Instruction for PARSEDAE

A table of contents

1 Objectives

The parsedae module converts the regulator-reaction equations (sanac file), which are constructed by the CADLIVE editor, into the differential and algebraic equations (DAEs) (checkdae file) including ordinary transcription and translation equations (TT), Conventional Mass Action (CMA), General Mass Action (GMA), simplified Michaelis-Menten equations (MM), and S-system. In addition, the parsedae module extracts the regulator-reaction equations from the sanac file to output the list of regulator-reaction equations in the XML format. The checkdae file employs the indexes of species' names so that one can understand the model or edit it manually. One is allowed to edit the checkdae file (mathematical equations) directly according to the instruction. The checkdae file is written in the text format.

sanac: synthesis and analysis of network architecture in computer. XML representation for regulator-reaction equations, an extension of SBML level 2.

2 Function

2.1 Mathematical conversion

Different conversion methods are selected for the two layers: the gene-protein layer and the metabolic layer. For the gene-protein layers, users select four types of conversion methods: CMA, TPP_STEADYSTATE_1, TPP_STEADYSTATE_2, and TPP_RAPID, which are suitable for the system where the concentrations of complexes neither can be neglected nor canceled. GMA and MM, which are generally employed for enzyme reactions, are not applicable for the gene layer, because they usually cancel the concentrations of the complexes. For the metabolic layer, users are able to select two conversion methods, GMA or MM. Of course, it is possible to apply the conversion of CMA to the metabolic layer, but they are not useful. Since the different conversion methods are applied to the different layers, the differential equations for the species common to both layers must be combined appropriately.

2.2 Extraction of regulator-reaction equations

The parsedae module extracts the regulator-reaction equations from the sanac file to output the list of regulator-reaction equations in the XML format.

3. Specification of input/output

3.1 Input of sanac file

See the commentary of parsedae.

3.2 Output of checkdae file

The parsedae program converts the sanac file into the checkdae file that consists of header, list of labels for used

parameters, parameter list, list of independent variables, variable list, temporary mathematical equations, differential and algebraic equations. Users are allowed to edit the checkdae file directly or manually.

The data structures are specified by the identification line starting from "==" or "=". The line starting from "==" indicates a block; the line from "=" a subblock. The line starting from "#" describes comments, which are skipped by the checkdae module. One data must be described in one line, but mathematical equations are allowed to occupy multiple lines by using CR-key. The format of mathematical models is described later.

The following characters are allowed to name species.

Species naming rules

Initial character

- upper and lower alphabets [a-z]｜[A-Z]
- number [0-9]
- left parentheses (

Second characters

- upper and lower alphabets [a-z]｜[A-Z]
- underbar _
- number [0-9]
- left and right parentheses ( )
- colon :
- hyphen ―
- period .

The period may be used for indicating a decimal point.

The grammar for mathematical equations mainly obeys the C programming language. Exceptions are:

・The use of minus sign (-) is allowed, but that of plus (+) not at the heads of equations.

・The sign of minus (-) never follows the symbols of addition (+), subtraction (-), multiplication (*), division (/), and power (^). The minus sign must be contained in parentheses. See the examples in the database.

3.2.1 Header part

The header is used to describe the comments that contain the title of a model, its associated information, and a conversion method. Users are not allowed to edit this part. It is identified by "== Model ==", as follows.

```
== Model ==
TITLE:Heat Shock (test);
INFO:This is a sample case;
CONVERSION TYPE
    GENE-PROTEIN:CMA;
    METABOLIC:GMA;
```

The labels are described with upper characters, and end at the colon ":". The data end at the semicolon ";". For each layer, the conversion type is selected from CMA / TPP_STEADYSTATE_1 / TPP_STEADYSTATE_2 / TPP_RAPID / GMA / MM / NONE.

3.2.2 List of labels for used parameters

The label of all the parameters used in the checkdae file are listed in the place identified by "==Used Parameters ==". Each label is registered at one line, and the lines for registration end at semicolons.

```
==Used Parameters ==
ka;
kd;
kp;
kpd;
kother;
```

The labels are given for each conversion type, as follows.

For CMA and DAEs (TPP)

ka	binding_association_rate_constant
kd	dissociation_rate_constant
kx	reaction_rate_constant
Kb	binding_constant
kp	translation_rate_constant
kpd	decomposition_rate_constant
km	transcription_rate_constant
kmd	decomposition_rate_constant
ktr	transport_rate_constant

For GMA,

kxg	reaction_rate_constant
f	power_coefficient(Target-Variable)

"Target-Variable" is named by combining the name of the species regarding "f" with "_GMA", as shown by "glutamine_GMA".

For MM,

Q	reaction_rate_constant
Kmich	Michaelis_constant(*Target-Variable*)

"Target-Variable" is named by combining the name of the species regarding "f" with "_MM", as shown by "glutamine_MM".

3.2.3 Parameter list

All the parameters employed in the DAEs are listed following the identification line "==Parameters==". A parameter is registered at one line.

```
== Parameters ==

### ka: binding association rate constant
ka[1] = ; binding_association_rate_constant_RNAP.cyt_s70.cyt
ka[2] = ; binding_association_rate_constant_RNAP.cyt _s32.cyt
ka[3] = ; binding_association_rate_constant_RNAP.cyt _D.cyt
ka[4] = ; binding_association_rate_constant_s32.cyt _DnaK.cyt
------
```

Put a value between = and ;, the value is used for calculation. The tag name for identifying each parameter follows a semicolon. When users create the label of a new parameter in DAEs, the label is required to be registered in the list of "used parameters". The parameter name is created by combining the name of modifier with the names of reactants by using the underbar, where the reactant names are sorted in the alphabetical order.

3.2.4 List of independent variables

The parsedae module registers the following species as independent variables (constants): the species with "totalAmount = constant" and with "massBalance=off", and the species for which the right hand side of the resultant differential equation are zero. If users need new constants, add them to this part. Duplicated indexes are not allowed. All the independent variables (constants) that are fixed as constant values despite time changes are listed following the identification line of " ==ConstantPlayers==". These constants must not be registered in the list of "Used Parameters".

```
== Constant Players ==

constantPlayer[ 1] = ;    D.cyt
constantPlayer[ 2] = ;    gene(s32).cyt
constantPlayer[ 3] = ;    gene(Protein).cyt
------
```

The constants are named by connecting the name of the species to its compartment with a period, as shown by "gene(s32).cyt". The initial three characters for the attribute of "compartment" are used. When the attribute of "initialAmount" has been set in the sanac file, its values are written between "=" and ";".

3.2.5 Variable list

All the variables employed in the DAEs file are listed following the identification line "==Variables==". The variables for algebraic and differential equations are registered following the identification line "= Algebraic Equation =" and "= Differential Equation =", respectively.

```
== Variables ==

= Algebraic Equation =

x[RNAP.cyt] = ;
x[s70.cyt] = ;
x[s32.cyt] = ;
------
```

```
= Differential Equation =

y[Protein-f.cyt] = ;
y[mRNA(s32) .cyt] = ;
y[mRNA(DnaK) .cyt] = ;
------
```

The label of the variables are provided by x and y for algebraic and differential equations, respectively. The variables are distinguished using the index of species. The name of [ ] is made by combining the name of the species with its compartment with a period, as shown by "mRNA(s32).cyt". The initial three characters for the attribute of "compartment" are used. When the attribute of "initialAmount" has been set in the sanac file, its values must be written between "=" and ";". Users can put a value between = and ;, the value is used for calculation.

3.2.6 Temporary mathematical equations

All the temporary mathematical equations in the checkdae file are listed following the identification line "==Temporary Expression==". We define two types of temporary expressions. One is for expressing the total concentration of enzymes, which is essential for connecting a gene regulatory network to a metabolic network. The other is for expressing the fluxes of GMA and MM. The former expressions are listed following the identification line "=Total Enzyme=". The latter ones following "=Flux=".

```
== Temporary Expressions ==

= Total Enzyme =
T_enzyme.cyt = x[enzyme.cyt] + x[enzyme:A._cyt] + x[enzyme:B.cyt];
------

= Flux =
flux_E.cyt_S1.cyt _S2.cyt _to_P1.cyt _P2.cyt = kxg[1]*y[E.cyt]^f[1]*y[S1.cyt]^f[2]*y[S2.cyt]^f[3];
------

= Flux =
flux_E.cyt _S1.cyt _S2.cyt _to_P1.cyt _P2.cyt = Q[1]*y[E.cyt]*y[S1.cyt]
                    /(Kmich[1]+y[S1.cyt]) *y[S2.cyt]/(Kmich[2]+y[S2.cyt]);
------
```

If expressions are long in length, users can make a new line.

The total concentration of an enzyme is expressed by adding "T_" to its name that includes the initial three characters of compartment.

The flux is expressed by adding "flux_" to the head of the sequential names consisting of a modifier, reactants and products, where each name is connected with an underbar, and "_to" is inserted between the reactants and the products. The names of the reactants and products are sorted in alphabetical order, respectively.

3.2.7 Differential and algebraic equations (DAEs)

All the algebraic equations and differential equations are listed following the identification line "== Algebraic Eqs. ==" and " == Differential Eqs. ==", respectively.

```
== Algebraic Eqs. ==

aeq_1: y[TRNAP.cyt] = x[RNAP.cyt] + x[RNAP:s70.cyt] + x[RNAP:s32.cyt] + x[RNAP:D.cyt];
aeq_2: y[Ts70.cyt] = x[s70.cyt] + x[RNAP:s70.cyt] + x[RNAP:s70:Enhancer(s32).cyt] +
x[RNAP:s70:D.cyt];
aeq_3: y[Ts32.cyt] = x[s32.cyt] + x[s32:DnaK.cyt] + x[s32:DnaK:FtsH.cyt] + x[s32:FtsH.cyt];
aeq_4: y[TDnaK.cyt] = x[DnaK.cyt] + x[s32:DnaK.cyt] + x[s32:DnaK:FtsH.cyt] + x[Protein:DnaK.cyt];
aeq_5: y[TFtsH.cyt] = x[FtsH.cyt] + x[s32:DnaK:FtsH.cyt] + x[s32:FtsH.cyt];
------

== Differential Eqs. ==

deq_1:dydt[Protein-f.cyt] = -kpd[4]*y[Protein-f.cyt] - kx[2]*y[Protein-f.cyt] + kx[3]*x[Protein:DnaK.cyt];
deq_2:dydt[mRNA(s32).cyt] = + km[2]*(x[RNAP:s70:Enhancer(s32).cyt] /y[TEnhancer(s32).cyt])
*constantPlayer[2] - kmd[4] *y[mRNA(s32).cyt];
deq_3:dydt[mRNA(DnaK).cyt] = + km[3]*(x[RNAP:s32:Enhancer(HSP).cyt]/y[TEnhancer(HSP).cyt])
*constantPlayer[4] - kmd[1] *y[mRNA(DnaK).cyt];
deq_4:dydt[mRNA(FtsH).cyt] = + km[4]*(x[RNAP:s32:Enhancer(HSP).cyt]/y[TEnhancer(HSP).cyt])
*constantPlayer[5] - kmd[2] *y[mRNA(FtsH).cyt];
deq_5:dydt[mRNA(Protein).cyt] = + km[1]*constantPlayer[3] - kmd[3] *y[mRNA(Protein).cyt];
------
```

The description of deq_#: are not necessary. Users can make a new line for one equation.

Users are able to edit DAEs directly in the file. Notice that the employed parameters and variables have to be declared following the identification lines: “==Used Parameters ==”, "==Parameters==", " ==ConstantPlayers==", or "==Variables==". In addition, users are able to create new parameters, variables, or equations by registering them using the above identification lines. Consequently, users are allowed to construct DAEs at almost free hands, if they keep the rules of the checkdae file.

3.3 Output for regulator-reaction equations

Data structure of the list of regulator-reaction equations is written in XML as follows:

```
chemical_eqs
├─ 1t title                    model name
├─ 1t memo                     model information
└─ + equations (layer)         list of regulator-reaction equations
     └─+t equation             regulator-reaction equations
```

Legend:

1: an element that appears once. *: an element that repeats more than zero.

+: an element that repeats more than one. ?: an element that appears once or zero.

t: an element that has text in content. (): provided as attributes

The element of <equations> has the attribute of "layer", showing which layer a regulator-reaction equation belongs

to. The "layer" is the gene, protein or metabolic layer. When multiple regulators act on the same reaction, e.g., gene1, using the tag
 groups these equations.

4 Startup

Users start and execute the parsedae program on the command line.

4.1 Environment

The parsedae module requires

- Java™ 2 SDK, Standard Edition Version 1.4
- Xerces2 Java Parser

We verifies the function by using Java™ 2 SDK, Standard Edition Version 1.4.1and Xerces2 Java Parser 2.2.1 Release. The following environmental variables are set.

- LIFESIM_ROOT Base directory for the simulator
- JAVA_HOME The directory where Java™ 2 SDK is installed
- LD_LIBRARY_PATH The path to Java™ 2 SDK and DLLs for the simulator

LD_LIBRARY_PATH is set as follows;

LD_LIBRARY_PATH=$JDK/jre/lib/i386:$JDK/jre/lib/i386/client:$JDK/jre/lib/i386/native_threads:$LIFESIM_ROOT/Tools/lib

4.2 Conversion

Command for parsing the sanac file:

parsedae –NMathNCE.xml –UMathUsage.txt –DMathDAE.txt [-gCONVTYPE] [-mCONVTYPE]

The file MathNCE.xml is parsed. No space between the option and the file is allowed.

- MathNCE.xml Input: sanac file
- MathUsage.txt Output: Usage file
- MathDAE.txt Outpur: checkdae file

The index –g and –m determines the conversion method for the gene-protein layer and the metabolic layer. The CONVTYPE (conversion type) is provided by:

- CMA conventional mass action

- TPP_STEADYSTATE_1 two-phase partition: steady state I
- TPP_STEADYSTATE_2 two-phase partition: steady state II
- TPP_RAPID two-phase partition: rapid balance
- GMA Generalized Mass Action
- MM Michaelis-Menten type

Users may sort the option arbitrally, but the options, -N, -U, and –D are required. The CONVTYPE appointed by –g and –m, which are indispensable, is restricted to the above conversion. Otherwise, conversion fails.

4.3 Function for extracting regulator-reaction equations

Command:

%parsedae –CMathNCE.xml –PMathNCEphp.xml

The space between the flag and the file is not allowed.

- MathNCE.xml Input :sanac file (regulator-reaction equations created by the CADLIVE editor)
- MathNCEphp.xml Output : file for the list of regulator-reaction equations

Users sort the option arbitraly, but must appoint both.

5 Lists of error message

5.1 Argument error

"initialize: invarid argument !!"

"initialize: 'GMA' and 'MM' are not allowed for Gene-Protein layer."

"initialize: The conversion method is not specified to both Gene-Protein and Metabolic layer."

"initialize: The processing method is not set up correctly."

5.2 Read error

In error messages, %NODE_NAME% is the node name, %ATTR_NAME% is the attribute name, %NUM% is the number, %STRING% is the characters.

"element SANAC not found."

"node '%NODE_NAME%' not found."

"node '%NODE_NAME%' found 2 or more times."

"ListOfCompartments.numberOfCompartments is not equal to actual number of compartments."
"ListOfSpecies.numberOfSpecies is not equal to actual number of species."
"ListOfElements.numberOfElements is not equal to actual number of elements."
"ListOfReactions.numberOfReactions is not equal to actual number of reactions."
"ListOfReactants.numberOfReactants is not equal to actual number of reactants."
"ListOfProducts.numberOfProducts is not equal to actual number of products."
"ListOfModifiers.numberOfModifiers is not equal to actual number of modifiers."

"specieReference.specieID [%NUM%] is not defined in ListOfSpecies."

"attribute '%ATTR_NAME%' in '%NODE_NAME%' not found."

"invalid %ATTR_NAME% value '%STRING%'."

5.3 Conversion error (CMA / TPP / gene1)

In error messages, %NAME% is the name of the specie、%REACTION_TYPE% is the reaction type, %NUM% is the number.

"Decomposition: The element of complex '%NAME%(specieID=%NUM%)' must be a variable of 'CMA/TPP'."

"Decomposition: '%NAME%(specieID=%NUM%)' has an element which is not a variable of 'CMA/TPP'."

"checkStoichiometricDiscordance2: rightpart has %NUM% elements of complex, but leftpart has %NUM% monomerized players.)"

"checkStoichiometricDiscordance2: '%NAME%(specieID=%NUM%)', element of rightpart complex matches for no leftpart monomerized players."

"checkStoichiometricDiscordance: No products and modifierComplex: rightpart of binding reaction must include modifierComplex or one product."

"convertReaction_CMA_gene1: gene1 reaction needs 1 reactant and 1 product."

"convertReaction_CMA_gene1: modifier '%NAME%(specieID=%NUM%)' includes no transcriptional regulator."

"convertReaction_CMA_gene1: modifier '%NAME%(specieID=%NUM%)' has illegal modifierType 'enzyme' for gene1 reaction."

"convertPlayer_CMA: constant variable '%NAME%(specieID=%NUM%)' is must be decomposition:off."

"no reactants exists."
"no products exists."
"no modifiers exists."
"no modifierComplex exists."

"modifiers exists."
"modifierComplex exists."

"classifyReaction_CMA: invalid reactionType for reaction with modifier."

"classifyReaction_CMA: invalid reactionType for reaction without modifier."

"reacReactions_CMA: reaction conversion failed.(reactionID=%NUM%,reactionType=%REACTION_TYPE%)"

5.4 Conversion error (GMA / MM)

In error messages, %NAME% is the name of the specie 、 %REACTION_TYPE% is the reaction type, %SPECIE_CLASS% is the attribute of class , %NUM% is the number.

"createTotalEnzymeEq: '%NAME%(specieID=%NUM%)' must not be a reactant or product of 'GMA/MM', because its massBalance is 'ON'."

"createTotalEnzymeEq: '%NAME%(specieID=%NUM%)' must not be a reactant or product of 'GMA/MM', because its specieClass is %SPECIE_CLASS%."

"checkPlayer_GMA: constant variable '%NAME%(specieID=%NUM%)' is must be decomposition:off."

"checkPlayer_GMA: '%NAME%(specieID=%NUM%)' must not be a player of 'GMA/MM', because its a complex of 'CMA/TPP'."

"checkReaction_GMA: invalid reactionType for GMA/MM conversion.(reactionID=%NUM%,reactionType =%REACTION_TYPE%)"

"checkReaction_GMA: The error detected with checking reactant.(reactionID=%NUM%)"

"checkReaction_GMA: lack of reactants.(reactionID=%NUM%)"

"checkReaction_GMA: The error detected with checking product.(reactionID=%NUM%)"

"checkReaction_GMA: The error detected with checking modifier.(reactionID=%NUM%)"