SBML To SAL Translator Documentation

Sri Paladugu

August 22, 2003

I. Introduction

This manual is a reference for the SBML_SAL_Translator. As the name aptly suggests SBML_SAL_Translator translates SBML (Systems Biology Markup Language) files to HybridSAL files. Currently this program supports SBML level 2 version 1. For more information about SBML please visit http://www.sbml.org/ on the internet. This document also explains the mappings between the SBML and HybridSAL components and the API library in detail.

SBML_SAL_Translator extensively uses Apache Xerces Parser version 1.4.4. This is available from http://xml.apache.org/xerces-j. Since the Translator is implemented in JAVA it is assumed that JDK 1.4 is installed on the machine where the translator is to be executed.

Some important features of this parser include:

1. The parser is event based SAX 2.0 (Simple API for XML) and loads the SBML data into java data structures that mirror the classes in SBML specification. No intermediate DOM (Document Object Model) is used which greatly reduces the run time memory usage.
2. The translator is written with SBML level 2.0 version 1 in mind, so the translator does not support earlier versions of SBML.
3. The translator uses the Apache Xerces-JAVA XML library, which supports full XML schema validation. All Xml and Schema warning, fatal error messages are logged with line and column number information and may be retrieved and manipulated pragmatically.

II. API Specification

Compartment

A compartment structure acts as a container for finite volume of Species.
<u>Getter Methods</u>:
String getId()
Returns the Compartment's identifier.

Returns the size of ArrayList holding the Event Assignment Expressions.
ArrayList getListOfEventAssignments()
Returns an ArrayList of Event Assignment Expressions.
int getListOfEventVariablesSize()
Returns the size of ArrayList containing variables (species) involved in constructing the Event Assignment Expressions.
ArrayList getListOfEventVariables()
Returns the ArrayList containing the variables (species) involved in constructing the Event Assignment Expression.
Caveat:
Presently there are no methods to get the value of Event Id and timeUnits fields. A getter method for getting the timeUnits field can be added in future if time delay feature is supported by SAL.

FunctionDefinition

A FunctionDefinition structure associates an identifier with a function definition. This identifier can be used subsequently in mathematical formulas.
Getter Methods:
String getFunctionId()
Returns the Identifier associated with the FunctionDefinition.
int getListOfFunctionVariablesSize()
Returns the size of the ArrayList containing the arguments to the Function.
ArrayList getListOfFunctionVariables()
Returns the ArrayList containing the arguments to the Function.
String getFunctionExpression()
Returns the Expression making up the FunctionDefinition.

KineticLaw

A KineticLaw structure contains information about the rate at which a reaction takes place.
Setter Methods:
void cloneListOfParameters(ArrayList listOfParameters)
Copies the input listOfParameters to the ListOfParameters in the KinecticLaw Structure.
Getter Methods:
String getFormula()

Returns the kineticFormula.
int getListOfParametersSize()
Returns the size of ArrayList containing the parameters involved in defining the Kinetic Formula.
ArrayList getListOfParameters()
Returns the ArrayList containing the parameters involved in defining the Kinetic Formula.

ModifierSpeciesReference

A ModifierSpecies is one which is neither a reactant nor a product but just acts as a catalyst or inhibitor in a reaction. The ModifierSpeciesReference structure can be used to hold reference information about such kind of Species. It is important to note that the ModifierSpeciesReference merely serves as a reference holder for the modifier speices listed in model's listOfSpecies.
Getter Methods:
String getSpecies()
Returns the identifier of the ModifierSpecies.

Parameter

A Parameter structure can be used to declare a variable for use in Mathematical formulas. Parameters can be local to a reaction or global to an entire model.
Getter Methods:
String getId()
Returns the identifier associated with the parameter.
String getValue()
Returns the value assigned to the parameter.

Reaction

A reaction structure can be used to hold the information about a chemcial reaction which can transform the amount of one or more species. A reaction structure can hold information about a reactant, product and modifier species along with the KineticLaw describing the rate at which a reaction takes place.
Getter Methods:

String getId()
Returns the Identifier associated with a reaction.
String getReversible()
Returns a String ("true"/"false") indicating whether a reaction is reversible or not.
int getListOfReactantsSize()
Returns the size of ArrayList containing Reactant species involved in defining the Reaction.
ArrayList getListOfReactants()
Returns the ArrayList containing Reactant species involved in defining the Reaction.
int getListOfProductsSize()
Returns the size of ArrayList containing Product species involved in defining the Reaction.
ArrayList getListOfProducts()
Returns the ArrayList containing Product species involved in defining the Reaction.
int getListOfModifiersSize()
Returns the size of ArrayList containing Modifier species involved in defining the Reaction.
ArrayList getListOfModifiers()
Returns the ArrayList containing Modifier species involved in defining the Reaction.
KineticLaw getKineticLaw()
Returns the KineticLaw guiding the rate of Reaction.
Setter Methods:
void cloneListOfReactants(ArrayList listOfReactants)
Copies the listOfReactants to the ArrayList holding the Reactant Species information in the Reaction Structure.
void cloneListOfProducts(ArrayList listOfProducts)
Copies the listOfProducts to the ArrayList holding the Product Species information in the Reaction Structure.
void cloneListOfModifiers(ArrayList listOfModifiers)
Copies the listOfModifiers to the ArrayList holding the Modifier Species information in the Reaction Structure.
void cloneKineticLaw(KineticLaw kineticLaw)
Copies the kineticLaw to the KinecticLaw object in the Reaction Structure.

Rule

A Rule structure can be used to hold information about different kinds of constraints placed on variables (species/paramters). The same Rule Structure can hold information regarding three different Rules (Rate/Algebraic/Assignment) defined in SBML level 2
Getter Methods:
String getVariable()
Returns the Left Hand Side of Rule expression. It will be zero in case of Algebraic Rule, a Species or Parameter in case of Assignment and Rate Rules. It is important to note that only the Variable in the Left Hand Side of Rate Rule is stored in the Rule structure instead of the derivative along with the Variable.
String getExpression()
Returns the Right Hand Side of the Rule Expression.
int getListOfRuleVariablesSize()
Returns the size of ArrayList containing the species/parameters involved in definining the Rule.
ArrayList getListOfRuleVariables()
Returns the ArrayList containing the species/parameters involved in defining the Rule.

Species

A Species structure can be used to hold information about chemical entities (reactant/product/modifier) that take part in reactions.
Getter Methods:
String getId()
Returns the identifier associated with the Species.
String getInitialAmount()
Returns the initial quantity of the species in the compartment.
String getCompartment()
Returns the identifier of the Compartment to which the species belongs.
String getBoundaryCondition()
Returns a String (true/false) indicating whether the species is on the boundary of a reaction system or not.
String getConstant()
Returns a String (true/false) indicating whether the species concentration

can be determined by rules and reactions or not.
If this method returns true, then the species concentration remains constant and cannot be determined by rules or reactions.

SpeciesReference

A SpeciesReference structure can be used to hold information about a species (reactant/product only) listed in listOfSpecies. A SpeciesReference structure merely serves as a information holder to refer to the species listed in the model's listOfSpecies. This is similar to ModifierSpeciesReference structure which is used to hold reference for a Modifier Species.
<u>Getter Methods</u>:
String getSpecies()
Returns the identifier associated with a species.
String getStoichiometry()
Returns the stoichiometry associated with a species.
<u>Caveat</u>:
Presently there are no APIs corresponding to Unit Structure and Unit Definition Structure present in SBML. We assume that the units for concentration, time etc., are consistent across all the species.

III. Mappings between SBML components and Hybrid SAL components.

SBML Components	**Hybrid SAL Place-Holders**
Model	Context
Compartment	Module
Species	LOCAL
Parameter	LOCAL/INPUT/OUTPUT
Assignment Rule	INVARIANT
Algebraic Rule	INVARIANT
Rate Rule	TRANSITION
Event	TRANSITION

Global Parameters and Global Rules of SBML will be stored in ENVIRONMENT Module of Hybrid SAL. The ENVIRONMENT Module acts as a place holder for variables which are common to more than one module. Thus information regarding Global parameters and Global Rules of SBML will be stored in ENVIRONMENT Module of Hybrid SAL and communicated to other Modules through the INPUT/OUTPUT sections.

IV. Installation

Sbml_Sal_Translator heavily depends on Xerces Java Parser 1.4.4 for its parsing functionality. This requires the user to install Xerces Java Parser 1.4.4 on the local machine which is freely available from http://xml.apache.org/xerces-j/. After installation the location of xerces.jar needs to be added to the CLASSPATH environment variable. All the SBML Classes which the translator uses are packaged under the directory sbmlClasses. Place the sbmlClasses directory in the same directory where Sbml_Sal_Translator is present.

The translator can be invoked using the following command in Unix/Linux:
Sbml_Sal_Translator {SBML File Name}

The above command results in the creation of a Hybrid SAL file with the same name as the value of Model Identifier in the input SBML file.

V. References

[1] A.Tiwari. HybridSAL: Modeling and Abstracting Hybrid Systems, 2003. Computer Science Laboratory, SRI International, MenloPark, CA.
[2] A.Finney, M.Hucka. Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions, 2003. http://www.sbml.org.