Pathway Logic and Symbolic Systems Biology

Carolyn Talcott
SRI International
June 2008

PLAN

- Symbolic systems biology
- Executable Specification in RWL
- Biological Processes (What to model)
- Building a PL knowledge base
- Computing with PL models
  - Small KB
  - Egf Stimulation

SYMBOLIC SYSTEMS BIOLOGY

Biological Systems

- Biological processes are complex
  - genes, proteins, metabolites
  - cells, organs, organisms
- Dynamics that range over huge timescales
  - microseconds to years
- Spatial scales over 12 orders of magnitude
  - single protein to cell, cell to whole organism
- Oceans of experimental biological data generated
- Important intuitions captured in mental models that biologists build of biological processes

Symbolic Systems Biology

- Symbolic -- represented in a logical framework
- Systems -- how things interact and work together, integration of multiple viewpoints and levels of abstraction
- Which biology? Causal networks of biomolecular interactions and reactions
- Goals:
  - Develop formal models that are as close as possible to domain expert's mental models
  - Compute with, analyze and reason about these complex networks
  - New insights into / understanding of biological mechanisms

Logical Framework

- Making description and reasoning precise
- Language
  - for describing things and/or properties
  - given by a signature and rules for generating expressions (terms, formulas)
- <u>Semantic model</u> -- mathematical structure (meaning)
  - interpretation of terms
  - satisfaction of formulas: M |= wff
- Reasoning -- rules for inferring valid formulae
- <u>Symbolic model</u> -- theory (axioms) used to answer questions

Executable Symbolic Models

- Describe system states and rules for change
- Transition graph
  - nodes -- reachable states
  - edges -- rules connecting states
- Path -- sequence of nodes and edges in transition graph (computation / derivation)
- Execution strategy -- picks a path

Symbolic Analysis I

- Static Analysis
  - how are elements organized -- sort hierarchy
  - control flow / dependencies
  - detection of incompleteness
- Forward simulation from a given state
  - run model using a specific strategy
  - fast, first exploration of a model

Symbolic Analysis II

- Forward search from a given state
  - breadth first search of transition graph
  - find ALL possible outcomes
  - find only outcomes satisfying a given property
- Backward search from a given state S
  - run a model backwards from S
  - find initial states leading to S
  - find transitions that contribute to reaching S

Symbolic Analysis III

- Model checking
  - determines if all pathways from a given state satisfy a given property, if not a counter example is returned
  - example property:
    - molecule X is never produced before Y
  - counter example:
    - pathway in which Y is produced after X

Symbolic Analysis IV

- Constraint solving
  - Find values for a set of variables satisfying given constraints.
  - MaxSat deals with conflicts
    - weight constraints
    - find solutions that maximize the weight of satisfied constraints
  - Finding possible steady state flows of information or chemicals through a system can be formulated as a constraint problem.

Symbolic Analysis V

- Meta analysis -- reasoning about the model itself
  - find transitions producing / consuming X
  - find all phosphorylation reactions
  - check that transitions satisfy some property such as stoichiometry
  - transform a model and property to another logic (for access to tools)

A Sampling of Formalisms

- Rule-based + Temporal logics
- Petri nets + Temporal logics
- Membrane calculi -- spatial process calculi / logics
- Statecharts + Live sequence charts
- Stochastic transitions systems and logics
- Hybrid Automata + Abstraction

Executable Specification in Rewriting Logic (Maude)

About Rewriting Logic

- Rewriting Logic is a logical formalism that is based on two simple ideas
  - states of a system are represented as elements of an algebraic data type
  - the behavior of a system is given by local transitions between states described by rewrite rules
- It is a logic for executable specification and analysis of software systems, that may be concurrent, distributed, or even mobile.
- It is also a (meta) logic for specifying and reasoning about formal systems, including itself (reflection!)

Rewriting

- Rewrite theory: (Signature, Labels, Rules)
- Signature: (Sorts, Ops, Eqns) -- an equational theory
  - Describe data types, structure of system state
- Rules have the form label : t => t' if cond
- Rewriting operates modulo equations
  - rules apply locally
  - generates computations (pathways)

MAUDE

- Maude is a language and tool based on rewriting logic
- See: http://maude.cs.uiuc.edu
- Features:

  Executability -- position /rule/object fair rewriting

  High performance engine --- {ACI} matching

  Modularity and parameterization

  Builtins -- booleans, number hierarchy, strings

  Reflection -- using descent and ascent functions

  Search and model-checking

Maude Functional Modules
the equational part

A Maude functional module describes an equational theory specifying data types, data constructors, and functions

```
fmod <modname> is
  <imports>       *** reuse, modularity
  <sorts>         *** data types and subtyping
  <opdecls>       *** names/and arities of operations
  <eqns>          *** how to compute functions
endfm
```

- The semantics of an fmod is an initial model
  - no junk---every data value is constructable
  - no confusion---two terms are equal only if forced by the equations

An Equational Specification

Natural numbers are specified by

```
fmod NAT is
  sorts Zero NzNat Nat .
  subsort Zero NzNat < Nat .
  op 0 : -> Zero [ctor] .
  op s_ : Nat -> NzNat [ctor] .
      ....
endfm
```

Examples: `0, s s s 0 (also prints as 3)`

An Equational Specification III

Extend NATLIST with a function to sum the elements of a list

```
var n : Nat. var nl : NatList .

op sum : NatList -> Nat .
eq sum(nil) = 0 .
eq sum(n nl) = n + sum(nl) .
```

Equations are used to reduce terms to canonical form

```
sum(1 2 3) = 1 + sum(2 3)
           = 1 + 2 + sum(3)
           = 1 + 2 + 3
           = 6
```

Maude System Modules
the Rules Part

System dynamics are specified using rewrite rules

```
mod <modname> is
*** functional part
  <imports>        *** reuse, modularity
  <sorts>          *** data types and subtyping
  <opdecls>        *** names/and arities of operations
  <eqns>           *** how to compute functions
***
  <rules>
endm
```

A system module defines a set of computations (derivations) over the data types specified by the functional part.

Rule Delcarations

The <rules> part of a system module consists of rule declarations having one of the forms

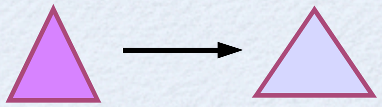
```
     rl[<id>]: <lhs> => <rhs>   .

    crl[<id>]: <lhs> => <rhs> if <cond> .
```

`<lhs>`, `<rhs>`, `<cond>` are terms possibly containing variables. A rule applies to a term T if there is a substitution S (mapping variables to terms) such that S(`<lhs>`) is a subterm of T (`<lhs>` matches a subterm of T) and S(`<cond>`) rewrites to true.

In this case T can be rewritten by replacing the matched subterm by the matching instance of `<rhs>`, S(`<rhs>`).

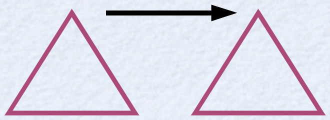
Rewriting Rules

one step rewrite:

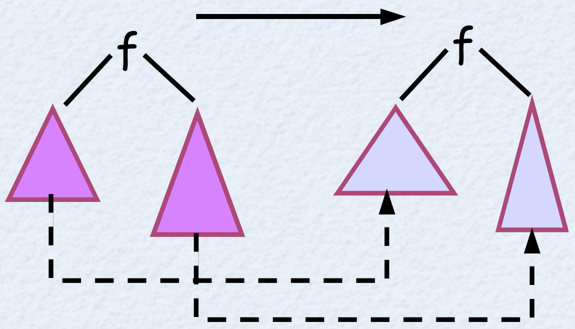


closed under

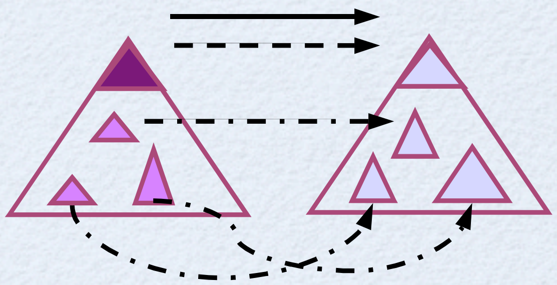
reflexivity:



congruence:



replacement:



Rule Application

Suppose we have the following rules (in a suitable module).

```
    rl[fa2b]: f(a,x) => f(b,x) .
    rl[hh2h]: h h(y) => h(y) .
```

then

```
    g(c,f(a,d)) => g(c,f(b,d))
```

and

```
    h h(g(c,f(a,d))) => h(g(c,f(b,d)))
```

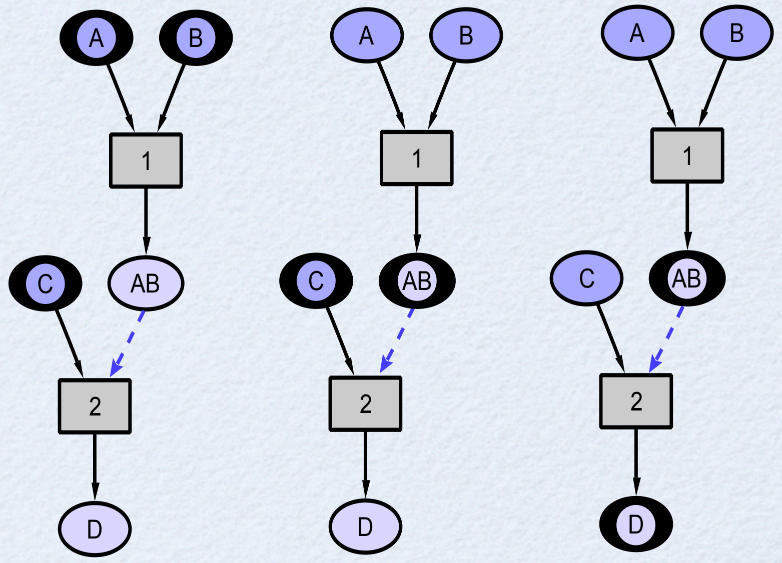
also using replacement.

Before applying a rewrite rule, Maude reduces a term to canonical form using equations. The rule lhs is not reduced.

About Petri Nets

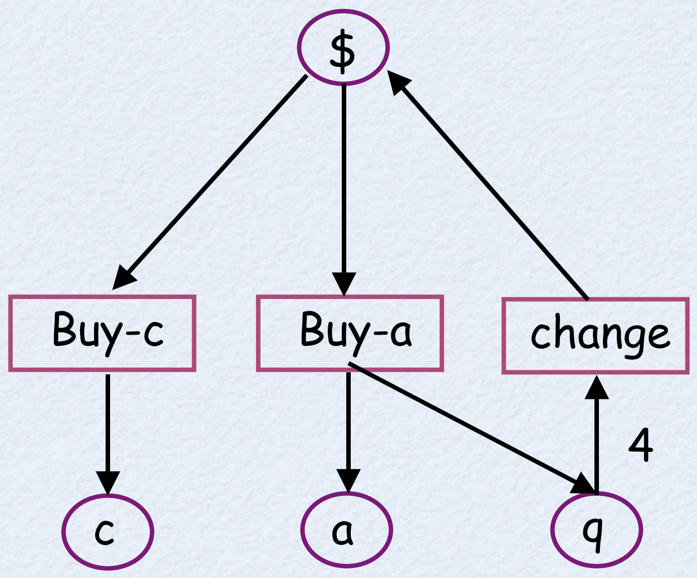
A Petri net is represented as a graph with two kinds of nodes:
  * transitions/rules (reactions--squares)
  * places/occurrences (reactants, products, modifiers--ovals)

A Petri net process has <u>tokens</u> on some of its places. A rule can fire if all of its inputs have tokens. Firing a rule moves tokens from input to output.



An execution is a sequence of rule firings.
A pathway is represented as an execution subgraph.

Petri Net Model of a VendingMachine



```
mod VENDING-MACHINE is
  sorts Coin Item Place Marking .
  subsorts Coin Item < Place < Marking .
  op null : -> Marking .
             *** empty marking
  ops $ q : -> Coin .
  ops a c : -> Item .
  op _ _ : Marking Marking -> Marking
           [assoc comm id: null] .
           *** multiset
  rl[buy-c]: $ => c .
  rl[buy-a]: $ => a q .
  rl[change]: q q q q => $ .
endm
```

Using the Vending Machine

What is one way to use 3 $s?

```
Maude> rew $ $ $ .
result Marking: q a c c
```

How can I get 2 apples with 3 $s?

```
Maude> search $ $ $ =>! a a M:Marking .

Solution 1 (state 8)
M:Marking --> q q c

Solution 2 (state 9)
M:Marking --> q q q a

No more solutions.
states: 10  rewrites: 12)
```

MODEL CHECKING I

- Algorithm for determining if M |= P (M satisfies P) where M is a `model' and `P' is a property.
- In our case a model is a Maude specification of a system together with a staAlgorithm for determining if M |= P (M satisfies P) where M is a `model' and `P' is a property.
- In our case a model is a Maude specification of a system together with a state of interest.

Model Checking II

- Temporal formulas are built from propositions about states, using boolean connectives, and temporal operators [] (always) and <> (eventually).
- Satisfaction for M |= A is axiomatized by equations
  - M |= P if C |= P for every computation C of M
  - C |= A if the first state of C satisfies A
  - C |= [] P if every suffix of C satisfies P
  - C |= <>P if some suffix of C satisfies P

Model Checking
Defining Properties

```
mod MC-VENDING-MACHINE is
  inc VENDING-MACHINE .  inc MODEL-CHECKER .  inc NAT .

  op vm : Marking -> State [ctor] .
  op countPlace : Marking Place -> Nat .
  op value : Marking -> Nat .   *** in units of quarters
  ....

  ops fiveQ lte4Q : -> Prop .
  ops nCakes nApples val : Nat -> Prop .

  eq vm(M) |= fiveQ = countPlace(M,q) == 5 .
  eq vm(M) |= lte4Q = countPlace(M,q) <= 4 .
  eq vm(M) |= nApples(n) = countPlace(M,a) == n .
  eq vm(M) |= val(n) = value(M) == n .
endm
```

Model Checking the Vending Machine 1

Starting with 5 $s, can we get 6 apples without accumulating more than 4 quarters? Model check the claim that we can't.

```
Maude>
   red modelCheck(vm($ $ $ $ $),[]~(lte4Q U nApples(6))) .
result ModelCheckResult: counterexample(
  {vm($ $ $ $ $),'buy-a}
  {vm($ $ $ $ q a),'buy-a}
  {vm($ $ $ q q a a),'buy-a}
  {vm($ $ q q q a a a),'buy-a}
  {vm($ q q q q a a a a),'change}
  {vm($ $ a a a a),'buy-a}
  {vm($ q a a a a a), 'buy-a},
  {vm(q q a a a a a a),deadlock})
```

A counterexample to a formula is a pair of transition lists representing an infinite compuation which fails to satisfy the formula. A transition is a state and a rule identifier. The second list (deadlock) represents a loop.

Model checking the vending machine II

Starting with 5 $s, can we get 5 quarters then 6 apples?

```
Maude>
red modelCheck(vm($ $ $ $ $),
               []~(<>fiveQ /\ (fiveQ |-> nApples(6))) .
result ModelCheckResult: counterexample(...)
```

Is value conserved?

```
Maude> red modelCheck(vm($ $ $ $ $),[]val(20) .
result Bool: true
```

Biological Processes

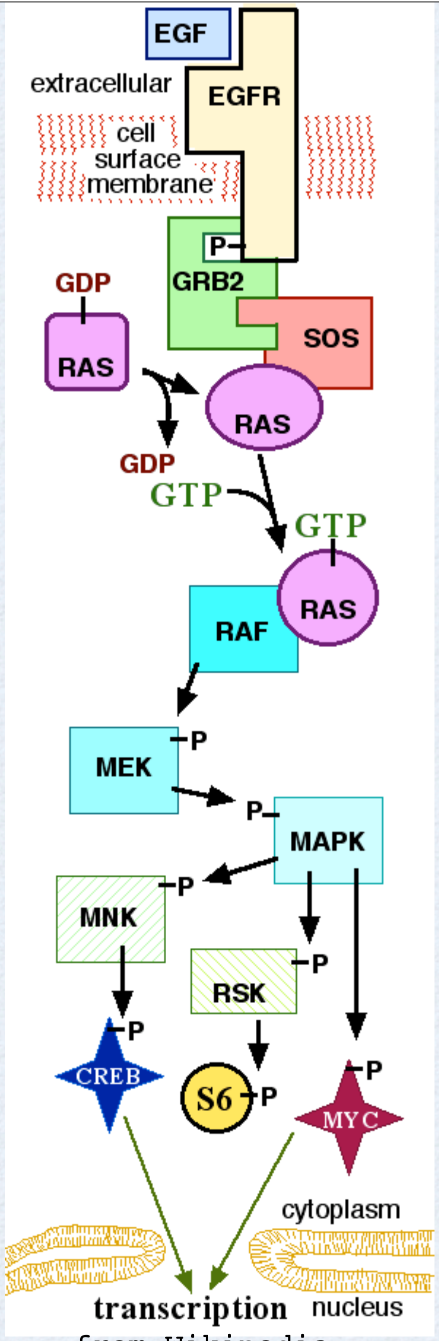
What to Model

Cellular Signaling

- Cells respond to changes in their environment through biochemical pathways that detect, transduce, and transmit information to effector molecules within different cellular compartments.
- Most signaling pathways involve hierarchical assembly in space and time of multi-protein complexes that regulate the flow of information according to logical rules.
- Biological subnetworks interact to produce high levels of physiological organization (e.g., circadian clock subnetworks are integrated with metabolic, survival, and growth subnetworks).

Signaling Pathways

- Signaling pathways involve the modification and/or assembly of proteins and other molecules within cellular compartments into complexes that coordinate and regulate the flow of information.
- Signaling pathways are distributed in networks having stimulatory (positive) and inhibitory (negative) feedback loops, and other concurrent interactions to ensure that signals are propagated and interpreted appropriately in a particular cell or tissue.
- Signaling networks are robust and adaptive, in part because of combinatorial complex formation (several building blocks for forming the same type of complex), redundant pathways, and feedback loops.



from Wikipedia

Egf stimulation of the Mitogen Activated Protein Kinase (MAPK) pathway.

Egf → EgfR → Grb2 → Sos1 → Ras → Raf1 → Mek → Erk

- Egf (EGF) binds to the Egf receptor (EgfR) and stimulates its protein tyrosine kinase activity to cause autophosphorylation, thus activating EgfR.
- The adaptor protein Grb2 (GRB2) and the guanine nucleotide exchange factor Sos1 (SOS) are recruited to the membrane, binding to EgfR.
- The EgfR complex activates a Ras family GTPase
- Activated Ras activates Raf1, a member of the RAF serine/threonine protein kinase family.
- Raf1 activates the protein kinase Mek (MEK), which then activates Erk (MAPK)
- ...

Features 1

- Naming
  - different biologists use different names for the same protein Egf vs EGF, Erk vs MAPK, EgfR vs ErbB1 vs HerbB1
  - link name to `standard' source: SwissProt, KEGG, HUGO ...
- Activity / state -- a protein may need to be is a specific state (active) to carry out its function
- Location -- what compartment, where in the compartment
  - media -- outside a cell
  - Cell compartment
    - Membrane -- integral, surface, interior
    - Cytoplasm

Features II

- Roles / functions
  - Ligand -- Egf
  - Receptor -- EgfR -- binds ligand
  - Scaffold / adaptor -- complex formation
  - Kinase -- Raf1, Mek, Erk
- Processes
  - recruiting
  - postranslational modification
    - phosphorylation (by kinase)
    - ubiquitination

Pathway Logic (PL) Representation of Signaling

http://pl.csl.sri.com/

About Pathway Logic

Pathway Logic (PL) is an approach to modeling biological processes as executable formal specifications (in Maude)

The resulting models can be queried

- using formal methods tools: given an initial state
  - execute --- find some pathway
  - search --- find all reachable states satisfying a given property
  - model-check --- find a pathway satisfying a temporal formula
- using reflection
  - find all rules that use / produce X (for example, activated Rac)
  - find rules down stream of a given rule or component

PATHWAY LOGIC ORGANIZATION

A Pathway Logic (PL) system has four parts

- Theops --- sorts and operations
- Components --- specific proteins, chemicals ...
- Rules --- signal transduction reactions
- Dishes --- candidate initial states

Knowledge base: Theops + Components + Rules

Equational part: Theops + Components

A PL cell signaling model is generated from

- a knowledge base
- a dish

THEOPS

Specifies sorts and operations (data types) used to represent cells:

- Proteins and other compounds
- Complexes
- Soup --- mixtures / solutions / supernatant ...
- Post-translational modifications
- Locations --- cellular compartments refined
- Cells --- collection of locations
- Dishes --- for experiments, think Petri dish

Sample from Components

```
sort ErbB1L . subsort ErbB1L < Protein . *** ErbB1 Ligand

op Egf : -> ErbB1L [metadata "(\
  (spname EGF_HUMAN)\
  (spnumber P01133)\
  (hugosym EGF)\
  (category Ligand)\
  (synonyms \"Pro-epidermal growth factor precursor, EGF\" \
            \"Contains: Epidermal growth factor, Urogastrone \"))"] .


op EgfR : -> Protein [metadata "(\
  (spname EGFR_HUMAN)\
  (spnumber P00533)\
  (hugosym EGFR)\
  (category Receptor)\
  (synonyms \"Epidermal growth factor receptor precursor\" \
            \"Receptor tyrosine-protein kinase ErbB-1, ERBB1 \"))"] .


op PIP2 : -> Chemical [metadata "(\
   (category Chemical)\
   (keggcpd C04569)\
   (synonyms \"Phosphatidylinositol-4,5P \" ))"] .
```

THEOPS: CELLS & DISHES

```
mod CELL is inc LOCATION .
  sorts Cell CellType .
  subsort Cell < Soup .
  op [_|_] : CellType Soup -> Cell .
  op Cell : -> CellType .
  ...
endm
```

Example cell:

```
    [Cell | {CLm | EgfR PIP2}{CLi | [Hras - GDP] Src}
            {CLc | Gab1 Grb2 Pi3k Plcg Sos1}]
```

```
mod DISH is inc CELL .
  sort Dish .
  op PD : Soup -> Dish .
endm
```

Example rasDish:

```
 PD(Egf [Cell | {CLm | EgfR PIP2}{CLi | [Hras - GDP] Src}
                {CLc | Gab1 Grb2 Pi3k Plcg Sos1}])
```

THEOPS: SOUP

```
fmod THING is inc PROTEIN .     *** Basic building blocks
  sort Thing Family Composite Complex Chemical Signature .
  subsorts Protein Family Composite Complex Chemical Signature
          < Thing .
  op (_:_) : Thing Thing -> Complex [assoc comm] .
  ... endfm
```

Example complex: Egf : [EgfR - act]

```
fmod SOUP is inc THING .         *** mixtures
  sort Soup .                    *** a multiset of Thing
  subsort Thing < Soup .
  op empty : -> Soup .
  op __ : Soup Soup -> Soup [assoc comm id: empty ] .

  op _has_ : Soup Thing -> Bool .
  eq (T1:Thing S:Soup ) has T2:Thing =
     if T1:Thing == T2:Thing
     then true else S:Soup has T2:Thing fi .
  eq (S:Soup has T:Thing) =  false [owise] .
  ... endfm
```

Example soups:

```
       Gab1 Grb2 Pi3k Plcg Sos1
       [Hras - GDP] Src
```

TheOps: Modifications & Locations

```
fmod MODIFICATION is pr SOUP .
  sorts  Modification ModSet .    *** multisets of modifications
  subsort Modification < ModSet .
  op none : -> ModSet .
  op __  : ModSet ModSet -> ModSet [assoc comm id: none] .
  op [_-_] : Protein   ModSet -> Protein [right id: none ] .
  op act : -> Modification .
  op phos :       -> Modification .
  op Yphos : -> Modification .
  ops GTP GDP : -> Modification .    *** used for small GTPases
endfm
```

Examples: [EgfR - act] [Hras - GTP]

```
fmod LOCATION is inc MODIFICATION .
sort Location LocName .
subsort Location < Soup .
op {_|_} : LocName Soup -> Location [format (n d d t d d)] .

ops CLo CLm CLi CLc : -> LocName .   *** Cell - out,mem,in,cytosol
ops NUo NUm NUi NUc : -> LocName .   *** Nucleus - out,mem,in,cytosol
...
endfm
```

PL Rules

- A PL rule specifies the change in a cell due to an enabled reaction. The rule label gives a hint as to what happens.
- In addition rules must be annotated with evidence
  - literature citations

    pubmed id (type: review, data) brief description
  - curator notes

RULE 1: RECEPTOR BINDING

If a dish contains an EgfR ligand (?ErbB1L:ErbB1L) outside a cell with EgfR in the cell membrane then the ligand binds to exterior part of the receptor and the receptor is activated.

```
rl[1.EgfR.act]:
 ?ErbB1L:ErbB1L  [CellType:CellType | ct {CLm | clm EgfR}]
  =>
 [CellType:CellType | ct {CLm | clm ([EgfR - act] : ?ErbB1L:ErbB1L)} ] .
```

Rule 1 applies to rasDish with the match

```
  ?ErbB1L:ErbB1L := Egf
  clm := PIP2
  ct := {CLi | [Hras - GDP]  Src} {CLc | Gab1 Grb2 Pi3k Plcg Sos1}
```

and the resulting dish (rasDish1) is

```
  PD([Cell |
     {CLm | ([EgfR - act] : Egf)  PIP2}
     {CLi | [Hras - GDP]  Src}
     {CLc | Gab1 Grb2 Pi3k Plcg Sos1}]) .
```

Rule 5: Recruitment

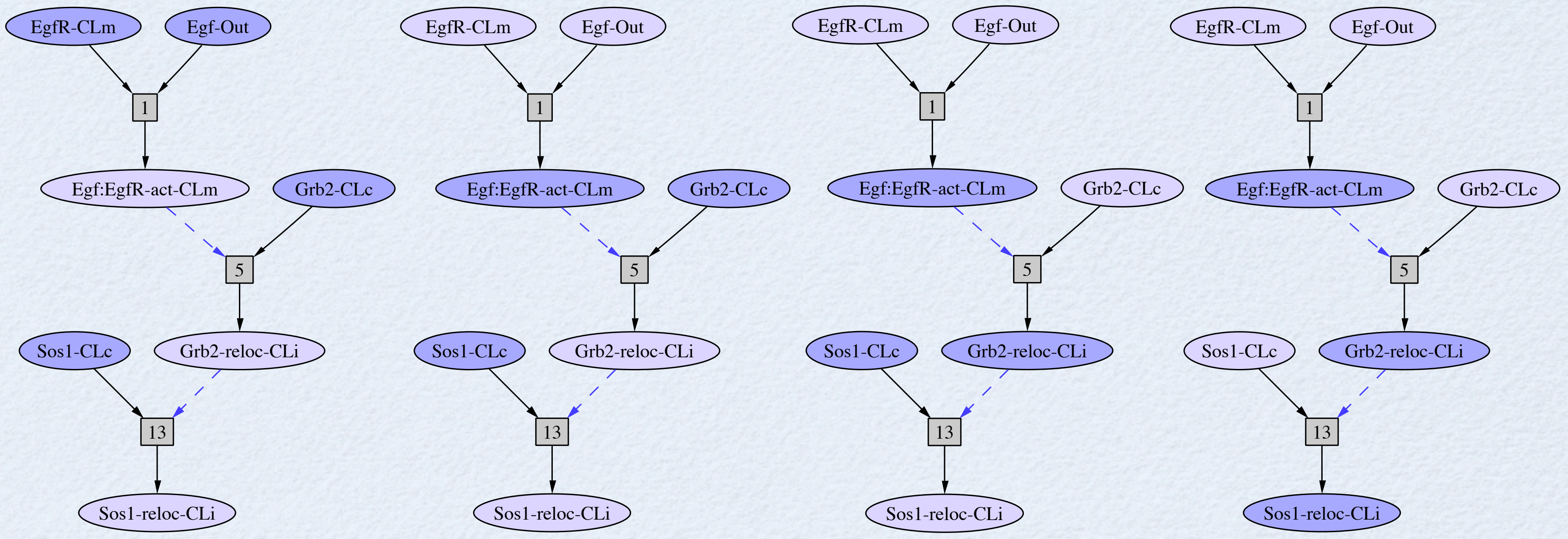
Activated EgfR recruits Grb2 to the inside of the cell membrane

```
rl[5.Grb2.reloc]:
   {CLm | clm [EgfR - act]        }
   {CLi | cli                     }
   {CLc | clc Grb2                }
   =>
   {CLm | clm [EgfR - act]        }
   {CLi | cli [Grb2 - reloc]      }
   {CLc | clc                     } .
```

Rewriting rasDish1 with rule 5 results in

```
PD([Cell |
      {CLm | ([EgfR - act] : Egf)  PIP2}
      {CLi | [Hras - GDP]  Src [Grb2 - reloc]}
      {CLc | Gab1 Pi3k Plcg Sos1}]) .
```

Rule Execution as Petri Nets



rasDish =rule1=> rasDish1 =rule5=> rasDish2 =rule13=> rasDish3

Ovals are occurrences -- components in locations.
Dark ovals are present in the current state (marked).
Squares are rules.
Dashed edges connect components that are not changed.

A Small KB
Very early EGF stimulation

rasNet

Rule instances relevant to Hras activation

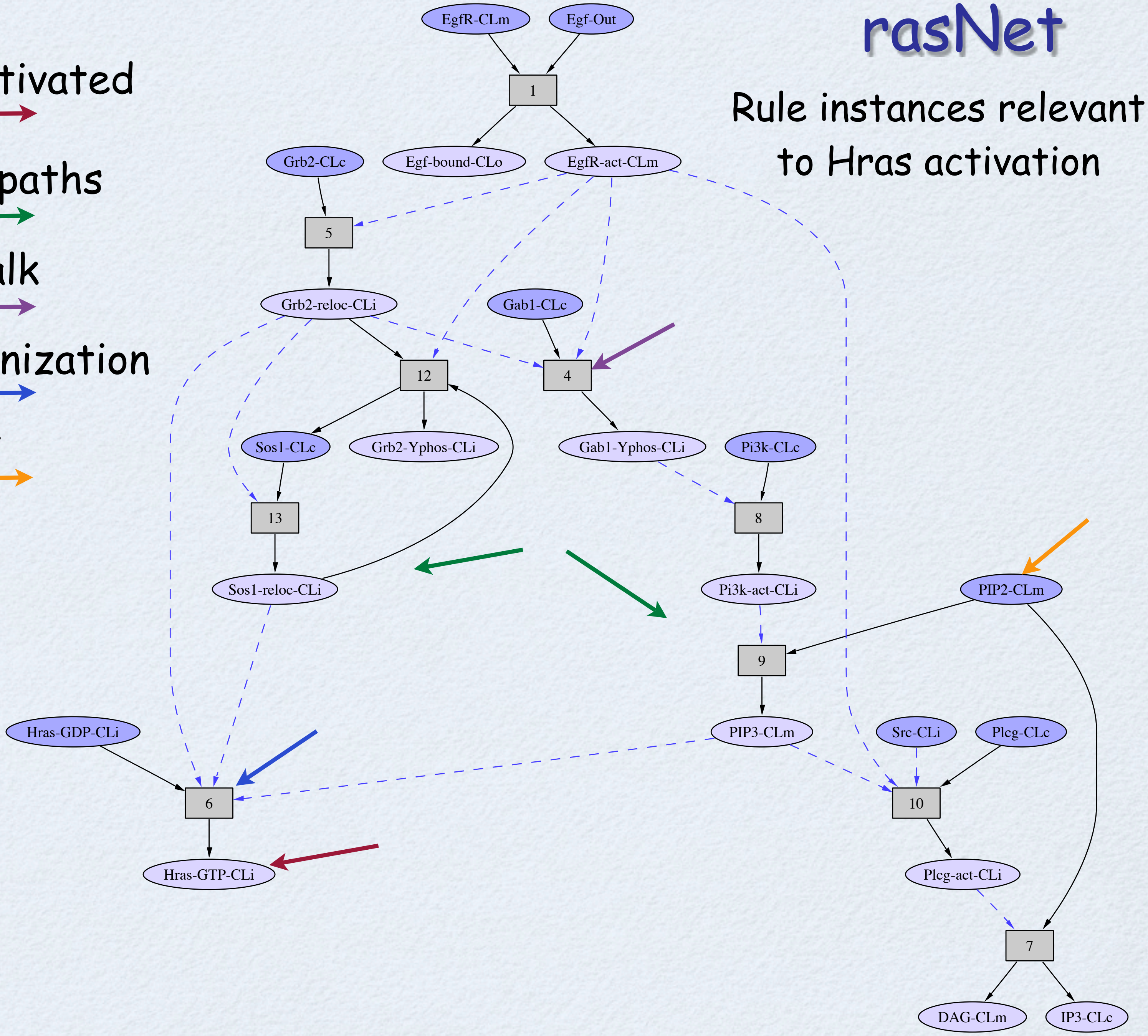
Hras activated

Parallel paths

Cross talk

Synchronization

Conflict



Where did the graph come from?

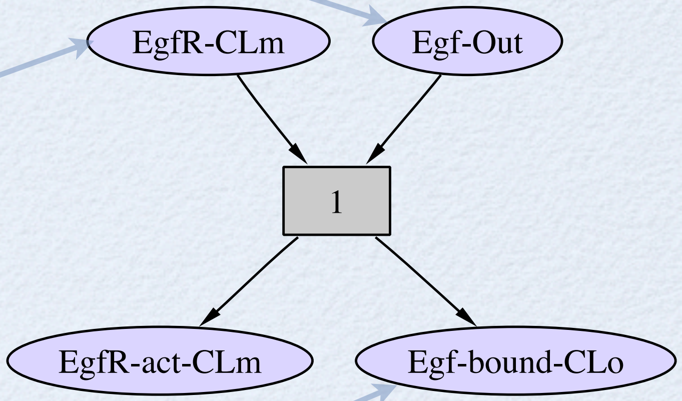
A rule can be represented as a rule node with lhs/rhs represented as incoming/ outgoing `occurrence' nodes.

A occurrence label represents a components modifications and location.

EgfR-CLm represents EgfR in the CLm location.

EgfR-act-CLm represents activated EgfR ([EgfR - act]) in the CLm location.

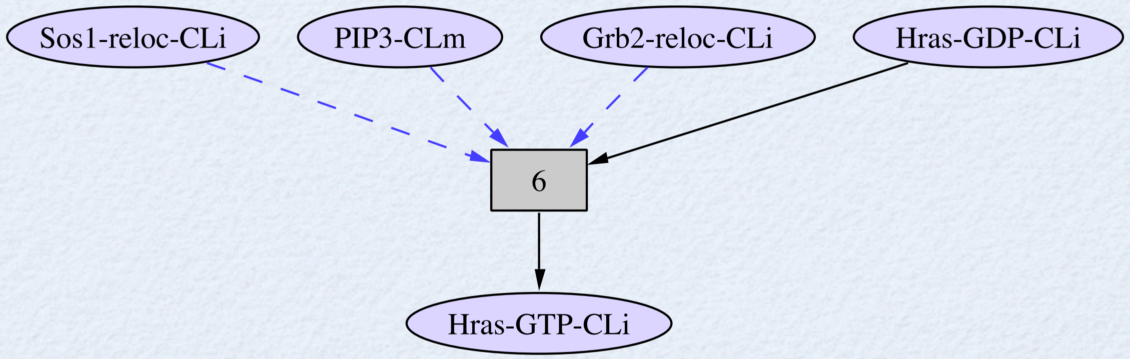
```
rl[1.EgfR.act]:
  Egf
  [CellType:CellType | ct
  {CLo | clo                  }
  {CLm | clm EgfR             } ]
  =>
  [CellType:CellType | ct
  {CLo | clo [Egf - bound] }
  {CLm | clm [EgfR - act]  } ] .
```



Rules with Modifiers

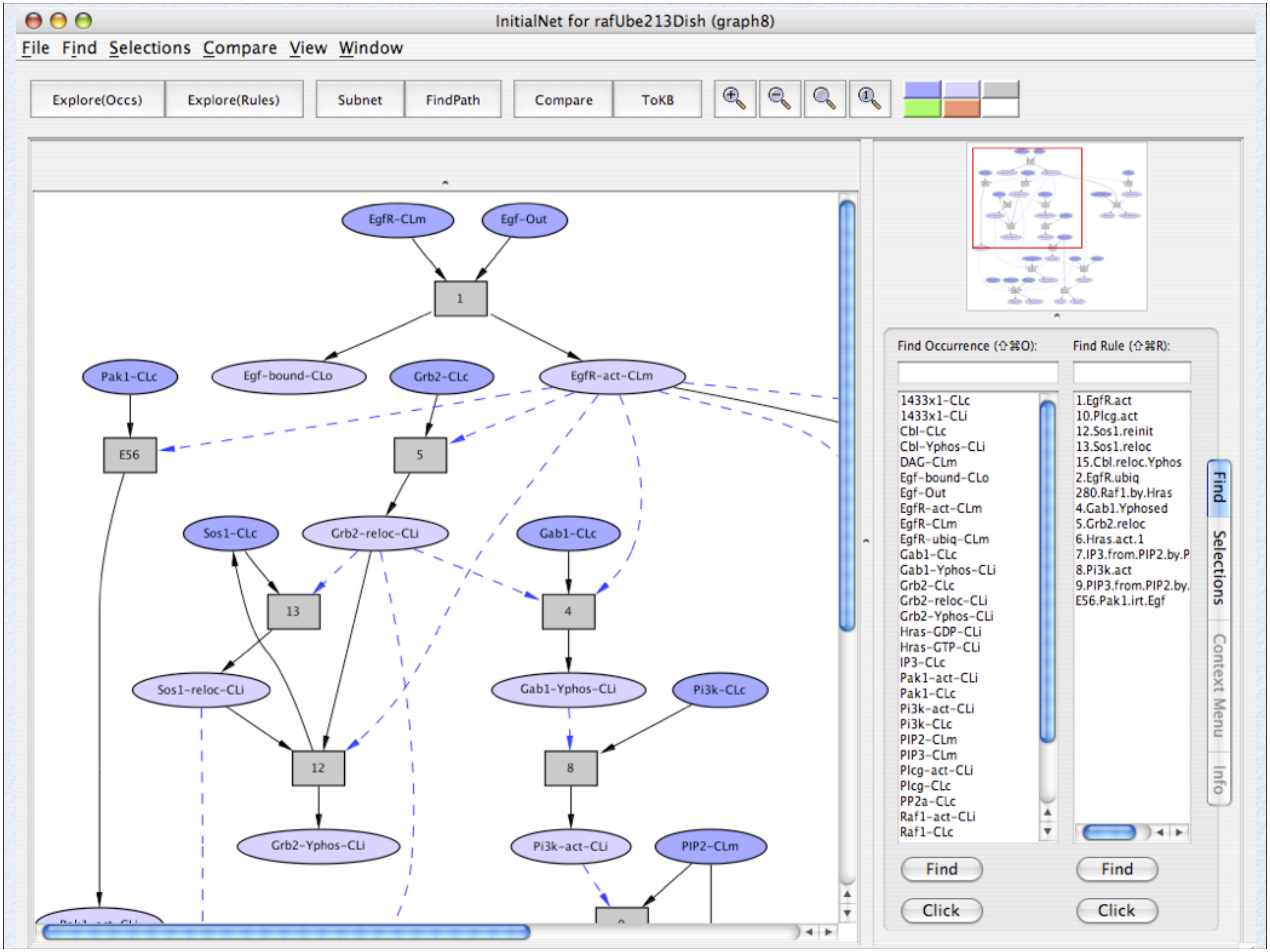
A rule element that is the same on the left and right hand sides is represented using one occurrence node connected to the rule with a dashed arrow.

```
rl[6.Hras.act.1]:
  {CLm | clm PIP3                                     }
  {CLi | cli [Grb2 - reloc] [Sos1 - reloc] [Hras - GDP] }
  =>
  {CLm | clm PIP3                                     }
  {CLi | cli [Grb2 - reloc] [Sos1 - reloc] [Hras - GTP] } .
```



The Pathway Logic Assistant (PLA)

- Provides a means to interact with a PL model
- Manages multiple representations
  - Maude module (logical representation)
  - PetriNet (process representation for efficient query)
  - Graph (for interactive visualization)
- Exports Representations to other tools
  - Lola (and SAL model checkers)
  - Dot -- graph layout
  - JLambda (interactive visualization, Java side)
  - SBML (xml based standard for model exchange)



Find Occurrence (⇧⌘O):

1433x1-CLc
1433x1-CLi
Cbl-CLc
Cbl-Yphos-CLi
DAG-CLm
Egf-bound-CLo
Egf-Out
EgfR-act-CLm
EgfR-CLm
EgfR-ubiq-CLm
Gab1-CLc
Gab1-Yphos-CLi
Grb2-CLc
Grb2-reloc-CLi
Grb2-Yphos-CLi
Hras-GDP-CLi
Hras-GTP-CLi
IP3-CLc
Pak1-act-CLi
Pak1-CLc
Pi3k-act-CLi
Pi3k-CLc
PIP2-CLm
PIP3-CLm
Plcg-act-CLi
Plcg-CLc
PP2a-CLc
Raf1-act-CLi
Raf1-CLc

Find Click

Find Rule (⇧⌘R):

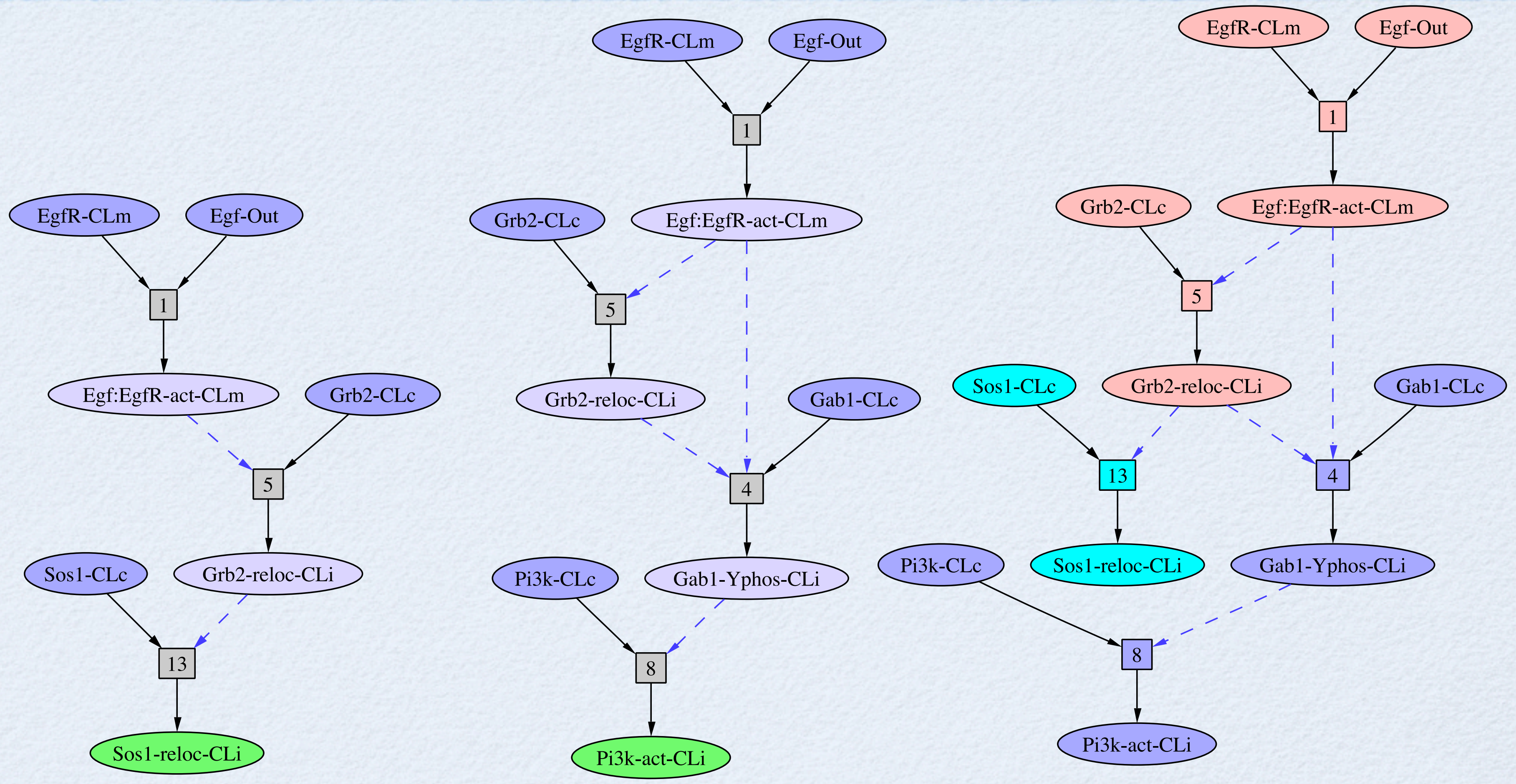
1.EgfR.act
10.Plcg.act
12.Sos1.reinit
13.Sos1.reloc
15.Cbl.reloc.Yphos
2.EgfR.ubiq
280.Raf1.by.Hras
4.Gab1.Yphosed
5.Grb2.reloc
6.Hras.act.1
7.IP3.from.PIP2.by.P
8.Pi3k.act
9.PIP3.from.PIP2.by.
E56.Pak1.irt.Egf

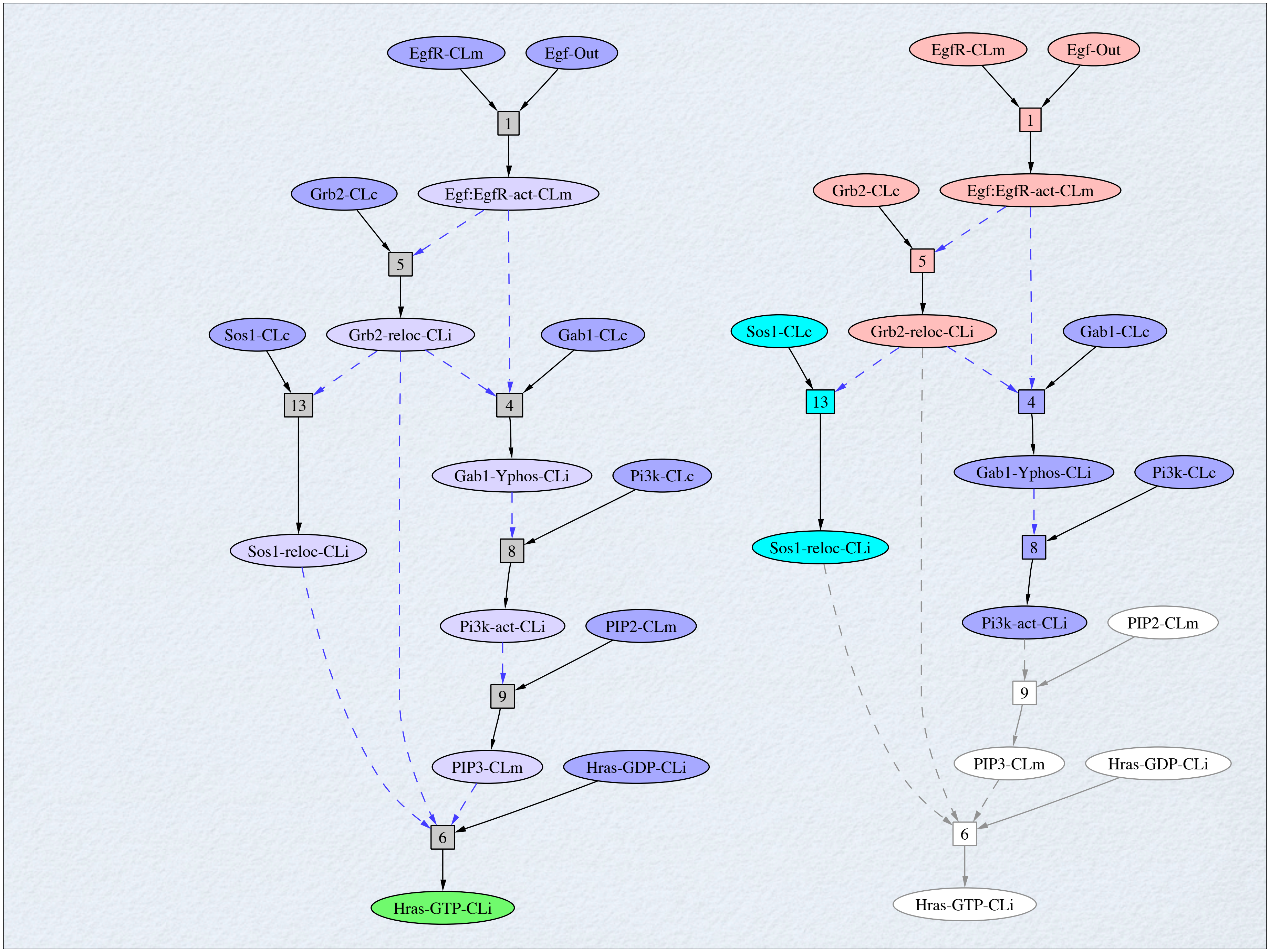
Find Click

A Simple Query Language

- Given a Petri net with transitions P and initial marking O (for occurrences) there are two types of query
  - subnet
  - findPath - a computation / unfolding
- For each type there are three parameters
  - G: a goal set---occurrences required to be present at the end of a path
  - A: an avoid set---occurrences that must not appear in any transition fired
  - H: as list of identifiers of transitions that must not be fired
- findPath returns a pathway (transition list) generating a computation satisfying the requiremments.
- subnet returns a subnet containing all (minimal) such pathways.

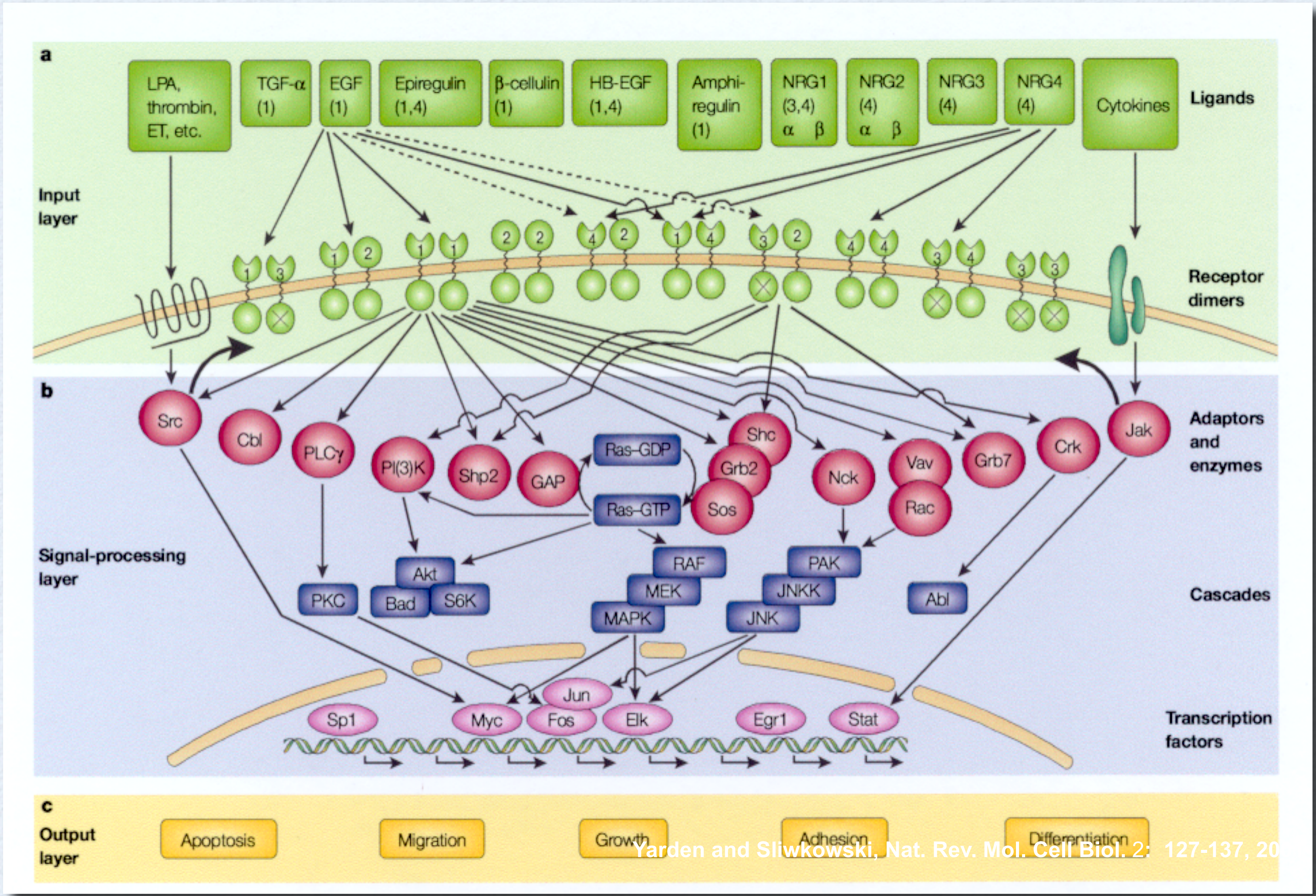
Pathway Examples




EgfR-CLm
Egf-Out
1
Egf:EgfR-act-CLm
Grb2-CLc
5
Grb2-reloc-CLi
Sos1-CLc
Gab1-CLc
13
4
Gab1-Yphos-CLi
Pi3k-CLc
Sos1-reloc-CLi
8
Pi3k-act-CLi
PIP2-CLm
9
PIP3-CLm
Hras-GDP-CLi
6
Hras-GTP-CLi
EgfR-CLm
Egf-Out
1
Egf:EgfR-act-CLm
Grb2-CLc
5
Sos1-CLc
Grb2-reloc-CLi
Gab1-CLc
13
4
Gab1-Yphos-CLi
Pi3k-CLc
Sos1-reloc-CLi
8
Pi3k-act-CLi
PIP2-CLm
9
PIP3-CLm
Hras-GDP-CLi
6
Hras-GTP-CLi

Full Model of EGF Stimulation

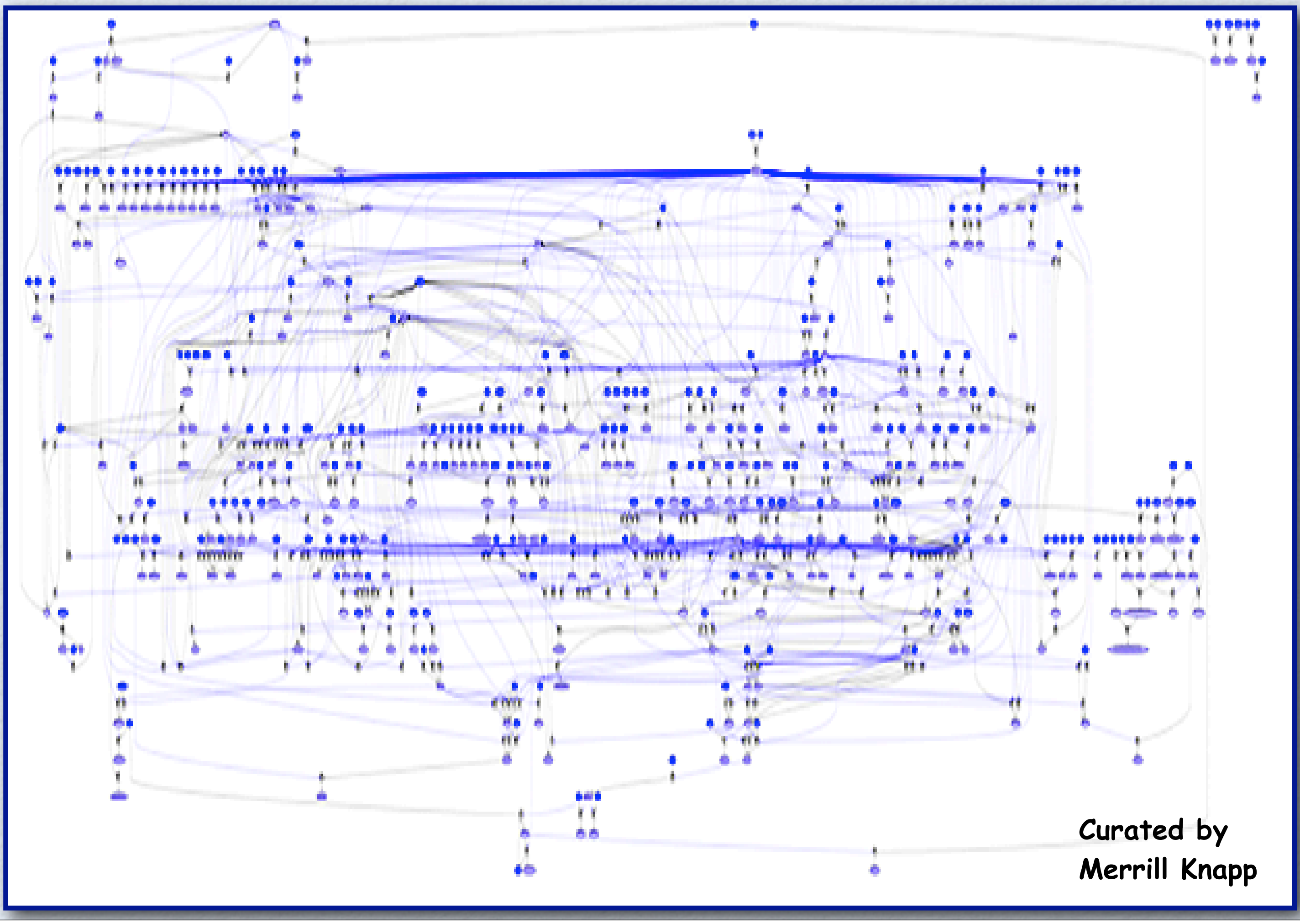
The ErbB Network (cartoon form)



Yarden and Sliwkowski, Nat. Rev. Mol. Cell Biol. 2: 127-137, 20

PL EGF MODEL

Events that could occur in response to Egf



Canonical EGF-ERK Pathway

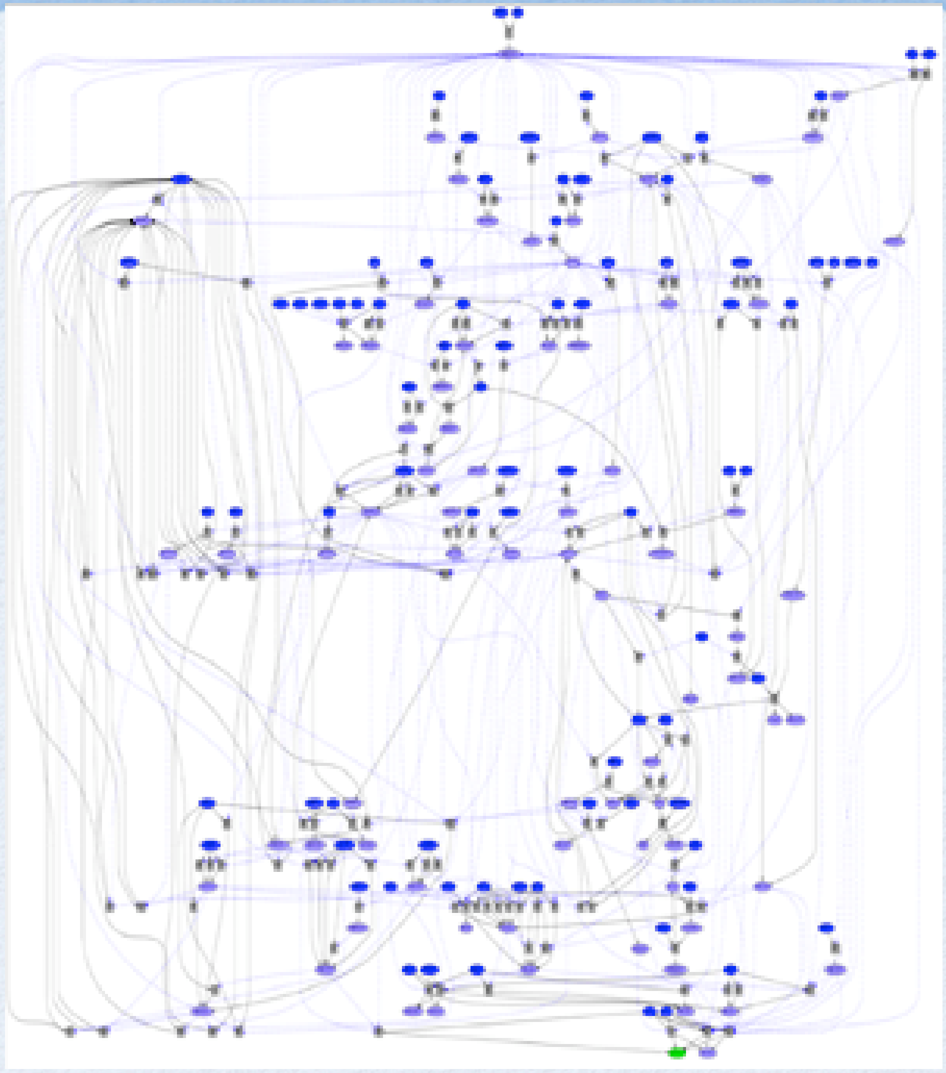
Egf ➔ EgfR ➔ Grb2 ➔ Sos1 ➔ a Ras family member
➔ Raf1 ➔ MEK1/2 ➔ ERK1/2

- Egf binds to the EGF receptor (EgfR), stimulates its protein tyrosine kinase activity, causing autophosphorylation (and activation).
- a complex containing the adaptor protein Grb2 and the guanine nucleotide exchange factor Sos1 docks to the activated EgfR.
- the Sos1-containing EgfR complex activates a Ras family GTPase,
- the activated Ras protein activates Raf1, a member of the RAF serine/threonine protein kinase family.
- Raf1 then activates the dual-specificity protein kinases Mek1 and/or Mek2 (MEK1/2),
- MEK1/2 then activates Erk1 and/or Erk2 (ERK1/2).

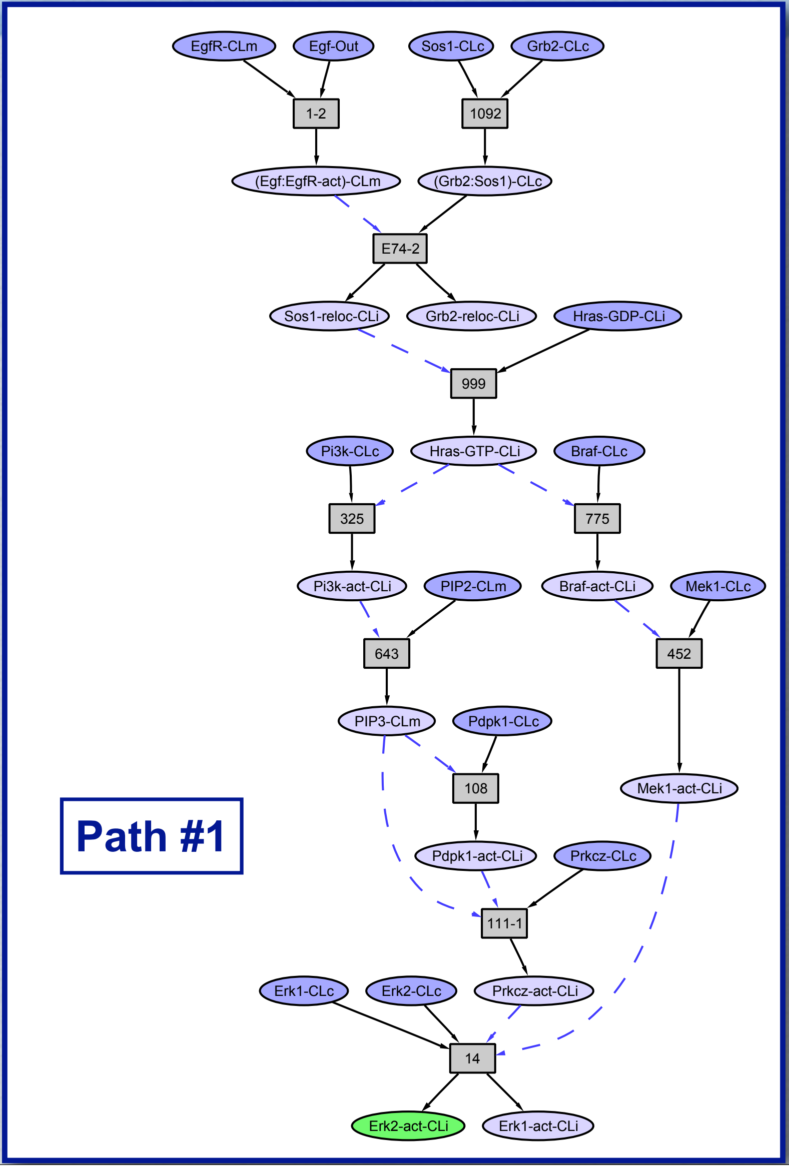
Subnet relevant to Erk activation

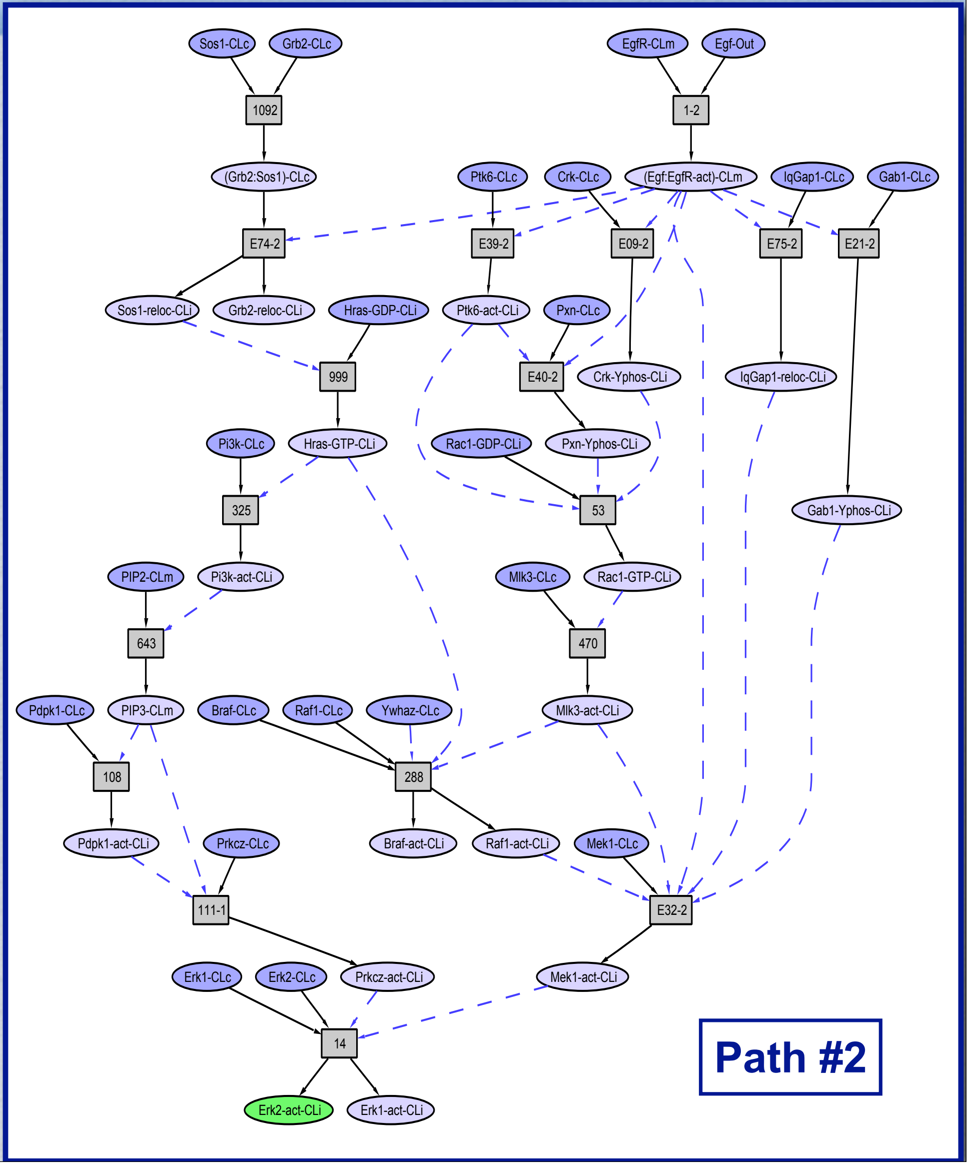
Subnet containing all pathways leading to activation of Erk.

Obtained by backwards followed by forwards collection



Possible Pathways to ERK



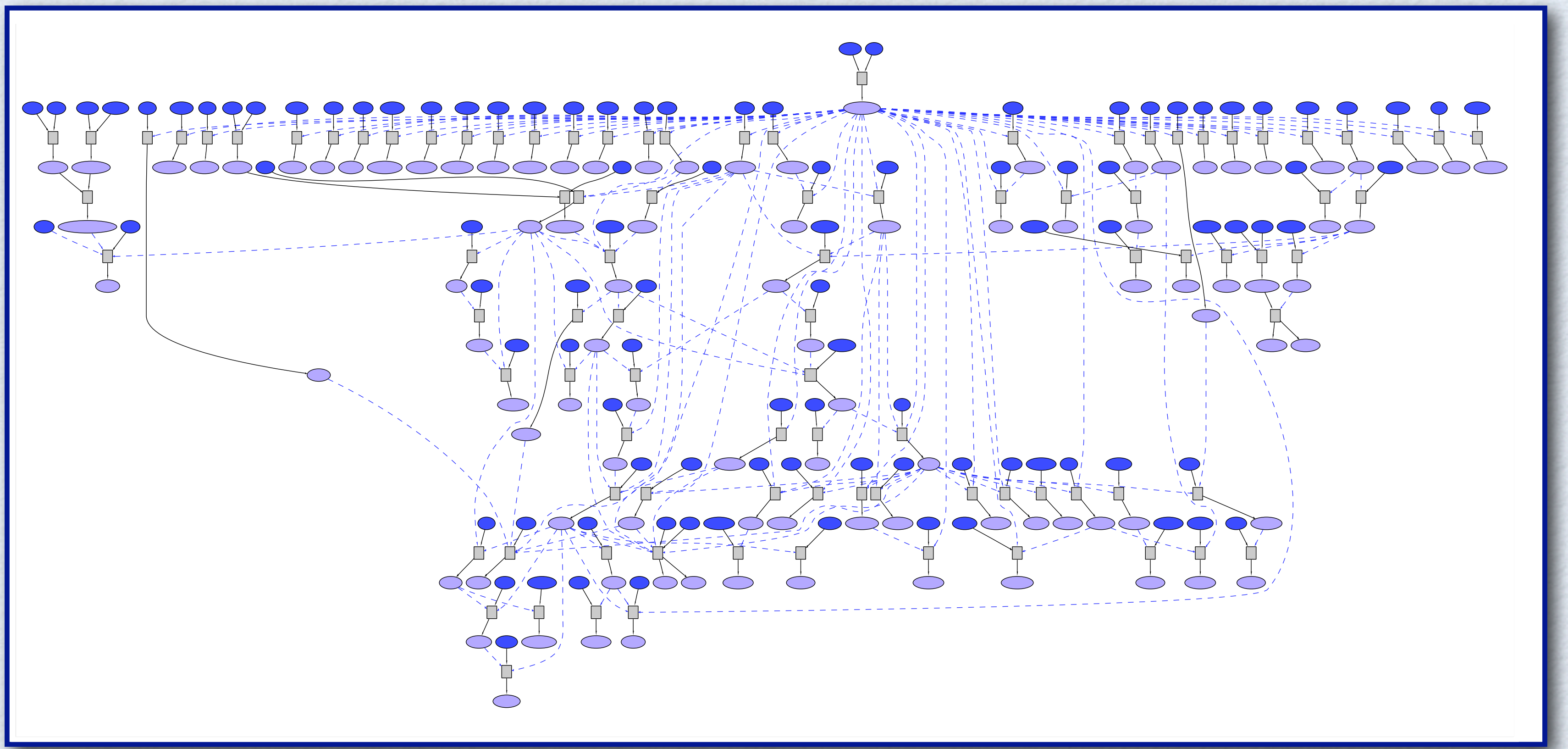


Constraining the EGF Map

The idea is to go from all possible pathways to a plausible set, given the context.

- a list of 85 protein state changes demonstrated experimentally to occur in response to a short stimulus with Egf were set as goals and a set of concurrent paths were produced by PLA. This subnet ensures that the paths used to reach chosen goals are mutually compatible.
- (reachability of all of the goals is also a test of the model)
- Egf Rules, with requirements specific to Egf signaling, were given preference over Common Rules

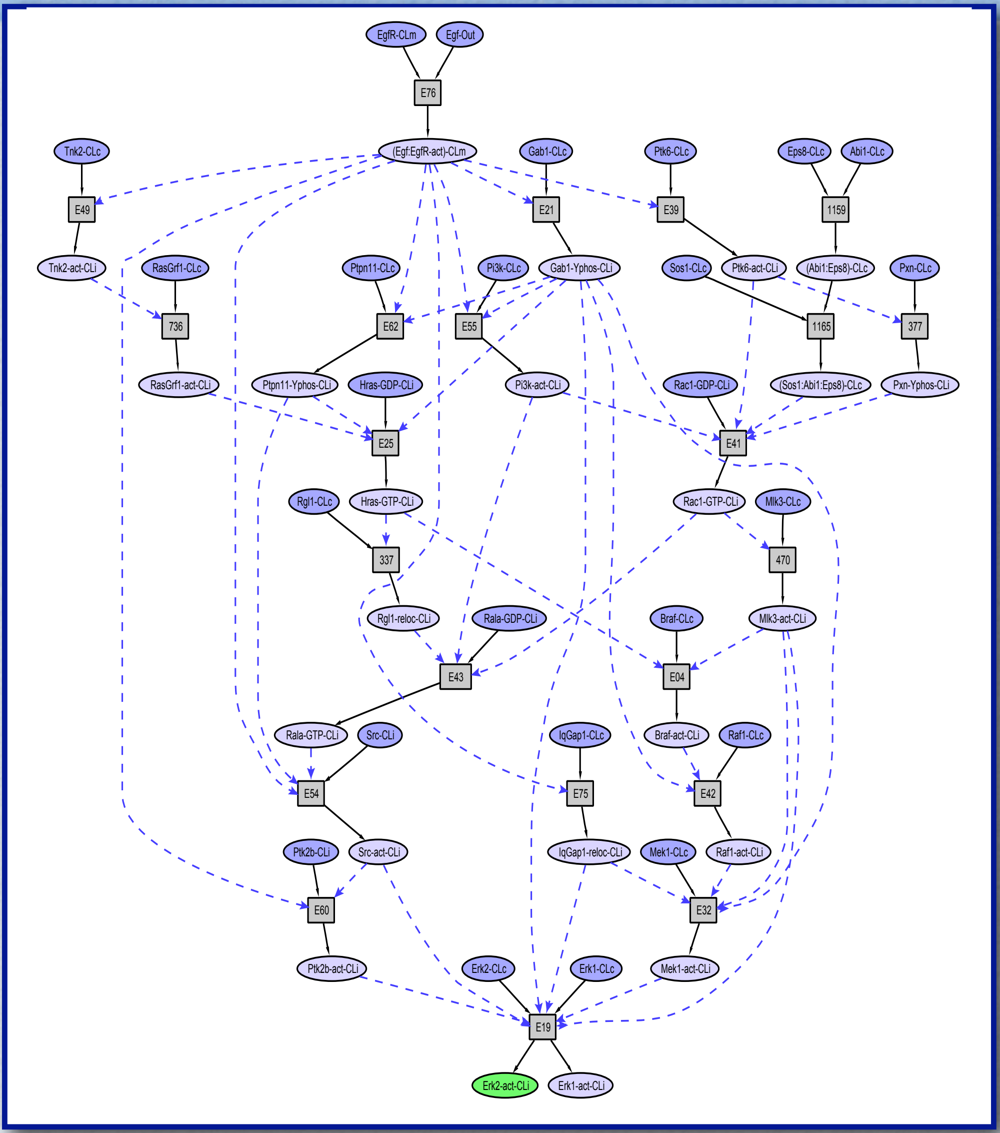
The Constrained EGF Map



Activation of Erk irt Egf

The path leading to activation of Erk1/2 in the constrained Egf network.

This path exists in the context of all the other experimental observations,


EgfR-CLm
Egf-Out
E76
Tnk2-CLc
(Egf:EgfR-act)-CLm
Gab1-CLc
Ptk6-CLc
Eps8-CLc
Abi1-CLc
E49
E21
E39
1159
Tnk2-act-CLi
RasGrf1-CLc
Ptpn11-CLc
Pi3k-CLc
Gab1-Yphos-CLi
Sos1-CLc
Ptk6-act-CLi
(Abi1:Eps8)-CLc
Pxn-CLc
736
E62
E55
1165
377
RasGrf1-act-CLi
Ptpn11-Yphos-CLi
Hras-GDP-CLi
Pi3k-act-CLi
Rac1-GDP-CLi
(Sos1:Abi1:Eps8)-CLc
Pxn-Yphos-CLi
E25
E41
Rgl1-CLc
Hras-GTP-CLi
Rac1-GTP-CLi
Mlk3-CLc
337
470
Rgl1-reloc-CLi
Rala-GDP-CLi
Braf-CLc
Mlk3-act-CLi
E43
E04
Rala-GTP-CLi
Src-CLi
IqGap1-CLc
Braf-act-CLi
Raf1-CLc
E54
E75
E42
Ptk2b-CLi
Src-act-CLi
IqGap1-reloc-CLi
Mek1-CLc
Raf1-act-CLi
E60
E32
Ptk2b-act-CLi
Erk2-CLc
Erk1-CLc
Mek1-act-CLi
E19
Erk2-act-CLi
Erk1-act-CLi