2” positionRef=”position2”>

     <layout>

       <csml-layout:csmlLayout/>

     </layout>

  </view>

  <view viewID=”View6” name=”sample view6” refModelID=”model1”
refShapeID=”shape2” positionRef=”position2”>

    <layout>

      <csml-layout:csmlLayout/>

    </layout>

    <refElement refID=”e1”>

       <position position=”static”/>

    </refElement>

  </view>

</viewSet>
```

1. `View1` and `View2`

`View1` and `View2` are the same model with the same shape for each element except for the position of all elements.

2. `View1` and `View3`

`View1` and `View3` are the same model with the sample position except for the shape of all elements.

3. `View4`

The shape and position of `e1` is defined in the `<refElement>` and overrided the view.

4. `View5`

The automatic layout is applied to the `model1`.

5. `View6`

The automatic layout is applied to the `model1` except for the `e1` element. The position of the `e1` element stays `x=20` and `y=35`.

2.10 Chart Property

2.10.1 Set of Chart: <chartSet>

```
<project>

  <chartSet>

    <chart>+

  </chartSet>*

</project>
```

The <chartSet> contains a list of all simulation charts for a model and submodels. A chart size and position can be modified in each of views while sharing other properties – e.g. which properties of elements are plotted - among them.

2.10.2 Chart: <chart>

```
<chart chartID="" name="">

  <property/>*

  <refChartElement/>*

</chart>
```

The <chart> tag consists of <property> (see Section 2.10.2.1) and <refChartElement> (see Section 2.10.2.2).

2.10.2.1 Property: <property>

```
<property key="csml-chartProperty:" value="" type="" />
```

The <property> tag is used to set properties of the chart, e.g. (i) x-axis minimum value , (ii) x-axis maximum value, and (iii) grid should be displayed or not. The predefined keys for chart in CSML3.0 always start from csml-chartProperty:. The keys are summarized in Section 3.12 “Predefined Chart Properties in CSML3.0”.

2.10.2.2 Reference Chart Element: <refChartElement>

```
<refChartElement refID="" refProperty="" name=""/>
```

The <refChartElement> tags specify which elements should be plotted on the chart. The "refID" attribute is used to specify the referenced element and "refProperty" is used to specify which property of the element should be plotted on the chart. The acceptable values are in Table 7.

2.10.3 Reference Chart: <refChart>

```
<view>

  <refChart chartIDRef="">

   <property/>*

  </refChart>*

</view>
```

The <refChart> tag in the <view> has "chartIDRef" attribute the value of which references to the "chartID" attribute in the <chart> tag. The contents of <property> are the same to <property> in <chart>. If the same key is defined in the <property> of <refChart> and that of <chart>, the <refChart> value is used.

Example 21

```
<project>

<model>

<entity id="e1"/>

<entity id="e2"/>

<process id="r1" type="biological">

  <connector refID="e1" type="input:process:biological"/>

  <connector refID="e2" type="output:process:biological"/>

</process>

</model>
```

```
<chartSet>
<chart chartID=”c1” name=”Chart1”>
  <property key=”csml-chartproperty:” value=””/>*
  <refChartElement refID=”e1”
refProperty=”csml-accessibleProperty:simulation.currentValue”
name=”current entity value”/>
  <refChartElement refID=”p1”
refProperty=”csml-accessibleProperty:simulation.flow” name=”current flow
of reaction”/>
  <viewProperty>
    <shape shapeID=”shape1”>
      <svg:svg><svg:rect width=”30” height=”80”
exttype=”chart”/></svg:svg>
    </shape>
    <shape shapeID=”shape2”>
      <svg:svg><svg:rect width=”30” height=”50”
exttype=”chart”/></svg:svg>
    </shape>
    <position x=”50” y=”50” positionID=”position1”/>
    <position x=”70” y=”200” positionID=”position2”/>
  </viewProperty>
</chart>
</chartSet>

<viewSet>
```

```
 <view viewID=”v1” refModelID=”main” refShapeID=”shape1”
refPositionID=”position1”>
  <refElement />
  <chart chartIDRef=”c1”/>
 </view>
 <view viewID=”v2” refModelID=”main” refShapeID=”shape1”
refPositionID=”position2”>
  <refElement />
  <refChart chartIDRef=”c1”/>
 </view>
<view viewID=”v2” refModelID=”main” refShapeID=”shape1”
refPositionID=”position2”>
  <refElement />
  <refChart chartIDRef=”c1”>
    <shape>
      <svg:svg><svg:rect width=”30” height=”50”
exttype=”chart”/></svg:svg>
    </shape>
    <position x=”70” y=”200” positionID=”position2”/>
  </refChart>
 </view>
</viewSet>
```

2.11 Animation Property

2.11.1 Global Animation Property: <globalAnimationProperty>

For a view, CSML3.0 can define animations during dynamic simulation and can modify properties of shape and position of each element depending on other properties of the element.

```
<project>

 <globalAnimationProperty>

   <animation/>+

 </globalAnimationProperty>*

</project>
```

The <globalAnimationProperty> section defines global animation settings of model and submodels in the child tag <globalAnimation>.

2.11.2 Global Animation: <globalAnimation>

```
<globalAnimation animationID="" name="">

  <select/>* | <filter/>*

  <svg:svg>

  <svg:animate>* | <svg:set>* |<svg:animateMotion>* | <svg:animateColor>*

  </svg:svg>

</globalAnimation>
```

In one <globalAnimation> section, elements are selected and for these elements some animations are assigned with <svg:set>, <svg:animate>, <svg:animateMotion>, and <svg:animateColor> tags. For the selection of elements, the same tags <select> and <filter> in the <subModel> are used with the same manner (see Sections 2.7.3 and 2.7.4).

The "animationID" attribute should set a unique value among <globalAnimation> tags and <animation> tags in Section 2.11.4. The "name" should be human understandable words.

```
<svg:animate>

<svg:set>

<svg:animateMotion>

<svg:animateColor>
```

The <svg:animate>, <svg:set>, <svg:animateMotion>, <svg:animateColor> tags, and their child tags use the same tag in SMIL2.0 (Synchronized Multimedia Integration Language) format part of which is adopted in SVG1.1 format. The SMIL is developed to be the standard format for writing interactive multimedia presentation. The complete document will be obtained from http://www.w3.org/TR/SMIL2/.

In CSML3.0, all attributes in <svg:set>, <svg:animate>, <svg:animateColor>, and <svg:animateMotion> are the same exept for one attribute "refProperty". The details are described in "Special SMIL Animation Tags and Attributes in CSML3.0".

```
<svg:set refProperty="csml-accessibleProperty:xx" svg:targetElement=""
svg:attributeName="" svg:from="" svg:to="" svg:min="" svg:max=""/>
```

The <svg:set> section updates the shape and position attributes of selected elements depending on the referenced property in the "refProperty" attribute, the available values of which are described in "Accessible Element Properties in CSML3.0".

```
<svg:animate refProperty="csml-accessibleProperty:xx"
svg:targetElement="" attributeName="" from="" to=""
begin="csml-accessibleProperty:xx" dur=""/>
```

The <svg:animate> section specifies the animation of target shape in "svg:targetElement" with the referenced property in the "refProperty".

```
<svg:animateColor refProperty=”csml-accessibleProperty:xx”
svg:targetElement=”” svg:attributeName=”” svg:from=”” svg:to=””/>
```

The <svg:animateColor> section specifies the animation color of target shape in “svg:targetElement” with the referenced property in the “refProperty”.

```
<svg:animateMotion refProperty=”csml-accessibleProperty:xx”
svg:targetElement=”” svg:from=”” svg:to=””
svg:begin=”csml-accessibleProperty:xx” svg:dur=”” />
```

The <svg:animateMotion> section specifies the motion of target shape in “svg:targetElement” with the referenced property in the “refProperty”

Some special values of “begin” and “end” attributes in <animate>, <animateColor> and <animateMotion> tags are predefined in CSML3.0. The acceptable values are summarized in Table 6 of Section 3.7 “Accessible Element Properties in CSML3.0”. For example, begin=”csml-accessibleProperty:simulation:activityEventOn” means that the animation starts a time when the selected process is active.

Example 22

The “animation1” means that the background colors of all entities are changed from red to yellow, depending on each of their initial values. In addition, the background colors of them should be changed from red to yellow, depending on each of their current values during simulation.

The “animation2” means that the background colors of all activated processs should be changed from red to yellow with 5s.

The “animation3” means that the widths of all processs should be changed from default width to 10px depending on each of the flowing speed during simulation.

```
<globalAnimationProperty>

  <globalAnimation animationID=”animation1”>
```

```
    <filter type=”csml-filter:elementType”>
     <parameter key=”elementType” value=”entity”/>
     <parameter key=”selectionType” value=”match”/>
    </filter>
    <svg:svg>
    <svg:set attributeName=”backgroundColor” from=”red” to=”yellow”
refProperty=”csml-accessibleProperty:simulation:initialValue” min=”0”
max=”5”/>
    <svg:set attributeName=”backgroundColor” from=”red” to=”yellow”
refProperty=”csml-accessibleProperty:simulation:currentValue” min=”0”
max=”5”/>
   </svg:svg>
   </globalAnimation>
   <globalAnimation animationID=”animation2”>
    <svg:animate attributeName=”backgroundColor” from=”#99cc99”
to=”#993300” begin=”csml-accessibleProperty:simulation:firingEvent”
dur=”5s”/>
   </animation>
   <animation animationID=”animation3”>
    <filter type=”csml-filter:elementType”>
     <parameter key=”elementType” value=”entity”/>
     <parameter key=”selectionType” value=”match”/>
    </filter>
    <svg:svg>
     <set refProperty=”csml-accessibleProperty:simulation:flow”
attributeName=”width” from=”default” to=”10” min=”0” max=”5”/>
```

```
    </svg:svg>

   </globalAnimation>

</globalAnimationProperty>
```

2.11.3 Animation Property: <animationProperty>

```
<entity|process|connector|fact|importModel>

  <animationProperty>

    <animation/>+

  </animationProperty>{0,1}

</entity|process|connector|fact|importModel>
```

CSML3.0 supports to define each animation in <animationProperty> as child of element tags <entity|process|connector|fact|importModel>. If the same value of animationID in <animation> of <globalAnimationProperty> and in <animation> of <animationProperty>, the animation is replaced with the local definition with <animationProperty>.

2.11.4 Animation: <animation>

```
<animation animationID="" name="">

    <svg:svg>

     <svg:animate>*| <svg:set>* |<svg:animateMotion>* |
<svg:animateColor>*

    </svg:svg>

</animation>
```

The syntax rule in the <animation> is similar to global animation settings <globalAnimation>. The difference is <filter> and <select> tags are removed because of the target element of local animation setting is always its parent element.

```
<view refAnimationID="" …>
```

```
  <refElement refID=””>

    …

    <animation/>*

  </refElement>+

</view>
```

Each <refElement> of <view> section can also take <animation> tag as its child. By using the <animation> tag, referenced animation setting with “refAnimationID” attribute in <view> is replaced, i.e. CSML3.0 can specifies advanced local animation of the referenced element in its view.

2.12 Biological Property

2.12.1 Biological Property: <biologicalProperty>

For knowledge representation of biopathway, the information of biological property is important. In CSML3.0, these information can be described in the <biologicalProperty> tag, which is the child tag of elements in <model>, i.e. <entity>, <process>, <fact>, <connector>, and <importModel>. From CSML3.0 Ontology point of view, they are child classes of the ElementBase class.

```
 <entity|process|connector|fact|importModel>

  <biologicalProperty refID="" refOrganismID="" chemicalFormula=""
molecularWeight="" refMoleculeRoleID="" refBiologicalEventID=""
refCellComponentID="" shortName="" sequence="" deltaH="" ECNumber=""
deltaS="" synonym="" availability="">

     <property key="" value="" type="" />*

     <chemicalStructure structureFormat="CML|SMILES|InChl">

         <structureData></structureData>

         <comments>{0,1}

     </chemicalStructure>*

     <sequenceFeature name="" shortName="" refUnificationXrefID=""
refPublicationXrefID="" refRelationXrefID="" erfFeatureTypeID="">

         <synonym>xxx</synonym>*

         <sequenceInterval>

              <sequenceIntervalBegin sequencePosition=""
positionStatus="EQUAL|LESS-THAN|GREATER-THAN"/>

              <sequenceIntervalEnd sequencePosition=""
positionStatus="EQUAL|LESS-THAN|GREATER-THAN"/>

         </sequenceInterval>*
```

```
          <sequenceSite sequencePosition=”” positionStatus=”EQUAL|LESS-THAN|GREATER-THAN”/>

          </sequenceSite>*

          <comments/>{0,1}

     </sequnceFeature>

     <evidence evidenceID=”” refUnificationXrefID=”” refPublicationXrefID=”” refRelationshipXrefID=”” refEvidenceCodeID=””>

          <confidence confidenceValue=”” refPublicationXrefID=””/>*

          <experimentalForm refID=”” refExperimentalFormType=””/>*

          <comments/>{0,1}

     </evideince>*

     <dataSource name=”” refPublicationXref=”” refUnificationXref=””>

           <comments/>{0,1}

     </dataSource>*

     <deltaGPrimeO deltaGPrimeO=”” ionicStrength=”” kPrime=”” ph=”” pmg=”” temperature=””>

           <comments/>{0,1}

     </deltaGPrimeO>*

     <kPrime kPrime=”” ionicStrength=”” ph=”” pmg=”” temperature=””>

           <comments/>{0,1}

     </kPrime>*

     <comments/>{0,1}

  </biologicalProperty>

 </entity|process|connector|fact|importModel>
```

The contents of <biologicalProperty> depends on the "type" of element. With the "type", the source class in CSML3.0 Ontology can be specified. The source class takes some SLOTs, which are mapped to some tags or attributes in CSML XML Format. Thus, depending on the "type" of element, the acceptable child tags of <biologicalProperty> are desided.

For example, in CSML3.0, all classes that corresponds to elements takes a PROPERTY slot. With the mapped tag of PROPERTY slot to CSML XML format is <property> tag, which holds "key", "value", and "type" attributes. The "key" is the property name and "value" is the value of the property with type "type"..

As another example, if the type is "mRNA", the original class mRNA in Ontology can take PROPERTY, SYNONYM, COMMENT, DATASOURCE, SHORTNAME, AVAILABILITY, UNIFICATIONXREF, PUBLICATIONXREF, RELATIONSHIPXREF, SEQUENCEFEATURELIST and ORGANISM as its slots. Thus, corresponding tags and attributes, <property>, <synonym>, <comment>, <dataSource>, "shortName", "availability", "refUnificationXrefID", "refPublicationXrefID", "refRelationshipXref", "sequenceFeatureList", and "refOrganismID" are acceptable in the <biologicalProperty> tag. The mapping details are described in the separate document **Mapping Rule between CSML 3.0 and CSML 3.0 Ontology**. The slots for each class, biological meaning of each class and slot are described in the separate document **CSML3.0 Ontology**.

2.12.2 GlobalBiological Property: <globalBiologicalProperty>

```
<project>

 <globalBiologicalProperty>

    <cellType cellTypeID="" refUnificationXrefID="" term="">

         <comments/>{0,1}

    </cellType>*

    <tissue refUnificationXref="" term="">

         <comments/>{0,1}

    </tissue>*

    <organism organismID="" refCellTypeID="" refTissueID=""
refUnificationXrefID="">

        <comments/>{0,1}

    </organism>*
```

```
     <biologicalRole biologicalRoleID="" refUnificationXrefID="">
        <term>xxx</term>*
        <comments/>{0,1}
     </biologicalRole>*
     <biologicalEvent biologicalEventID="" refUnificationXref=""
        <term>xxx</term>*
        <comments/>{0,1}
     </biologicalEvent>*
     <cellComponent cellComponentID="" refUnificationXrefID="">
        <term>xxx</term>*
        <comments/>{0,1}
     </cellComponent>
     <evidenceCode evidenceCodeID="" refUnificationXrefID="">
        <term>xxx</term>*
        <comments/>{0,1}
     </evidenceCode>*
     <experimentalFormType experimentalFormTypeID=""
refUnificationXrefID="">
        <term>xxx</term>*
        <comments/>{0,1}
     </experimentalFormType>*
     <featureType refUnificationXrefID="">
        <term>xxx</term>*
        <comments/>{0,1}
```

```
    </featureType>*

  </globalBiologicalProperty>

</project>
```

Biological properties that will be shared with many elements, i.e. <entity>, <process>, <connector>, <fact>, and <importedModel>, are defined in the <globalBiologicalProperty> tag. The defined global properties are referenced in <biologicalProperty> tag in these elements with refXXXID attributes, e.g. refCellComponentID, refBiologicalEventID, refBiologicalRoleID, and refOrganismID. The mapping details are described in the separate document **Mapping Rule between CSML 3.0 and CSML 3.0 Ontology**. The slots for each class, biological meaning of each class and slot are described in the separate document **CSML3.0 Ontology**.

2.13 Comment Property

Many tags of CSML3.0 can take <comments> tag. Especially, all tags that derives from the subclasses of elementBase in CSML3.0 Ontology can take <comments>.

2.13.1 Comments: <comments>

```
<comments>

  <comment/>+

</comments>
```

The <comments> tag can contain more than one <comment> tag.

2.13.2 Comment: <comment>

```
<comment name="" type="RDF">

   RDF related tags.

</comment>

<comment name="" type="annotation">

   Annodation related tags.

</comment>

<comment name="" type="xhtml">

   XHTML related tags.

</comment>

<comment name="" type="text">XXX</comment>
```

The <comment> tag has two attributes "name", and "type". The "name" should assign human understandable words. The value of "type" can be selected from RDF, xhtml, annotation, and text. If the value is xhtml, xhtml tags can use in the contents. If the value is RDF, RDF tags

can use in the contents. If the value is annotation, any tags can use in the contents. If the value is text, just text can take in the tag. The <comment type="RDF"> is the counterpart for <RDF:annotations> in SBML and CellML. The <comment type="annotation"> and <comment type="xhtml"> is the counterpart of <sbml:annotation> and <sbml:notes> in SBML, respectively.

Usually, <comment type="RDF"> and <comment type="annotation"> is used for machine readability and <comment type="xhtml"> and <comment type="text"> is used for human readability.

CSML Ontology: Base::Comment

2.14 Reference Property

2.14.1 References: <references>

```
<references>

  <publicationXref/>*

  <relationshipXref/>*

  <unificationXref/>*

</references>
```

The <references> tag can contain three tags, <publicationXref>, <relationshipXref>, and <unificationXref>. These terms are the same to the PublicationXref, RelationshipXref, and unificationXref classes in BioPAX Ontology, respectively. It must note that their meaning are slightly different in BioPAX Ontology and CSML3.0 Ontology. In additionl, some extra slots are introduced in CSML3.0 Ontology.

2.14.2 External Reference to Publication: <publicationXref>

```
<publicationXref publicationXrefID="" name="" xlink:href="" db=""
dbVersion="" xrefID="" xrefIDVersion="" title="" year="">

   <author/>*

   <source/>*

   <comments/>{0,1}

</publicationXref>
```

The <publicationXref> tag is used for the external link, e.g. external database access. The <publicationXref> tag has a normal attributes "name", and XLink related attribute, "xlink:href.

The "name" should assign human understandable words. XLink is an XML used for creating hyperlinks within XML documents and standardized in W3C. The specification is in http://www.w3.org/1999/xlink/.

Other attributes "publicationXrefID", "db", "dbVersion", "xrefID", "xrefIDVersion", "title", and "year" are special attributes. The "publicationXrefID" should assign a unique value among all <publicationXref> tags. The meanings of "db", "dbVersion", "xrefID", and "xrefIDVersion" are the same to DB, DB-VERSION, ID, ID-VERSION slots of Xref class in BioPAX. The "db" is the name of the external database to which this <publicationXref> refers. The "dbVersion" is the version of the external database in which this <publicationXref> was last known to be valid. Resources may have recommendations for referencing dataset versions. For instance, the Gene Ontology recommends listing the data the GO terms were downloaded. The "xrefID" is the primary identifier in the external database of the object to which this xref refers. The "xrefIDVersion" is the version number of the "xrefID", e.g. The RefSeq accession number NM_005228.3 should be split into NM_05228 at the "xrefID" and 3 as the "xrefIDVersion."

The <publicationXref> has three child tags, <author>, <source>, and <comments> (see Section 2.13.1). The <author> is the author of this publication. The <source> is used for the source in which the reference was published, e.g. a book title, a journal title, volume, and pages.

CSML Ontology: Base::PublicationXref

2.14.3 External Reference to Related Information: <relationshipXref>

```
<relationshipXref relationshipXrefID="" relationshipType="" name=""
xlink:href="" db="" dbVersion="" xrefID="" xrefIDVersion="">

  <comments/>{0,1}

</relationshipXref>
```

The <relationshipXref> tag is the same to RefaltionShipXref class in BioPAX Ontology. That defines a reference to an entity in an external resource that does not have the same biological identity as the refferring entity. Attributes "name", "xlink:href", "db", "dbVersion", "xrefID", and "xerfIDVersin" in this tag is the same to <publicationXref> tag.

Other attributes "relationshipXrefID" and "relationshipType" are special attributes. The "relationshipXrefID" should assign a unique value among all <relationshipXref> tags.

The meaning of "relationshipType" is the same to the RELATIONSHIP-TYPE slot of RelationshipXref class in BioPAX. The "relationshipType" specifies the type of relationship between the CSML Ontology object linked from, and the external object linked to, e.g. "gene of this protein", or "protein with similar sequence".

The <relationshipXref> has a general child tag <comments> (see Section 2.13.1).

CSML Ontology: Base::RelationshipXref

2.14.4 External Reference for Unification: <unificationXref>

```
<unificationXref unificationXrefID="" name="" xlink:href="" db=""
dbVersion="" xrefID="" xrefIDVersion="">

  <comments/>{0,1}

</unificationXref>
```

The <unificationXref> tag is the same to UnificationXref class in BioPAX Ontology. That defines a reference to an entity in an external resource that has the same biological identity as the referring entity. All Attributes "name", "xlink:href", "db", "dbVersion", "xrefID", and "xerfIDVersin" in this tag is the same to <publicationXref> and <relationshipXref> tag.

The other attribute "unificationXrefID" is special. The "unificationXrefID" should assign a unique value among all <unificationXref> tags.

The <unificationXref> has a general child tag <comments> (see Section 2.13.1).

CSML Ontology: Base::UnificationXref

3 Detailed Specs of CSML3.0

3.1 Special SVG Shape Tags and Attributes in CSML3.0

The shape of CSML3.0 reuses SVG1.1 format except for the following three:

(i) The special attribute "exttype" is introduced in the <eclipse> and <rect> tags. The value of "exttype" is used for supporting primitive shapes of HFPNe. For example, <rect exttype="discrete"> is used to generate original shape of "discrete process" in HFPNe.

(ii) The special attribute "exttype" in the <line> and <polygon> tags are introduced for supporting inhibitory, association and process connectors in the HFPNe. For example, <line exttype="association" marker-end="url(#association)"> is used to generate original shape of "association connector" in HFPNe.

(iii) The special attribute "exttype" is introduced to <text> for supporting textarea that is usually used in graphical component libraries, e.g. Swing and SWT in Java. For example, <text exttype="textarea">test</text> is used to generate textarea with the "test" string.

Example 23

The attribute "exttype" with underline is the special attribute in CSML3.0.

```
<shape shapeID="shape1">

  <svg>

    <eclipse rx="10" ry="20"
exttype="discrete|continuous|generic|object"/>

    <rect width="30" height="30" exttype="discrete|continuous|generic"
visibility="hidden"/>

    <image width="30" height="30" xlink:href=""/>

    <line from="30" to="30" exttype="association|inhibitory|process"
marker-end="url(#association|bioInhibitory|process)"
marker-start="url(#ellipse|inhibitory|rectangle|triangle)"/>
```

```
   <rect width="30" height="30" visibility="hidden" />

   <text width="30" height="30" exttype="textarea"/>XXXX</text>

   <polyline points="850,75 958,173.5 742,173.5"
exttype="association|inhibitory|process"
marker-end="url(#association|bioInhibitory|process)"
marker-start="url(#ellipse|inhibitory|rectangle|triangle)"></polygon>

  </svg>

</shape>
```

```
<svg:set refProperty="csml-accessibleProperty:xx"  refSubShapeID=""
svg:attributeName="" svg:from="" svg:to="" svg:min="" svg:max=""/>

<svg:animate refProperty="csml-accessibleProperty:xx"
svg:targetElement="" attributeName="" from="" to=""
begin="csml-accessibleProperty:xx" dur=""/>

<svg:animateColor refProperty="csml-accessibleProperty:xx"
svg:targetElement="" svg:attributeName="" svg:from="" svg:to=""/>

<svg:animateMotion refProperty="csml-accessibleProperty:xx"
svg:targetElement="" svg:from="" svg:to=""
svg:begin="csml-accessibleProperty:xx" svg:dur="" />
```

The `<svg:set>` section updates the shape and position attributes of selected elements depending on the referenced property in the `"refProperty"` attribute. The value of the attribute always starts from csml-accessibleProperty: the detail of which is documented in Section 98"Accessible Element Properties in CSML3.0".

The `"refSubShapeID"` attribute is used to specify the target shape and position in the model. In CSML3.0, four patterns are assumed: (i) specify one shape or position in one element (ii) specify part of one shape in one element (iii) specify one shape or position in selected elements, and (iv) specify part of one shape in selected elements. Especially, the (iii)(iv) are used to apply the same animation setting to selected elements with `<filter>` or `<refElement>` tag.

For (i) and (iii) cases, the target is apparent from the structure of CSML3.0, e.g. if the `<animation>` is in an element tag, the target is the element, and if the `<animation>` is in an `<globalAnimationProperty>` tag, the target is filtered or selected elements. For (ii) and (iv)

cases, the target needs to be explicitly assigned with the rule in Section 3.7 “Accessible Element Properties in CSML3.0”.

3.2 Element Type in CSML3.0

```
<entity type=”xxx”>

<process type=”yyy”>
```

Table 1 Entity Subclasses and its full value and abbreviated value of “type” in <entity>.

Class Name in CSML3.0 Ontology (Full Name)	Abbriviated Name
Biological:Object:Complex	Complex
Biological:Object:Dna	Dna
Biological:Object:Rna:mRNA	mRNA
Biological:Object:Rna:miRNA	miRNA
Biological:Object:Rna:rRNA	rRNA
Biological:Object:Rna:tRNA	tRNA
Biological:Object:Rna:xxx	
Biological:Object:Rna:Unknown	-
Biological:Object:Rna:Other	-
Biological:Object:Protein	Protein
Biological:Object:SmallMolecule	SmallMolecule
Biological:Object:Unknown	-
Biological:Object:Undef	-
Biological:Object:Other	-
Biological:Compartment:ExtraCellular	ExtraCellular
Biological:Compartment:Cell	Cell
Biological:Compartment:	-
Biological:Environment	Environment
Biological:Unknown	-
Biological:Undef	-
Biological:Other	-
NonBiological:Primitive	Primitive
NonBiological:Unknown	-
NonBiological:Undef	-
NonBiological:Other	-

Table 2 Process Subclasses and its full value and abbreviated value of “type” in <process>.

Class Name in CSML3.0 Ontology (Full Name)	Abbriviated Name
Biological	Biological
NonBiological:Primitive	Primitive
NonBiological:Unknown	Unknown
NonBiological:Undef	Undef
NonBiological:Other	Other

Table 3 Connector Subclasses and its full value and abbreviated value of “type” in <connector>.

Class Name in CSML3.0 Ontology (Full Name)	Abbriviated Name
Input:Process:Biological	Process
Input:Association:Biological	Association
Input:Inhibition:Biological	Inhibition
Output:Process:Biological	Output
Input:Process:NonBiological:Primitive	-
Input:Process:NonBiological:Unknown	-
Input:Process:NonBiological:Undef	-
Input:Process:NonBiological:Other	-
Input:Association:NonBiological:Primitive	-
Input:Association:NonBiological:Unknown	-
Input:Association:NonBiological:Undef	-
Input:Association:NonBiological:Other	-
Input:Inhibition:NonBiological:Primitive	-
Input:Inhibition:NonBiological:Unknown	-
Input:Inhibition:NonBiological:Undef	-
Input:Inhibition:NonBiological:Other	-
Output:Process:NonBiological:Primitive	-
Ouput:Process:NonBiological:Unknown	-
Output:Process:NonBiological:Undef	-
Output:Process:NonBiological:Other	-

Table 4 Connector Subclasses and its full value and abbreviated value of “type” in <fact>.

Class Name in CSML3.0 Ontology (Full Name)	Abbriviated Name
Input:Process:Biological	Process
Input:Association:Biological	Association

The subclasses of <fact> is not completed yet.

	Suffix
relation:view:	group,relation
relation:simulation	same-firing-behaviour, relation
relation:other	child, parent, cluster
relation :estimate :geneNetwork	regulation

3.3 Primitive Variable Type in CSML3.0

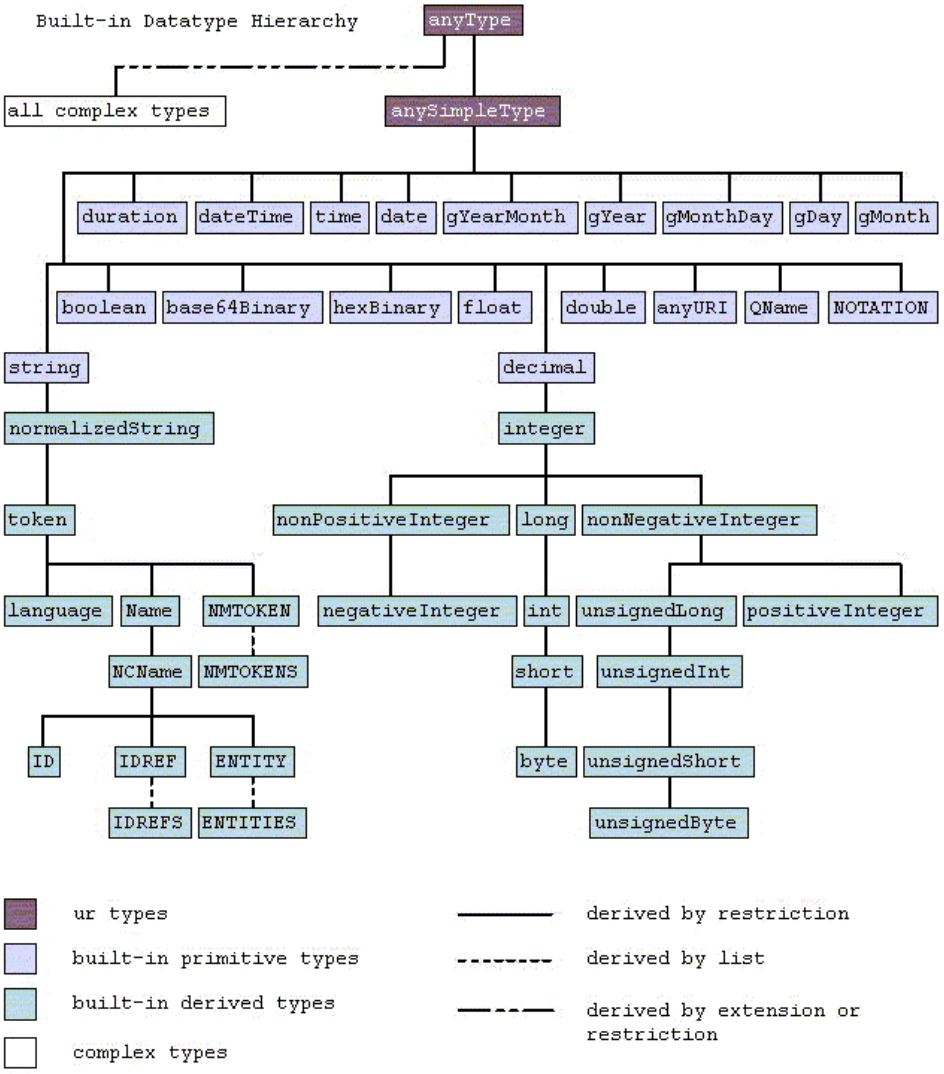
```
<variable type="double"/>

<parameter type="string"/>
```

The acceptable types will be specified and maintained in other document Ver.1.0 and also http://www.csml.org/csml-variable1_0.html. At the release time of CSML3.0, the following variable types are reserved

(Reserved terms of variable types)

`boolean, float, double, string, number, integer, long, int, short, byte.`



Sited from XML Schema Part 2: Datatypes

The meaning of above reserved datatypes are the exactly same to those of XML Schema Part2: Datatypes in http://www.w3.org/TR/xmlschema-2/. Other datatypes defined in XML Schema Part2 are removed in the　Ver.1.0.

3.4 Entity Object Type in CSML3.0

In CSML3.0, a `<variable>` in the `<entity><entitySimulationProperty><variable type="">` has a `"type"` attribute. The value of the `"type"` can be primitive type in Section 3.3 "Primitive Variable Type in CSML3.0" as the `"type"` attribute of `<parameter>` tag.

Moreover, the entity can take a predefined object type that is customized for biopathway simulation. In CSML3.0, following values for `"type"` are reserved. All values start from `"csml-object:"`. The complete predefined types and definitions are separately maintained at the document "CSML Entity Object Type Ver1.0" and "http://www.csml.org/objectType/1_0/".

Table 5

mRNA<T>	degradation	T can be numeric primitive type in Primitive Variable Type in CSML3.0.
protein<T>	degradation	T can be numeric primitive type in Primitive Variable Type in CSML3.0.

3.5 Variable Parameter in CSML3.0

```
<variable type="">

<parameter key="csml-variable:parameter:" value="" type""/>

</variable>
```

A variable can take some parameters with `<parameter>` tag. The `<parameter>` tag has three attributes `"key"`, `"value"`, and `"type"`. The acceptable values of `"key"` will be changed with the `"type"` of the variable.

In CSML3.0, following values for `"key"` are predefined. All values start from `"csml-variable:parameter"`. The complete predefined keys will be separately maintained at the document "CSML Variable Parameter Ver1.0" and "http://www.csml.org/varaibleParameter1_0.html."

1. initialValue

   The key is used to specify the initial value of the variable.

2. minimumValue

   The key is used to specify the minimum value of the variable.

3. maximumValue

   The key is used to specify the maximum value of the variable.

4. evaluateScriptOnce

   The key is used to specify the current value of variable should be replaced with the evaluated result of value of `"value"` attribute. If `evaluateScriptOnce="true"` then the value of "`value`" is evaluated just at the initial step. If `evaluateScriptOnce="false"` then the value of "`value`" is evaluated at every simulation steps.

5. unit

   The key is used to specify the physical unit of the variable.

6. constant

   The key is used to specify the variable is static value or not.

7. refCompartmentID

   The key is used to specify the compartment position of the variable.

3.6 Global Simulation Options in CSML3.0

```
<optionSet>

   <parameter key=”csml-simParameter:enhancedFiring”
value=”(true|false)” type=”:boolean”/>

   <parameter key=”csml-simParameter:samplingInterval” value=”(double
number)”type=”double”/>

   <parameter key=”csml-simParamter:logUpdateInterval” value=”(double
number)”type=”double”/>

   <parameter key=”csml-simParamter:plotUpdateInterval” value=”(double
number)”type=”double”/>

   <parameter key=”csml-simParamter:simulationTime” value=”(double
number)”type=”double”/>

</optionSet>
```

The `<optionSet>` specifies global options that are related with dynamic simulation. The tag contains `<parameter>` tag as its child. The “key” is the key of the parameter and `“value”` is the value of the key. The type of the key can be defined with `“type”`.

The key possible values are: enhancedFiring (backwards compatibility; always true), samplingInterval (the interval time of the simulation), plotUpdateInterval (defines how often the chart windows should be updated; it has no impact on simulation results), logUpdateInterval (defines how often the internal simulation log should be updated; it has no impact on simulation results; it is ignored if the simulation is played in step mode), simulationTime (defines how long the simulation should last; the ratio `simulationTime/samplingInterval` defines the number of steps to be executed in the simulation).

Example 24

```
<optionSet>
```

```
  <parameter key=”csml-simParameter:enhancedFiring” value=”true”
type=”:boolean”/>

  <parameter key=”csml-simParameter:samplingInterval” value=”0.1”
type=”double”/>

  <parameter key=”csml-simParamter:logUpdateInterval” value=”1.0”
type=”double”/>

  <parameter key=”csml-simParamter:plotUpdateInterval” value=”1.0”
type=”double”/>

   <parameter key=”csml-simParamter:simulationTime” value=”2000.0”
type=”double”/>

</optionSet>
```

3.7 Accessible Element Properties in CSML3.0

In CSML3.0, each element has some biological, simulation, view and animation properties. Biological and simulation properties are referenced from view and animation properties for updating the view and animation status in the model or submodel. View properties are referenced from animation properties for updating the animation status in the model or submodel.

Current predefined properties of elements are summarized in this section. The predefined properties of elements will be added after the release of CSML3.0. The complete predefined keys will be separately maintained at the document "Accessible Element Properties Ver1.0" and "http://www.csml.org/accessibleProperty/1_0/".

3.7.1 Basic Property

Elements in a model have some primitive properties, e.g. name. These properties always start from the prefix csml-accessibleProperty:basic: as in Table 6.

Table 6

Target element	Type	Value
entity		name id type
relation		name, id type
connector		name, id

		type
project		projectID projectVersionID
subModel		name modelID

3.7.2 Simulation Property

Simulation related properties always start from the prefix csml-accessibleProperty:simulation:. The simulation properties can be categorized into three: (i) static property that does not change the value among simulation steps, e.g. simulation parameter, (ii) dynamic property that changes the value among simulation steps, e.g. current value of entity, (iii) event property that will be triggered at a simulation step. Supported accessible properties are summarized in Table 7. For some case, the acceptable values depend on the targeting element and its `"type"`.

Table 7

Target element	Type	Value	Static Dynamic Event	Log is acceptable	Plottable in Chart
entity		name	static		
		unit	static		
		type	static		
		compartment	static		
		initialValue	static		yes
		maximumValue	static		yes
		minimumValue	static		yes
		currentValue	dynamic	yes	yes
relation		name	static		
		kineticStyle	static		
		calcStyle	static		
		delayStyle	static		
		firingStyle	static		
		currentActivity	dynamic	yes	yes
		currentFiringCount	dynamic	yes	yes
		currentDelayingStatus	dynamic	yes	yes

		currentDelayingTime eventActivityOn eventActivityOff	dynamic event event	yes	yes
relation	calcStyle=”speed” calcStyle=”update” calcStyle=”method” calcStyle=”check”	currentSpeed| currentUpdateResult| currentMethodResult| currentCheckResult	dynamic dynamic dynamic dynamic	yes yes yes yes	yes yes yes yes
connector		name currentFlow currentConnectorActivity eventConnectorActivityOn eventConnectorActivityOff	name dynamic dynamic event event	 yes yes	 yes yes

3.7.3 Biological Property

Biological related properties always start from the prefix "`csml-accessibleProperty:biological:`". The acceptable values depend on the targeting element and its "`type`" as in the Table 8.

Table 8

Target Element	Type	Value	Directly Defined Attribute Name
element tags		organismID moleculeRoleID biologicalEventID cellComponentID	

3.7.4 View Property

View related properties always start from the prefix csml-accessibleProperty:view:. The acceptable values depend on the target element and its "`type`" as in Table 9.

Table 9

Target Element	Type	Value	Directly Defined Attribute Name

Example 25

```
<entity id="e1">

<viewProperty>
```

```
  <position x=”20” y=”20” positionID=”position1”/>
  <shape shapeID=”shape1”>
   <svg:svg id=”r0”>
    <rect id=”r1” width=”20” height=”50”/>
    <rect id=”r2” width=”50” height=”70” />
   </svg:svg>
  </shape>
</viewProperty>
</entity>
<entity id=”e2”>
<viewProperty>
  <position x=”20” y=”50” positionID=”position1”/>
  <shape shapeID=”shape1”>
   <svg:svg id=”r0”>
    <rect id=”r1” width=”20” height=”50”/>
    <rect id=”r2” width=”50” height=”70” />
   </svg:svg>
  </shape>
</viewProperty>
</entity>
```

In SVG1.1, all SVG tags can take “id” (say svg “id”) for identification of the figure (see the above sample). In CSML3.0, the “id” is also used for specifying the shape as an animation target property. It must be noted that different <shape><svg> sections can take the same “id”. The feature is important to apply the same animation behavior to different elements with the same svg “id”. In order to specify the target property of animation, two attributes “svg:attributeName” and “csml:targetElement” are used. The first attribute is already

defined in SVG format and is used to set the target graphic property of animation. The second one is the special attribute in CSML3.0 and used to specify which figure element should be updated. The acceptable value of "csml:targetElement" is two types (i) concatenation of an element "id" and one of its svg "id"s with "." or (ii) just svg "id".

The case (i) is used to specify just one shape of an element. For the above example, <svg:set csml:targetElement="e1.r1" svg:attributeName="width"/> accesses the width property of <rect> with id="r1" of entity "e1".

The case (ii) is used to specify set of shape with the same svg "id". For the above example, <svg:set csml:targetElement="r1" svg:attributeName="width"/> accesses the width property of <rect> with id="r1" of entities "e1" and "e2".

3.8 Calc Style and Kinetic Style in CSML3.0

The kinetic style "kineticStyle" is used in the <kinetic> tag for specifying the detailed behavior of relation at simulation steps. The abstract behavior at simulation steps is defined with another attribute "calcStyle" in the same tag. Thus, the available value of "kineticStyle" depends on the value of "calcStyle". In CSML3.0, all predefined values of "calcStyle"/"kineticStyle" start from the csml-calcStyle:/csml-kineticStyle: prefix for making a sharp distinction between predefined values and user defined values, respectively. Depends on the value of csml-kineticStyle:, the number of child <parameter> tags in the <kinetic> tag and the "key" attribute value of each <parameter> tag are restricted.

All predefined values in CSML3.0 at the release of CSML3.0 are in Table 10.

Table 10

Value of KineticStyle	Simulation Algorithm	Value of CalcStyle	Parameters of Relation	Parameters of Connectors
custom	The reaction speed of connectors depends on the evaluated result of "custom" parameter.	speed	custom	
	The evaluated result of "custom" parameter is added to output connectors and subtracted from input connectors.	add		
mass	The reaction speed depends on the general mass action law.	speed	coefficient1 coefficient2	stoichiometry
stochasticMass	The reaction speed depends on the general mass action law with stochastic behavior.	speed	coefficient1 coefficient2 standard deviation	stoichiometry
logStochasticMass	The reaction speed depends on the general mass action law with log	speed	coefficient1 coefficient2	stoichiometry

	stochastic behavior.		standard deviation	
connectorCustom	The reaction speed of each connector depends on the evaluated result of "custom" parameter in each connector.	speed		custom
	The evaluated result of "custom" parameter is added to the entity of output connector and subtracted from the entity of input connectors..	add		
	The evaluated result of "update" parameter is used to update the entity of connectors.	update		
MM	The reaction speed depends on the Michaelis-Mentens kinetic.	speed	Km k2	
s-system	The reaction speed depends on the S-System kinetic.	speed	active_coefficient inhibit_coefficient	coefficient

The number of predefined values will be added after the release of CSML3.0. The complete predefined values will be separately maintained at the document "Kinetic Style Ver1.0" and "http://www.csml.org/kineticStyle/1.0".

3.9 Connector FiringStyle in CSML3.0

The connector firing style "connectorFiringStyle" attribute is used in the <firing> child tag of <connector> tags with the value of type contains input - calls *input connector tags* - for specifying the detailed behavior of activity for the connector at simulation steps.

In CSML3.0, all predefined values of "connectorFiringStyle" start from the csml-connectorFiringStyle: prefix for making a clear distinction between predefined values and user defined values.

All predefined values in CSML3.0 at the release of CSML3.0 are as follows;

1. threshold

If the value is set to the <firing> child tag of a <connector type="input:association:xxx"> tag, the activity of the connector is decided whether the current value of connected entity is greater than or equal to the value of "value" attribute in the child <parameter> tag with "value" as its "name" attribute. If the condition is filled the activity of the connector is true.

In contrast, if the value is set to the <firing> child tag of a <connector type="input:inhibition:xxx"> tag, the activity of the connector is decided with the almost inversive way. The activity of the connector is decided whether the current value of connected entity is greater than the value of "value" attribute in the child <parameter> tag with "value" as its "name" attribute. If the condition is filled the activity of the connector is false.

2. nocheck

If the value is set to a input connector, the activity of the connector is always true.

3. script

if the value is set to a input connector, the activity of the connector is the evaluated result of script in <script> child tag of the connector tag.

3.10 CSML Biological Properties with Element Type

The biological property `<property>` tag is used in the `<biologicalProperty>` tag for specifying the biological characteristic and associated information that cannot be described with other child tags of `<biologicalProperty>`. The tag has three attributes `"key"`, `"value"`, and `"type"`. All predefined keys start from `"csml-biologicalProperty:"` prefix is reserved for a sharp distinction between predefined keys and user defined keys. At the release time of CSML3.0, no special keys are used.

Table 11

element type	acceptable keys	

The number of predefined keys might be added after the release of CSML3.0. The complete predefined values will be separately maintained at the document "Biological Properties Ver1.0" and "http://www.csml.org/biologicalProperty/1_0/".

3.11 Predefined Log Properties in CSML3.0

```
<chart>

  <property key="csml-logProperty:" value="" type=""/>

</chart>
```

The reserved keys are as follows:

Table 12

Key	Description	Type
fileName	Name of the file to which the simulation log was written.	string
status	Current status of the simulation.	string, the following values are possible: running | finished | unknown | interrupted
timeStamp	Date and time of the simulation.	date
server	Reserved for future use.	string, the following values are possible: localhost
author	Name of the user that run the simulation.	string

3.12 Predefined Chart Properties in CSML3.0

```
<chart>

  <property key=″csml-chartProperty:″ value=″″ type=″″/>

</chart>
```

The reserved keys are as follows:

Table 13

Key	Description	Type
x	Left-top x position of the chart.	double
y	Left-top y position of the chart.	double
width	Width of the chart in canvas.	double
height	Height of the chart in canvas.	double
gridOn	Chart should show the grid or not.	boolean
toolTipOn	Chat should show the (x,y) value of cursor position.	boolean
autoScale	The x-axis range and y-axis range should be automatically set or not.	boolean
x-axisMin	The min value of x-axis. (It works when “autoScale” is ”true”.)	double
x-axisMax	The max value of x-axis. (It works when “autoScale” is ”true”.)	double
y-axisMin	The min value of y-axis. (It works when “autoScale” is ”true”.)	double
y-axisMax	The max value of y-axis. (It works when “autoScale”	double

	is ”true”.)	

3.13 Merging Specification in CSML3.0

```
<elementMerge type="csml-merge:">

  <parameter key="" value=""/>+

</elementMerge>
```

The <elementMerge> tag is used for merging the part of imported elements and set of elements in <entitySet> and <relationSet> tags. The merging method is set with "type" attribute and its parameters are configured with <parameter> tags in <elementMerge> tag. All predefined types starts from "csml-merge:" prefix for making a sharp distinction between predefined types and user defined types. The acceptable keys depend on the "type" of <filter> and all keys for predefined filters start from "csml-mergeParameter:".

Table 14 **Some predefined types and their keys of <elemnetMerge>**

Type of <elementMerge>	The key and value of <paramter>
csml-merge :biologicalProperty :geneID	mapAllMatchedRelation (true|false)
csml-merge :entity.name	mapAllMatchedRelation (true|false)
csml-merge :relation.name	mapAllMatchedEntity (true|false)

The number of predefined keys will be added after the release of CSML3.0. The complete predefined values will be separately maintained at the document "CSML Merge Ver1.0" and http://www.csml.org/merge/1_0/.

3.14 Filtering Specification in CSML3.0

```
<filter type=”csml-filter:”>

  <parameter key=”” value=””/>+

</filter>
```

The <filter> tag is used for selecting the set of elements that fills the condition that is specified in that tag. The filtering method is set with “type” attribute and its parameters are configured with <parameter> tags in <filter> tag. All predefined types are starts from csml-filter: prefix for making a sharp distinction between predefined types and user defined types. The acceptable keys depend on the “type” of <filter> and all keys for predefined filters start from csml-filterParameter:.

Table 15 **Some predefined types and their acceptable key and value of parameter.**

Type of Filter	Key and Value of Parameter
intersectionSet	fromModelList : modelID toModelList : modelID
unionSet	targetModelIDList : modelID
differenceSet	fromModelList : modelID toModelList : modelID
elementType	elementType : ”entity|relation” selectionType : ”match|unmatch”
relationType	relationType : ”PPI|phosphorylation|…”*, selectionType :”match|unmatch”
select-all	

The number of predefined keys will be added after the release of CSML3.0. The complete predefined values will be separately maintained at the document “CSML Filter Ver1.0” and http://www.csml.org/filter/1_0/.

3.15 Script Language in CSML3.0

The supported script languages are Pnuts (the standard language) and MathML. CSML facilitates the scripting various elements of the model. The meaning of "scripting" in CSML is two-fold:

- A numeric value in a model may be specified using an expression instead of a fixed value;
- Interaction of a model element with other elements may be specified using a set of expressions.

The following numeric values can be specified with expressions:

- Entity initial value,
- Relation firing, kinetic, and delay.
- Function definitions.

```
<script>
```

The `script` element contains the function definition for its parent element. For elements that may have a `value` attribute, such as `delay`, it is used instead of that attribute. It is illegal for those elements to have both a `value` attribute and an inner `script` element. The return type of the function should be consistent with the type of the parent element value.

The script element may contain language attribute (the values "`pnuts`" and "`mathml`" are supported).

Example 26

```
<script>

  <![CDATA[PI()*2.4]]>

</script>
```

4 Derived External Format of CSML

4.1 CSML Simulation Updating Specification

In CSML3.0, each element of a project has just zero or one simulation model. For updating the simulation parameter in the project, the other CSML updating XML format is developed. The detail is documented in “CSML Simulation Updater Ver3.0” and “http://www.csml.org/updater/3_0/”.

4.2 CSML Layout Specification

In CSML3.0, main model and its submodels can take automatic layout algorithms as their views. To deal with this specification, the other CSML layout format is developed. The detail is documented in “CSML Layout Format Ver1.0” and “http://www.csml.org/layout/1_0/”.

Example 27

```
<csml-layout:layout>

  <layoutOptionSet>

    <crossBioGridLayoutOptin gridSize=”1000” canvasGridWidth=”150”
canvasGridHeight=”150” degradationDelete=”true” comboWeightType=”3”/>

    <randomGridLayoutOption gridSize=”1000” canvasGridWidth=”150”
canvasGridWidth=”150” mutationRatio=”50”/>

    <sugiyamaLayoutOption alignment=”left|center|right”/>

  </layoutOptionSet>

</csml-layout:layout>
```