Tutorial for the CellML Metabolic Component Library (MCL)

Introduction

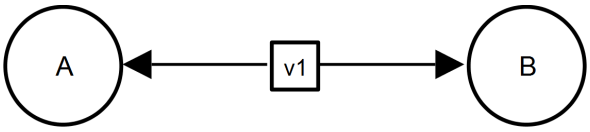
Metabolic networks often reuse kinetic equations from a standard set of defined reaction mechanisms. Consequently, a library of these standard components in a standard format could facilitate the development of metabolic models. Reuse of annotated components could support the identification of common patterns and provide an additional meta-level for the understanding of kinetic metabolic modeling. A standard metabolic component library was developed in CellML, a standard XML based language for the encoding of kinetic models, and usage examples are provided. The current version of the library is available via
http://models.cellml.org/w/matthiaskoenig/MetabolicComponentLibrary
http://www.charite.de/sysbio/people/koenig/software/cellml-mcl/

Examples

The usage of MCL components in models via the import statement of CellML is demonstrated in the following examples. All examples are provided with the library and are found in the examples folder

MassAction 1

Example MassAction1 is a simple reaction network consisting of one reaction (v1) of the the type MassActionUniUni. The example shows the basic use of components via the import function in CellML 1.1.



v1 (A ↔ B) [MassActionUniUniRev]

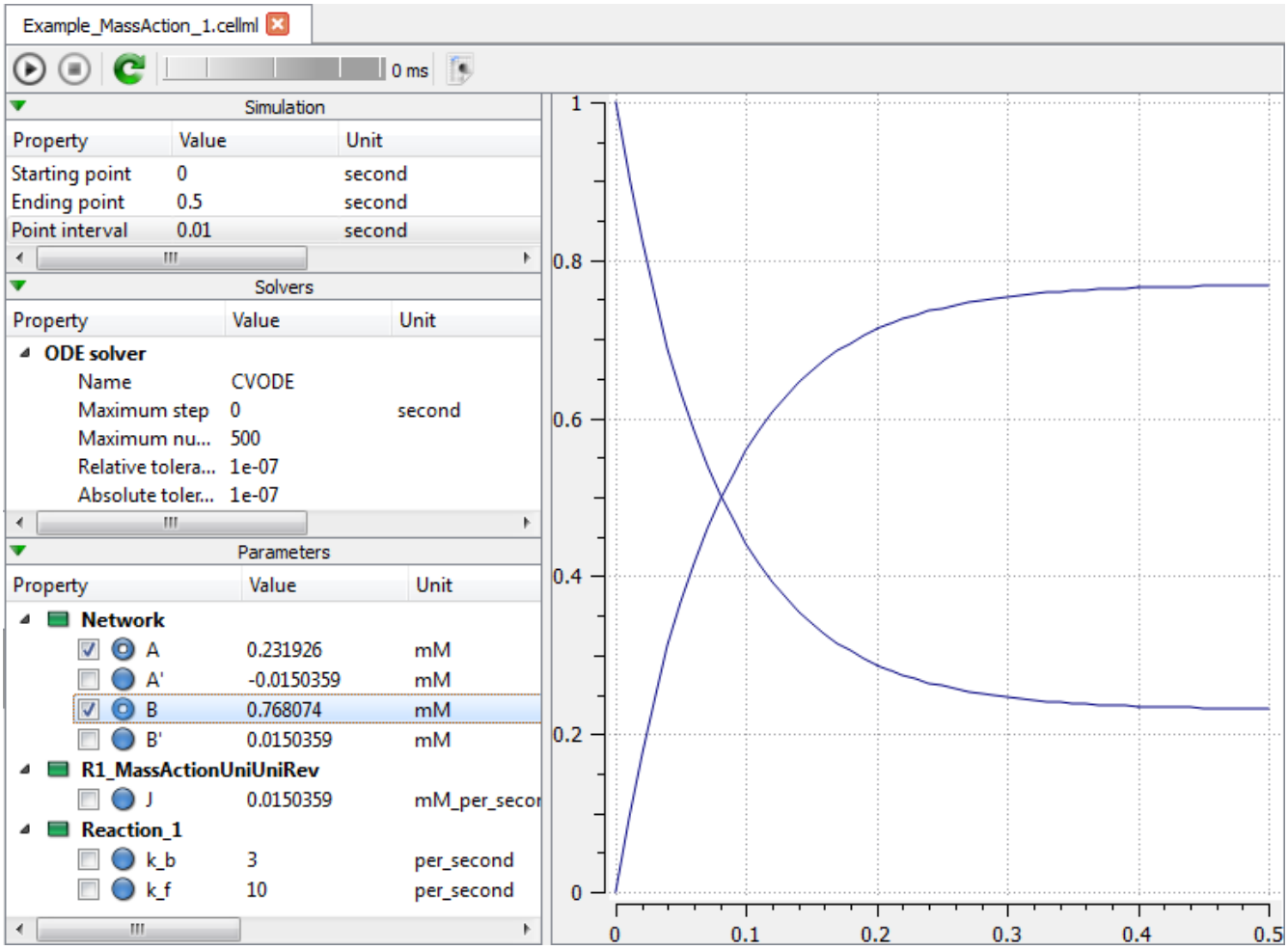
The MassActionUniUniRev component as well as the units of the MCL are imported.

```
<model xmlns="http://www.cellml.org/cellml/1.1#"
xmlns:cmeta="http://www.cellml.org/metadata/1.1#" cmeta:id="Example_MassAction_1"
name="Example_MassAction_1">

<import xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="MetabolicComponentLibrary-v0.1.cellml">
      <component component_ref="MassActionUniUniRev" name="R1_MassActionUniUniRev"/>
      <units name="mM" units_ref="mM"/>
      <units name="per_second" units_ref="per_second"/>
      <units name="mM_per_second" units_ref="mM_per_second"/>
      <units name="per_mM_per_second" units_ref="per_mM_per_second"/>
      <units name="per_mM" units_ref="per_mM"/>
</import>
...
</model>
```

The ODEs are defined in the top component Network. The component Reaction_1 is the reaction which calculates the flux v1. The actual implementation of the flux calculation is performed in the MCL component MassActionUniUniRev.

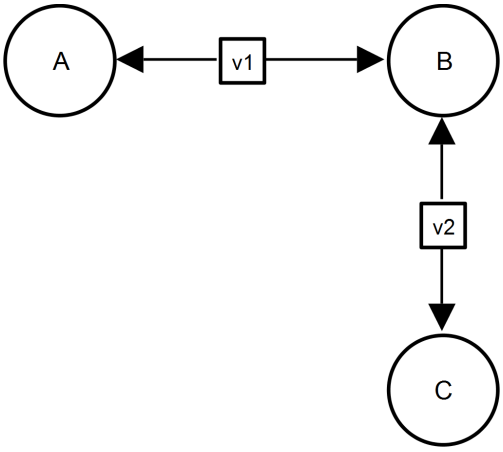
```
def comp Network as
	var t: second;
	var v1: mM_per_second {pub: in};
	var A: mM {init: 1, pub: out};
	var B: mM {init: 0, pub: out};

	ode(A, t) = -v1;
	ode(B, t) = v1;
enddef;

def comp Reaction_1 as
	var v1: mM_per_second {pub: out, priv: in};
	var A: mM {pub: in, priv: out};
	var B: mM {pub: in, priv: out};
	var k_f: per_second {priv: out};
	var k_b: per_second {priv: out};

	k_f = 10{per_second};
	k_b = 3{per_second};
enddef;

def map between Network and Reaction_1 for
	vars v1 and v1;
	vars A and A;
	vars B and B;
enddef;

def group as encapsulation for
	comp Reaction_1 incl
		comp R1_MassActionUniUniRev;
	endcomp;
enddef;

def map between Reaction_1 and
R1_MassActionUniUniRev for
	vars v1 and J;
	vars A and S;
	vars B and P;
	vars k_f and k_f;
	vars k_b and k_b;
enddef;
```

Simulation results from OpenCOR:


Example_MassAction_1.cellml
0 ms
Simulation
Property
Value
Unit
Starting point
0
second
Ending point
0.5
second
Point interval
0.01
second
Solvers
Property
Value
Unit
ODE solver
Name
CVODE
Maximum step
0
second
Maximum nu...
500
Relative tolera...
1e-07
Absolute toler...
1e-07
Parameters
Property
Value
Unit
Network
A
0.231926
mM
A'
-0.0150359
mM
B
0.768074
mM
B'
0.0150359
mM
R1_MassActionUniUniRev
J
0.0150359
mM_per_secor
Reaction_1
k_b
3
per_second
k_f
10
per_second
1
0.8
0.6
0.4
0.2
0
0
0.1
0.2
0.3
0.4
0.5

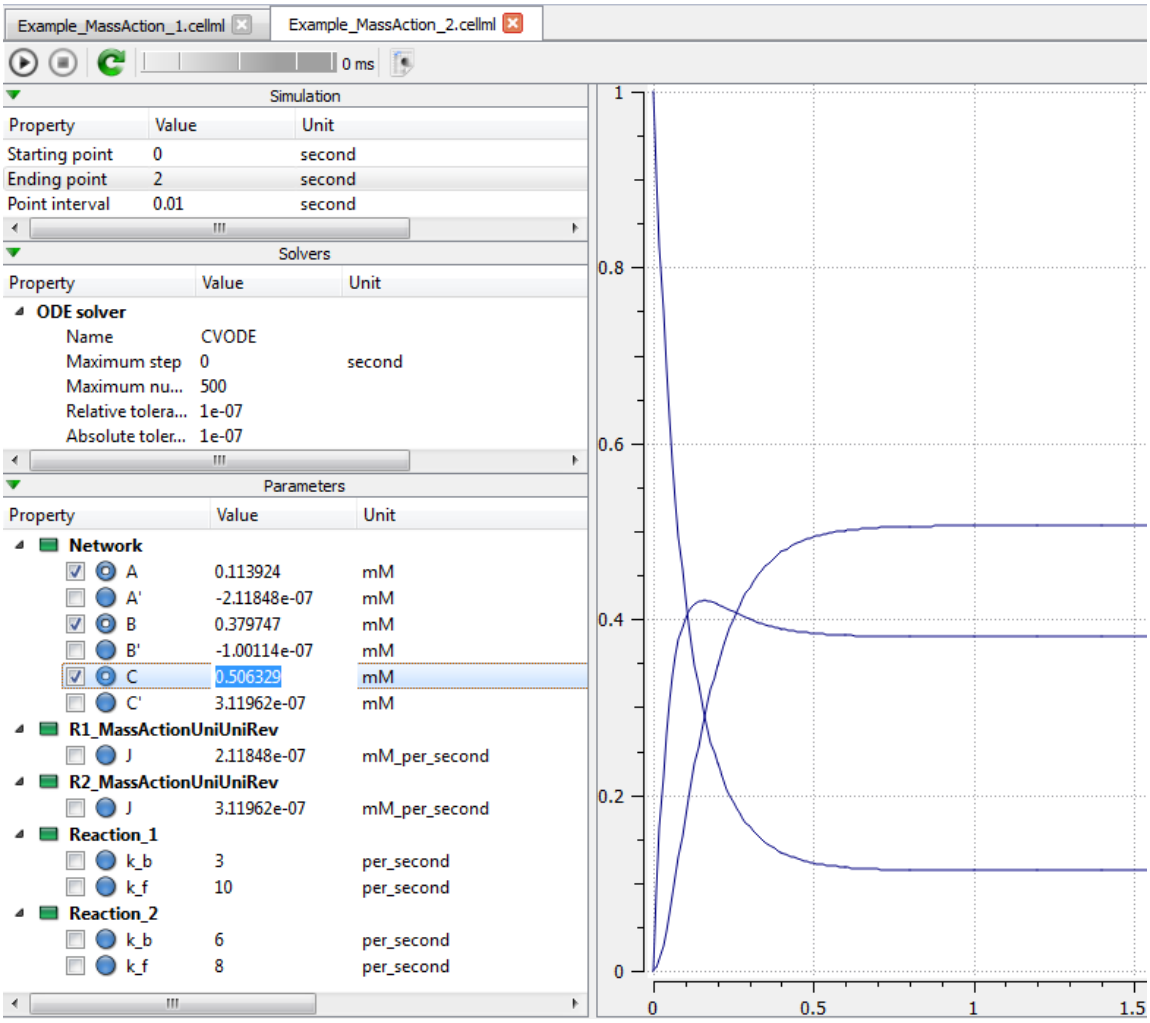
Mass Action 2

Example MassAction2 is a simple reaction network consisting of two reactions (v1, v2) of the type MassActionUniUniRev. This example demonstrates the multiple reuse of a component from the MCL.



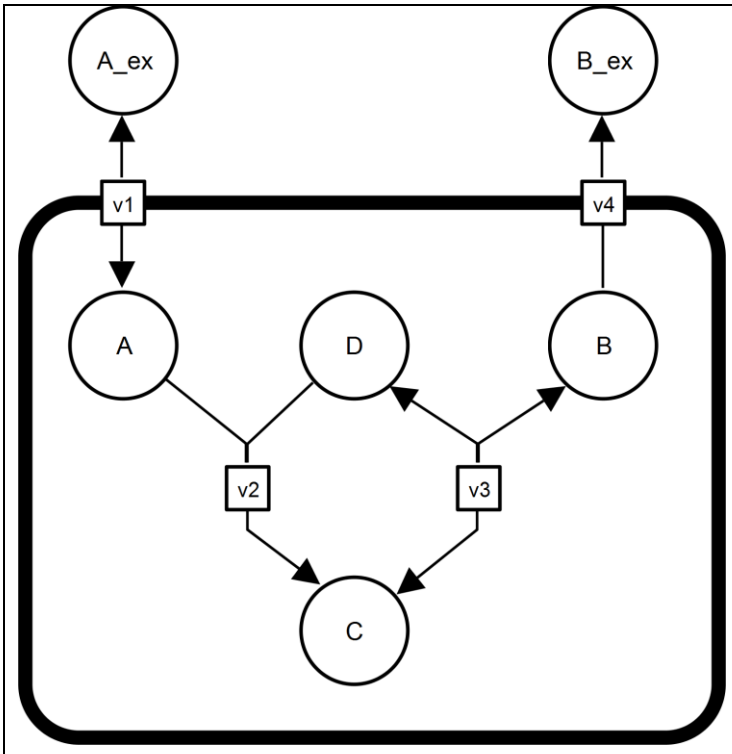
v1 (A ↔ B) [MassActionUniUniRev]

v2 (A ↔ B) [MassActionUniUniRev]



Mass Action 3

Example MassAction3 is a simple reaction demonstrates the use of multiple different components from the MCL.



v1 (A_ex ↔ A) [MassActionUniUniRev]

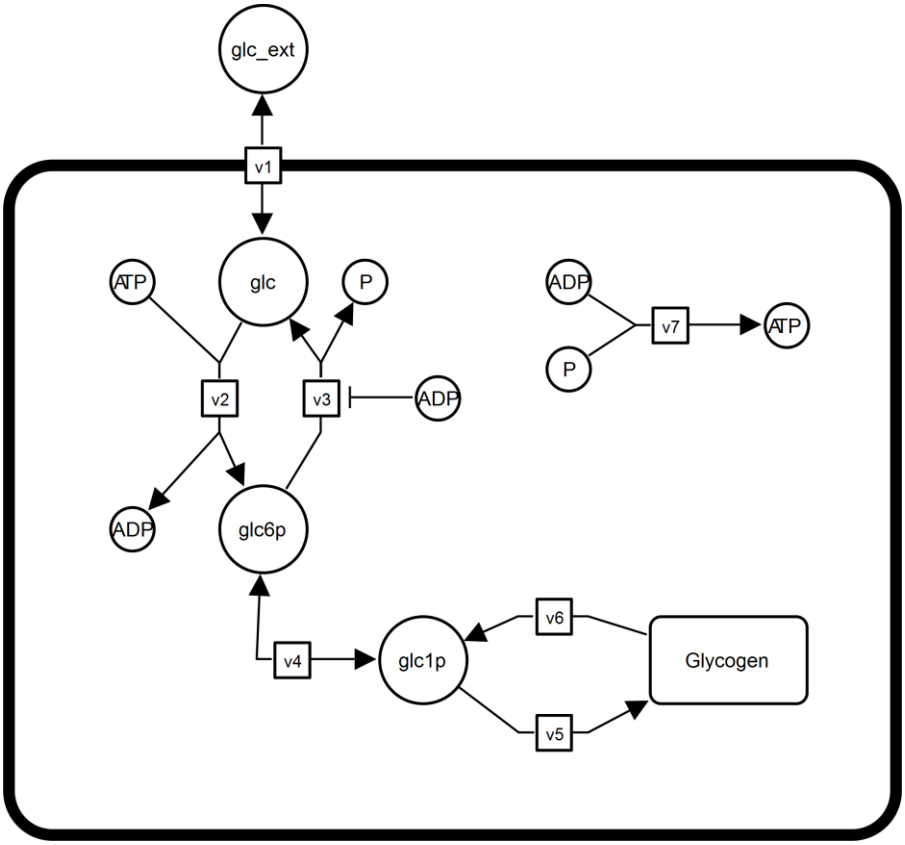
v2 (A+D ↔ C) [MassAction]

v3 (C ↔ B+D) [MassAction]

v4 (B ↔ B_ex) [MassAction]

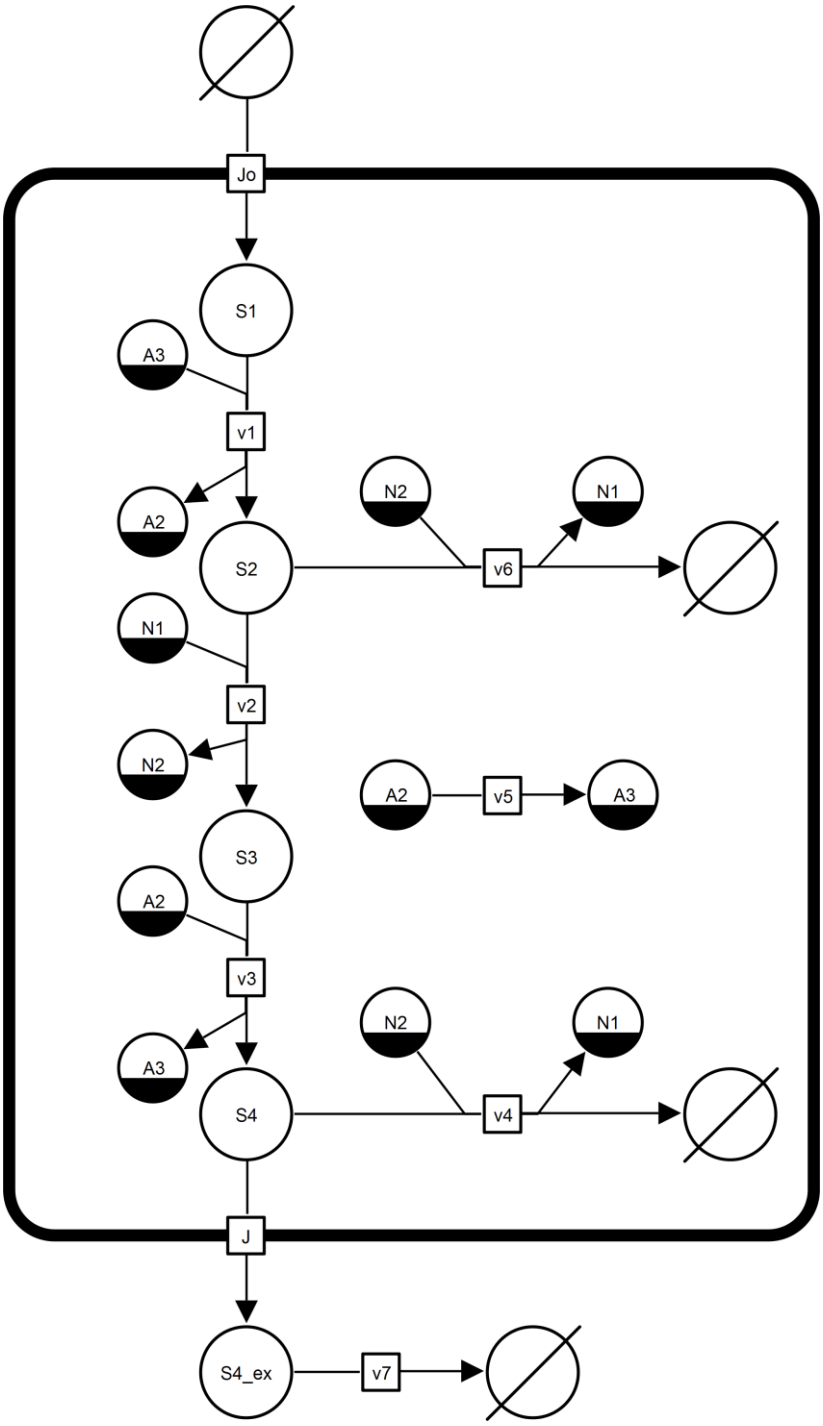
Example Full 1

More complex real world example.



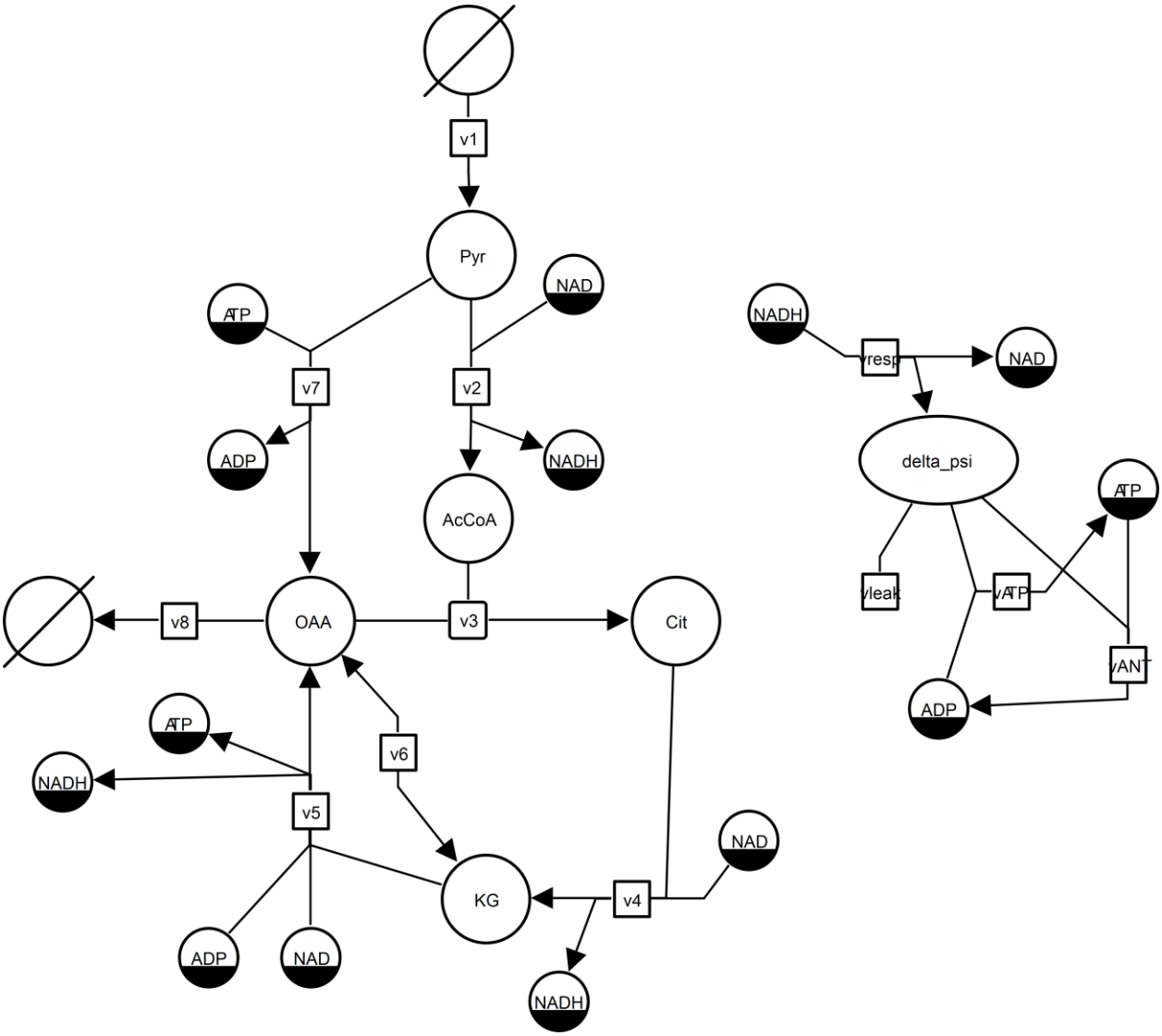
Wolf and Heinrich (2000)

Real World example.



Nazareth et al. (2009)

Real World example.



Koenig et al. (2012)

Real World example.

Using Library Components in CellML

Best practice for model encoding & structuring of component libraries

Currently information is lacking on how to best code a model in CellML. Furthermore, no information is available how libraries of components should be structured and used. A standard document should be provided with examples to provide guidelines (start with an electro-physiological library). How should libraries be documented and be made accessible? How to handle changes in versions? Notifications? Tests of the single components? Provide backwards compatibility?

Lack of tools for reuse of components & the editing with imports

Tools are missing which support the rapid model encoding with the use of imports of predefined components

- COR and OpenCell are not solving these issues and result often non-functioning models. COR is always overwriting the CellML header comments and converts files from CellML 1.1 to 1.0, thereby breaking the imports.
- JSim is not supporting imports
- COR is not supporting imports
- VCell did not work at all with CellML
- OpenCOR does currently not have a working editor

To create the examples XML parts had to be written from scratch. The infrastructure supporting imports and a component library is currently not fully available! Nobody will reuse the components if the software is not supporting it!
There is no working CellML editor supporting imports!

pub:in / pub:out / priv:in /priv:out & encapsulation nightmare

The connecting of the components via mappings is too difficult and not very intuitive. Especially the necessity to encapsulate if variables are passed through an intermediate layer is very annoying.
This has to be solved automatically by the CellML-API and a modeler should not worry about the priv/pub definitions. With additional encapsulation layer the priv/pub definitions have to be changed again and again.

How to best encode metabolic networks?

With the removal of the reaction component it is completely unclear how to encode metabolic networks in CellML (problem related to best practice guidelines). A mayor problem is the lack of the stoichiometric matrix (which comes in SBML naturally from the species/reaction model. How can I best obtain the stoichiometric matrix from a CellML model not encoded with the ‘reaction’ component. Especially for visualization this is a critical issue.

Annotation standard & annotation tools ?

“*…model authors are strongly encouraged to appropriately and comprehensively annotate their models, as well as reuse their models or those of others as often as possible.”* (Garny, 2008)

CellML API functions are necessary to write and read annotations in a standard way, namely MIRIAM annotations. Currently, this is not possible from the available standard editors and the API.

Outlook

- Provide simple example files and the implementation of 2-3 standard models
- SBO annotations of reaction mechanisms
- LinLog kinetics, generalized kinetics, power law based kinetics
- Mass-Action & Power-Law example generator from stoichiometric matrix and given reversibility/irreversibility