

# **Introduction to Rule-Based Modeling of Biochemical Systems with BioNetGen and RuleBender**

**Jim Faeder, Ph.D.**  
Univ. of Pittsburgh, MMBioS

MCell Workshop  
Pittsburgh Supercomputing Center  
April 28-30, 2014

# Biomedical Technology Research Center (BTRC)

## High Performance Computing *for*

# Multiscale Modeling of Biological Systems

*Overarching biological theme:*

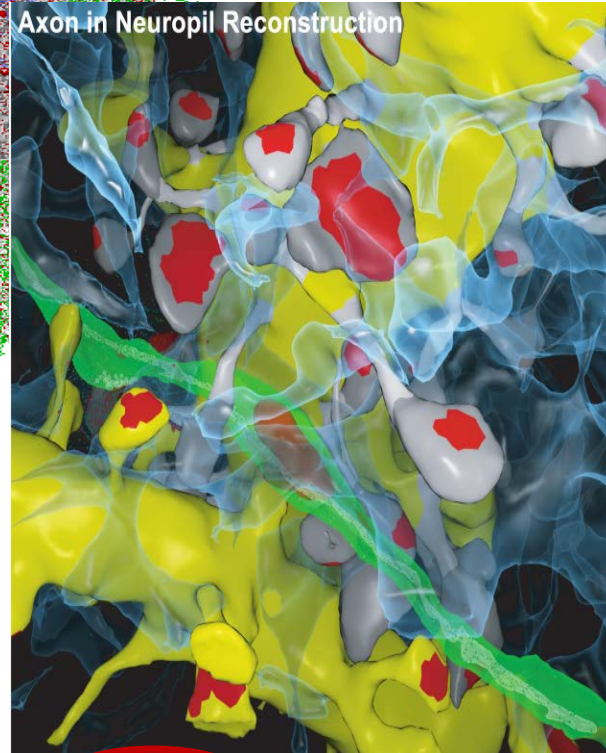
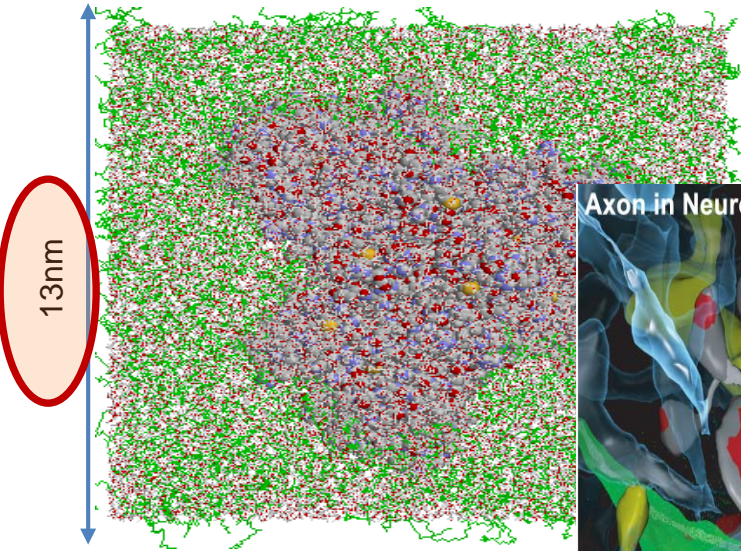
- Spatial organization
  - Temporal evolution
- of (neuro)signaling systems/events



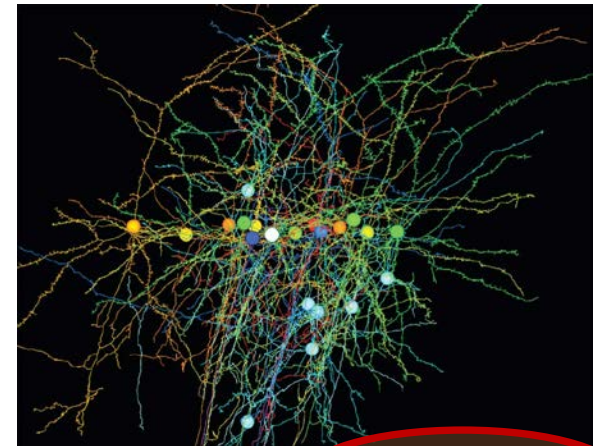
From small molecules, to multimeric assemblies,

to cellular architecture,

to neural circuits

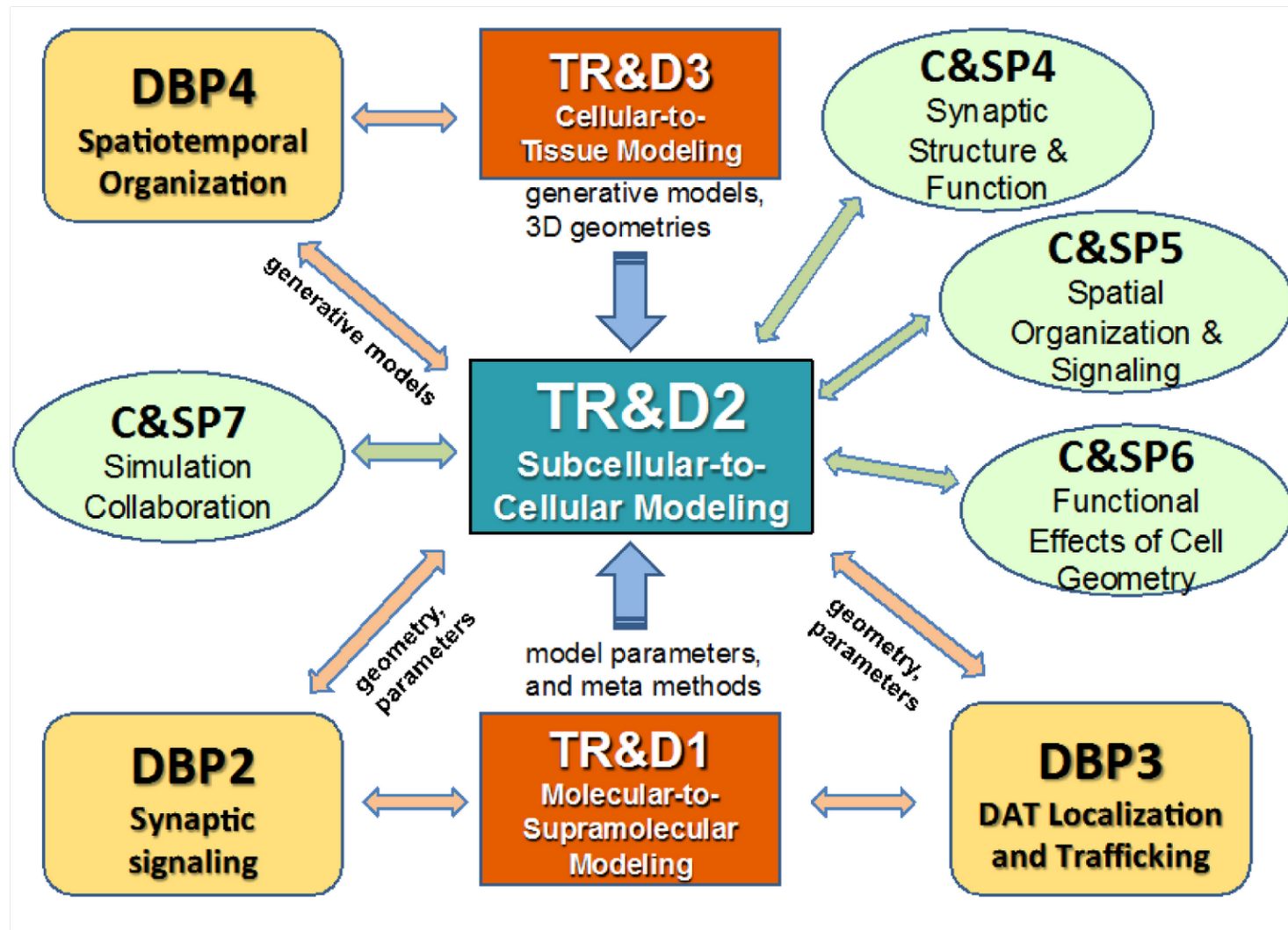


from  $6 \times 6 \times 5 \mu\text{m}^3$  sample of adult rat hippocampal stratum radiatum neuropil



From SSEM images,  $400 \times 400 \times 50 \mu\text{m}^3$

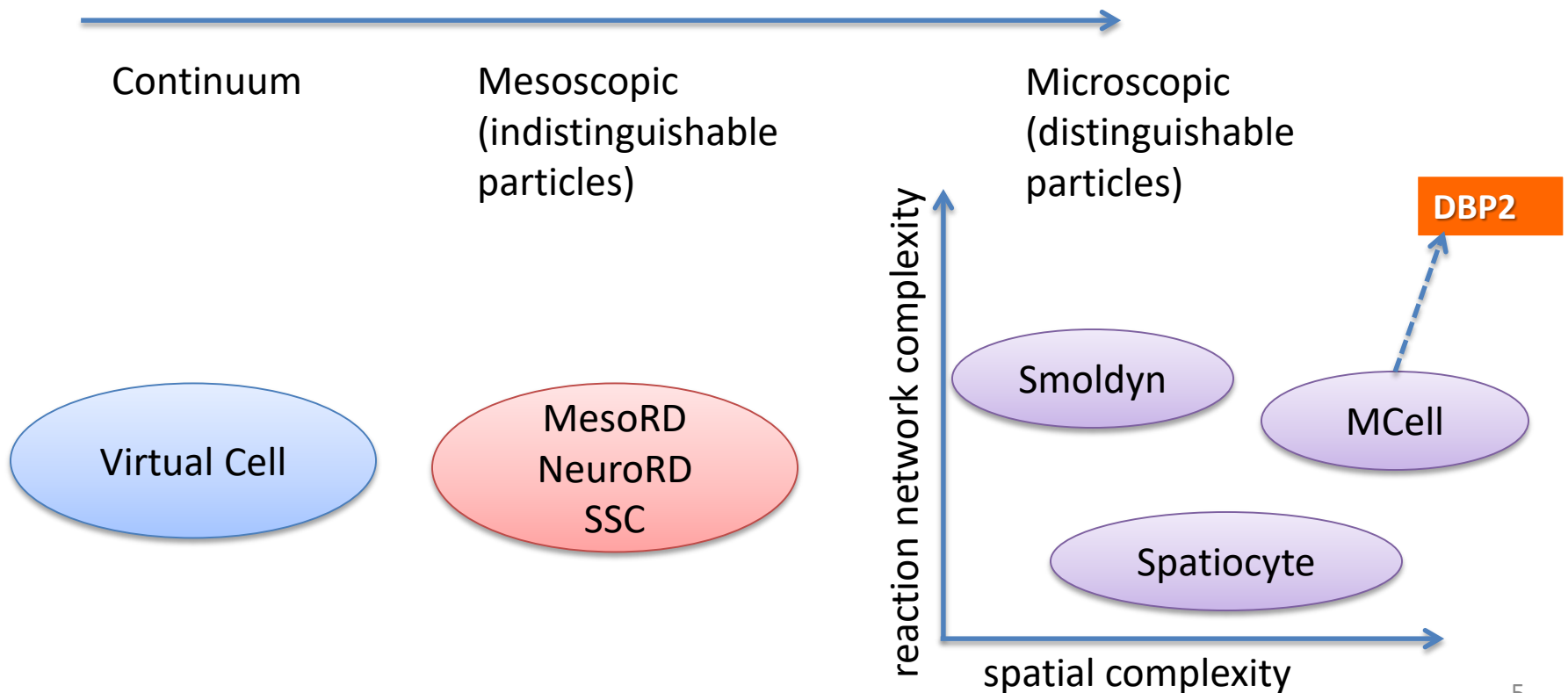
# Role of MCell in the BTRC



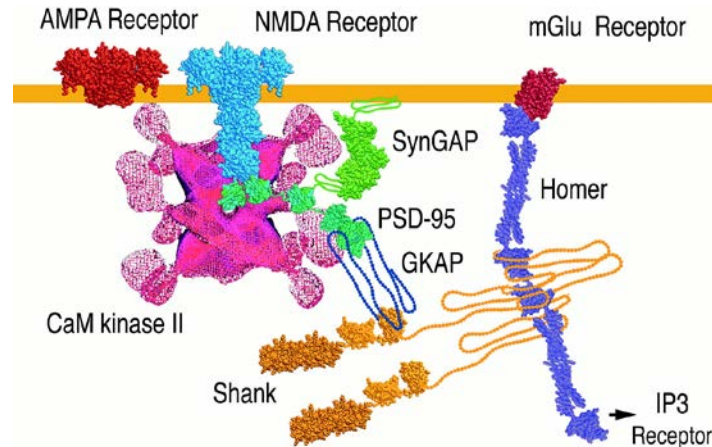


# Comparison of MCell with other tools for spatial modeling of biological systems

Particle resolution



# Motivating example for Rule-Based Modeling



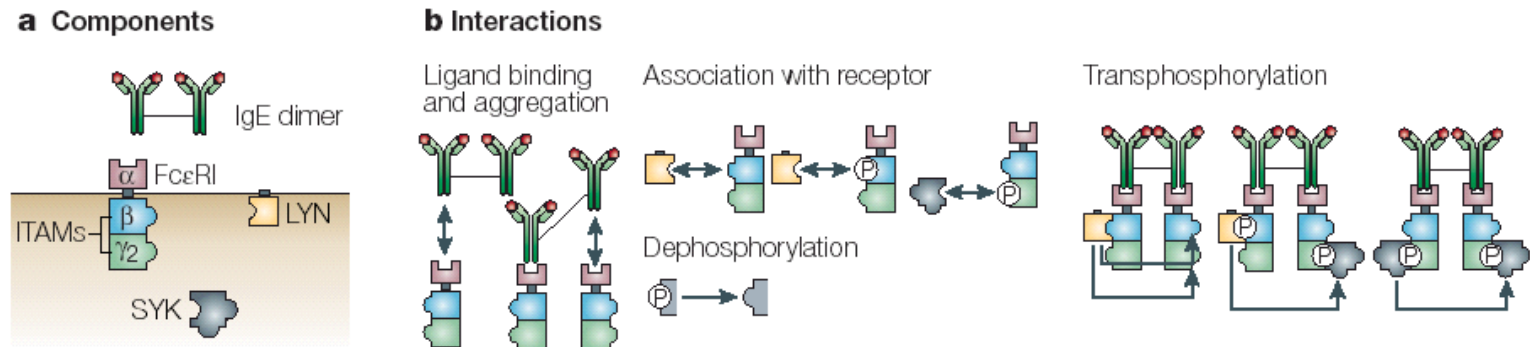
**Molecular machines in the PSD**

**Estimated number of states of  
CAMKII-CaM complexes:**

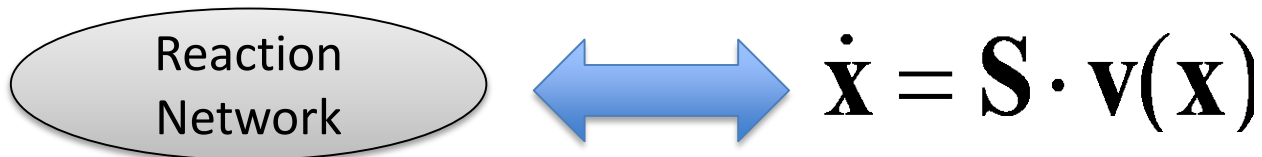
**$40^{12}$**

# Standard modeling protocol

1. Identify components and interactions.



2. Write model reactions / equations



3. Determine **concentrations** and **rate constants**



4. Simulate and analyze the model

ODE's

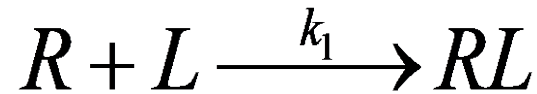
SSA

PDE's

BD

# Reactions to Differential Equations

Consider the reaction

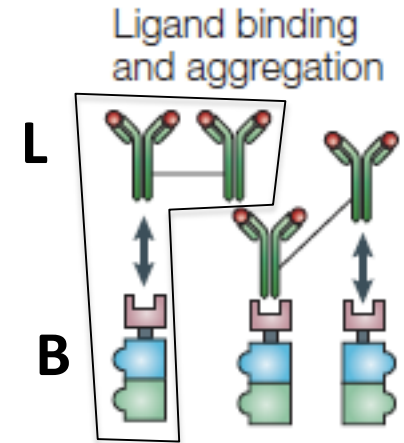


The reaction rate is given by

$$v_1 = k_1 R \cdot L$$

Rate of change of species concentrations (numbers) are

$$\begin{aligned} \frac{dR}{dt} &= -v_1 + \dots \\ \frac{dL}{dt} &= -v_1 + \dots \\ \frac{dRL}{dt} &= +v_1 + \dots \end{aligned}$$

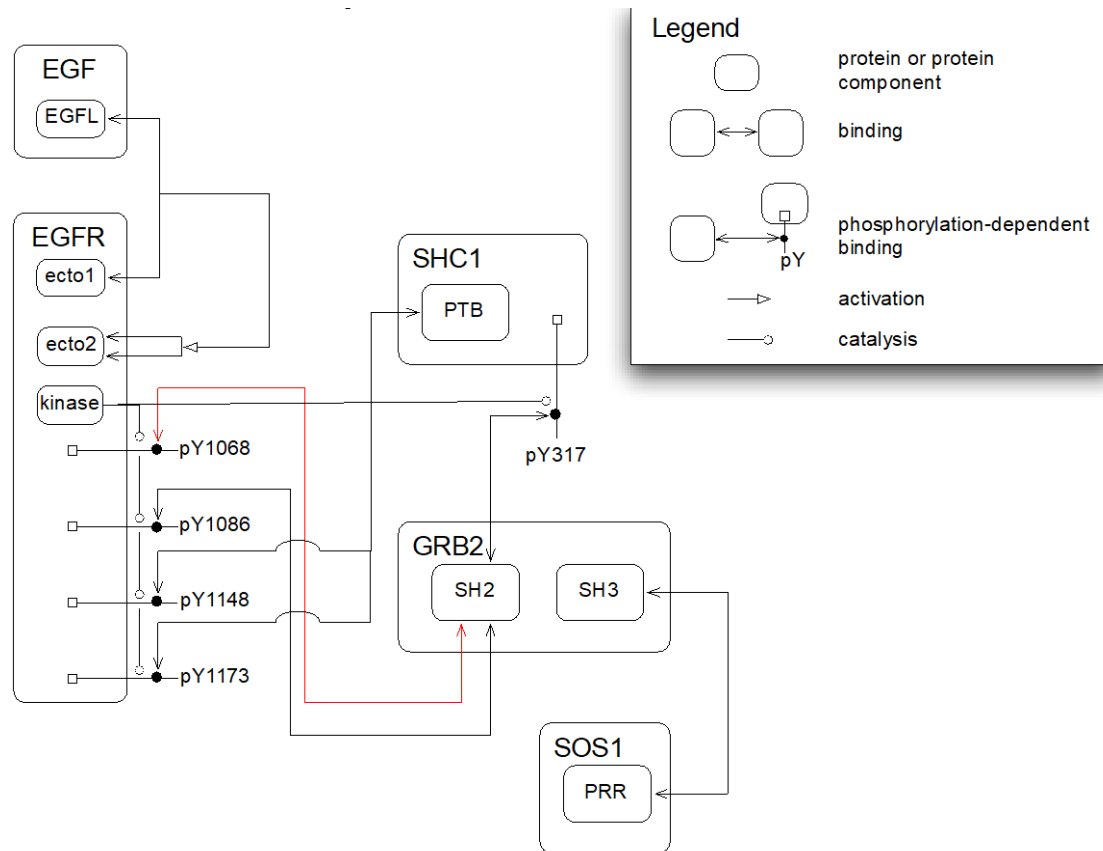


Here I have indicated that there may be additional terms from other reactions in the network. Reaction fluxes combine through *addition*.





# Combinatorial complexity in a prototypical signaling module

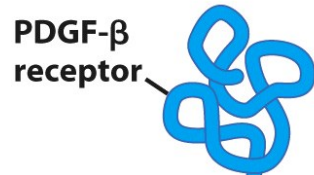


5 proteins, 20 interactions → 170,000 species

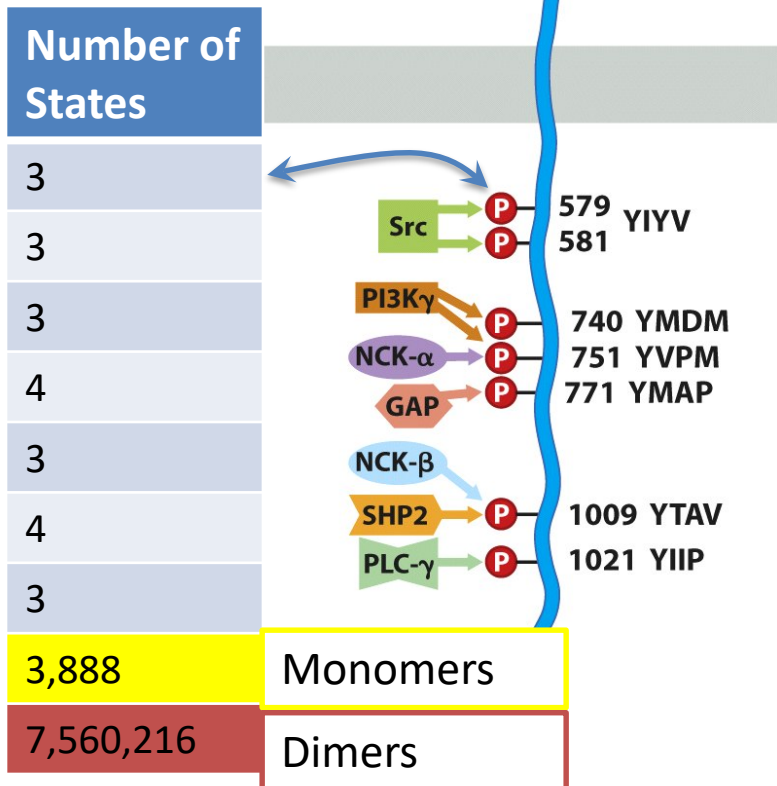
... but only 20 (or so) **rules**.

# Combinatorial Complexity

(A)



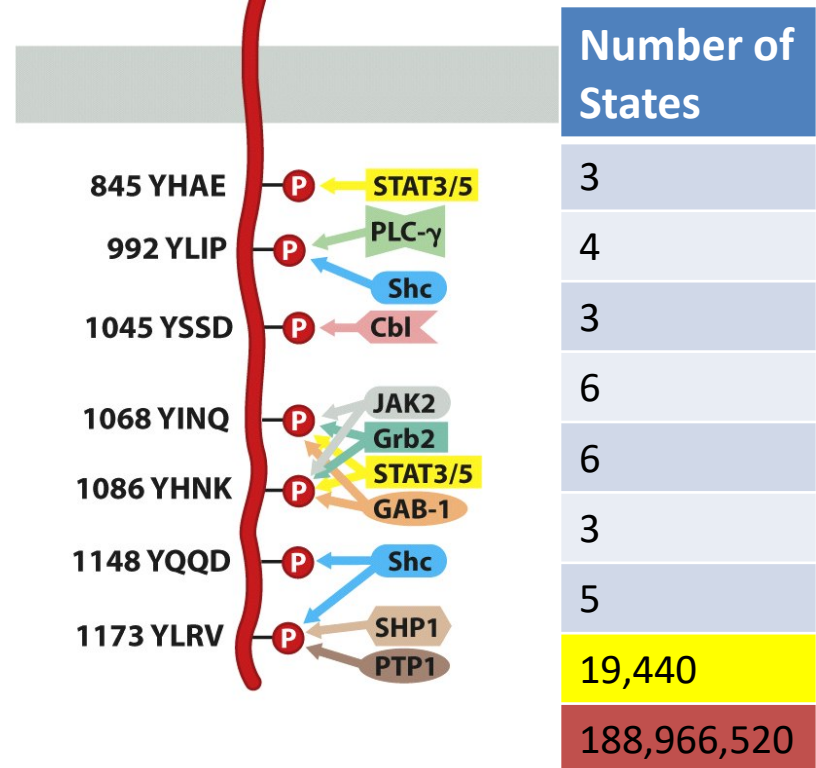
PDGF- $\beta$   
receptor



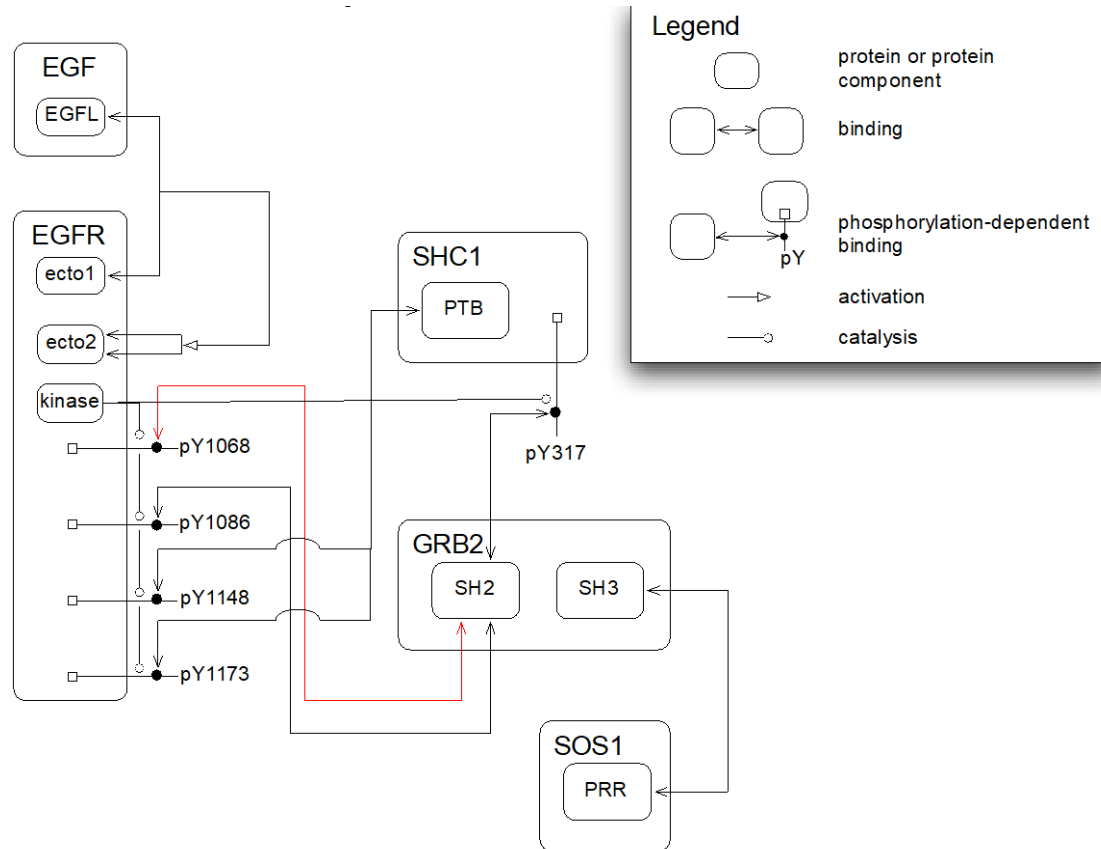
(B)



EGF  
receptor



# Rules provide a scalable way to model molecular interactions



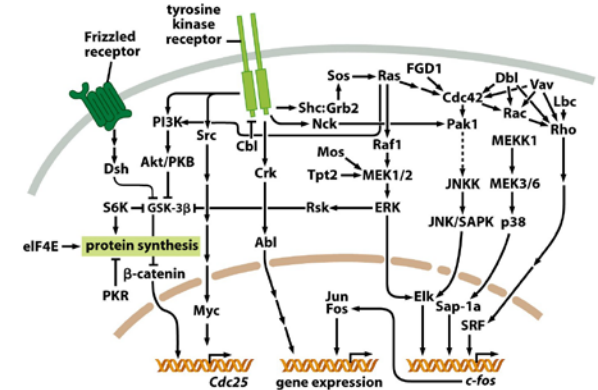
Rules  $\sim$  number of interactions  $\ll$  number of species

# Rule-Based Modeling: An Intermediate Level Abstraction for Systems Biology

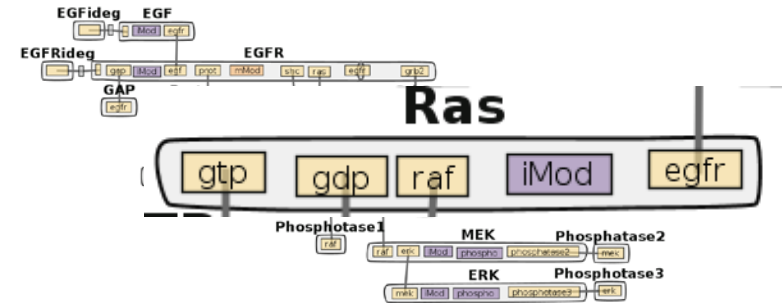
abstraction level

Reaction Networks

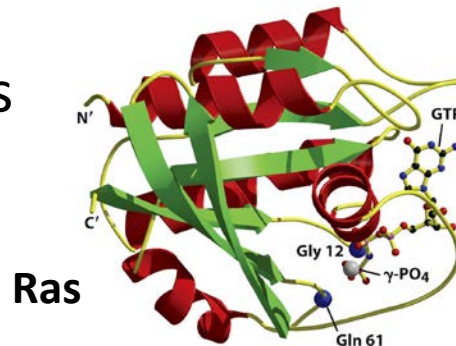
differential equations



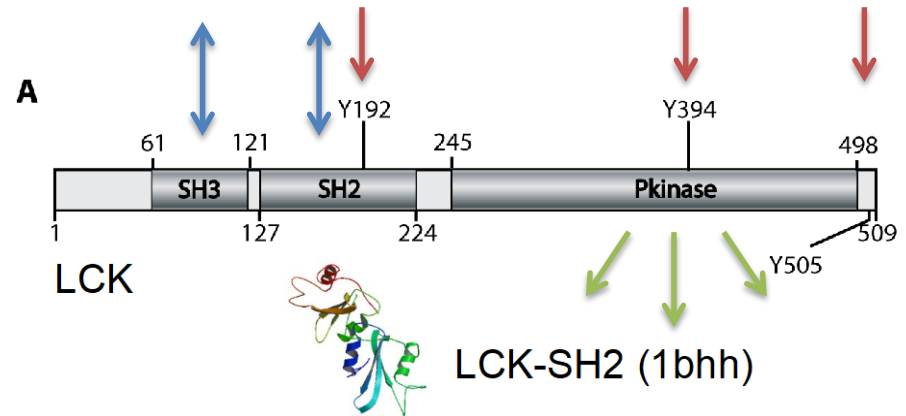
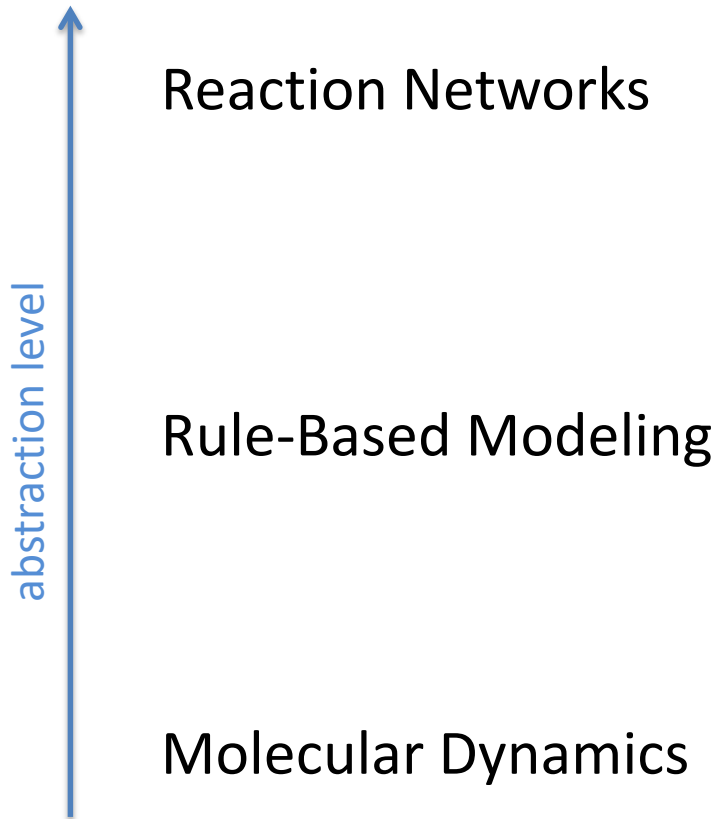
Rule-Based Modeling



Molecular Dynamics



# Rule-Based Modeling: An Intermediate Level Abstraction for Systems Biology

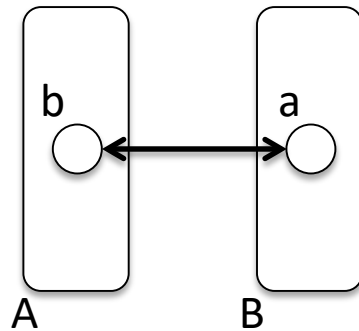


Abstraction that matches signaling knowledge.



# Rules Describe *Local* Interactions

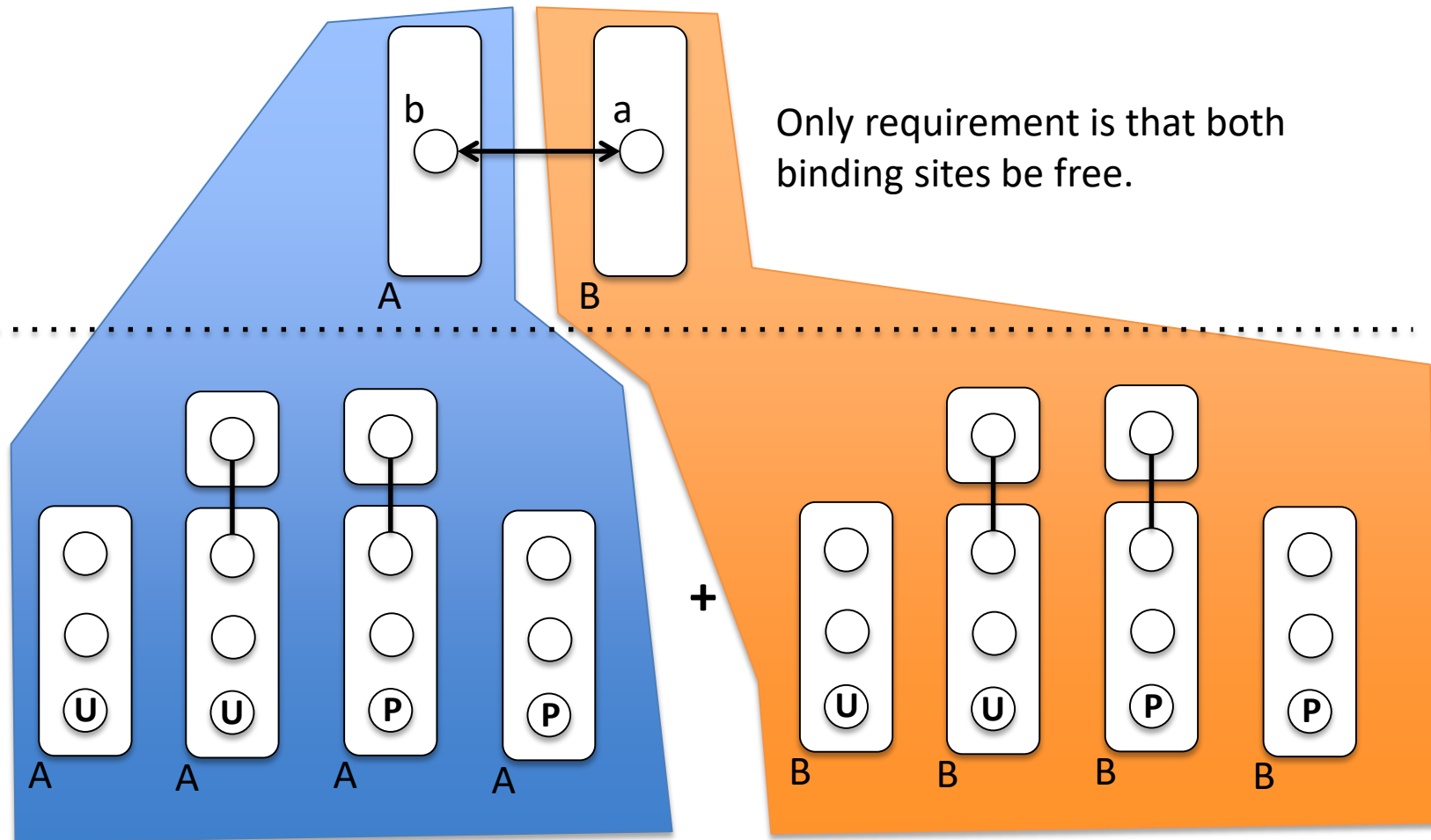
## Simple Binding/Unbinding Rule



Only requirement is that both binding sites be free.

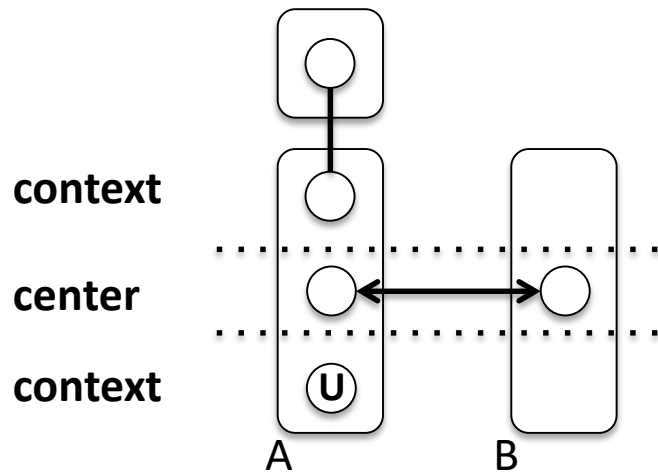
# One Rule May Generate Many Reactions

## Simple Binding/Unbinding Rule



# Rules Have Two Parts

Binding/Unbinding Rule with context



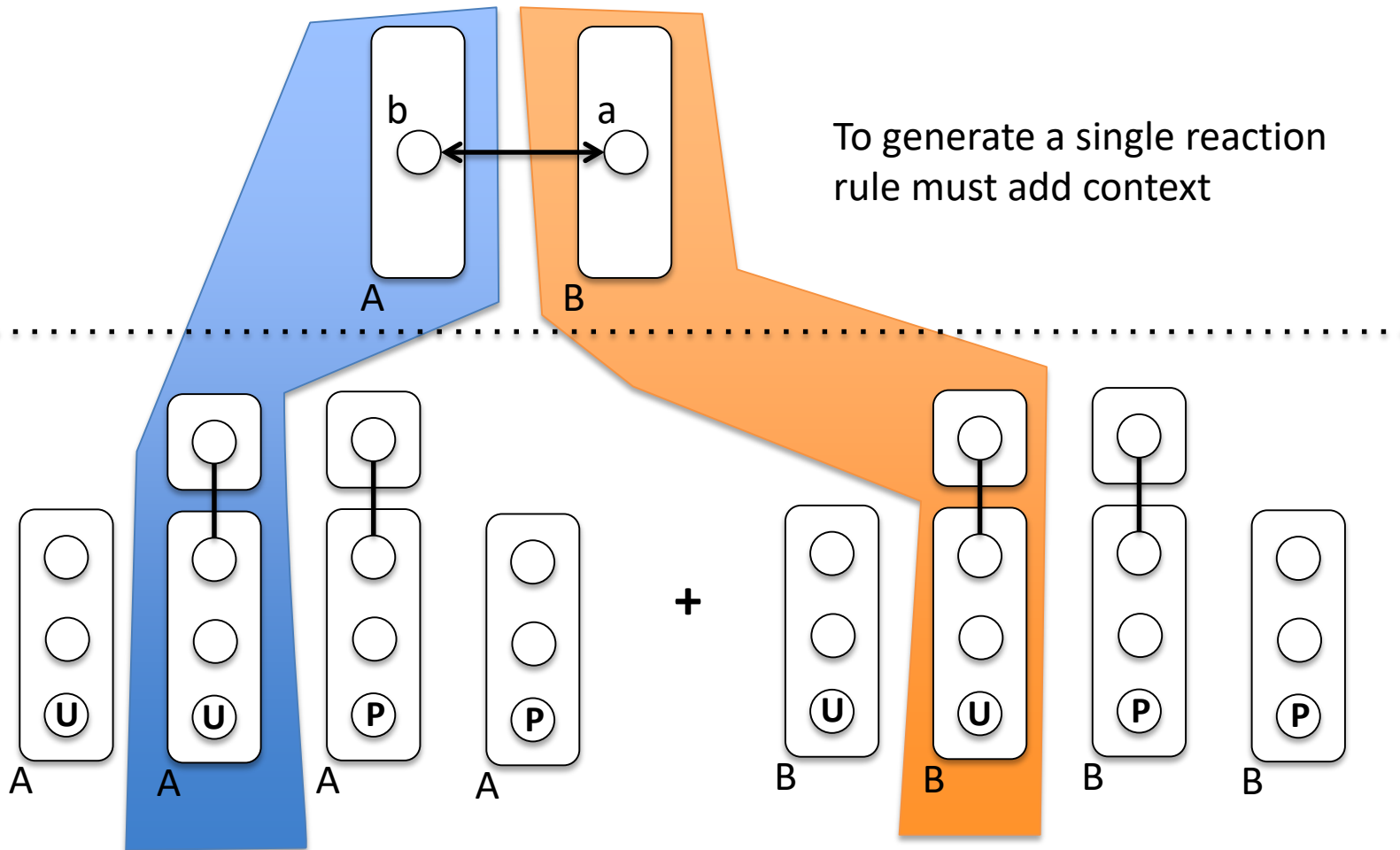
Binding now requires additional properties of A.

Simple rules generate more reactions and species – possible combinations of A and B.

*More specific rules require additional knowledge, e.g., cooperative or allosteric effects.*

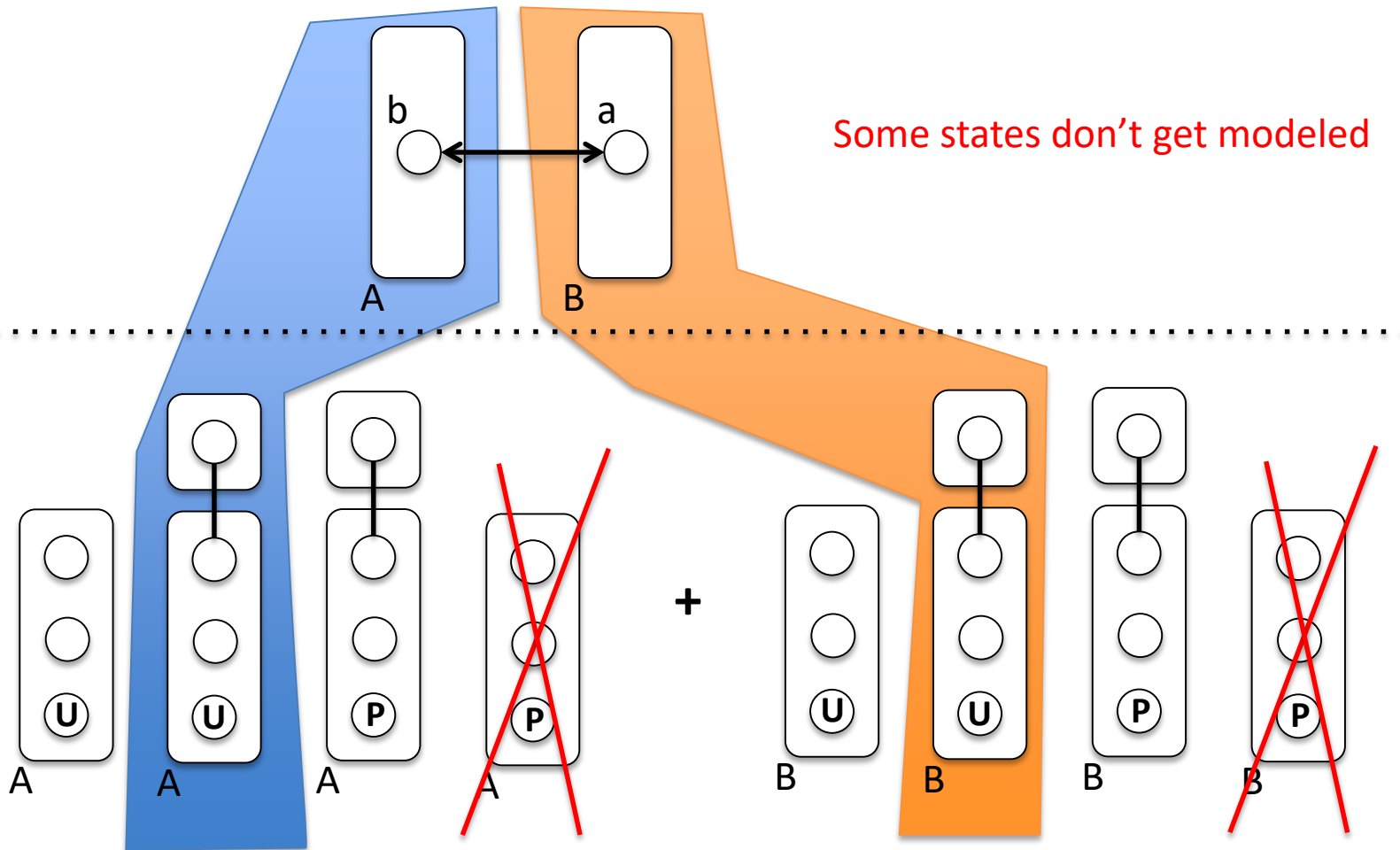
# Standard Approach Involves Hidden Assumptions

## Simple Binding/Unbinding Rule

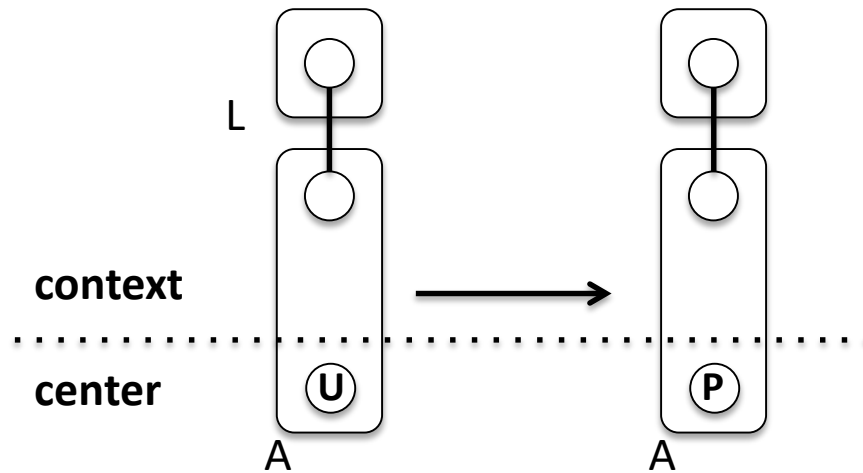


# Standard Approach Involves Hidden Assumptions

Simple Binding/Unbinding Rule



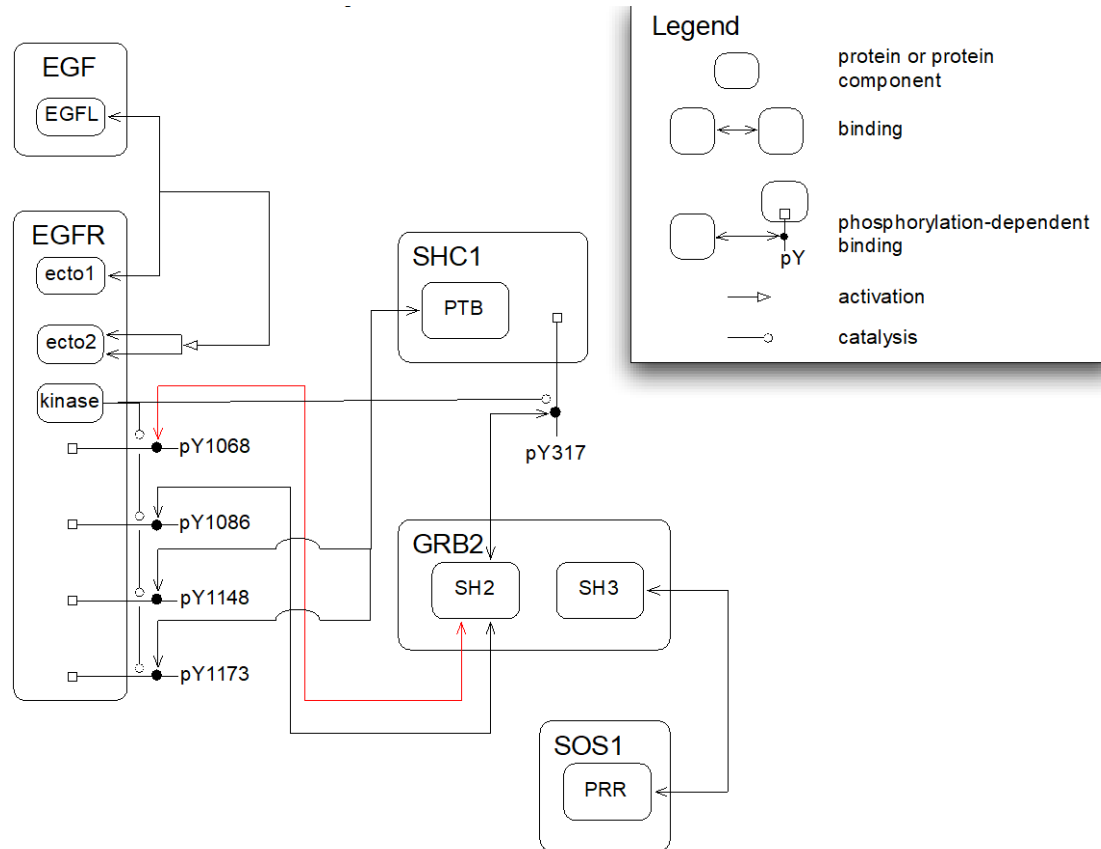
# Phosphorylation Rule with Context



L binding is required for phosphorylation



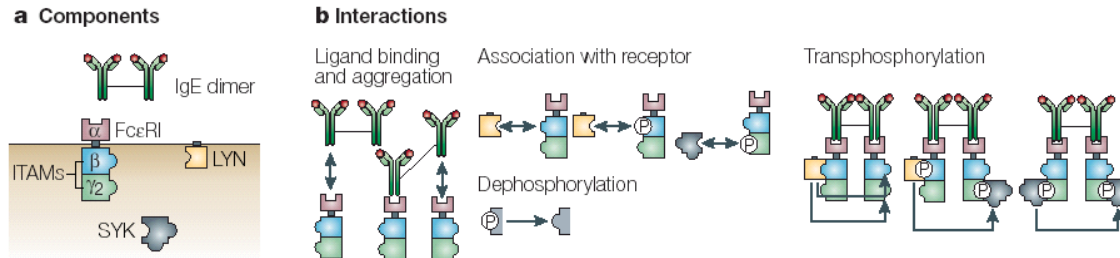
# Rules provide a scalable way to model molecular interactions



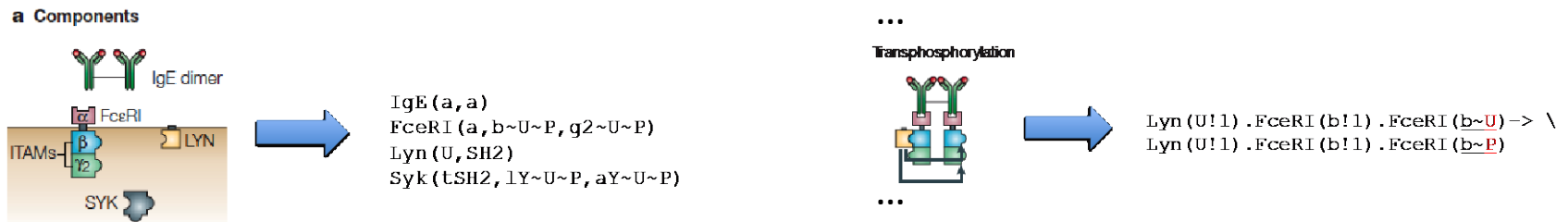
Rules  $\sim$  number of interactions  $\ll$  number of species

# Rule-Based Modeling protocol

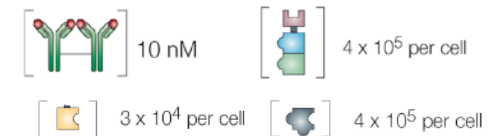
## 1. Identify components and interactions.



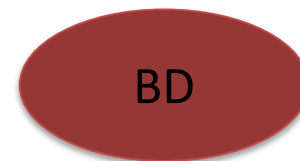
## 2. Translate into objects (molecules) and rules



## 3. Determine concentrations and rate constants

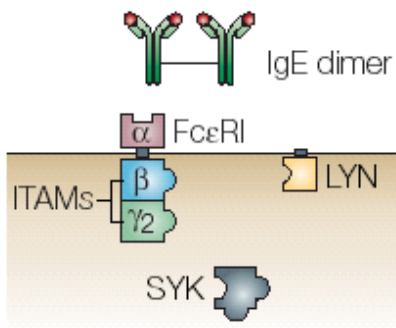


## 4. Simulate and analyze the model

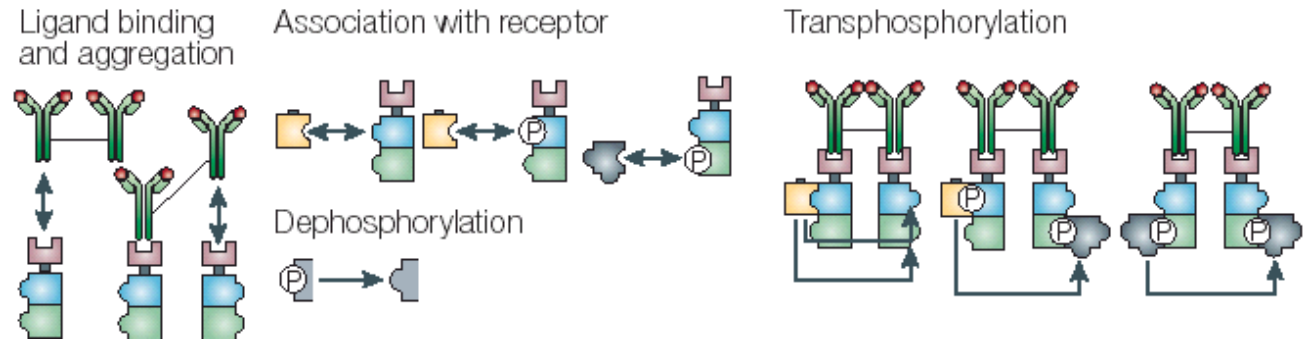


# Composition of a Rule-Based Model

## a Components



## b Interactions



## Molecules

```
begin molecules
Lig(1,1)
Lyn(U,SH2)
Syk(tSH2,1~U~P,a~U~P)
Rec(a,b~U~P,g~U~P)
end molecules
```

## Reaction Rules

```
begin reaction_rules
# Ligand-receptor binding
1 Rec(a) + Lig(1,1) <-> Rec(a!1).Lig(1!1,1) kp1, km1
  Rec(a) + Lig(1,1) <-> Rec(a!1).Lig(1!1,1) kp1, km1

# Receptor-aggregation
2 Rec(a) + Lig(1,1!1) <-> Rec(a!2).Lig(1!2,1!1) kp2,km2

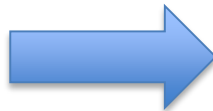
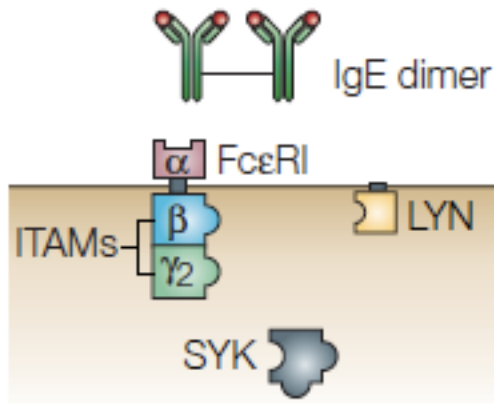
# Constitutive Lyn-receptor binding
3 Rec(b~Y) + Lyn(U,SH2) <-> Rec(b~Y!1).Lyn(U!1,SH2) kpL, kmL
...
```

## BioNetGen language

# **SPECIFYING A RULE-BASED MODEL**

# Defining Molecules

**Molecules** are the basic objects in a BNG model



## BIONETGEN Language

IgE ( a , a )

FcεRI ( a , b~U~P , g2~U~P )

Lyn ( U , SH2 )

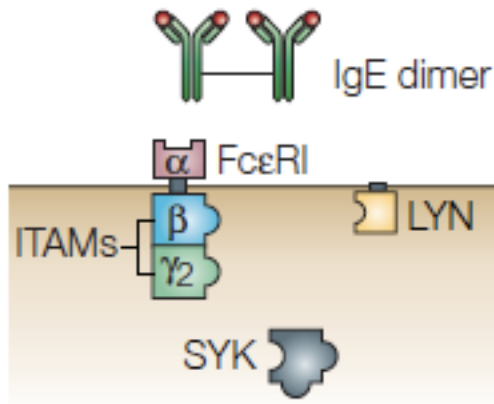
Syk ( tSH2 , lY~U~P , aY~U~P )

**Components** represent molecule elements

- Domains
- Motifs
- Properties

# Defining Molecules

**Molecules** are the basic objects in a BNG model



## BIONETGEN Language

```
IgE ( a , a )  
FceRI ( a , b~U~P , g2~U~P )  
Lyn ( U , SH2 )  
Syk ( tSH2 , lY~U~P , aY~U~P )
```

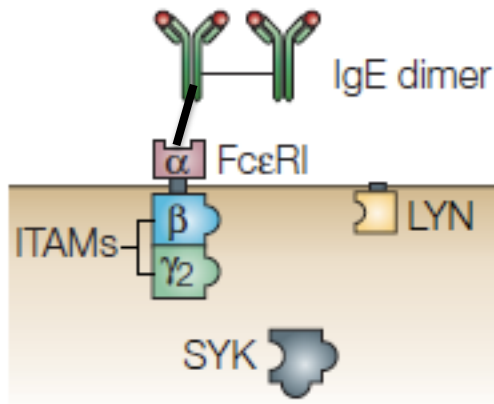
**Components** may have different **states** representing

- posttranslational modifications
- conformational state
- ...

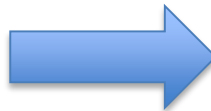


# Binding

**Molecules** bind other molecules through components



## BIONETGEN Language



`IgE(a, a!1) . FceRI(a!1, b~U, g2~U)`

**Bonds** are formed by linking two components. The '.' indicates a set of molecules forming a complex.

`FceRI(a, b~U!1, g2~U) . Lyn(U!1)`

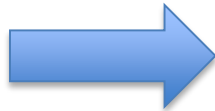
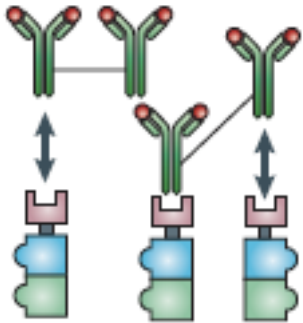
Components may have both states and bonds.

`Lyn(SH2!1, Cterm~P!1)`

Bonds may occur within a molecule.

# Defining Interaction Rules

Ligand binding  
and aggregation

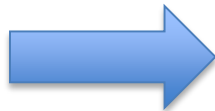
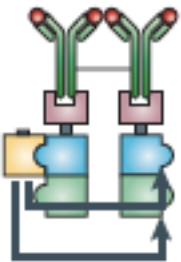


BIONETGEN Language

```
IgE(a, a) + FceRI(a) <-> IgE(a, a!1) . FceRI(a!1)  
...
```

binding and dissociation

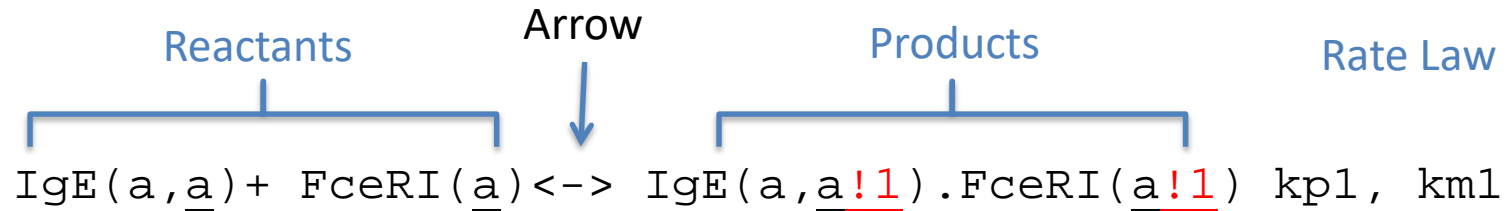
Transphosphorylation



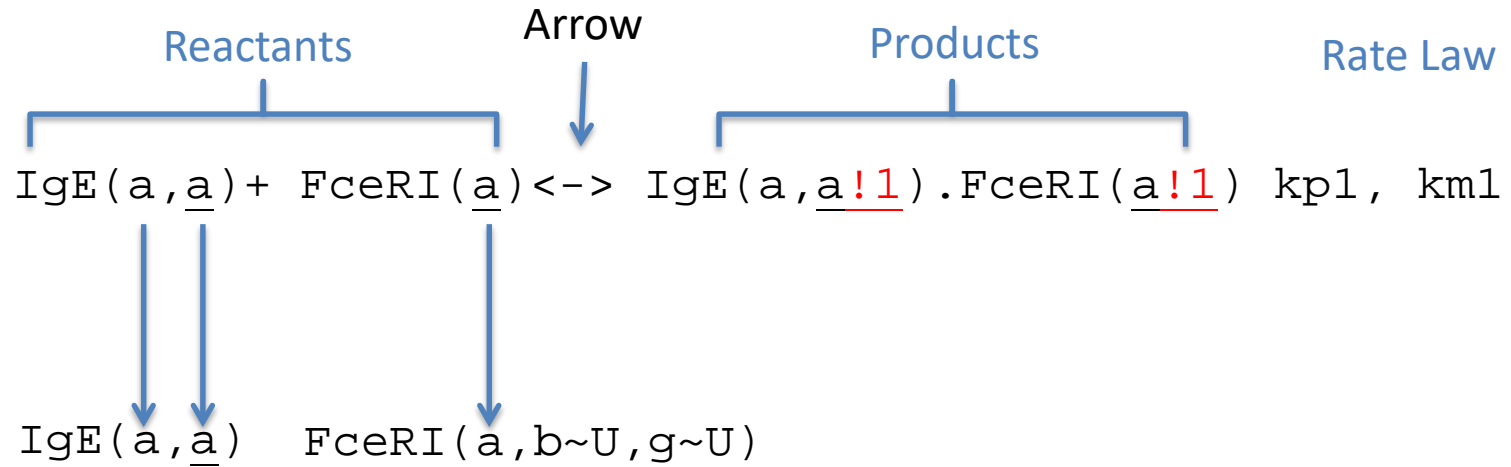
```
Lyn(U!1) . FceRI(b!1) . FceRI(b~U) -> \  
Lyn(U!1) . FceRI(b!1) . FceRI(b~P)
```

component state change

# Parts of a rule



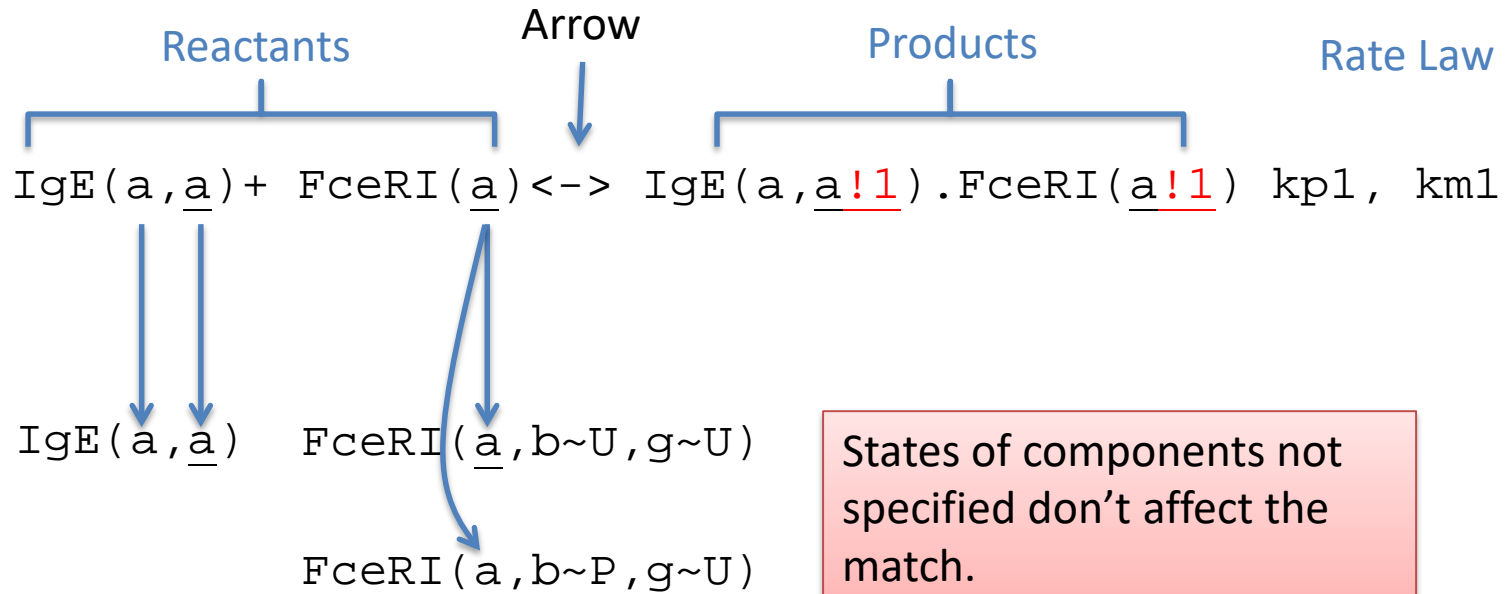
# Parts of a rule



## Reactant patterns

select properties of  
each reactant  
molecule.

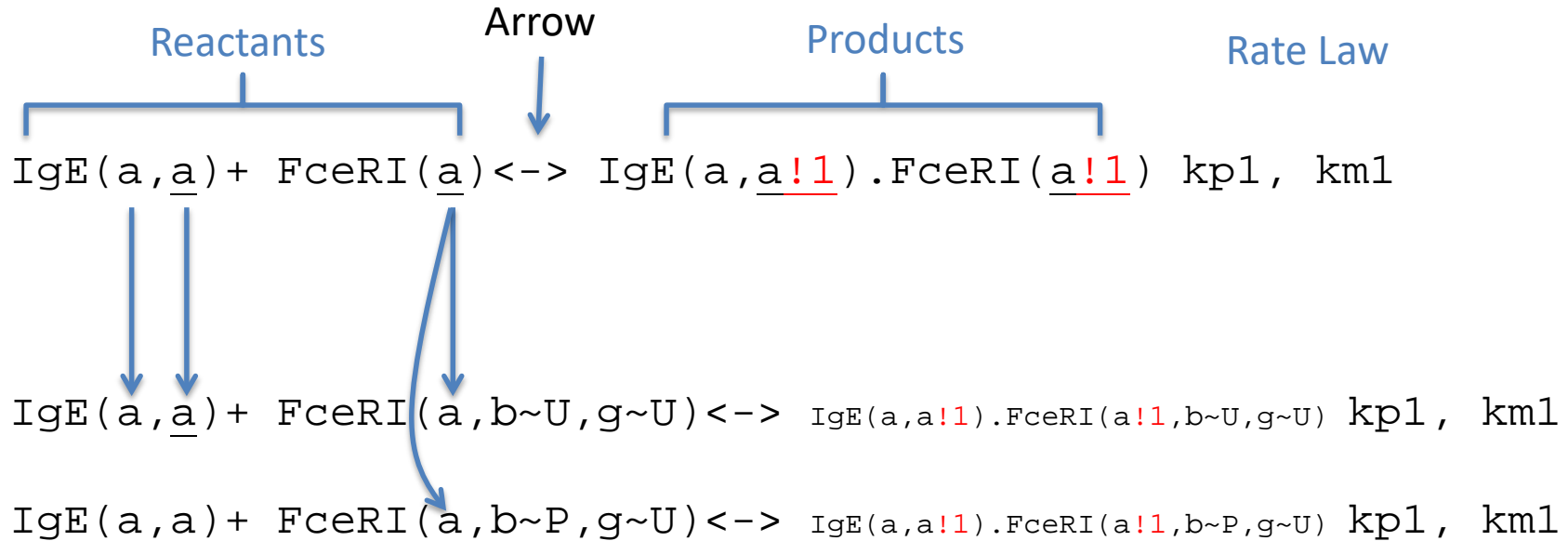
# Parts of a rule



States of components not specified don't affect the match.  
*"Don't write, don't care."*

**Reactant patterns**  
 select properties of  
 each reactant  
 molecule.

# Parts of a rule



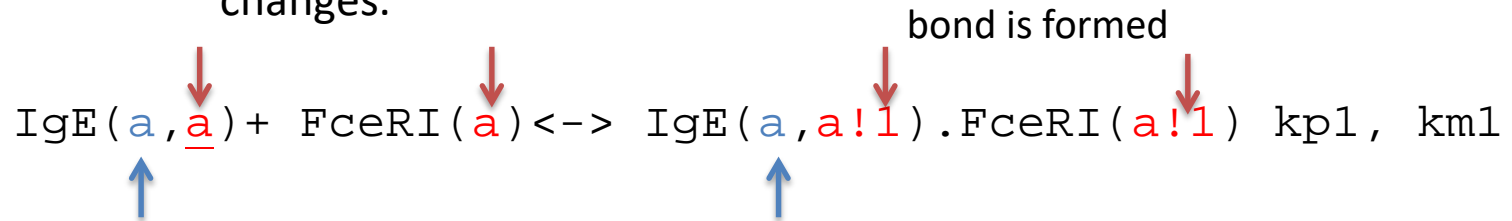
**Reactant patterns**  
select properties of  
each reactant  
molecule.

Because patterns can match  
many different species,  
each rule can generate  
many reactions.



# Center and context

The **center** of a rule is the part that the rule changes.

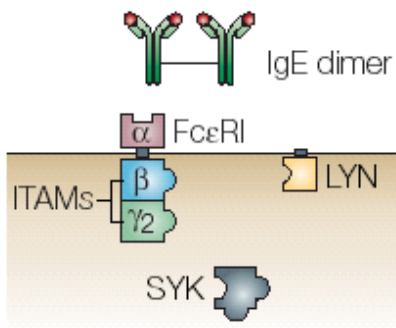


The **context** is the part that is necessary for the rule to happen but is unchanged.

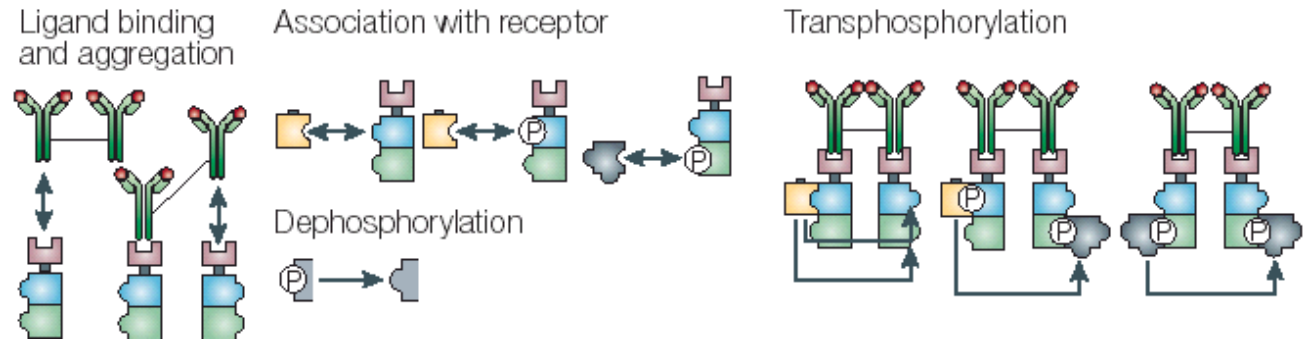


# Composition of a Rule-Based Model

## a Components



## b Interactions



## Molecules

```
begin molecules
Lig(1,1)
Lyn(U,SH2)
Syk(tSH2,1~U~P,a~U~P)
Rec(a,b~U~P,g~U~P)
end molecules
```

## Reaction Rules

```
begin reaction_rules
# Ligand-receptor binding
1 Rec(a) + Lig(1,1) <-> Rec(a!1).Lig(1!1,1) kp1, km1
  Rec(a) + Lig(1,1) <-> Rec(a!1).Lig(1!1,1) kp1, km1

# Receptor-aggregation
2 Rec(a) + Lig(1,1!1) <-> Rec(a!2).Lig(1!2,1!1) kp2,km2

# Constitutive Lyn-receptor binding
3 Rec(b~Y) + Lyn(U,SH2) <-> Rec(b~Y!1).Lyn(U!1,SH2) kpL, kmL
...
```

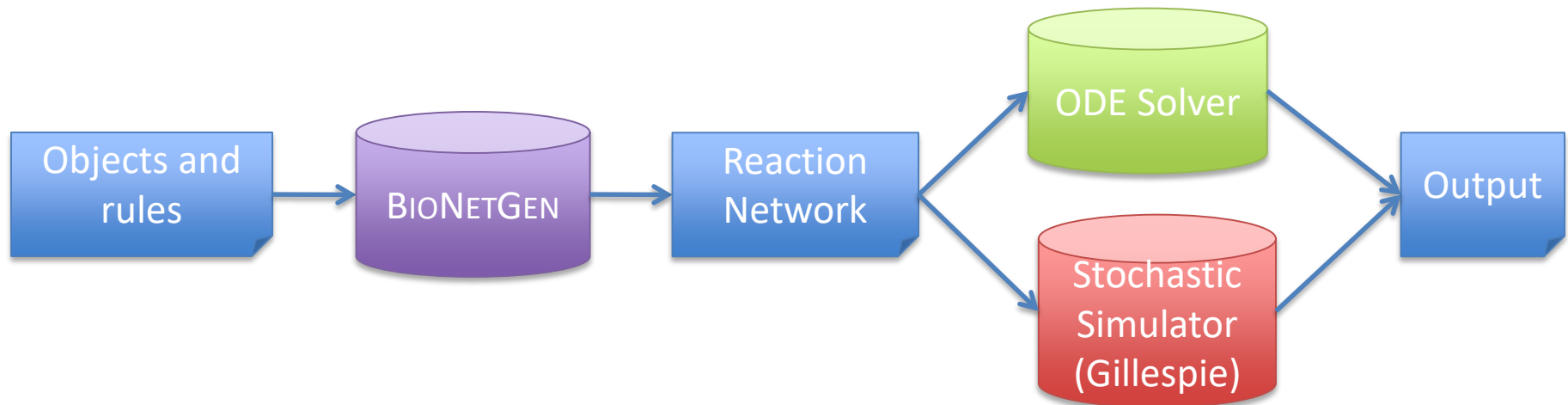
## BioNetGen language

# Applications

- Immunoreceptor Signaling
- Growth factor receptor signaling
- Multivalent binding
- Scaffold effects
- Yeast pheromone signaling
- For a complete list of BioNetGen Applications see [http://bionetgen.org/Model\\_Examples](http://bionetgen.org/Model_Examples).

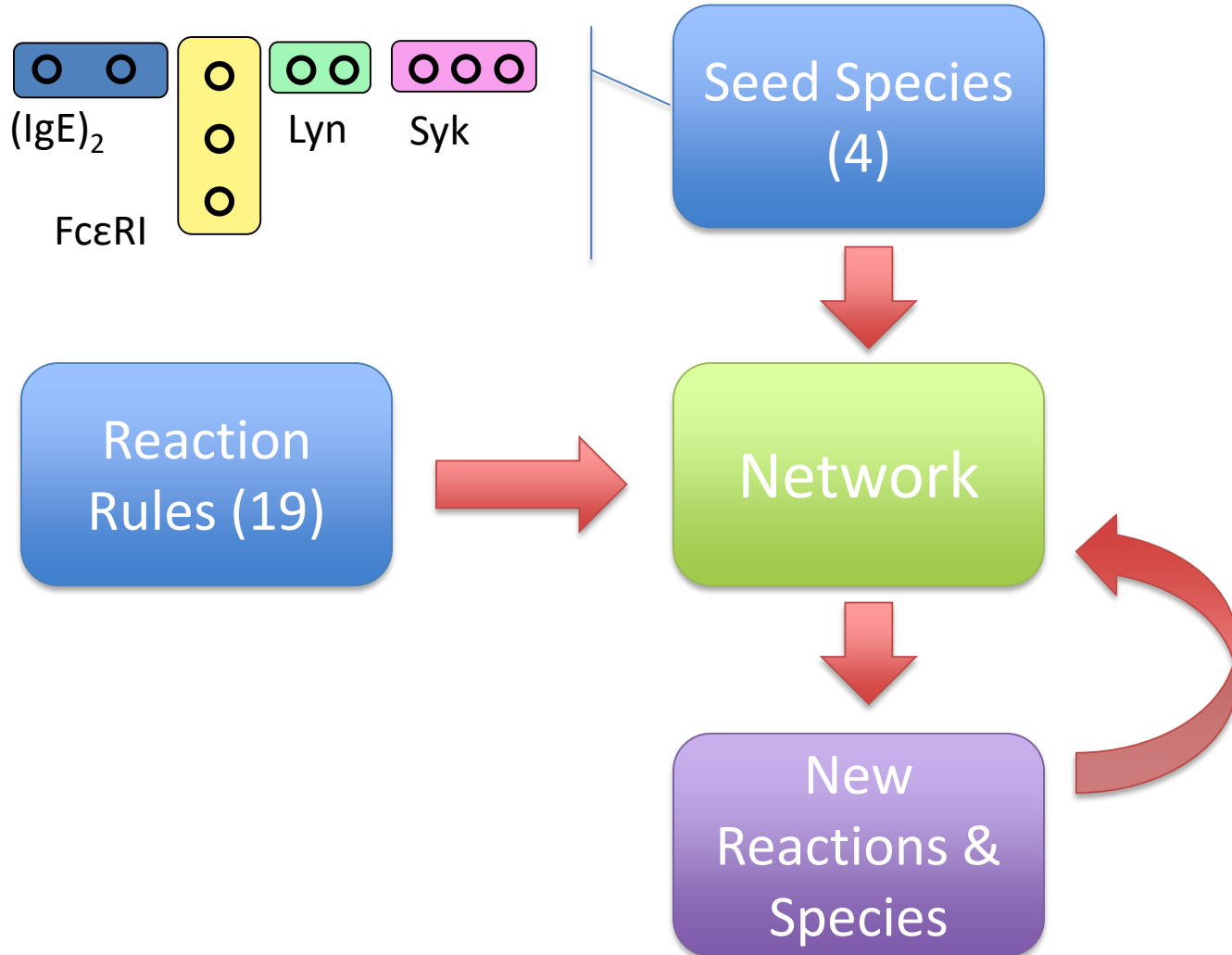
# **SIMULATING A RULE-BASED MODEL**

# Basic RBM workflow with BioNetGen



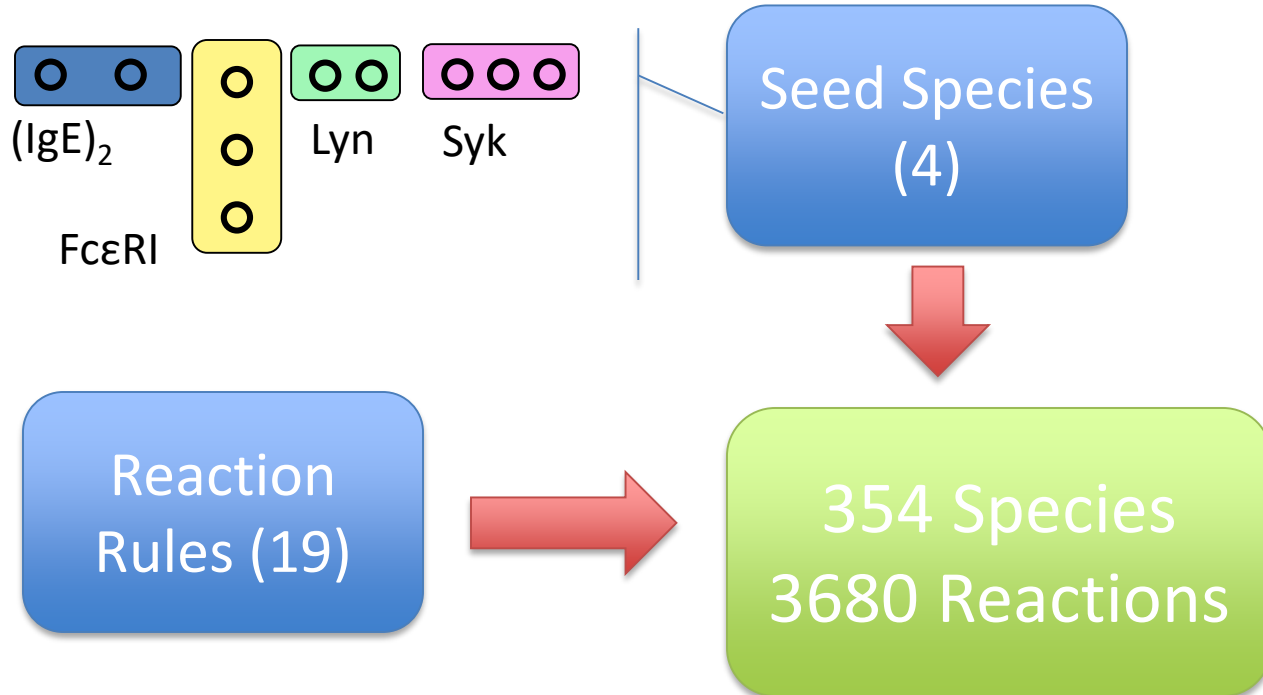
# Automatic Network Generation

## FcεRI Model



# Automatic Network Generation

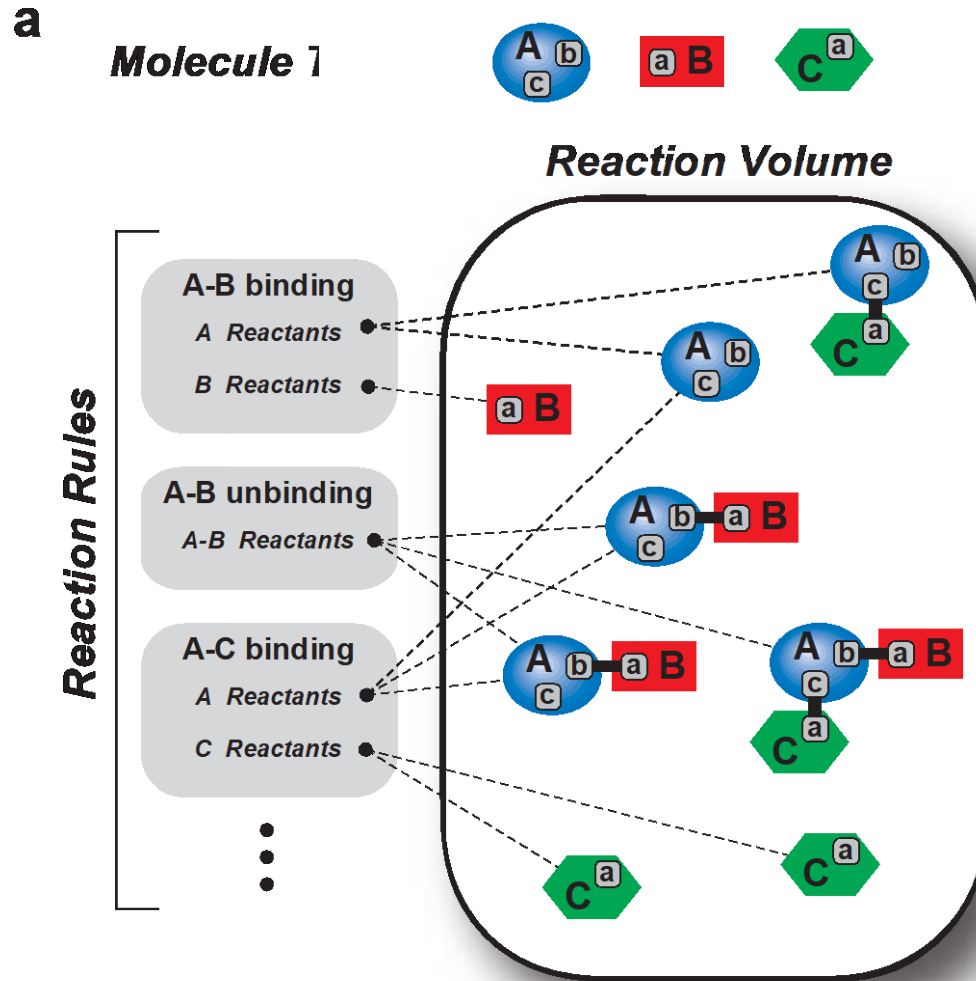
## FcεRI Model





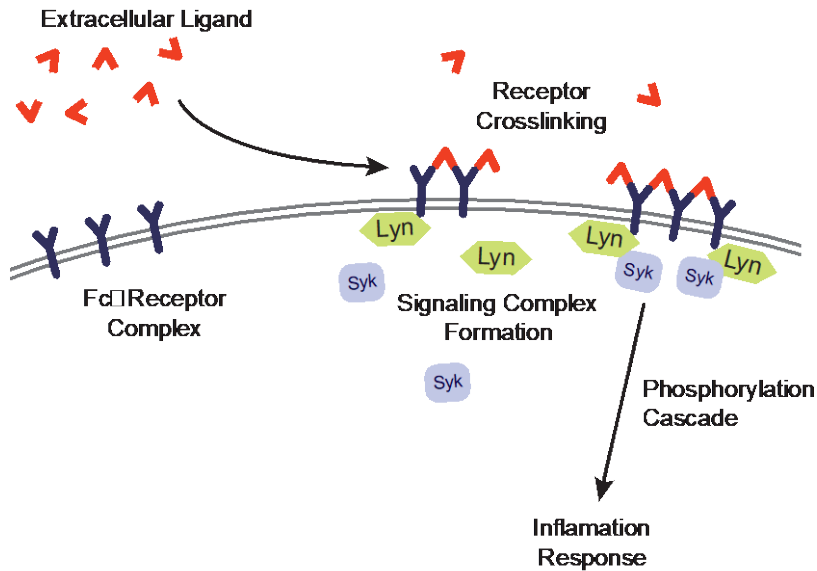
# NFSIM\*

## Network-Free Stochastic Simulator

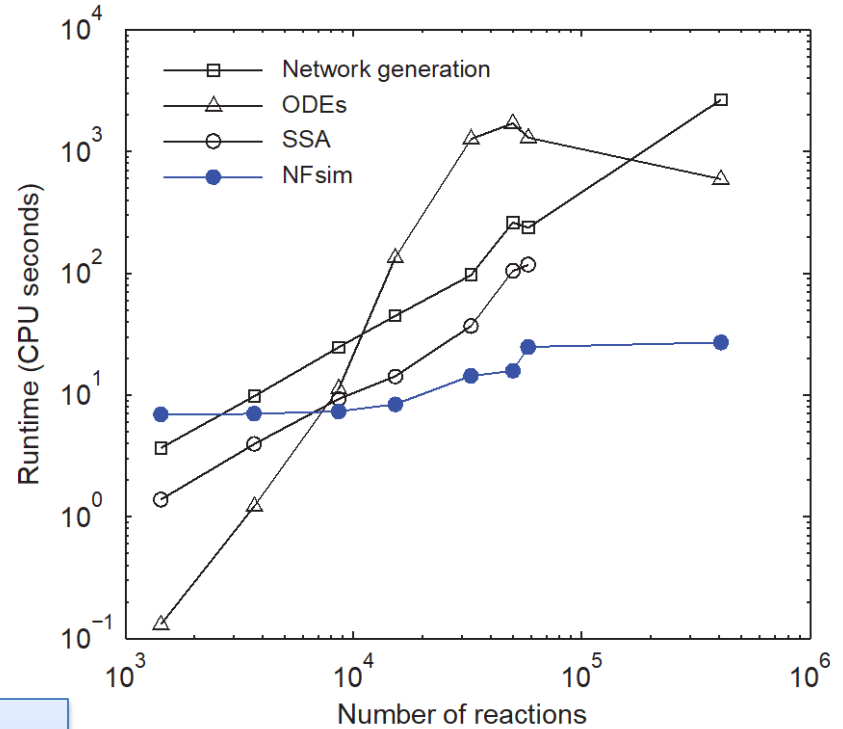


# FcεRI signaling models

a



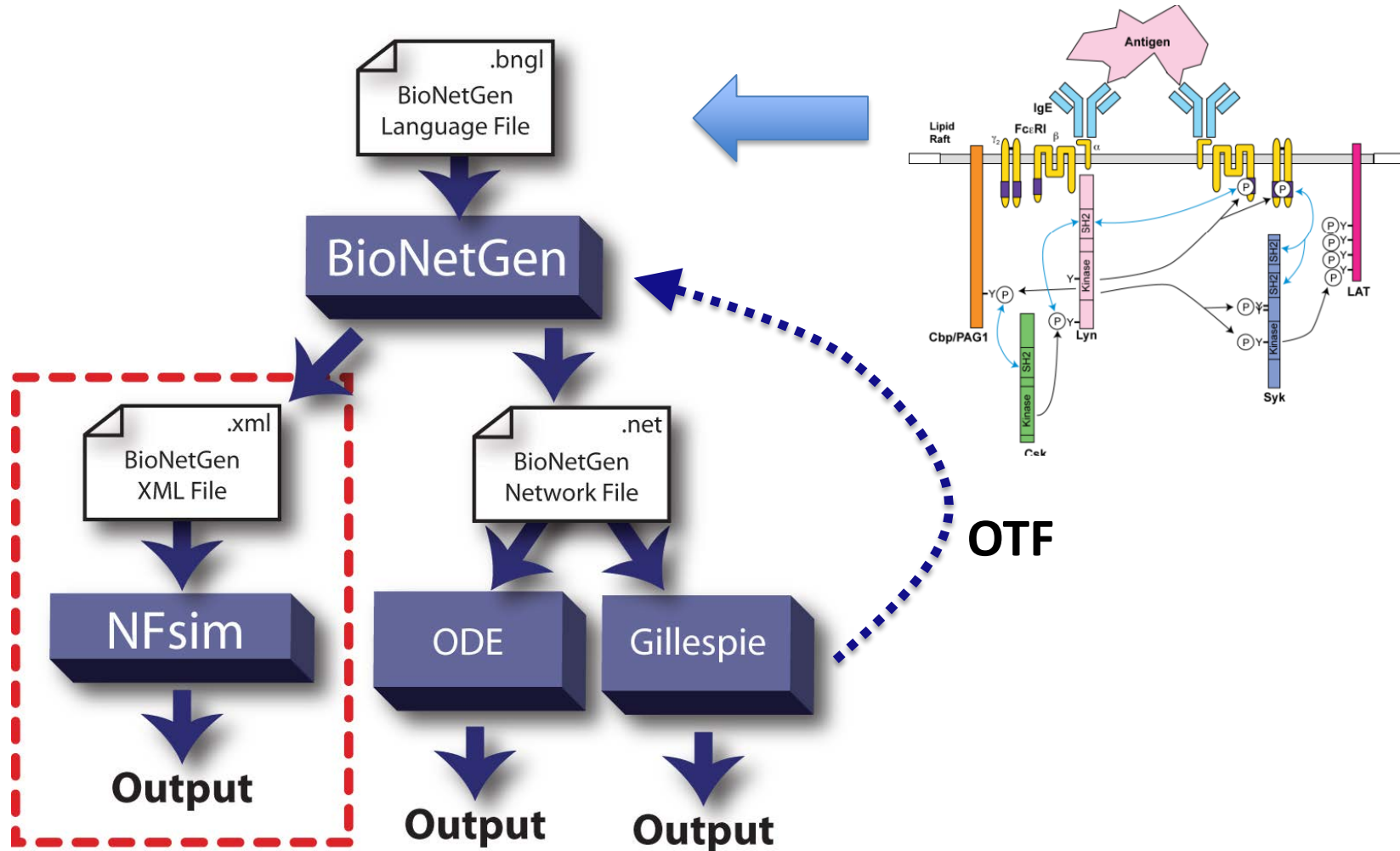
b



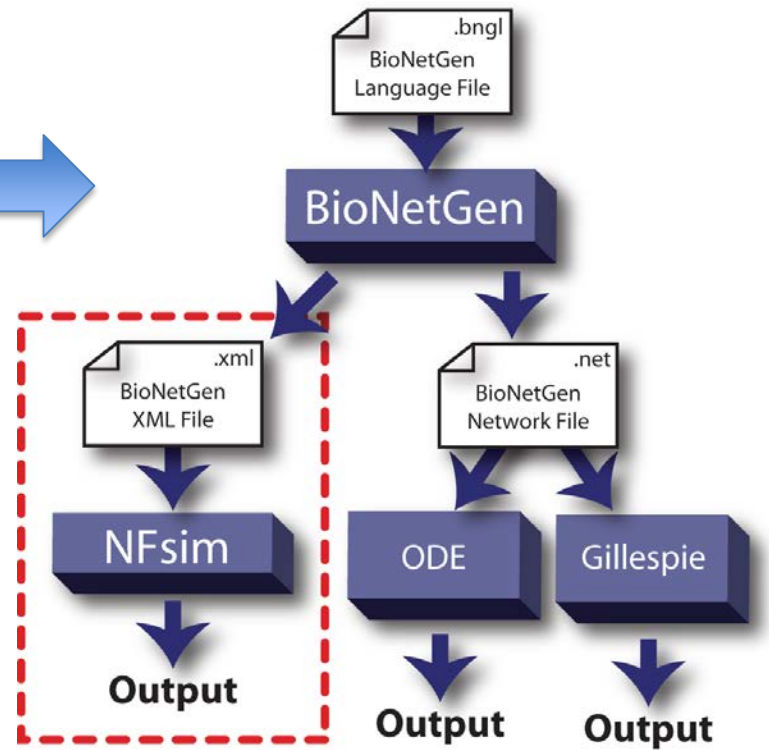
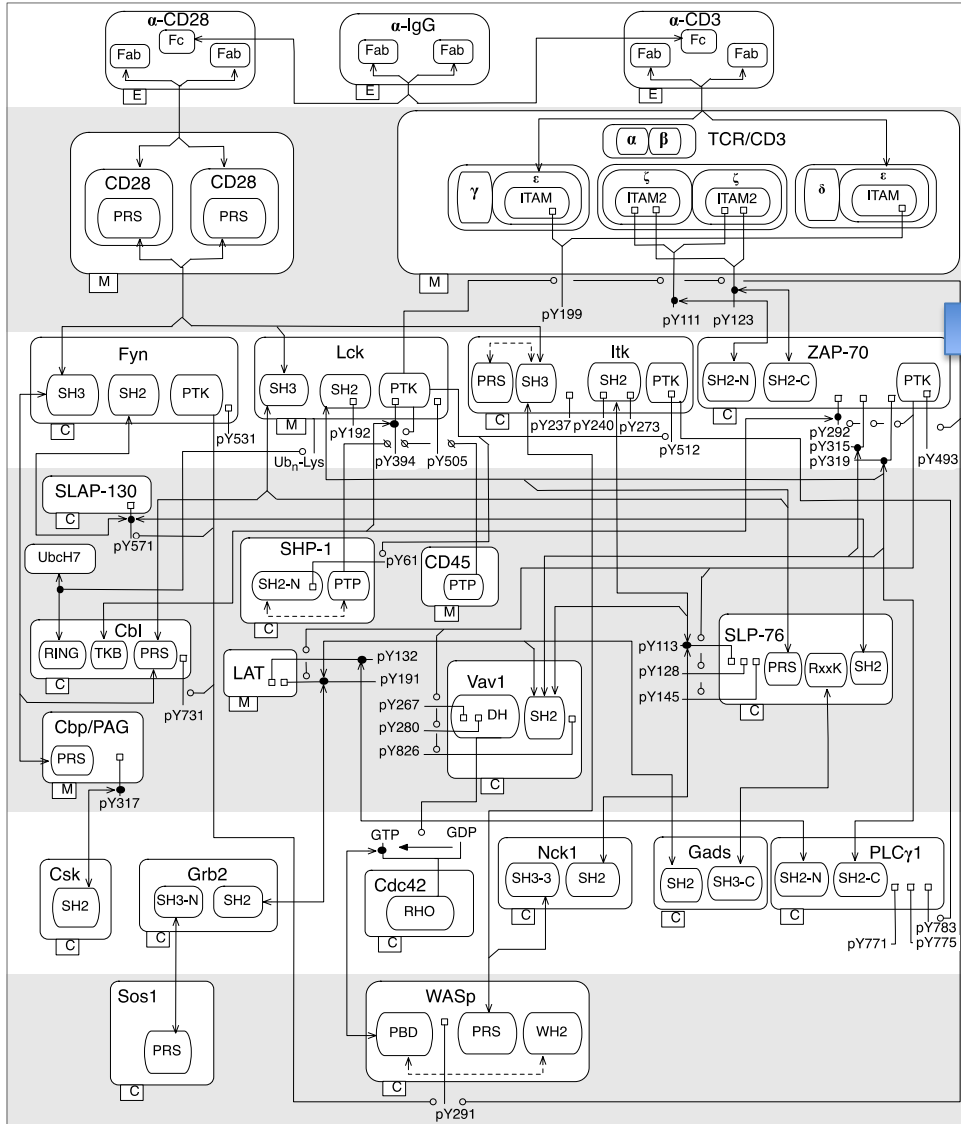
NFsim can simulate models of greatly increased complexity with manageable increase in cost.

→  
Increasing complexity

# Integration with BIO.NETGEN



# Large Scale TCR Signaling Model



Hu, Chylek, and Hlavacek, unpublished.

# RuleBender

Built in Eclipse RCP

<http://rulebender.org>

The screenshot displays the RuleBender application window, which is built on the Eclipse RCP. The interface is divided into several panes:

- Model Pane:** Shows the BNGL model file `egfr_net.bngl`. The model contains several reaction rules, including dephosphorylation, Shc transphosphorylation, and Y1068 activity. The rule `egfr(Y1068~pY) + Grb2(SH2,SH3) <-> egfr(Y1068~pY!1).Grb2(SH2!1,SH3!2) <-> egfr(Y1068~pY!1).Grb2(SH2!1,SH3!2) + Sos(dom) <-> egfr(Y1068~pY!1).Grb2(SH2!1,SH3)` is highlighted in green.
- Contact Map Pane:** Displays a network diagram showing interactions between proteins: `egfr`, `Shc`, `Grb2`, and `Sos`. Specific phosphorylation sites are labeled, such as `Y1068`, `Y1148`, `Y317`, `SH2`, and `SH3`.
- Properties Pane:** Shows the details of the selected rule, including the rule expression and label.
- Problems Pane:** Indicates a single error: `rule parameter_def failed pre...` in `egfr_net.bngl` at line 17, categorized as a BNGL Error.

Property	Value
Rule Expression	<code>egfr(Y1068~pY) + Grb2(SH2,SH3) &lt;-&gt;</code>
Rule Label	Rule11

Description	Resource	Path	Location	Type
rule parameter_def failed pre...	<code>egfr_net.bngl</code>	<code>/EGFR</code>	line 17	BNGL Error

# **HANDS-ON TUTORIAL**

# Technical Overview of the BioNetGen Language

parameters

(compartments)

molecule types

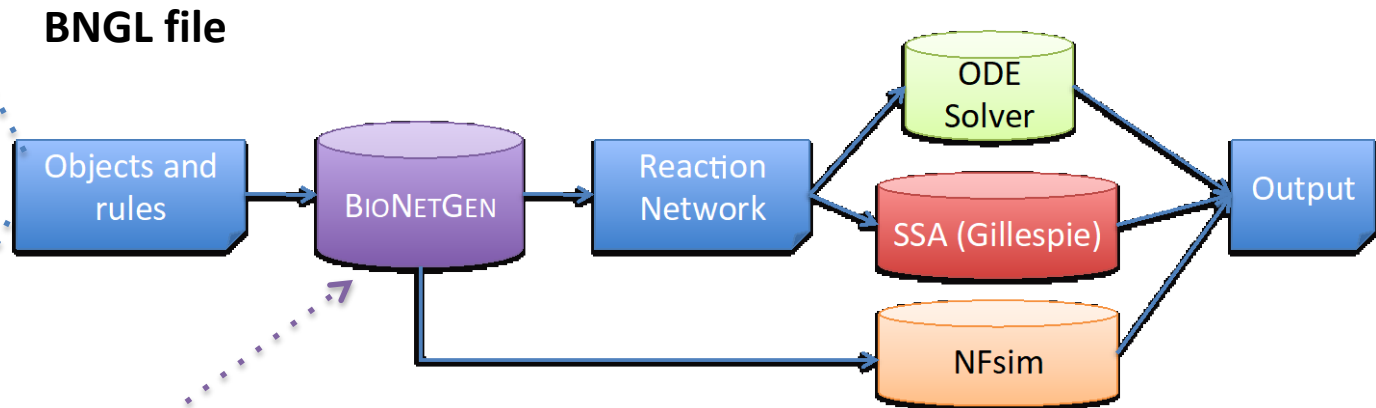
seed species

observables

(functions)

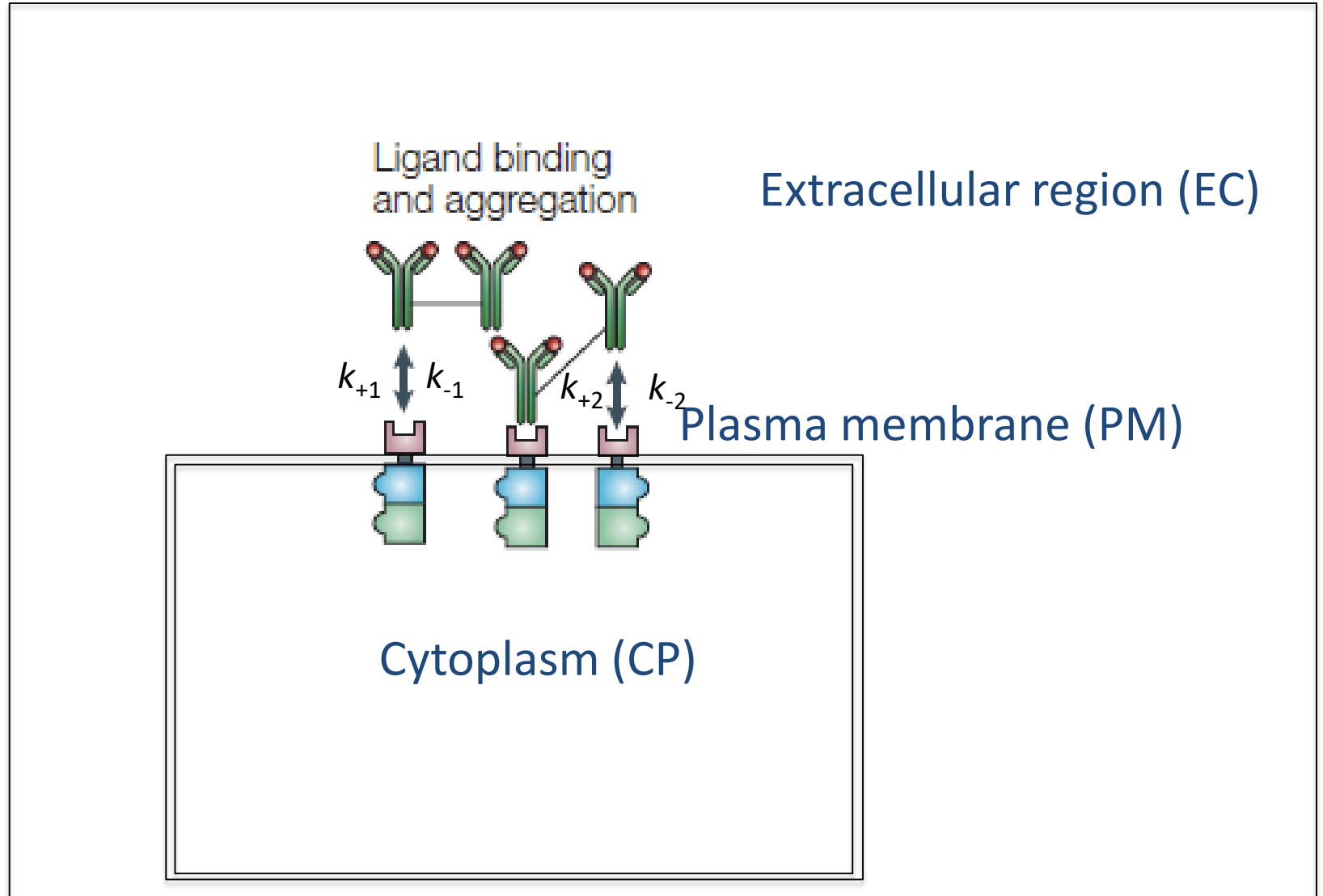
reaction rules

actions



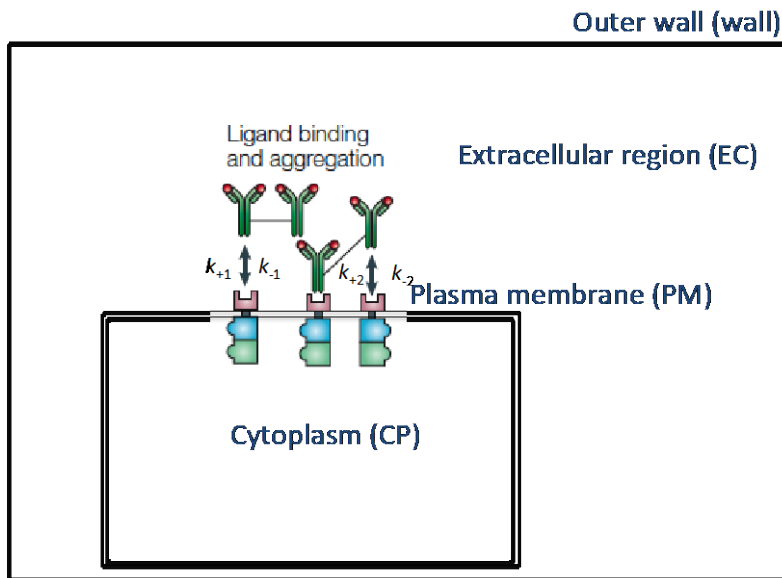
# Dimerization Model

Outer wall (wall)





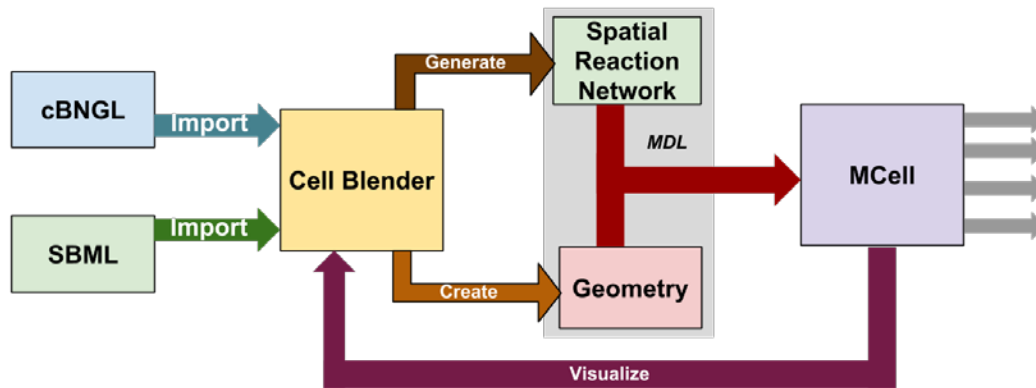
# Compartment Specification



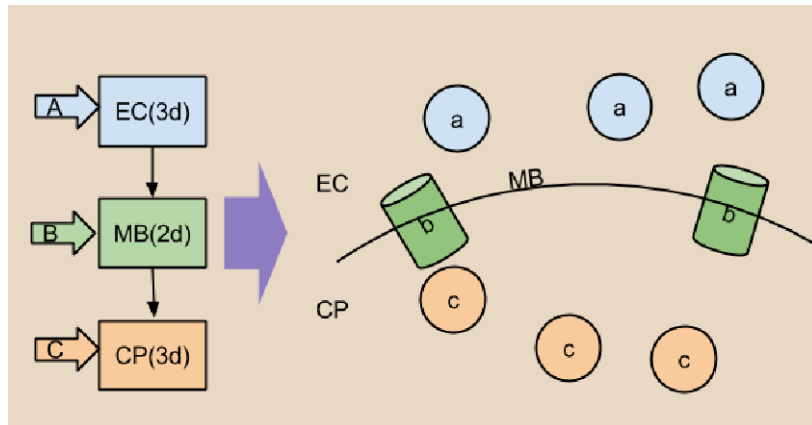
begin compartments  
wall 2 vol\_wall  
EC 3 vol\_EC wall  
PM 2 vol\_PM EC  
CP 3 vol\_CP PM  
end compartments

Volume of surface compartment = Area\*thickness  
thickness = 10 nm = 0.01  $\mu\text{m}$

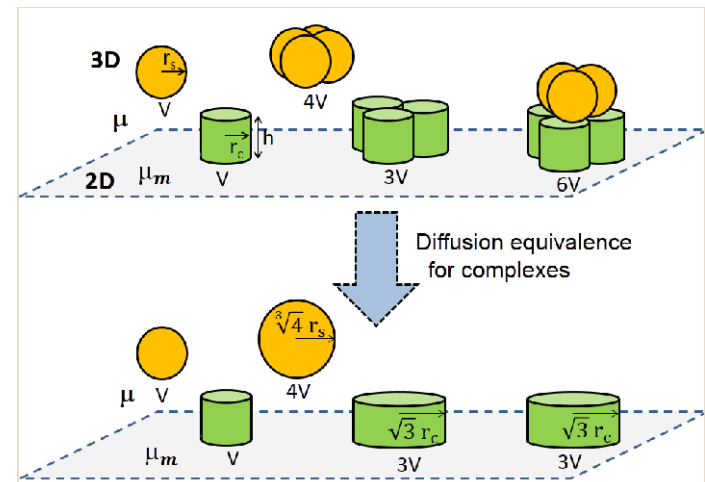
# Import of Rule-Based Models into MCell



## Topology



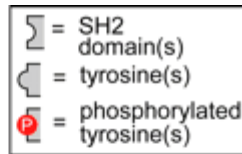
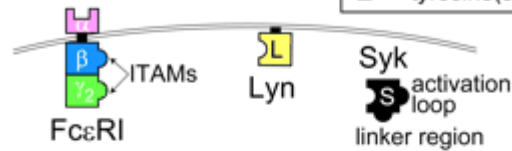
## Diffusion of complexes



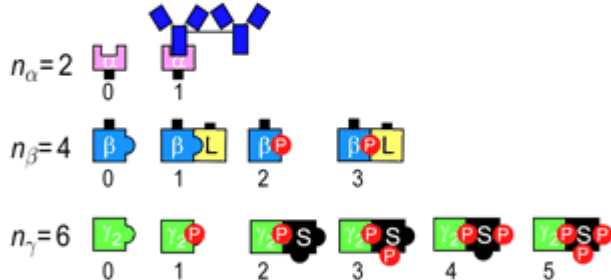
See Poster by Jose-Juan Tapia and Dipak Barua for more details

# Example: Comprehensive FcεRI signaling model

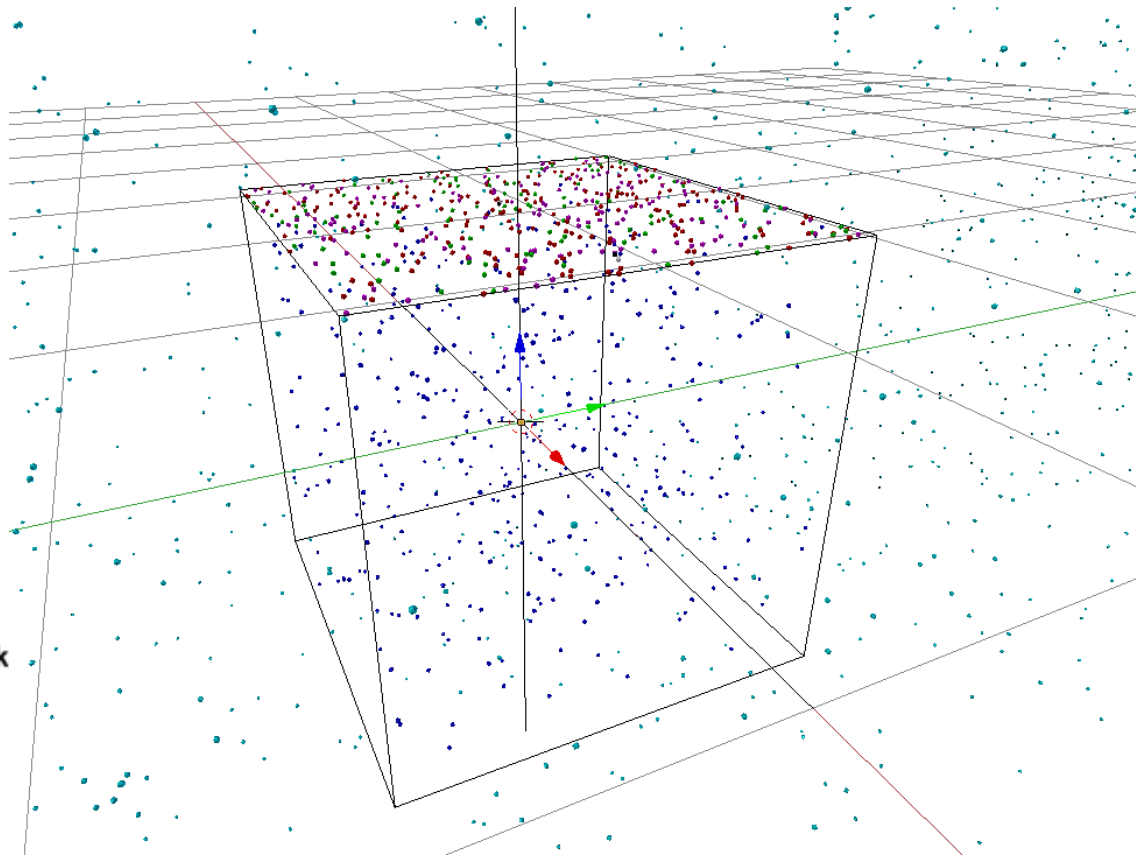
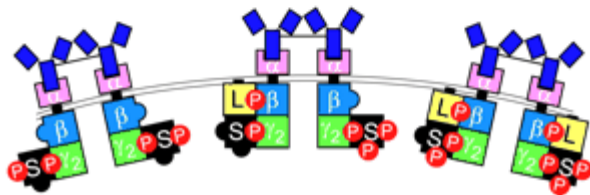
## a Components



## b Possible states of receptor subunits

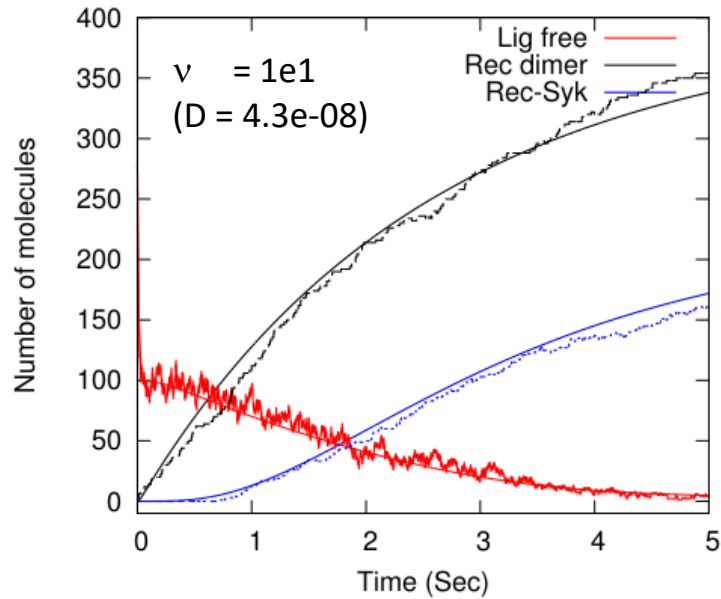


## c A few of the 164 dimer states with active Syk

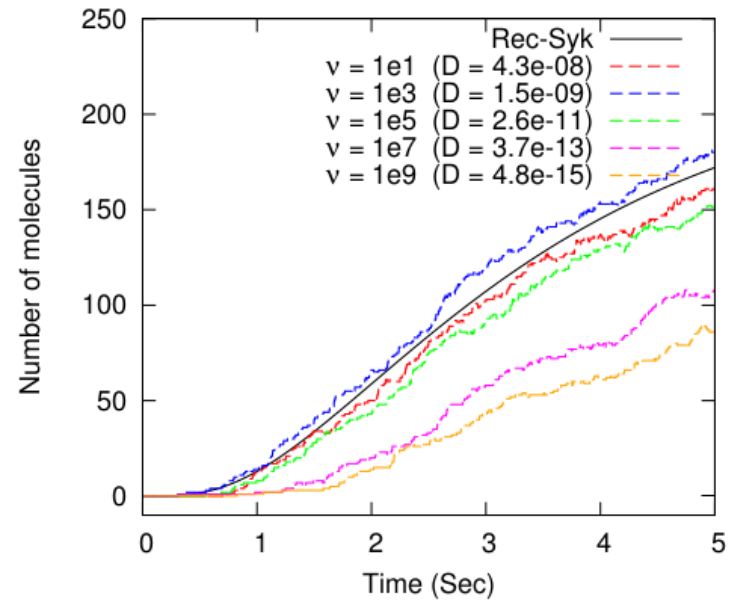


# Comprehensive FcεRI signaling model:

Solid lines - cBNG ODE simulations  
Broken lines – MCell simulations



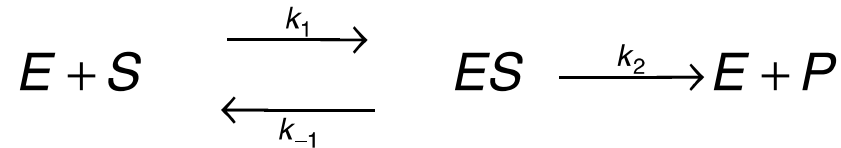
$\mu$  - Membrane viscosity, cp  
 $D$  - Diffusivity of single receptor, cm<sup>2</sup>/s  
(Saffman-Delbrück)



# **BACKUP EXAMPLE**

# Example 1: MM Mechanism

parameters



molecule types

seed species

A BioNetGen model consists of a set of blocks, each beginning and ending with `begin <blockname> / end <blockname>` respectively.

observables

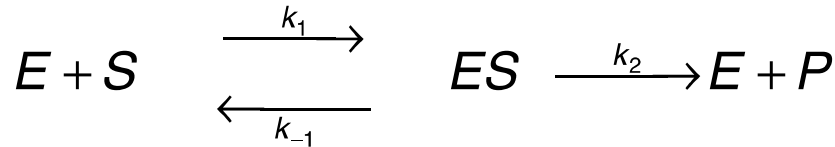
functions

reaction rules

actions

# Example 1: MM Mechanism

parameters



molecule types

parameters – model constants are defined here. *The user is responsible for using a consistent set of units, which should be indicated in the associated comments.*

seed species

observables

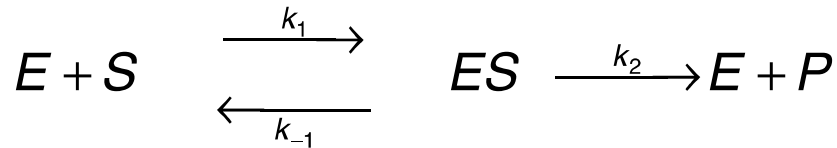
functions

reaction rules

actions

# Example 1: MM Mechanism

parameters



molecule types

```
begin parameters
# Avogadro's number- scaled for umol
NA 6.02e23/1e6
# Cell volume
V 1e-12 # liters - typical for eukaryote
# Rate constants
kp1 1.0/(NA*V) # 1/uM 1/s-> 1/molec 1/s
km1 1.0e-1 # 1/s
k2 1.0e-2 # 1/s

# Initial concentrations
E0 0.01*NA*V # uM -> molec / cell
S0 1.0*NA*V # uM -> molec / cell

end parameters
```

seed species

observables

functions

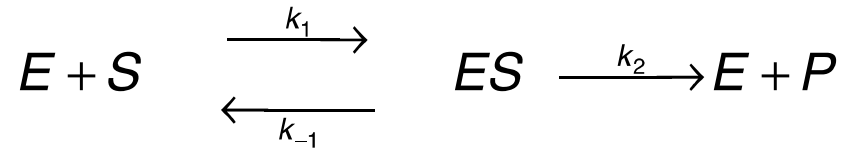
reaction rules

actions



# Example 1: MM Mechanism

parameters



molecule types

molecule types— molecules, their components, and their allowed component states are declared here.

seed species

observables

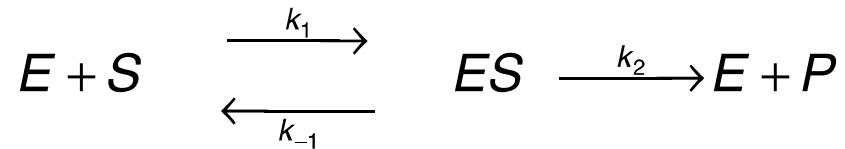
functions

reaction rules

actions

# Example 1: MM Mechanism

parameters



molecule types

```
begin molecule types
```

```
E(s)
```

seed species

```
S(Y~0~P)
```

```
end molecule types
```

observables

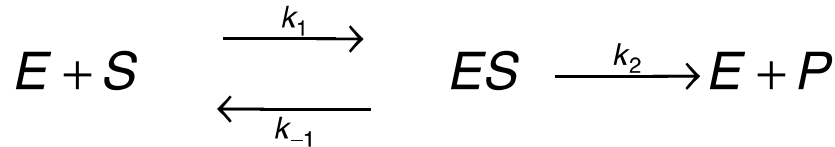
functions

reaction rules

actions

# Example 1: MM Mechanism

parameters



molecule types

**seed species**

seed species– species initially present in the system at time  $t=0$  followed by their initial concentration. Standard is all molecule types in their “ground state” with basal expression levels. May include complexes. All components of molecules that have states must be in a specified state. All complexes must be connected.

observables

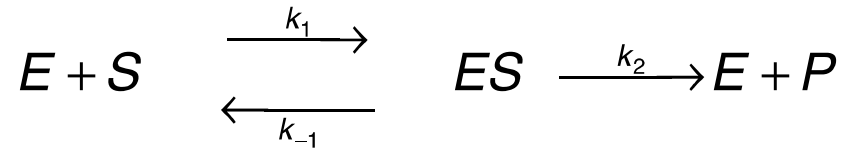
functions

reaction rules

actions

# Example 1: MM Mechanism

parameters



molecule types

seed species

observables

functions

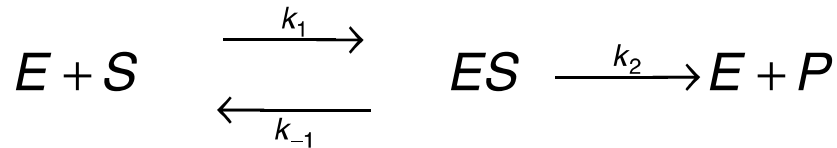
```
begin seed species
  E(s)      E0
  S(Y~0)    S0
end seed species
```

reaction rules

actions

# Example 1: MM Mechanism

parameters



molecule types

seed species

observables– Defined sums of concentrations of species with specified properties. Syntax is <type> <name> <pattern>. Types considered here are Molecules and Species, which indicate weighted and unweighted sums respectively. These are used to define model outputs and are used as to make the default plot in RuleBender.

observables

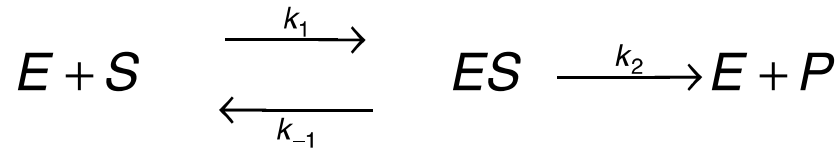
functions

reaction rules

actions

# Example 1: MM Mechanism

parameters



molecule types

begin observables

seed species

Molecules SU S(Y~0)

Molecules SP S(Y~P)

Molecules ES E(s!1).S(Y!1)

observables

end observables

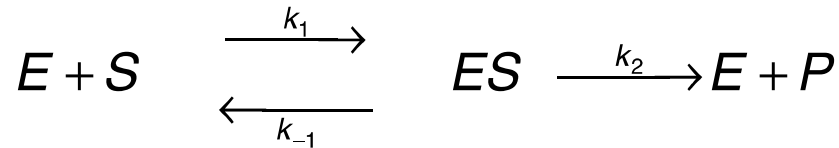
functions

reaction rules

actions

# Example 1: MM Mechanism

parameters



molecule types

begin observables

seed species

Molecules SU S(Y~0)

Molecules SP S(Y~P)

Molecules ES E(s!1).S(Y!1)

end observables

observables

observable SU S(Y~0)

S(Y~0)

functions

matches

not

species

S(Y~0)

E(s!1).S(Y~0!1)

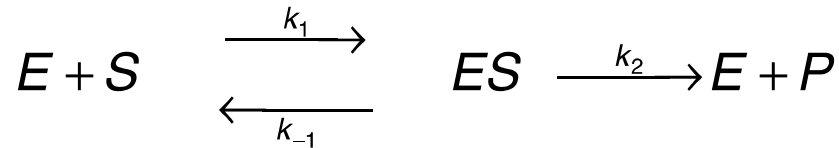
reaction rules

SU = sum of concentration of matches = [S(Y~0)]

actions

# Example 1: MM Mechanism

parameters



molecule types

begin observables

seed species

Molecules SU S(Y~0)

Molecules SP S(Y~P)

Molecules ES E(s!1).S(Y!1)

end observables

observables


observable ES E(s!1).S(Y!1)

functions

matches

species

E(s!1).S(Y~0!1)



reaction rules

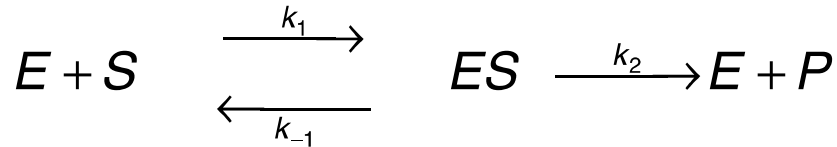
ES = sum of concentration of matches = [E(s!1).S(Y~0!1)]

actions



# Example 1: MM Mechanism

parameters



molecule types

seed species

reaction rules– Rules that generate reactions based on selecting reactants with specified properties and transforming them in a specified way with the specified rate law. Syntax is <name>: <reactants> <arrow> <products> <rate law>. Name is optional but useful.

observables

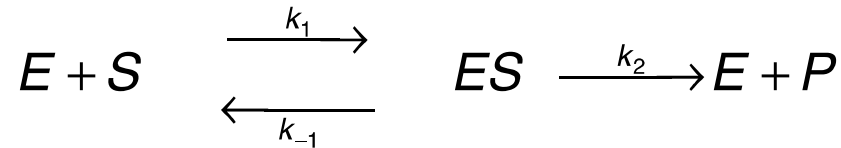
functions

reaction rules

actions

# Example 1: MM Mechanism

parameters



molecule types

seed species

```
begin reaction rules
```

observables

```
ESbind: \  
  E(s) + S(Y~0) <-> E(s!1).S(Y~0!1) kp1, km1
```

functions

```
ESconvert: \  
  E(s!1).S(Y~0!1) -> E(s) + S(Y~P) k2
```

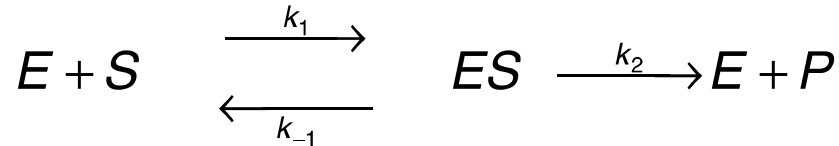
reaction rules

```
end reaction rules
```

actions

# Example 1: MM Mechanism

parameters



molecule types

actions– Need not be enclosed in block. Come after model definition and specify simulation protocol for a model.

seed species

```
generate_network( {} );  
simulate_ode( {t_end=>1000, n_steps=>100} );
```

observables

functions

reaction rules

actions