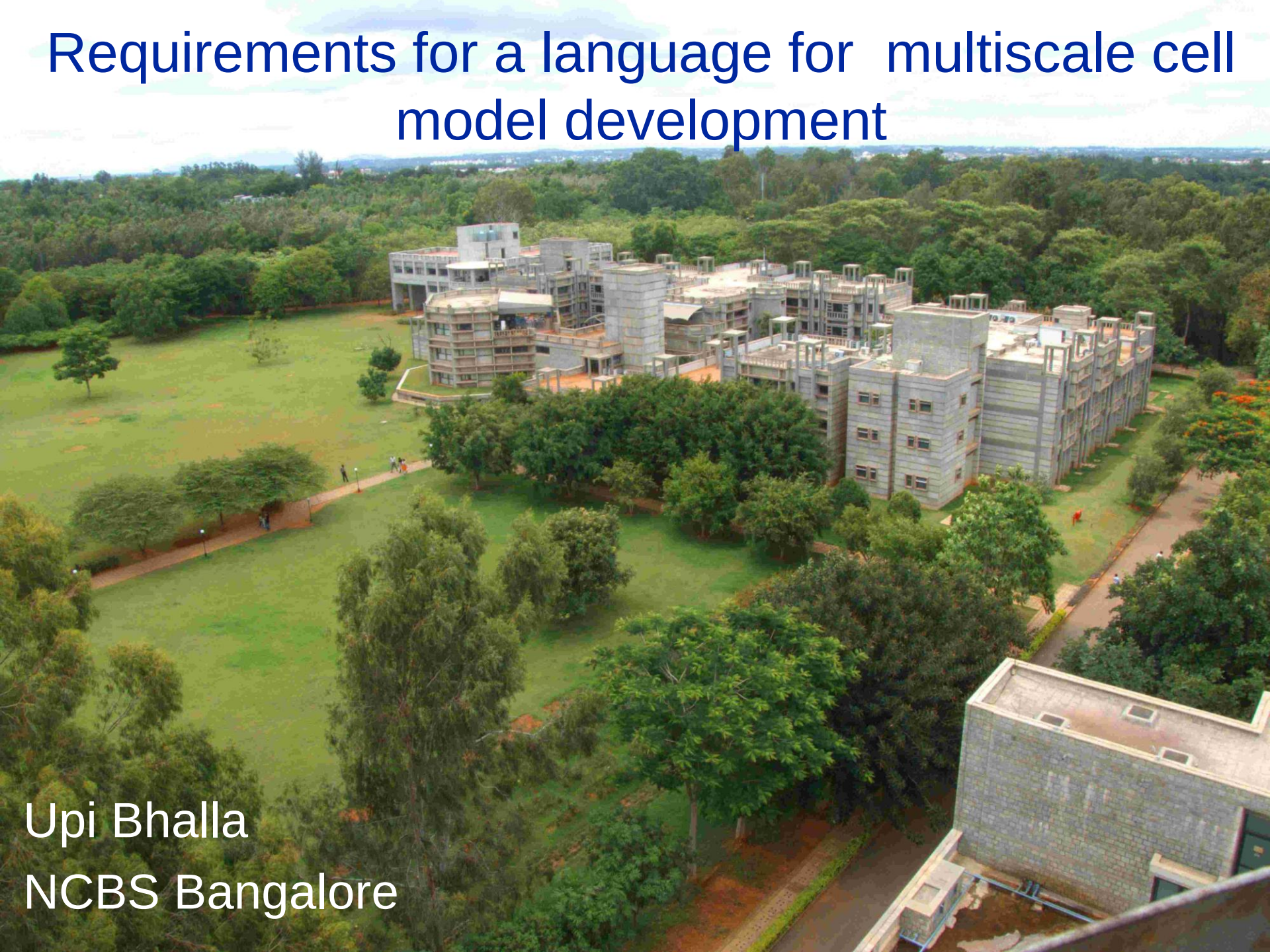
Requirements for a language for multiscale cell model development



Upi Bhalla
NCBS Bangalore

Levels of description

- Behaviour
- Systems
- Areas
- Circuits
- Neurons
- Dendrites
- Synapses
- Molecules

$g = g_{max} \cdot t/\tau_p \cdot \exp(1-t/\tau_p)$

$g = g_{max} \cdot m^x \cdot h^y$

$1\text{-}m \underset{\beta(V)}{\overset{\alpha(V)}{\rightleftharpoons}} m$

$$\tau_m \frac{\partial V}{\partial t} = E - V + \lambda^2 \frac{\partial^2 V}{\partial x^2}$$

$E = RT/zF \cdot \ln([out]/[in])$

$$\frac{\partial \phi}{\partial t} = D\nabla^2 \phi(\vec{r}, t)$$

A + B <===> C

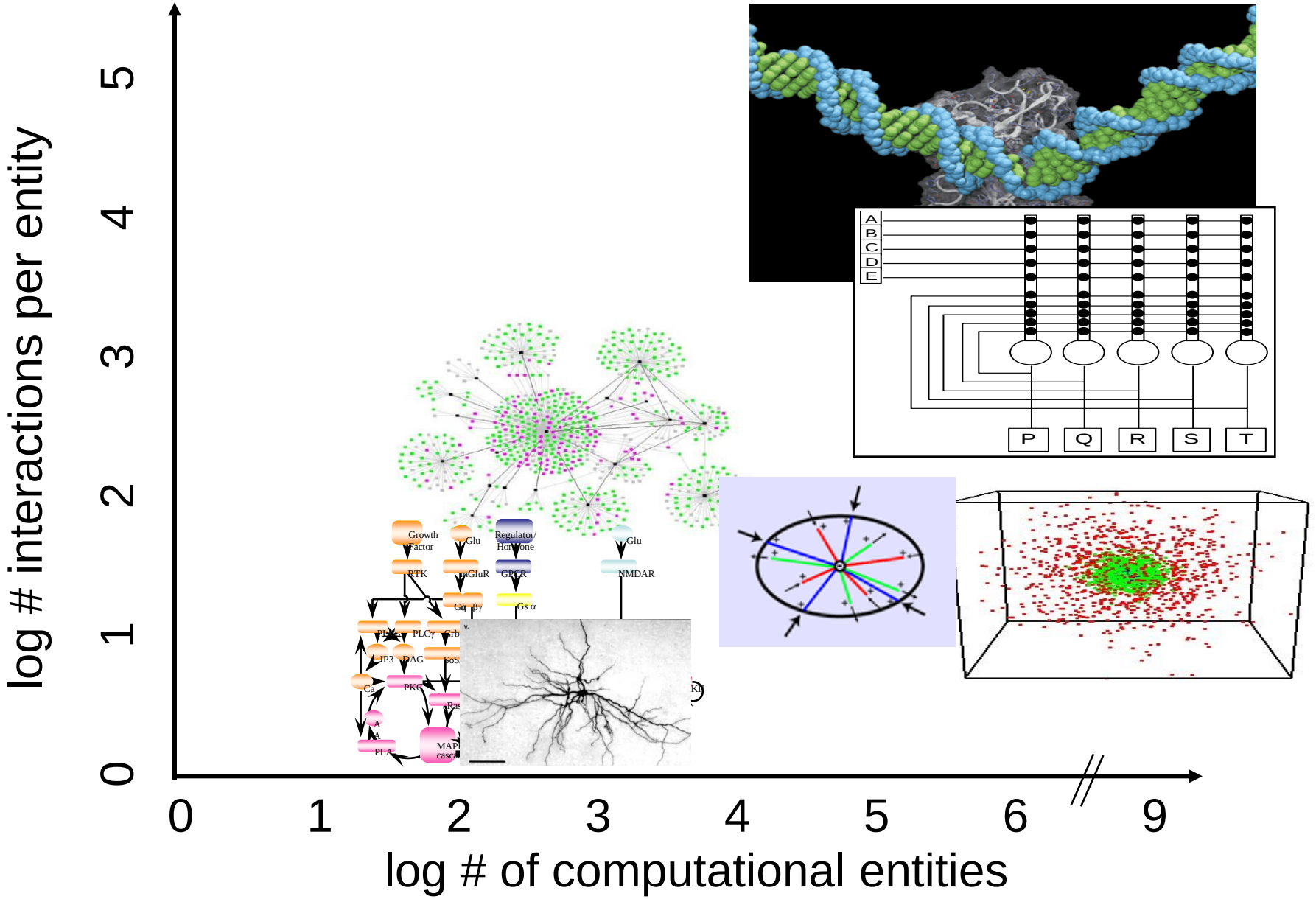
dA/dt = – kf.A.B + kb.C

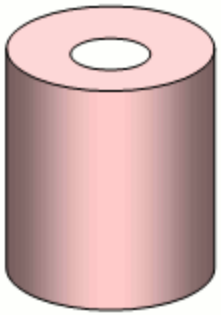
Stochastic forms

Brownian motion

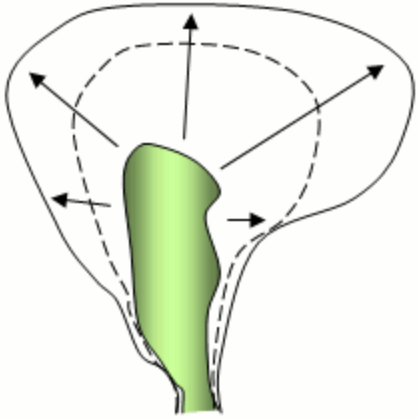
Mechanics:
Tensegrity
Bending moments
Motors
Bulk flow

Problem sizes

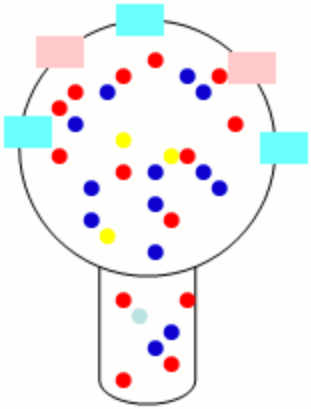




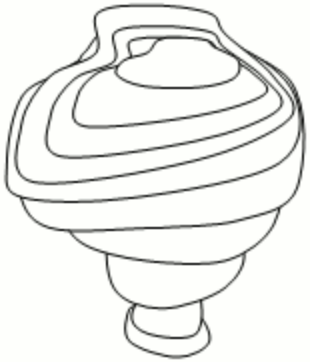
ChannelML



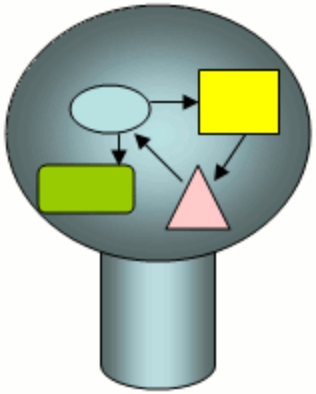
MechML



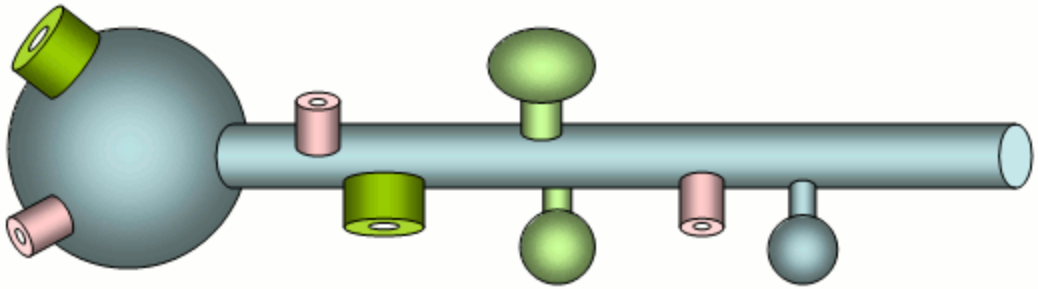
3DMCML



MeshML

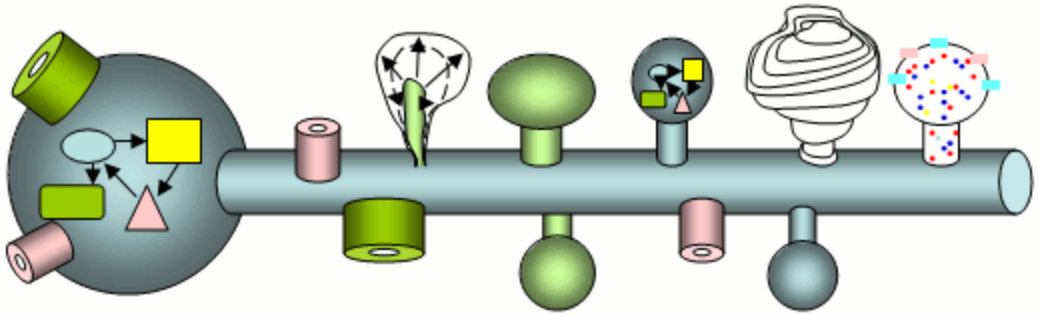


SBML



NeuroML

SEDML



Composition
Math
How to link
How to solve

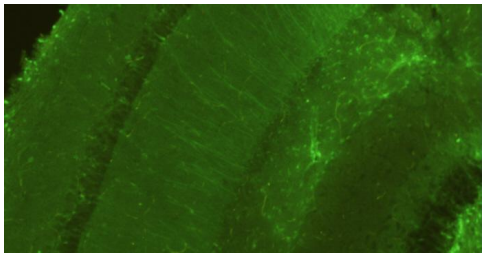
A typical multiscale model

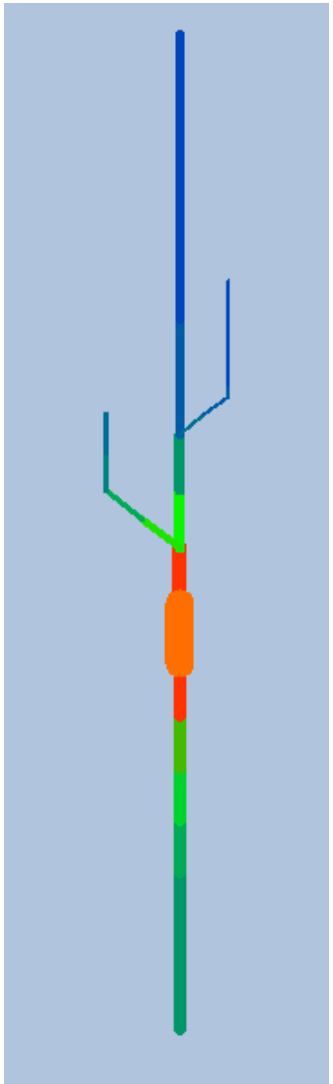
Network

Cell

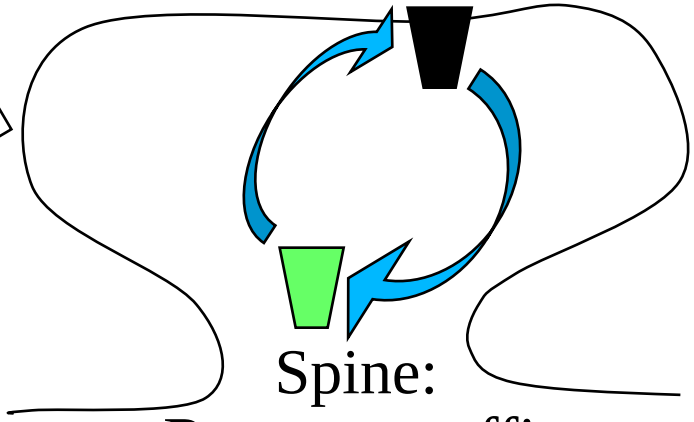
Biophysics

Molecules





G_{AMPAR}



Spine:
Receptor traffic

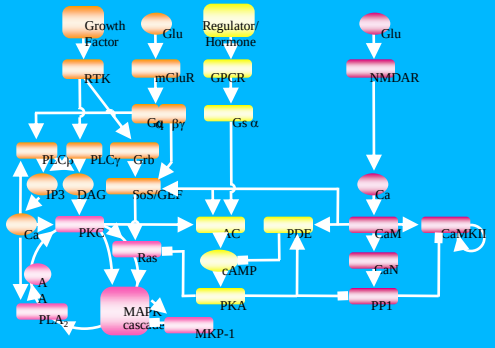
Network activity

Electrode stimulus

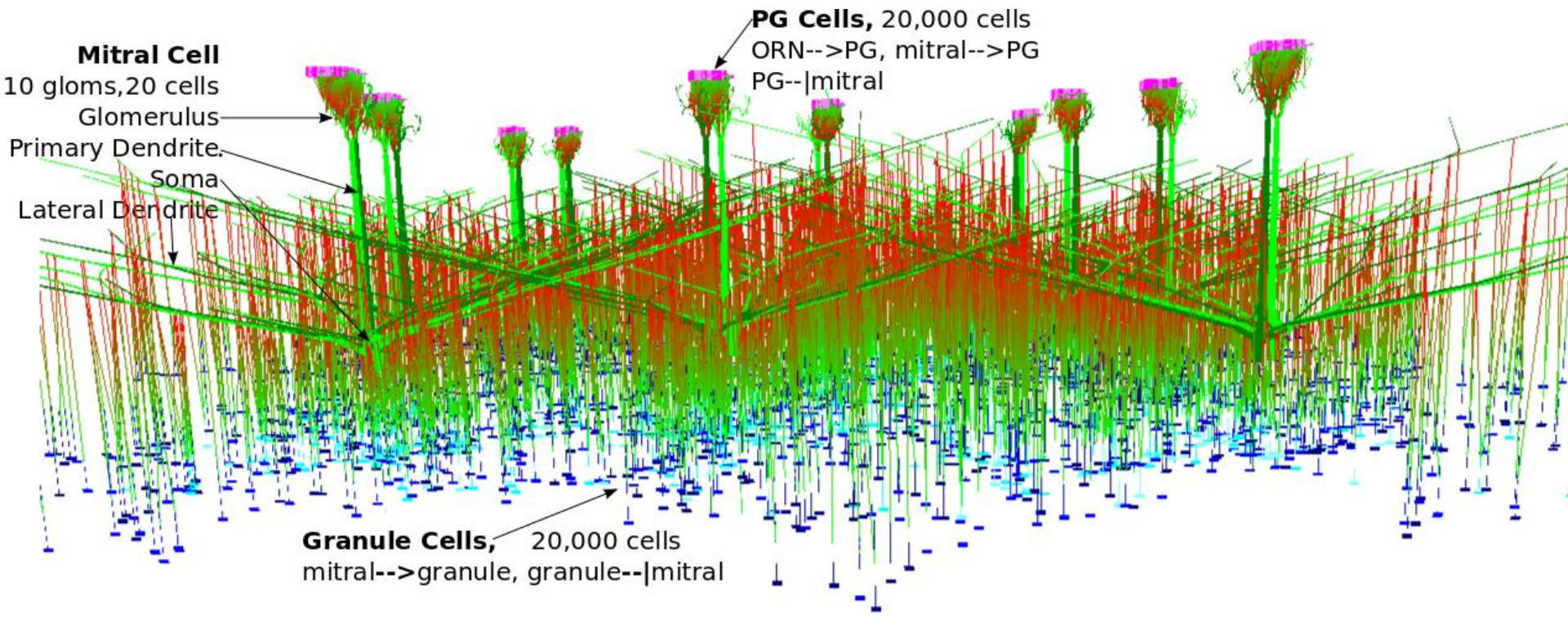
Ca^{2+}

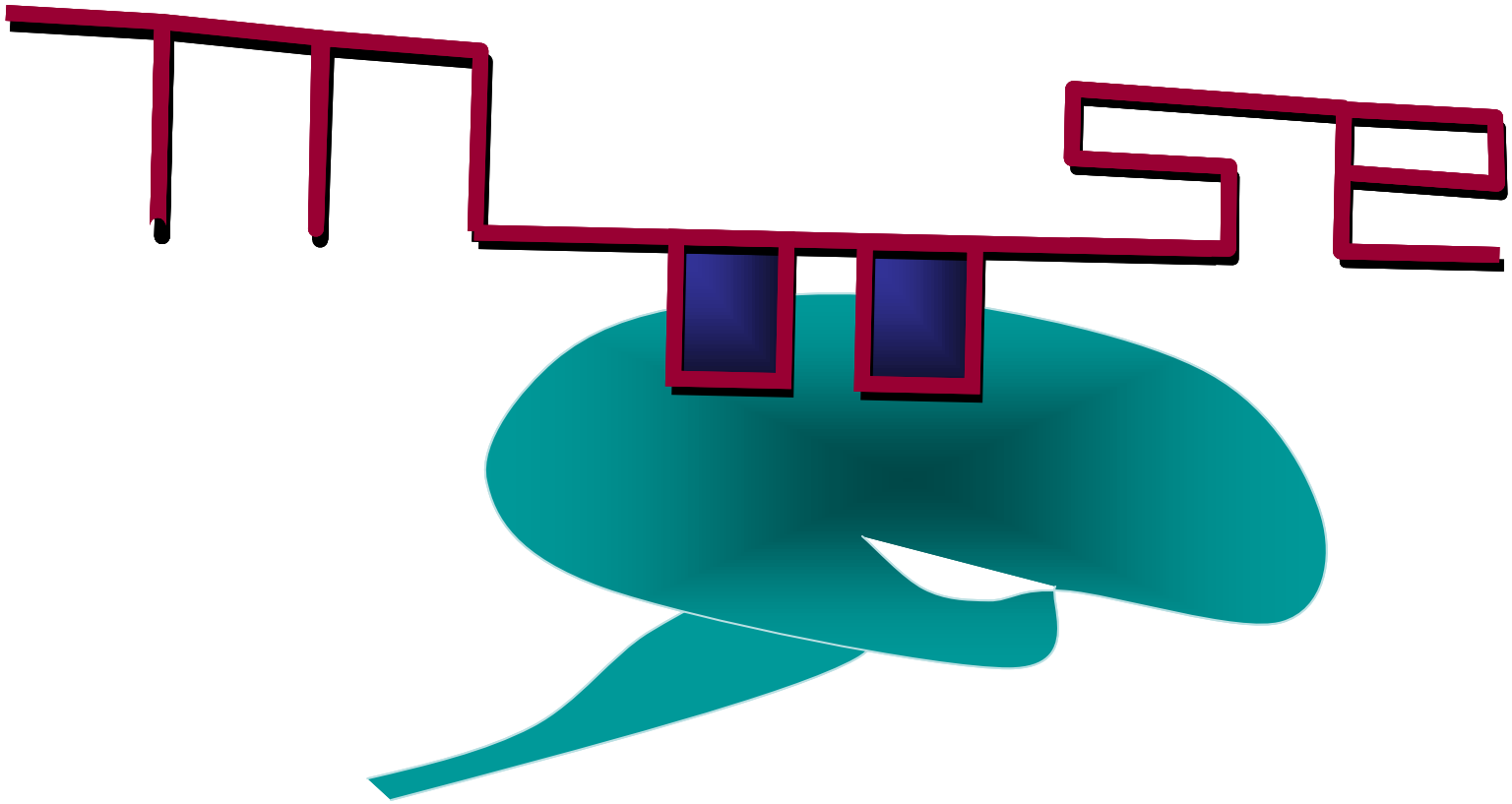
G_{KA}

Dendrite: MAPK feedback



Biophysically-detailed network model





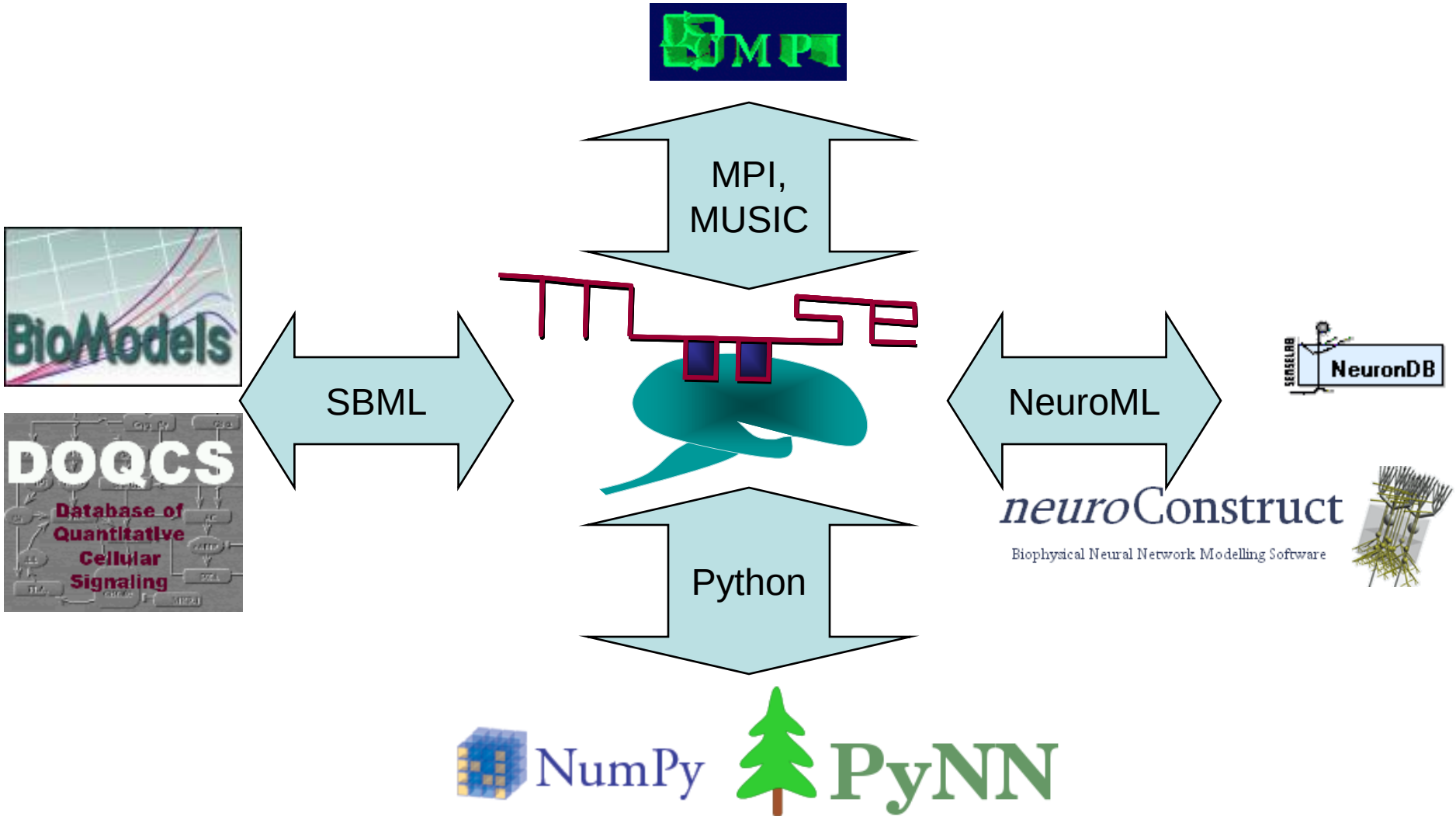
The Multiscale Object-Oriented Simulation Environment

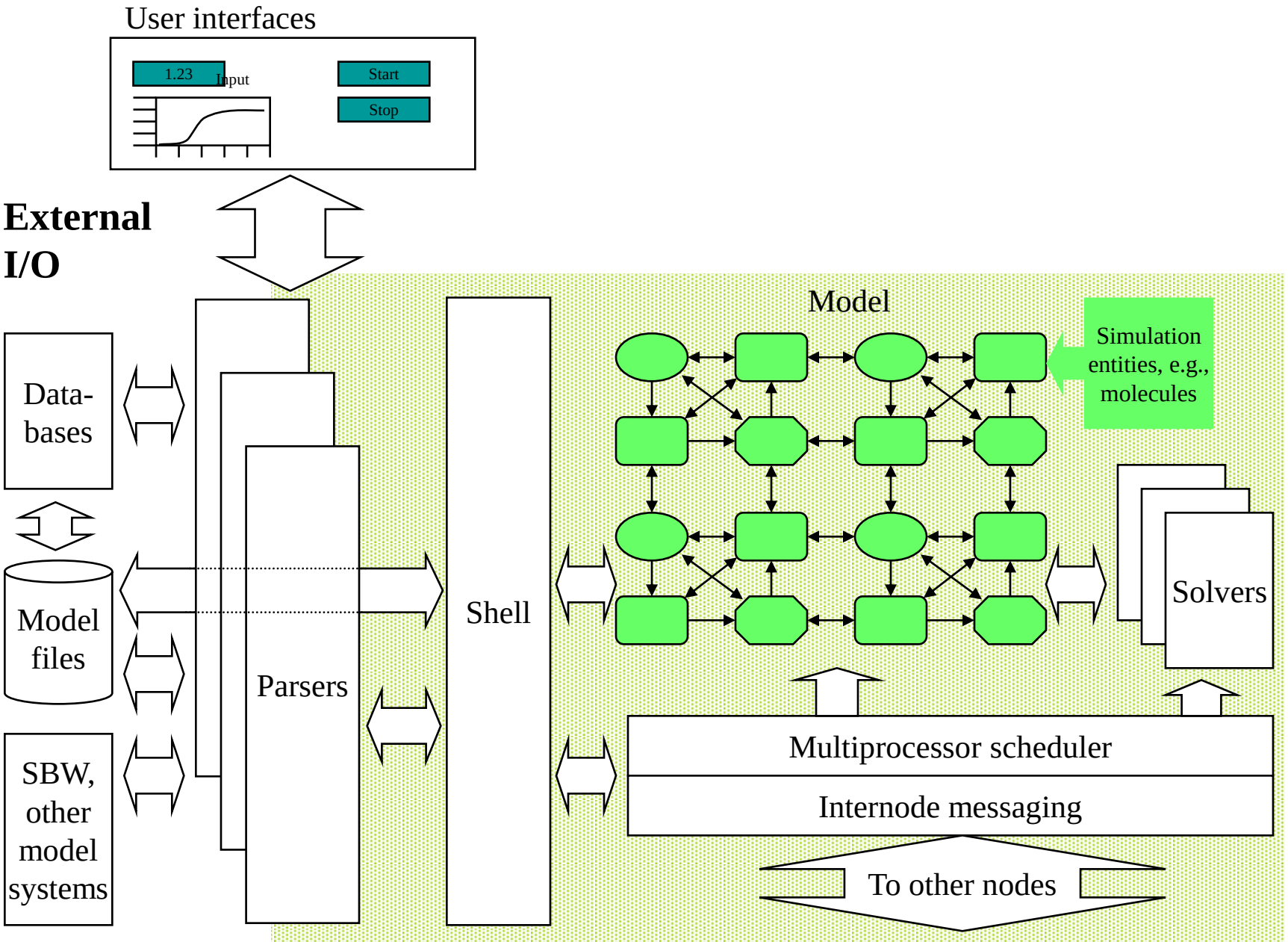
http://moose.ncbs.res.in,
http://sourceforge.net/projects/moose/

LGPL but uses GPL modules

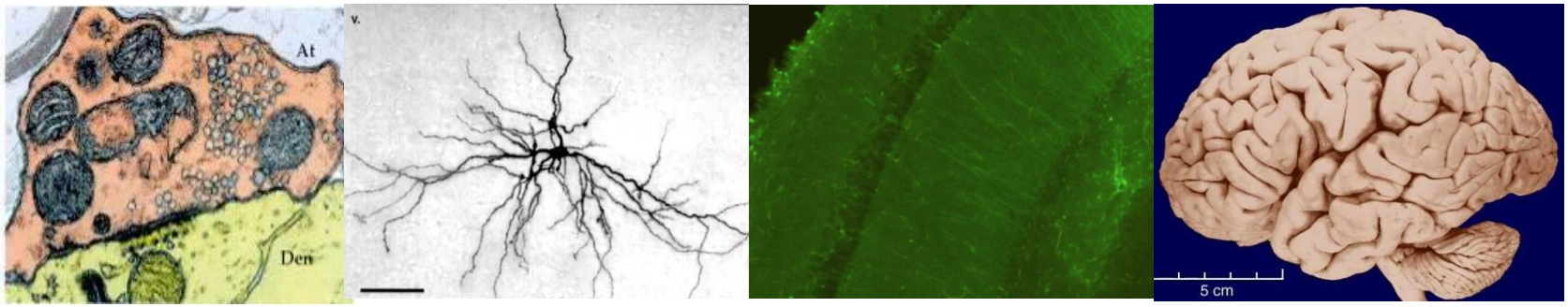
C++/Python/Qt/GSL/OpenGL and many many more…

MOOSE supports standards




User interfaces
1.23
Input
Start
Stop
External I/O
Data-bases
Model files
SBW, other model systems
Parsers
Shell
Model
Simulation entities, e.g., molecules
Solvers
Multiprocessor scheduler
Internode messaging
To other nodes

Modelling across scales



Single particle (Smoldyn plugin)

Compartmental ODE and stochastic Steady-state solver

Cellular biophysics

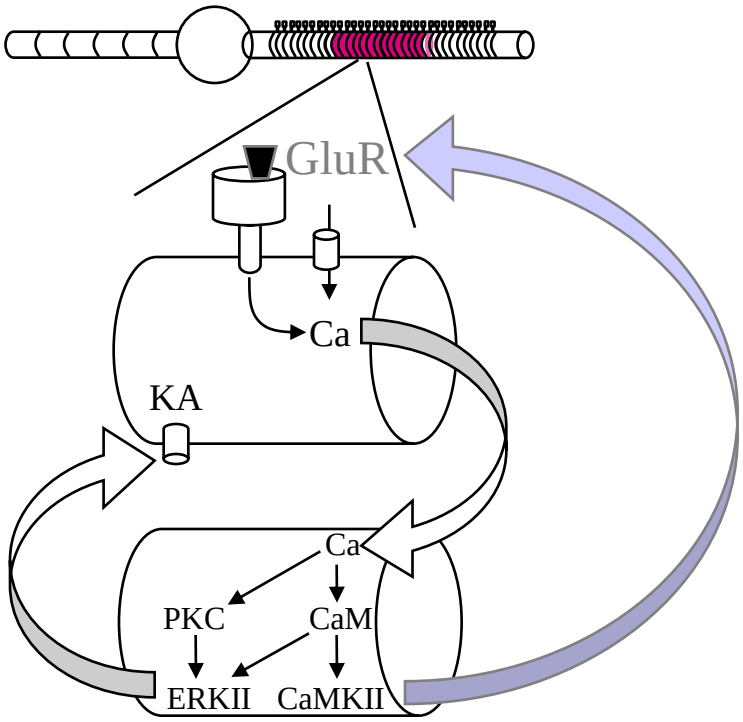
Network

Multiscale: biophysical network

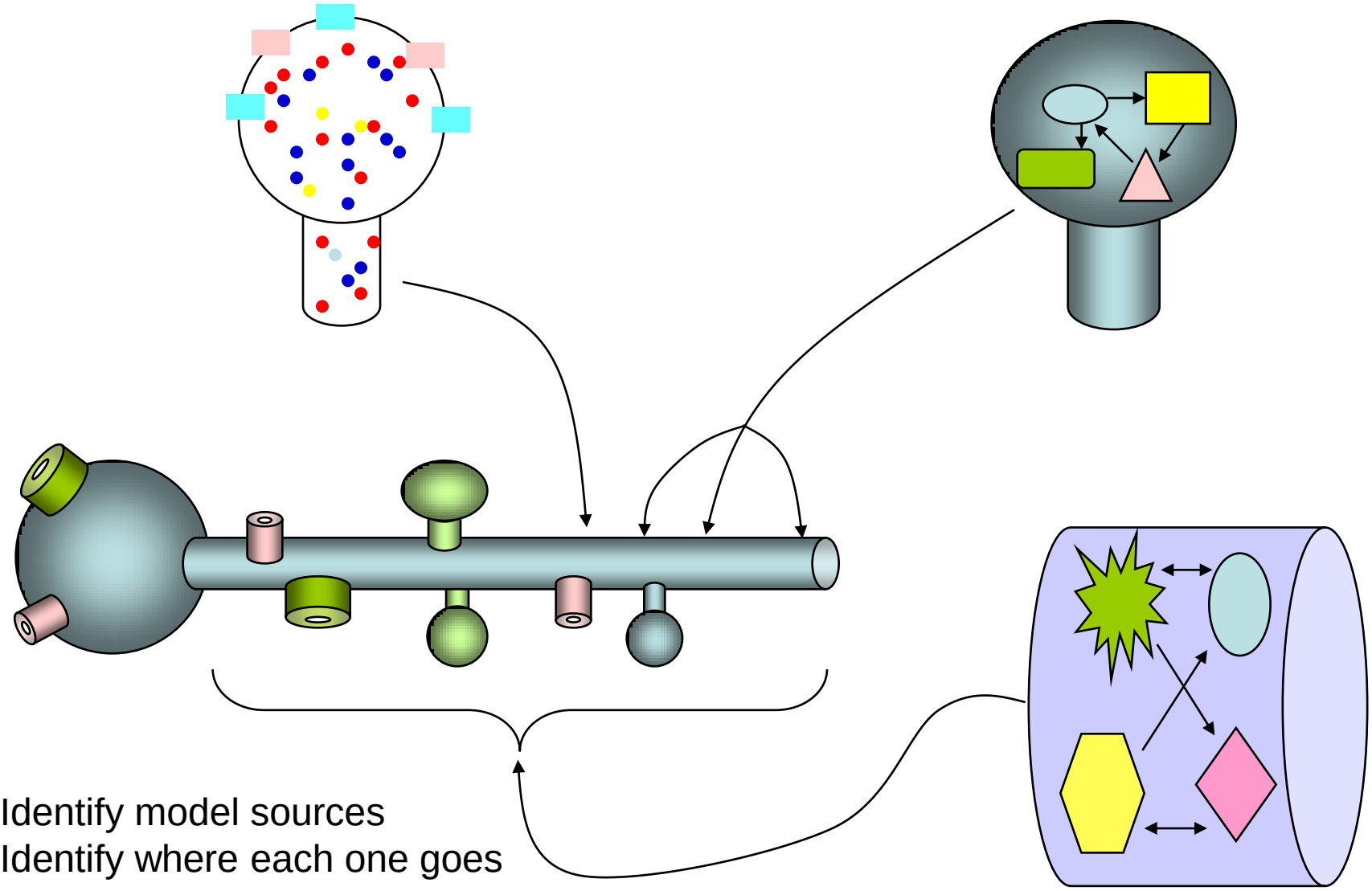
Multiscale: SigNeur

Multiscale language requirements

- Composition specification:
  - Single NeuroML model
  - Multiple SBML models
    - Diffusion specification
    - Entity mapping
- Interface specification
  - Molecules -> channel properties
  - Ion flux (Ca) -> Signaling effects
  - Synaptic input -> Ligand molecules
  - Molecular gradients/junctions
- Geometry specification
  - Spines
  - Junctions
  - Caps
  - Mapping to detailed morphology
  - Spatial transforms and writhing
- Mechanics specification
- Control specifications:
  - Solvers
  - Runtime
  - Output variables

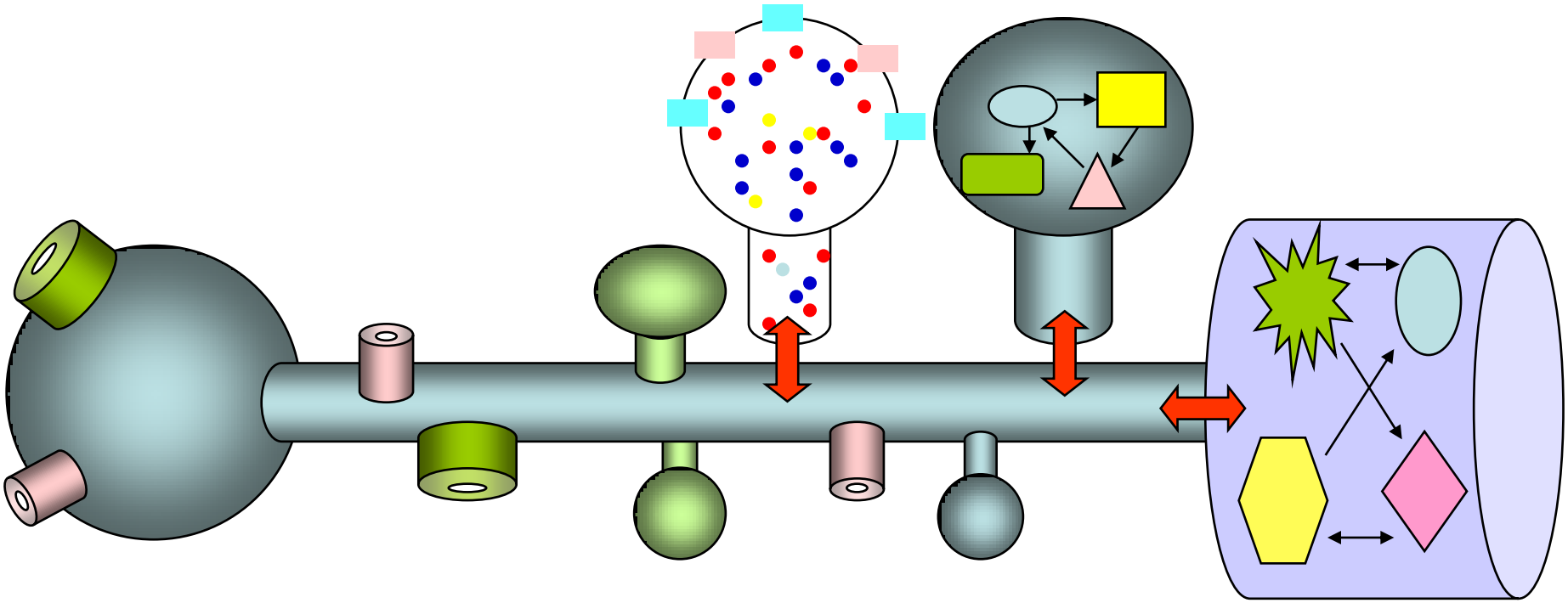


1. Composition specification



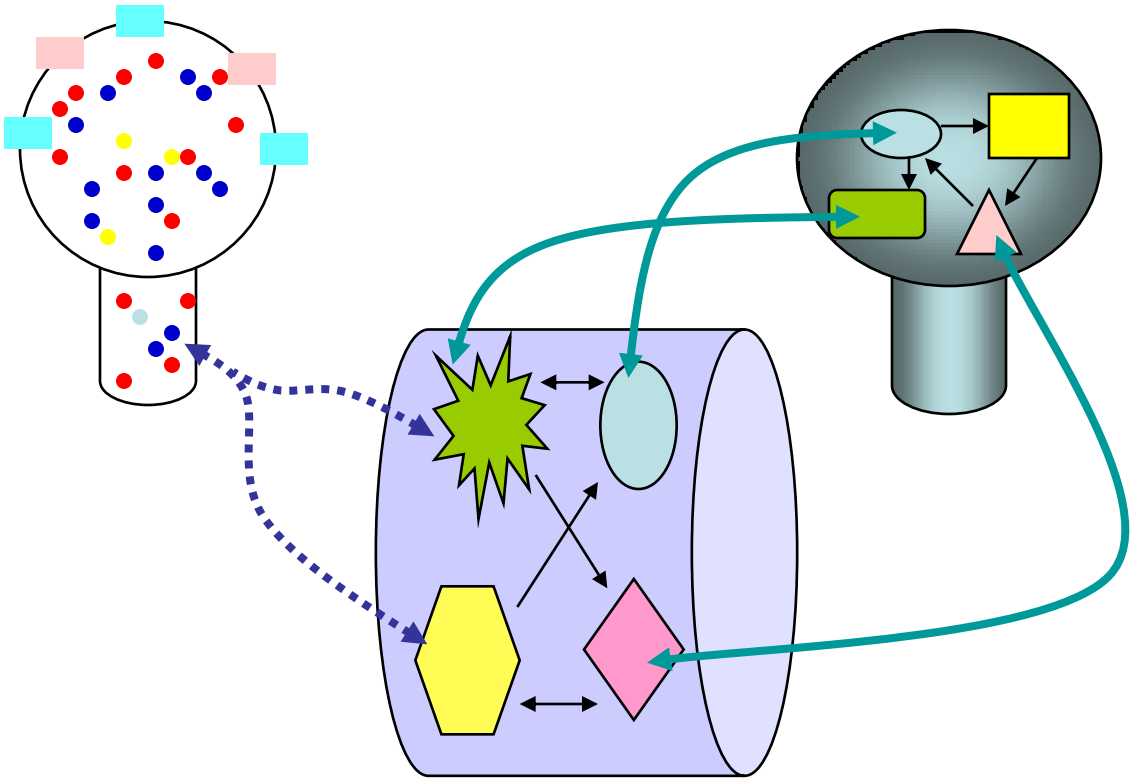
a. Identify model sources
b. Identify where each one goes

1. Composition specification

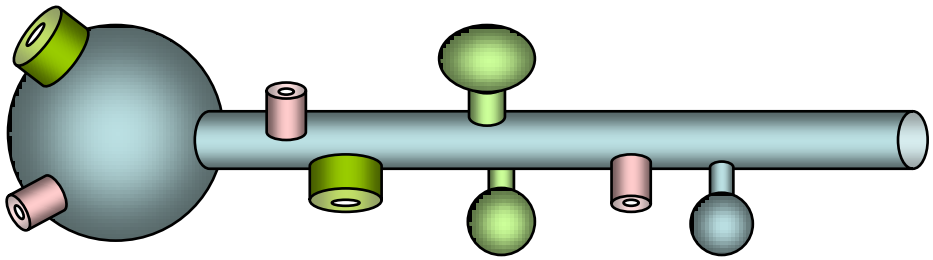


a. Identify model sources
b. Identify where each one goes
c. Diffusion along compartments
d. Diffusion into spines

1. Composition specification

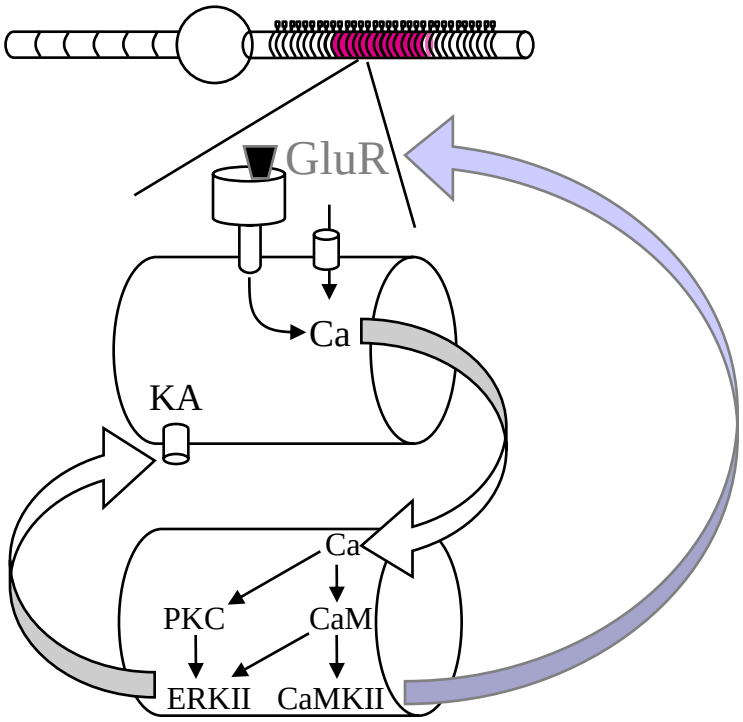


a. Identify model sources
b. Identify where each one goes
c. Diffusion along compartments
d. Diffusion into spines
e. Entity mapping: Unique
f. Entity mapping: complex



2. Interface specification

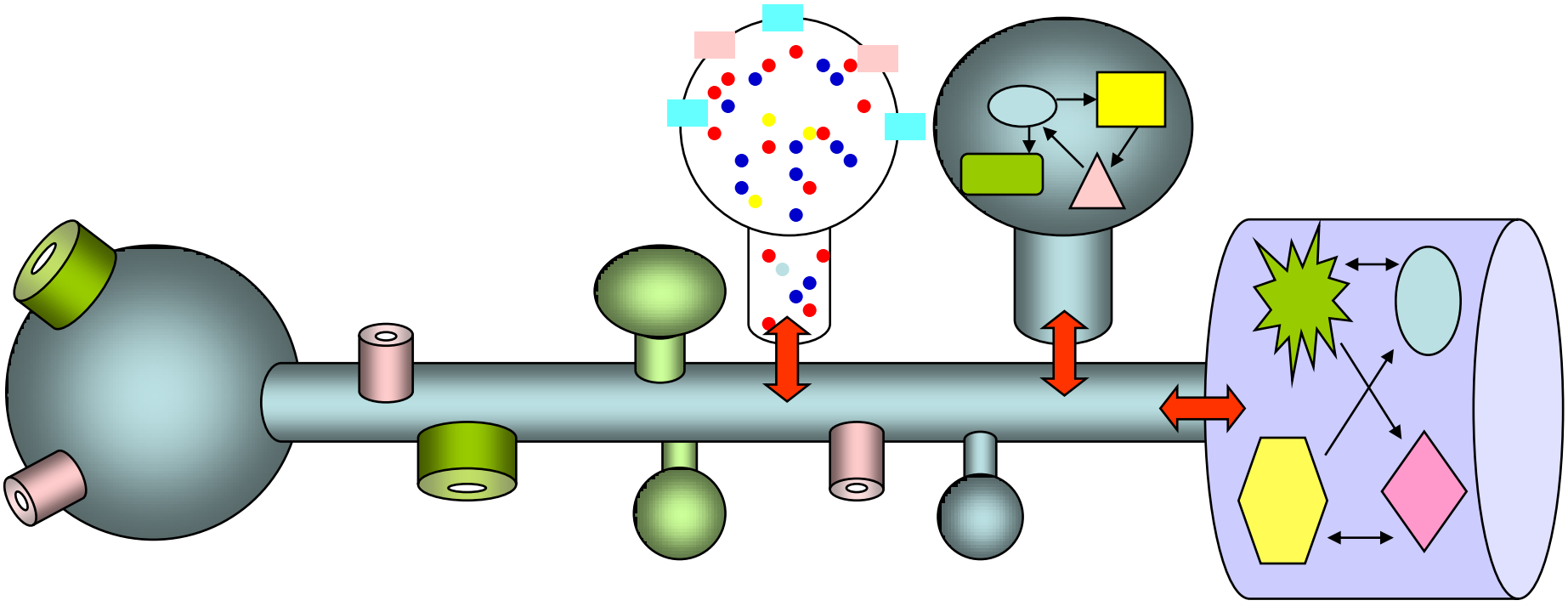
a. Molecules->Channel conductance
b. Molecules-> Channel kinetics
c. Ion flux (Ca) -> Signaling effects
d. Synaptic input -> Ligand molecules



2. Interface specification

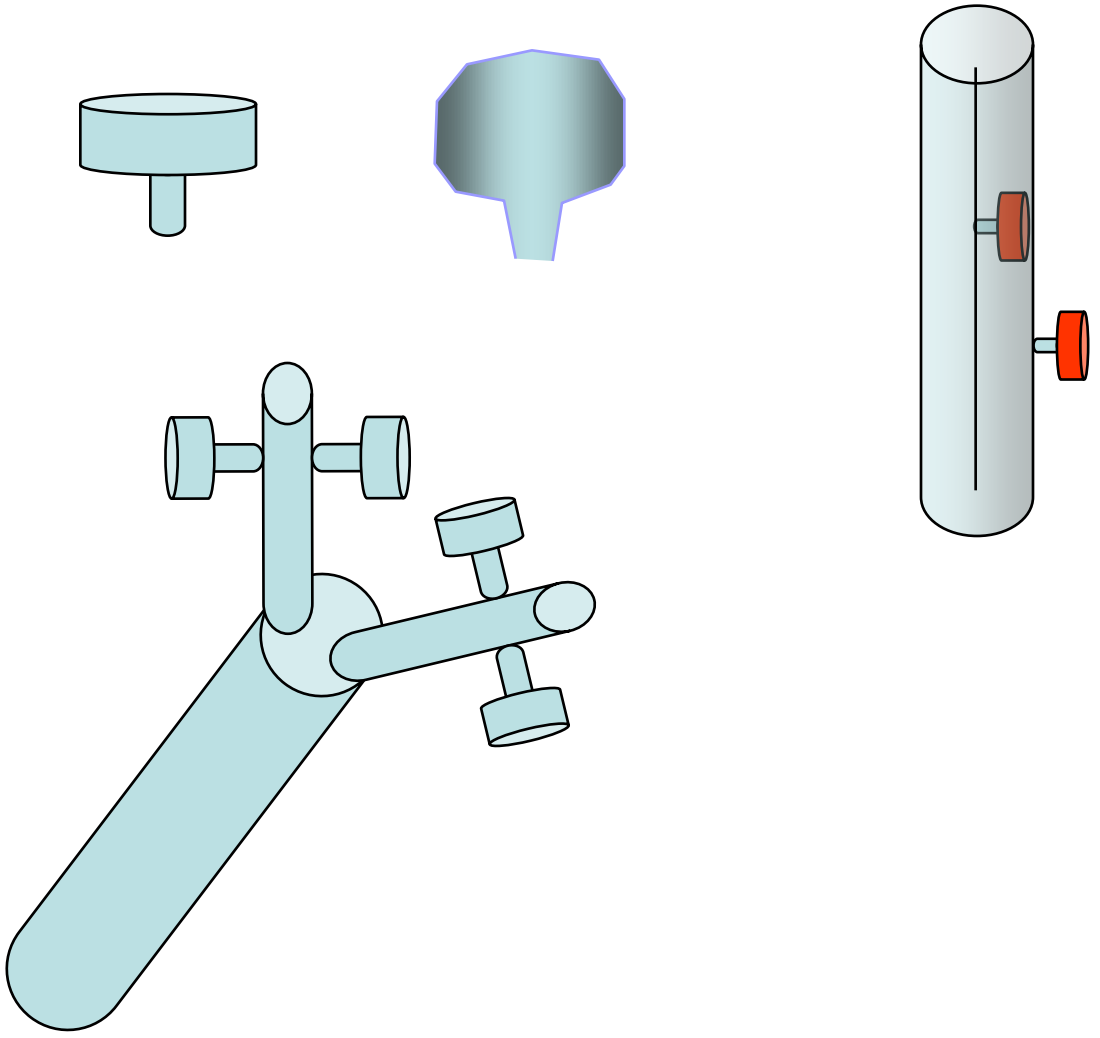
a. Molecules->Channel conductance
b. Molecules-> Channel kinetics
c. Ion flux (Ca) -> Signaling effects
d. Synaptic input -> Ligand molecules
e. Molecular gradients/junctions

All involve scaling, offset, unit conversion and so on.



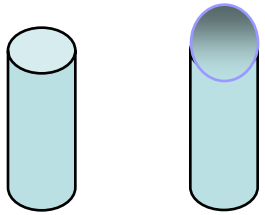
Geometry specification

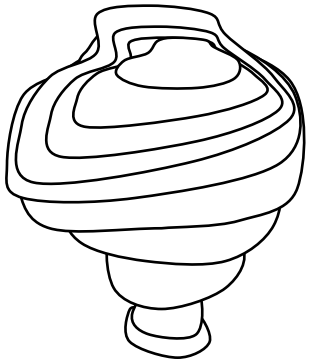
a. Spines
b. Junctions

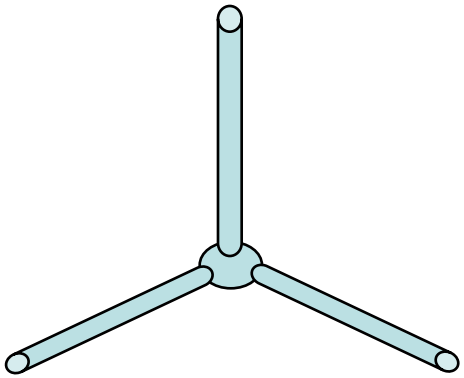


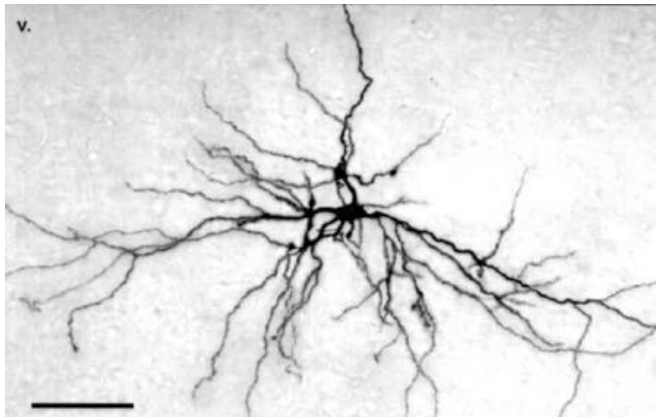
Geometry specification

a. Spines
b. Junctions
c. Caps
d. Detailed morphology: meshes
e. Detailed morphology: mappings



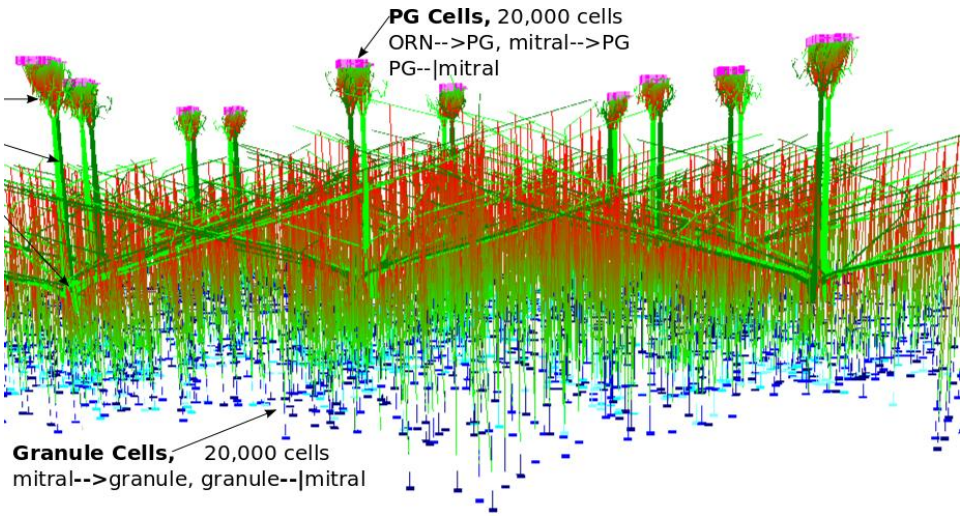


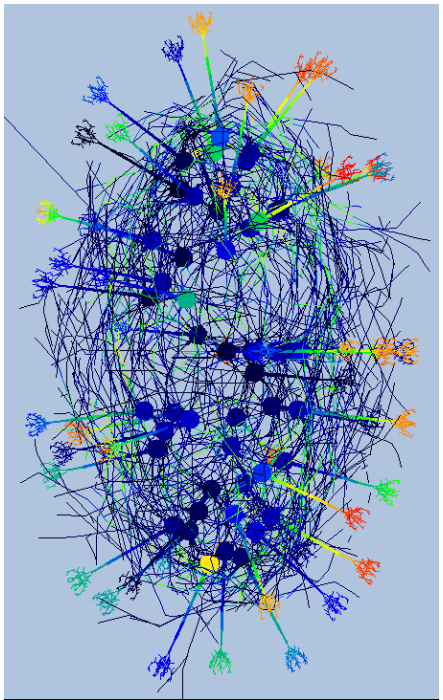


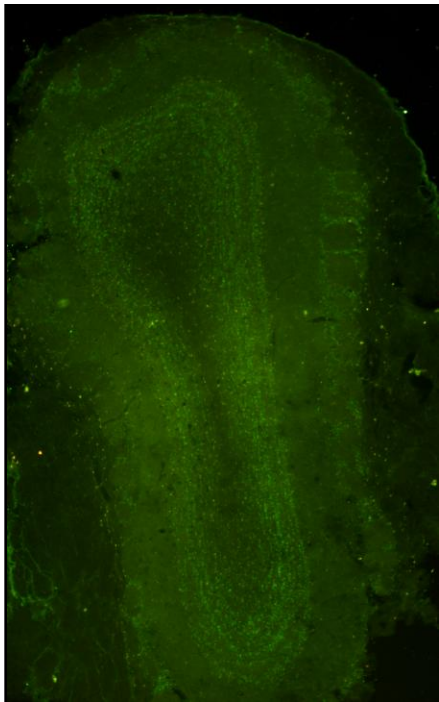


Geometry specification

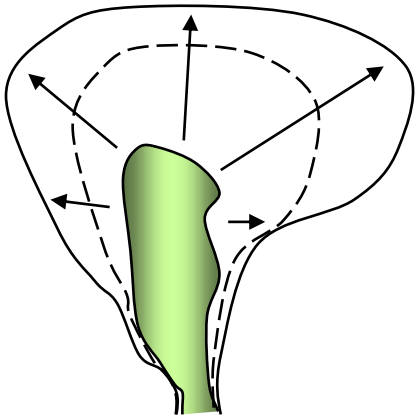
a. Spines
b. Junctions
c. Caps
d. Detailed morphology: meshes
e. Detailed morphology: mappings
f. Spatial transforms
g. Writing

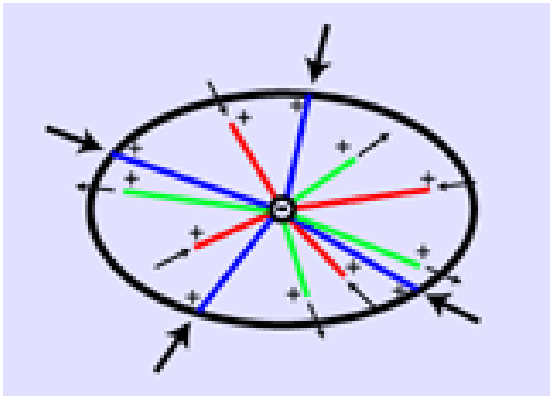






4. Mechanics specification





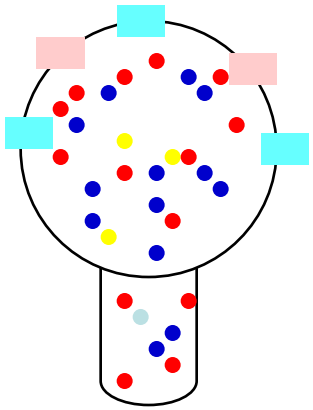
5. Control

a. Solvers

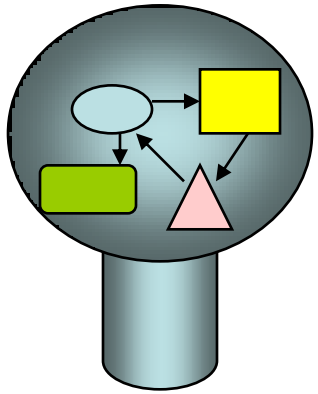
b. Runtime control

c. Output variables and formats

SEDML



Vs.



Multiscale language requirements

- Composition specification:
  - Single NeuroML model
  - Multiple SBML models
    - Diffusion specification
    - Entity mapping
- Interface specification
  - Molecules -> channel properties
  - Ion flux (Ca) -> Signaling effects
  - Synaptic input -> Ligand molecules
  - Molecular gradients/junctions
- Geometry specification
  - Spines
  - Junctions
  - Caps
  - Mapping to detailed morphology
  - Spatial transforms and writhing
- Mechanics specification
- Control specifications:
  - Solvers
  - Runtime
  - Output variables

Multiscale language requirements

- Composition specification:
  - Single NeuroML model
  - Multiple SBML models
    - Diffusion specification
    - Entity mapping
- Interface specification
  - Molecules -> channel properties
  - Ion flux (Ca) -> Signaling effects
  - Synaptic input -> Ligand molecules
  - Molecular gradients/junctions
- Geometry specification
  - Spines
  - Junctions
  - Caps
  - Mapping to detailed morphology
  - Spatial transforms and writhing
- Mechanics specification
- Control specifications:
  - Solvers
  - Runtime
  - Output variables

Ease of use: Sensible defaults

Options

- Will SEDML handle arbitrary model control?
- NeuroML vs composition ML?
- Problem-specific compositionML, or are there generic ways to assemble MLs?
- Forget the compositionML.
  - Use Python or script, local to simulator
  - Push for modularity in other MLs.

Thank You

Siji,
Harsha,
Aditya,
Niraj,
Oliver
Subhasis,
[Karan,
Pragati,
Arnold]

INCF
SBCNY
NCBS/TIFR
DAE
DBT
EU-India Grid