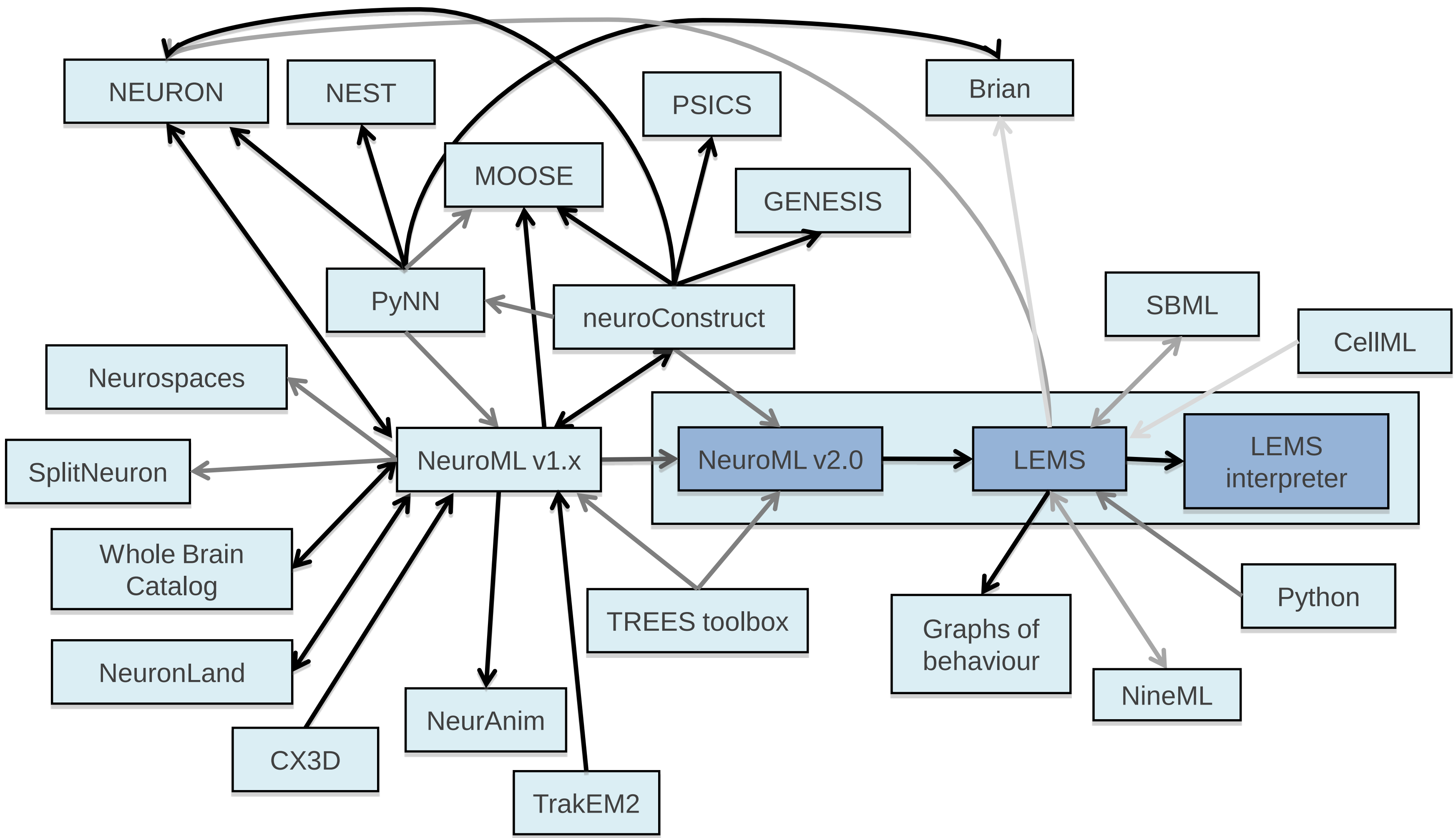
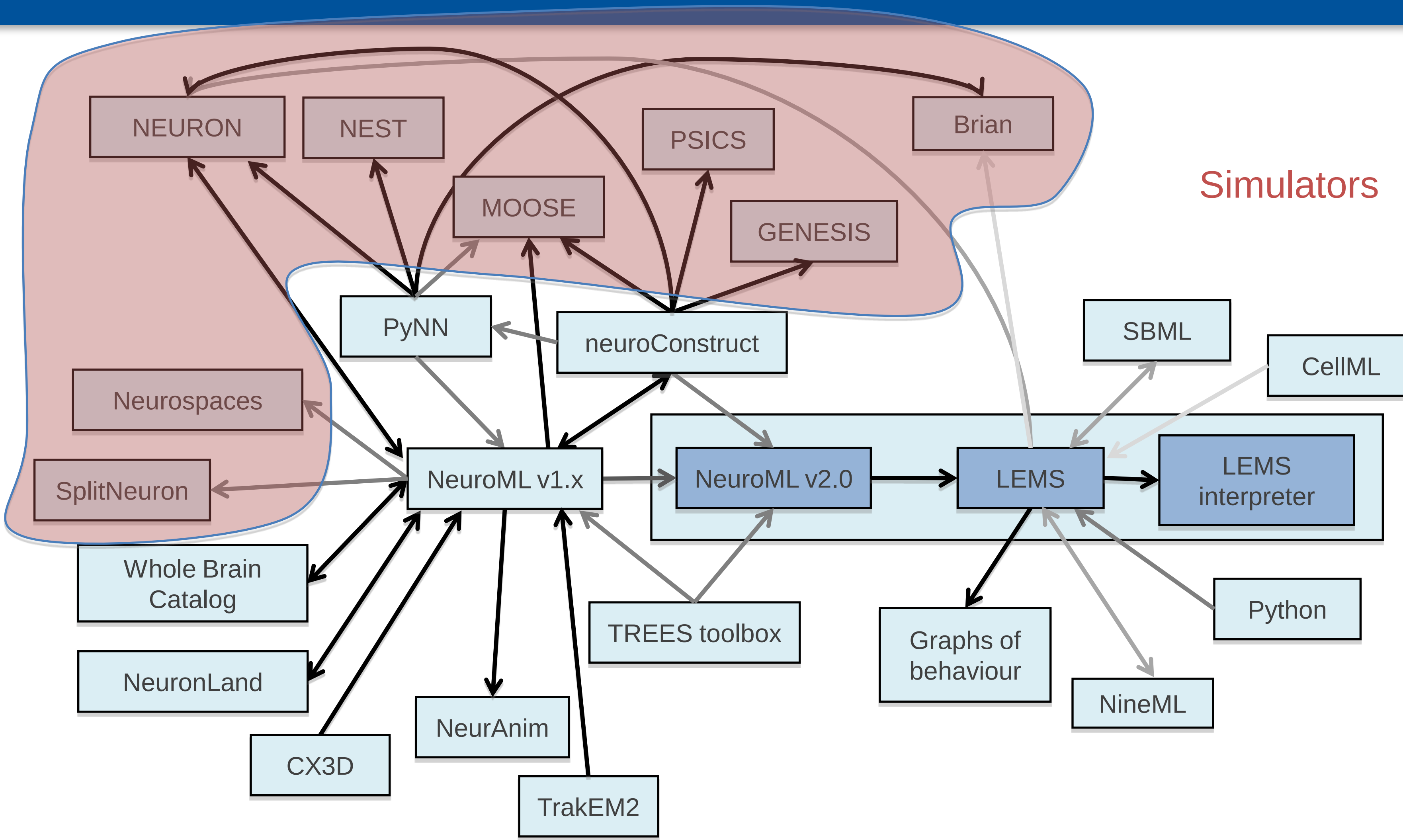
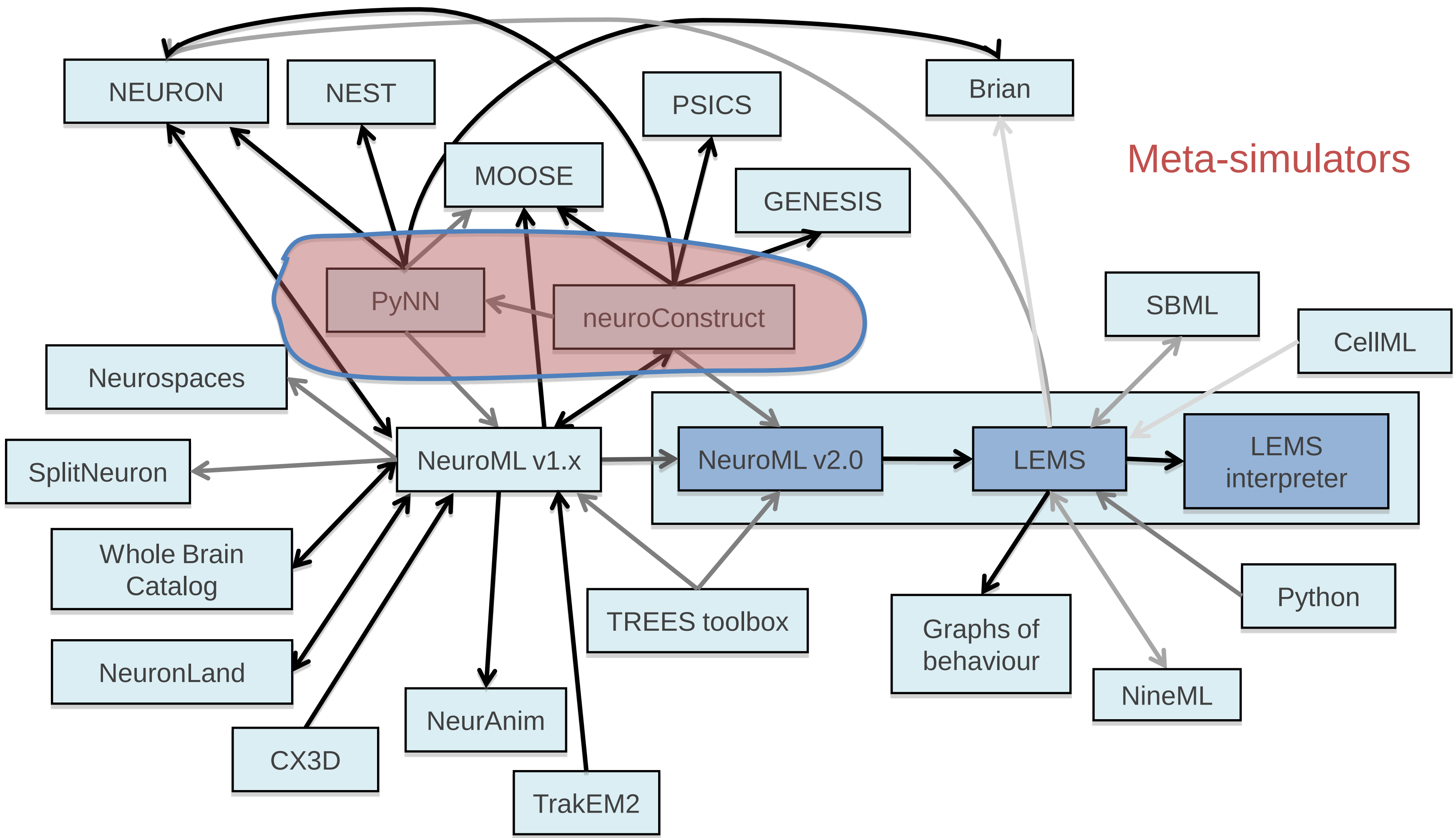
Transforming LEMS/NeuroML v2.0 models to & from other formats

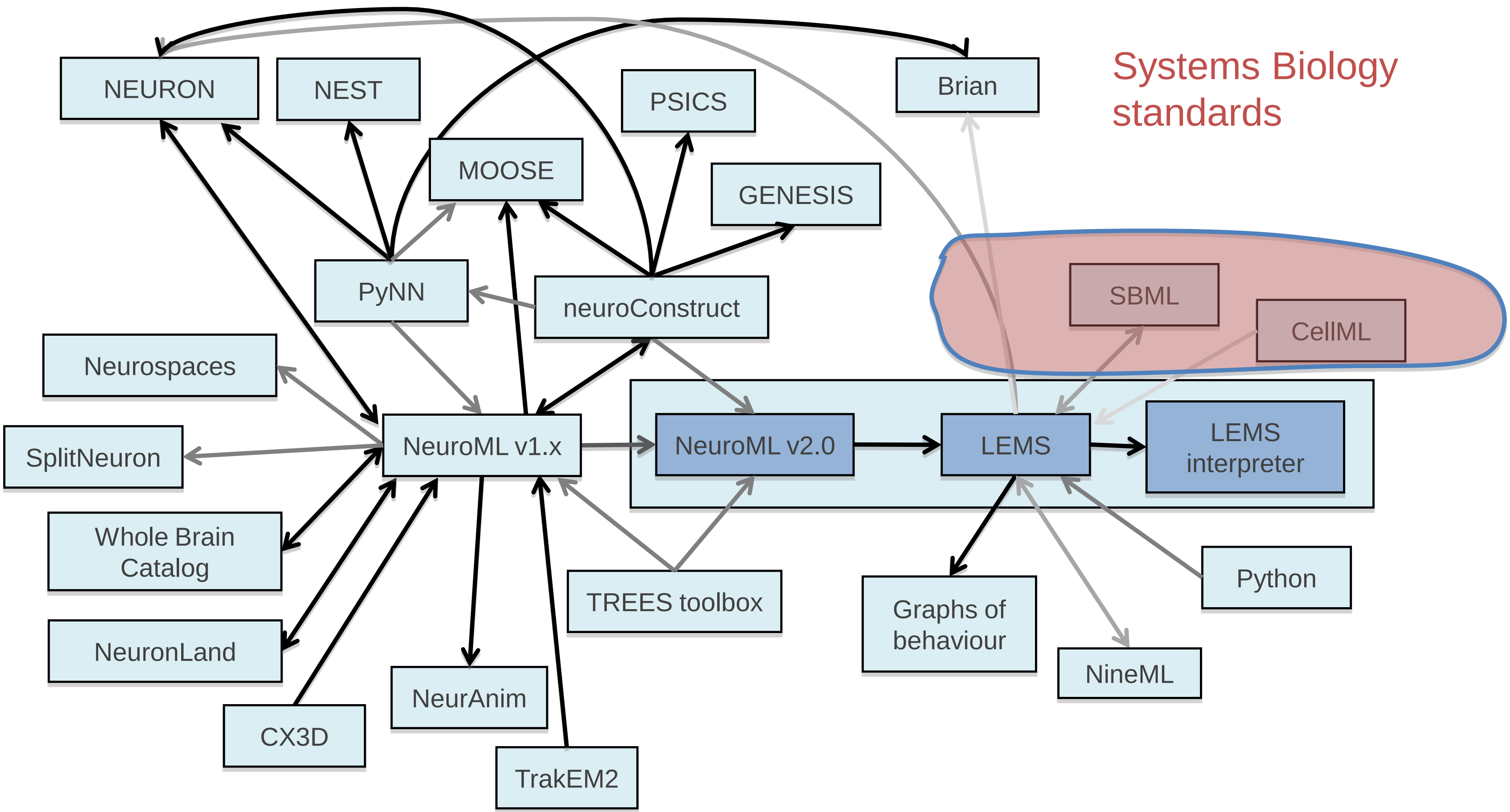
Padraig Gleeson

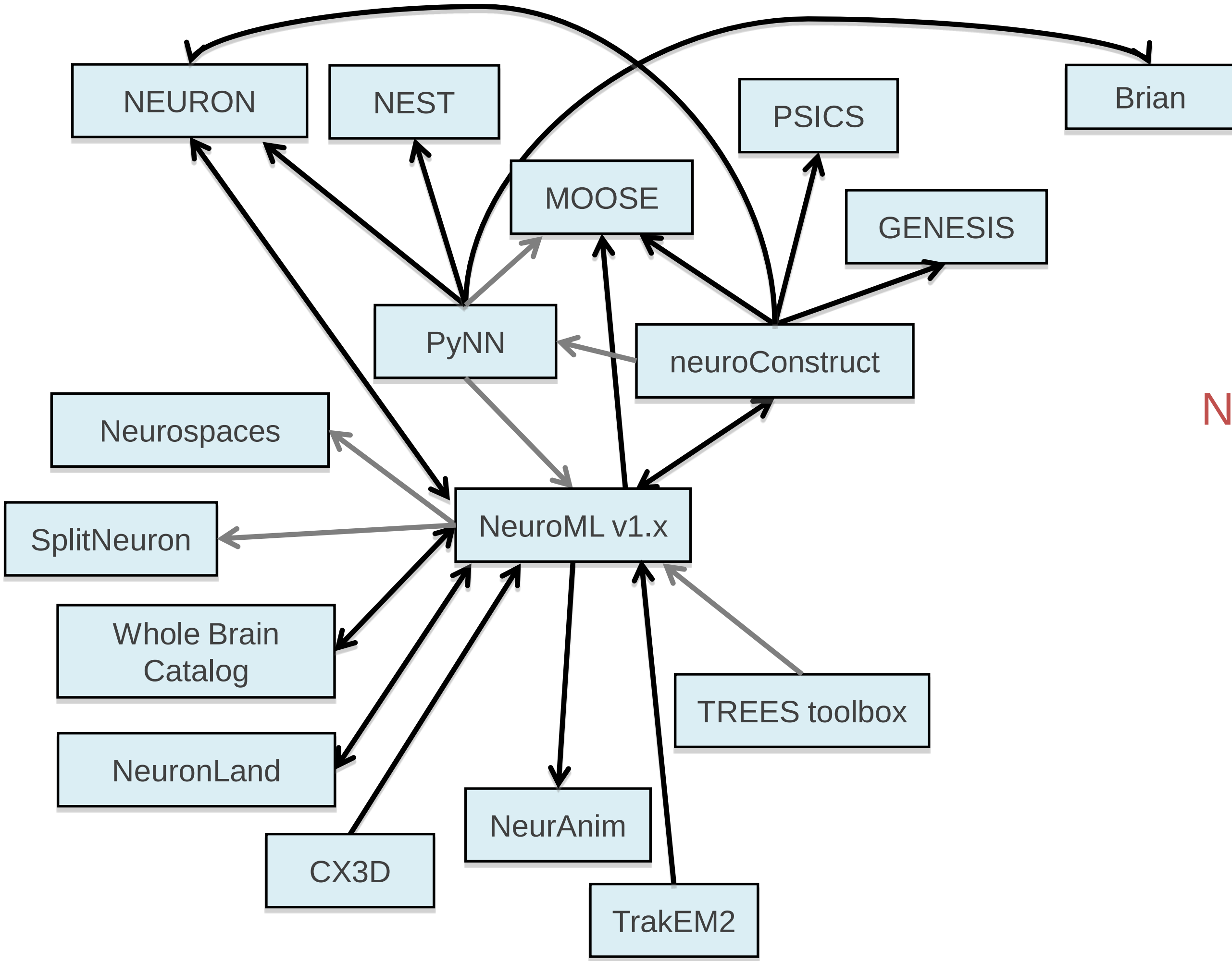
*Lab of **Prof. R. Angus Silver***
Department of Neuroscience, Physiology and Pharmacology
University College London


NEURON
NEST
PSICS
Brian
MOOSE
GENESIS
PyNN
neuroConstruct
SBML
CellML
Neurospaces
SplitNeuron
NeuroML v1.x
NeuroML v2.0
LEMS
LEMS interpreter
Whole Brain Catalog
NeuronLand
CX3D
NeurAnim
TrakEM2
TREES toolbox
Graphs of behaviour
NineML
Python

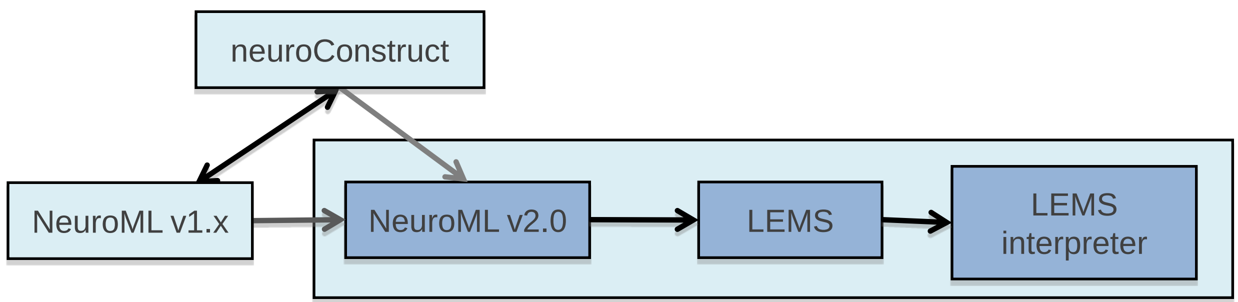

Simulators
NEURON
NEST
PSICS
Brian
MOOSE
GENESIS
PyNN
neuroConstruct
Neurospaces
SplitNeuron
NeuroML v1.x
NeuroML v2.0
LEMS
LEMS interpreter
SBML
CellML
Whole Brain Catalog
NeuronLand
CX3D
NeurAnim
TrakEM2
TREES toolbox
Graphs of behaviour
NineML
Python

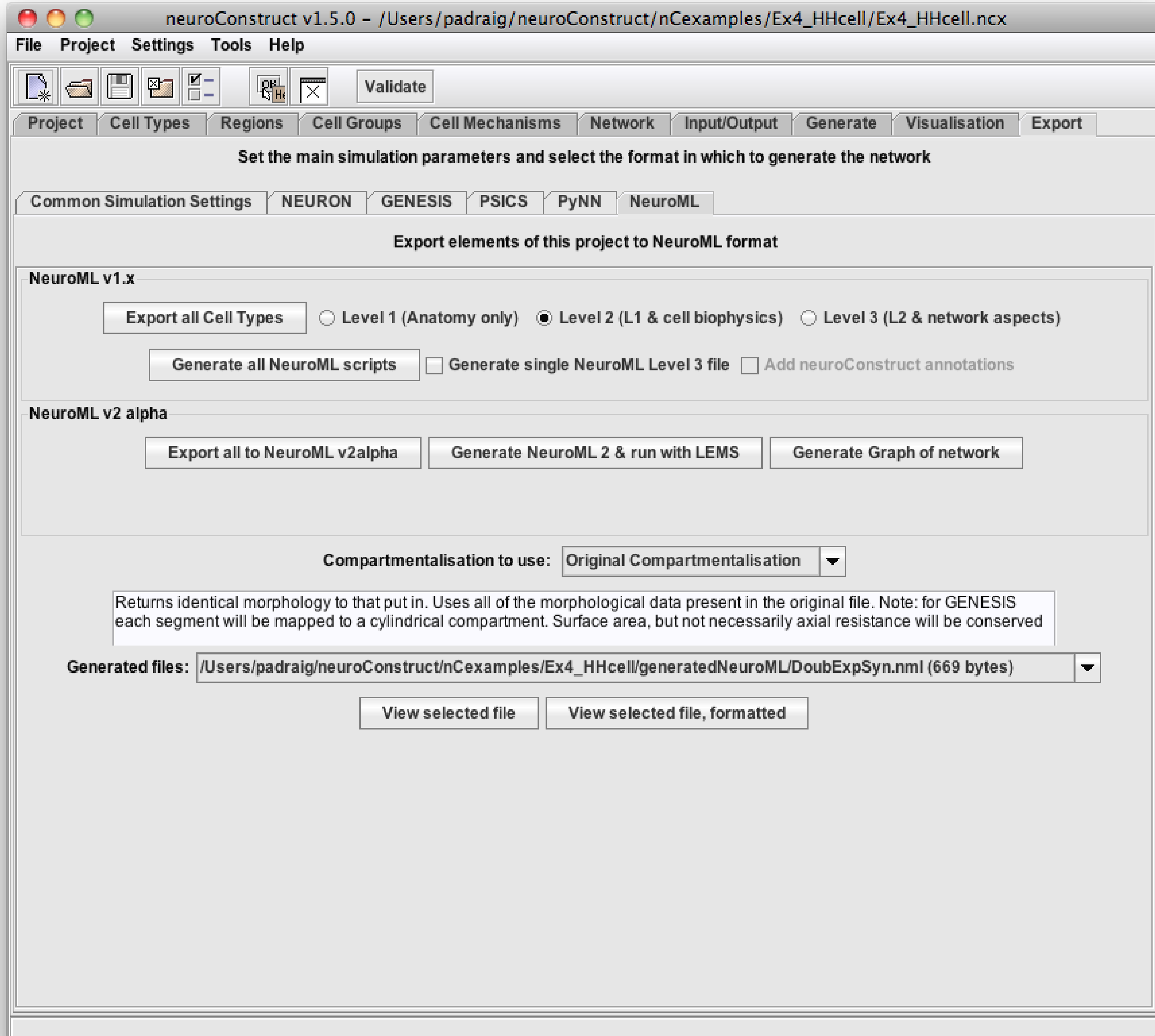

NEURON
NEST
PSICS
Brian
MOOSE
GENESIS
Meta-simulators
PyNN
neuroConstruct
SBML
CellML
Neurospaces
SplitNeuron
NeuroML v1.x
NeuroML v2.0
LEMS
LEMS interpreter
Whole Brain Catalog
NeuronLand
CX3D
NeurAnim
TrakEM2
TREES toolbox
Graphs of behaviour
NineML
Python


NEURON
NEST
PSICS
Brian
Systems Biology standards
MOOSE
GENESIS
PyNN
neuroConstruct
SBML
CellML
Neurospaces
SplitNeuron
NeuroML v1.x
NeuroML v2.0
LEMS
LEMS interpreter
Whole Brain Catalog
NeuronLand
CX3D
NeurAnim
TrakEM2
TREES toolbox
Graphs of behaviour
NineML
Python



NeuroML version 1.x
tool support

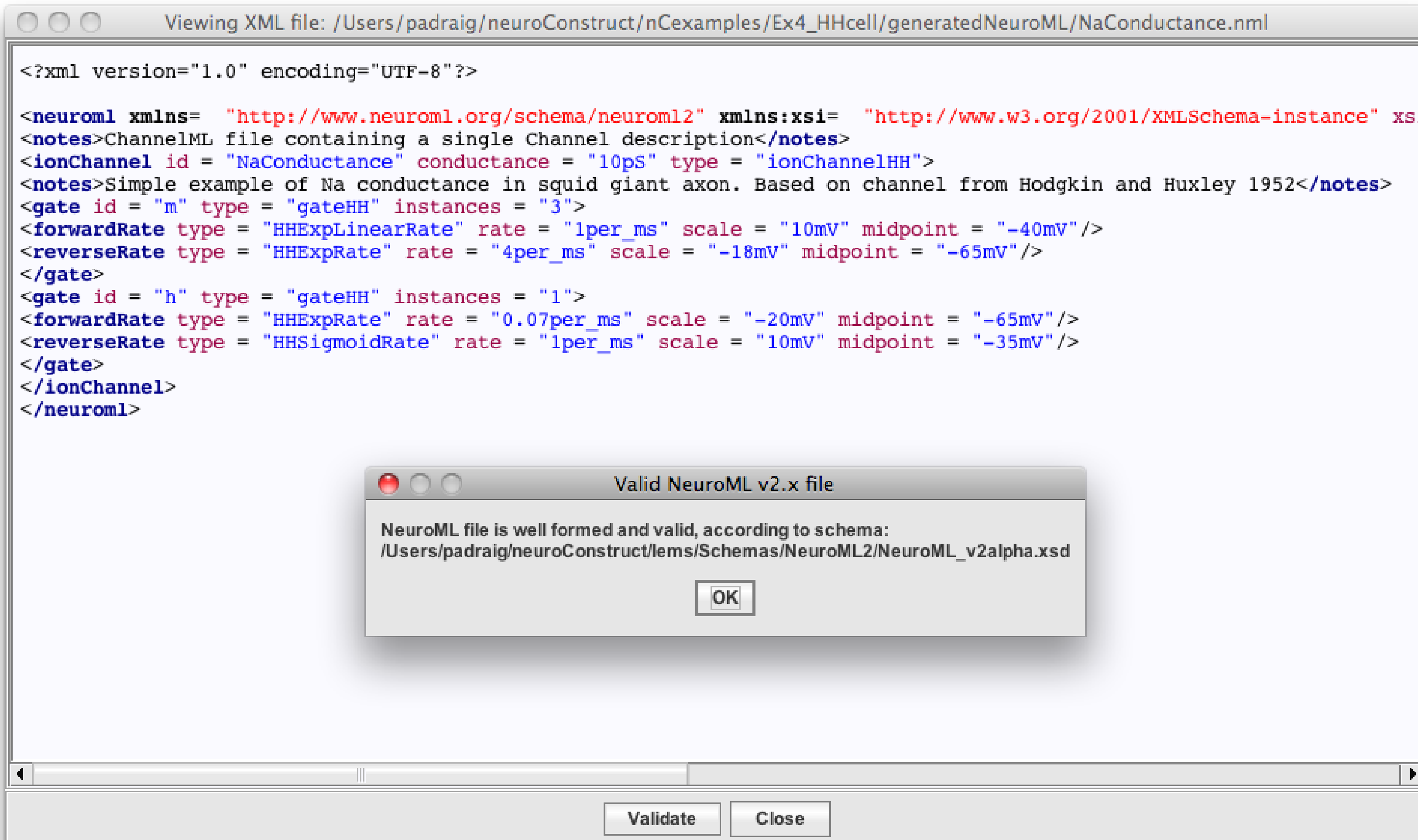

neuroConstruct
NeuroML v1.x
NeuroML v2.0
LEMS
LEMS interpreter

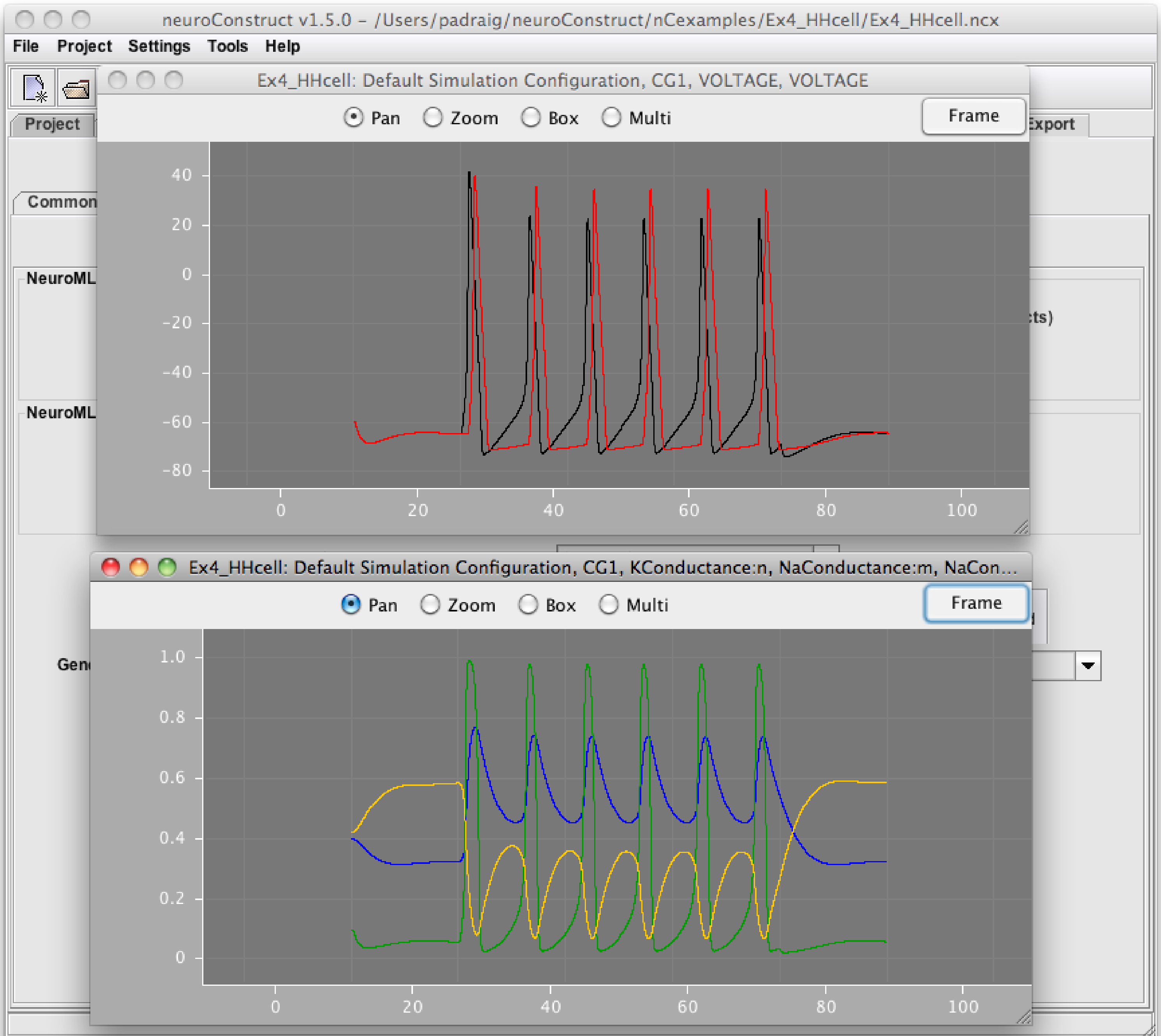

neuroConstruct v1.5.0 – /Users/padraig/neuroConstruct/nCexamples/Ex4_HHcell/Ex4_HHcell.ncx
File
Project
Settings
Tools
Help
Validate
Project
Cell Types
Regions
Cell Groups
Cell Mechanisms
Network
Input/Output
Generate
Visualisation
Export
Set the main simulation parameters and select the format in which to generate the network
Common Simulation Settings
NEURON
GENESIS
PSICS
PyNN
NeuroML
Export elements of this project to NeuroML format
NeuroML v1.x
Export all Cell Types
Level 1 (Anatomy only)
Level 2 (L1 & cell biophysics)
Level 3 (L2 & network aspects)
Generate all NeuroML scripts
Generate single NeuroML Level 3 file
Add neuroConstruct annotations
NeuroML v2 alpha
Export all to NeuroML v2alpha
Generate NeuroML 2 & run with LEMS
Generate Graph of network
Compartmentalisation to use:
Original Compartmentalisation
Returns identical morphology to that put in. Uses all of the morphological data present in the original file. Note: for GENESIS each segment will be mapped to a cylindrical compartment. Surface area, but not necessarily axial resistance will be conserved
Generated files:
/Users/padraig/neuroConstruct/nCexamples/Ex4_HHcell/generatedNeuroML/DoubExpSyn.nml (669 bytes)
View selected file
View selected file, formatted

```xml
<neuroml xmlns= "http://www.neuroml.org/schema/neuroml2" xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation = "http:

    <include href = "NaConductance.nml"/>

    <include href = "LeakConductance.nml"/>

    <include href = "KConductance.nml"/>

    <cell id = "TestCell_ChannelML">

        <notes>A Simple cell with HH channels specified in ChannelML</notes>

        <morphology id = "morphology_TestCell_ChannelML">

            &lt;segment id = "0" name = "Soma"&gt;
                <proximal x = "0.0" y = "0.0" z = "0.0" diameter = "16.0"/>
                <distal x = "0.0" y = "0.0" z = "0.0" diameter = "16.0"/>
            &lt;/segment&gt;

            <segmentGroup id = "Soma">
                <member segment = "0"/>
            </segmentGroup>

            <segmentGroup id = "all">
                <include segmentGroup = "Soma"/>
            </segmentGroup>

            <segmentGroup id = "soma_group">
                <include segmentGroup = "Soma"/>
            </segmentGroup>


        </morphology>

                <!-- Adding the biophysical parameters -->

        <biophysicalProperties id = "biophys">

            <membraneProperties>

                <channelDensity id = "NaConductance_all" ionChannel = "NaConductance" condDensity = "120.0 mS_per_cm2" erev = "50.0 mV"/>

                <channelDensity id = "LeakConductance_all" ionChannel = "LeakConductance" condDensity = "0.3 mS_per_cm2" erev = "-54.3 mV"/>

                <channelDensity id = "KConductance_all" ionChannel = "KConductance" condDensity = "36.0 mS_per_cm2" erev = "-77.0 mV"/>

                <spikeThresh value = "0 mV"/>

                <specificCapacitance value = "1.0 uF_per_cm2"/>

                <initMembPotential value = "-60.0 mV"/>

            </membraneProperties>
```

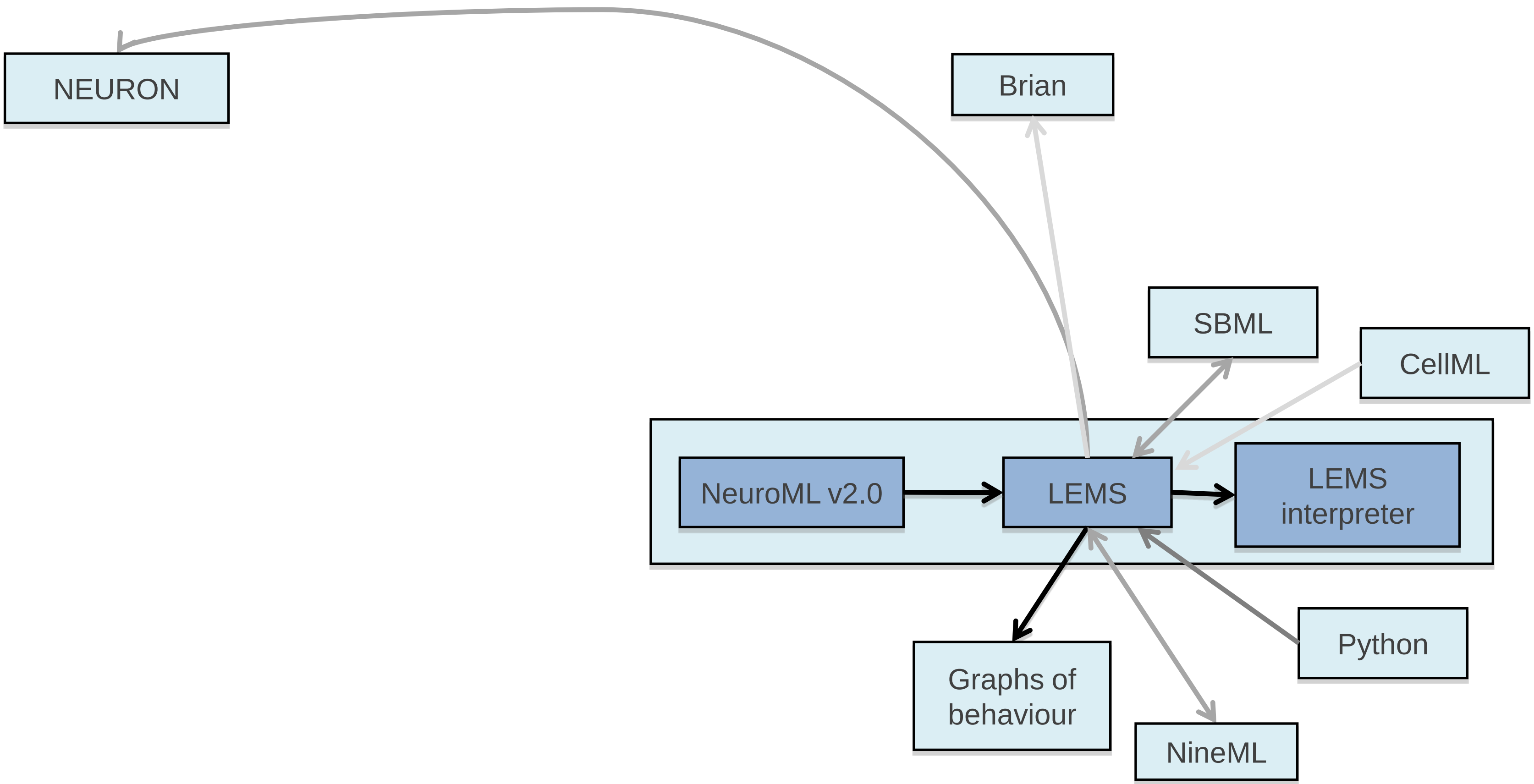
lems/ChannelMLConvert/ChannelML2NeuroML2.xsl

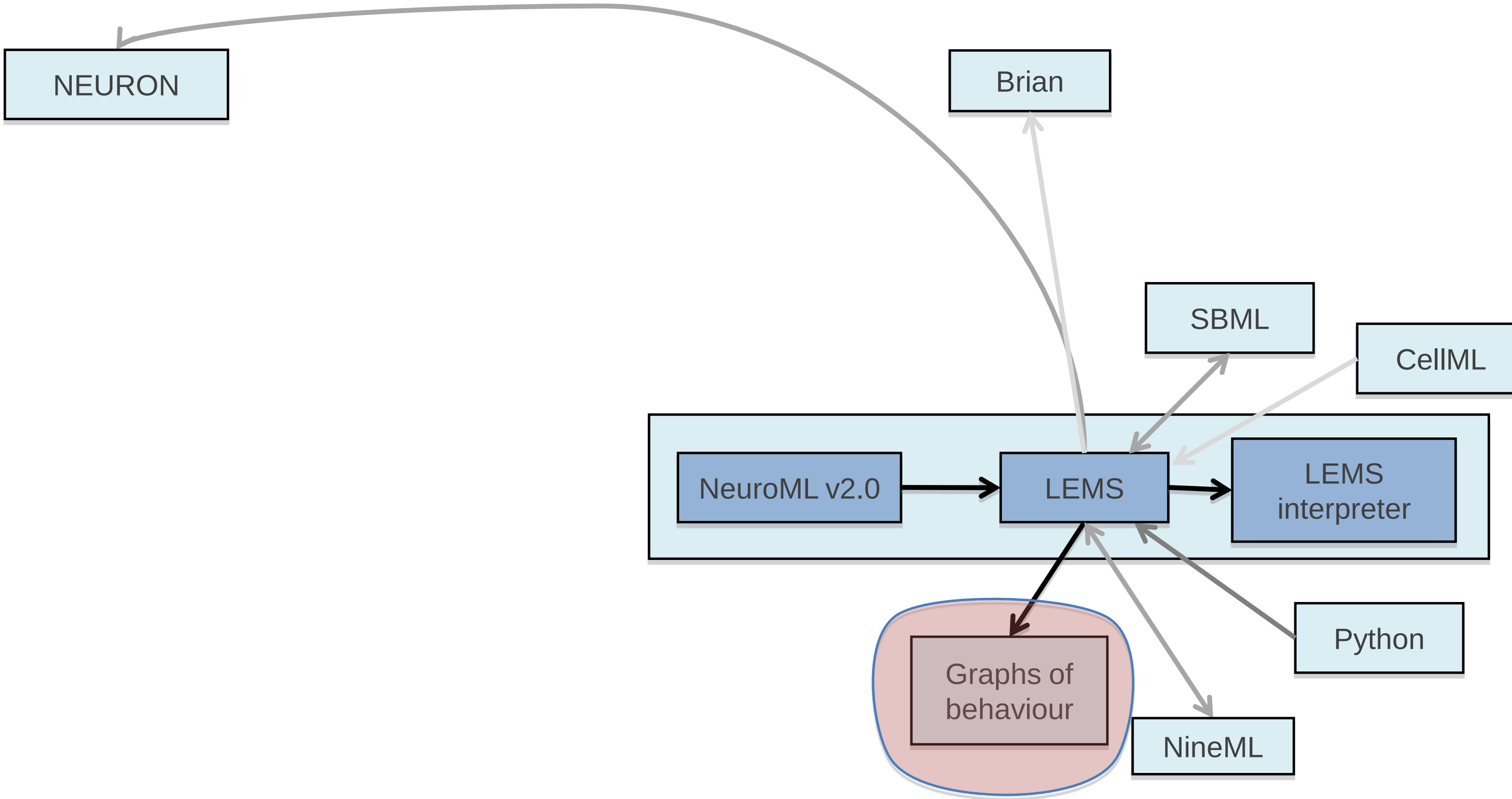
enables mapping ChannelML -> NeuroML 2 for (simple) channels & synapses





neuroConstruct v1.5.0 – /Users/padraig/neuroConstruct/nCexamples/Ex4_HHcell/Ex4_HHcell.ncx
File Project Settings Tools Help
Ex4_HHcell: Default Simulation Configuration, CG1, VOLTAGE, VOLTAGE
Pan Zoom Box Multi
Frame
Ex4_HHcell: Default Simulation Configuration, CG1, KConductance:n, NaConductance:m, NaCon...
Pan Zoom Box Multi
Frame


NEURON
Brian
SBML
CellML
NeuroML v2.0
LEMS
LEMS interpreter
Python
Graphs of behaviour
NineML


NEURON
Brian
SBML
CellML
NeuroML v2.0
LEMS
LEMS interpreter
Graphs of behaviour
Python
NineML

Adaptive Exponential Integrate & Fire neuron

- Brette & Gernstner 2005

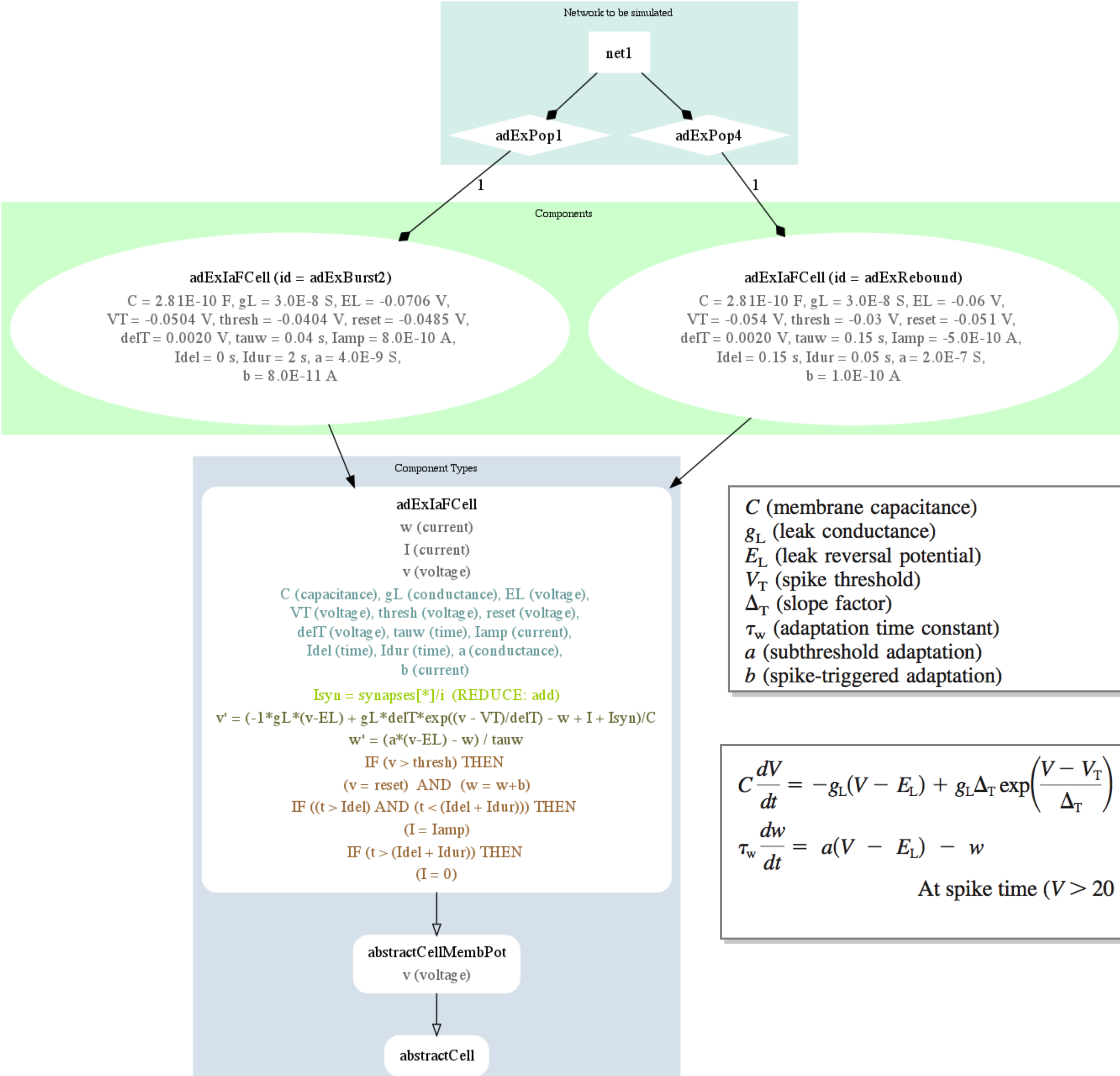
$$C\frac{dV}{dt} = -g_L(V - E_L) + g_L\Delta_T \exp\left(\frac{V - V_T}{\Delta_T}\right) - g_e(t)(V - E_e) - g_i(t)(V - E_i) - w$$

$$\tau_w \frac{dw}{dt} = a(V - E_L) - w$$

$$\text{At spike time } (V > 20 \text{ mV}): \begin{array}{l} V \rightarrow EL \\ w \rightarrow w + b \end{array}$$

Example network in NeuroML 2.0

```
<adExIaFCell id="adExBurst2"     C="281pF" gL="30nS" EL="-70.6mV" reset="-48.5mV" VT = "-50.4mV"
             thresh = "-40.4mV" delT="2mV" tauw="40ms"  a ="4nS"   b = "0.08nA"
             Iamp="0.8nA"  Idel="0ms"   Idur="2000ms"/>

<adExIaFCell id="adExRebound"    C="281pF" gL="30nS" EL="-60mV"   reset="-51mV"   VT = "-54mV"
             thresh = "-30mV"   delT="2mV" tauw="150ms" a ="200nS" b = "0.1nA"
             Iamp="-0.5nA" Idel="150ms" Idur="50ms"/>


<network id="net1">
    <population id="adExPop1" component="adExBurst2" size="1"/>
    <population id="adExPop4" component="adExRebound" size="1"/>
</network>
```



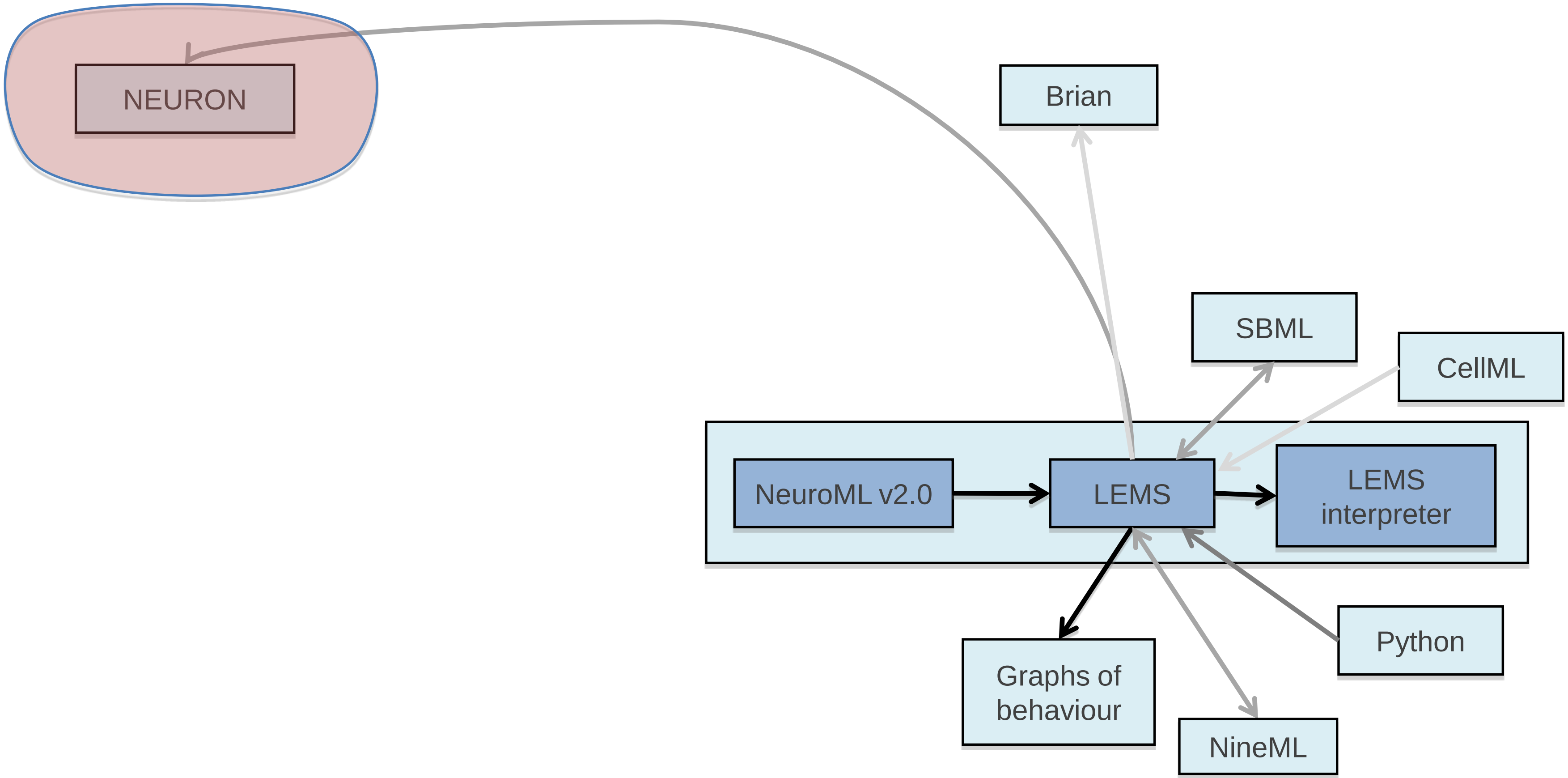
C (membrane capacitance)	281 pF
g_L (leak conductance)	30 nS
E_L (leak reversal potential)	-70.6 mV
V_T (spike threshold)	-50.4 mV
Δ_T (slope factor)	2 mV
τ_w (adaptation time constant)	144 ms
a (subthreshold adaptation)	4 nS
b (spike-triggered adaptation)	0.0805 nA

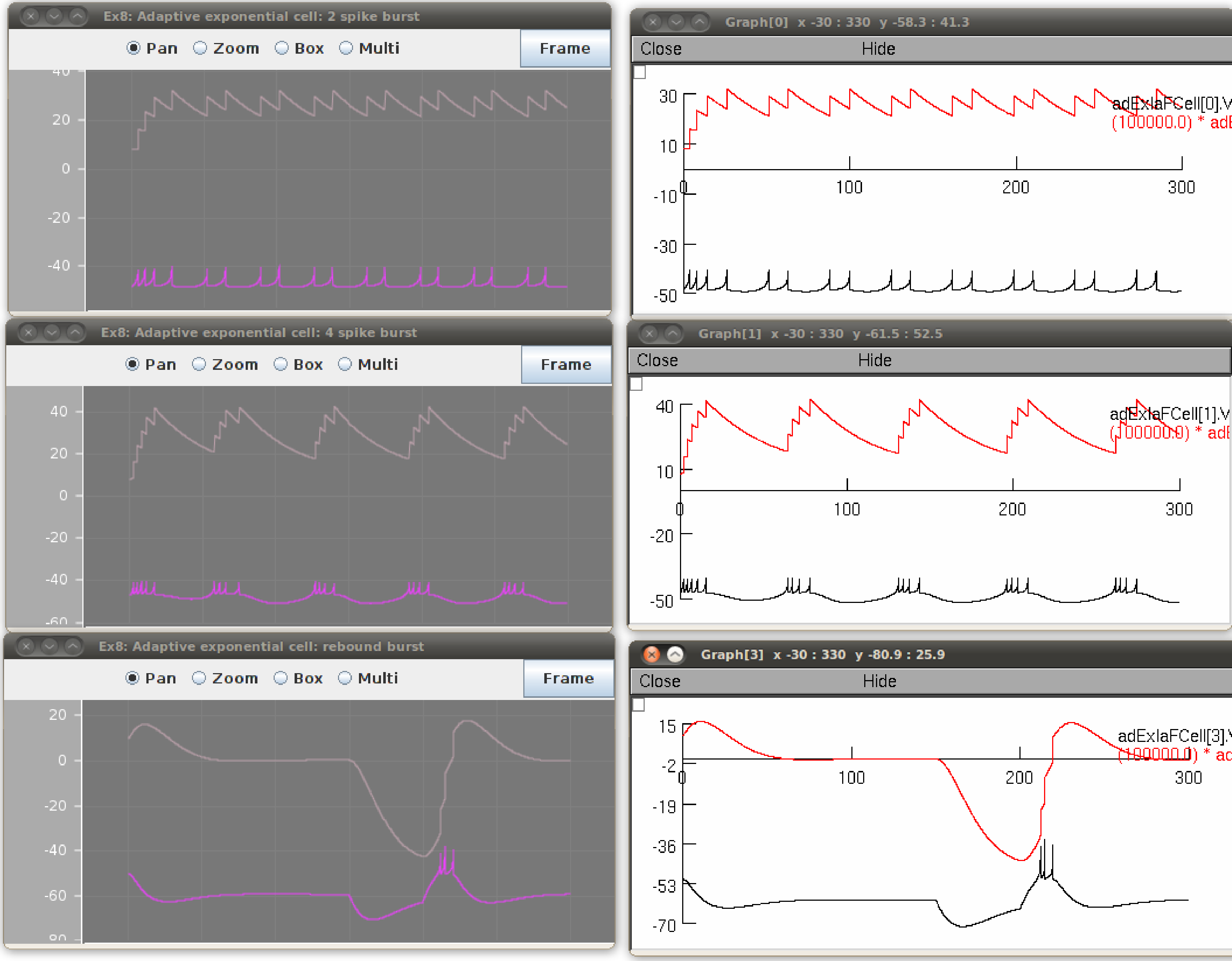
$$C\frac{dV}{dt} = -g_L(V - E_L) + g_L\Delta_T \exp\left(\frac{V - V_T}{\Delta_T}\right) - g_e(t)(V - E_e) - g_i(t)(V - E_i) - w$$

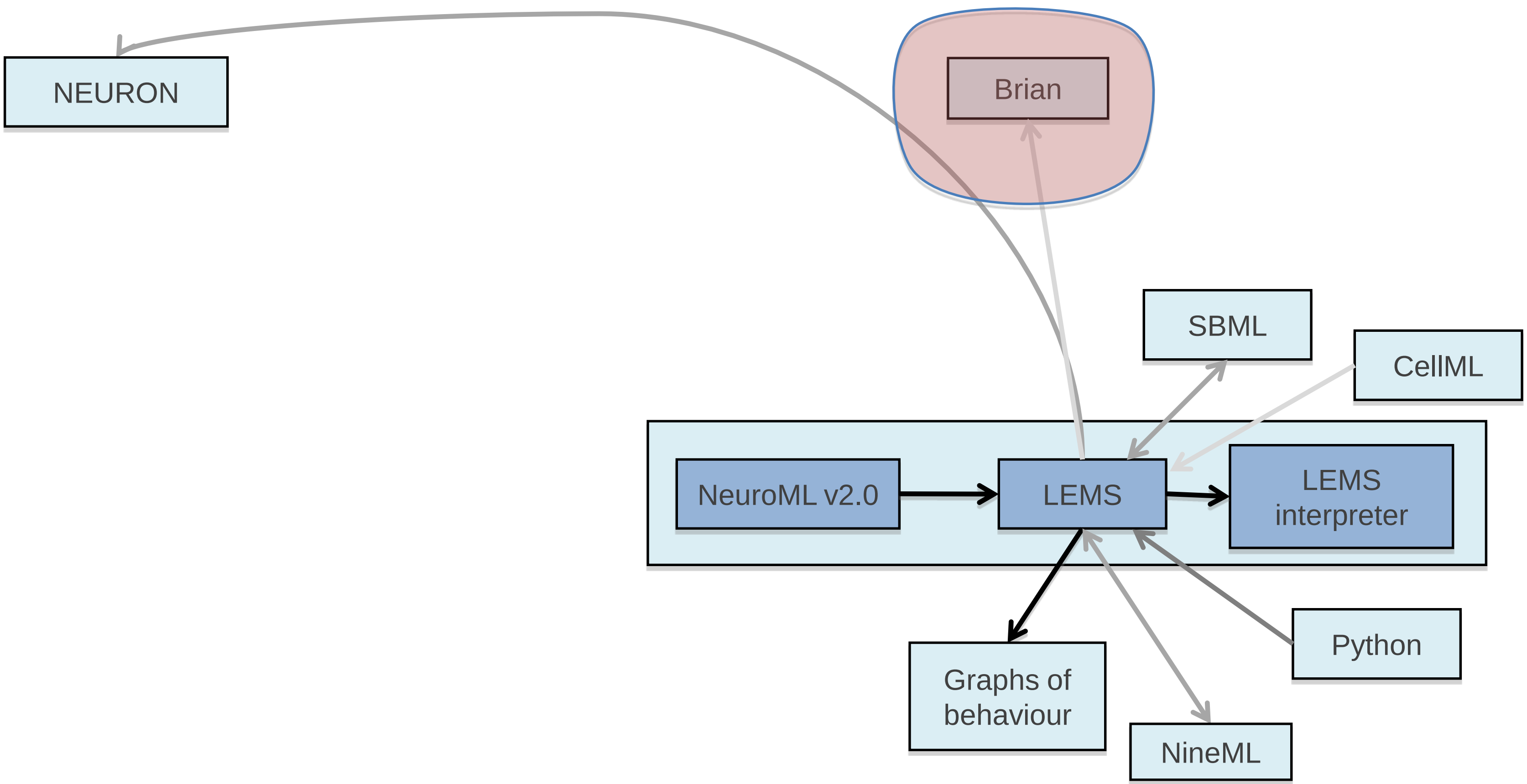
$$\tau_w \frac{dw}{dt} = a(V - E_L) - w$$

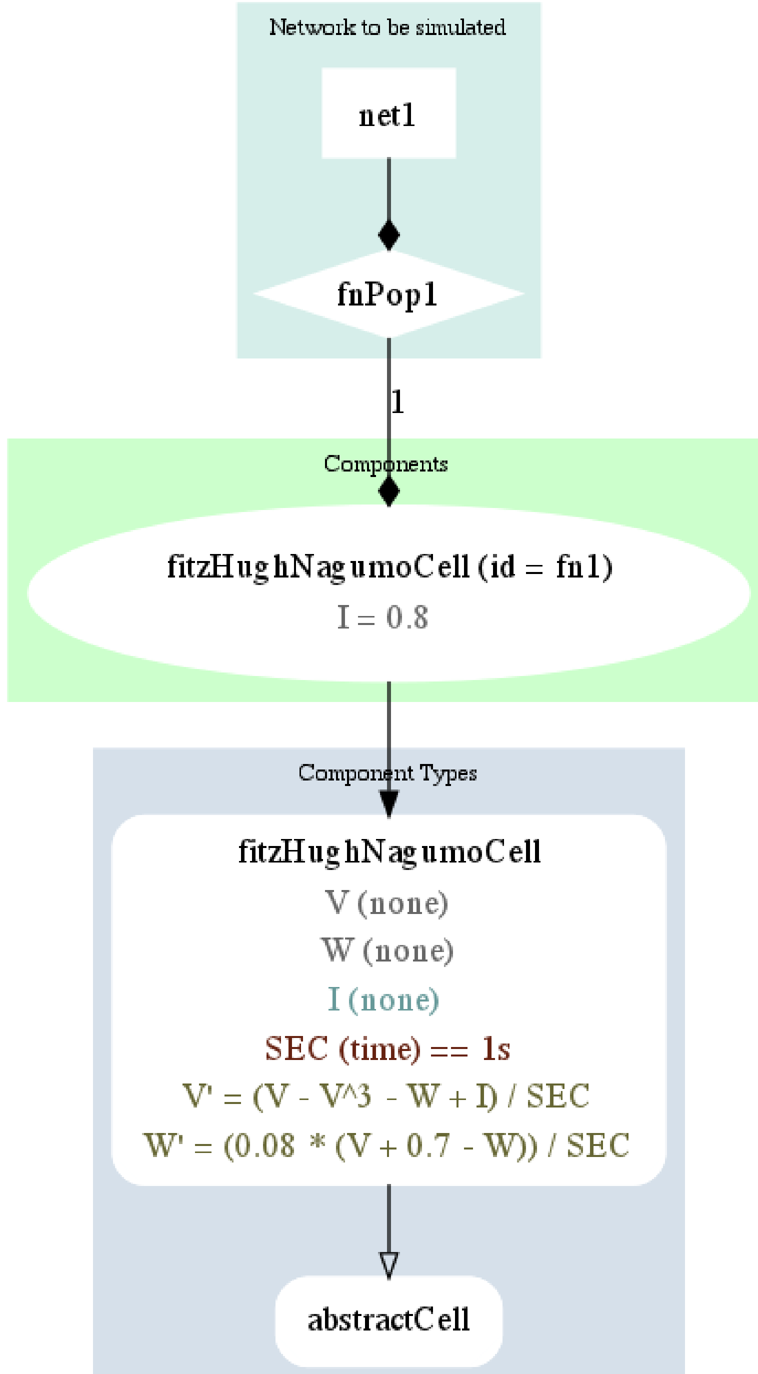
At spike time ($V > 20$ mV): $V \rightarrow EL$

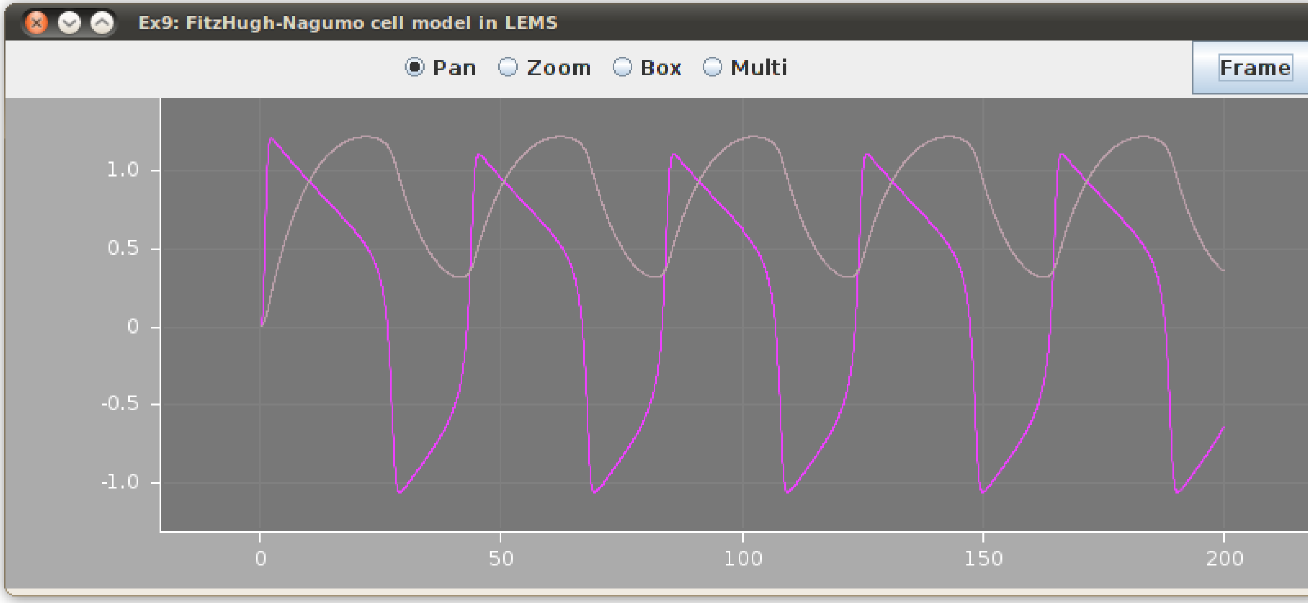
$w \rightarrow w + b$


NEURON
Brian
SBML
CellML
NeuroML v2.0
LEMS
LEMS interpreter
Python
Graphs of behaviour
NineML


Ex8: Adaptive exponential cell: 2 spike burst
Pan
Zoom
Box
Multi
Frame
20
0
-20
-40
Graph[0] x -30 : 330 y -58.3 : 41.3
Close
Hide
30
10
-10
-30
-50
0
100
200
300
adExIaFCell[0].V
(100000.0) * adE
Ex8: Adaptive exponential cell: 4 spike burst
Pan
Zoom
Box
Multi
Frame
40
20
0
-20
-40
Graph[1] x -30 : 330 y -61.5 : 52.5
Close
Hide
40
10
-20
-50
0
100
200
300
adExIaFCell[1].V
(100000.0) * adE
Ex8: Adaptive exponential cell: rebound burst
Pan
Zoom
Box
Multi
Frame
20
0
-20
-40
-60
Graph[3] x -30 : 330 y -80.9 : 25.9
Close
Hide
15
-2
-19
-36
-53
-70
0
100
200
300
adExIaFCell[3].V
(100000.0) * adE


NEURON
Brian
SBML
CellML
NeuroML v2.0
LEMS
LEMS interpreter
Graphs of behaviour
Python
NineML


Network to be simulated
net1
fnPop1
1
Components
fitzHughNagumoCell (id = fn1)
I = 0.8
Component Types
fitzHughNagumoCell
V (none)
W (none)
I (none)
SEC (time) == 1s
V' = (V - V^3 - W + I) / SEC
W' = (0.08 * (V + 0.7 - W)) / SEC
abstractCell


Ex9: FitzHugh-Nagumo cell model in LEMS
Pan
Zoom
Box
Multi
Frame
1.0
0.5
0
-0.5
-1.0
0
50
100
150
200

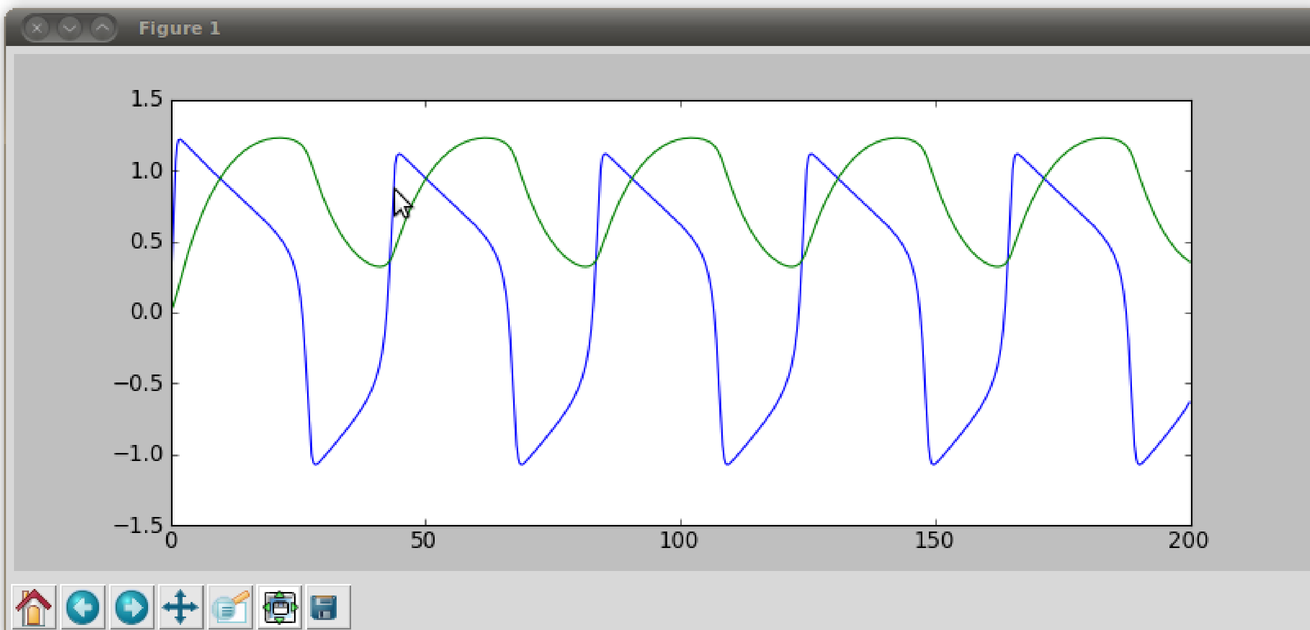
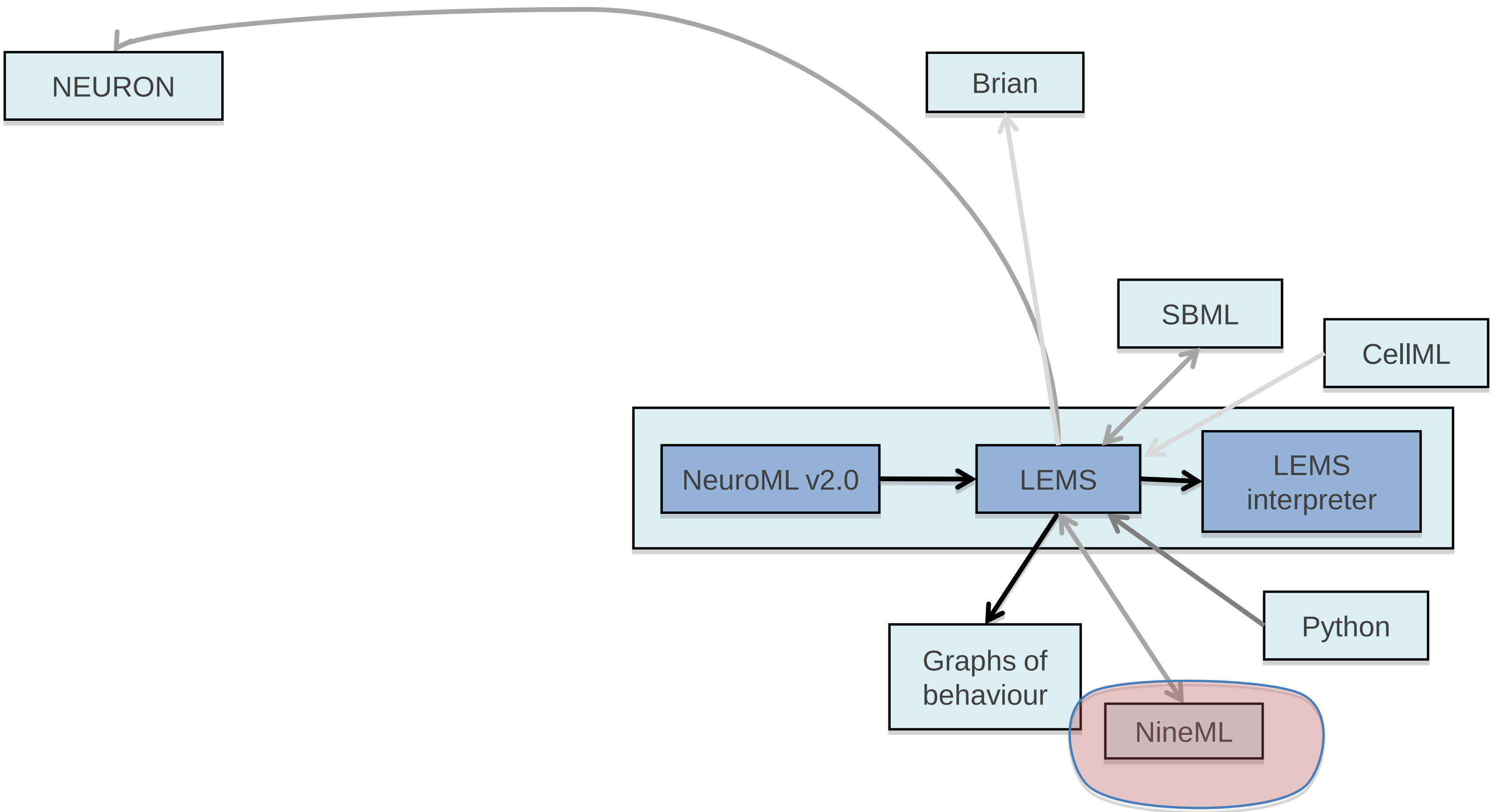


Figure 1
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
50
100
150
200

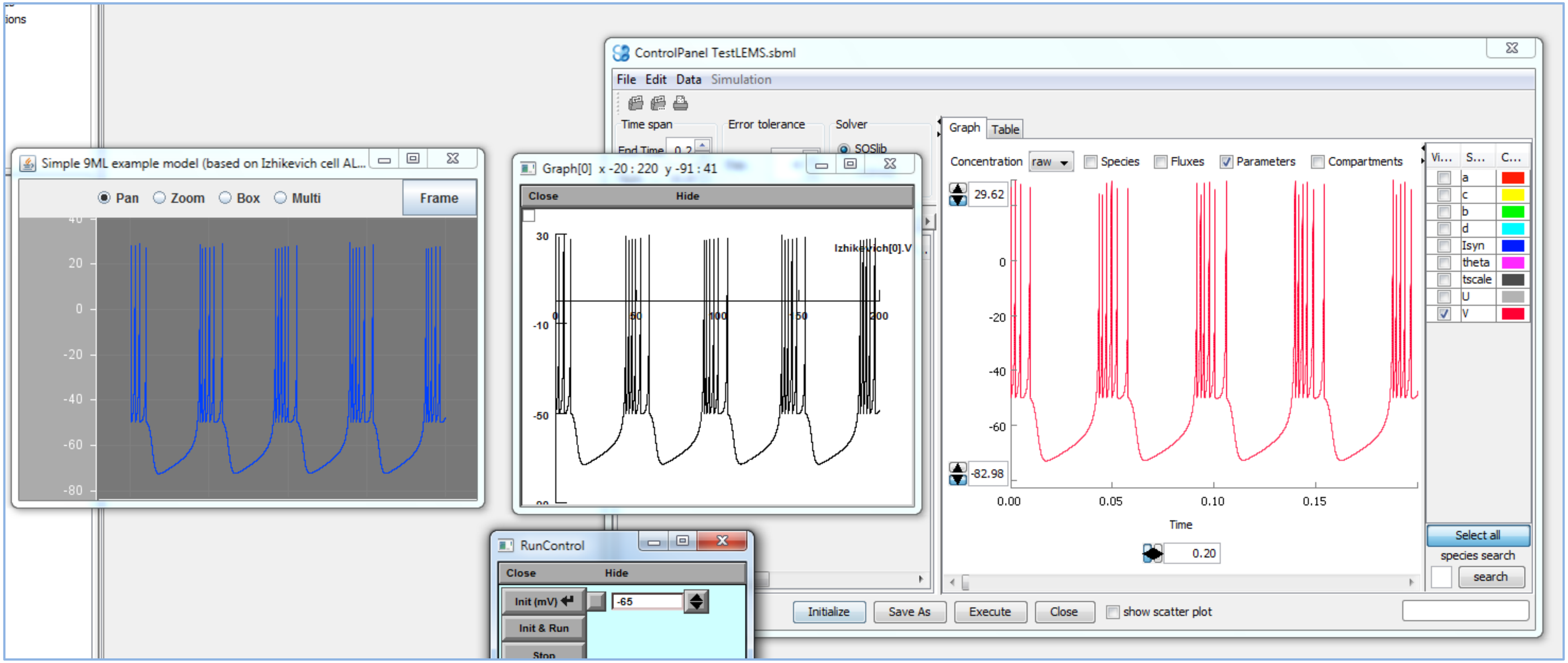

NEURON
Brian
SBML
CellML
NeuroML v2.0
LEMS
LEMS interpreter
Graphs of behaviour
Python
NineML

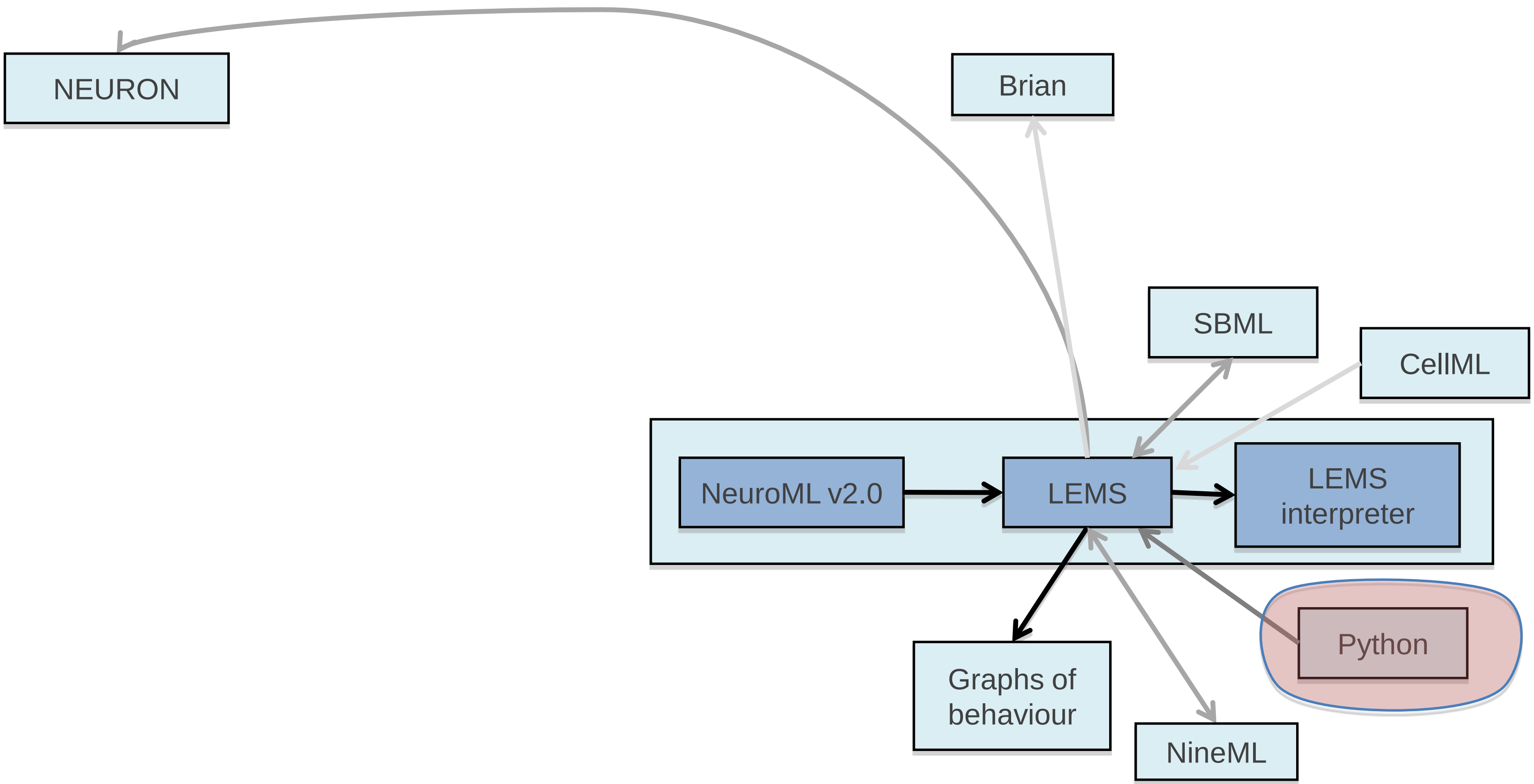
NineML

- NineML (Network Interchange format for NEuroscience) is being developed as part of the INCF Multiscale Modelling Program
- Introduced by Andrew Davison
- Language for describing large scale models of spiking neurons

- LEMS can export simple (e.g. Izhikevich cell) models as NineML
- The Python API for LEMS can be used to import NineML via libnineml

9ML example exported to 3 formats: LEMS, NEURON & SBML

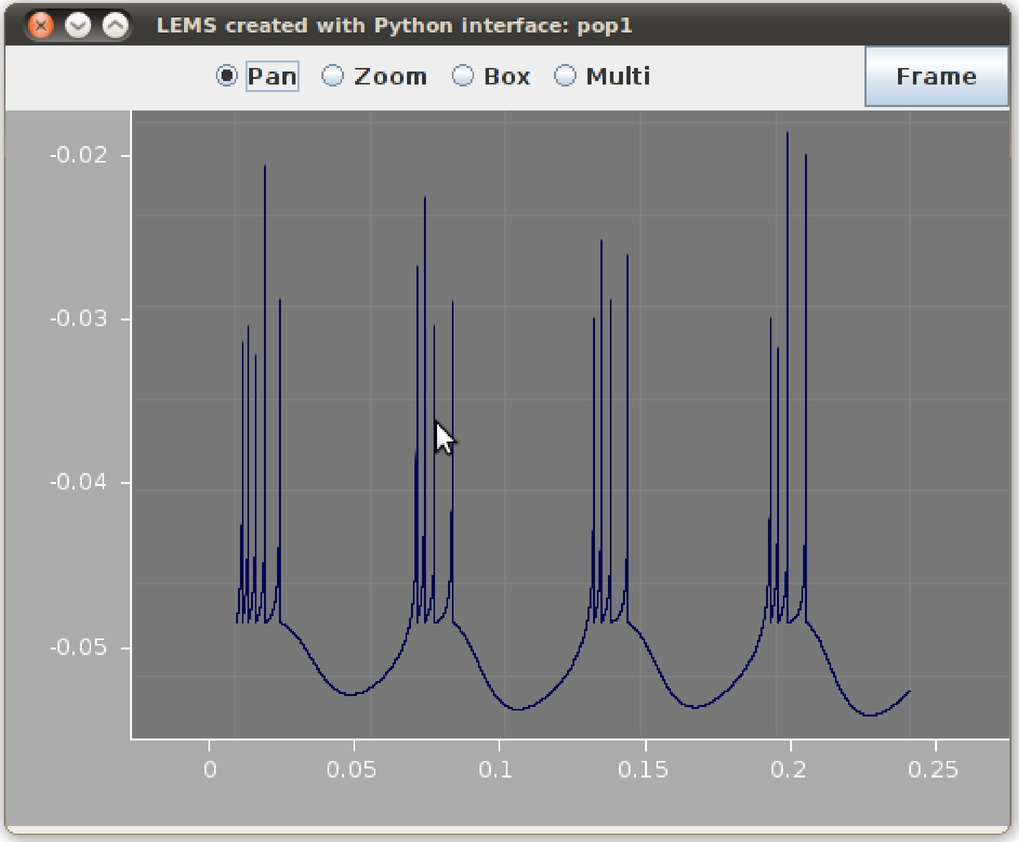



NEURON
Brian
SBML
CellML
NeuroML v2.0
LEMS
LEMS interpreter
Python
Graphs of behaviour
NineML

Python API for LEMS

```
print "Building simulation using Python API for LEMS"

lems_model = lems.Lems()

comp = lems.Component(NML2StdCompTypes.adExIaFCell, "burster", C="281pF", \
                 gL="30nS", EL="-70.6mV", reset="-47.2mV", VT = "-50.4mV", \
                 thresh = "-20.4mV", delT="2mV", tauw="40ms",  \
                 a ="4nS",   b = "0.08nA", Iamp="0.8nA",  \
                 Idel="0ms",   Idur="2000ms")

lems_model.add_component(comp)

net = lems.Network("Network1")

net.add_population(lems.Population("pop1", comp.id, 1))

lems_model.add_network(net)

lems_model.gen_sim_with_default_plots(net.id, 250, 0.01)

lems_model.write_lems_and_run("AdEx.xml")
```

Python API for LEMS



Van der Pol oscillator

Analysis

When x is small, the quadratic term x^2 is negligible and the system becomes a linear differential equation with a negative damping $-\epsilon\dot{x}$. Thus, the fixed point $(x=0, \dot{x}=0)$ is unstable (an unstable focus when $0<\epsilon<2$ and an unstable node, otherwise). On the other hand, when x is large, the term x^2 becomes dominant and the damping becomes positive. Therefore, the dynamics of the system is expected to be restricted in some area around the fixed point. Actually, the van der Pol system (1) satisfies the Liénard's theorem ensuring that there is a stable limit cycle in the phase space.The van der Pol system is therefore a Liénard system.

Using the **Liénard's transformation** $y = x - x^3/3 - \dot{x}/\epsilon$, equation (1) can be rewritten as

$$\dot{x} = \epsilon\left(x - \frac{1}{3}x^3 - y\right) \tag{2}$$

$$\dot{y} = \frac{x}{\epsilon} \tag{3}$$

which can be regarded as a special case of the FitzHugh-Nagumo model (also known as Bonhoeffer-van der Pol model).

Small Damping

When ϵ << 1, it is convenient to rewrite equation (1) as

$$\dot{x} = \epsilon\left(x - \frac{1}{3}x^3\right) - y \tag{4}$$

$$\dot{y} = x \tag{5}$$

where the transformation $y = \epsilon(x - x^3/3) - \dot{x}$ was used. When $\epsilon = 0$, the system preserves the energy and has the solution $x = A\cos(t+\phi)$ and $y = A\sin(t+\phi)$. To obtain the approximated solution for small ϵ, new variables (u, v) which rotate with the unperturbed solution, *i.e.*,

$$u = x\cos t + y\sin t$$
$$v = -x\sin t + y\cos t$$

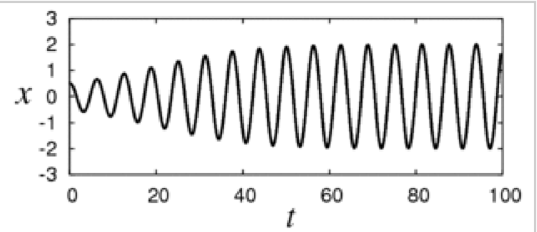
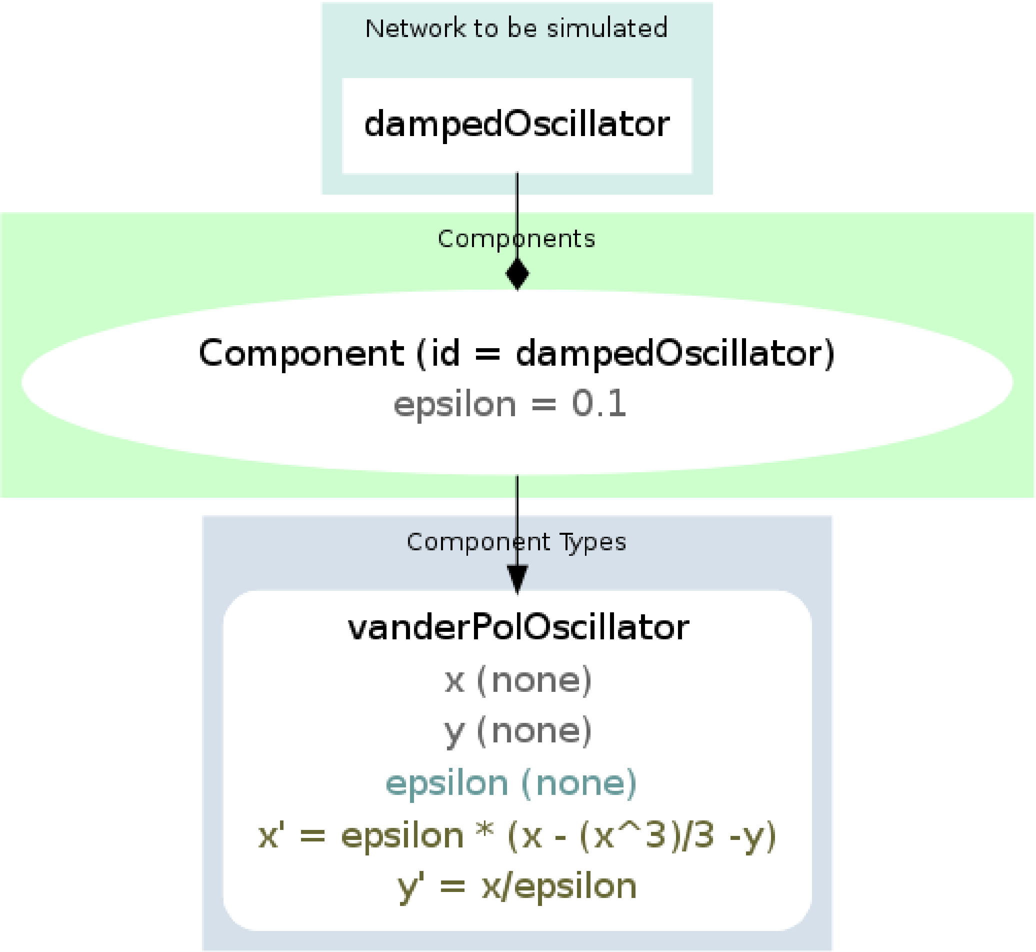


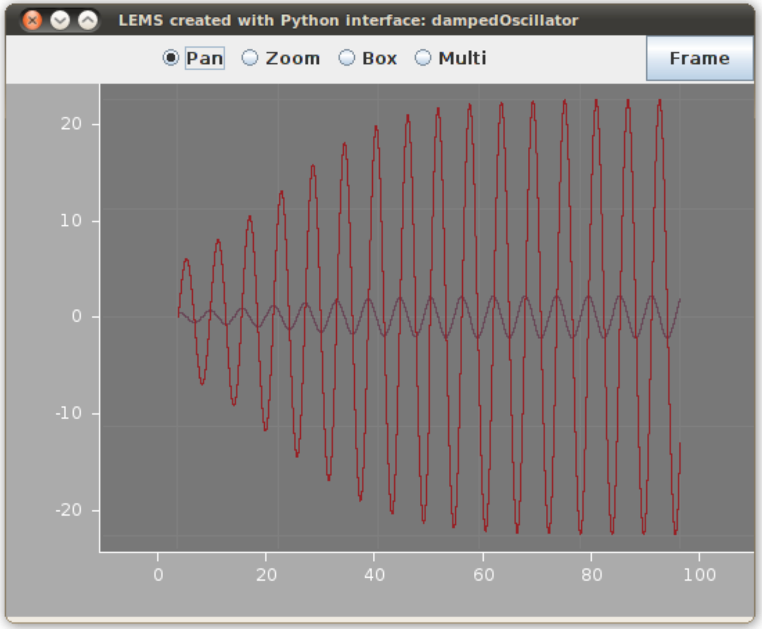
Figure 2: Change in x over time for $\epsilon = 0.1$ with $x(0) = 0.5$ and $y(0) = 0$.

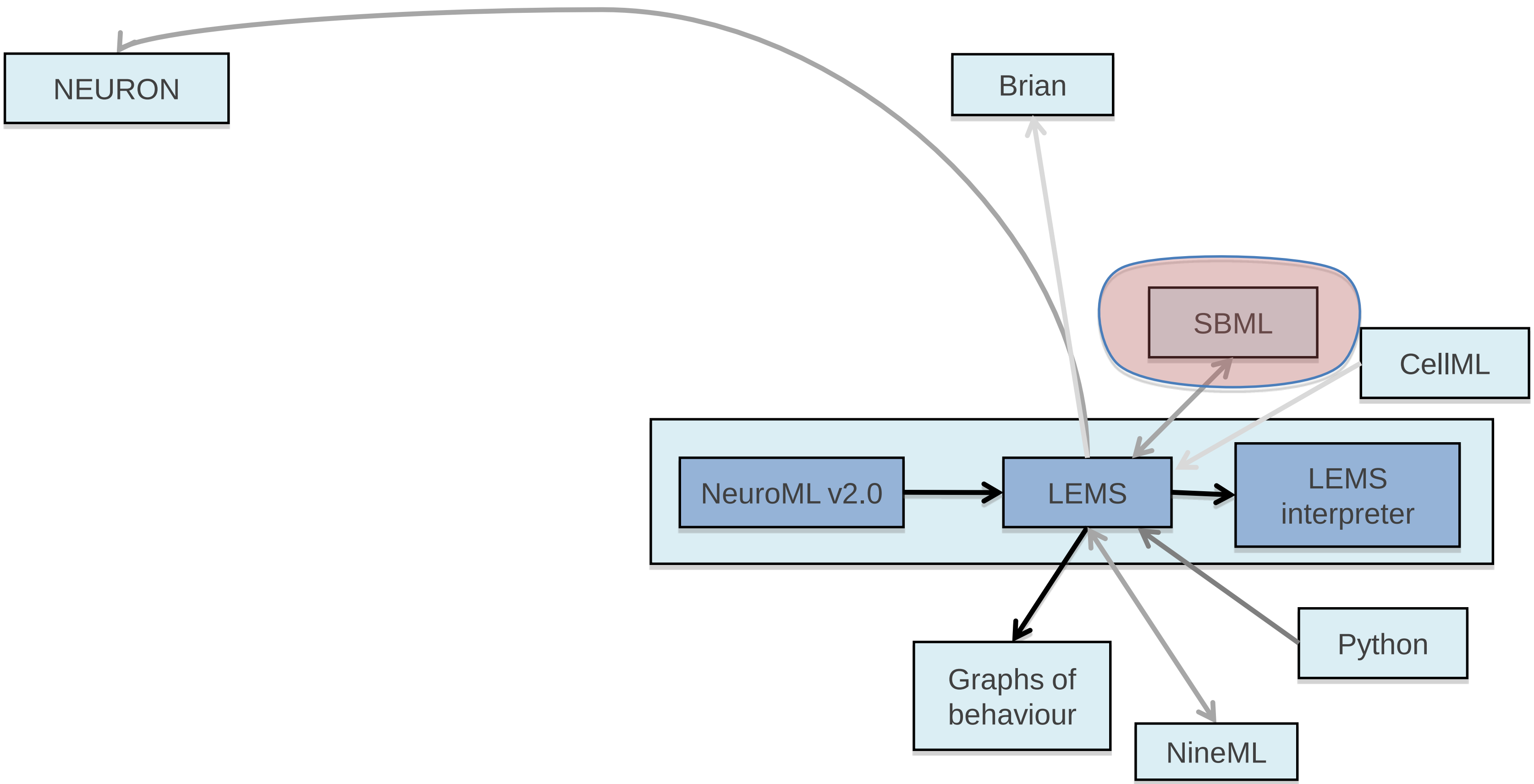
Van der Pol oscillator

```
print "Building simulation using Python API for LEMS"

lems_model = lems.Lems(include_neuroml2_types=False)

# Define ComponentType based on http://www.scholarpedia.org/article/Van_der_Pol_oscillator
comp_type = lems.ComponentType("vanderPolOscillator")

comp_type.parameters.append(lems.Parameter("epsilon"))

comp_type.behaviors[0].add_state_variable(lems.StateVariable("x"))
comp_type.behaviors[0].add_state_variable(lems.StateVariable("y"))

init = lems.OnStart()
init.state_assignments.append(lems.StateAssignment("x", "0.5"))
comp_type.behaviors[0].on_starts.append(init)

comp_type.behaviors[0].time_derivatives.append(lems.TimeDerivative("x", "epsilon * (x - (x^3)/3 -y)"))
comp_type.behaviors[0].time_derivatives.append(lems.TimeDerivative("y", "x/epsilon"))

lems_model.add_component_type(comp_type)

# Add instance of ComponentType
comp = lems.Component(comp_type.name, "dampedOscillator", epsilon="0.1")
lems_model.add_component(comp)


lems_model.gen_sim_with_default_plots(comp.id, 100000, 1) # 100 seconds

lems_model.write_lems_and_run("vanderPol.xml")
```

Van der Pol oscillator






NEURON
Brian
SBML
CellML
NeuroML v2.0
LEMS
LEMS interpreter
Graphs of behaviour
Python
NineML

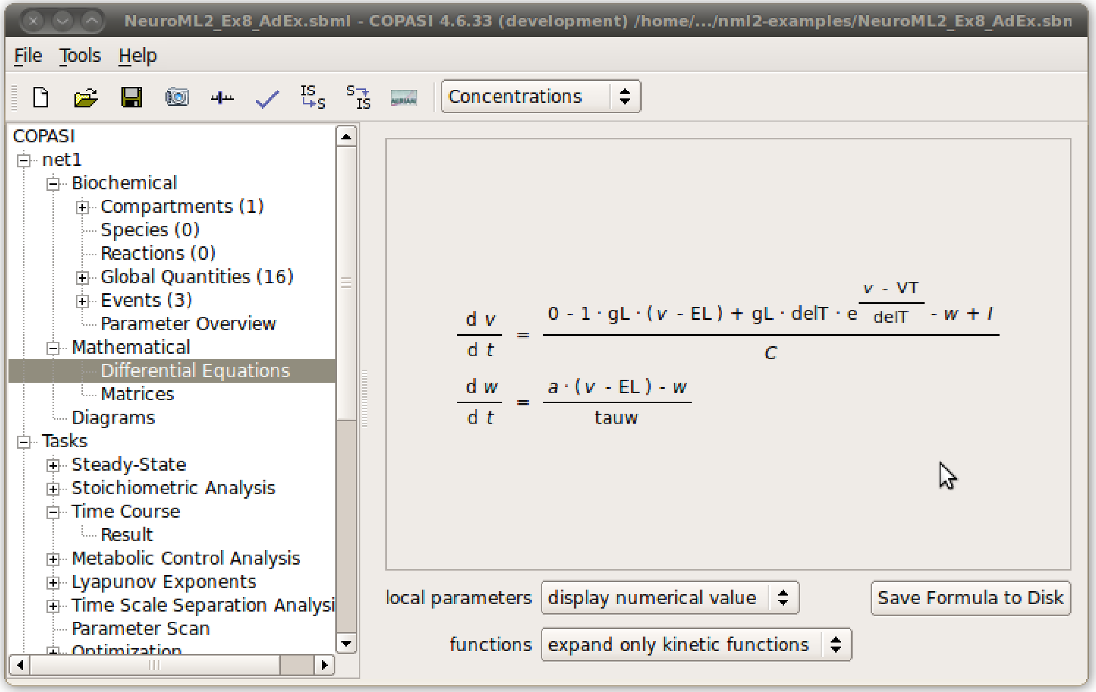
SBML support

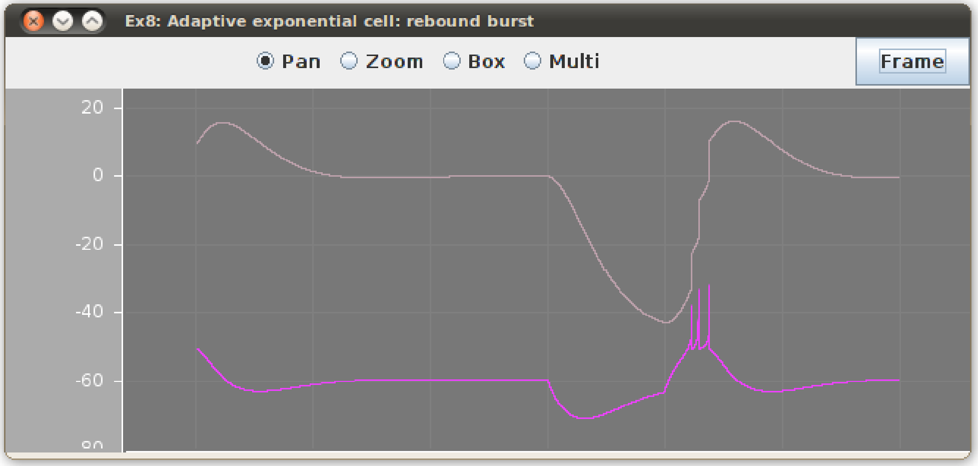
- Export of simple models
  - Izhikevich
  - I & F
  - Adaptive exponential
- Import of many SBML models into LEMS
  - Can be reused as ComponentTypes for any LEMS simulation

SBML export

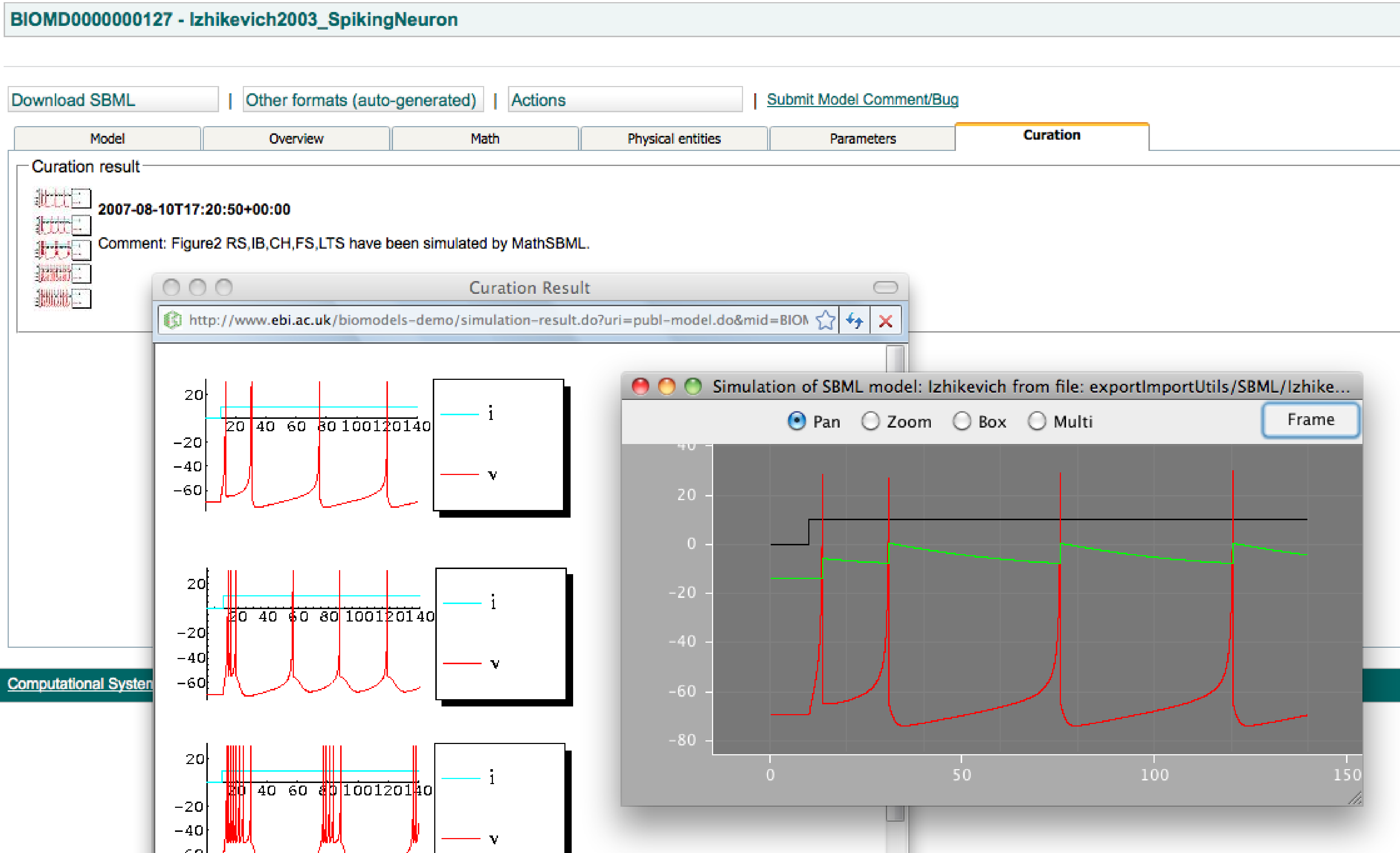

Copasi Plot: plot_0
Print
Save Image
Save Data
Zoom out
Show All
Hide All
Close
plot_0
s
Values[w]|Time


Copasi Plot: plot_1
Print
Save Image
Save Data
Zoom out
Show All
Hide All
Close
plot_1
s
Values[v]|Time


NeuroML2_Ex8_AdEx.sbml - COPASI 4.6.33 (development) /home/.../nml2-examples/NeuroML2_Ex8_AdEx.sbn
File
Tools
Help
Concentrations
COPASI
net1
Biochemical
Compartments (1)
Species (0)
Reactions (0)
Global Quantities (16)
Events (3)
Parameter Overview
Mathematical
Differential Equations
Matrices
Diagrams
Tasks
Steady-State
Stoichiometric Analysis
Time Course
Result
Metabolic Control Analysis
Lyapunov Exponents
Time Scale Separation Analysi
Parameter Scan
$\frac{dv}{dt} = \frac{0 - 1 \cdot gL \cdot (v - EL) + gL \cdot delT \cdot e^{\frac{v - VT}{delT}} - w + I}{C}$
$\frac{dw}{dt} = \frac{a \cdot (v - EL) - w}{tauw}$
local parameters
display numerical value
Save Formula to Disk
functions
expand only kinetic functions


Ex8: Adaptive exponential cell: rebound burst
Pan
Zoom
Box
Multi
Frame

SBML import

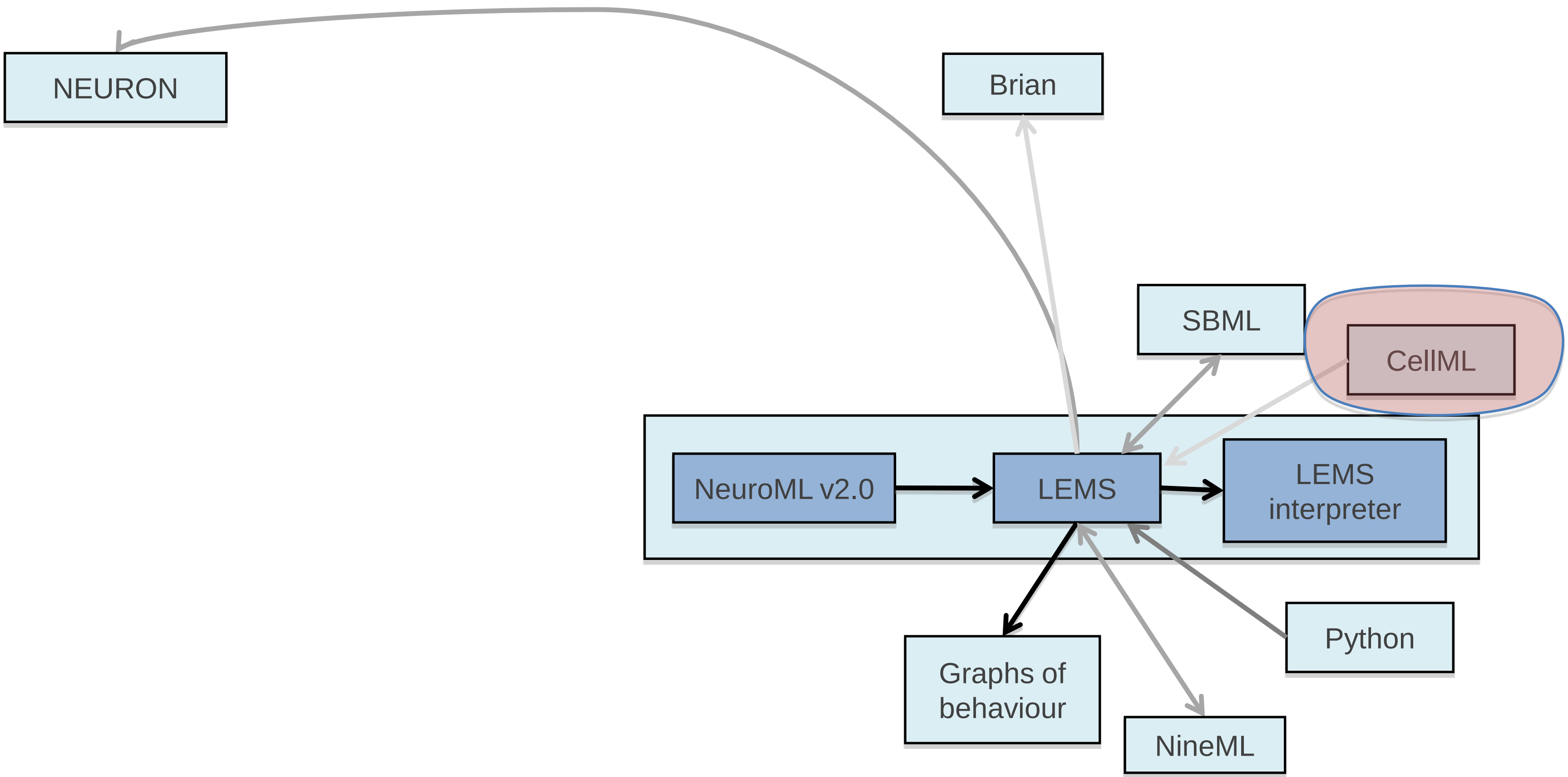

BIOMD0000000127 - Izhikevich2003_SpikingNeuron
Download SBML
Other formats (auto-generated)
Actions
Submit Model Comment/Bug
Model
Overview
Math
Physical entities
Parameters
Curation
Curation result
2007-08-10T17:20:50+00:00
Comment: Figure2 RS,IB,CH,FS,LTS have been simulated by MathSBML.
Curation Result
http://www.ebi.ac.uk/biomodels-demo/simulation-result.do?uri=publ-model.do&mid=BIOM
Simulation of SBML model: Izhikevich from file: exportImportUtils/SBML/Izhike...
Pan
Zoom
Box
Multi
Frame
Computational System

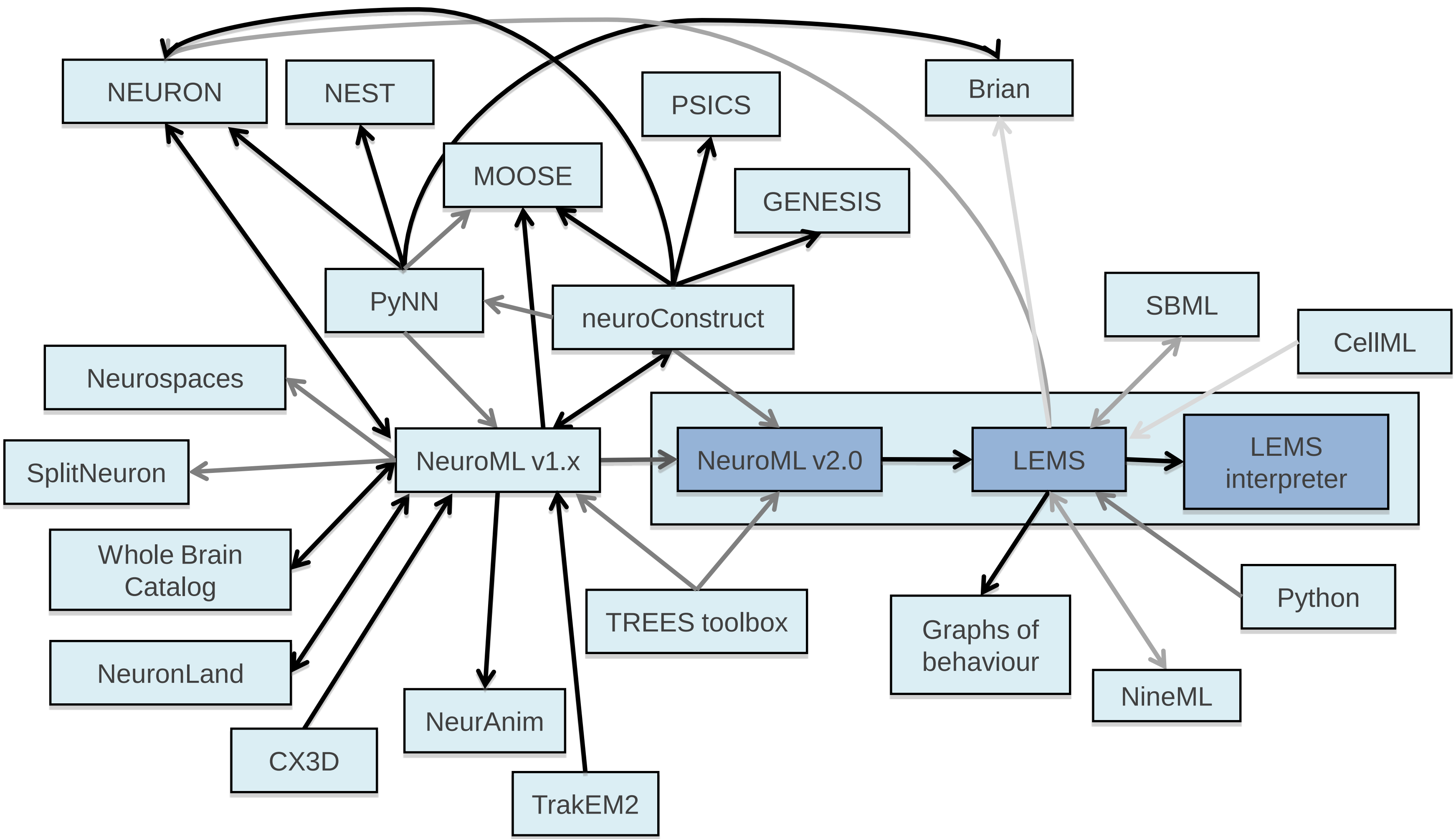
SBML import


BIOMD0000000039 - Marhl2000_CaOscillations
Download SBML
Other formats (auto-generated)
Model
Overview
Math
Curation result
2006-12-17T11:12:29+00:00
Comment: The figure reproduces Calcium bursting oscilla
Computational Systems Neurobiology Group, European Bioinformatics Institute.
Curation Result
http://www.ebi.ac.uk/biomodels-demo/simulation-result.do?uri=publ-model.do&mid=BIOMD0000000039
p.Ca_cyt
2006-12-17T11:12:29+00:00
Done
Simulation of SBML model: Marhl_Calcium_Oscillations_role_of_mitochondria_and_cytosolic_proteins f...
Pan
Zoom
Box
Multi
Frame

SBML support

- Enabled by JSBML
- SBML Test Suite
  - 327 out of 947 SBML models successful matching target data…


NEURON
Brian
SBML
CellML
NeuroML v2.0
LEMS
LEMS interpreter
Graphs of behaviour
NineML
Python


NEURON
NEST
PSICS
Brian
MOOSE
GENESIS
PyNN
neuroConstruct
SBML
CellML
Neurospaces
NeuroML v1.x
NeuroML v2.0
LEMS
LEMS interpreter
SplitNeuron
Whole Brain Catalog
TREES toolbox
Graphs of behaviour
Python
NineML
NeuronLand
NeurAnim
CX3D
TrakEM2