



Towards the CellML IDE

Justin Marsh (Auckland Bioengineering Institute)



Other CellML Tools

- Cell Elite
- insilicoIDE
- JSim
- mozCellML



Core components

- Compilation
- Editing
- Exploration



Core components

- Simulation
- Editing
- Display



Core components

- Not every facet, but enough
- Round tripping
- Inputs and outputs
- External tools for extended functionality



Simulation

- Compilation
 - CellML transformed into a programming language form, which is compiled and executed.
- Inputs and outputs
 - Parameter sets
 - Compiled code, data sets
- Simulation Metadata



Editing

- Tree view
 - Filtered views
- XML view
- Validator



Display

- Tree view
 - Filtered views
- XML view
- Validator
- Graphing
 - Graphing Metadata
- Session files
 - Linked content

Tree view

Type	Value	Units
Components		
environment		
a		
a	3	picomolar
c	time	dimensionless
o		nanomolar
B		nanomolar
time		minute
Mathematics	1	
Equation		
Equation		
o		
Groups		
Connections		
a, environment		
o, environment		
a, o		
Units		
picomolar	Derived	
nanomolar	Derived	
per_minute	Derived	
minute	Derived	

Tree view

Type	Value	Units
▷ C	0.38	dimensionless
▷ gamma	1.0	dimensionless
▷ K	0.1	dimensionless
▷ RP	1.0	dimensionless
▷ sigma	10	dimensionless
▷ thetal	1.0	dimensionless
▷ lambda	1.0	dimensionless
▷ thetaE	0.01	dimensionless
▷ VCs	1.0	dimensionless
▷ Vsm	1.0	dimensionless
▷ KsE	0.1	dimensionless
▷ V1m	50.0	dimensionless
▷ K1C	0.1	dimensionless
▷ K1	0.0001	dimensionless
▷ V2m	40	dimensionless
▷ K2	0.0001	dimensionless
▷ V3m	3000	dimensionless

Tree view

Type	ID	Parent Component	
▶ Mathematics	1	C	
▶ Mathematics	2	K	
▶ Mathematics	3	RP	
▶ Mathematics	4	E	
▶ Mathematics	5	KP	
▼ Mathematics	<input type="text" value="6"/>	KPI	
≡ Equation			
▶ Mathematics	7	I	
▶ Mathematics	8	RE	
▶ Mathematics	8a	R	
▶ Mathematics	9	Vs	
▶ Mathematics	10	V1	
▶ Mathematics	11	V2	
▶ Mathematics	12	V3	
▶ Mathematics	13	V4	
▶ Mathematics	14	Vd	



External tools

- Physiome Model Repository
- CellML to CM
- CellML to OpenCMISS
- CellML2DOT
- CellMLSimulator



CellML API

- Each module as a service
 - Validation, Code Generation, Integration (Simulation), Annotation, CellML DOM instantiation and manipulation, Units conversion, other helper services
- Communication and invocation via XPCOM, CORBA, language bindings or FFI



CellML API

- Use case: basis for external tools
 - Extract information on components
 - Instantiate a CellML file into a DOM like model
 - Validate CellML models
 - Generate code from CellML
- Services have minimal dependencies



Round tripping

- Name is misleading, but standard
- More to do with a method of ensuring synchronisation
- Changes occur at one point, and propagate out



Round tripping

- Export generated code
- Use in external application
- Alter CellML based on results, rather than altering the generated code directly

Round tripping

The screenshot displays the Physiome CellML Environment interface. A 'Procedural Code Export' dialog box is open, showing the following code:

```
ALGEBRAIC(7) = p[0];
ALGEBRAIC(8) = p[1];
ALGEBRAIC(9) = p[2];
}

function RATES = model(VOI, STATES)

STATES(1) = 0.38;
CONSTANTS(1) = 1.0;
STATES(2) = 0.1;
STATES(3) = 1.0;
CONSTANTS(2) = 10;
CONSTANTS(3) = 1.0;
CONSTANTS(4) = 1.0;
CONSTANTS(5) = 0.01;
CONSTANTS(6) = 1.0;
CONSTANTS(7) = 1.0;
CONSTANTS(8) = 0.1;
CONSTANTS(9) = 50.0;
CONSTANTS(10) = 0.1;
CONSTANTS(11) = 0.0001;
CONSTANTS(12) = 40;
CONSTANTS(13) = 0.0001;
CONSTANTS(14) = 3000;
CONSTANTS(15) = 0.0001;
CONSTANTS(16) = 3.0;
CONSTANTS(17) = 0.0001;
CONSTANTS(18) = 1000.0;
CONSTANTS(19) = 0.005;

rootfind_0(VOI, CONSTANTS, RATES, STATES, ALGEBRAIC, pret);
ALGEBRAIC(6) = CONSTANTS(12)*(ALGEBRAIC(4)/(CONSTANTS(13)+ALGEBRAIC(4)));
ALGEBRAIC(2) = CONSTANTS(9)*(STATES(1)/(CONSTANTS(10)+STATES(1)))*(STATES(2)
/(CONSTANTS(11)+STATES(2)));
RATES(2) = ALGEBRAIC(6) - ALGEBRAIC(2);
rootfind_1(VOI, CONSTANTS, RATES, STATES, ALGEBRAIC, pret);
ALGEBRAIC(11) = CONSTANTS(14)*ALGEBRAIC(4)*(ALGEBRAIC(9)/(CONSTANTS(15)+ALGEBRAIC(9)));
ALGEBRAIC(1) = CONSTANTS(16)*(STATES(3)/(CONSTANTS(17)+STATES(3)));
RATES(3) = ALGEBRAIC(11) - ALGEBRAIC(1);
ALGEBRAIC(12) = STATES(1) + CONSTANTS(18)*ALGEBRAIC(8)*
(STATES(1)/(CONSTANTS(19)+STATES(1)));
ALGEBRAIC(10) = CONSTANTS(6) + CONSTANTS(7)*(ALGEBRAIC(8)/(CONSTANTS(8)+ALGEBRAIC(8)));
```

The background interface shows a tree view of components (environment, C, K, RP, E, KP, KPI, I, RE, R, Vs, V1, V2, V3, V4, Vd) and connections. The 'Basic settings' tab is active, showing options for 'Start time point', 'Point density_max', 'End time point', 'Maximum step size' (set to 1), and 'Algorithm' (set to BDF 1-5 with solve).

Round tripping

The screenshot displays the 'Optimization Tool' window, which is divided into several sections for configuring an optimization problem.

Problem Setup and Results

- Solver:** fmincon - Constrained nonlinear minimization
- Algorithm:** Large scale
- Problem**
 - Objective function:** @model
 - Derivatives:** Gradient supplied in objective function
 - Start point:** 0
- Constraints:**
 - Linear inequalities:** A: [ALGEBRAIC STATES RATES] b: [1 5 1 1 0 0 1 0 1 9 9 0 3 3 0 9 1 1]
 - Linear equalities:** Aeq: beq:
 - Bounds:** Lower: Upper:
 - Nonlinear constraint function:**
 - Derivatives:** Approximated by solver
- Run solver and view results**
 - Buttons: Start, Pause, Stop
 - Current iteration: Clear Results
- Final point:**

Options

- Stopping criteria
- Function value check
- User-supplied derivatives
- Approximated derivatives
- Algorithm settings
- Multiojective problem settings
- Inner iteration stopping criteria
- Plot functions
- Output function
- Display to command window

Round tripping

The screenshot shows the Physiome CellML Environment interface. The main window displays a list of parameters for a model named 'hatzimanikatis_...'. The parameters are listed in a table with columns for Name, Type, Value, and Units. The 'RP' parameter is highlighted, and its value is 1.0. Below the table, a summary for the 'RP' parameter is shown:

Name	Type	Value	Units
C	dimensionless	0.38	dimensionless
gamma	dimensionless	1.0	dimensionless
K	dimensionless	0.1	dimensionless
RP	dimensionless	1.0	dimensionless
sigma	dimensionless	10	dimensionless
thetal	dimensionless	1.0	dimensionless
lambda	dimensionless	1.0	dimensionless
thetaE	dimensionless	0.01	dimensionless
Vcs	dimensionless	1.0	dimensionless
Vsm	dimensionless	1.0	dimensionless
KsE	dimensionless	0.1	dimensionless
V1m	dimensionless	50.0	dimensionless
K1C	dimensionless	0.1	dimensionless
K1	dimensionless	0.0001	dimensionless
V2m	dimensionless	40	dimensionless
K2	dimensionless	0.0001	dimensionless
V3m	dimensionless	3000	dimensionless
K3	dimensionless	0.0001	dimensionless
V4m	dimensionless	3.0	dimensionless
K4	dimensionless	0.0001	dimensionless
VdEm	dimensionless	1000.0	dimensionless
KdC	dimensionless	0.005	dimensionless

RP : {
 Units : dimensionless
 Initial Value : 1.0
 Private Interface : None
 Public Interface : Out

Node closed



Round tripping

- Export generated MathML
- Use in document
- When the model changes, regenerate the document with fresh MathML

Round tripping

The screenshot displays the Physiome CellML Environment interface. On the left, a 'Models' pane shows a tree view with 'chen_model_2007' selected. The main workspace contains a table of model components:

Name	Type	Value	Units
Fe3		0.3	micromolar
Fe3_Arg			
Fe2			
Fe2_Arg			
Fe3_O2_Arg			
Fe3_NOHA			
Fe2_NOHA			
Fe3_O2_NOHA			
Fe3_NO			
Fe2_NO			
NO			
citrulline			
NO3			
NOHA			
O2			
Arg			
k1			
k_1			
k2			
k3			
k4			

Below the table are 'Basic settings' and 'Advanced settings' tabs. The 'Basic settings' tab includes fields for 'Start time point', 'Point density_{max}', 'End time point', 'Maximum step size', and 'Algorithm' (set to BDF 1). An 'Integrate' button is located at the bottom of the workspace.

A 'Summary of Mathematics' dialog box is open in the foreground, displaying the following content:

```
==== Summary of Equations ====
==== Component environment =====
==== Component Fe3 =====
<apply xmlns="http://www.w3.org/1998/Math/MathML"><eq/><apply>
<diff/><bvar><ci>time</ci></bvar><ci>Fe3</ci></apply><apply>
<minus/><apply><plus/><apply><times/><ci>k_1</ci></apply>
<ci>Fe3_Arg</ci></apply><apply><times/><ci>k13</ci></apply>
<ci>Fe3_NO</ci></apply><apply><times/><ci>k12</ci></apply>
<ci>Fe2_NO</ci><ci>O2</ci></apply></apply><apply><plus/>
<apply><times/><ci>k1</ci><ci>Arg</ci><ci>Fe3</ci></apply>
<apply><times/><ci>k2</ci><ci>Fe3</ci></apply></apply>
</apply></Valid: d(Fe3)/d(time) = k_1 * Fe3_Arg + k13 * Fe3_NO + k12 *
Fe2_NO * O2 - (k1 * Arg * Fe3 + k2 * Fe3)
==== Component Fe3_Arg =====
<apply xmlns="http://www.w3.org/1998/Math/MathML"><eq/><apply>
<diff/><bvar><ci>time</ci></bvar><ci>Fe3_Arg</ci></apply><apply>
<minus/><apply><times/><ci>k1</ci><ci>Fe3</ci><ci>Arg</ci></apply>
<apply><plus/><apply><times/><ci>k_1</ci></apply>
<ci>Fe3_Arg</ci></apply><apply><times/><ci>k3</ci></apply>
<ci>Fe3_Arg</ci></apply></apply></Valid:
d(Fe3_Arg)/d(time) = k1 * Fe3 * Arg - (k_1 * Fe3_Arg + k3 * Fe3_Arg)
==== Component Fe2 =====
<apply xmlns="http://www.w3.org/1998/Math/MathML"><eq/><apply>
<diff/><bvar><ci>time</ci></bvar><ci>Fe2</ci></apply><apply>
<minus/><apply><plus/><apply><times/><ci>k2</ci><ci>Fe3</ci></apply>
<ci>Fe3_Arg</ci></apply><apply><times/><ci>k_5</ci></apply>
<ci>Fe3_O2_Arg</ci></apply><apply><times/><ci>k4</ci></apply>
<ci>Fe2</ci><ci>Arg</ci></apply></apply><apply><plus/><apply>
<times/><ci>k5</ci><ci>Fe2_Arg</ci><ci>O2</ci></apply><apply>
<times/><ci>k_4</ci><ci>Fe2_Arg</ci></apply></apply></apply>
</apply></Valid: d(Fe2)/d(time) = k3 * Fe3_Arg + k_5 * Fe3_O2_Arg + k4
* Fe2 * Arg - (k5 * Fe2_Arg * O2 + k_4 * Fe2_Arg)
==== Component Fe3_O2_Arg =====
```

Model chen_model_2007 loaded.

Round tripping

- A document such as this

$$\frac{d(\text{Fe3})}{d(\text{time})} = k_1 \cdot \text{Fe3}_{\text{Arg}} + k_{13} \cdot \text{Fe3}_{\text{NO}} + k_{12} \cdot \text{Fe2}_{\text{NO}} \cdot \text{O2} - (k_1 \cdot \text{Arg} \cdot \text{Fe3} + k_2 \cdot \text{Fe3})$$



COR

- Text input language
 - Faster composition of models
 - Easier to take code from Matlab
 - Only supports CellML 1.0
- Modal
- Follows Windows UI idioms



Features in PCEnv from COR

- More convenience facilities in UI
- Graphical renderings
 - Rendered Maths, Units, Connections, Variables



PCEnv

- Multi platform
- Uses the CellML API
- Support for CellML 1.1
- Extensible



Concluding remarks

- Focus on individually reusable components
- Focus on IDE traits
- Opening up development process
 - Bugtracker at <https://tracker.physiomeproject.org/>
 - SVN repository at <https://svn.physiomeproject.org/svn/physiome/>