Rule-Based Kinetic Modeling of Signal Transduction Networks

Part II. Tutorial

Jim Faeder, Michael Blinov, William Hlavacek

Center for Cell Analysis and Modeling, University of Connecticut Health Center

Theoretical Biology and Biophysics Group, Los

bionetgen.lanl.gov

Leigh Fanning
Matthew Fricke
Jeremy Kozdon
Nathan Lemons
Michael Blinov
James Faeder
William Hlavacek
Byron Goldstein*

Ambarish Nag
Michael Monine

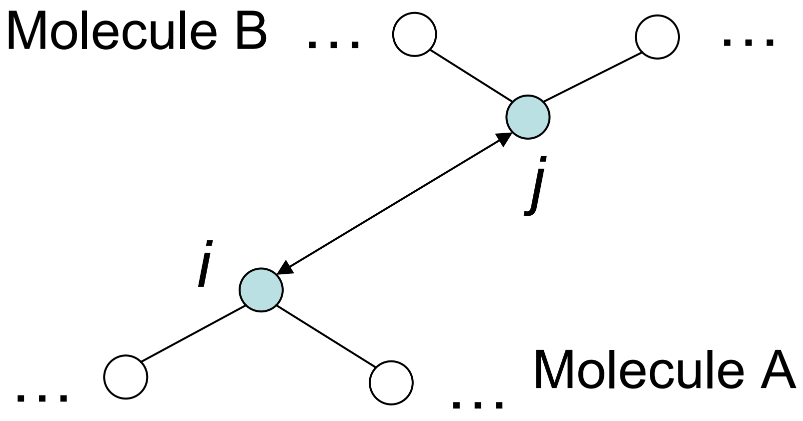
http://bionetgen.lanl.gov

Funding
NIH
DOE
LANL-LDRD

BioNetGen and BioNetGen Language (BNGL)

A site-based formulation of chemical kinetics

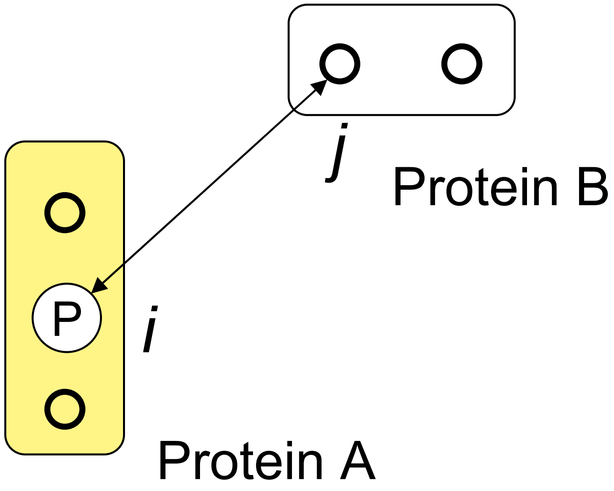
Molecular Dynamics (MD)



Force between *i* and *j*

$$m\ddot{x}_i = F(|x_i - x_j|)$$

Chemical Kinetics

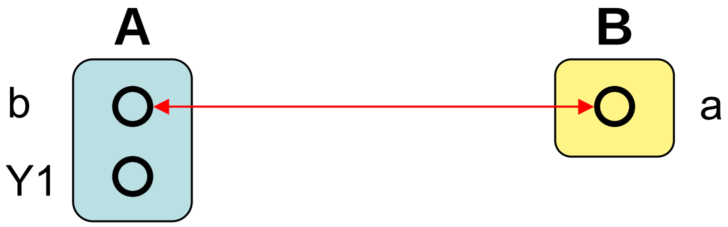


Reaction probability between *i* and *j*

$$p_A(t, t+\Delta t) = p_{i \text{ reacts with } j}(|x_i - x_j|)$$

BioNetGen provides explicit representation of molecular components and interactions

Molecules

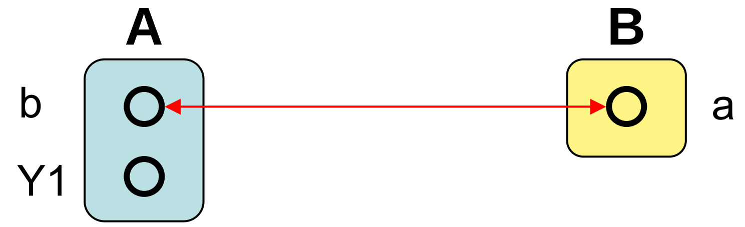


`A(b,Y1)`

`B(a)`

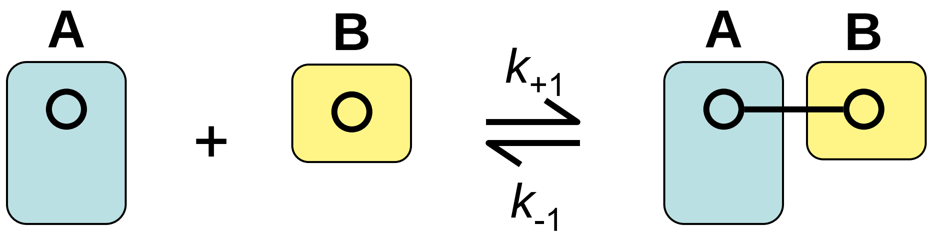
BioNetGen provides explicit representation of molecular components and interactions

Molecules



A(b,Y1)　　　　B(a)

Rule



```
A(b) + B(a) <-> A(b!1).B(a!1) kp1,km1
```

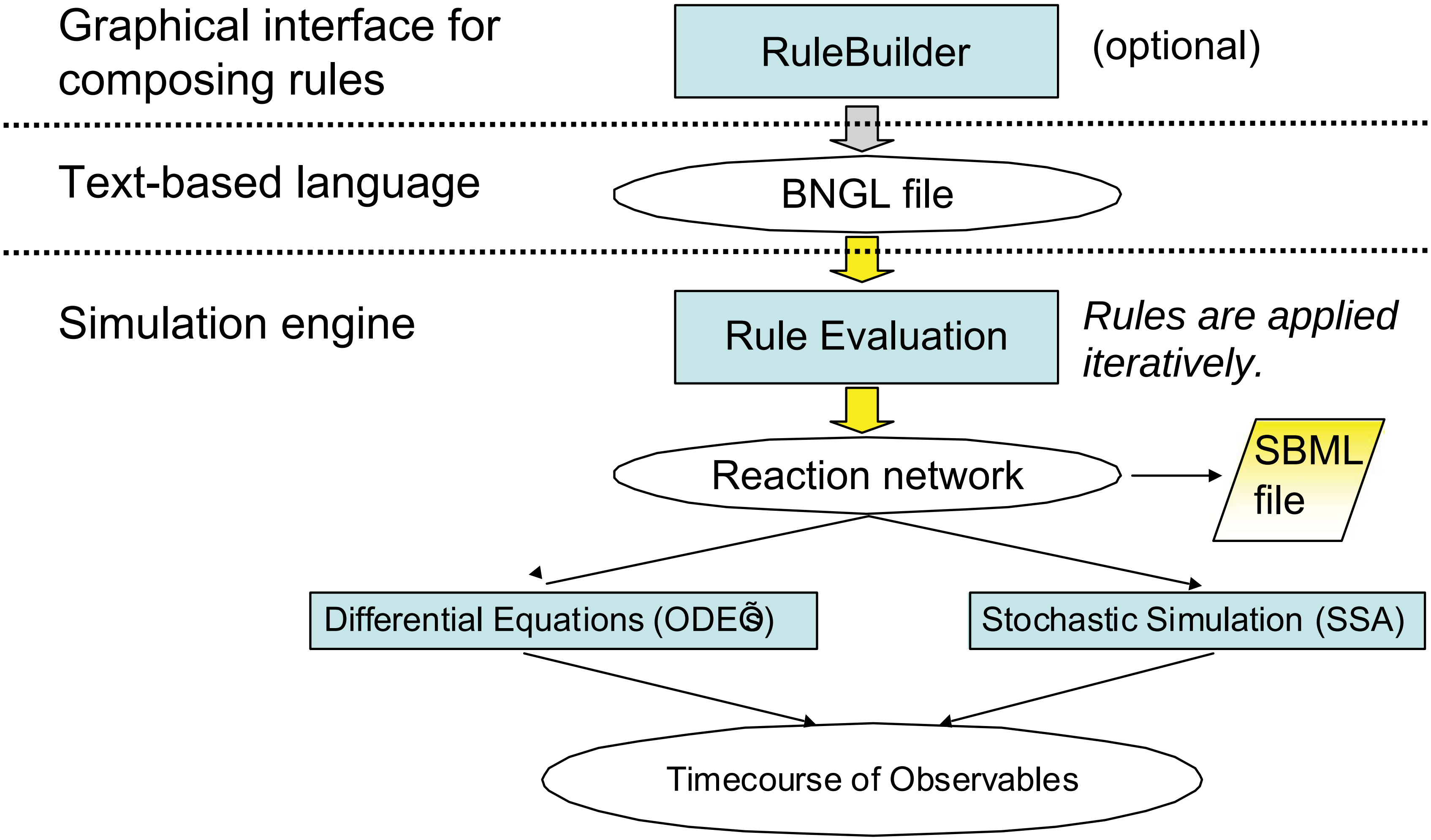
a bond between two components

Advantages of BNGL

- Enables construction of precise and flexible models
- Common format can be created and processed by multiple applications
- Forms basis for our proposal to extend SBML, already a common exchange format used by modelers
- Rules can be embedded in databases, wiki's, and papers
- Molecules and rules are reusable
- Molecule and rule definition could be automated using databases of protein-protein interactions as a source

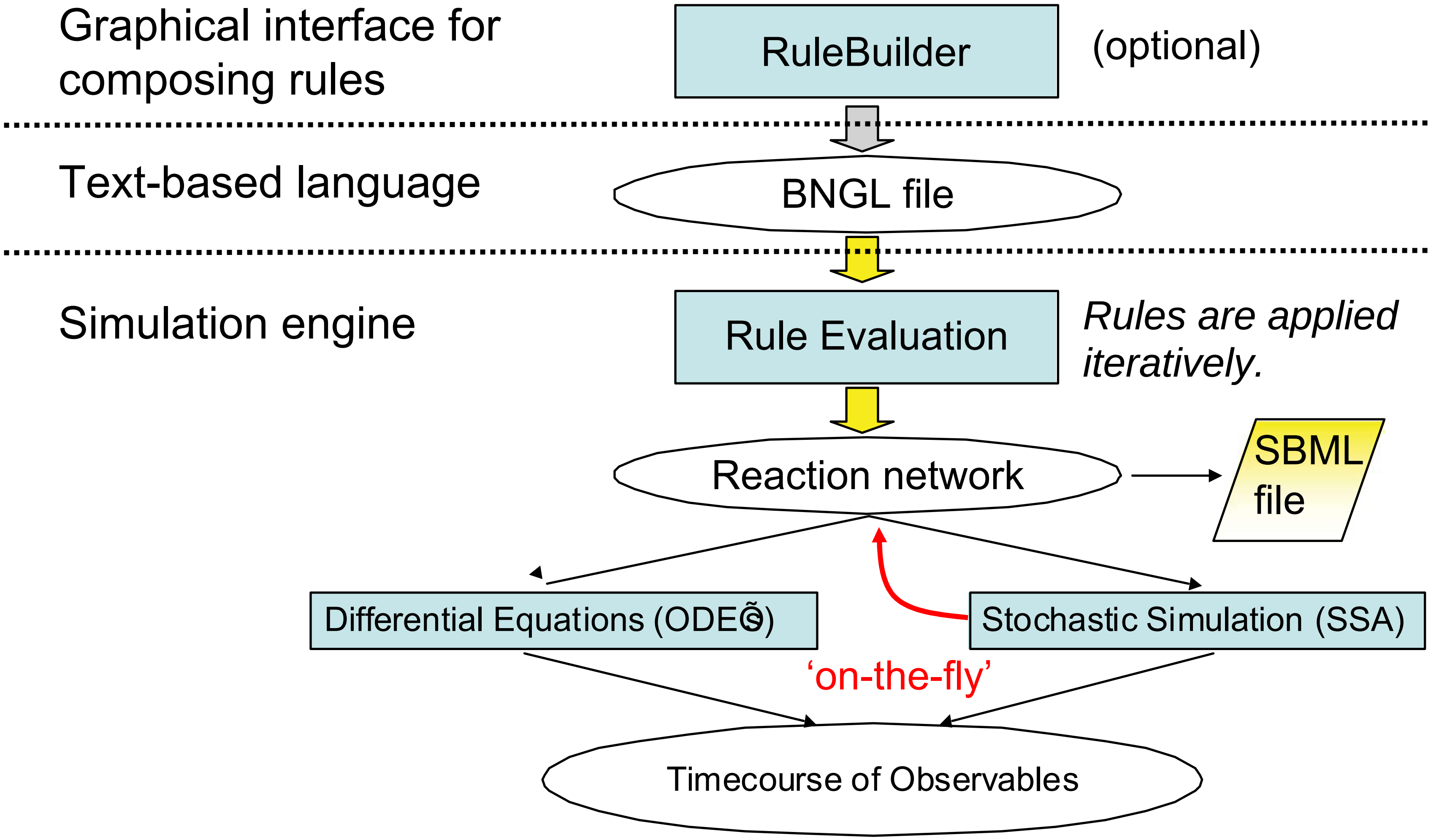
Hlavacek et al. (2006) *Science STKE*, **2006**, re6.

BioNetGen2: Software for graphical rule-based modeling


Graphical interface for composing rules
RuleBuilder
(optional)
Text-based language
BNGL file
Simulation engine
Rule Evaluation
Rules are applied iteratively.
Reaction network
SBML file
Differential Equations (ODEš)
Stochastic Simulation (SSA)
Timecourse of Observables

http://bionetgen.lanl.gov

BioNetGen2: Software for graphical rule-based modeling



Graphical interface for composing rules

RuleBuilder

(optional)

Text-based language

BNGL file

Simulation engine

Rule Evaluation

Rules are applied iteratively.

Reaction network

SBML file

Differential Equations (ODEs̃)

Stochastic Simulation (SSA)

‘on-the-fly’

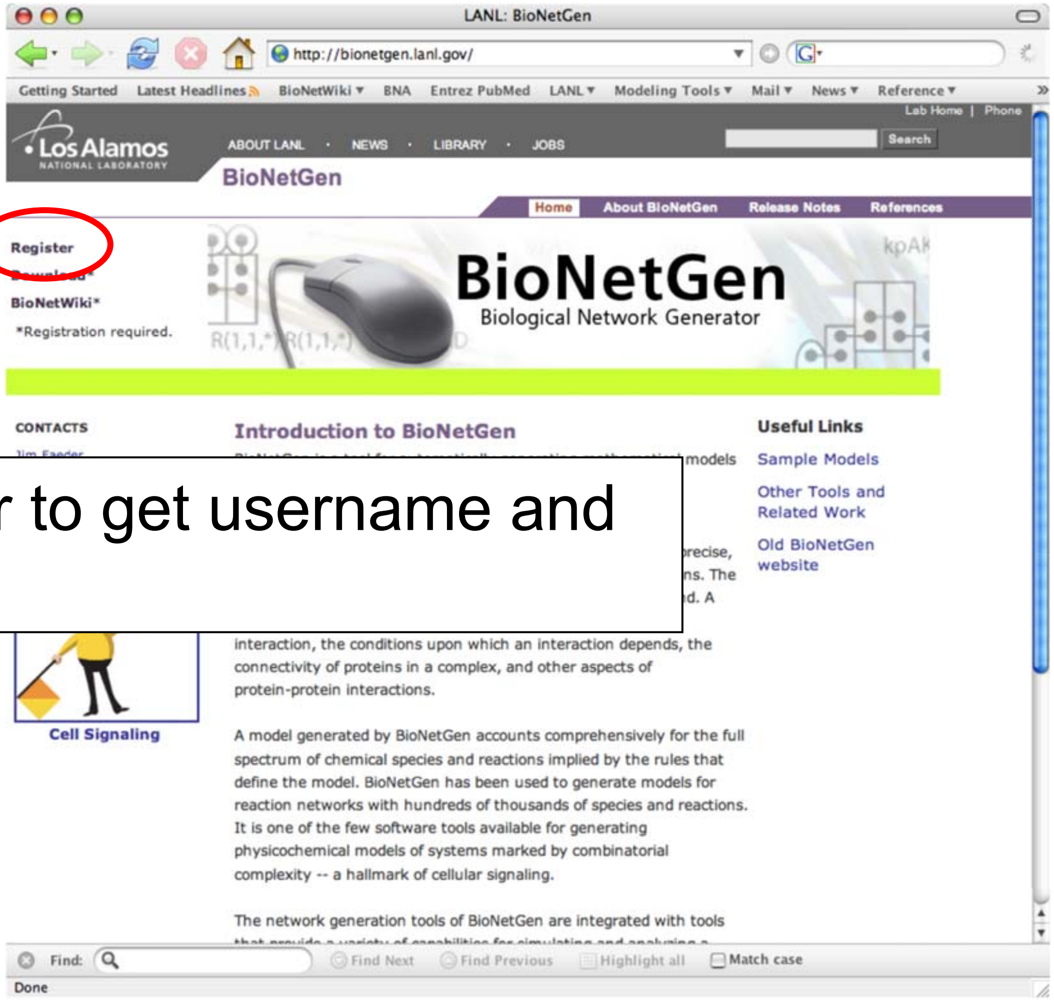
Timecourse of Observables

http://bionetgen.lanl.gov

The BioNetGen Website

bionetgen.lanl.gov

BioNetGen is *free, open-source* software.



Introduction to BioNetGen

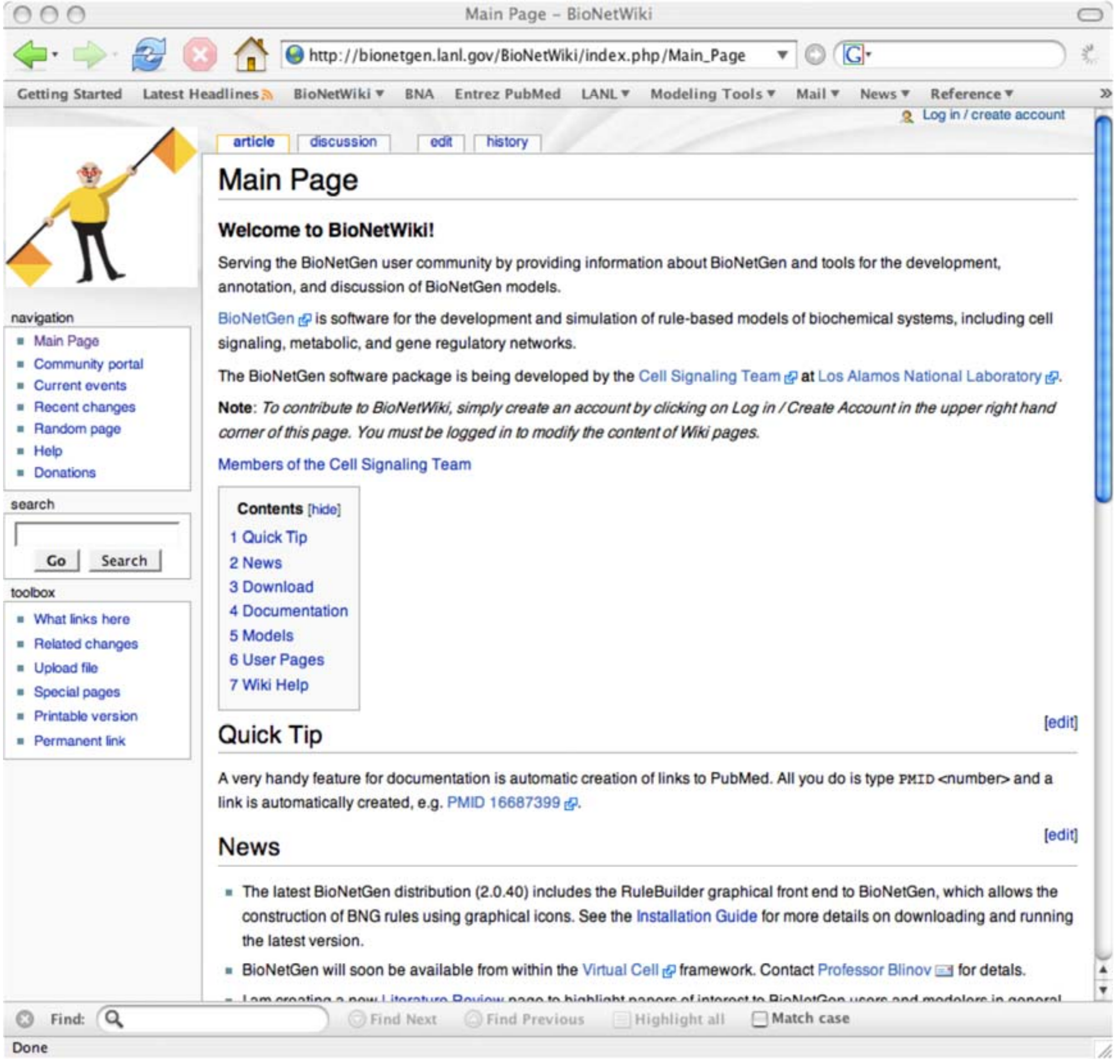
interaction, the conditions upon which an interaction depends, the connectivity of proteins in a complex, and other aspects of protein-protein interactions.

A model generated by BioNetGen accounts comprehensively for the full spectrum of chemical species and reactions implied by the rules that define the model. BioNetGen has been used to generate models for reaction networks with hundreds of thousands of species and reactions. It is one of the few software tools available for generating physicochemical models of systems marked by combinatorial complexity -- a hallmark of cellular signaling.

The network generation tools of BioNetGen are integrated with tools

Step 1: Register to get username and password.

BioNetWiki



Main Page

Welcome to BioNetWiki!

Serving the BioNetGen user community by providing information about BioNetGen and tools for the development, annotation, and discussion of BioNetGen models.

BioNetGen is software for the development and simulation of rule-based models of biochemical systems, including cell signaling, metabolic, and gene regulatory networks.

The BioNetGen software package is being developed by the Cell Signaling Team at Los Alamos National Laboratory.

Note: *To contribute to BioNetWiki, simply create an account by clicking on Log in / Create Account in the upper right hand corner of this page. You must be logged in to modify the content of Wiki pages.*

Members of the Cell Signaling Team

Contents [hide]

1 Quick Tip
2 News
3 Download
4 Documentation
5 Models
6 User Pages
7 Wiki Help

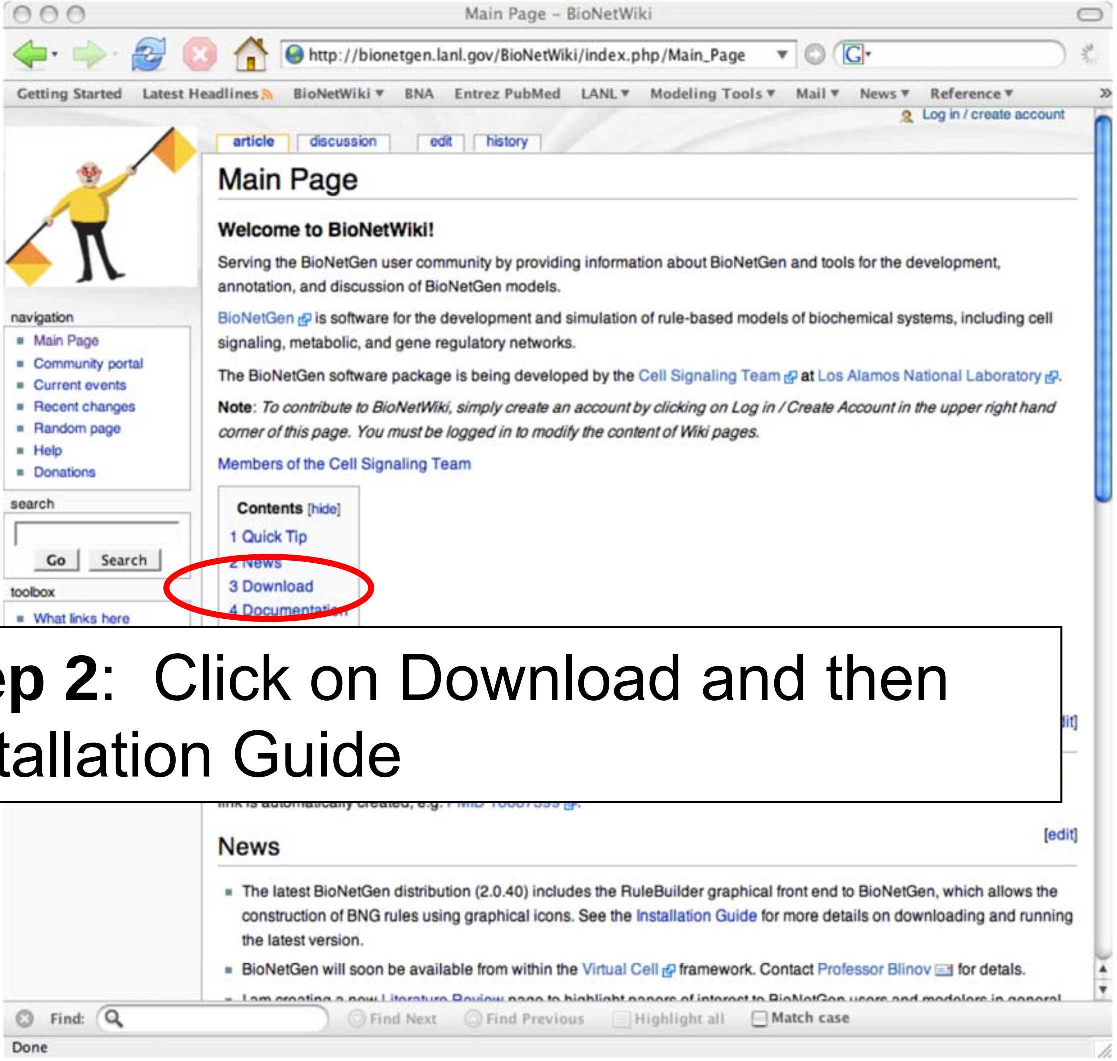
Quick Tip [edit]

A very handy feature for documentation is automatic creation of links to PubMed. All you do is type `PMID` <number> and a link is automatically created, e.g. PMID 16687399.

News [edit]

- The latest BioNetGen distribution (2.0.40) includes the RuleBuilder graphical front end to BioNetGen, which allows the construction of BNG rules using graphical icons. See the Installation Guide for more details on downloading and running the latest version.
- BioNetGen will soon be available from within the Virtual Cell framework. Contact Professor Blinov for detals.
- I am creating a new Literature Review page to highlight papers of interest to BioNetGen users and modelers in general

BioNetWiki



Main Page

Welcome to BioNetWiki!

Serving the BioNetGen user community by providing information about BioNetGen and tools for the development, annotation, and discussion of BioNetGen models.

BioNetGen is software for the development and simulation of rule-based models of biochemical systems, including cell signaling, metabolic, and gene regulatory networks.

The BioNetGen software package is being developed by the Cell Signaling Team at Los Alamos National Laboratory.

Note: *To contribute to BioNetWiki, simply create an account by clicking on Log in / Create Account in the upper right hand corner of this page. You must be logged in to modify the content of Wiki pages.*

Members of the Cell Signaling Team

Contents [hide]

1 Quick Tip
2 News
3 Download
4 Documentation

Step 2: Click on Download and then Installation Guide

link is automatically created, e.g. PMID 16687559.

News

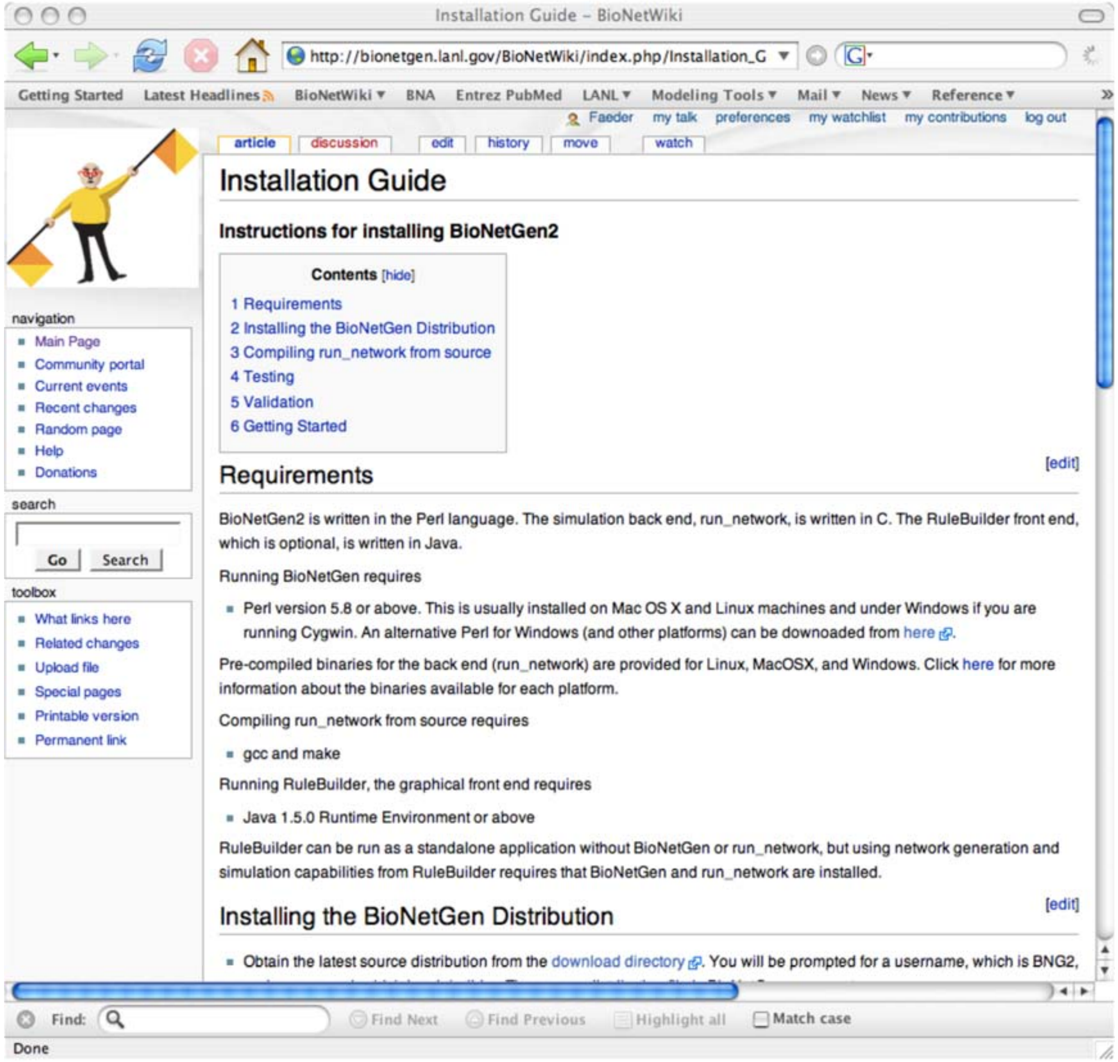
[edit]

- The latest BioNetGen distribution (2.0.40) includes the RuleBuilder graphical front end to BioNetGen, which allows the construction of BNG rules using graphical icons. See the Installation Guide for more details on downloading and running the latest version.
- BioNetGen will soon be available from within the Virtual Cell framework. Contact Professor Blinov for detals.
- I am creating a new Literature Review page to highlight papers of interest to BioNetGen users and modelers in general

Installation Guide

Under Linux and Mac, BNG is ready to run after the distribution is downloaded and unzipped.

Windows installation may require download and installation of Perl.



Installation Guide

Instructions for installing BioNetGen2

Contents [hide]

1 Requirements
2 Installing the BioNetGen Distribution
3 Compiling run_network from source
4 Testing
5 Validation
6 Getting Started

Requirements

[edit]

BioNetGen2 is written in the Perl language. The simulation back end, run_network, is written in C. The RuleBuilder front end, which is optional, is written in Java.

Running BioNetGen requires

- Perl version 5.8 or above. This is usually installed on Mac OS X and Linux machines and under Windows if you are running Cygwin. An alternative Perl for Windows (and other platforms) can be downoaded from here.

Pre-compiled binaries for the back end (run_network) are provided for Linux, MacOSX, and Windows. Click here for more information about the binaries available for each platform.

Compiling run_network from source requires

- gcc and make

Running RuleBuilder, the graphical front end requires

- Java 1.5.0 Runtime Environment or above

RuleBuilder can be run as a standalone application without BioNetGen or run_network, but using network generation and simulation capabilities from RuleBuilder requires that BioNetGen and run_network are installed.

Installing the BioNetGen Distribution

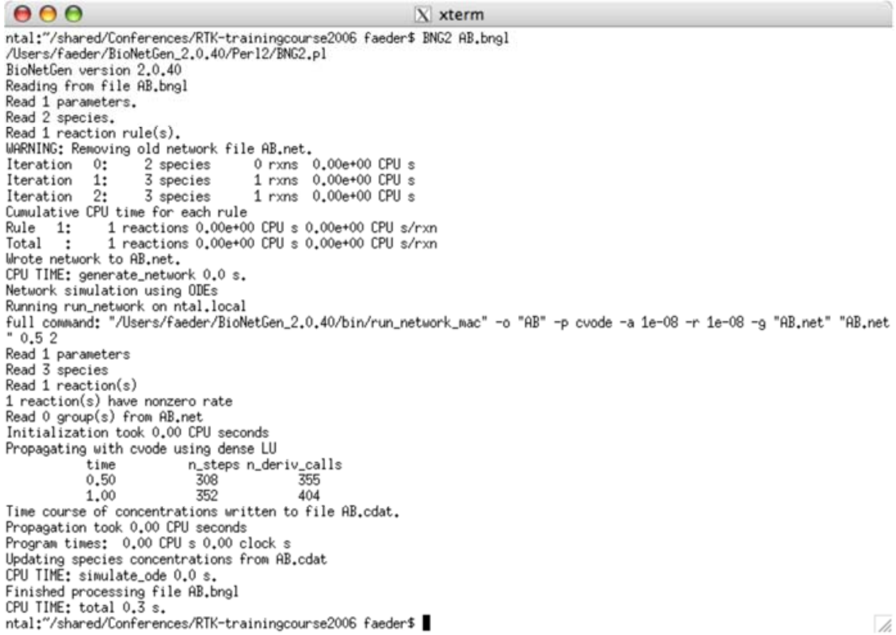
[edit]

- Obtain the latest source distribution from the download directory. You will be prompted for a username, which is BNG2,

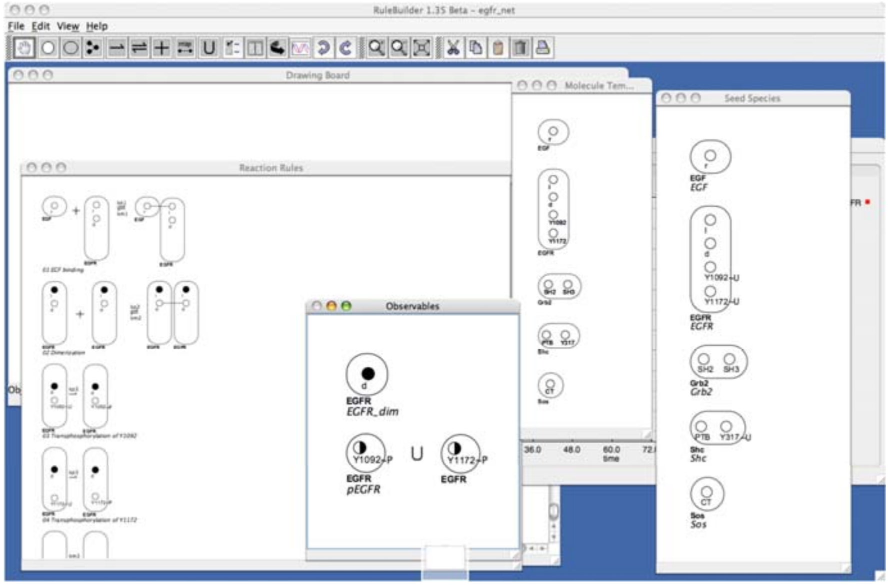
Two interfaces to BNG

Terminal interface
(text-based input)

RuleBuilder GUI

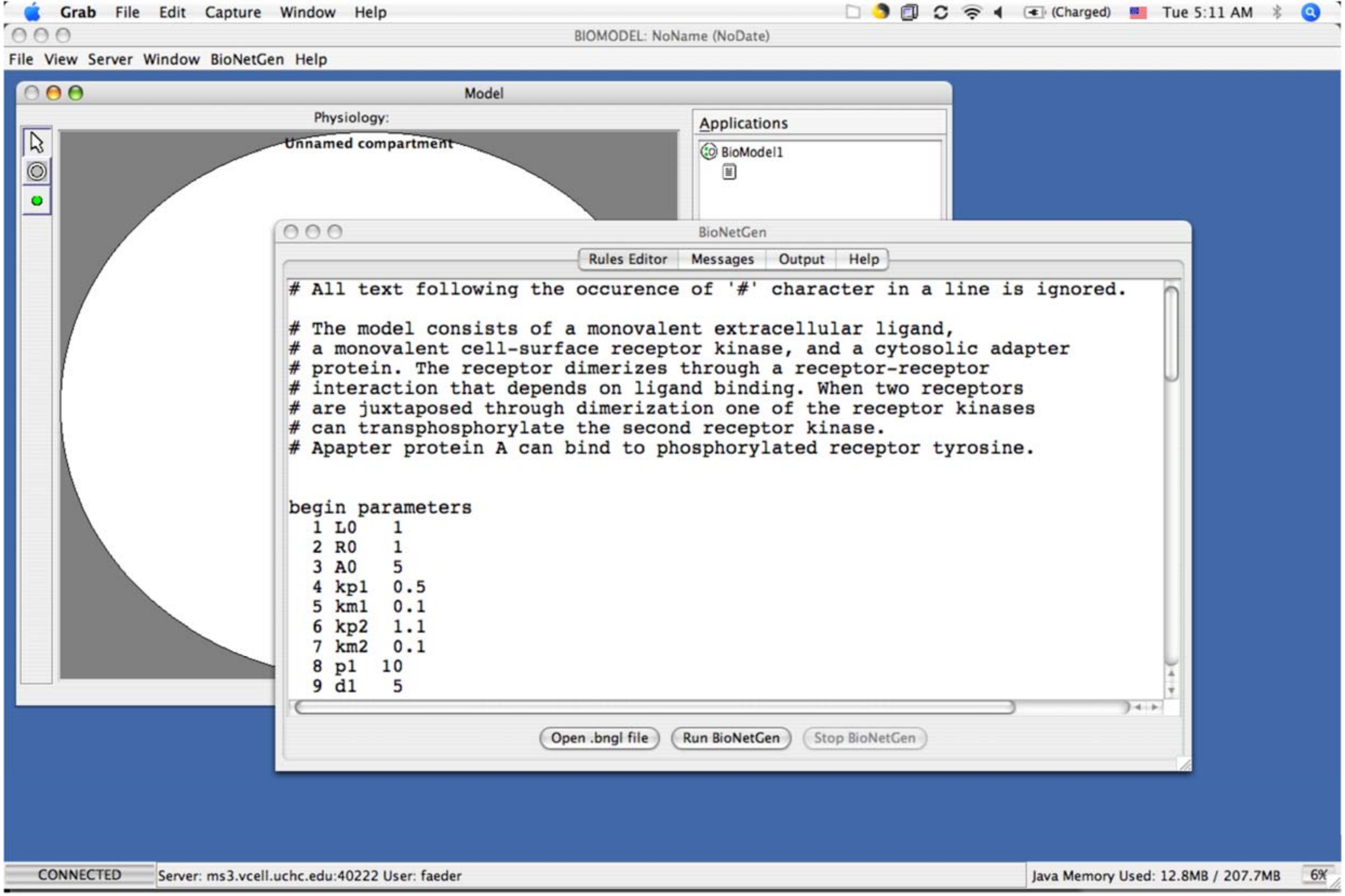


```
ntal:~/shared/Conferences/RTK-trainingcourse2006 faeder$ BNG2 AB.bngl
/Users/faeder/BioNetGen_2.0.40/Perl2/BNG2.pl
BioNetGen version 2.0.40
Reading from file AB.bngl
Read 1 parameters.
Read 2 species.
Read 1 reaction rule(s).
WARNING: Removing old network file AB.net.
Iteration   0:     2 species      0 rxns  0.00e+00 CPU s
Iteration   1:     3 species      1 rxns  0.00e+00 CPU s
Iteration   2:     3 species      1 rxns  0.00e+00 CPU s
Cumulative CPU time for each rule
Rule   1:     1 reactions 0.00e+00 CPU s 0.00e+00 CPU s/rxn
Total   :     1 reactions 0.00e+00 CPU s 0.00e+00 CPU s/rxn
Wrote network to AB.net.
CPU TIME: generate_network 0.0 s.
Network simulation using ODEs
Running run_network on ntal.local
full command: "/Users/faeder/BioNetGen_2.0.40/bin/run_network_mac" -o "AB" -p cvode -a 1e-08 -r 1e-08 -g "AB.net" "AB.net
" 0.5 2
Read 1 parameters
Read 3 species
Read 1 reaction(s)
1 reaction(s) have nonzero rate
Read 0 group(s) from AB.net
Initialization took 0.00 CPU seconds
Propagating with cvode using dense LU
           time           n_steps n_deriv_calls
           0.50            308          355
           1.00            352          404
Time course of concentrations written to file AB.cdat.
Propagation took 0.00 CPU seconds
Program times:  0.00 CPU s 0.00 clock s
Updating species concentrations from AB.cdat
CPU TIME: simulate_ode 0.0 s.
Finished processing file AB.bngl
CPU TIME: total 0.3 s.
ntal:~/shared/Conferences/RTK-trainingcourse2006 faeder$ █
```



A Third Way - Virtual Cell Interface

BioNetGen@Vcell (http://www.vcell.org/bionetgen)


Grab File Edit Capture Window Help
(Charged) Tue 5:11 AM
BIOMODEL: NoName (NoDate)
File View Server Window BioNetGen Help
Model
Physiology:
Unnamed compartment
Applications
BioModel1
BioNetGen
Rules Editor
Messages
Output
Help
Open .bngl file
Run BioNetGen
Stop BioNetGen
CONNECTED
Server: ms3.vcell.uchc.edu:40222 User: faeder
Java Memory Used: 12.8MB / 207.7MB
6%

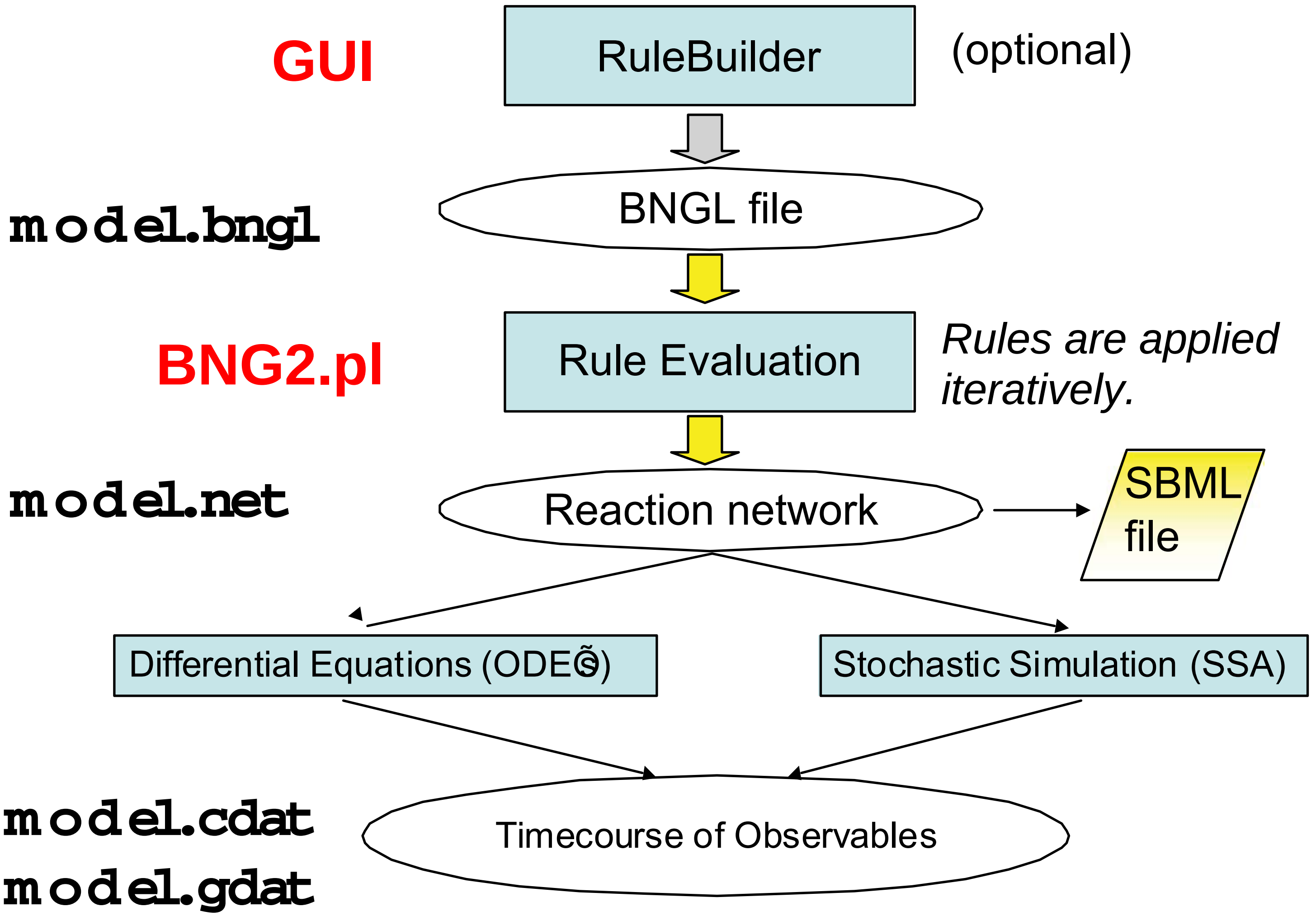
```
# All text following the occurence of '#' character in a line is ignored.

# The model consists of a monovalent extracellular ligand,
# a monovalent cell-surface receptor kinase, and a cytosolic adapter
# protein. The receptor dimerizes through a receptor-receptor
# interaction that depends on ligand binding. When two receptors
# are juxtaposed through dimerization one of the receptor kinases
# can transphosphorylate the second receptor kinase.
# Apapter protein A can bind to phosphorylated receptor tyrosine.


begin parameters
  1 L0    1
  2 R0    1
  3 A0    5
  4 kp1   0.5
  5 km1   0.1
  6 kp2   1.1
  7 km2   0.1
  8 p1   10
  9 d1    5
```

Live BioNetGen Demo

Software overview



GUI

RuleBuilder

(optional)

model.bngl

BNGL file

BNG2.pl

Rule Evaluation

Rules are applied iteratively.

model.net

Reaction network

SBML file

Differential Equations (ODEs̃)

Stochastic Simulation (SSA)

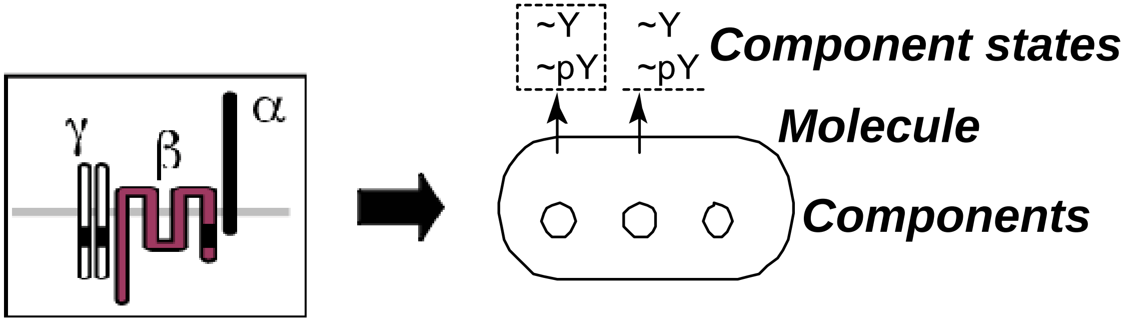
model.cdat

model.gdat

Timecourse of Observables

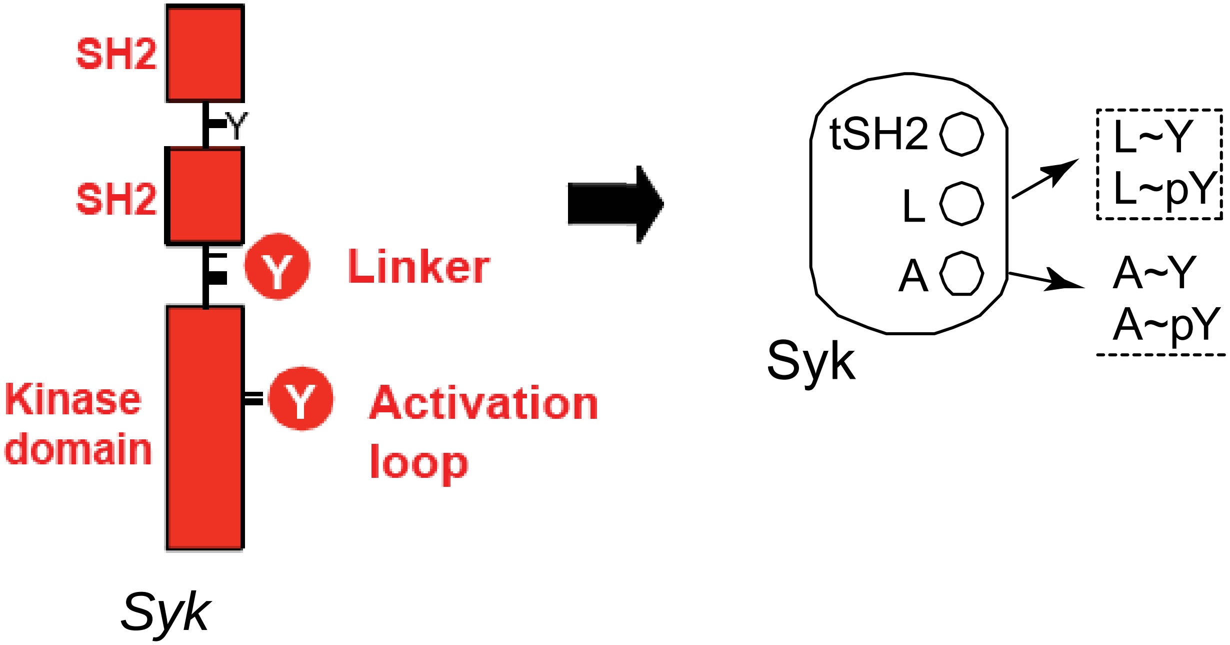
Signaling proteins and their complexes are represented as Graphs

A multi-subunit receptor

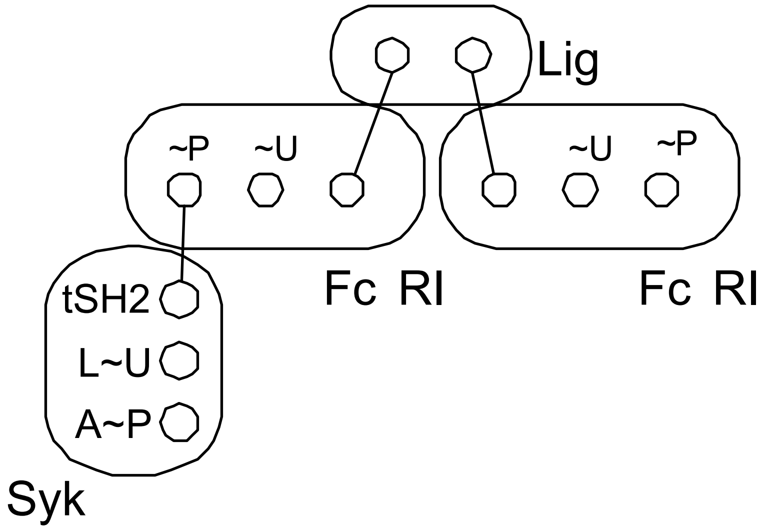


Fc RI
The high affinity
receptor for IgE

A kinase

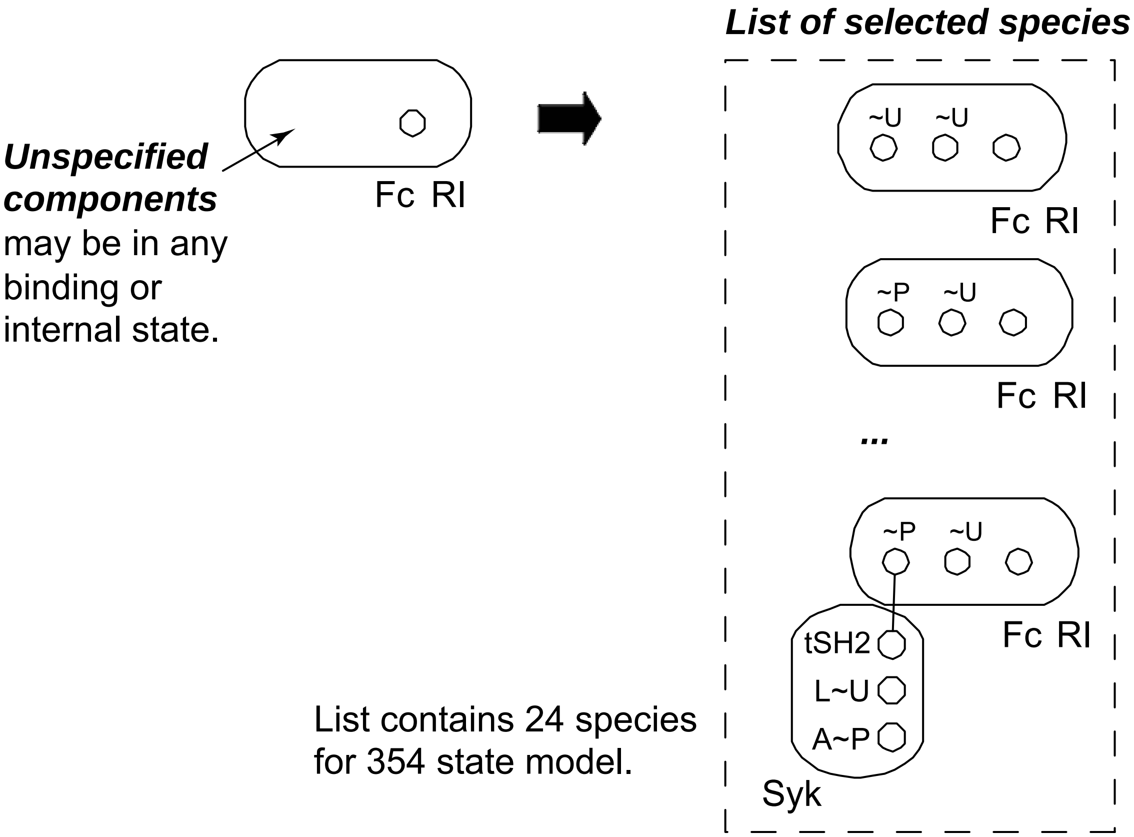

SH2
Y
SH2
Y
Linker
Kinase domain
Y
Activation loop
Syk
tSH2
L
A
Syk
L~Y
L~pY
A~Y
A~pY

A molecular complex



Patterns select groups of chemical species

A pattern that selects a receptor with free ligand binding site



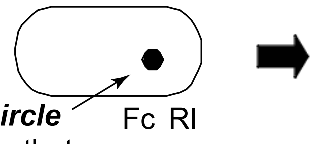
List of selected species

Unspecified components may be in any binding or internal state.

List contains 24 species for 354 state model.

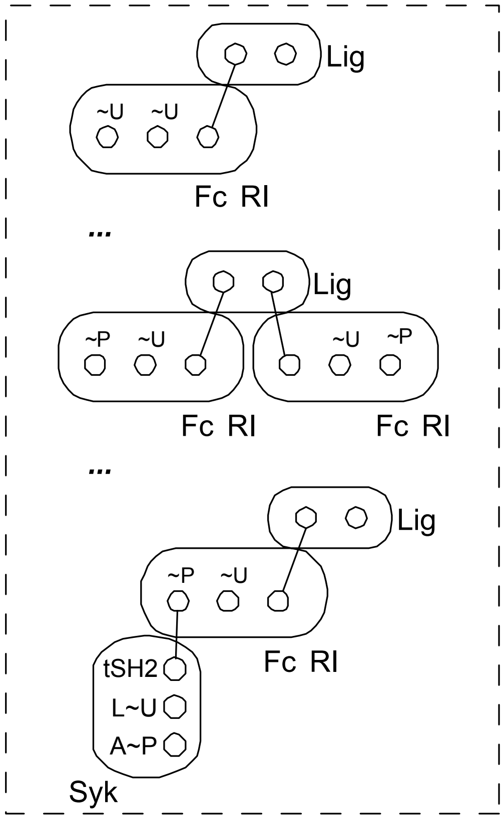
Another pattern example

A pattern that selects a receptor
with bound ligand site



Filled circle indicates that component has one or more unspecified bonds.

List of selected species



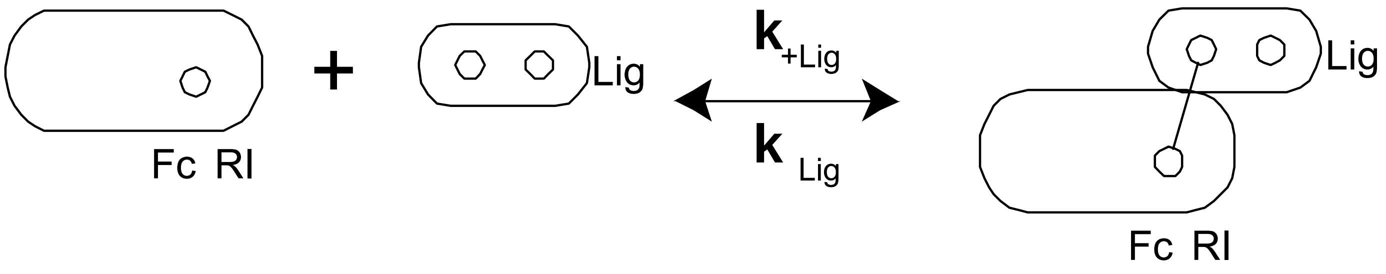
This species is selected twice becuase it contains two Fc RI molecules that match the pattern.

List contains 324 species for 354 state model.

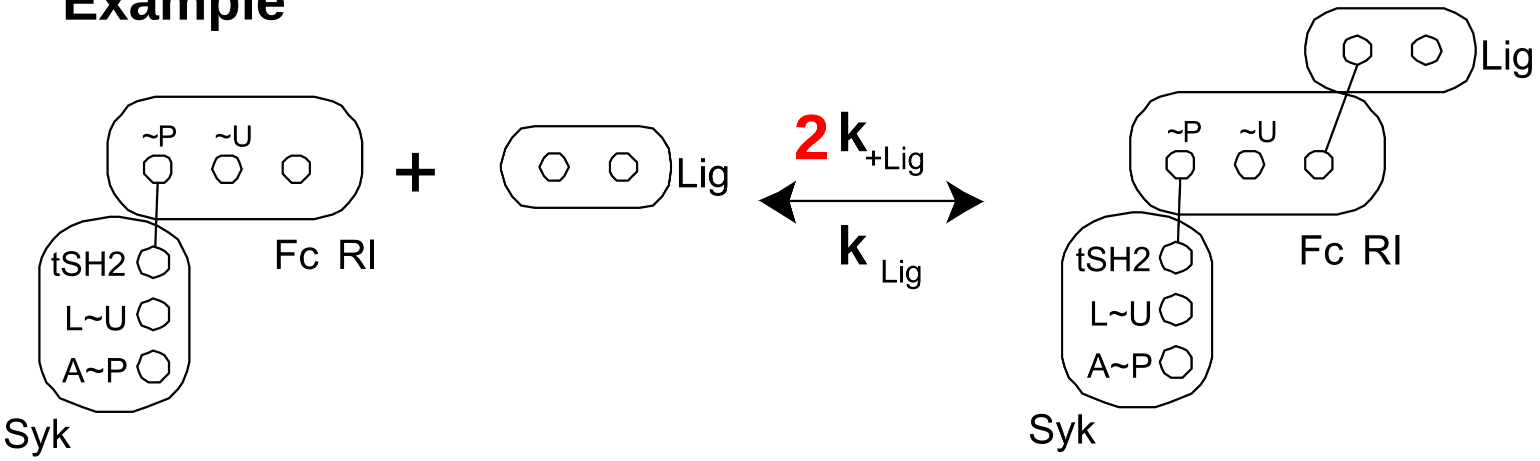
Reaction rules generate species and reactions

Ligand binding (48 reactions)

Rule



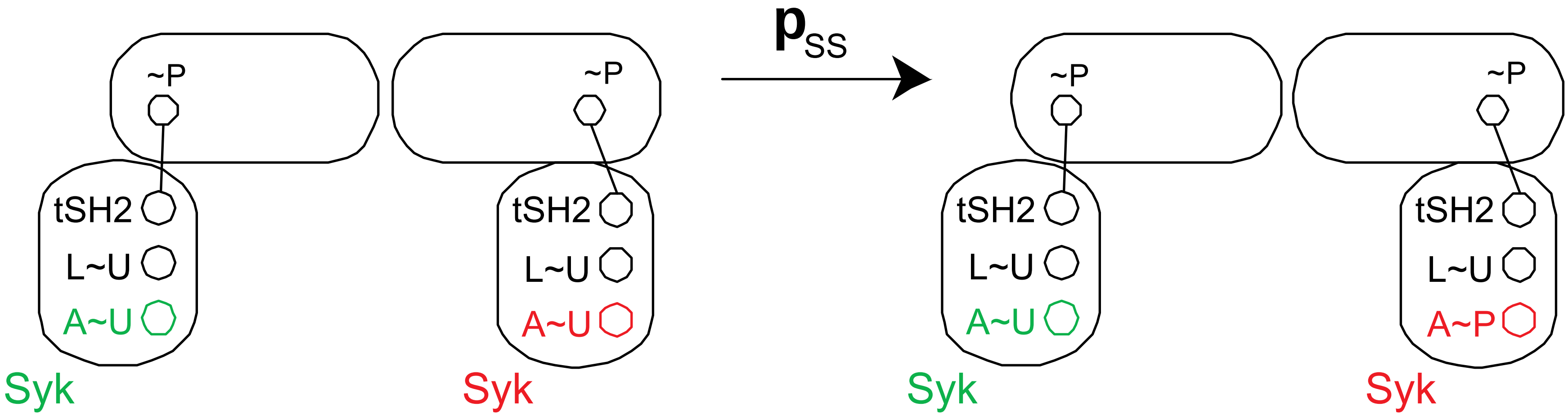
Example



Another rule example

Syk transphosphorylation (64 reactions)

Connectivity of receptors is suppressed

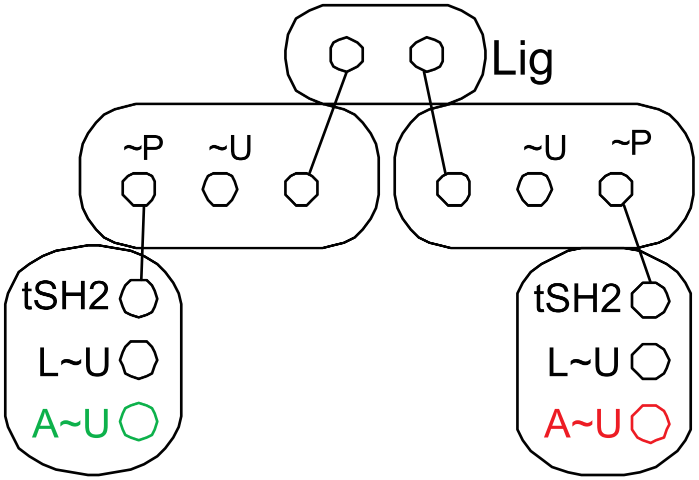


Green = Enzyme

Red = Substrate

Multiplicity example

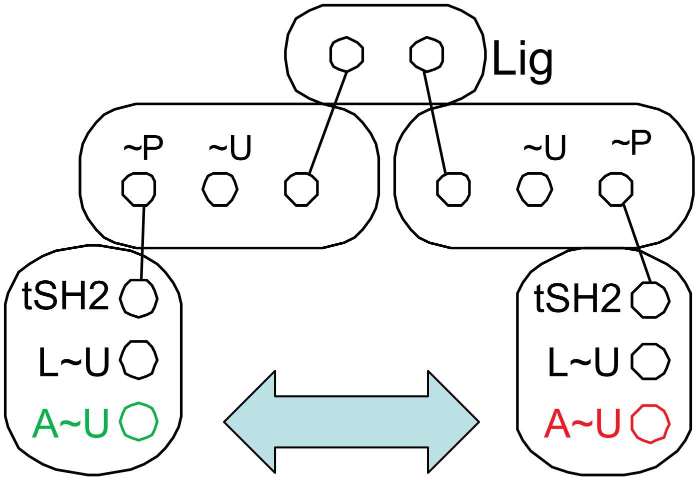
What is the multiplicity (prefactor) for the transphosphorylation rule applied to this complex?



Multiplicity example

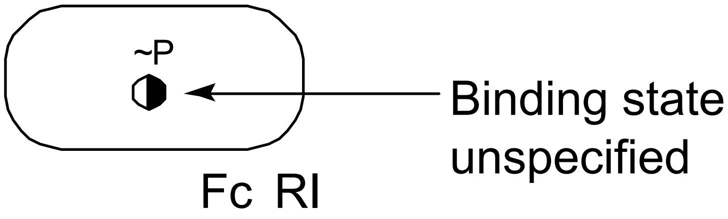
What is the multiplicity (prefactor) for the transphosphorylation rule applied to this complex?

2

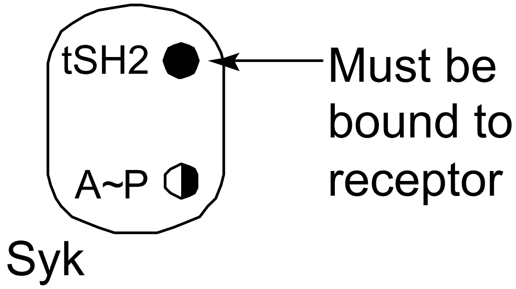


Observables define model outputs

Phosphorylated subunit (246 species)

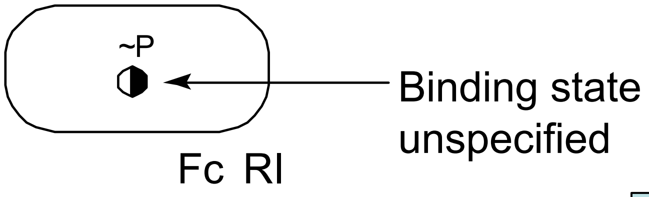


Activated Syk (180 species)



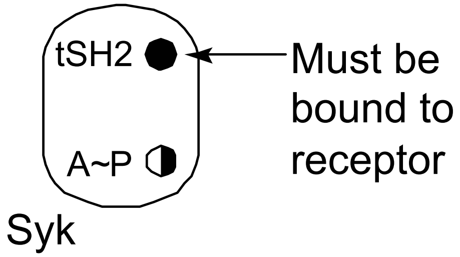
Observables define model outputs

Phosphorylated   subunit (246 species)

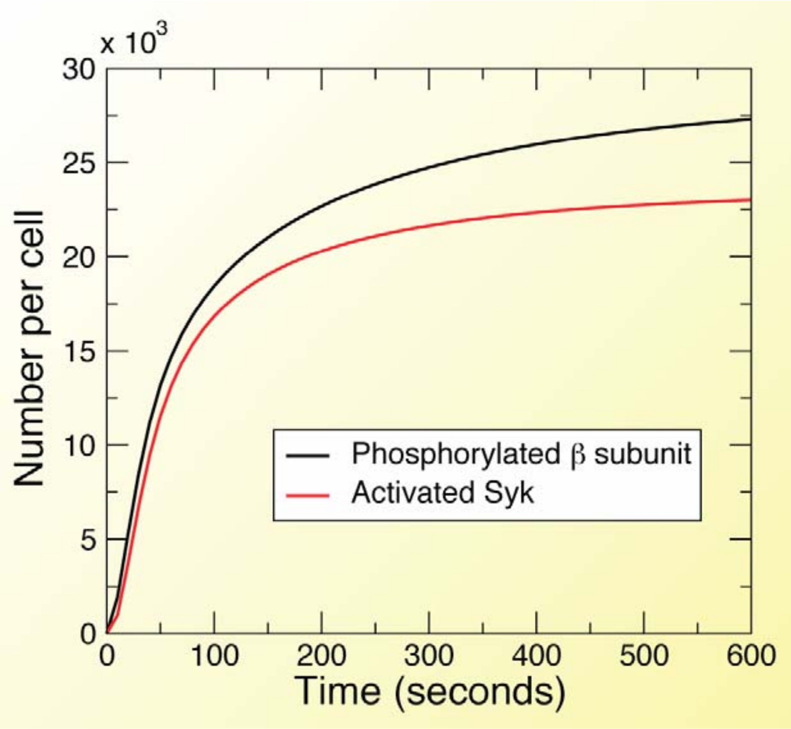


Binding state unspecified

Activated Syk (180 species)



Must be bound to receptor



How to write a model in BNGL

Following along in `example1.bngl`

Structure of the BNGL file

	file.bngl

Define named variables.	`begin parameters`
	`end parameters`

Define molecular types.	`begin molecule types`
	`end molecule types`

Define initial species and concentrations.	`begin species`
	`end species`

Define reaction types.	`begin reaction rules`
	`end reaction rules`

Define observables.	`begin observables`
	`end observables`

Generate, equilibrate, and simulate network.	`command1` …

Defining parameters

```
[index] parameter_name parameter_value
```

```
begin parameters
  1 R0   1
  2 kp1  0.5
  3 km1  0.1
  4 kp2  1e-3
  5 km2  0.1
  6 p1  10
  7 d1   5
  8 kpA  1-e4
  9 kmA  0.02
end parameters
```

Tips on Units

Consistent use of units in BNG is the user's responsibility. Any consistent set will work, but for switching between ODE and stochastic simulation methods, number per cell is the most convenient.

To get parameters in these units:

Concentrations: Multiply by $N_a \times V$, where V is $1/\rho_{\mathrm{cell}}$ for extracellular ligands, V_{cell} for other components.

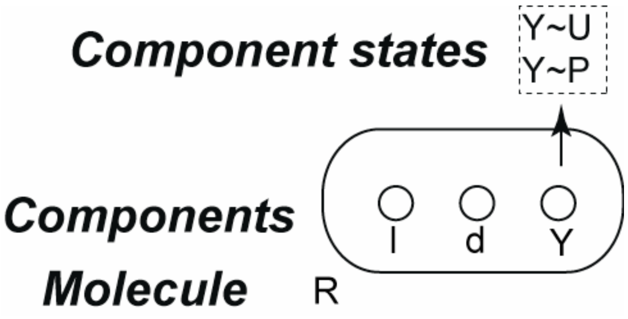
Uni-molecular rate constants: No conversion.

Bi-molecular rate constants: Divide by $N_a \times V$, where V is $1/\rho_{\mathrm{cell}}$ extracellular ligand binding, V_{cell} reactions involving 1 or more cytoplasmic proteins, and χV_{cell} for reactions occurring in the plasma membrane.

Defining molecules

The `molecule types` block is optional, but is always generated by the program (.NET file)

```
Molecule(comp1~s1~s2,... )
```



```
R(l,d,Y~U~P)
```

Components represent domains of proteins. May be binding sites, have conformational states, or both.

Defining initial species

```
[index] species_string [initial conc.]
```

defaults to 0

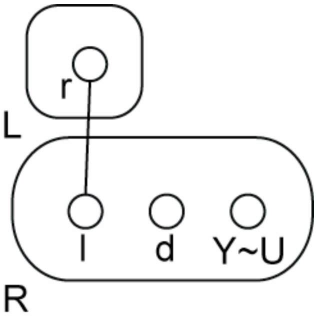
```
begin species
  1 L(r)       L0
  2 R(l,d,Y~U) R0
  3 A(SH2)     A0
end species
```

Key points

1. No spaces in species strings
2. States for components that take states
3. Initial concentration may be number or parameter

Bonds and complexes

Bonds are indicated by edges in the species graph. Bonds are indicated by an `!<number>`, where `<number>` is the index of the bond.



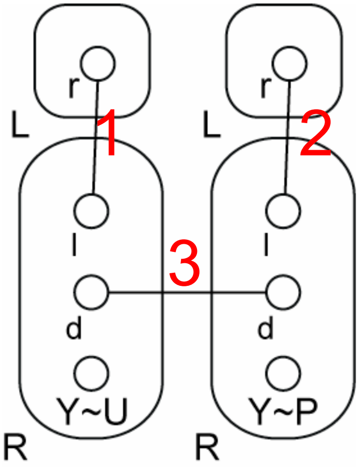
bond

```
L(r!1).R(l!1,d,Y~U)
```

‘.’ used to join molecules

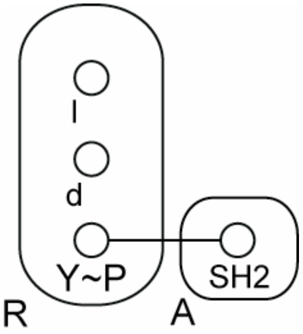
Note: bond index is used only to identify bond endpoints. All bonds are otherwise equivalent.

A more complex example



L(r!1).R(l!1,d!3,Y~U).L(r!2).R(l!2,d!3,Y~P)

Mixing states and edges

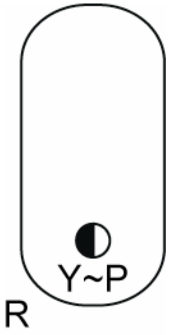


R(l,d,Y~P!1).A(SH2!1)

Patterns

Definition. A **pattern** is a graph in which some elements may be unspecified or may represent a range of values.

Patterns are used to select sets of species with common attributes on which to perform operations.



```
R(Y~P!?)
```

‘?’ indicates that bonding state is unspecified

Examples of matches

```
R(l,d,Y~P)
L(r!1).R(l!1,d,Y~P)
R(l,d,Y~P!1).A(SH2!1)
R(l,d!1,Y~P).R(l,d!1,Y~P)
R(l,d!1,Y~P).R(l,d!1,Y~P)
```

}**Two matches for same species**

Pattern conventions

1. Any unspecified component may take on any internal or binding state. In `R(Y~P!?)`both l and r are unspecified.
2. If a component is specified without an internal state, it may take on any internal state.
3. There are two edge wildcards:

   `!?` means may or may not be bound

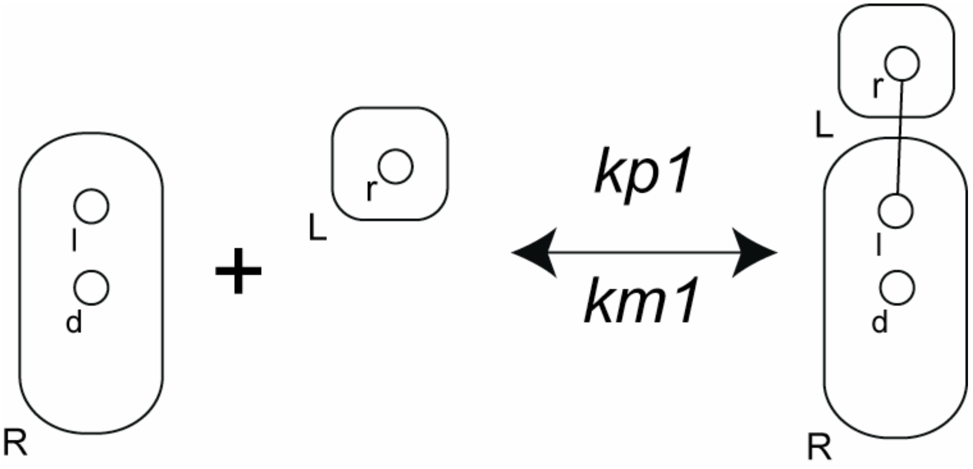
   `!+` means one or more additional bonds must be present

Reaction rules

Reaction rules consist of reactant and product patterns that are used to specify a transformation

```
reac_1 + … + reac_M -> prod_1 + … + prod_N k_f

reac_1 + … + reac_M <-> prod_1 + … + prod_N k_f, k_r
```



```
L(r) + R(l,d) <-> L(r!1).R(l!1,d) kp1, km1
```

Application of the reaction rule

```
L(r) + R(l,d) <-> L(r!1).R(l!1,d) kp1, km1
```

Forward

```
L(r) + R(l,d,Y~U) -> L(r!1).R(l!1,d,Y~U) kp1
L(r) + R(l,d,Y~P) -> L(r!1).R(l!1,d,Y~P) kp1
L(r) + R(l,d,Y~P!1).A(SH2!1) -> \
                        L(r!2).R(l!2,d,Y~P!1).A(SH2!1) kp1
```

Reverse

```
L(r!1).R(l!1,d,Y~U) -> L(r) + R(l,d,Y~U) km1
```

Observables

Definition. An **observable** is the sum of concentrations over a set of species selected by one or more patterns.

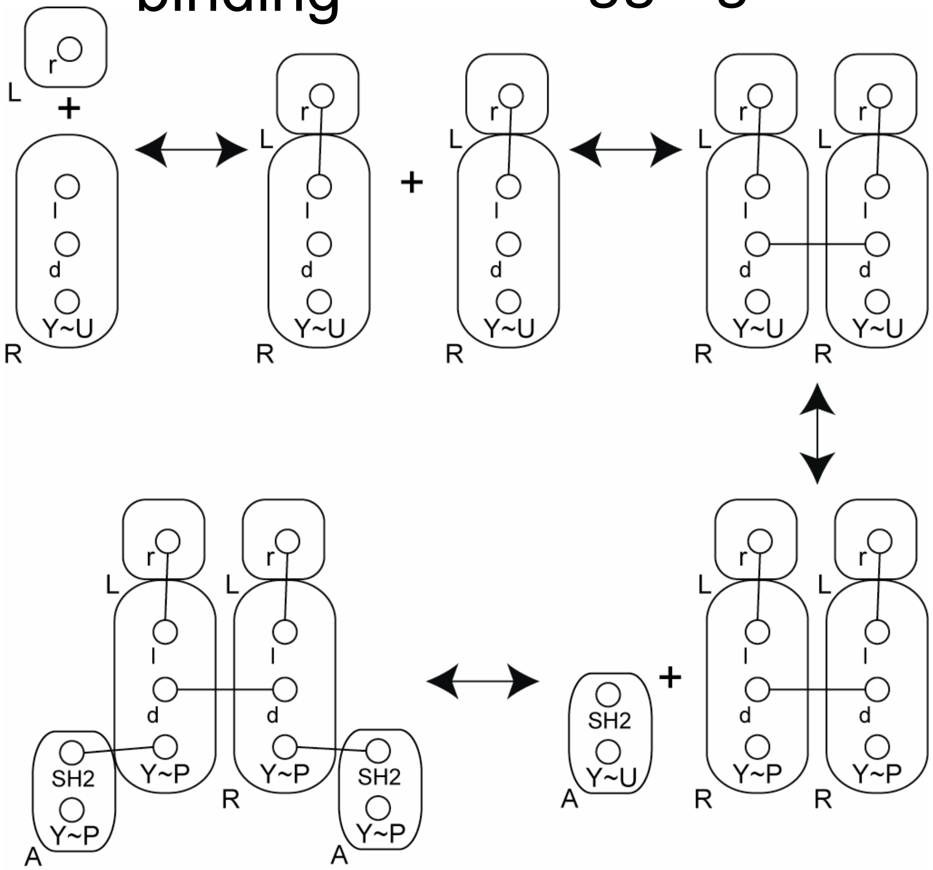
```
[type of observable] <observable name> patt1, …, patt_N
```

`Rdim R(d!+)`	Selects receptors with dimerization domain bound
`Rphos R(Y~P!?)`	Selects receptors with phosphorylated tyrosine

A toy model

Ligand binding

Receptor aggregation



Receptor trans-phosphorylation

Adaptor binding and transphosphorylation

Note: This is a schematic, not the actual reaction rules

Parameters

```
begin parameters
  1 L0   200      # Number of ligand molecules
  2 R0   200      # Number of receptor molecules
  3 A0   50              # Number of adaptor molecules
  4 kp1  0.01     # Ligand-receptor association
  5 km1  1               # Ligand-receptor dissociation
  6 kp2  1               # Dimer formation
  7 km2  1               # Dimer dissociation
  8 p1  10               # Receptor transphosphorylation
  9 d1   5               # Receptor dephosphorylation
 10 kpA  0.1      # Adaptor-receptor association
 11 kmA  0.1      # Adaptor-receptor dissociation
 12 p2   10              # Adaptor transphosphorylation
 13 d2    5              # Adaptor dephosphorylation
end parameters
```

Molecules and Species

```
begin species
  1 L(r)        L0
  2 R(l,d,Y~U) R0
  3 A(SH2,Y~U) A0
end species
```

Observables

```
begin observables
  Molecules R_dim  R(d!+)
  Molecules R_phos R(Y~P!?)
  Molecules A_R    A(SH2!1).R(Y~P!1)
  Molecules A_phos A(Y~P!?)
end observables
```

`Molecules` keyword indicates that each species concentration is multiplied by the number of matches.

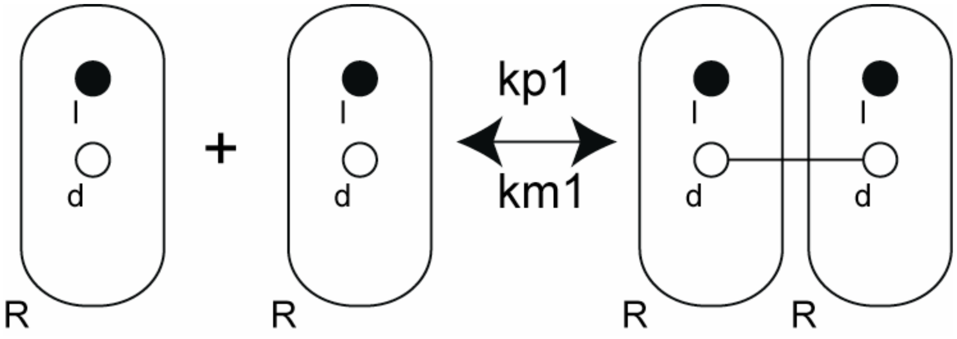
`Species` keyword indicates that concentration of each species is only added once.

Reaction Rules

```
begin reaction rules
  # Ligand-receptor binding
  1  L(r) + R(l,d) <-> L(r!1).R(l!1,d) kp1, km1
  # Receptor aggregation
  2  R(l!+,d) + R(l!+,d) <-> \
          R(l!+,d!2).R(l!+,d!2) kp2, km2
  # Receptor transphosphorylation
  3  R(d!+,Y~U) -> R(d!+,Y~P) p1
  # Receptor dephosphorylation
  4  R(Y~P) -> R(Y~U) d1
  # Adaptor association
  5  R(Y~P) + A(SH2) <-> R(Y~P!1).A(SH2!1) kpA, kmA
  # Adaptor transphosphorylation
  6  A(Y~U).A() -> A(Y~P).A() p2
  # Adaptor dephosphorylation
  7  A(Y~P) -> A(Y~U) d2
end reaction rules
```

Example of symmetric reaction

```
# Receptor aggregation
 2 R(l!+,d) + R(l!+,d) <-> \
        R(l!+,d!2).R(l!+,d!2) kp2, km2
```



Symmetry of reactant `R` molecules is preserved under this transformation. Rate constants are multiplied by factor of 1/2 to give correct rate, assuming `kp2` and `km2` are for single bond.

Commands

```
generate_network({overwrite=>1});
```

Apply reaction rules iteratively to generate species and reactions.

```
writeSBML();
```

Write reaction network to SBML Level 2 file.

```
simulate_ode({t_end=>5,n_steps=>50});
```

Solve ODE’s to obtain time course for species concentrations and observables.

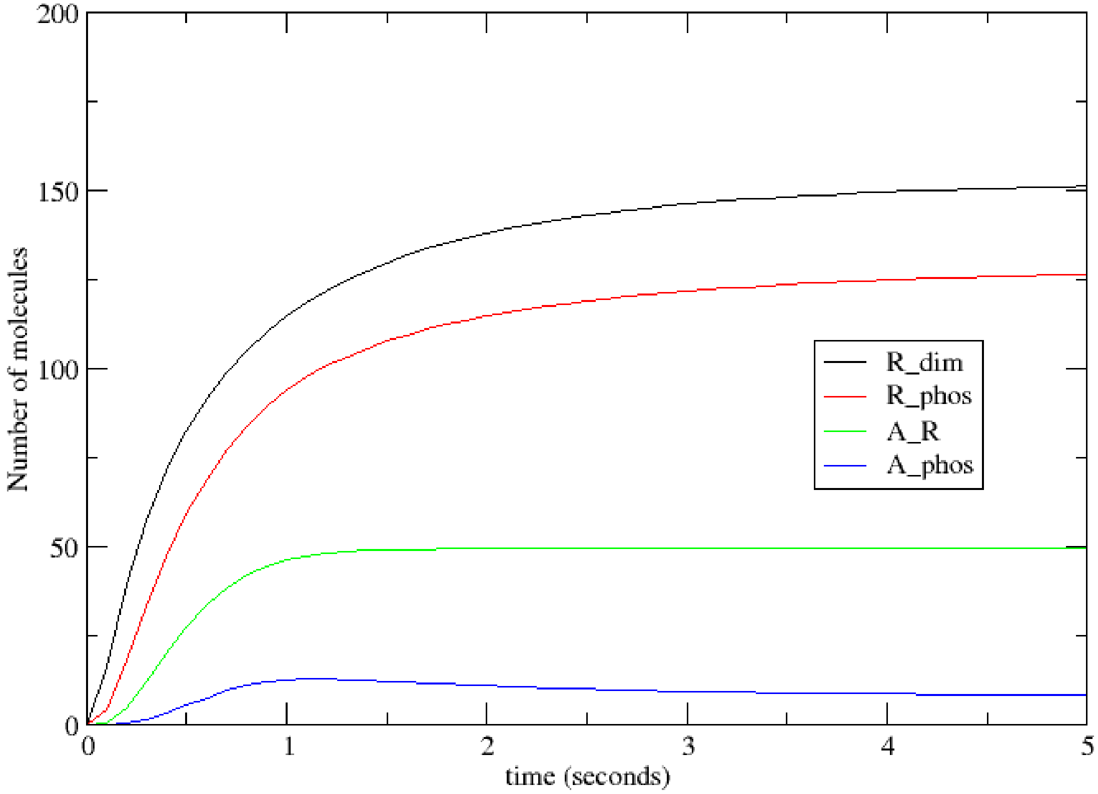
See tutorial file for more details on command parameters.

Output

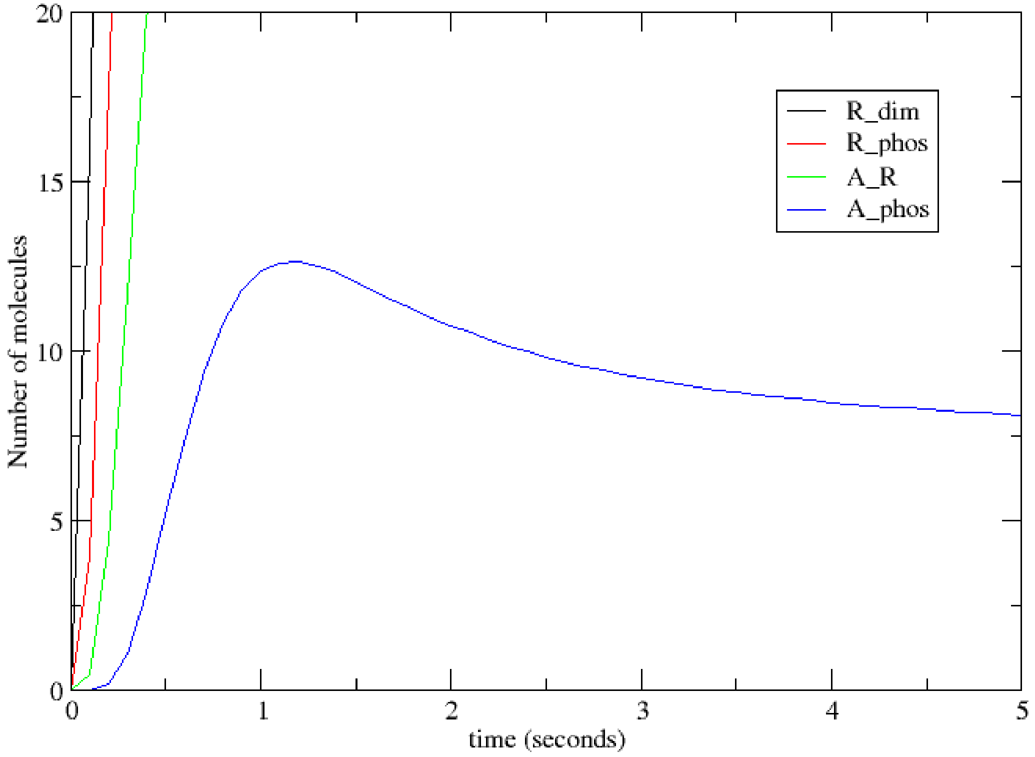
```
BioNetGen version 2.0.19+
Reading from file example1.bngl
Read 13 parameters.
Read 3 species.
Read 4 observable(s).
Adding P as allowed state of component Y of molecule R
Adding P as allowed state of component Y of molecule A
Read 7 reaction rule(s).
WARNING: Removing old network file example1.net.
Iteration   0:     3 species      0 rxns 0.00e+00 CPU s 0.00e+00 (4.01e+00) Mb real (virtual) memory.
Iteration   1:     4 species      1 rxns 2.00e-02 CPU s 4.03e+00 (2.94e+01) Mb real (virtual) memory.
Iteration   2:     5 species      3 rxns 1.00e-02 CPU s 4.04e+00 (2.94e+01) Mb real (virtual) memory.
Iteration   3:     6 species      5 rxns 4.00e-02 CPU s 4.06e+00 (2.94e+01) Mb real (virtual) memory.
Iteration   4:     9 species      9 rxns 5.00e-02 CPU s 4.09e+00 (2.94e+01) Mb real (virtual) memory.
Iteration   5:    12 species     20 rxns 1.10e-01 CPU s 4.14e+00 (2.94e+01) Mb real (virtual) memory.
Iteration   6:    14 species     32 rxns 1.10e-01 CPU s 4.17e+00 (2.94e+01) Mb real (virtual) memory.
Iteration   7:    15 species     37 rxns 8.00e-02 CPU s 4.19e+00 (2.94e+01) Mb real (virtual) memory.
Iteration   8:    19 species     42 rxns 8.00e-02 CPU s 4.24e+00 (2.94e+01) Mb real (virtual) memory.
Iteration   9:    21 species     64 rxns 2.30e-01 CPU s 4.28e+00 (2.94e+01) Mb real (virtual) memory.
Iteration  10:    21 species     71 rxns 6.00e-02 CPU s 4.28e+00 (2.94e+01) Mb real (virtual) memory.
```

Toy network has **21 species** and **71 reactions**.

Simulation Results



Adaptor phosphorylation exhibits transient peak

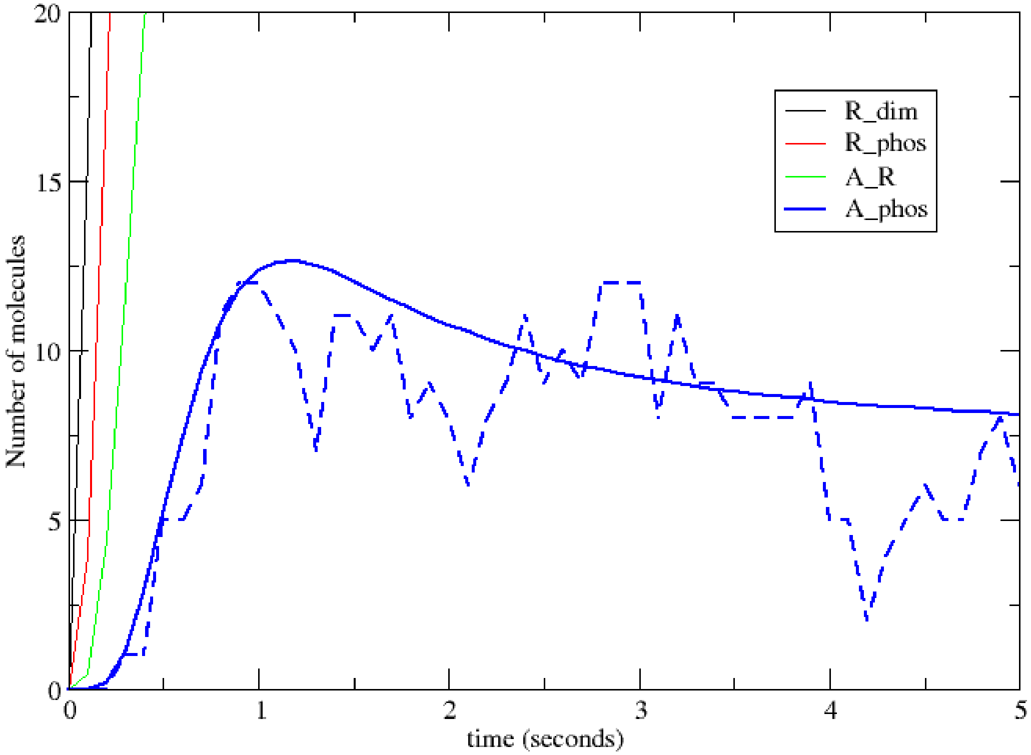


Stochastic simulation using Gillespie algorithm

Use `simulate_ssa` instead of `simulate_ode`

```
simulate_ssa({t_end=>5,n_steps=>50});
```

Results of Stochastic Simulation

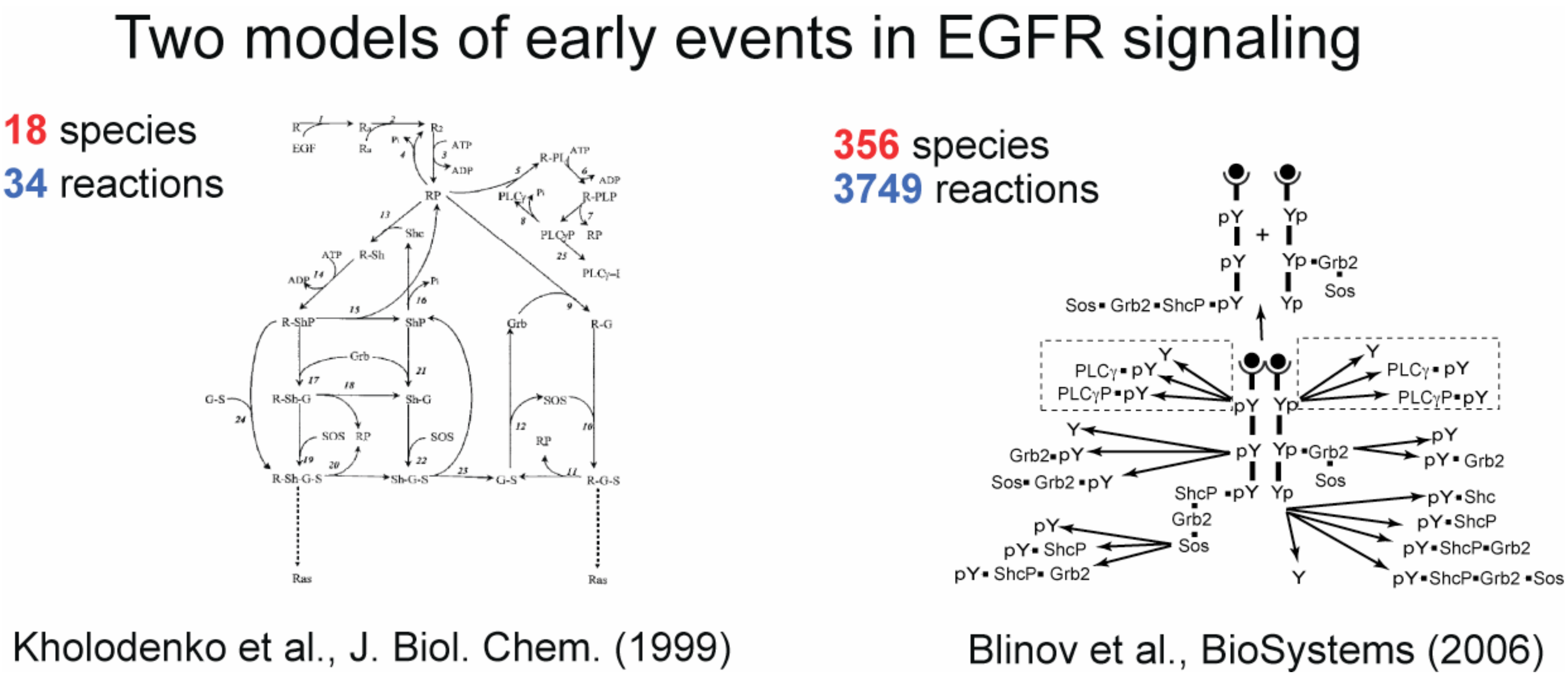


Suggested exercise

“Complexify” your favorite signaling model!

Suggested exercise

“Complexify” your favorite signaling model!



Two models of early events in EGFR signaling

18 species
34 reactions

356 species
3749 reactions

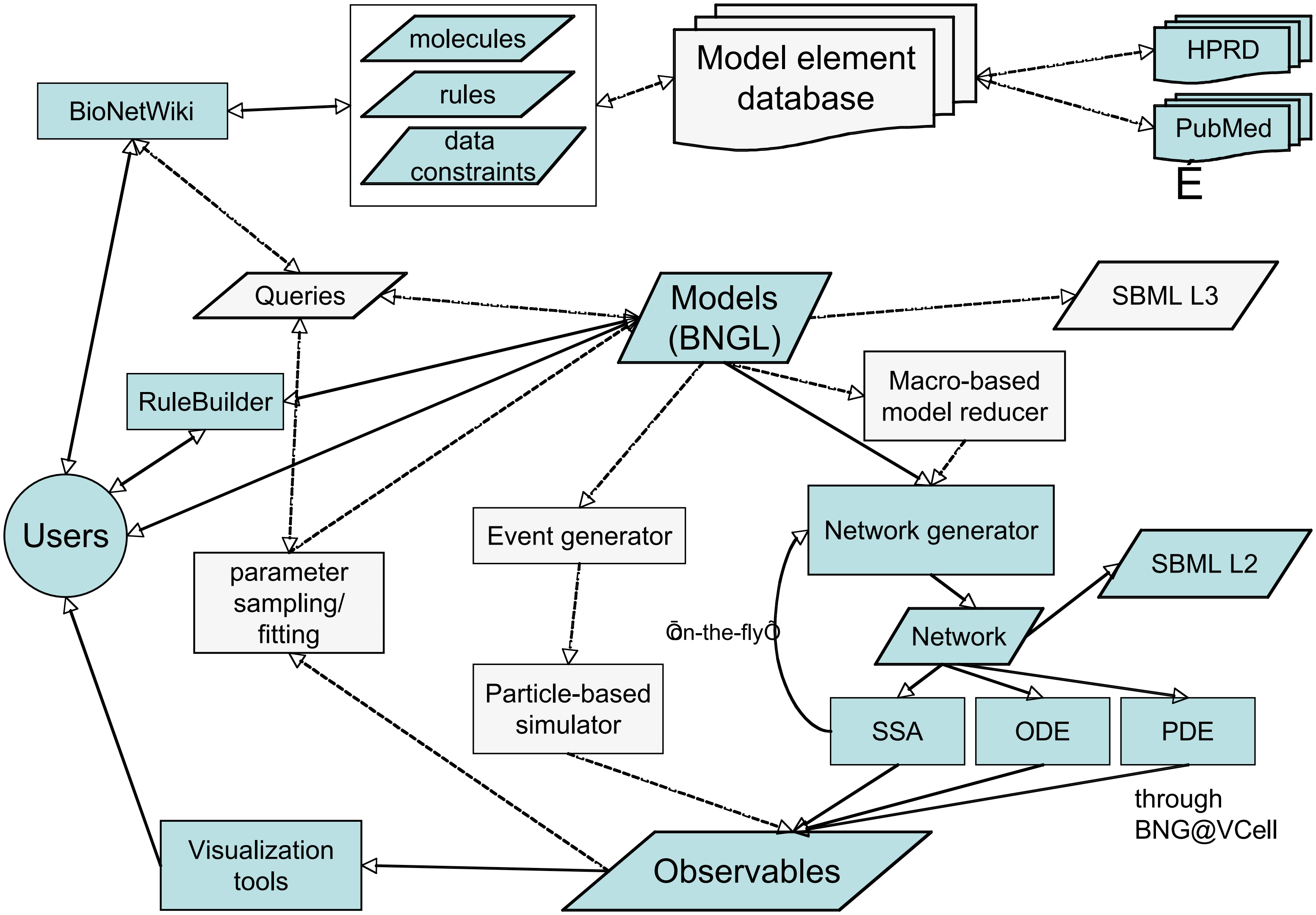
Kholodenko et al., J. Biol. Chem. (1999)

Blinov et al., BioSystems (2006)

Key Areas for BioNetGen Development

- Compartments
- Network generation efficiency
- Network simulation efficiency
  - Particle-based methods
- Spatial simulation capabilities
  - Vcell
  - Particle-based stochastic methods
- Parameter estimation
  - Scanning and fitting
  - Sensitivity / Uncertainty analysis
- Collaborative modeling using wiki servers and BNGL elements

BioNetGen Flow Chart


molecules
rules
data constraints
Model element database
HPRD
PubMed
É
BioNetWiki
Queries
Models (BNGL)
SBML L3
RuleBuilder
Macro-based model reducer
Users
Event generator
Network generator
SBML L2
parameter sampling/ fitting
Ōn-the-flyÓ
Network
Particle-based simulator
SSA
ODE
PDE
through BNG@VCell
Visualization tools
Observables