# Integrating NeuroML 2 with PyNN, Brian & CSA

**Padraig Gleeson**
Department of Neuroscience, Physiology and Pharmacology
University College London

## Overview

- Quick introduction to cells & synapses in LEMS & NeuroML 2
- Incorporating PyNN standard cells into this framework
- CSA & PyNN -> NeuroML 2
- NeuroML 2 -> Brian
- Future interaction of NineML & LEMS

# Example: Fitzhugh-Nagumo cell model

$$\dot{V} = V - V^3/3 - W + I$$
$$\dot{W} = 0.08(V + 0.7 - 0.8W)$$

Simplified version of 4 variable HH model

2 state variables, 2 ODEs

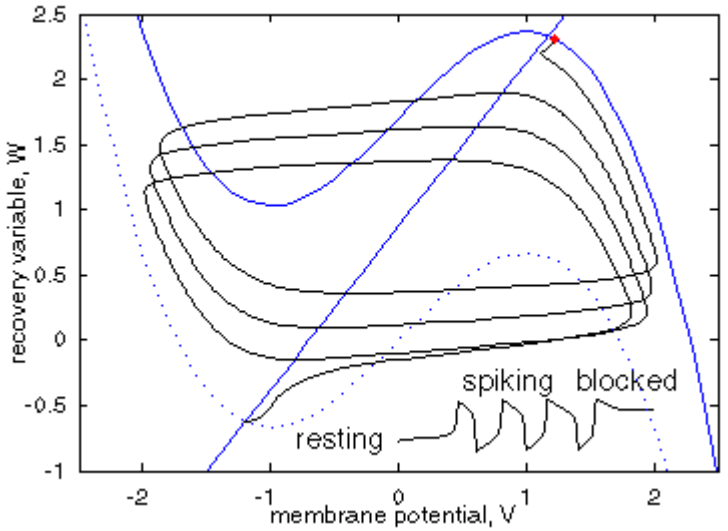



Image from Scholarpedia.org

## Original model

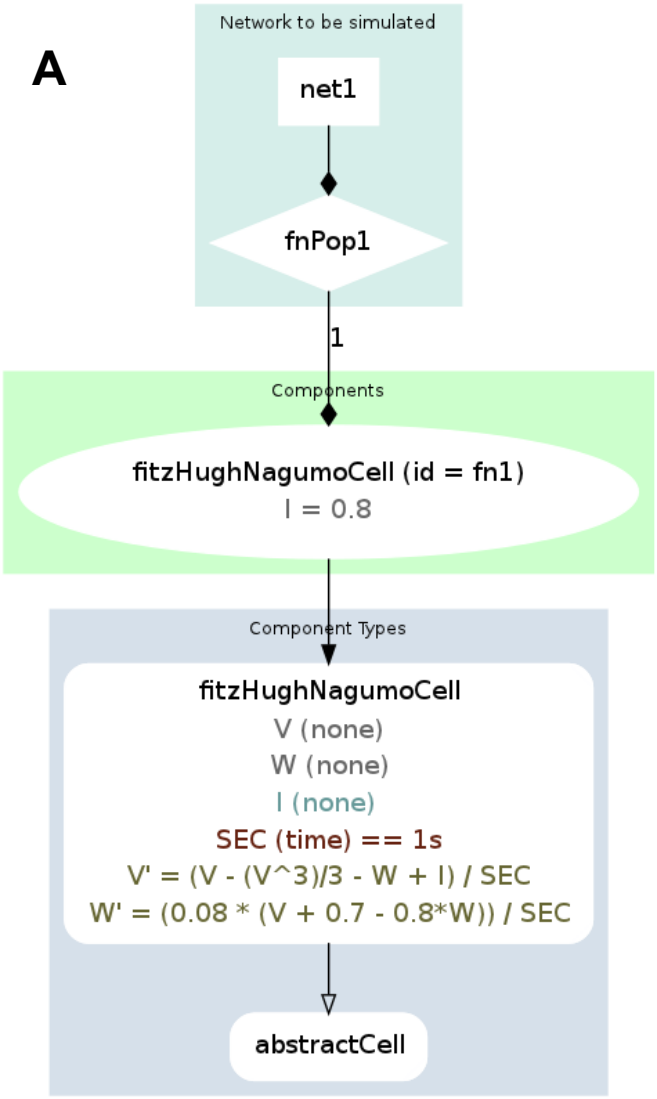
$$\dot{V} = V - V^3/3 - W + I$$
$$\dot{W} = 0.08(V + 0.7 - 0.8W)$$

## Model expressed in **LEMS (Low Entropy Model Specification language)**

```
<ComponentType name="fitzHughNagumoCell" extends="abstractCell">

    <Parameter name="I" dimension="none"/>

    <Constant name="SEC" dimension="time" value="1s"/>

    <Exposure name="V" dimension="none"/>
    <Exposure name="W" dimension="none"/>

    <Behavior>

        <StateVariable name="V" dimension="none" exposure="V"/>
        <StateVariable name="W" dimension="none" exposure="W"/>

        <TimeDerivative variable="V" value="(V - (V^3)/3 - W + I) / SEC"/>
        <TimeDerivative variable="W" value="(0.08 * (V + 0.7 - 0.8*W)) / SEC"/>

    </Behavior>
```

# Component instance in **NeuroML v2.0**

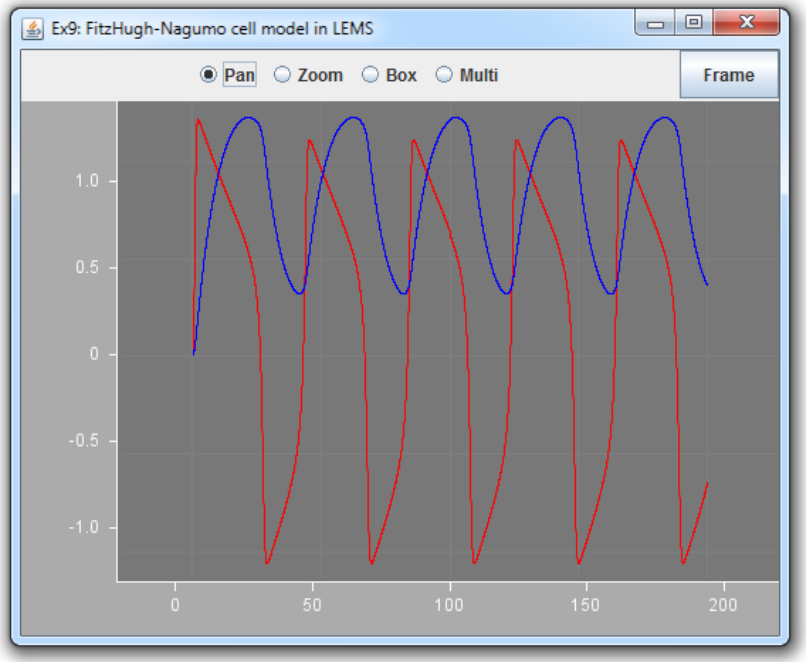
**A**



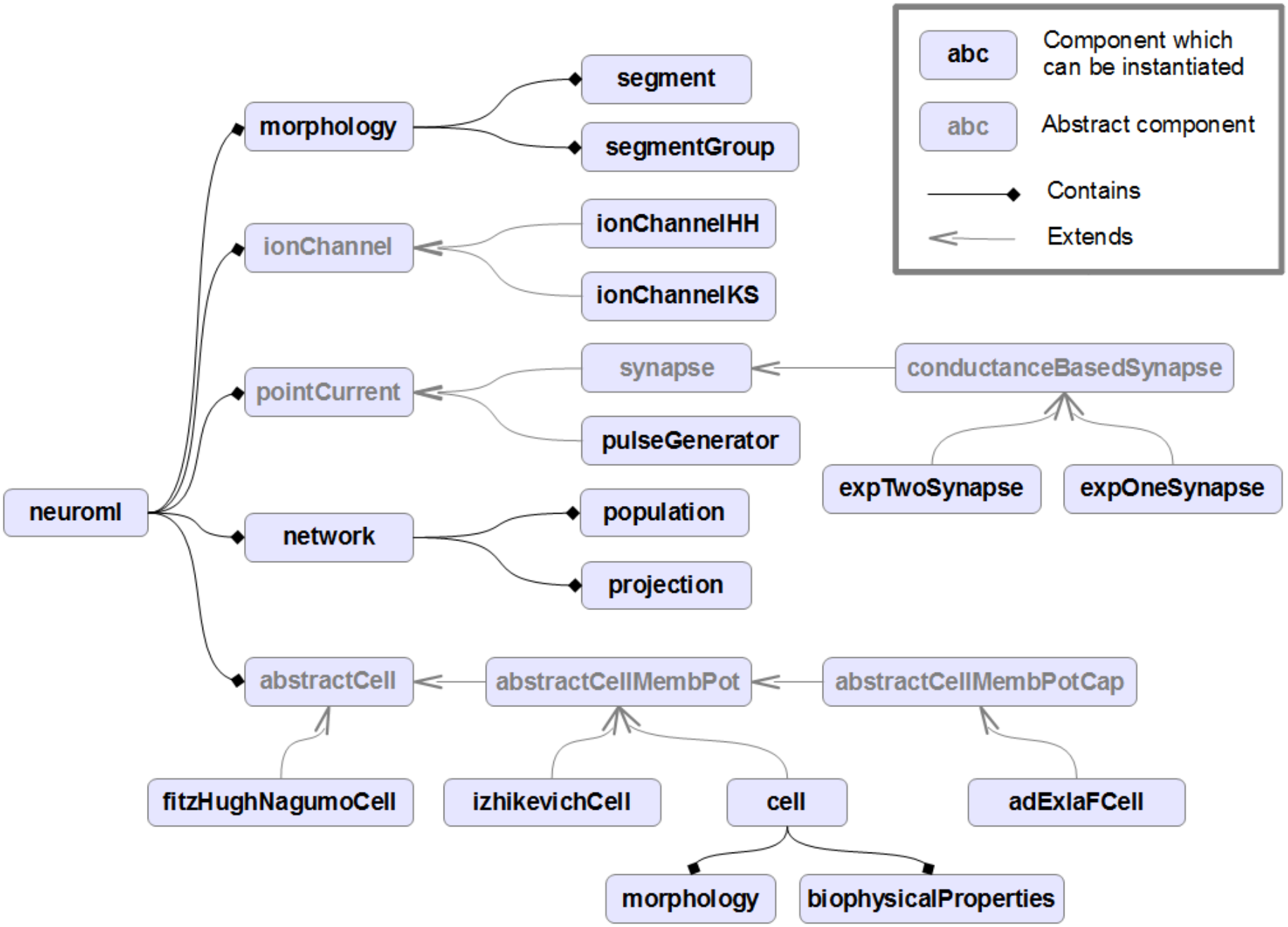

**B**

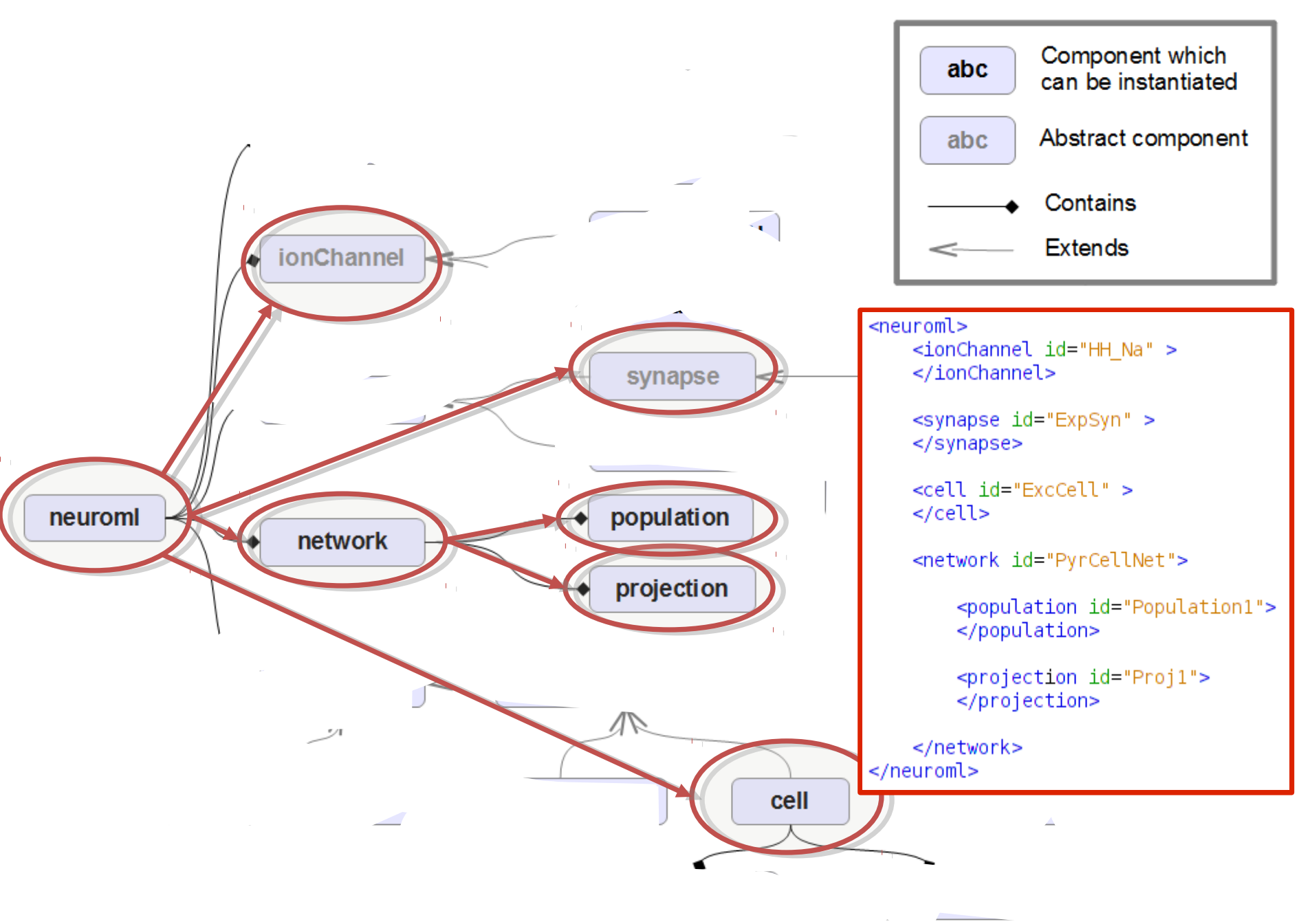
```
<fitzHughNagumoCell id="fn1" I="0.8"/>

<network id="net1">
    <population id="fnPop1" component="fn1" size="1"/>
</network>
```

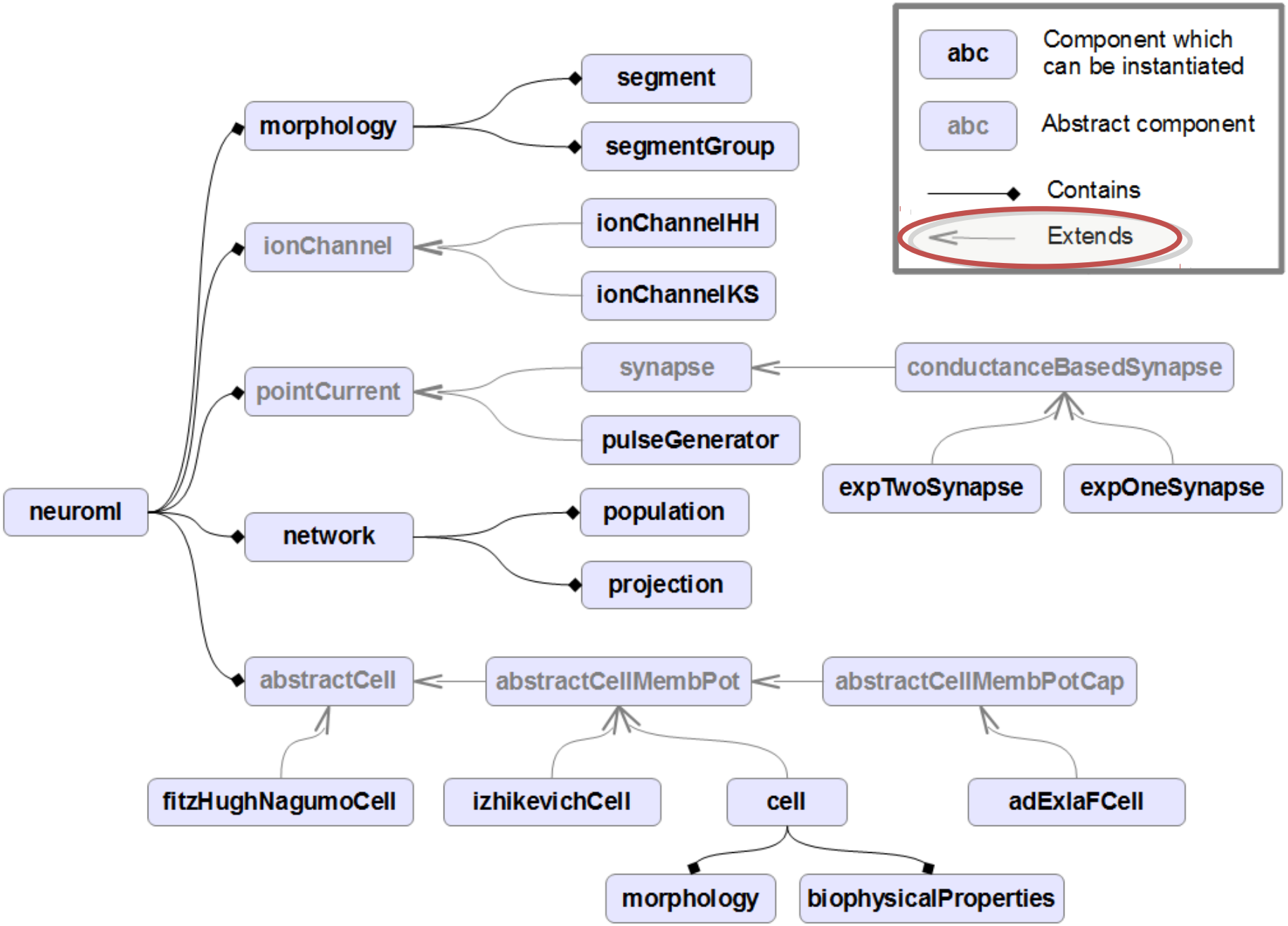
**C**

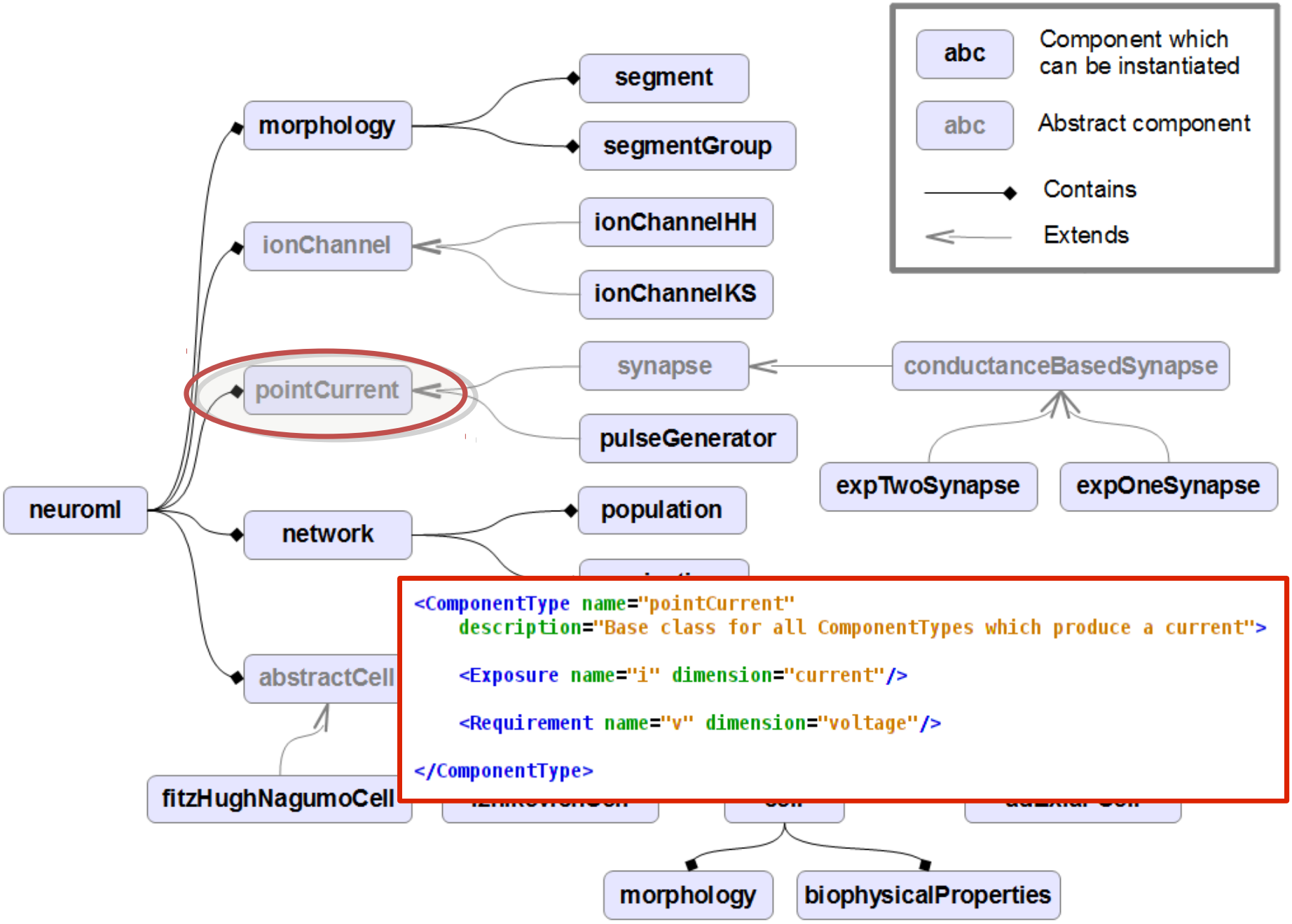


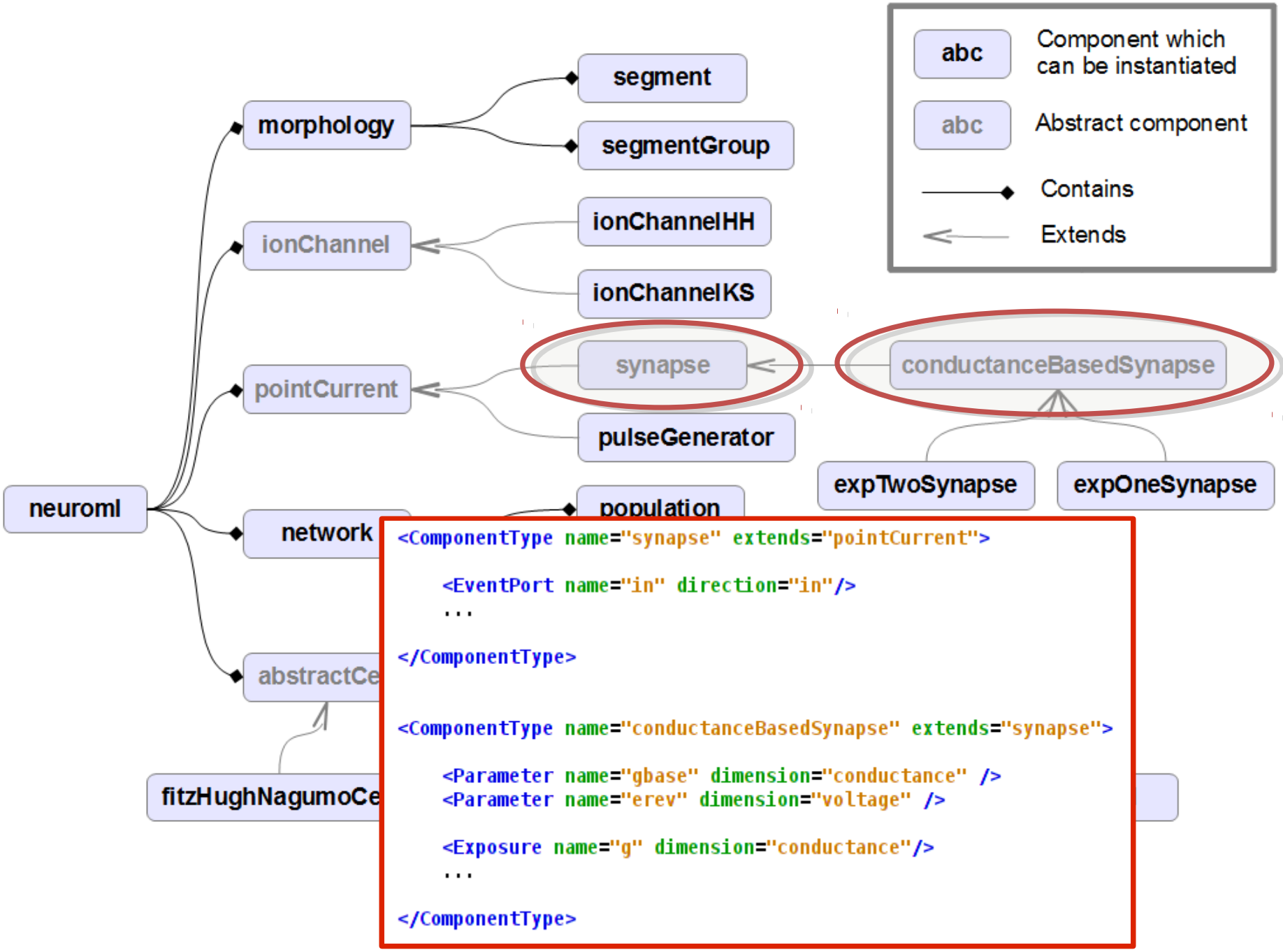

**V**
**W**

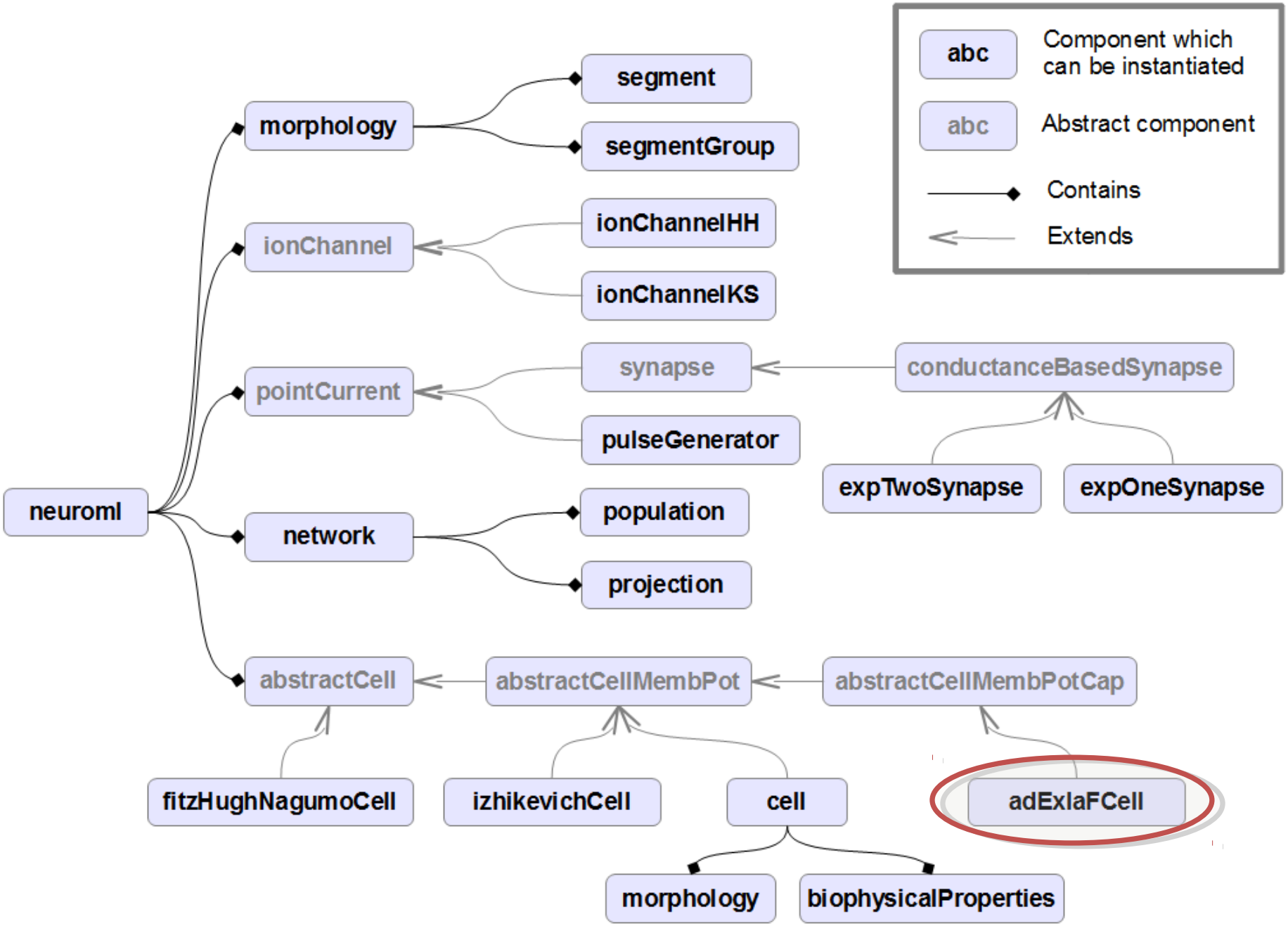

abc
Component which can be instantiated
abc
Abstract component
Contains
Extends
neuroml
morphology
segment
segmentGroup
ionChannel
ionChannelHH
ionChannelKS
pointCurrent
synapse
conductanceBasedSynapse
pulseGenerator
expTwoSynapse
expOneSynapse
network
population
projection
abstractCell
abstractCellMembPot
abstractCellMembPotCap
fitzHughNagumoCell
izhikevichCell
cell
adExIaFCell
morphology
biophysicalProperties


abc
Component which can be instantiated
abc
Abstract component
Contains
Extends
ionChannel
synapse
neuroml
network
population
projection
cell
<neuroml>
<ionChannel id="HH_Na" >
</ionChannel>
<synapse id="ExpSyn" >
</synapse>
<cell id="ExcCell" >
</cell>
<network id="PyrCellNet">
<population id="Population1">
</population>
<projection id="Proj1">
</projection>
</network>
</neuroml>


abc
Component which can be instantiated
abc
Abstract component
Contains
Extends
neuroml
morphology
segment
segmentGroup
ionChannel
ionChannelHH
ionChannelKS
pointCurrent
synapse
conductanceBasedSynapse
pulseGenerator
expTwoSynapse
expOneSynapse
network
population
projection
abstractCell
abstractCellMembPot
abstractCellMembPotCap
fitzHughNagumoCell
izhikevichCell
cell
adExIaFCell
morphology
biophysicalProperties


abc
Component which can be instantiated
abc
Abstract component
Contains
Extends
neuroml
morphology
segment
segmentGroup
ionChannel
ionChannelHH
ionChannelKS
pointCurrent
synapse
conductanceBasedSynapse
pulseGenerator
expTwoSynapse
expOneSynapse
network
population
abstractCell
fitzHughNagumoCell
morphology
biophysicalProperties
<ComponentType name="pointCurrent"
description="Base class for all ComponentTypes which produce a current">
<Exposure name="i" dimension="current"/>
<Requirement name="v" dimension="voltage"/>
</ComponentType>


abc
Component which can be instantiated
abc
Abstract component
Contains
Extends
neuroml
morphology
segment
segmentGroup
ionChannel
ionChannelHH
ionChannelKS
pointCurrent
synapse
pulseGenerator
conductanceBasedSynapse
expTwoSynapse
expOneSynapse
network
population
abstractCe
fitzHughNagumoCe
<ComponentType name="synapse" extends="pointCurrent">
<EventPort name="in" direction="in"/>
...
</ComponentType>
<ComponentType name="conductanceBasedSynapse" extends="synapse">
<Parameter name="gbase" dimension="conductance" />
<Parameter name="erev" dimension="voltage" />
<Exposure name="g" dimension="conductance"/>
...
</ComponentType>


abc
Component which can be instantiated
abc
Abstract component
Contains
Extends
neuroml
morphology
segment
segmentGroup
ionChannel
ionChannelHH
ionChannelKS
pointCurrent
synapse
conductanceBasedSynapse
pulseGenerator
expTwoSynapse
expOneSynapse
network
population
projection
abstractCell
abstractCellMembPot
abstractCellMembPotCap
fitzHughNagumoCell
izhikevichCell
cell
adExIaFCell
morphology
biophysicalProperties

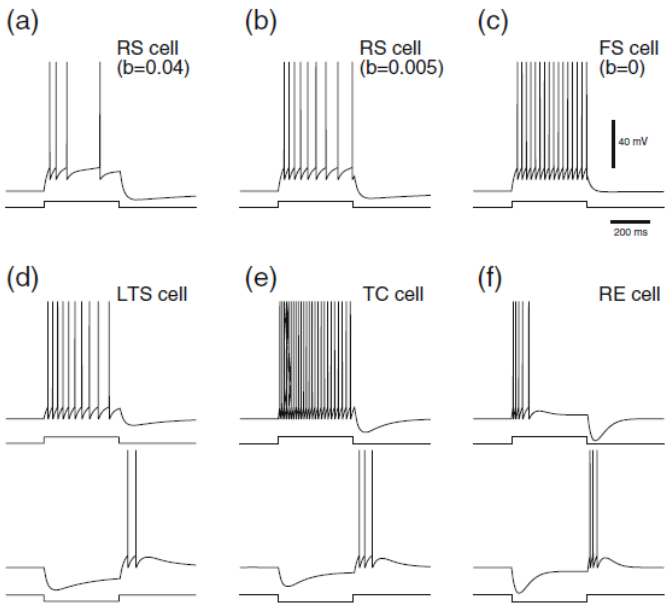
# Adaptive Exponential Integrate & Fire cell

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

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

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

$w \rightarrow w + b$




**adExIaFCell**

w (current)

I (current)

iSyn (current)

iMemb (current)

v (voltage)

gL (conductance), EL (voltage), VT (voltage), thresh (voltage), reset (voltage), delT (voltage), tauw (time), Iamp (current), Idel (time), Idur (time), a (conductance), b (current)

iSyn = synapses[*]/i (REDUCE: add)

iMemb = -1*gL*(v-EL) + gL*delT*exp((v - VT)/delT) - w + I + iSyn

v' = iMemb/C

w' = (a*(v-EL) - w) / tauw

IF (v > thresh) THEN

(v = reset) AND (w = w+b)

IF ((t > Idel) AND (t < (Idel + Idur))) THEN

(I = Iamp)

IF (t > (Idel + Idur)) THEN

(I = 0)

**abstractCellMembPotCap**

iSyn (current)

iMemb (current)

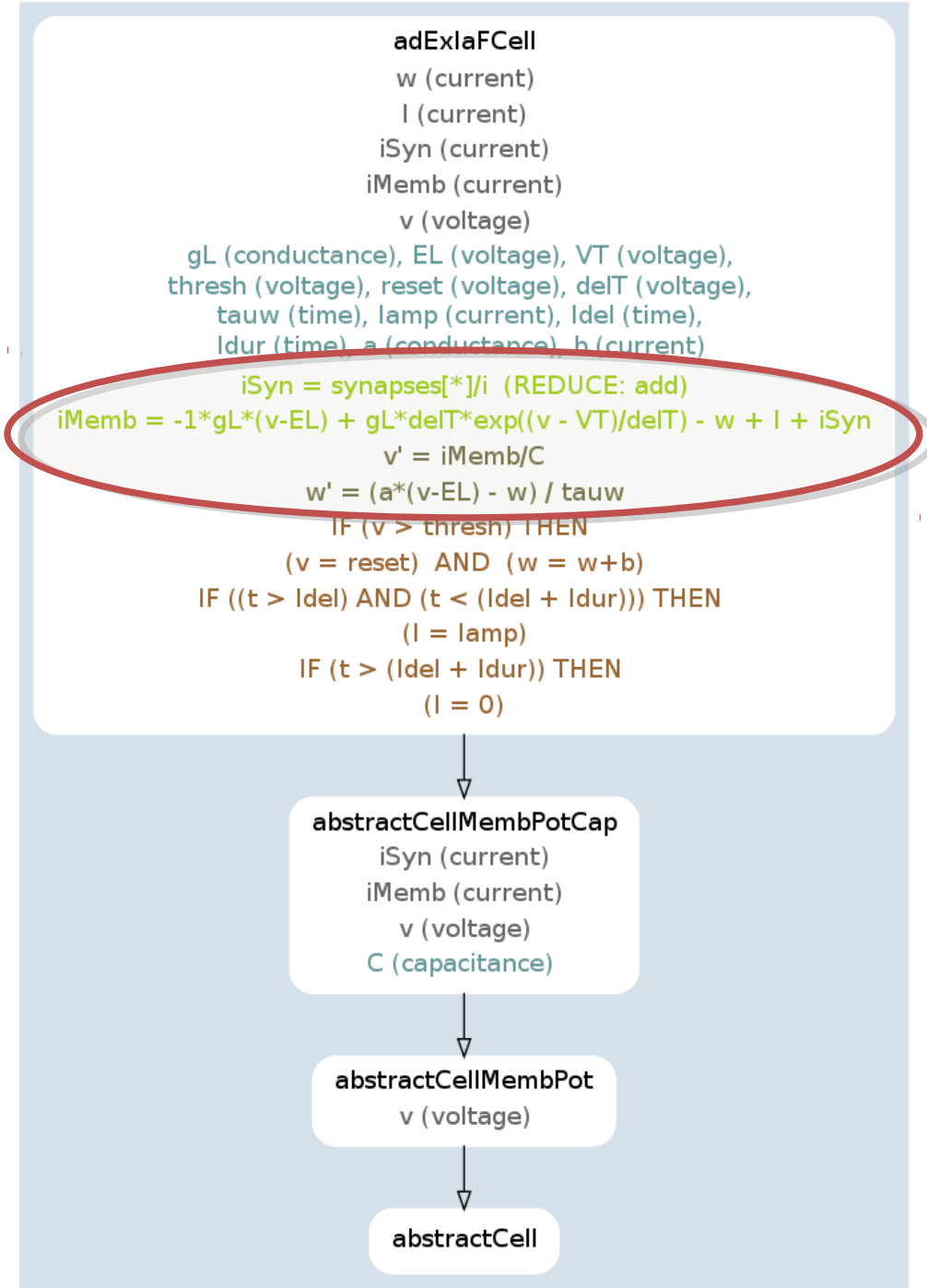
v (voltage)

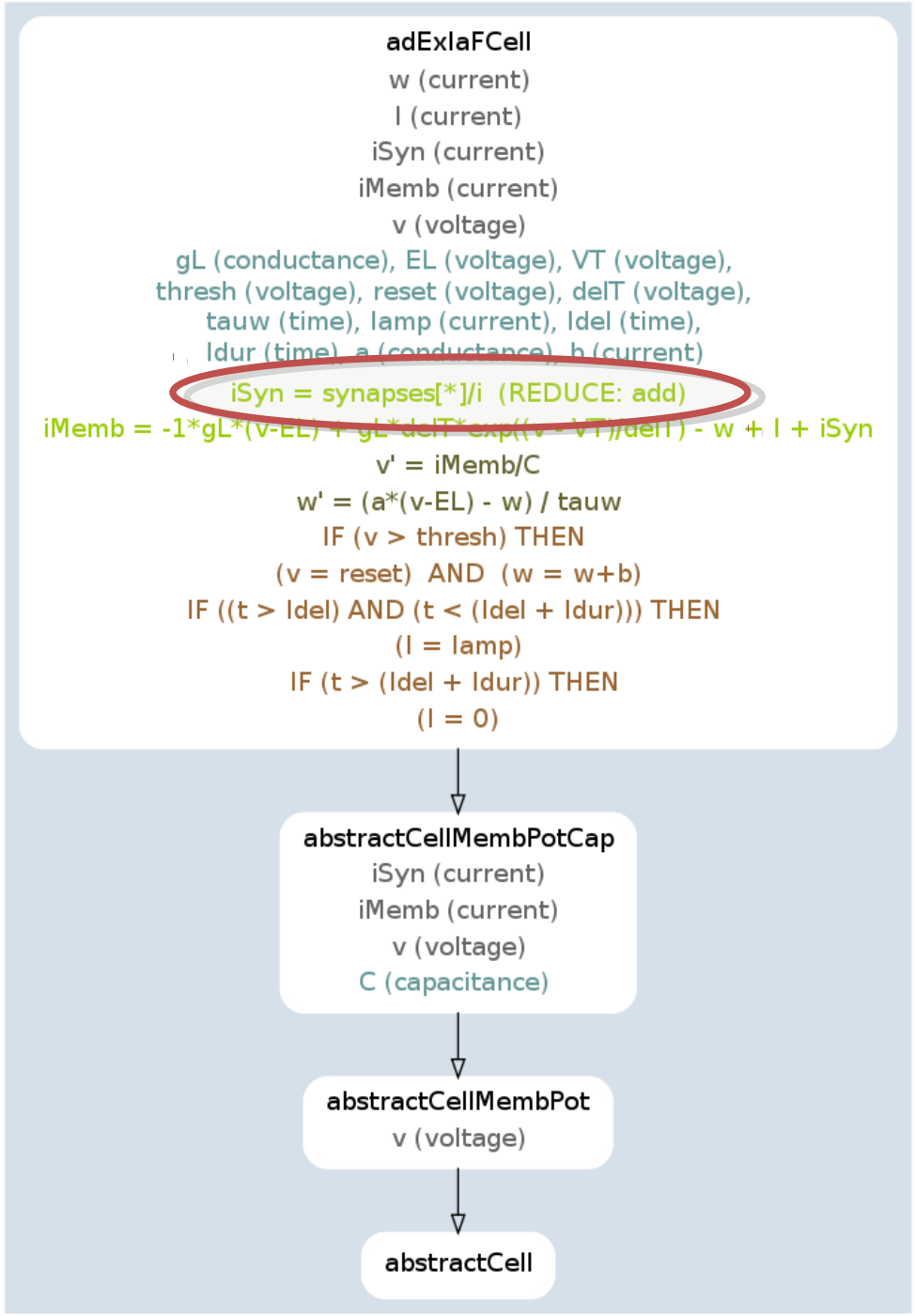
C (capacitance)

**abstractCellMembPot**

v (voltage)

**abstractCell**


adExIaFCell
w (current)
I (current)
iSyn (current)
iMemb (current)
v (voltage)
gL (conductance), EL (voltage), VT (voltage),
thresh (voltage), reset (voltage), delT (voltage),
tauw (time), Iamp (current), Idel (time),
Idur (time), a (conductance), b (current)
iSyn = synapses[*]/i (REDUCE: add)
iMemb = -1*gL*(v-EL) + gL*delT*exp((v - VT)/delT) - w + I + iSyn
v' = iMemb/C
w' = (a*(v-EL) - w) / tauw
IF (v > thresh) THEN
(v = reset) AND (w = w+b)
IF ((t > Idel) AND (t < (Idel + Idur))) THEN
(I = Iamp)
IF (t > (Idel + Idur)) THEN
(I = 0)
abstractCellMembPotCap
iSyn (current)
iMemb (current)
v (voltage)
C (capacitance)
abstractCellMembPot
v (voltage)
abstractCell


adExIaFCell
w (current)
I (current)
iSyn (current)
iMemb (current)
v (voltage)
gL (conductance), EL (voltage), VT (voltage),
thresh (voltage), reset (voltage), delT (voltage),
tauw (time), Iamp (current), Idel (time),
Idur (time), a (conductance), b (current)
iSyn = synapses[*]/i (REDUCE: add)
iMemb = -1*gL*(v-EL) + gL*delT*exp((v - VT)/delT) - w + I + iSyn
v' = iMemb/C
w' = (a*(v-EL) - w) / tauw
IF (v > thresh) THEN
(v = reset) AND (w = w+b)
IF ((t > Idel) AND (t < (Idel + Idur))) THEN
(I = Iamp)
IF (t > (Idel + Idur)) THEN
(I = 0)
abstractCellMembPotCap
iSyn (current)
iMemb (current)
v (voltage)
C (capacitance)
abstractCellMembPot
v (voltage)
abstractCell

# Incorporating PyNN into NeuroML 2

- PyNN is a Python package for simulator independent specification of neuronal network models
- Model code can be developed using the PyNN API and then run using NEURON, NEST, PCSIM, Brian or MOOSE, by replacing:

  **from pyNN.neuron import ***

  with

  **from pyNN.nest import ***

# Incorporating PyNN into NeuroML 2

- Initial implementation allowing export of the network structure to NeuroML 2 using:

  **from pyNN.neuroml2 import ***
- Maps instances of PyNN standard cell models on to equivalent defined in LEMS
- Maps populations in PyNN to <population ...> in LEMS
- Maps connections generated by PyNN to <explicitConnection ...> in LEMS

```
...

ifcell = create(IF_cond_exp, {  'i_offset' : 0.1,     'tau_refrac' : 3.0,
                                'v_thresh' : -51.0,   'tau_syn_E'  : 2.0,
                                'tau_syn_I': 5.0,     'v_reset'    : -70.0,
                                'e_rev_E'  : 0.,      'e_rev_I'    : -80.})

spike_sourceE = create(SpikeSourceArray, {'spike_times': [float(i) for i in range(5,105,10)]})
spike_sourceI = create(SpikeSourceArray, {'spike_times': [float(i) for i in range(155,255,10)]})

connE = connect(spike_sourceE, ifcell, weight=0.006, synapse_type='excitatory',delay=2.0)
connI = connect(spike_sourceI, ifcell, weight=0.02, synapse_type ='inhibitory',delay=4.0)

...
```

```
<?xml version="1.0" ?>
<neuroml id="PyNN2NeuroMLv2"
   xmlns="http://www.neuroml.org/schema/neuroml2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.neuroml.org/schema/neuroml2
   http://neuroml.svn.sourceforge.net/viewvc/neuroml/NeuroML2/Schemas/NeuroML2/NeuroML_v2alpha.xsd">

        <IF_cond_exp cm="1.0" e_rev_E="0.0" e_rev_I="-80.0" i_offset="0.1" id="cell_population0"
              tau_m="20.0" tau_refrac="3.0" tau_syn_E="2.0" tau_syn_I="5.0" v_init="-65"
              v_reset="-70.0"  v_rest="-65.0" v_thresh="-51.0">
               <notes>
                       Component for PyNN IF_cond_exp cell type
               </notes>
       </IF_cond_exp>
       <expCondSynapse e_rev="0.0" id="syn_e_cell_population0" tau_syn="2.0"/>
       <expCondSynapse e_rev="-80.0" id="syn_i_cell_population0" tau_syn="5.0"/>
       <spikeArray id="cell_population1">
               <spike time="5.000000ms"/>
               <spike time="15.000000ms"/>
               <spike time="25.000000ms"/>
               <spike time="35.000000ms"/>
               <spike time="45.000000ms"/>
               <spike time="55.000000ms"/>
               <spike time="65.000000ms"/>
               <spike time="75.000000ms"/>
               <spike time="85.000000ms"/>
               <spike time="95.000000ms"/>
       </spikeArray>
       <spikeArray id="cell_population2">
               <spike time="155.000000ms"/>
```

```
        <spikeArray id="cell_population1">
                <spike time="5.000000ms"/>
                <spike time="15.000000ms"/>
                <spike time="25.000000ms"/>
                <spike time="35.000000ms"/>
                <spike time="45.000000ms"/>
                <spike time="55.000000ms"/>
                <spike time="65.000000ms"/>
                <spike time="75.000000ms"/>
                <spike time="85.000000ms"/>
                <spike time="95.000000ms"/>
        </spikeArray>
        <spikeArray id="cell_population2">
                <spike time="155.000000ms"/>
                <spike time="165.000000ms"/>
                <spike time="175.000000ms"/>
                <spike time="185.000000ms"/>
                <spike time="195.000000ms"/>
                <spike time="205.000000ms"/>
                <spike time="215.000000ms"/>
                <spike time="225.000000ms"/>
                <spike time="235.000000ms"/>
                <spike time="245.000000ms"/>
        </spikeArray>
        <network id="network_PyNN2NeuroMLv2">
                <population component="cell_population0" id="population0" size="1"/>
                <population component="cell_population1" id="population1" size="1"/>
                <population component="cell_population2" id="population2" size="1"/>
                <synapticConnectionWD delay="2.0ms" from="population1[0]"
                    synapse="syn_e_cell_population0" to="population0[0]" weight="0.006"/>
                <synapticConnectionWD delay="4.0ms" from="population2[0]"
                    synapse="syn_i_cell_population0" to="population0[0]" weight="0.02"/>
        </network>
</neuroml>
```

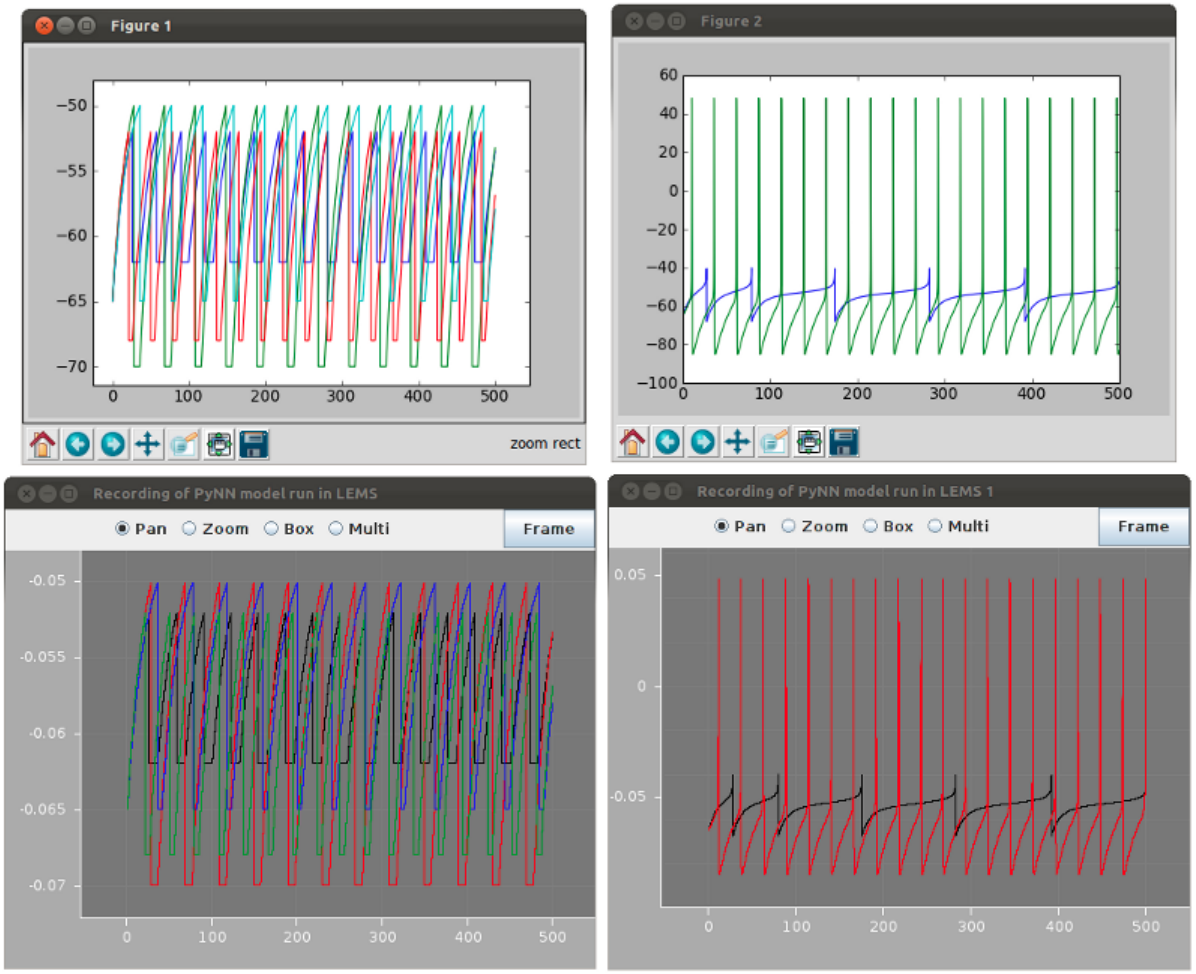
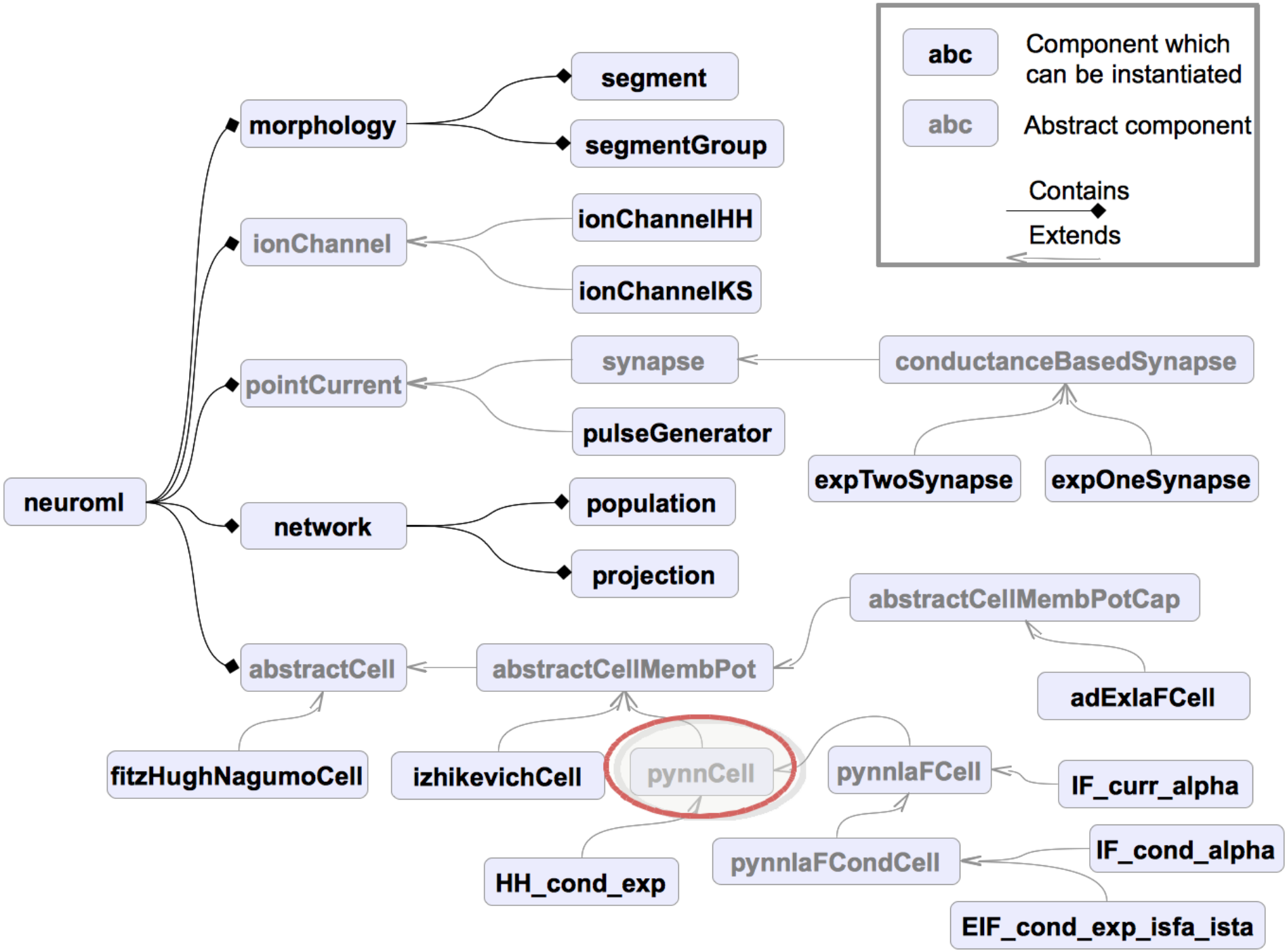
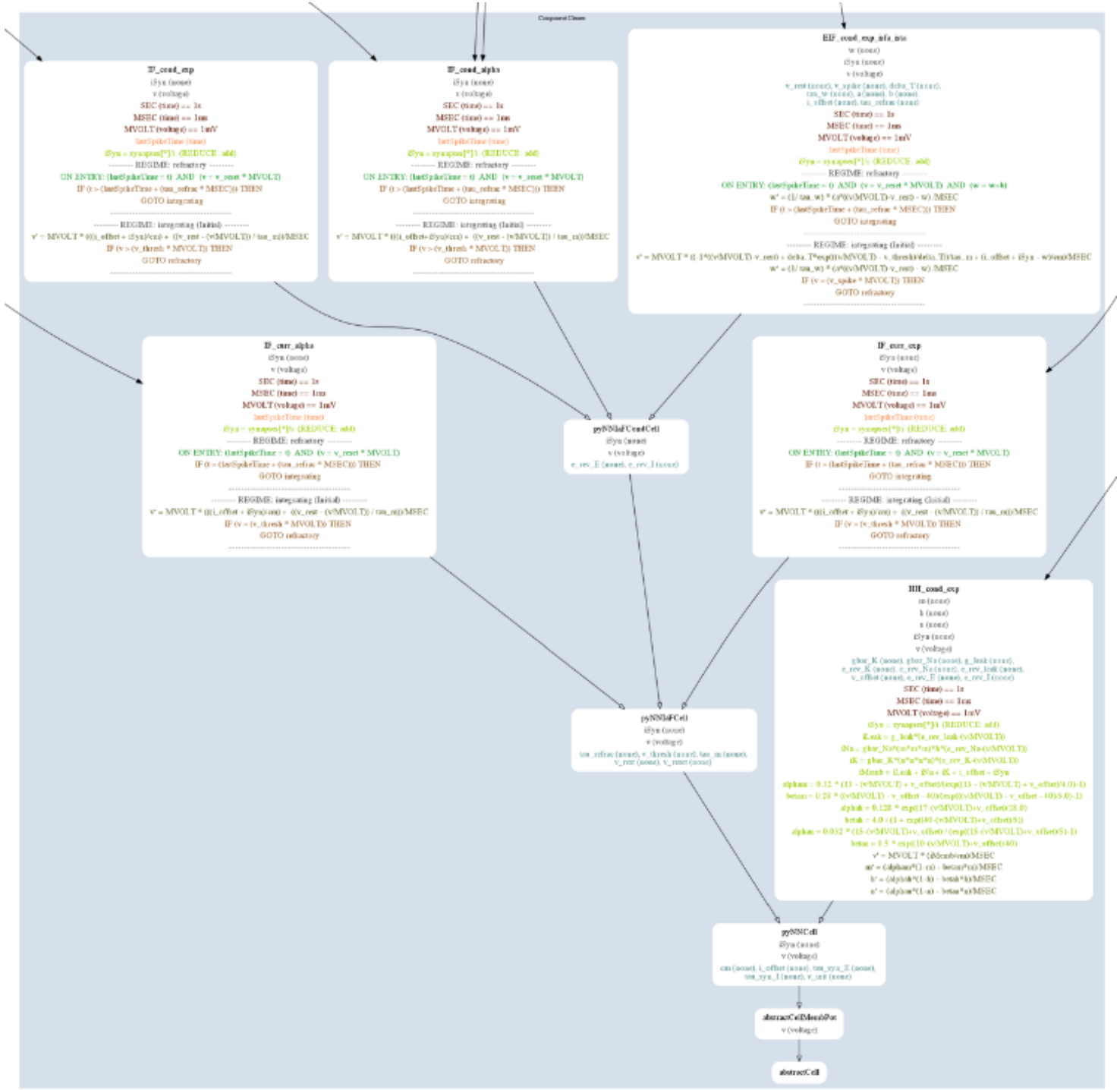


Figure 1
zoom rect
Figure 2
Recording of PyNN model run in LEMS
Pan Zoom Box Multi
Frame
Recording of PyNN model run in LEMS 1
Pan Zoom Box Multi
Frame


abc
Component which can be instantiated
abc
Abstract component
Contains
Extends
neuroml
morphology
segment
segmentGroup
ionChannel
ionChannelHH
ionChannelKS
pointCurrent
synapse
conductanceBasedSynapse
pulseGenerator
expTwoSynapse
expOneSynapse
network
population
projection
abstractCell
abstractCellMembPot
abstractCellMembPotCap
adExIaFCell
fitzHughNagumoCell
izhikevichCell
pynnCell
pynnIaFCell
IF_curr_alpha
HH_cond_exp
pynnIaFCondCell
IF_cond_alpha
EIF_cond_exp_isfa_ista


Component Classes
IF_cond_exp
iSyn (none)
v (voltage)
SEC (time) == 1s
MSEC (time) == 1ms
MVOLT (voltage) == 1mV
lastSpikeTime (time)
iSyn = synapses[*]\ (REDUCE: add)
-------- REGIME: refractory --------
ON ENTRY: (lastSpikeTime = 0) AND (v = v_reset * MVOLT)
IF (t > (lastSpikeTime + (tau_refrac * MSEC))) THEN
GOTO integrating
-------- REGIME: integrating (Initial) --------
v' = MVOLT * (((i_offset + iSyn)/cm) + ((v_rest - (v/MVOLT)) / tau_m))/MSEC
IF (v > (v_thresh * MVOLT)) THEN
GOTO refractory
IF_cond_alpha
iSyn (none)
v (voltage)
SEC (time) == 1s
MSEC (time) == 1ms
MVOLT (voltage) == 1mV
lastSpikeTime (time)
iSyn = synapses[*]\ (REDUCE: add)
-------- REGIME: refractory --------
ON ENTRY: (lastSpikeTime = 0) AND (v = v_reset * MVOLT)
IF (t > (lastSpikeTime + (tau_refrac * MSEC))) THEN
GOTO integrating
-------- REGIME: integrating (Initial) --------
v' = MVOLT * (((i_offset+ iSyn)/cm) + ((v_rest - (v/MVOLT)) / tau_m))/MSEC
IF (v > (v_thresh * MVOLT)) THEN
GOTO refractory
EIF_cond_exp_isfa_ista
w (none)
iSyn (none)
v (voltage)
v_reset (none), v_spike (none), delta_T (none),
tau_w (none), a (none), b (none),
i_offset (none), tau_refrac (none)
SEC (time) == 1s
MSEC (time) == 1ms
MVOLT (voltage) == 1mV
lastSpikeTime (time)
iSyn = synapses[*]\ (REDUCE: add)
-------- REGIME: refractory --------
ON ENTRY: (lastSpikeTime = 0) AND (v = v_reset * MVOLT) AND (w = w_b)
w' = ((1/ tau_w) * ((a*(v/MVOLT)-v_rest) - w))/MSEC
IF (t > (lastSpikeTime + (tau_refrac * MSEC))) THEN
GOTO integrating
-------- REGIME: integrating (Initial) --------
v' = MVOLT * (((-1)*((v/MVOLT)-v_rest) + delta_T*exp((1/MVOLT) - v_thresh/delta_T))/tau_m) + ((i_offset + iSyn - w)/cm))/MSEC
w' = ((1/ tau_w) * ((a*(v/MVOLT)-v_rest) - w))/MSEC
IF (v > (v_spike * MVOLT)) THEN
GOTO refractory
IF_curr_alpha
iSyn (none)
v (voltage)
SEC (time) == 1s
MSEC (time) == 1ms
MVOLT (voltage) == 1mV
lastSpikeTime (time)
iSyn = synapses[*]\ (REDUCE: add)
-------- REGIME: refractory --------
ON ENTRY: (lastSpikeTime = 0) AND (v = v_reset * MVOLT)
IF (t > (lastSpikeTime + (tau_refrac * MSEC))) THEN
GOTO integrating
-------- REGIME: integrating (Initial) --------
v' = MVOLT * (((i_offset + iSyn)/cm) + ((v_rest - (v/MVOLT)) / tau_m))/MSEC
IF (v > (v_thresh * MVOLT)) THEN
GOTO refractory
pyNNIaFCondCell
iSyn (none)
v (voltage)
e_rev_E (none), e_rev_I (none)
IF_curr_exp
iSyn (none)
v (voltage)
SEC (time) == 1s
MSEC (time) == 1ms
MVOLT (voltage) == 1mV
lastSpikeTime (time)
iSyn = synapses[*]\ (REDUCE: add)
-------- REGIME: refractory --------
ON ENTRY: (lastSpikeTime = 0) AND (v = v_reset * MVOLT)
IF (t > (lastSpikeTime + (tau_refrac * MSEC))) THEN
GOTO integrating
-------- REGIME: integrating (Initial) --------
v' = MVOLT * (((i_offset + iSyn)/cm) + ((v_rest - (v/MVOLT))) / tau_m))/MSEC
IF (v > (v_thresh * MVOLT)) THEN
GOTO refractory
pyNNIaFCell
iSyn (none)
v (voltage)
tau_refrac (none), v_thresh (none), tau_m (none),
v_rest (none), v_reset (none)
HH_cond_exp
m (none)
h (none)
n (none)
iSyn (none)
v (voltage)
g_bar_K (none), g_bar_Na (none), g_leak (none),
e_rev_K (none), e_rev_Na (none), e_rev_leak (none),
v_offset (none), e_rev_E (none), e_rev_I (none)
SEC (time) == 1s
MSEC (time) == 1ms
MVOLT (voltage) == 1mV
iSyn = synapses[*]\ (REDUCE: add)
iLeak = g_leak*(e_rev_leak-(v/MVOLT))
iNa = g_bar_Na*(m*m*m)*h*(e_rev_Na-(v/MVOLT))
iK = g_bar_K*(n*n*n*n)*(e_rev_K-(v/MVOLT))
iMembr = iLeak + iNa + iK + i_offset + iSyn
alpham = 0.32 * (13 - (v/MVOLT) + v_offset)/(exp((13 - (v/MVOLT) + v_offset)/4.0)-1)
betam = 0.28 * ((v/MVOLT) - v_offset - 40)/(exp((v/MVOLT) - v_offset - 40)/5.0)-1)
alphah = 0.128 * exp((17-(v/MVOLT)+v_offset)/18.0)
betah = 4.0 /(1 + exp((40-(v/MVOLT)+v_offset)/5))
alphan = 0.032 * (15-(v/MVOLT)+v_offset) / (exp((15-(v/MVOLT)+v_offset)/5)-1)
betan = 0.5 * exp((10-(v/MVOLT)+v_offset)/40)
v' = MVOLT * (iMembr/cm)/MSEC
m' = (alpham*(1-m) - betam*m)/MSEC
h' = (alphah*(1-h) - betah*h)/MSEC
n' = (alphan*(1-n) - betan*n)/MSEC
pyNNCell
iSyn (none)
v (voltage)
cm (none), i_offset (none), tau_syn_E (none),
tau_syn_I (none), v_rest (none)
abstractCellModelOut
v (voltage)
abstractCell

# NeuroML & Connection Set Algebra

- PyNN & CSA already well integrated...

```
input_population  = Population(10, SpikeSourceArray, {'spike_times': spike_times }, label="input")
output_population = Population(10, IF_curr_exp, cell_params, label="output")

g=csa.grid2d(3)
d=csa.euclidMetric2d(g,g)
connector = CSAConnector(csa.cset(csa.random(0.5), csa.gaussian(0.1,1.0)*d, 1.0))

projection = Projection(input_population, output_population, connector, rng=rng)
```

- Connections described with CSA can be used in PyNN scripts & exported to NeuroML 2

```
        <spike time="351.090862ms"/>
        <spike time="358.827891ms"/>
        <spike time="362.685506ms"/>
        <spike time="365.812919ms"/>
        <spike time="375.407497ms"/>
        <spike time="403.008045ms"/>
        <spike time="412.715018ms"/>
</spikeArray>
<IF_curr_exp cm="1.0" i_offset="0.0" id="cell_output" tau_m="20.0" tau_refrac="2.0" tau_syn_E="2.0" tau_syn_I="2.0" v_init="-65"
                        v_reset="-65.0" v_rest="-65.0" v_thresh="-50.0">

</IF_curr_exp>
<expCurrSynapse id="syn_e_cell_output" tau_syn="2.0"/>
<expCurrSynapse id="syn_i_cell_output" tau_syn="2.0"/>
<network id="network_PyNN2NeuroMLv2">
        <population component="cell_input" id="input" size="10"/>
        <population component="cell_output" id="output" size="10"/>
        <synapticConnectionWD delay="1.0ms" from="input[3]" synapse="syn_e_cell_output" to="output[0]" weight="0.00386592013947"/>
        <synapticConnectionWD delay="1.0ms" from="input[7]" synapse="syn_e_cell_output" to="output[0]" weight="8.63504075338e-13"/>
        <synapticConnectionWD delay="1.0ms" from="input[8]" synapse="syn_e_cell_output" to="output[0]" weight="4.9891093928e-20"/>
        <synapticConnectionWD delay="1.0ms" from="input[9]" synapse="syn_e_cell_output" to="output[0]" weight="0.0"/>
        <synapticConnectionWD delay="1.0ms" from="input[0]" synapse="syn_e_cell_output" to="output[1]" weight="0.00386592013947"/>
        <synapticConnectionWD delay="1.0ms" from="input[1]" synapse="syn_e_cell_output" to="output[1]" weight="1.0"/>
        <synapticConnectionWD delay="1.0ms" from="input[2]" synapse="syn_e_cell_output" to="output[1]" weight="0.00386592013947"/>
        <synapticConnectionWD delay="1.0ms" from="input[4]" synapse="syn_e_cell_output" to="output[1]" weight="0.00386592013947"/>
        <synapticConnectionWD delay="1.0ms" from="input[8]" synapse="syn_e_cell_output" to="output[1]" weight="8.63504075338e-13"/>
        <synapticConnectionWD delay="1.0ms" from="input[9]" synapse="syn_e_cell_output" to="output[1]" weight="0.0"/>
        <synapticConnectionWD delay="1.0ms" from="input[1]" synapse="syn_e_cell_output" to="output[2]" weight="0.00386592013947"/>
        <synapticConnectionWD delay="1.0ms" from="input[2]" synapse="syn_e_cell_output" to="output[2]" weight="1.0"/>
        <synapticConnectionWD delay="1.0ms" from="input[6]" synapse="syn_e_cell_output" to="output[2]" weight="4.9891093928e-20"/>
```

# NeuroML & Brian interoperability

- Brian is a pure Python simulator
- Easy to specify new neuron models

```
from brian import *
eqs = '''
dv/dt = (ge+gi-(v+49*mV))/(20*ms) : volt
dge/dt = -ge/(5*ms) : volt
dgi/dt = -gi/(10*ms) : volt
'''
P = NeuronGroup(4000, eqs, threshold=-50*mV, reset=-60*mV)
P.v = -60*mV
Pe = P.subgroup(3200)
Pi = P.subgroup(800)
Ce = Connection(Pe, P, 'ge', weight=1.62*mV, sparseness=0.02)
Ci = Connection(Pi, P, 'gi', weight=-9*mV, sparseness=0.02)
M = SpikeMonitor(P)
run(1*second)
raster_plot(M)
show()
```

# Example

```
<neuroml xmlns="http://www.neuroml.org/schema/neuroml2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.neuroml.org/schema/neuroml2  ../Schemas/NeuroML2/NeuroML_v2alpha.xsd"
    id="NML2_AbstractCells">


    <iafTauCell id="iafTau" leakReversal="-50mV" thresh="-55mV" reset="-70mV" tau="30ms"/>

    <iafCell id="iafCell" leakReversal="-50mV" thresh="-55mV" reset="-70mV" C="0.2nF" leakConductance="0.01uS"/>

    <izhikevichCell id="izBurst" v0 = "-70mV" thresh = "30mV" a ="0.02" b = "0.2" c = "-50.0" d = "2" Iamp="15" Idel="0ms" Idur="2000ms"/>

</neuroml>
```

```
eqs = '''
dv/dt = (0.04 * v**2 / MVOLT + 5 * v + (140.0 - U + I) * MVOLT)/MSEC : mvolt
dU/dt = a * (b * v / MVOLT - U) / MSEC : 1
I : 1
MSEC : msecond
MVOLT : mvolt
v0 : mvolt
a : 1
b : 1
c : 1
d : 1
thresh : mvolt
Iamp : 1
Idel : msecond
Idur : msecond
'''
```

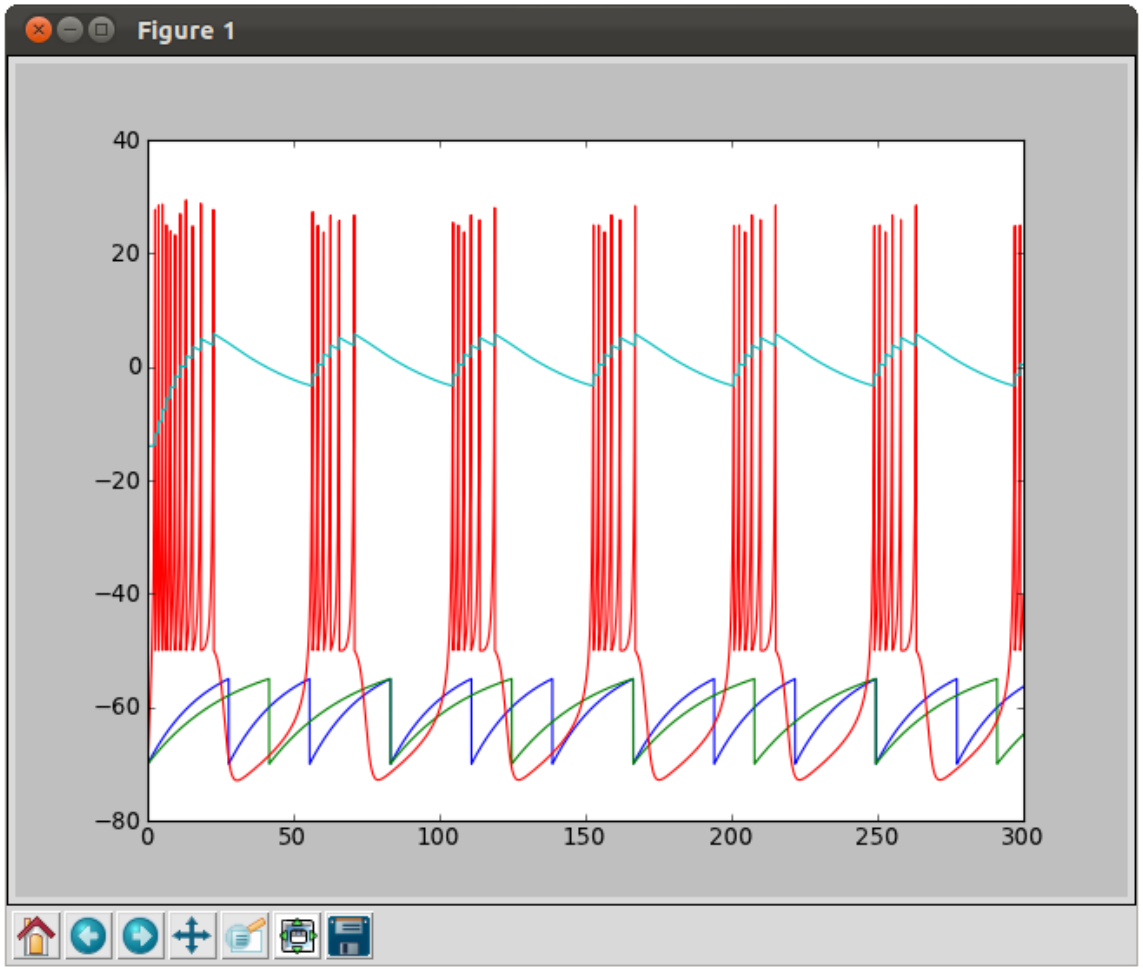


Figure 1
40
20
0
−20
−40
−60
−80
0
50
100
150
200
250
300

# LEMS/NeuroML & NineML

Much overlap between current NineML abstraction layer & LEMS

- Component – Component
- ComponentType – ComponentClass
- Parameter – Parameter

NineML missing key concepts like composition, extension

Proposed framework for interaction:

- Have consistent way to “flatten” LEMS descriptions
- Useful too for mappings to Brian, MATLAB, SBML...

## Conclusions

- Declarative specifications of cell model behaviour useful for cross simulator interoperability
- Procedural specification of network structure affords great flexibility
- Combination of the two will be useful for moving forward & sharing models