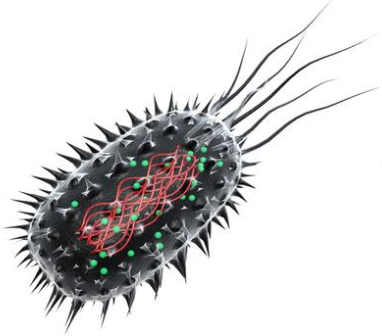
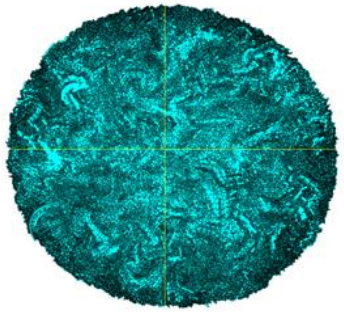
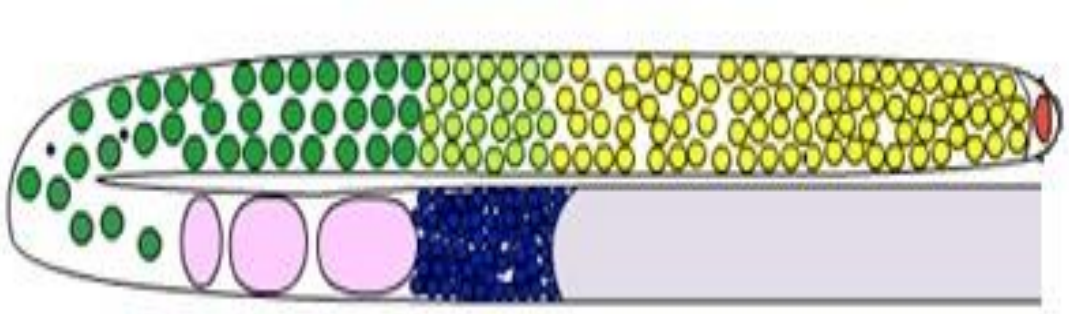
Formal Verification for Natural and Engineered Biological Systems

Hillel Kugler
Faculty of Engineering, Bar-Ilan University, Israel

FMCAD'20

21 September 2020

Formal Verification has proven useful in Reactive Systems Development (Software/Hardware)

What are the main uses / challenges / future research directions in Biology?

Why biology?
What has been achieved so far ?
Where the field is going?

Formal verification can be very powerful but we first need:

- Accurate Computational Models
- Relevant Biological Questions

In this tutorial:

- Do not cover lots of important work
- Recommend looking at proceedings of CMSB Computational Methods in Systems Biology annual conference and DNA Computing and Molecular Programming

Natural vs. Engineered

Biology – understanding life and predicting system dynamics

Building biological devices robustly

Gene Regulatory Networks
RE:IN

DNA Strand Displacement (DSD)
Network Base Biocomputation (NBC)

Logical Models,
Boolean Networks

Chemical Reactions Networks (CRN)

Natural Biological Systems

The basic unit is the Cell

Single Cell / Multi-Cellular

Genotype to Phenotype

Modeling Formalisms – Natural Systems

Case Study – *C. elegans* VPC

How cells decide to differentiate

System is ‘classical’ in Biology and attracted many modeling efforts

C. elegans

A Model Organism

Small (1mm long,959 cells)
Transparent

Short life cycle (~3 days)

Can freeze and use later
Fixed development
Genome is Sequenced
Powerful experimental
techniques available
Data on the same worm
Research community has a
tradition of sharing
resources

Success recognized in several Nobel Prizes

The Nobel Prize in Physiology or Medicine 2002

"for their discoveries concerning 'genetic regulation of organ development and programmed cell death'"

Sydney Brenner

H. Robert Horvitz

John E. Sulston

Programmed Cell Death

The Nobel Prize in Physiology or Medicine 2006

"for their discovery of RNA interference - gene silencing by double-stranded RNA"

Photo: L. Cicero/Stanford

Andrew Z. Fire

Photo: R. Carlin/UMMAS

Craig C. Mello

RNAi

The Nobel Prize in Chemistry 2008

Osamu Shimomura, Martin Chalfie, Roger Y. Tsien

The Nobel Prize in Chemistry 2008

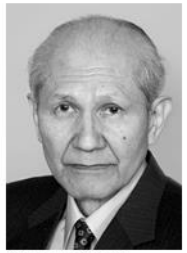


Photo: U. Montan

Osamu Shimomura

Photo: U. Montan

Martin Chalfie

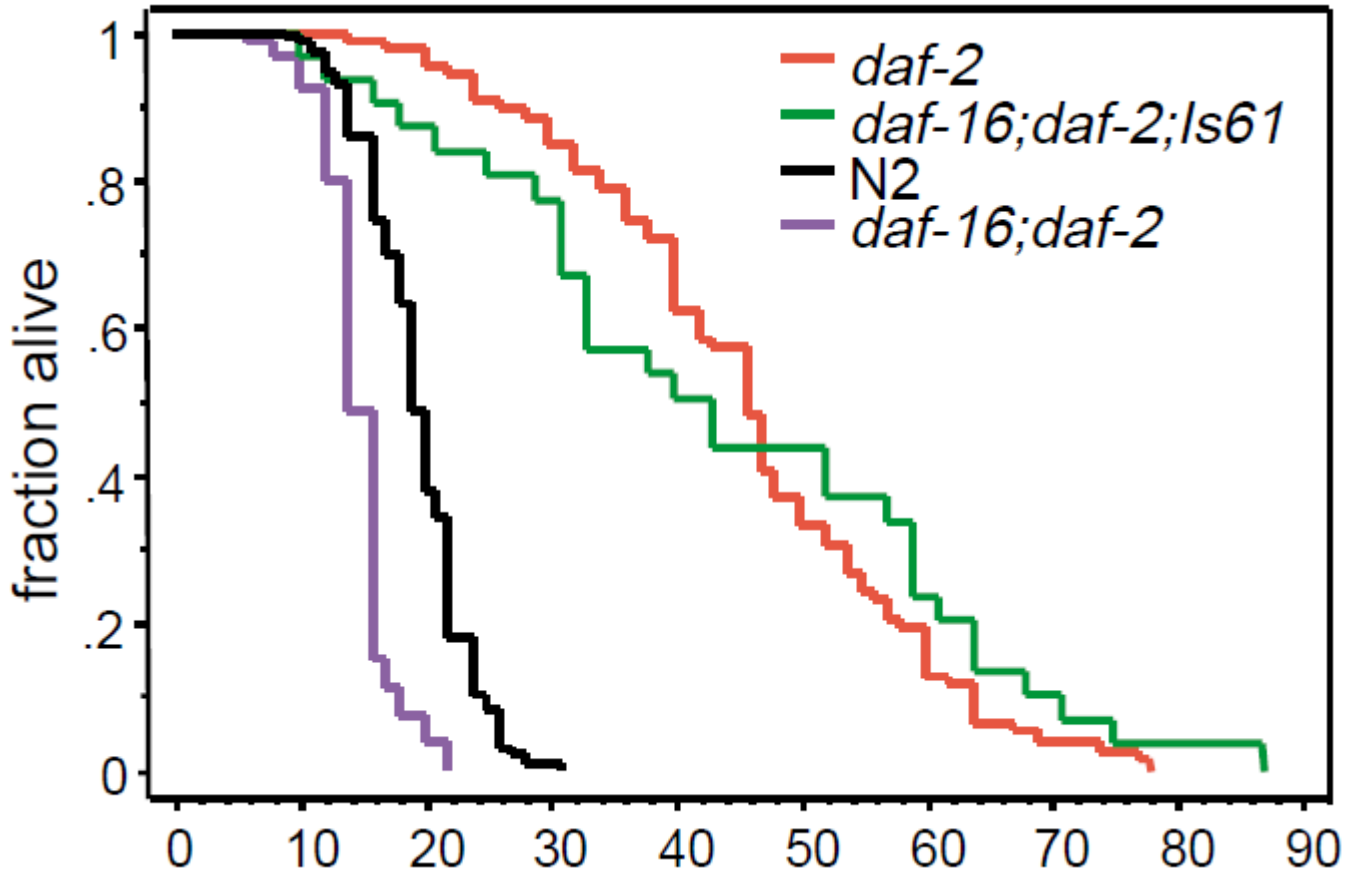
Photo: U. Montan

Roger Y. Tsien

The Nobel Prize in Chemistry 2008 was awarded jointly to Osamu Shimomura, Martin Chalfie and Roger Y. Tsien *"for the discovery and development of the green fluorescent protein, GFP".*

GFP

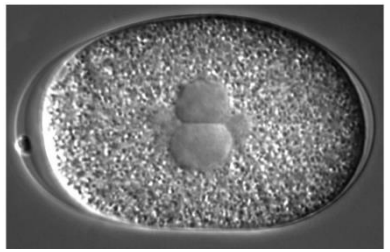
… and genetic regulation of aging

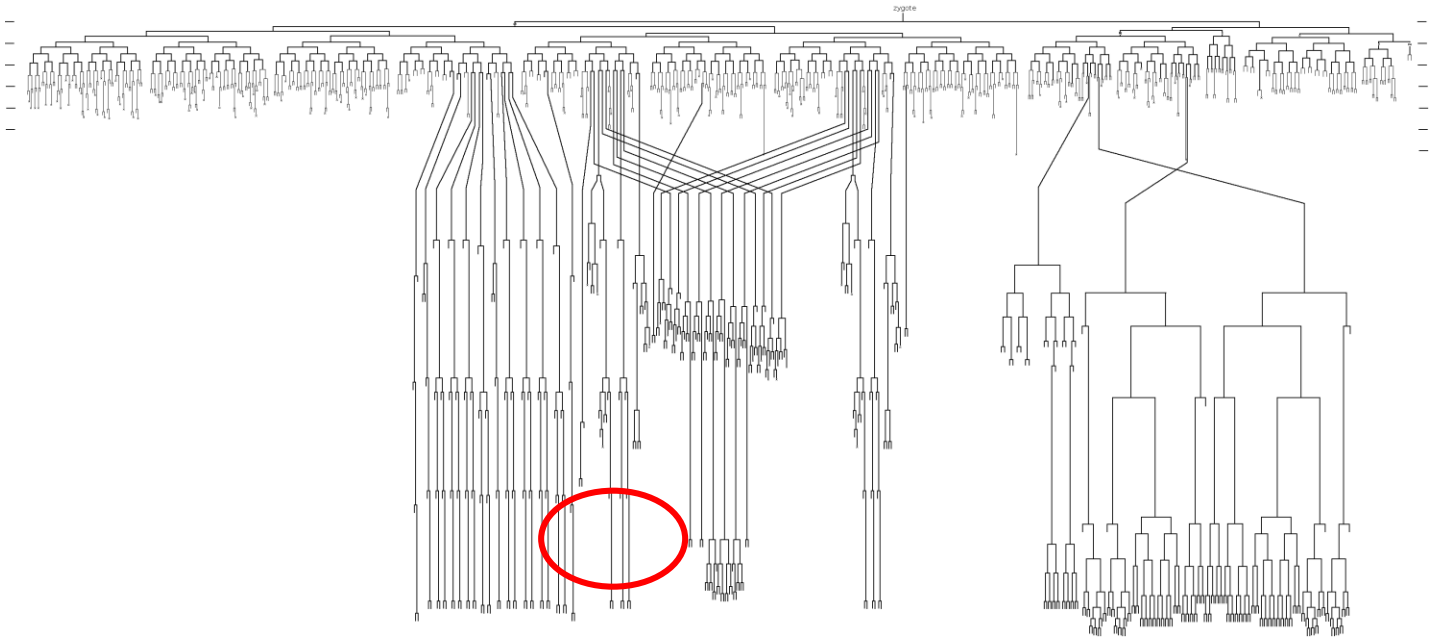


Kenyon et al. Nature 93

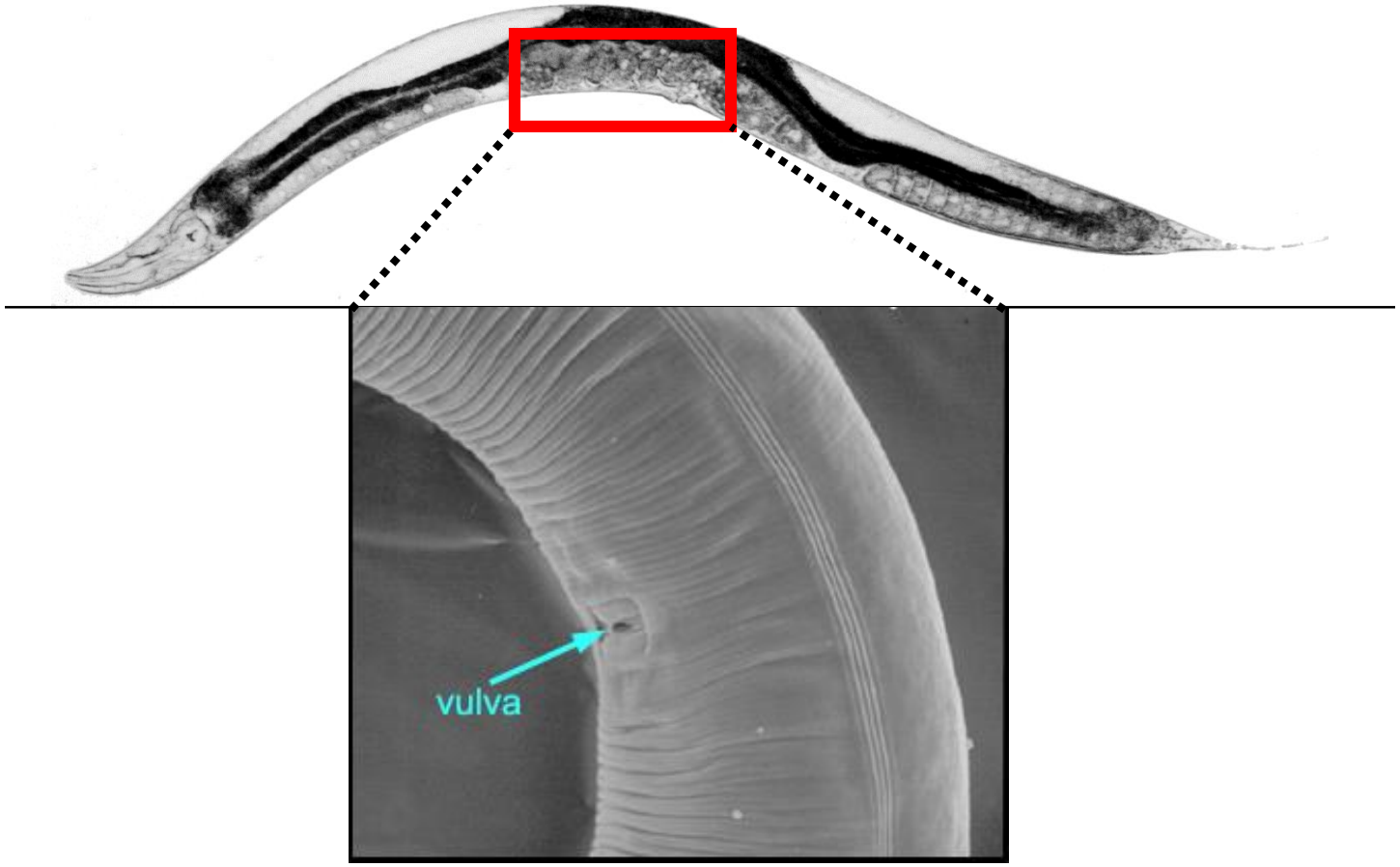
Cell fate specification

Sulston and Horvitz, 1977
Kimble and Hirsh, 1979
Sulston et al., 1983

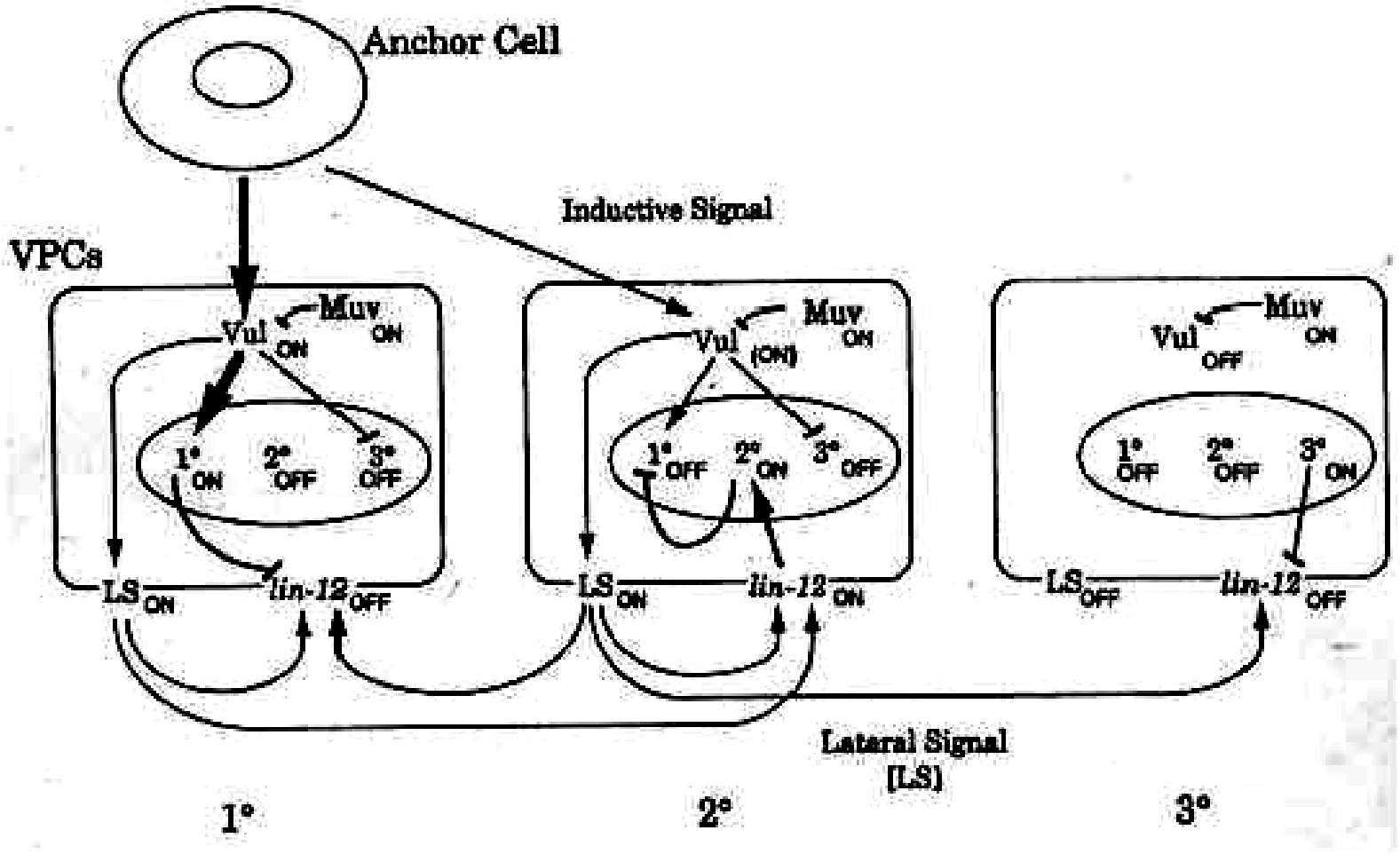




A Modeling Proof-of-Principle

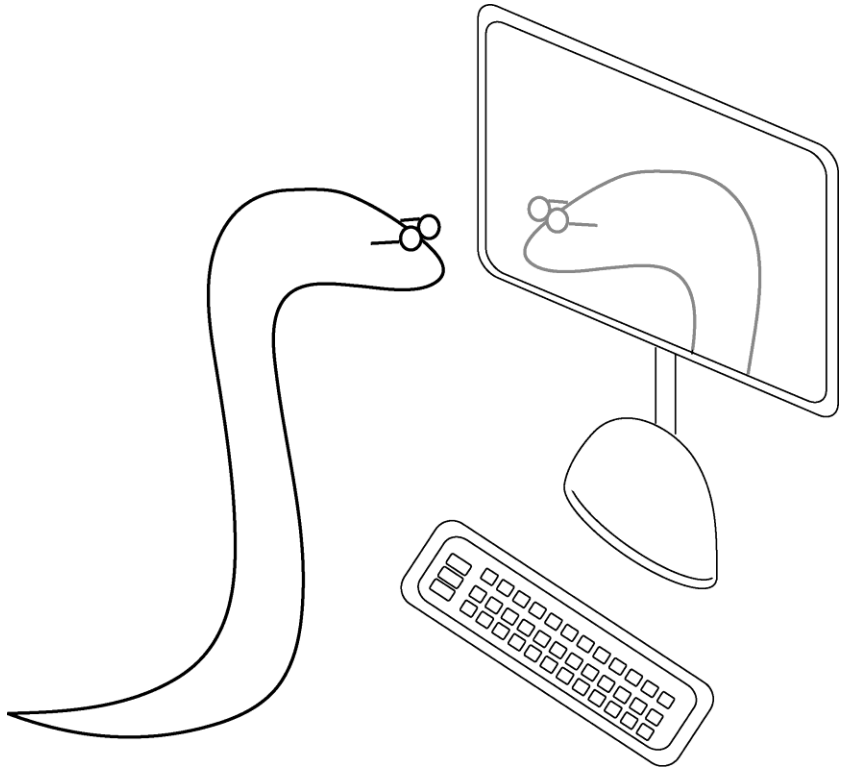


Biologists think in terms of models

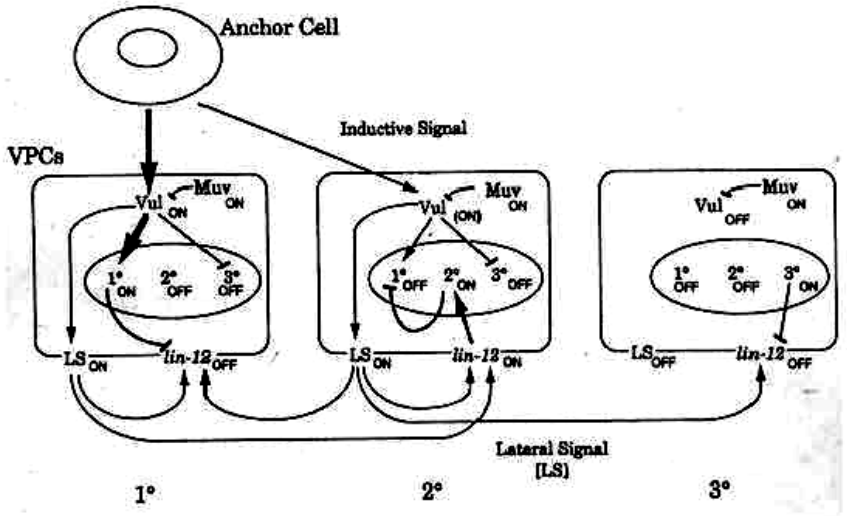


from Sternberg & Horvitz (1989) Cell 58:679

A Modeling Proof-of-Principle



What's wrong with our models?

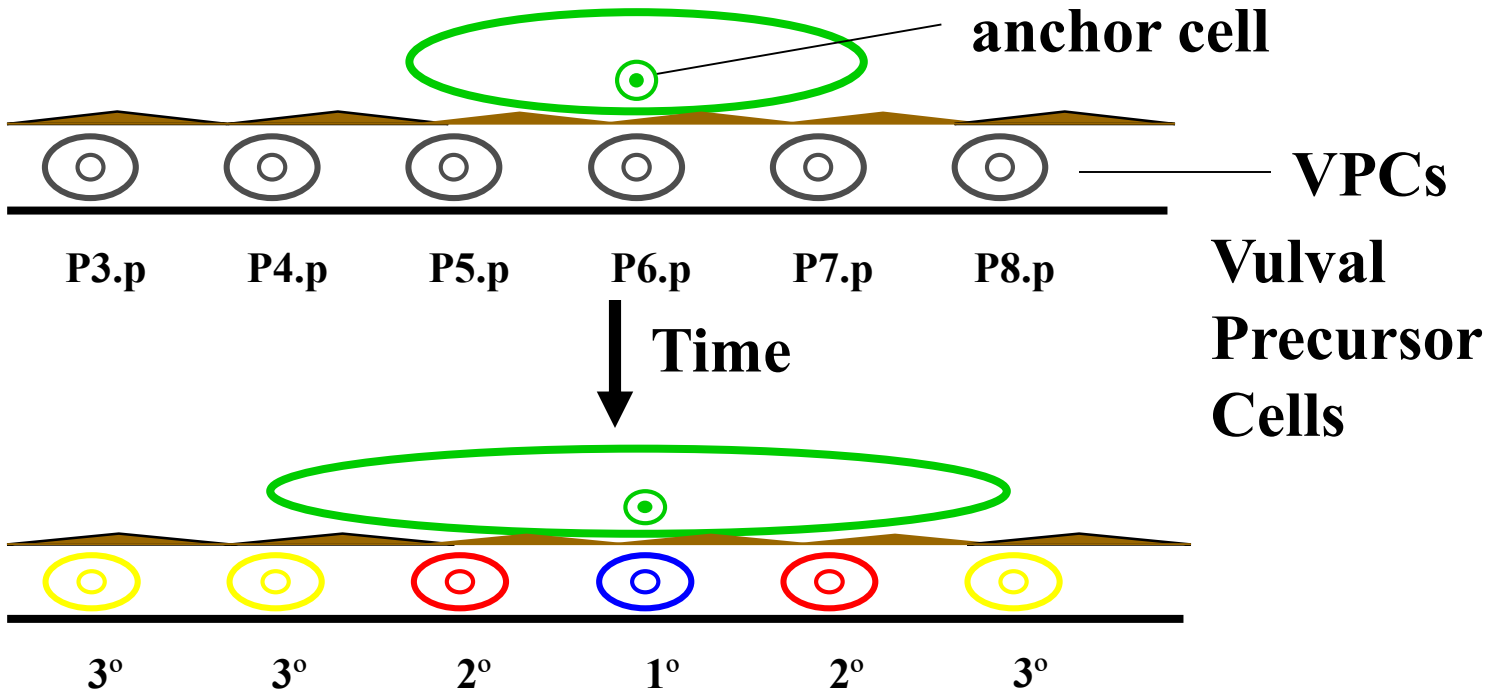


Difficult to predict system behavior

- Time
- Concurrency
- Distributed Control
- Interaction with other components

And this will get worse for larger systems !

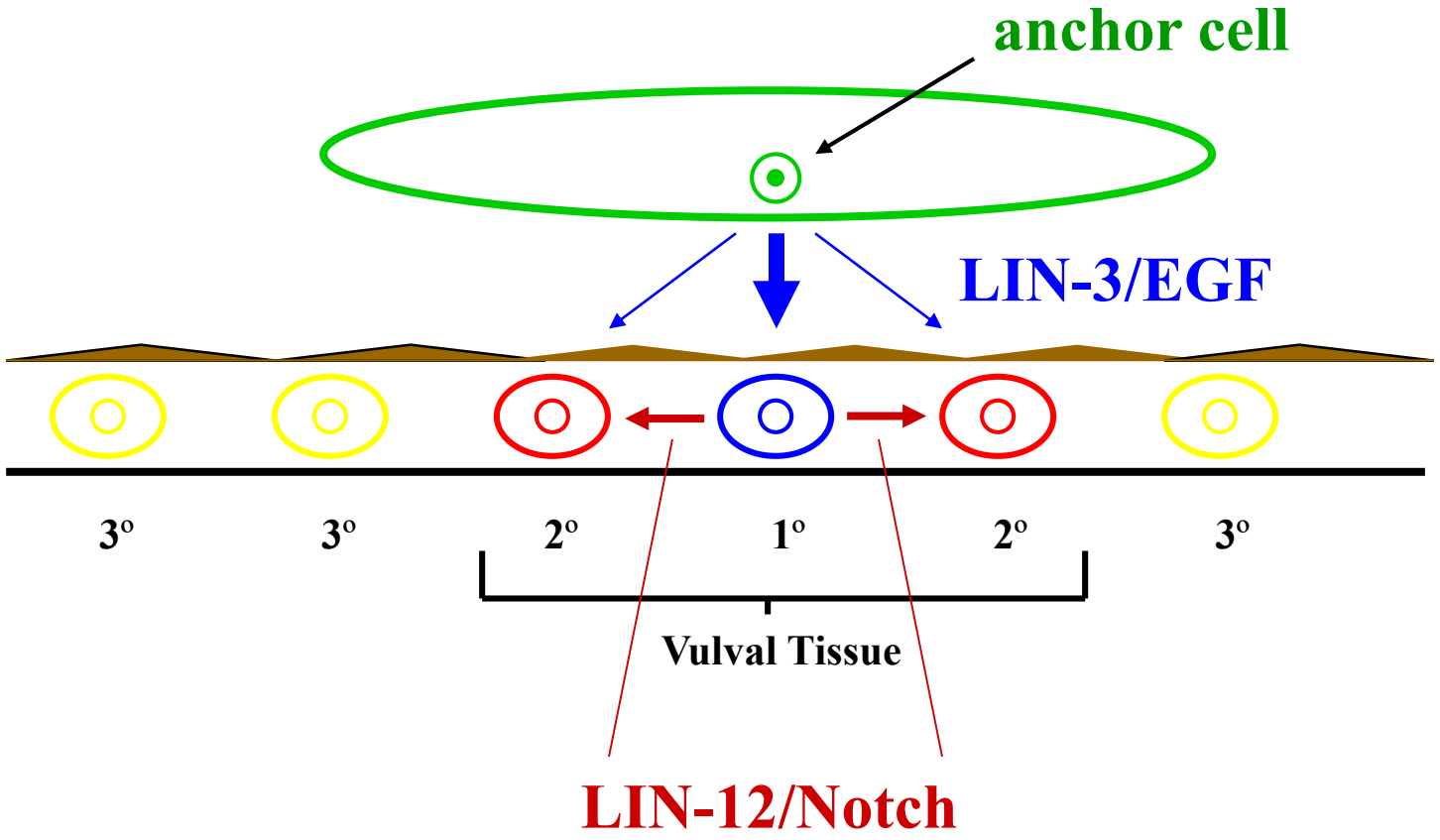
Vulval Fates



1° Fate **2° Fate** **3° Fate**

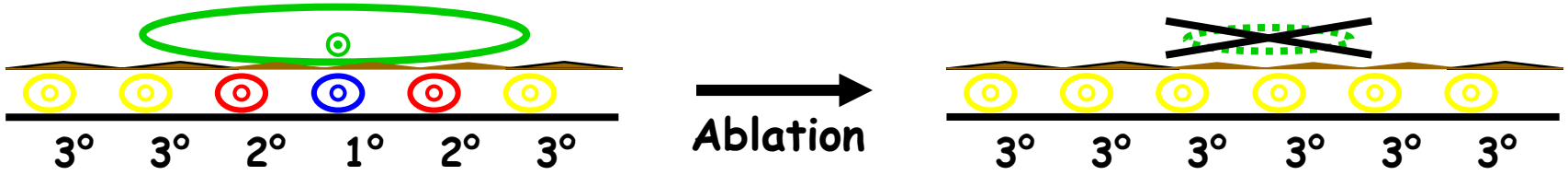
vulval fates **non-vulval fate**

VPCs form an equivalence group
The normal pattern of fates is specified by cell-cell interactions

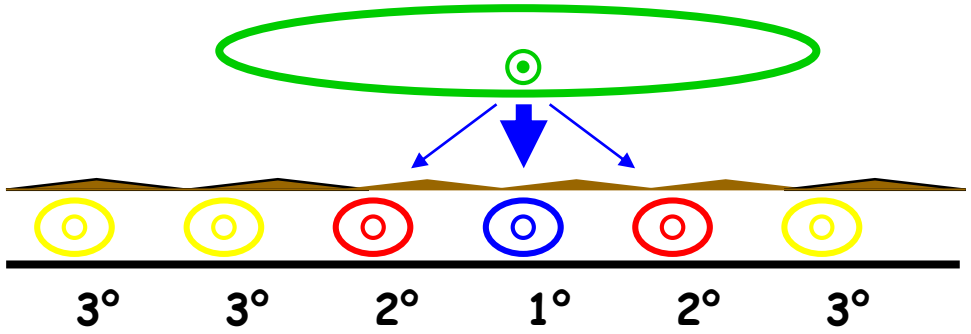


Biological understanding based on logical inferences

Condition/result: ablation of the gonad abolishes induction



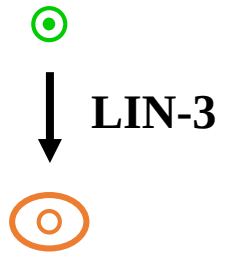
Inferred 'mechanism': a gonadal signal induces vulval formation



How do we express this so the computer can understand it?

Background for *lin-15(-)* Modeling

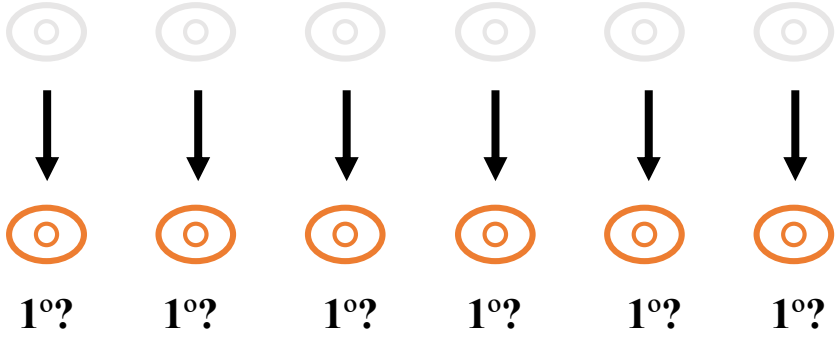
The AC induces VPCs to become 1°



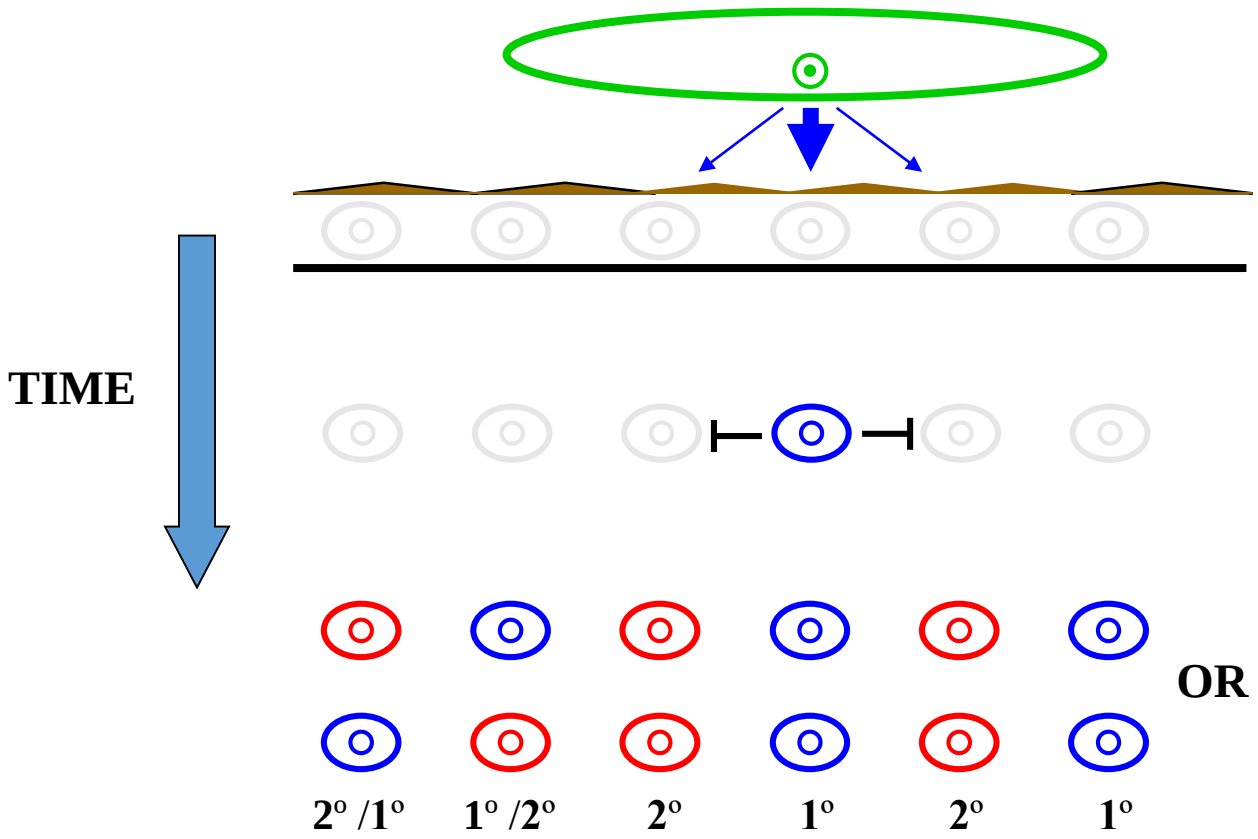
In *lin-15(-)*, all VPCs become 1° unless prevented by adjacent VPCs

1° VPCs prevent adjacent VPCs from becoming 1° (via LIN-12/Notch)

Thus, in *lin-15(-)* mutants, the VPCs all race to become 1°



Postulated Mechanism:
Early Activation of the Inductive Pathway Biases P6.p to Become 1º

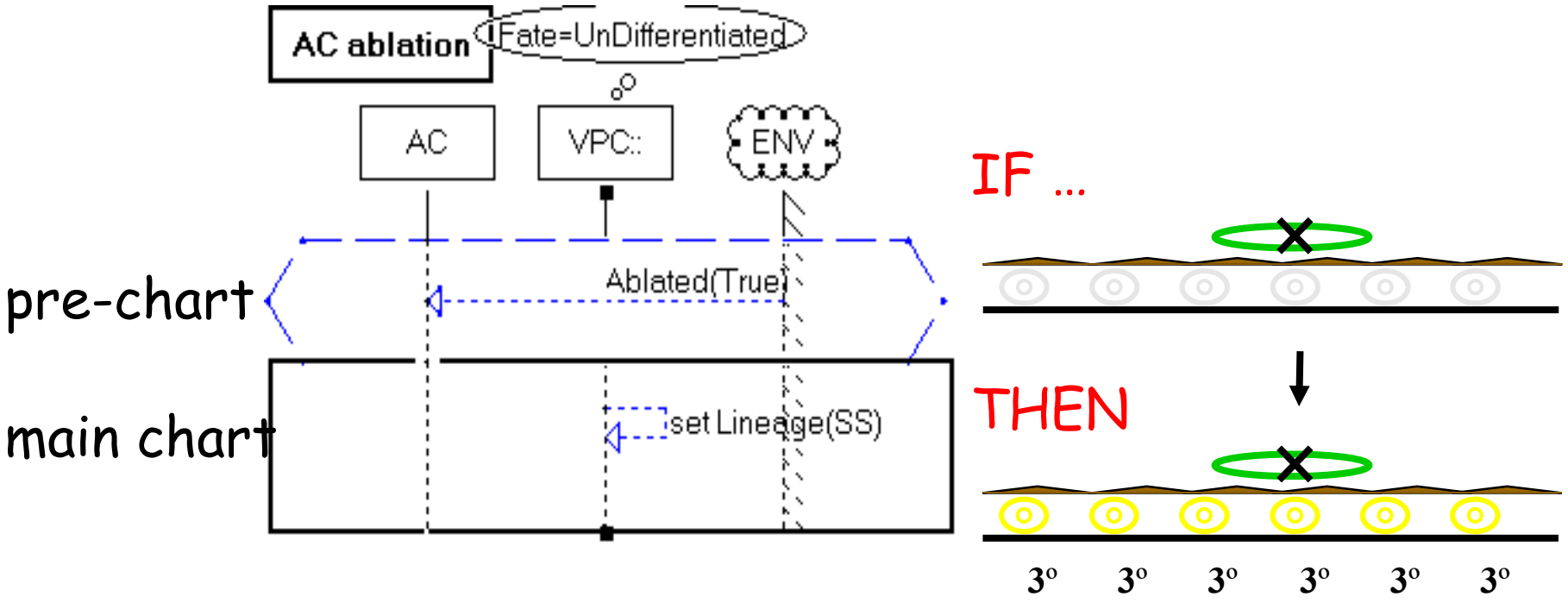


Modeling Formalisms for VPC Models

Temporal Logic
Live Sequence Charts
Statecharts, Reactive Modules
Petri Nets
Boolean Networks
Ordinary Differential Equations
Dynamic Bayesian Networks

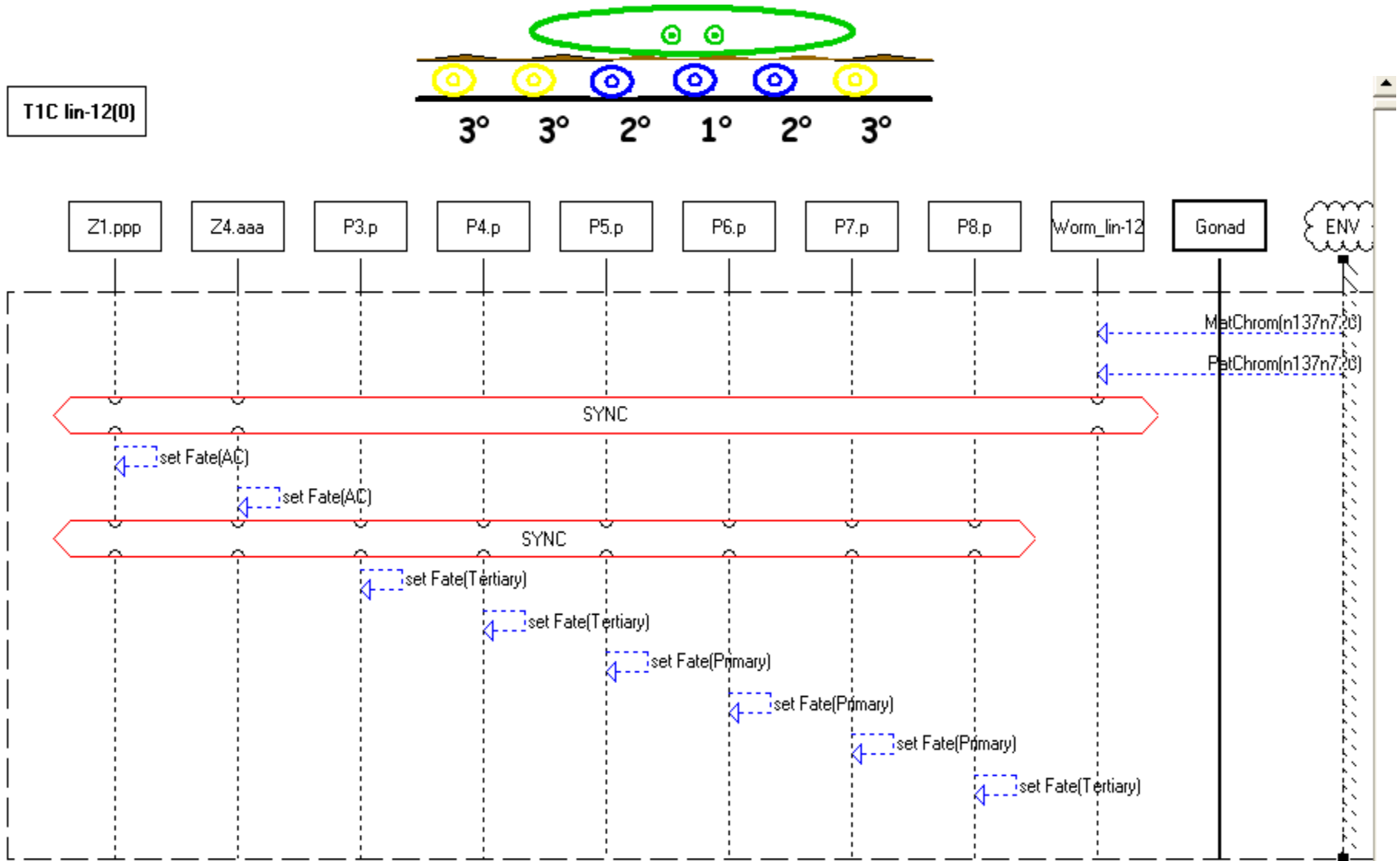
Basic form of a universal LSC

Structure is similar to an experiment or inference



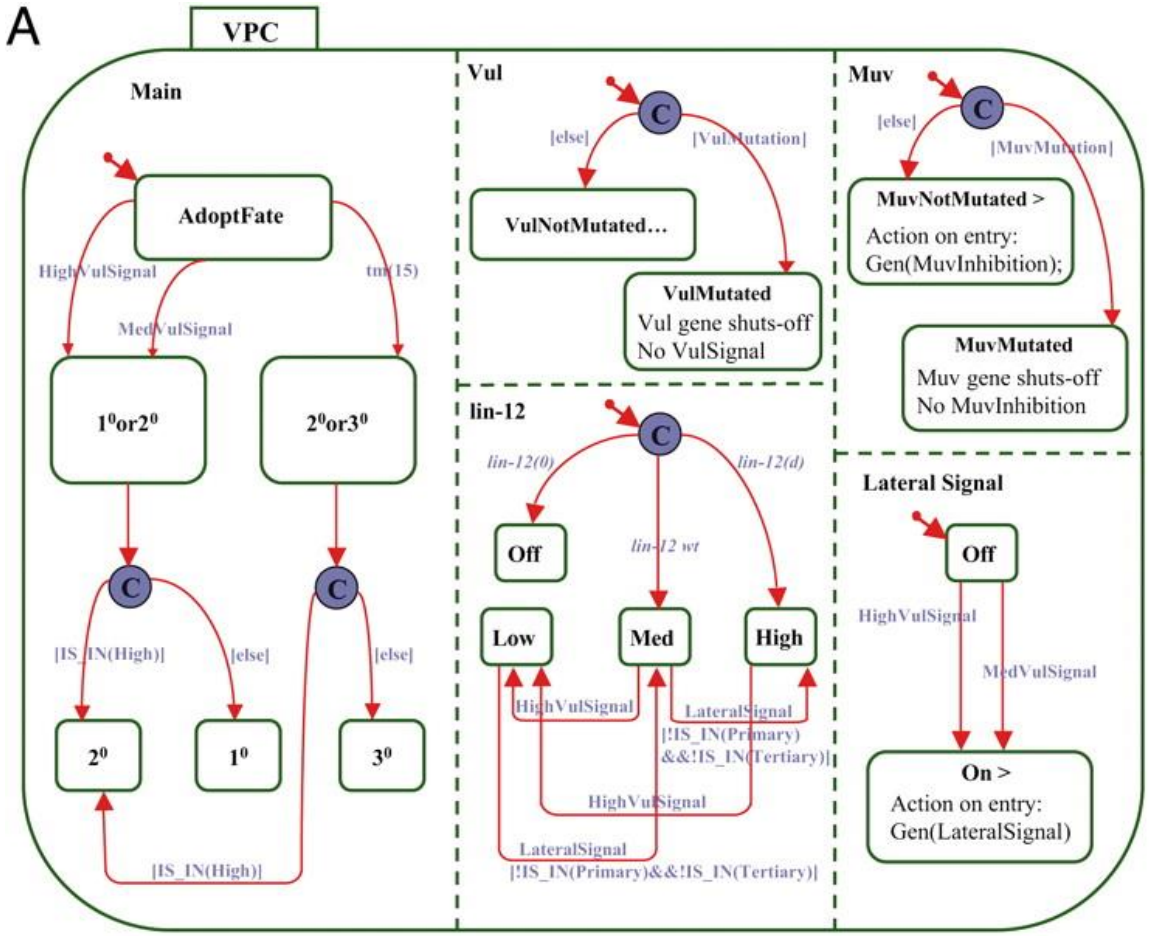
Kam et al 2004 *CMSB*, Kam et al 2008 *Dev Bio*

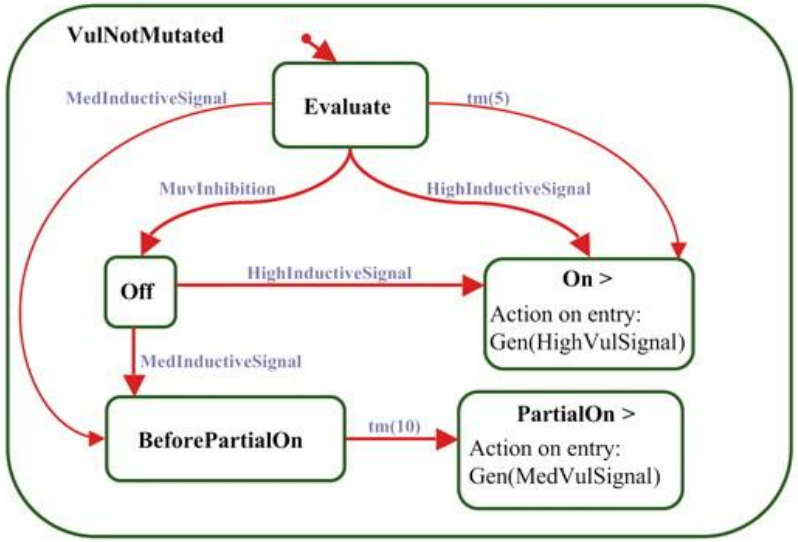
Existential LSC



Kam et al 2004 *CMSB*, 2008 *Dev Bio*

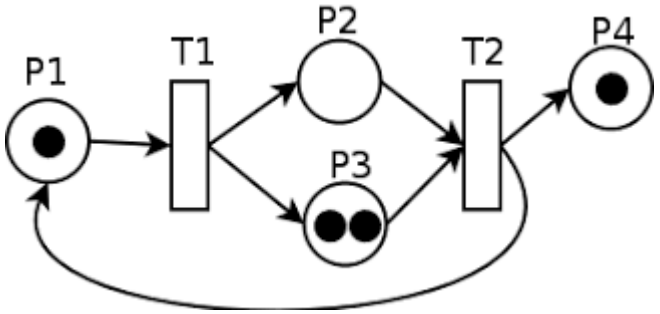
Statecharts (Harel 87)

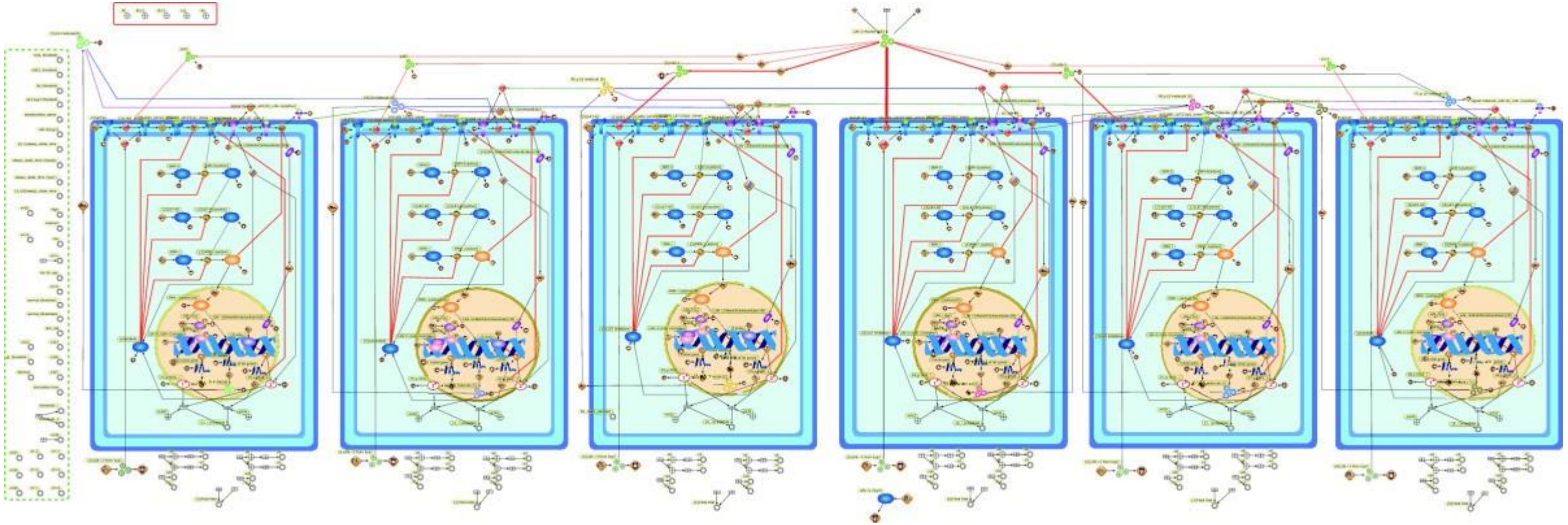




Fisher et al 2005 *PNAS*

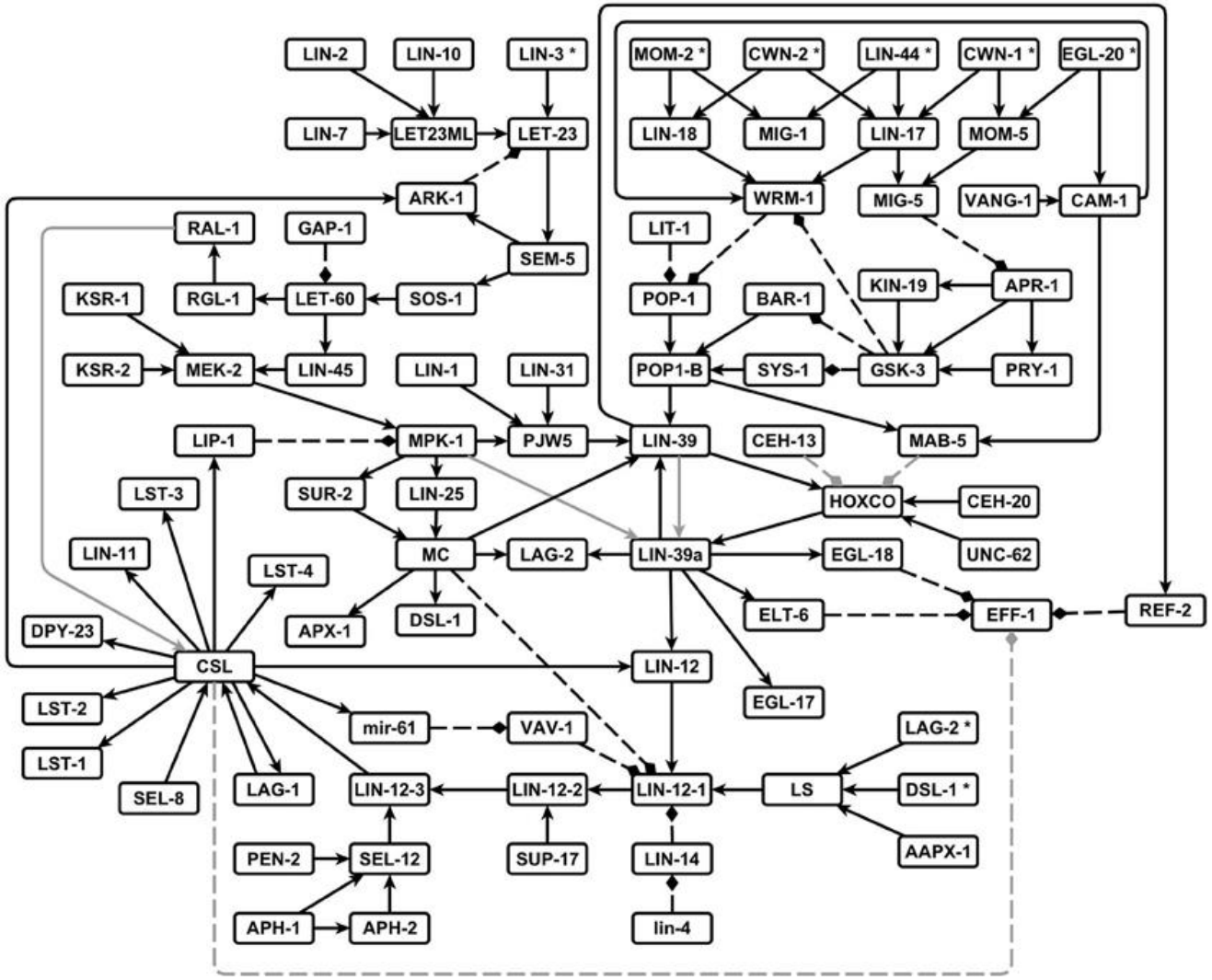
Petri Nets (Petri 63)





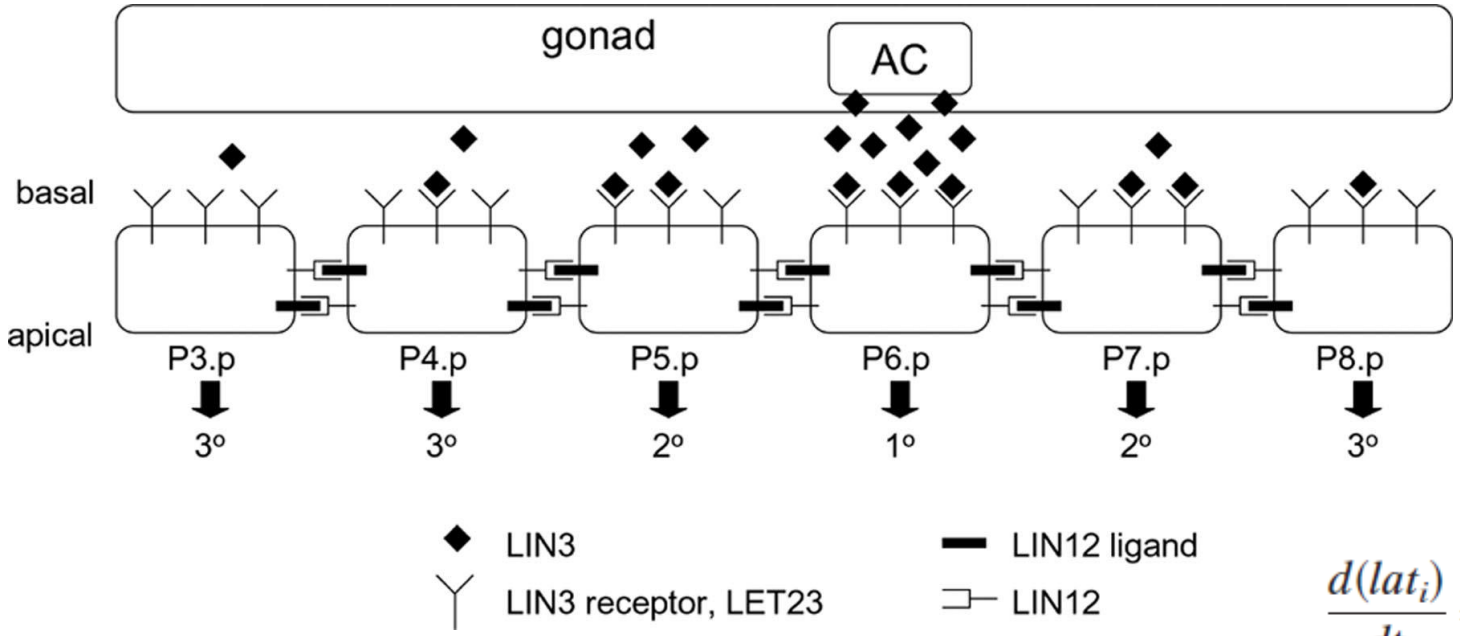
Weinstein and Mendoza 2013 *Front in Genetics*

Boolean Networks + Extensions (Kaufman 69)



Weinstein and Mendoza 2013 *Front in Genetics*

Ordinary Differential Equations



$$\frac{d(lat_i)}{dt} = k_n^+ - k_n^-(lat_i) - k_{x_2}(mpk_i^*)(lat_i)$$

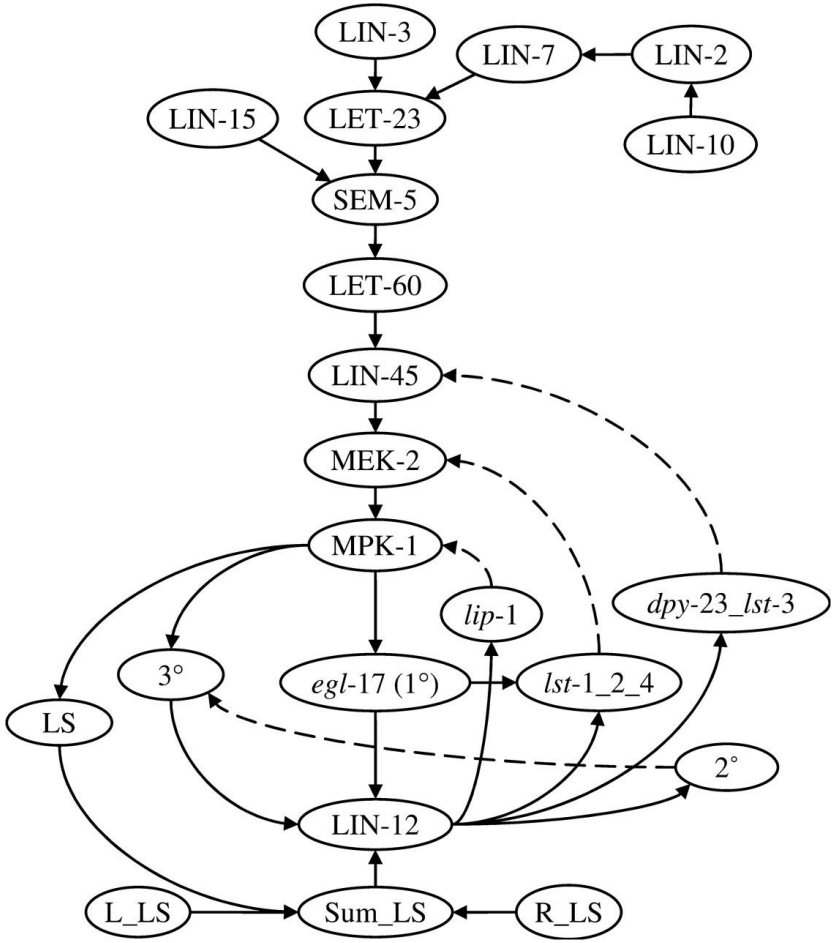
$$\frac{d(mpk_i^*)}{dt} = k_m^+ Ind_i(mpk_i) - k_m^-(Ph_T)(mpk_i^*)$$

$$- k_{x_1}\frac{(lat_i)^2}{K_{M_{lat}}^2 + (lat_i)^2}(mpk_i^*),$$

$$+ k_{x_3}\frac{\left(\frac{mpk_{i+1}^*}{\nu_{i+1}} + \frac{mpk_{i-1}^*}{\nu_{i-1}}\right)^2}{K_{M_{mpk}}^2 + \left(\frac{mpk_{i+1}^*}{\nu_{i+1}} + \frac{mpk_{i-1}^*}{\nu_{i-1}}\right)^2},$$

Giurumescu Sternberg, and Asthagiri 2005 *PNAS*

Dynamic Bayesian Networks



Sun and Hung 2007 *Bioinformatics*

Verification of VPC models

Temporal Logic

Sequence Charts

Statecharts

Boolean Networks

Petri Nets

Using Temporal Logic in Biology

Fisman and Kugler, ISOLA 2018

Using LTL:

"If p2 is not present to stimulate its pathway, but p1 is, is the p3 signal silent ?"

$\Box(\Box(p1 \wedge \neg p2) \rightarrow \neg \diamond p3)$ (alternatively, using truncated semantics in neutral view)

$$\underset{\sim}{\mathbf{G}}\,(\,\underset{\sim}{\mathbf{G}}\,(p1 \wedge \neg p2) \rightarrow \neg\, \underline{\mathbf{F}}\, p3)$$

Eker et al 01

Necessity of eventually reaching a state in which two signals p1 and p2 are activated from some initial state q1

Eker et al 04

$q1 \rightarrow \underline{\mathbf{F}}\,(p1 \wedge p2)$

Using Temporal Logic in Biology

Using CTL: Branching logic reasons about the tree of computations
E, A path quantifiers
E – there exists a path A – for all paths

[Montiero et al. 08] classify biological specification into patterns:

1) Occurrence/Exclusion pattern
"It is possible for a state p to occur" EF (p)
"It is not possible for a state p to occur" ¬ EF (p)

Could use LTL and then truncated semantics is potentially relevant :
$\underset{\sim}{\mathbf{G}}(\neg p)$ does not hold for occurrence EF (p)
$\underset{\sim}{\mathbf{G}}(\neg p)$ holds for exclusion ¬ EF (p)

Monteiro et al 08

Temporal Logics Patterns

2) Consequence pattern

"If a state p occurs then it is possibly followed by a state q"

AG(p → EF q)

"If a state p occurs then it is neccessarily followed by a state q"

AG(p → AF q)

AG(p → EF q) possible occurrence is not in LTL

$\mathbf{G}\,(p \rightarrow \underline{\mathbf{F}}\,q)$ holds for necessary consecution AG(p → AF q)

Monteiro et al 08

Temporal Logics Patterns

3) Sequence pattern

"A state q is reached and is possibly preceded at some time by a state p"

$EF(p \wedge EF(q))$

"A state q is reached and is possibly preceded at all times by a state p"

$E(p \, U \, q)$

"A state q is reached and is necessarily preceded at some time by a state p"

$EF(q) \wedge \neg E((\neg p) \, U \, q)$

"A state q is reached and is necessarily preceded at all times by a state p"

$EF(q) \wedge \neg E(true) \, U \, (\neg p \wedge E((true) \, U \, q)$

Monteiro et al 08

Temporal Logics Patterns

4) Invariance pattern

"A state p can persist indefinitely"

EG (p)

"A state p must persist indefinitely"

AG (p)

Monteiro et al 08

Additional related patterns:

"Can the system reach a given stable state s?"

EF (AG (s))

"Must the system reach a given stable state s?"

AF (AG (s))

AF (AG (s)) cannot be expressed in LTL (different than F G p)

Chabrier-Rivier et al 04

Invariance and Stabilization

Stabilization: $\exists k_1, k_2, \cdots k_n s.t. \underline{\mathbf{F}}\ \underset{\sim}{\mathbf{G}}\ (\forall v_i . v_i = k_i)$

Stabilization in BMA (Fisher) "Exists a unique state that is eventually reached in all executions"

Formula requires quantification on values and variables so cannot directly be expressed in propositional temporal logic

$\underline{\mathbf{F}}\ \underset{\sim}{\mathbf{G}}\ (s)$ cannot be expresses in CTL (is different than AF (AG (s)) discussed before)

BMA supports GUI for patterns

$\text{NEXT}\varphi \stackrel{\text{def}}{=} \underline{\mathbf{X}}\ \varphi$ $\quad$ $\text{ALWAYS}\varphi \stackrel{\text{def}}{=} \underset{\sim}{\mathbf{G}}\ \varphi$ $\quad$ $\text{EVENTUALLY}\varphi \stackrel{\text{def}}{=} \underline{\mathbf{F}}\ \varphi$

$\varphi_1 \text{UNTIL} \varphi_2 \stackrel{\text{def}}{=} [\varphi_1\ \underline{\mathbf{U}}\ \varphi_2]$ $\quad$ $\varphi_1 \text{WEAKUNTIL} \varphi_2 \stackrel{\text{def}}{=} [\varphi_1\ \underset{\sim}{\mathbf{W}}\ \varphi_2]$ $\quad$ $\varphi_1 \text{RELEASE} \varphi_2 \stackrel{\text{def}}{=} [\varphi_2\ \underset{\sim}{\mathbf{W}}\ (\varphi_1 \wedge \varphi_2)]$

Cook et al 11
Benque et al 12

Formal Verification for LSCs

Inherent nondeterminism in executing scenarios

Can be resolved using formal verification (Smart Play-Out)

$$G(\bigvee_{m_i \in M^U} (act_{m_i} = 1))$$

Existential charts can be considered as properties that system needs to satisfy

HKMP 2002, FHPSS 2005

Formal Verification for LSCs

LSCs can also be directly translated to temporal logic

Definition 3. *Let $w = m_1 m_2 m_3 ... m_k$ be a finite trace. Let $R = \{e_1, e_2, e_3 \cdots e_l\}$ be a set of events. The temporal logic formula ϕ_w^R is defined as:*

$$\phi_w^R = NU(m_1 \wedge (X(NU(m_2 \wedge (X(NU(m_3 ...)))))))),$$

where the formula N is given by $N = \neg e_1 \wedge \neg e_2 ... \wedge \neg e_l$.

Definition 4. *Let $LS = \langle M, amsg, mod \rangle$ be an LSC specification. For a chart $m \in M$, we define the formula ψ_m as follows:*

- *If $mod(m) = universal$, then $\psi_m = AG(amsg(m) \rightarrow X(\bigvee_{w \in \mathcal{L}_m^{trc}} \phi_w^R))$.*
- *If $mod(m) = existential$, then $\psi_m = EF(\bigvee_{w \in \mathcal{L}_m^{trc}} \phi_w^R)$.*

KHPLB05, KPP11

Statecharts (and other state-based languages)

Exhaustive testing of statechart based models [Sadot]

Challenges for verification

- Extensions of statecharts
  - C++ code
  - Variables
  - Dynamic object construction

Reactive Modules and Mocha tool [Fisher, and Henzinger]

Sadot et al. 2006 ACM/TCBB 2002, Fisher et al 2005

Petri Nets (Petri 63)

Computation of Attractors [Chatain et al]

Monte Carlo Simulations [Krepska et al]

Simulation Based Model Checking [Li and Miyano]

Colored Petri Nets Verification Tools [Liu and Heiner]

Chatain et al. *CMSB* 2014, Krepska et al *FMSB* 2008, Li et al. *BMC Sys Bio* 2009, Liu et al *JOBS* 2014

Boolean Networks + Extensions (Kaufman 69)

Temporal Logic and Model Checking of Boolean Networks, Synchronous and Asynchronous

Finding Fixed Points

Computing Attractors and Basins of Attraction

Stability Analysis (Modular Proof Techniques)

Identifying new Interactions

Weinstein and Mendoza 2013 *Front in Genetics,* Weinstein et al. *BMC Bioinformatics,* Cook et al. *VMCAI* 2005

Dynamic Bayesian Networks

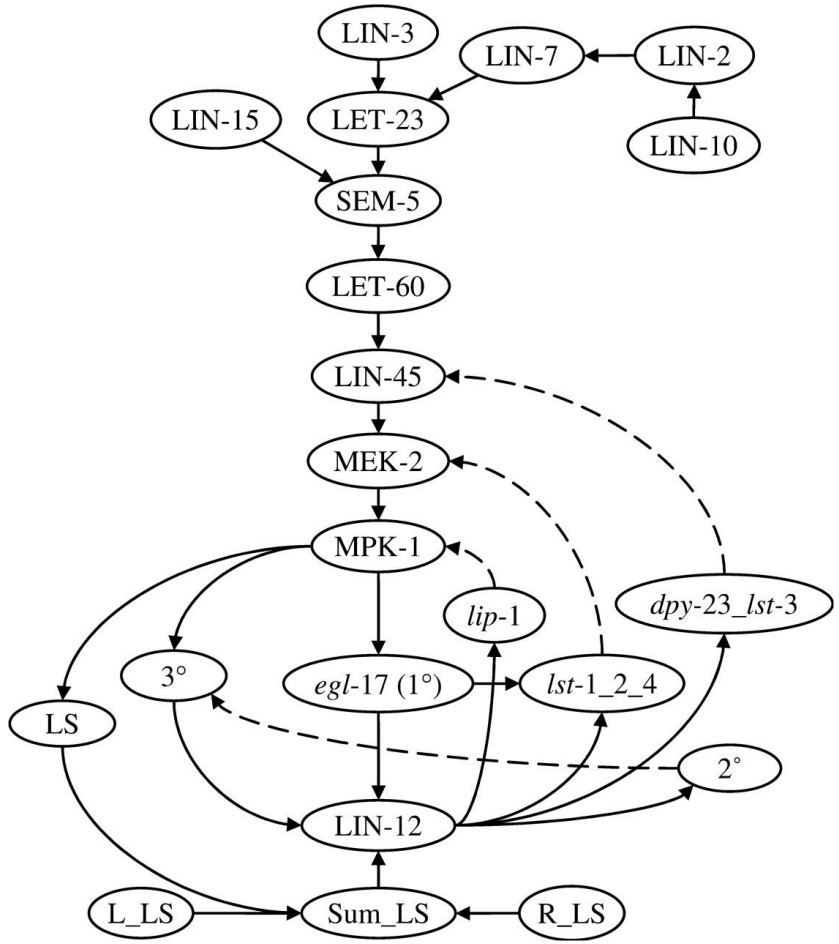
Learns network models from examples and assumptions on influence between components

Can learn different networks with confidence scores

Learning approaches are dominant in Gene Network Inferences

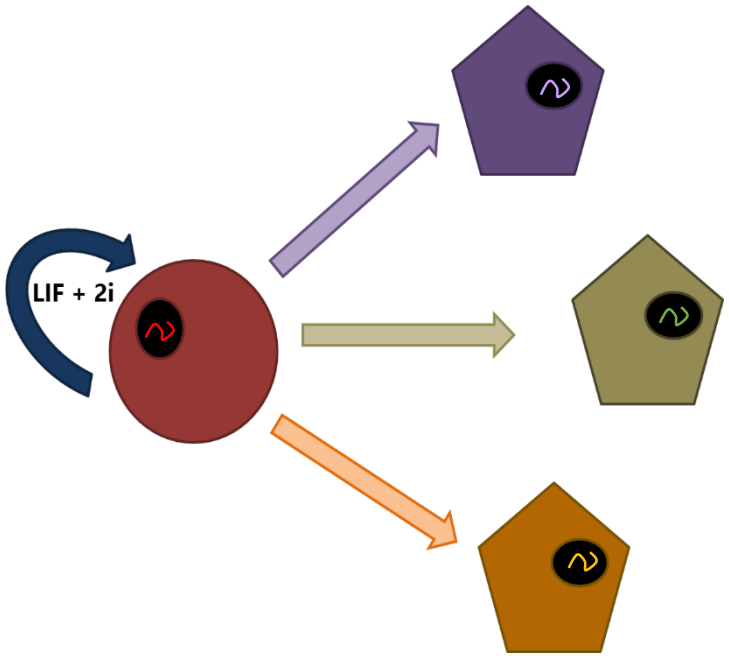
Pros - Deal with noise and stochastic behavior
Scalability

Cons - Limited in identifying inconsistencies
Not always mechanistic and hard to explain

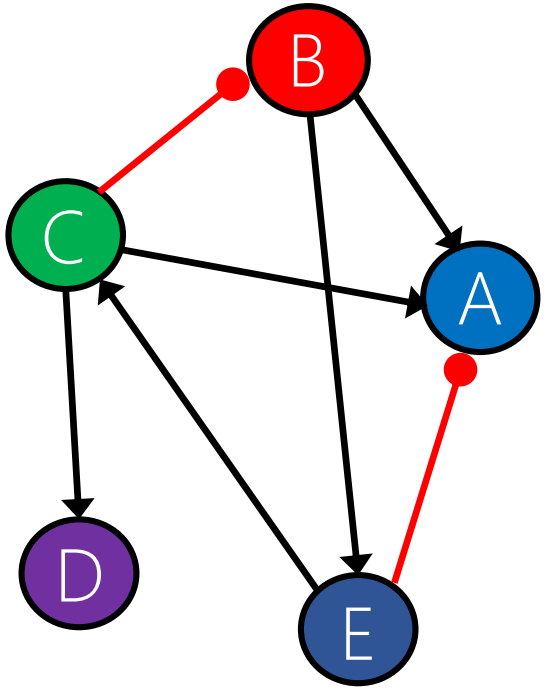


Sun and Hung 2007 *Bioinformatics*

Modeling Gene Regulatory Networks (GRNs)



Every cell's identity and function is defined by the different genes that it "expresses".



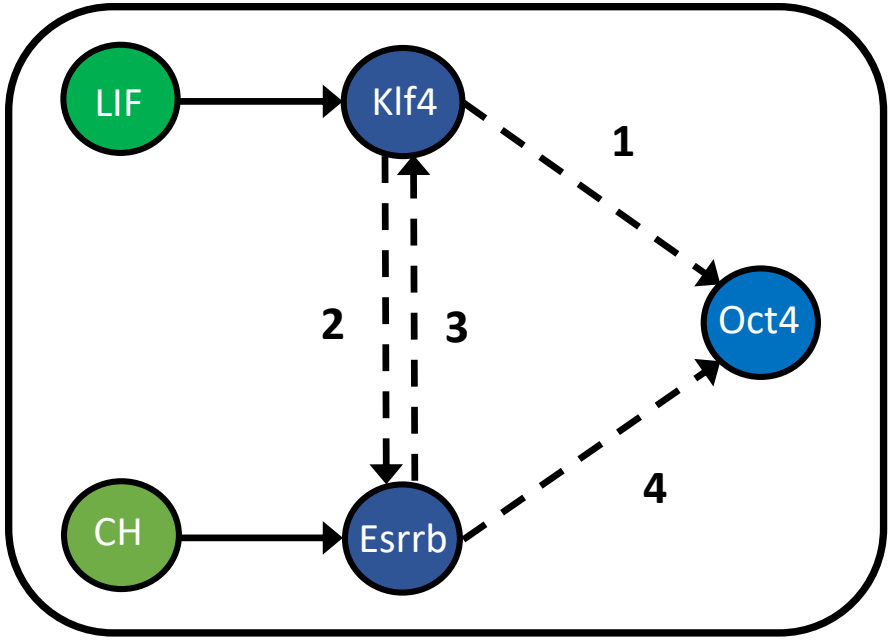
Genes can activate and inhibit each other's expression. Gene regulatory networks thus determine which genes are switched on, and which are switched off.

Computational Models can represent dynamics of GRN

- Mechanistic Models based on experimental data

- Allows to simulate new experiments in-slico
  - Starting from new conditions
  - Knockouts or Over Expressions

Example: A simple network of 5 genes



Inputs

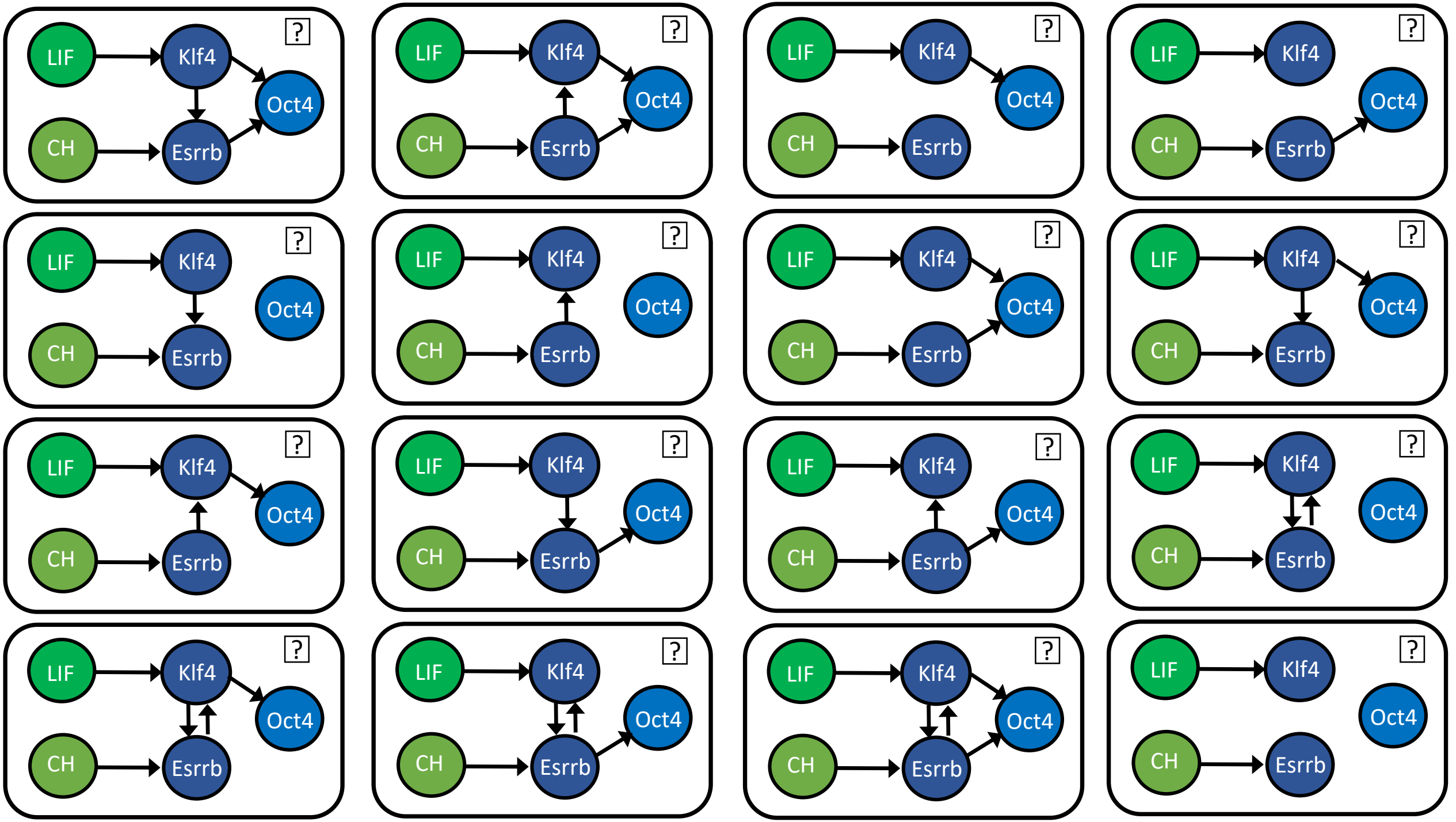
Gene	Experiment 1	Experiment 2
LIF	ON	OFF
CH	ON	ON
Klf4	ON	OFF
Esrrb	ON	ON
Oct4	ON	ON

Which of the optional interactions (1,2,3,4) are necessary to meet these two experimental conditions?

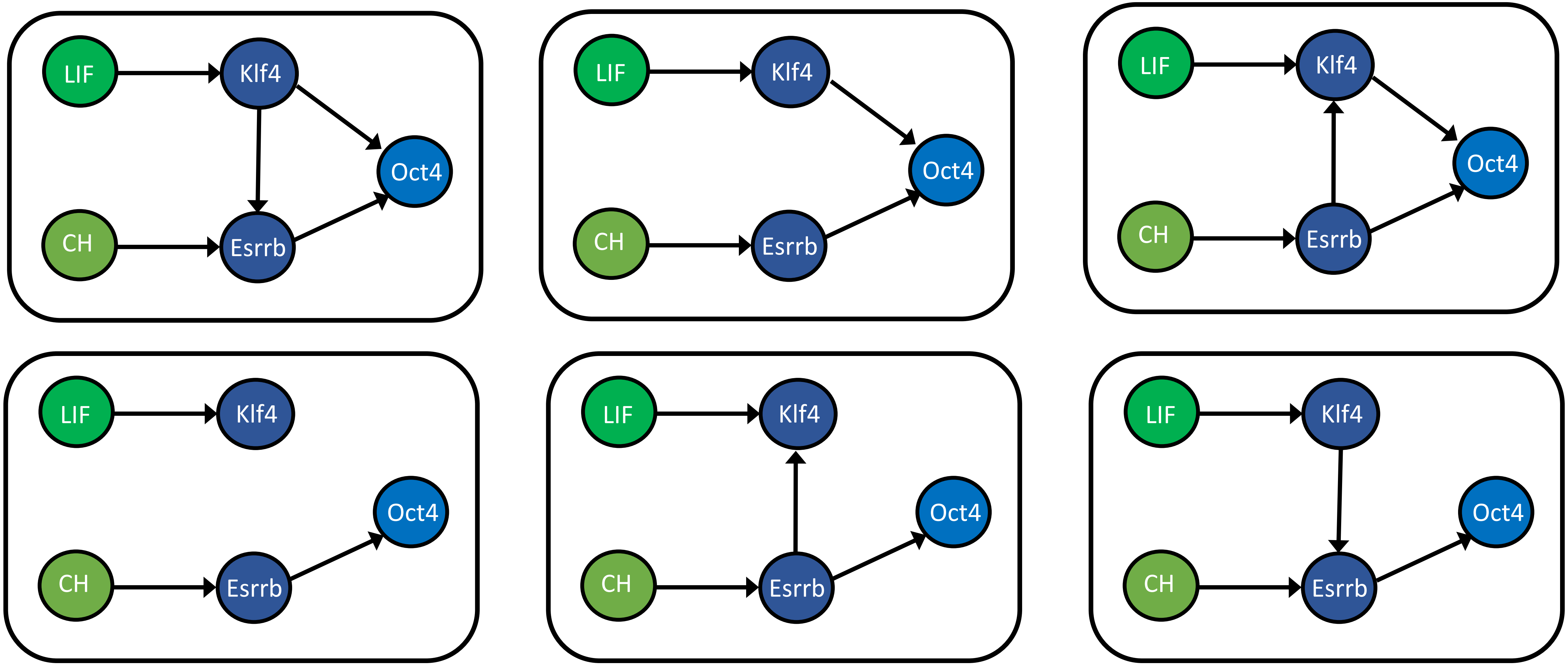
Yordanov et al., *Nature Sys Bio and App,* 2016

There are 16 possible networks, but not all of these will satisfy the experimental observations.

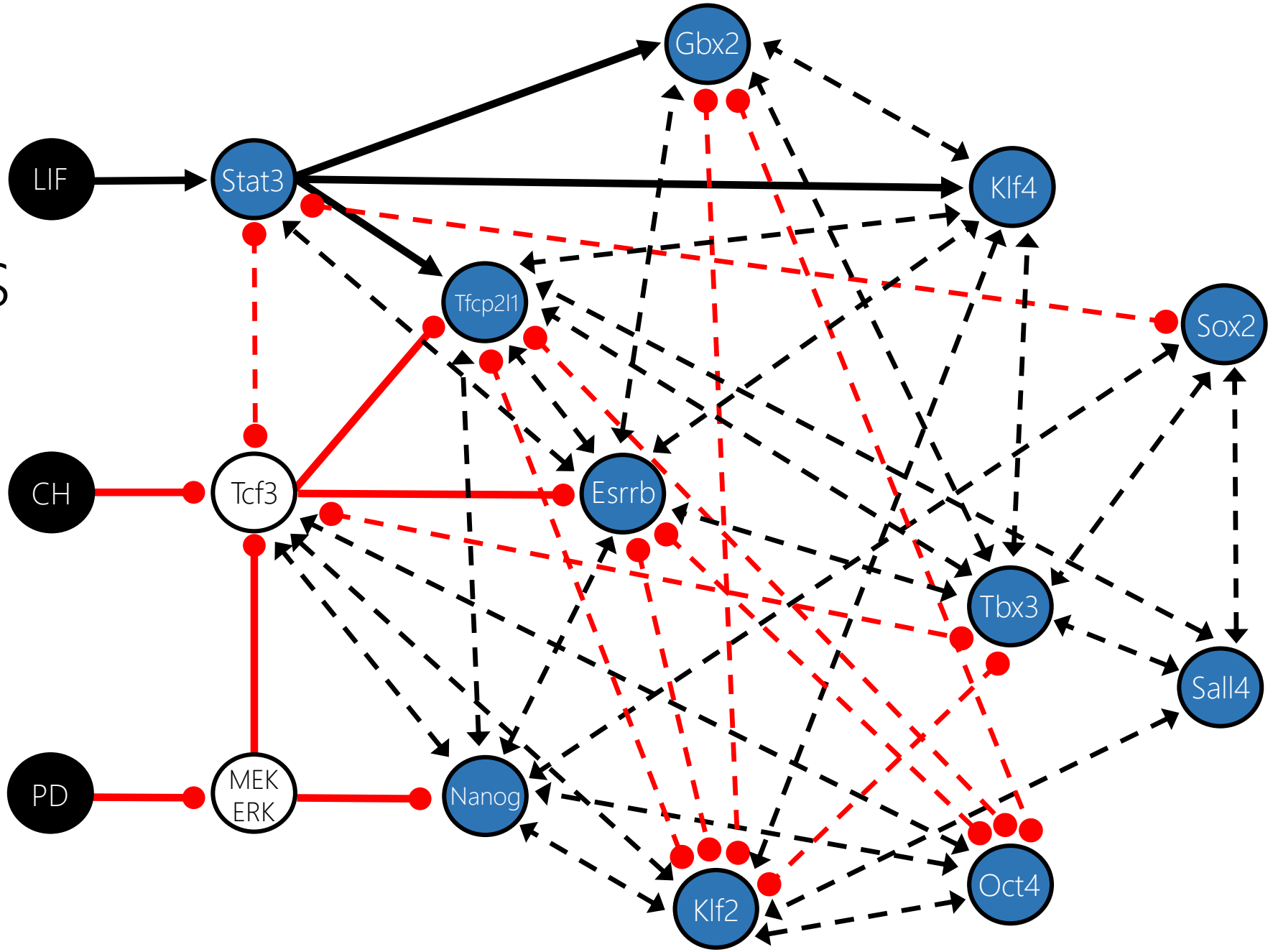
Do we have to check all of them?


?
LIF
CH
Klf4
Esrrb
Oct4

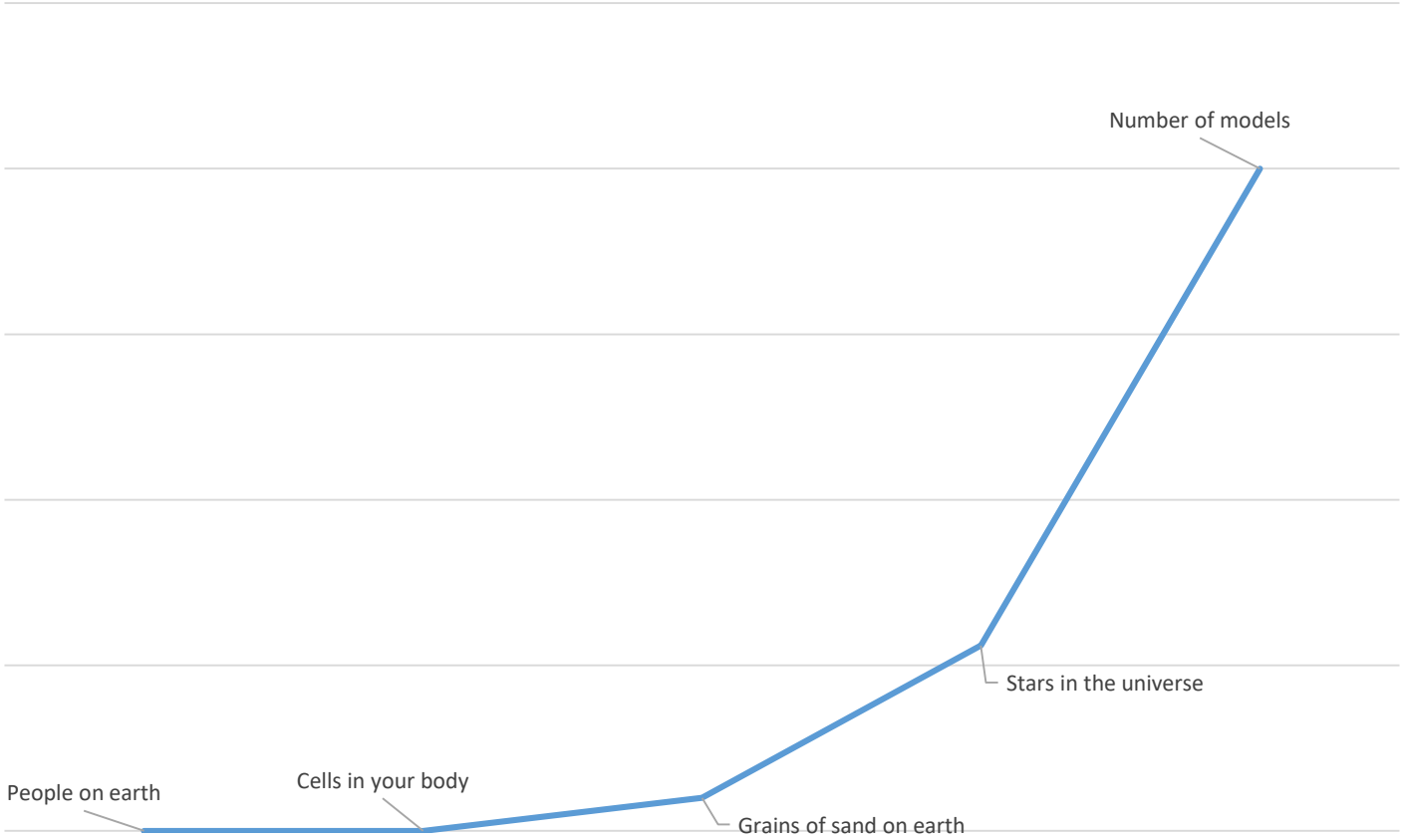
6 of the networks can explain the experimental data



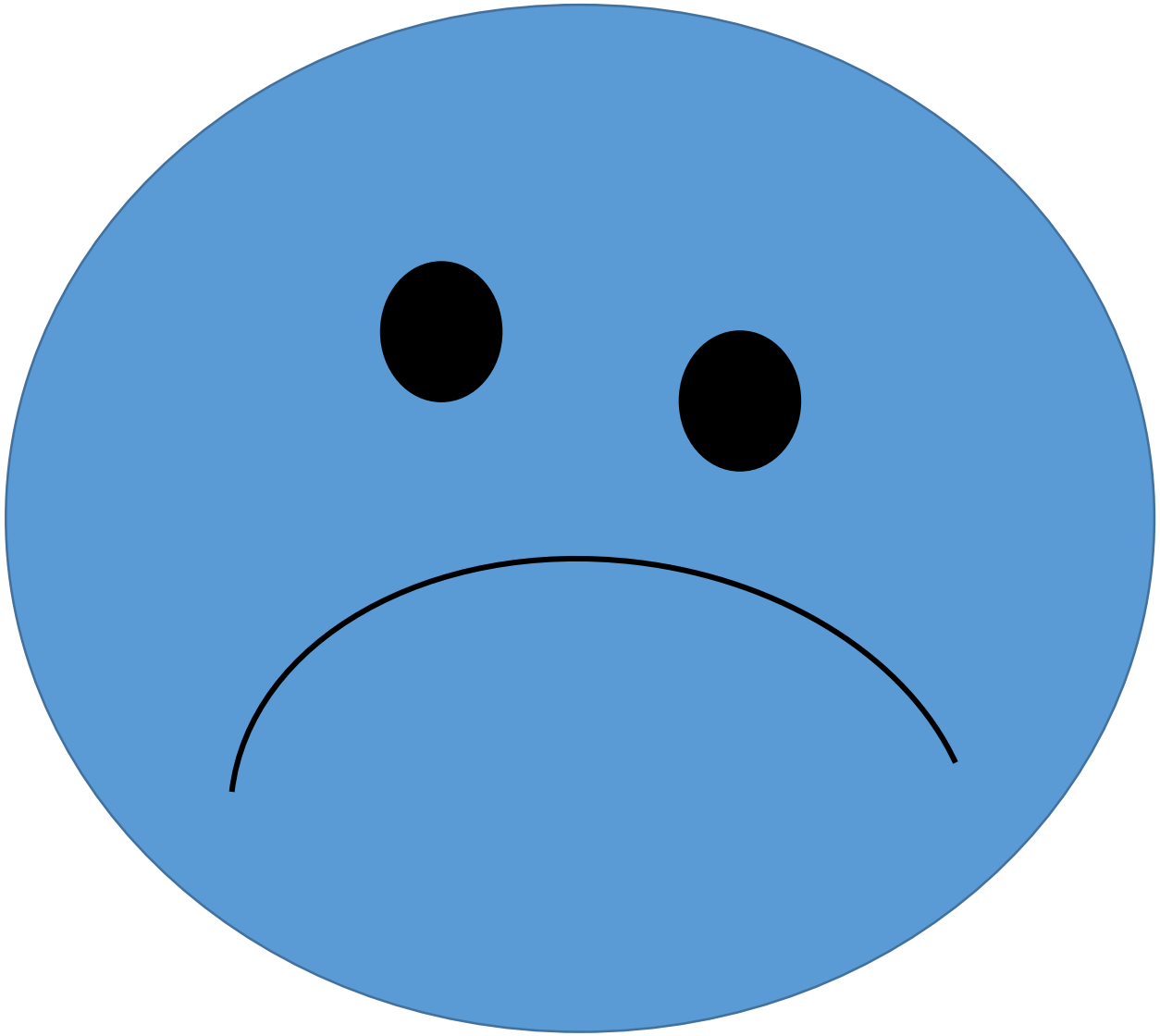
What happens when things get a little more complicated?

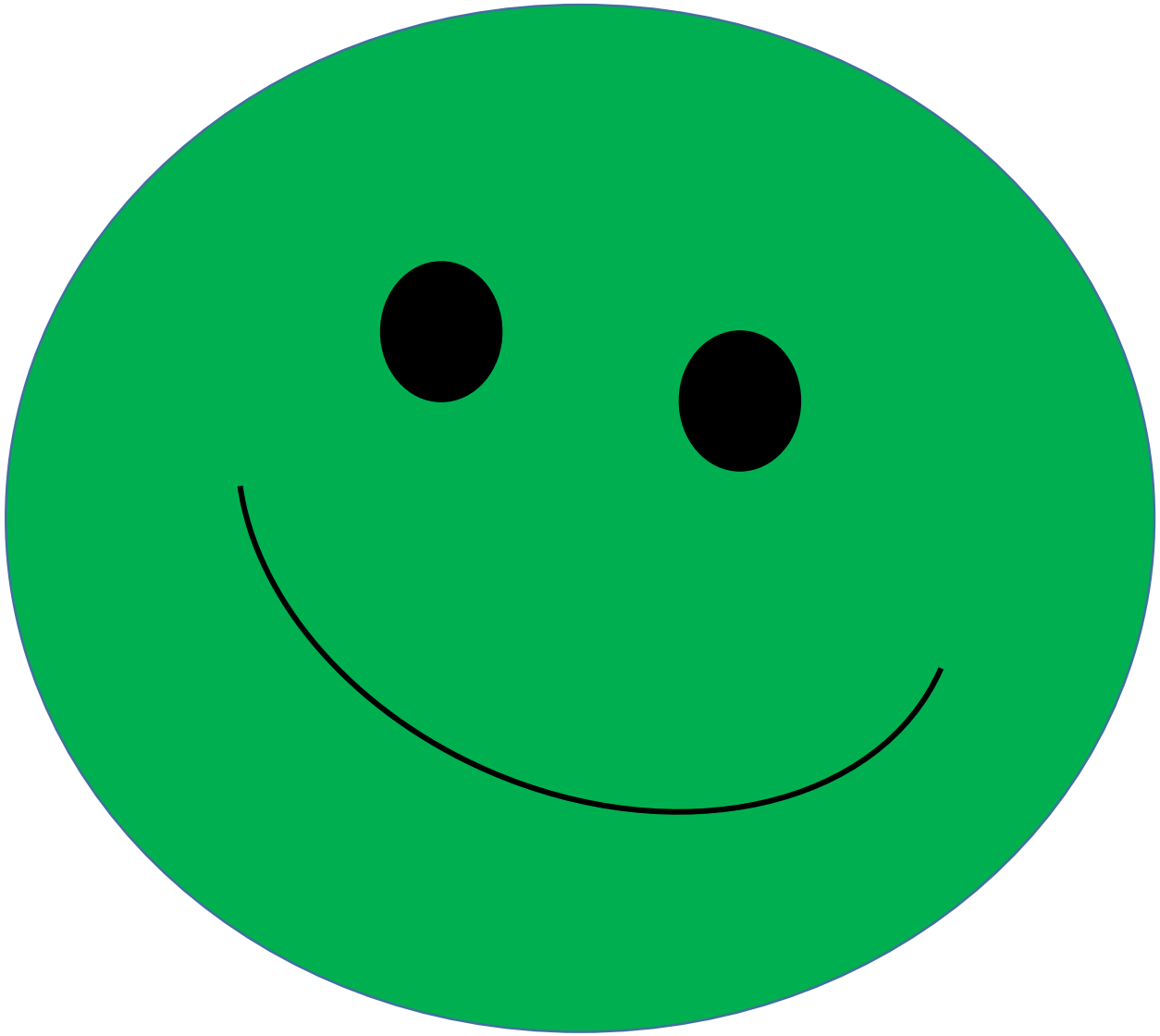


200,000,000,000,000,000,000,000,000

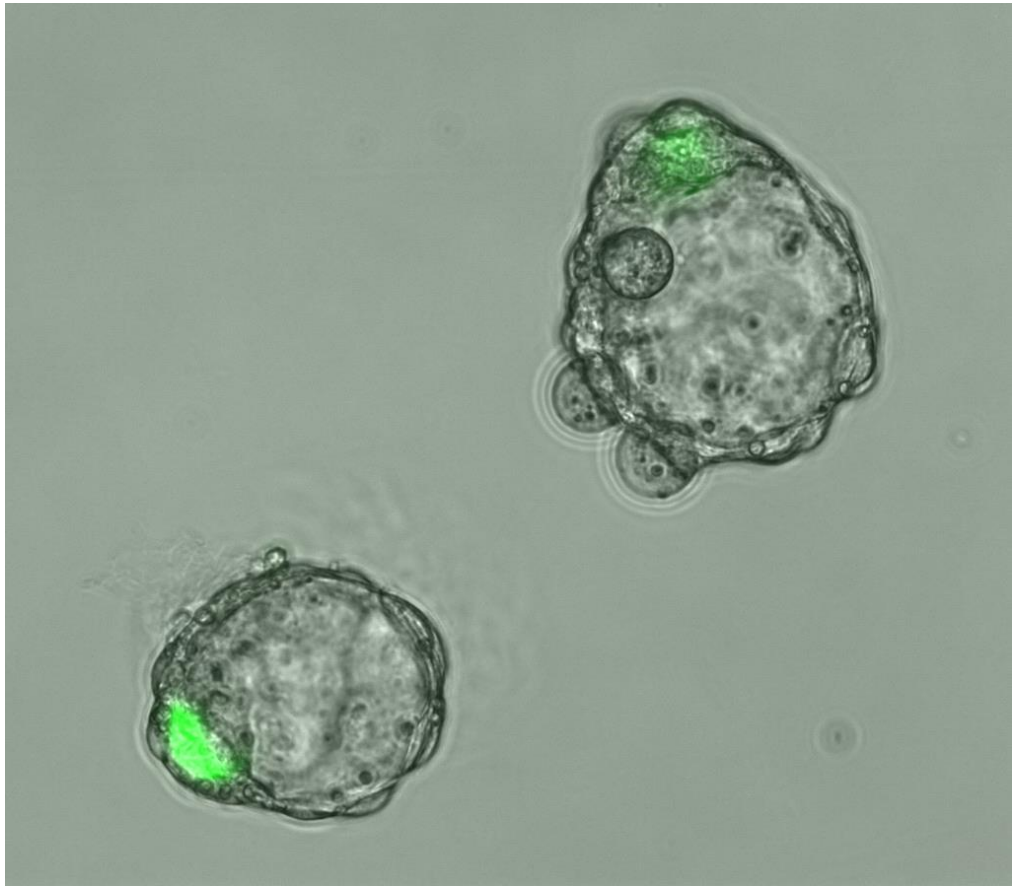


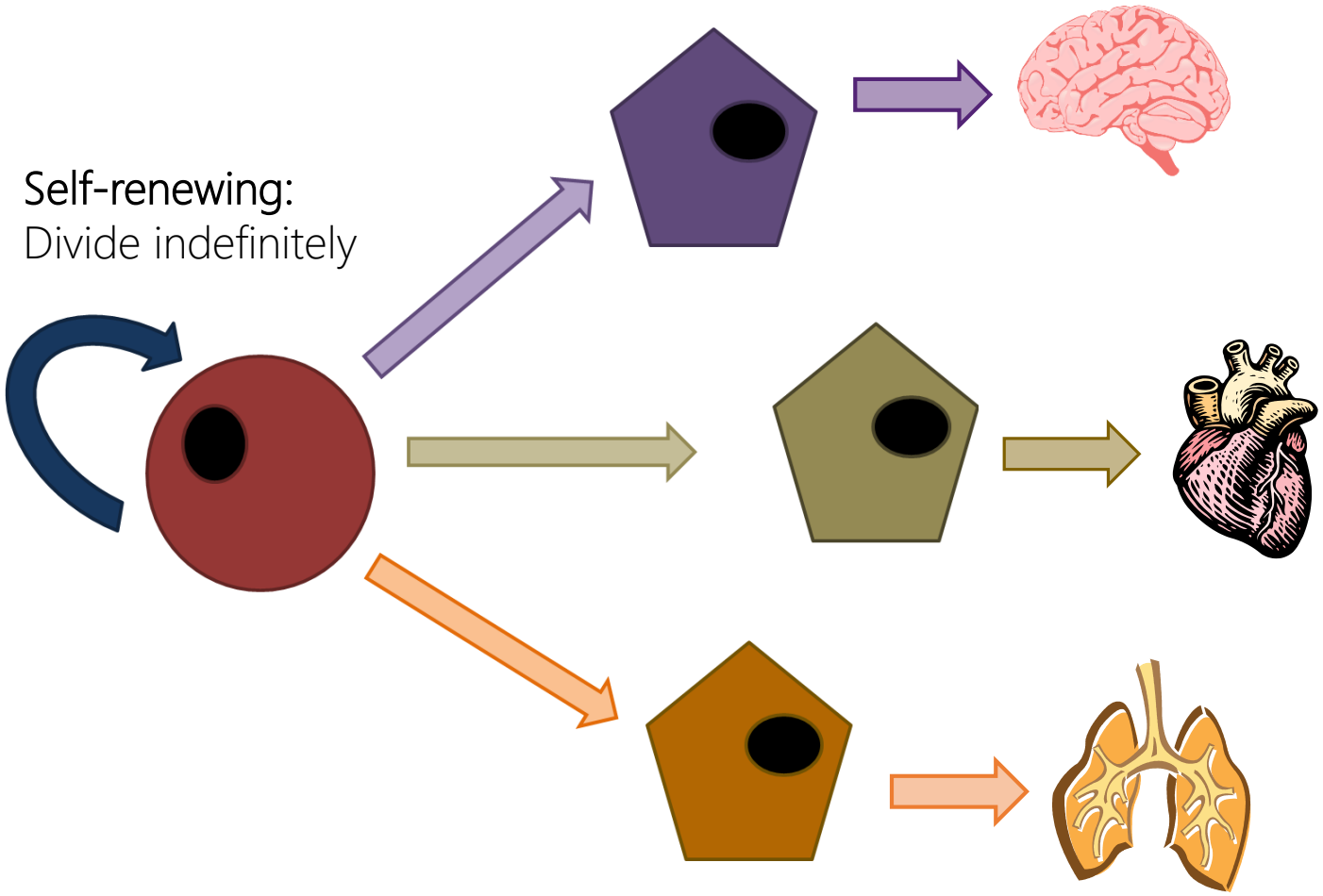
Do we have to check all of them?!





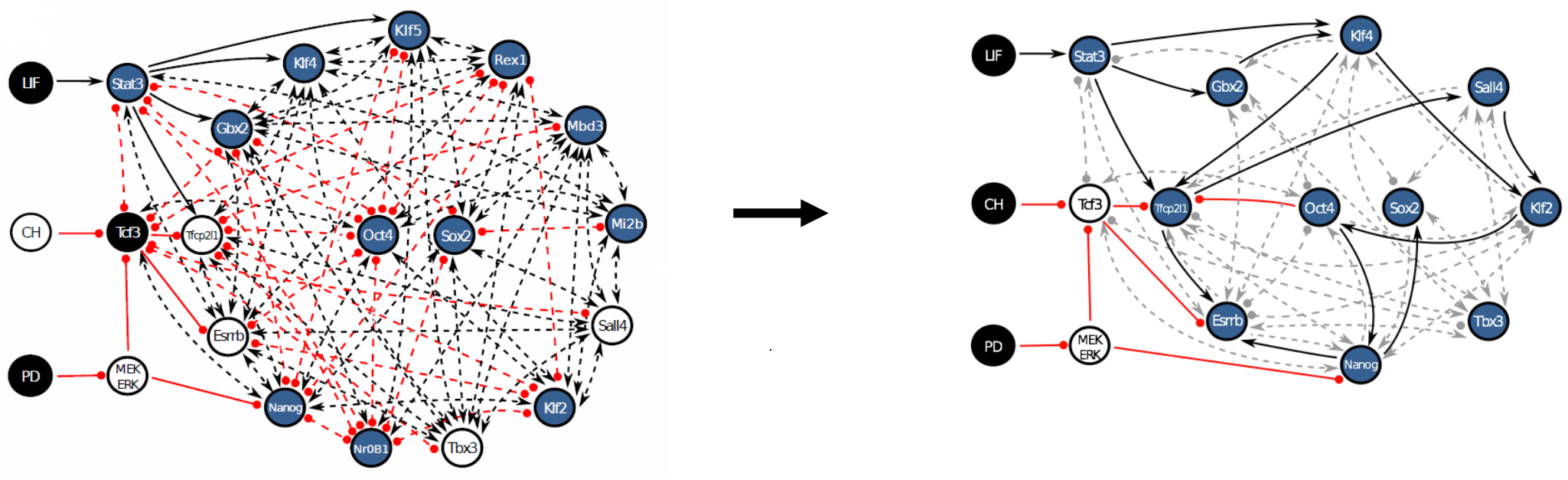
What are Embryonic Stem Cells?

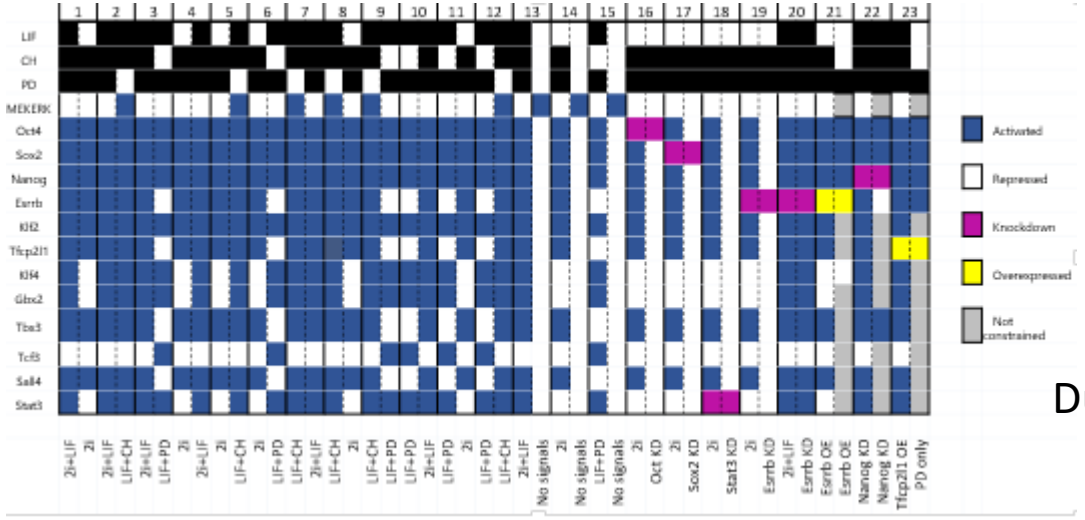




Pluripotent: Generate all adult cell types

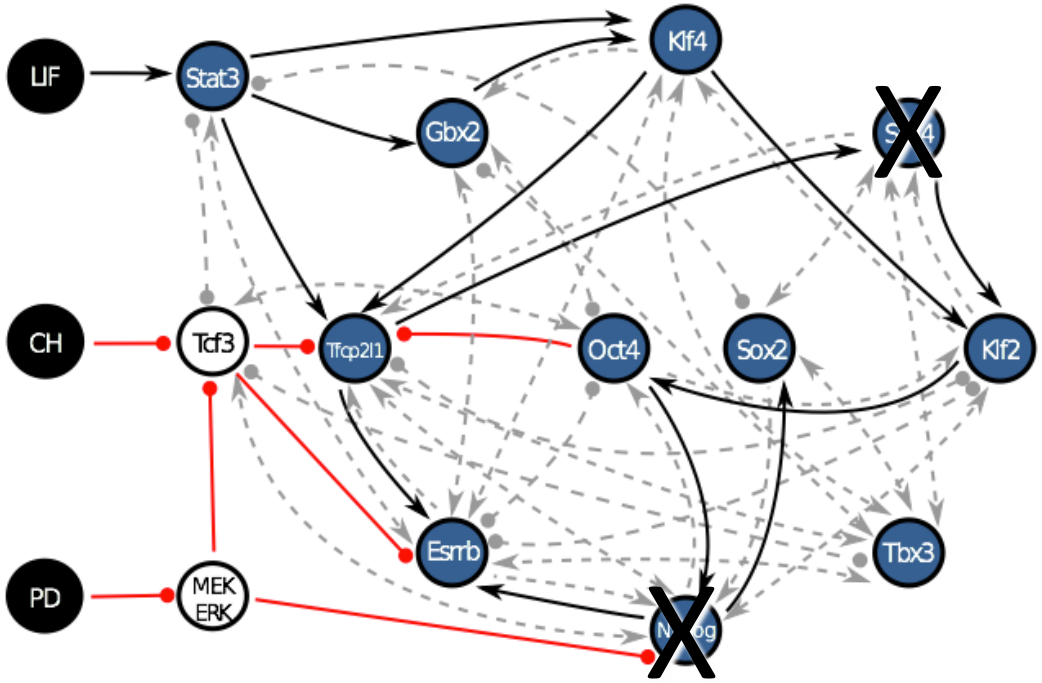
Constraining The Set of Possible Models





Dunn et al., *Science* 2014

Predictions of ES Cell Behaviour



Self-renewal?
Yes / no

Abstract Boolean Network (ABN)

Tuple $N = (G,\ E,\ E^{?},\ R)$

G - Finite set of genes

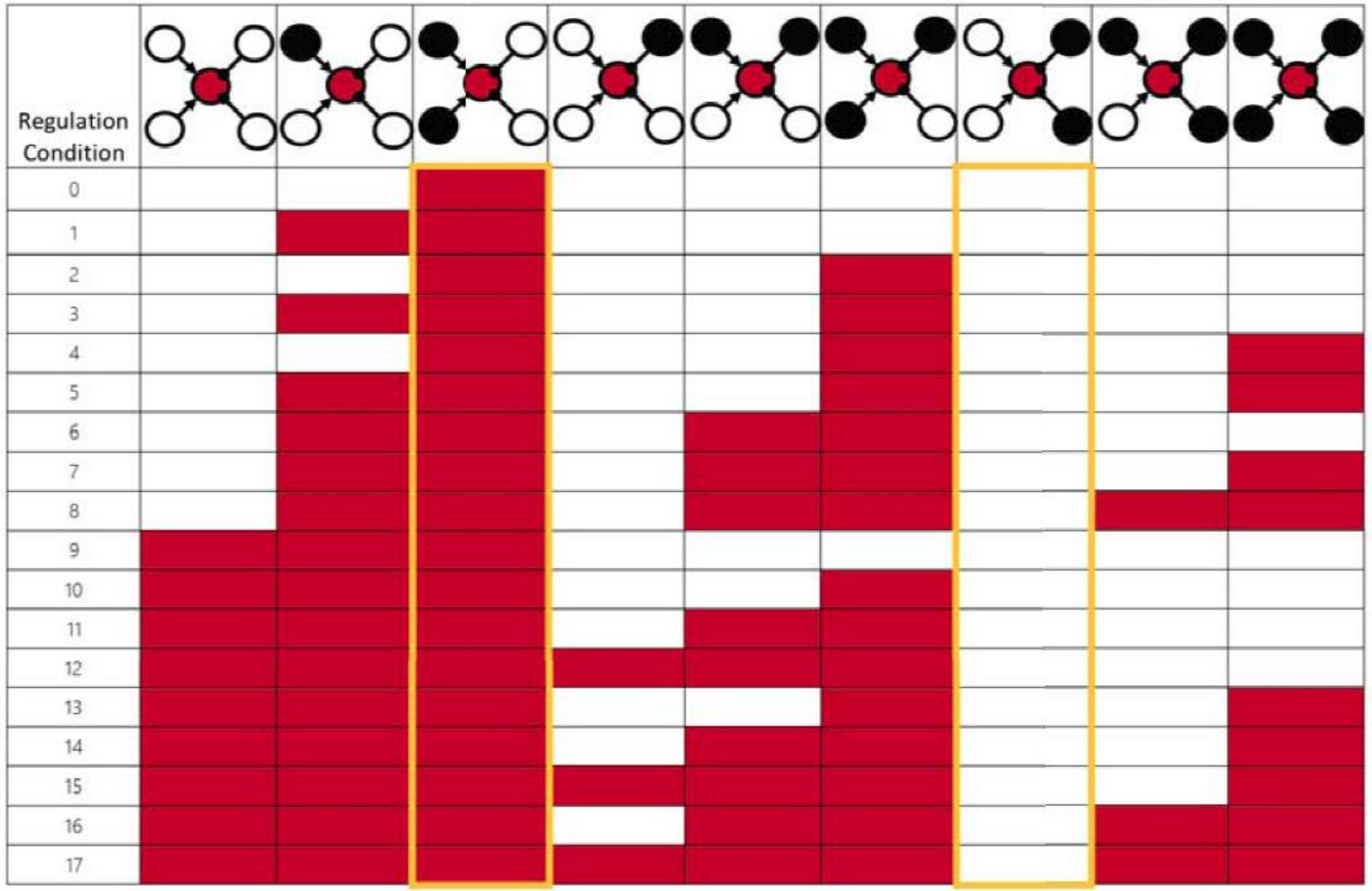
E - Definite interactions (positive or negative)

$E^{?}$ - Optional interactions (positive or negative)

R – Set of Regulation Conditions

Regulation Conditions

- Defines a set of logical function given positive and negative interactions
- Takes into account if none / some /all activators are present
- Restrict to monotonic functions
- Aims to capture biologically plausible regulation functions
- Recent unpublished work extends regulation conditions



Synthesize Concrete Boolean Network

Tuple $N = (G,\ E,\ E^{?},\ R)$

G - Finite set of genes

E - Definite interactions (positive or negative)

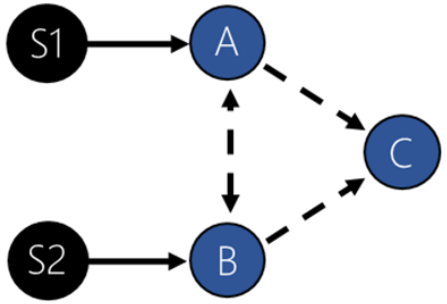
$E^{?}$ - Identify Optional interactions

R – Identify Regulation Conditions

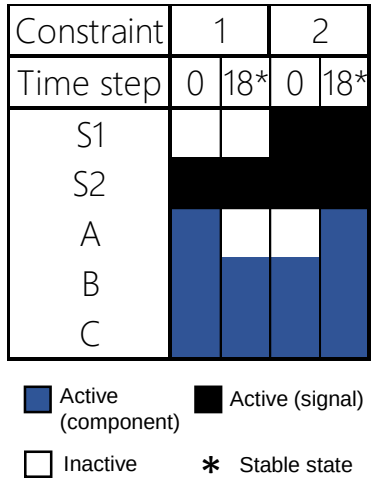
cABN – Biological Program

```
// Settings
directive regulation noThresholds;
directive updates sync;

// Components
S1(0); S2(0);                  // Signals
A(0..8); B(0..8); C(1,3,5);    // TFs

// Definite interactions
S1      S1      positive;
S2      S2      positive;
S1      A       positive;
S2      B       positive;

// Possible interactions
A       C       positive       optional;
A       B       positive       optional;
B       A       positive       optional;
B       C       positive       optional;

// Observation predicates
$Conditions1 := { S1 = 0 and S2 = 1};
$Conditions2 := { S1 = 1 and S2 = 1};
$Expression1 := {A = 1 and B = 1 and C = 1};
$Expression2 := {A = 0 and B = 1 and C = 1};

// Observations
#Experiment1[0] |= $Conditions1 and
#Experiment1[0] |= $Expression1 and
#Experiment1[18] |= $Expression2 and
fixpoint(#Experiment1[18]);

#Experiment2[0] |= $Conditions2 and
#Experiment2[0] |= $Expression2 and
#Experiment2[18] |= $Expression1 and
fixpoint(#Experiment2[18]);
```

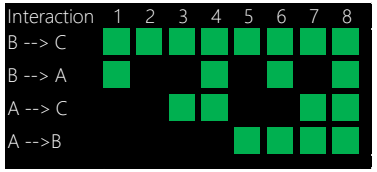
Structure



Behaviour



Synthesis Algorithm : Find Solutions that satisfy all constraints if possible (Z3-4Bio Framework)



Inconsistent : no concrete programs exist

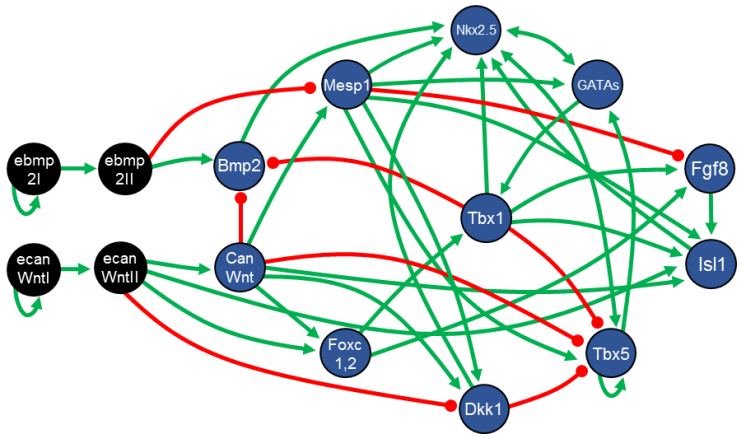
RE:IN Tool - A Method to Identify and Analyze Gene Regulatory Networks through Automated Reasoning

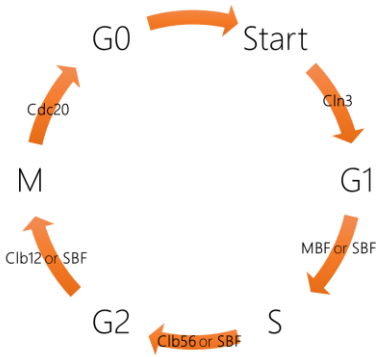
	Cardiac (FHF)			
	Initial	Transit1	Transit2	FHF
Time step	0	1	2	3*
ecanWnt				
eBmp2				
Bmp2				
canWnt				
Dkk1				
Fgf8				
FoxC1/2				
GATAs				
Isl1				
Mesp1/2				
Nkx2.5				
Tbx1				
Tbx5				

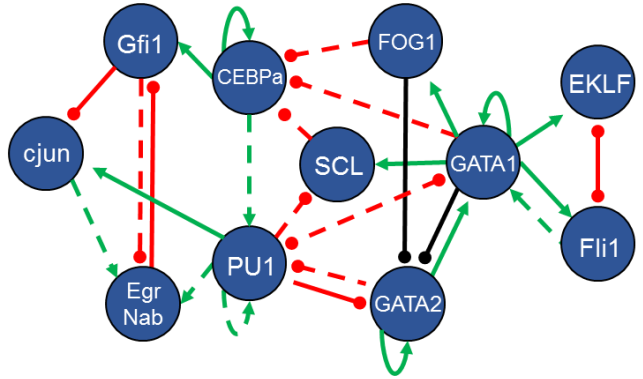
	Cardiac (SHF)			
	Initial	Transit1	Transit2	SHF
Time step	0	1	2	3*
ecanWnt				
eBmp2				
Bmp2				
canWnt				
Dkk1				
Fgf8				
FoxC1/2				
GATAs				
Isl1				
Mesp1/2				
Nkx2.5				
Tbx1				
Tbx5				

	Cell Cycle Phases							
	G0	Start	G1	S	G2	Early M	Late M	G0
Time step	0	1	2	..	..	..	..	13
Cell Size								
Cln3								
MBF								
SBF								
Cln1,2								
Cdh1								
Swi5								
Cdc20								
Clb5,6								
Sic1								
Clb1,2								
Mcm1								

	Interaction	1	2	3
*	CEBPa-->CEBPa			
*	CEBPa-->Gfi1			
	CEBPa-->PU1			
*	EKLF--|Fli1			
*	EgrNab--|Gfi1			
	FOG1--|CEBPa			
*	Fli1--|EKLF			
*	GATA1-->EKLF			
*	GATA1-->FOG1			
*	GATA1-->Fli1			
*	GATA1-->GATA1			
*	GATA1-->SCL			
*	GATA2-->GATA1			
*	GATA2-->GATA2			
*	Gfi1--|cjun			
*	PU1--|GATA2			
*	PU1-->cjun			
	cjun-->EgrNab			
	SCL--|CEBPa			
	GATA1--|CEBPa			



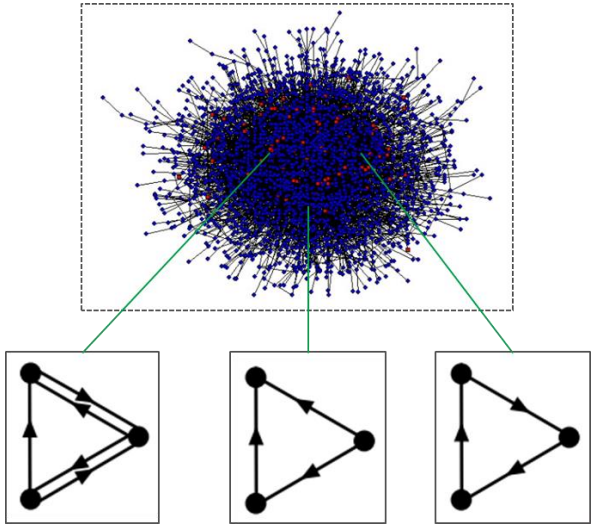




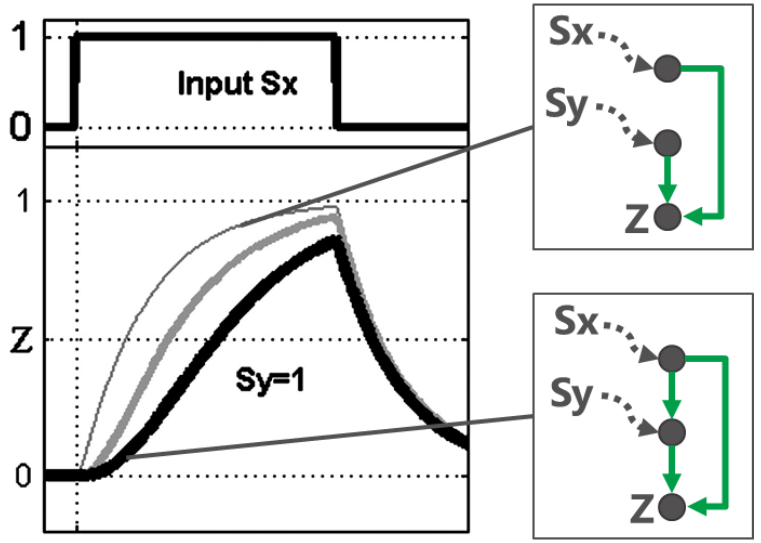
Yordanov et al., *Nature Sys Bio and App,* 2016

Network Motifs

Motif finding



Motif dynamics

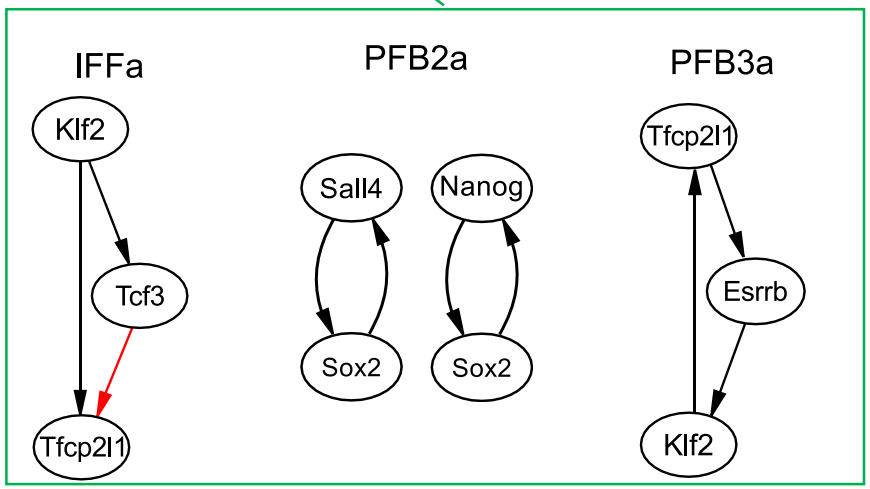


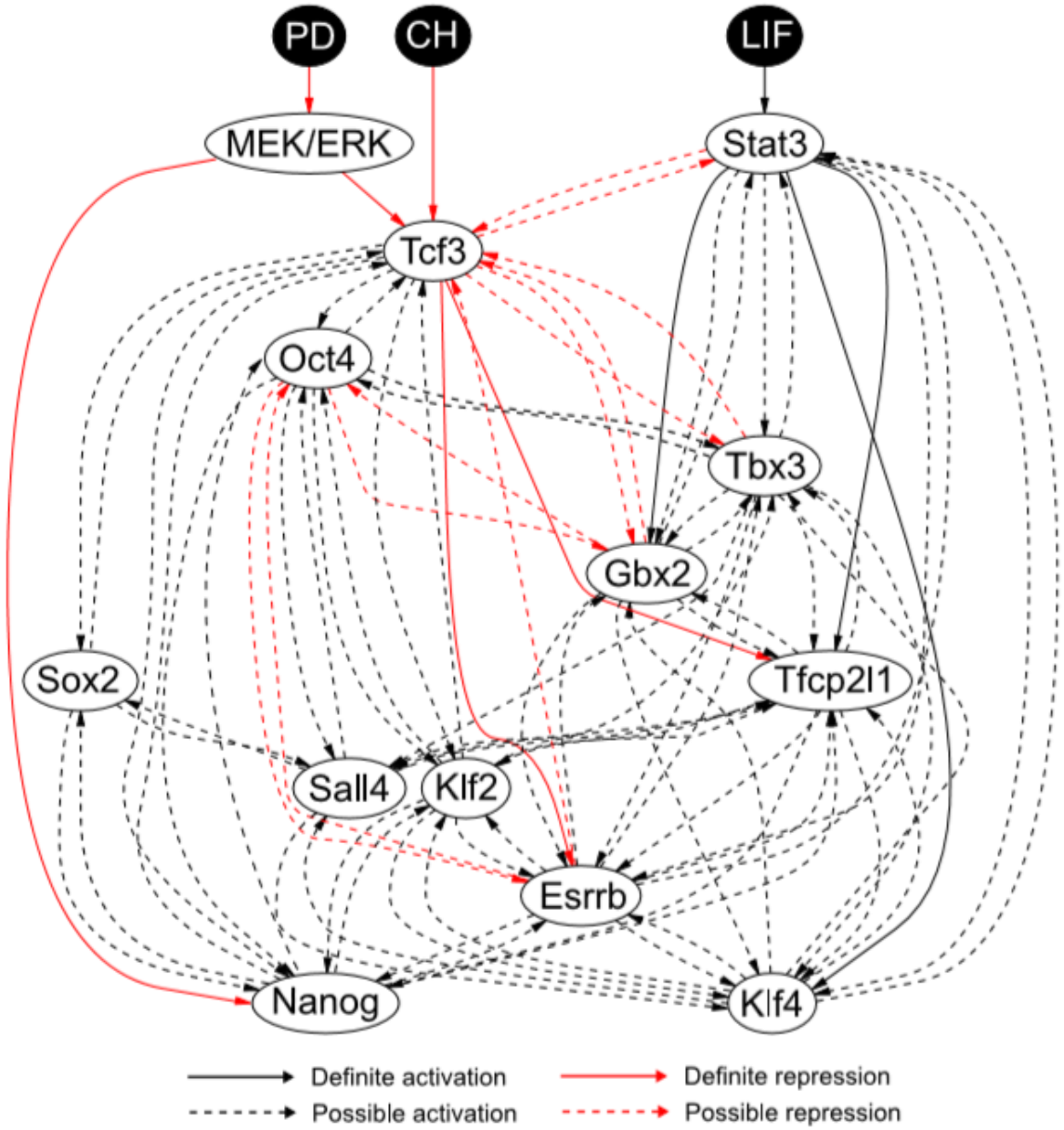
+ Scalable motif finding algorithms
- Often, static networks are considered
- Networks are rarely precisely known

+ Detailed quantitative predictions
- Motifs are studied in isolation
- Parameters are often not known

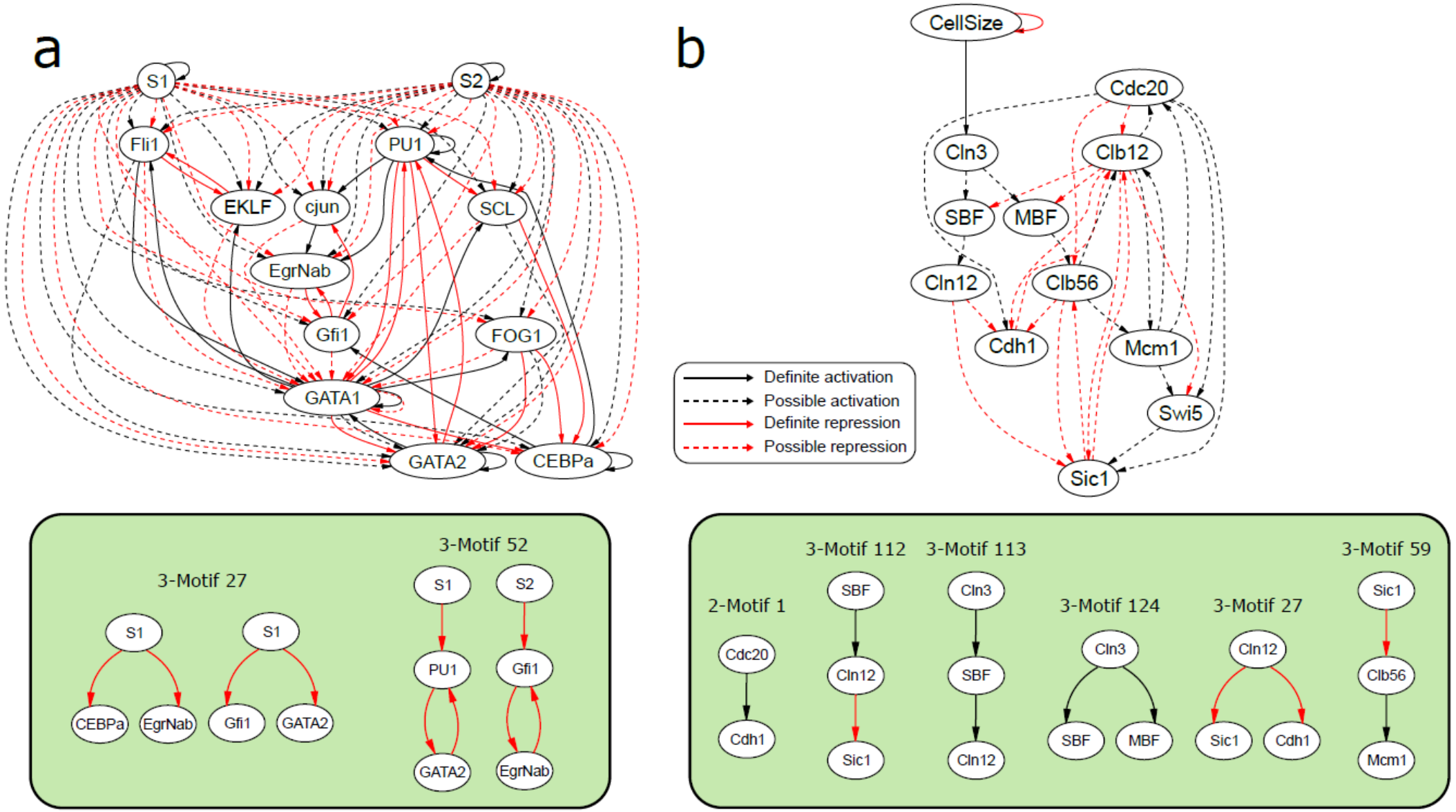
Stem Cell Motifs

Motif	$\mathcal{M}$	$\neg\mathcal{M}$	$t_{\mathcal{M}}$ $(t'_{\mathcal{M}})$	$t_{\neg\mathcal{M}}$ $(t'_{\neg\mathcal{M}})$
CPFFa	true	true	630.62(19.19)	460.10(18.65)
CPFFb	true	true	277.47(18.40)	586.21(18.17)
IFFd	**false**	true	26.27(19.23)	636.10(19.16)
CNFFa	true	true	489.93(18.43)	611.28(18.90)
CNFFb	true	true	537.20(18.28)	493.21(17.90)
IFFa	true	**false**	609.35(19.95)	540.18(18.39)
IFFb	true	true	568.20(19.15)	555.86(17.77)
IFFc	**false**	true	515.28(20.74)	619.15(19.11)
PFB2a	true	**false**	467.51(18.01)	112.02(18.40)
PFB2b	true	true	505.46(18.21)	505.40(18.74)
NFB2	**false**	true	25.05(17.91)	640.31(17.72)
PFB3a	true	**false**	575.88(17.62)	525.42(19.04)
PFB3b	true	true	410.93(17.84)	531.24(17.73)
NFB3a	**false**	true	25.04(17.73)	572.96(17.65)
NFB3b	true	true	522.14(18.03)	531.86(17.96)

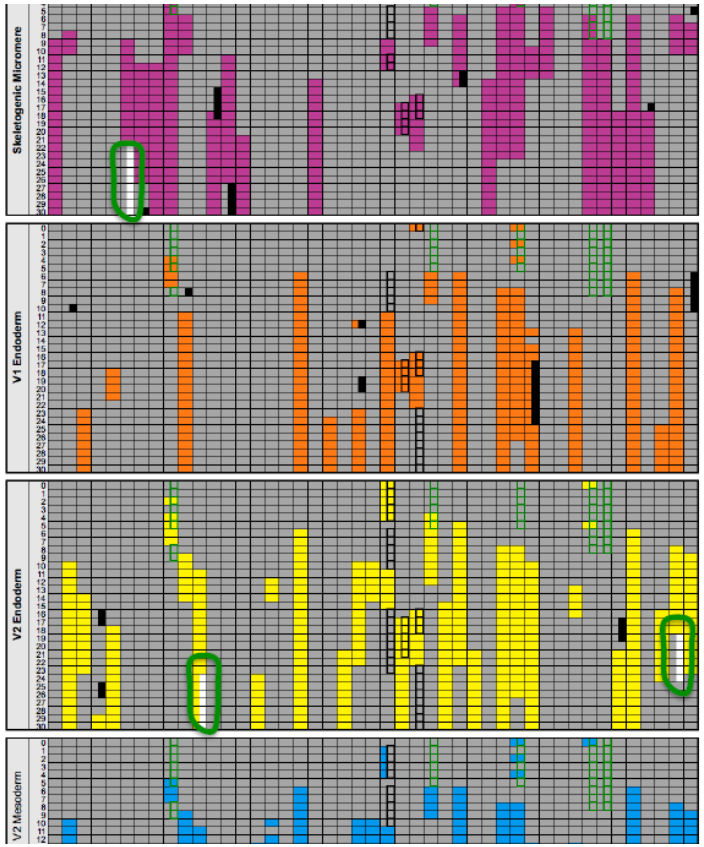




Systematic Motif Exploration

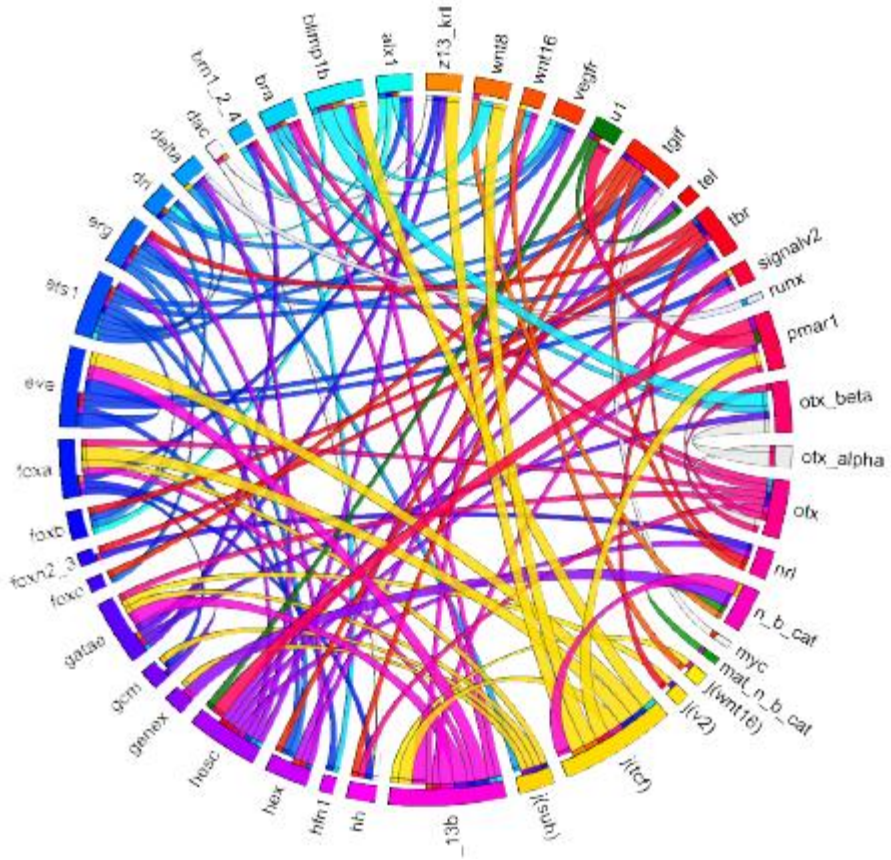


Temporal gene expression data + spatial domains

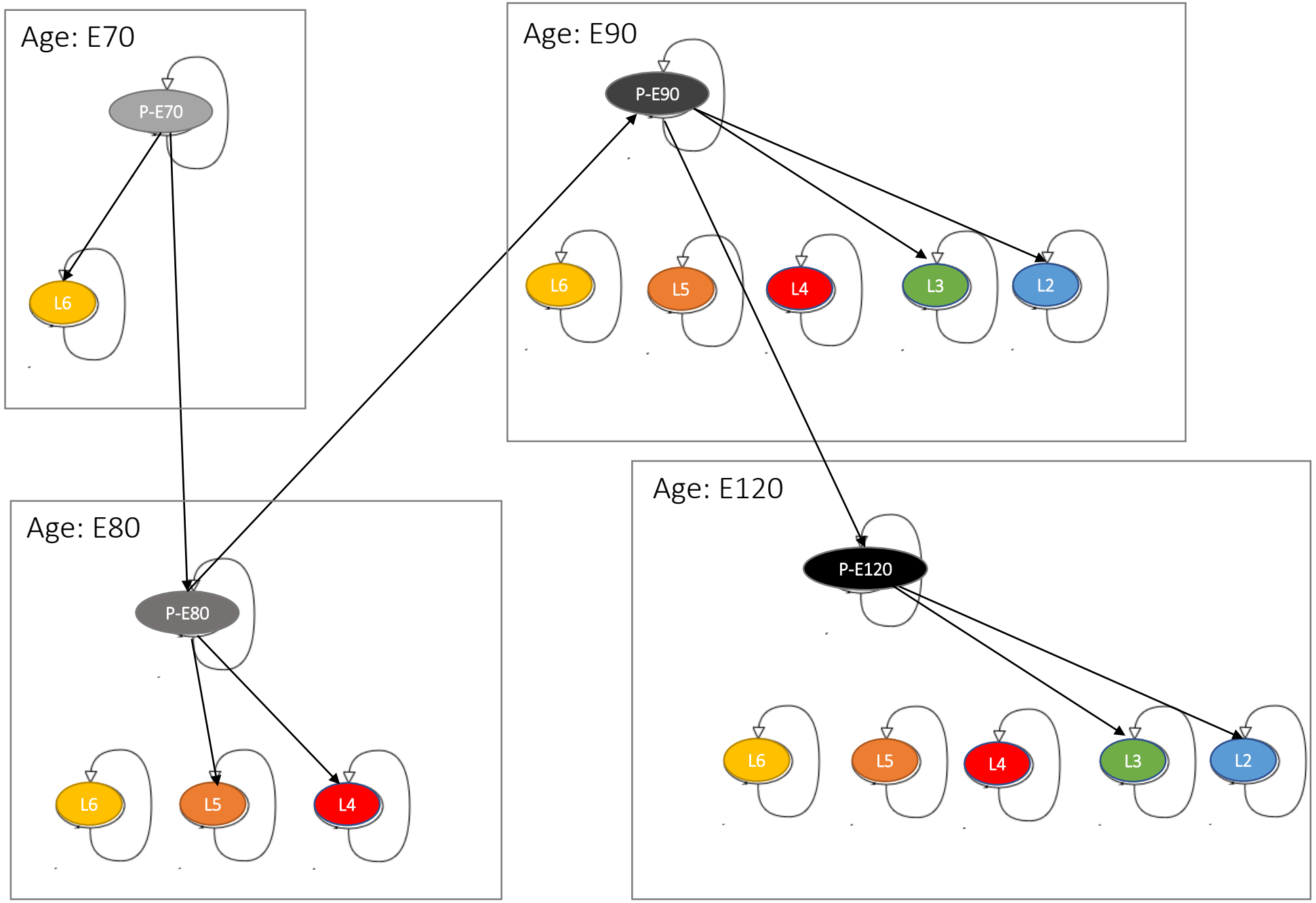


FULLY EXPLAINS EXPERIMENTAL DATA

- Solved discrepancies
- No need for hard-coded terms



Peter, Faure and Davidson. *PNAS*, 2012.
Paoletti, Yordanov, Wintersteiger, Hamadi, Kugler. *CAV* 2014



Neuron Specification in mammalian Cortex Shavit et al. (with Livesey Lab)

Engineered Biological Systems

Build new Computational Devices

- Fast
- Energy efficient

To better understand Biology

Interact with living systems

- Diagnostic
- Medicine

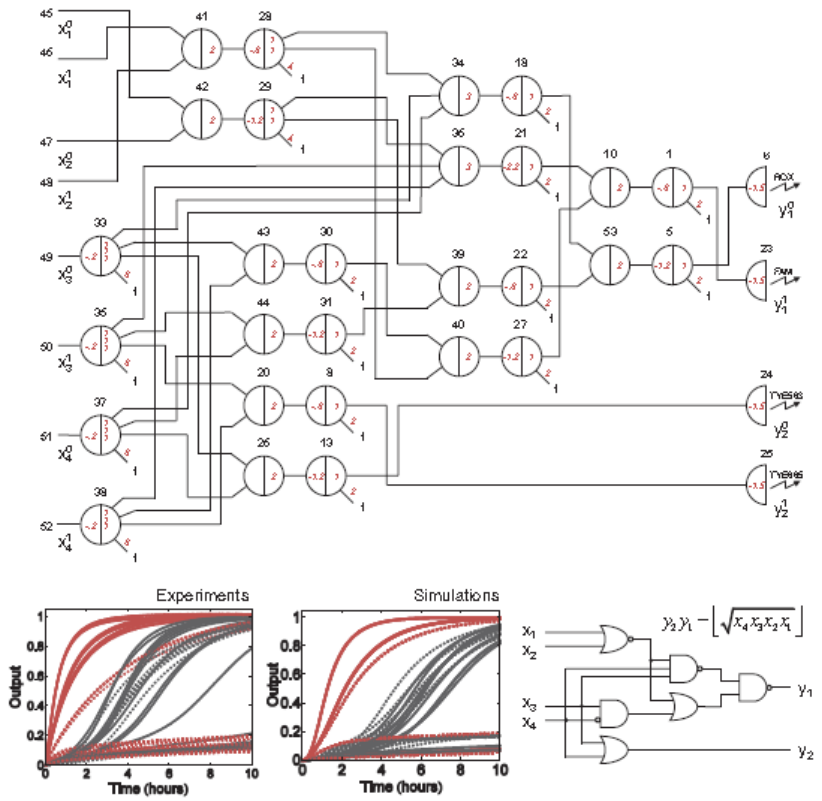
DNA Computing

Use biological material to design computational circuits (Adleman, 1994)

One promising paradigm is DNA Strand Displacement

Based on complementarity of DNA strands

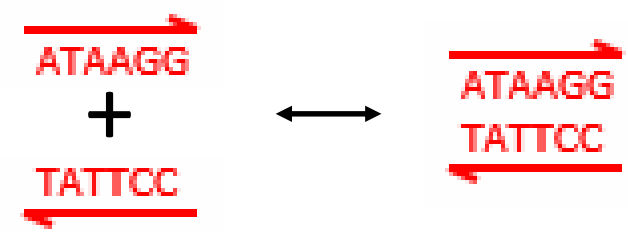
Programming Language and simulator translates to CRN representations



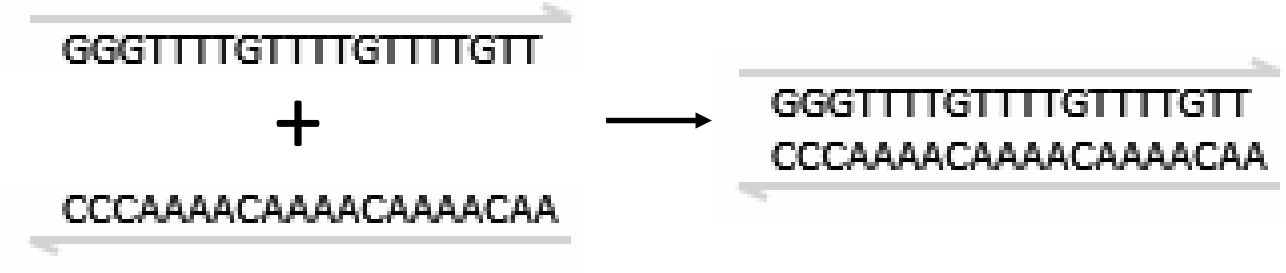
Qian and Winfree, *Science*, 2011; Qian, Winfree and Bruck, *Nature* 2011; Chen, Dalchau, Srinivas, Phillips, Cardelli, Soloveichik, Seelig. *Nature Nanotechnology, 2013*

Programmable DNA binding

- Short complementary domains bind *reversibly*

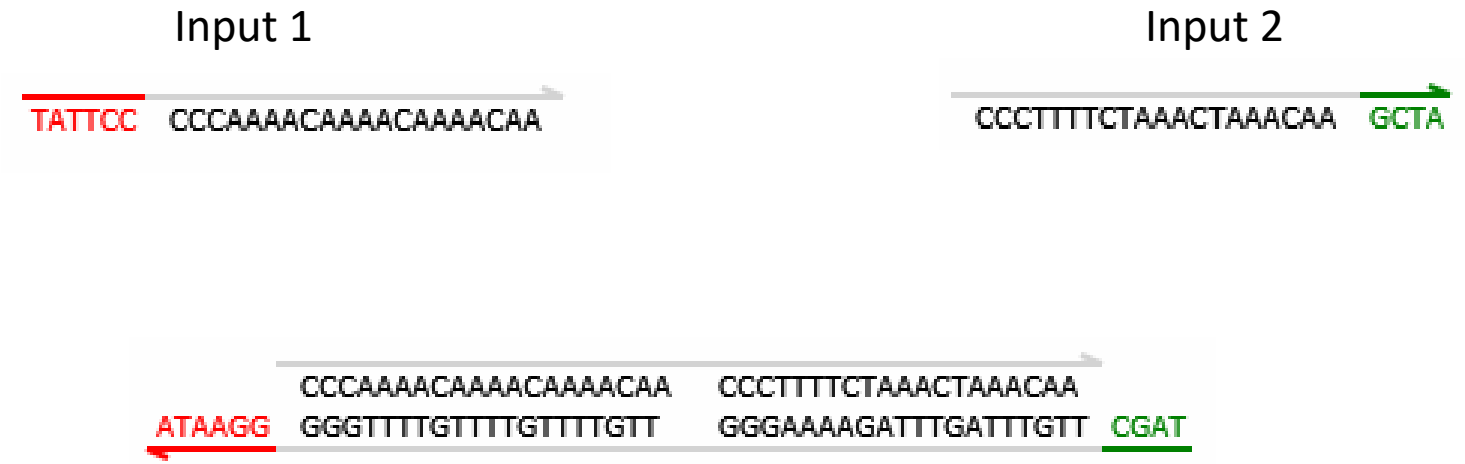


- Long complementary domains bind *irreversibly*



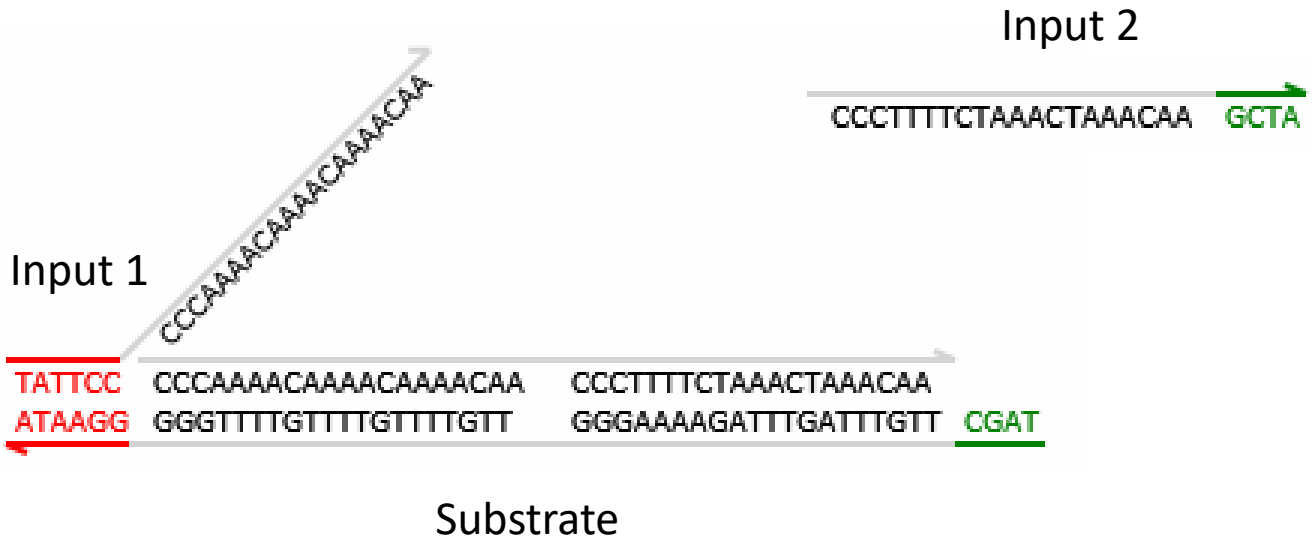
DNA Strand Displacement

DSD Logic Gate [Output = Input1 AND Input2]



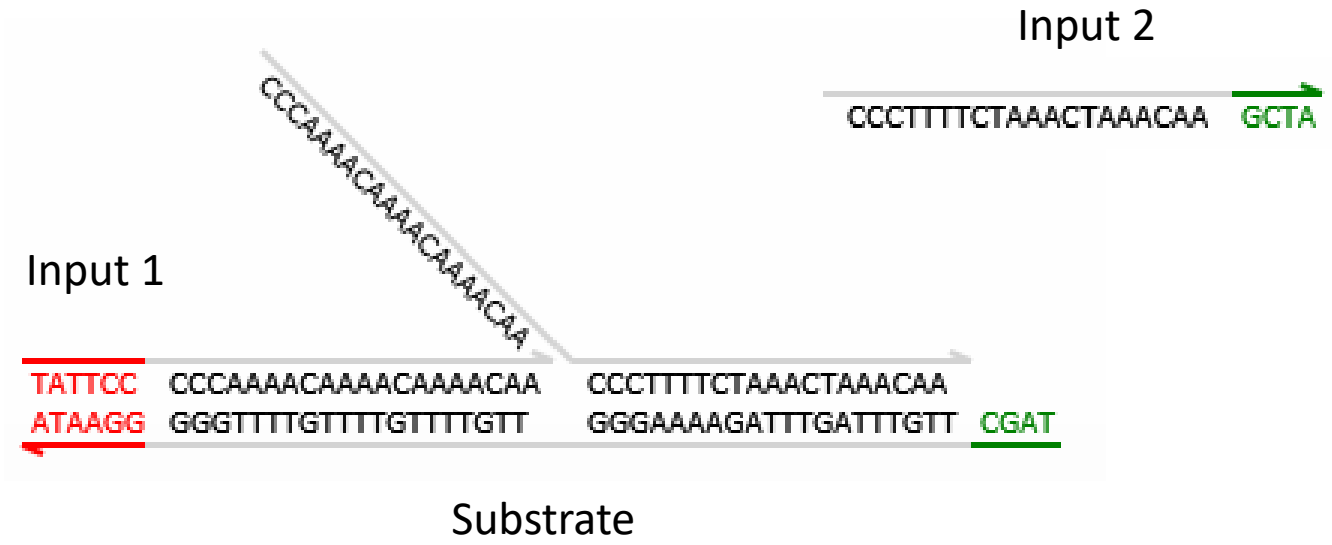
DNA Strand Displacement

DSD Logic Gate [Output = Input1 AND Input2]



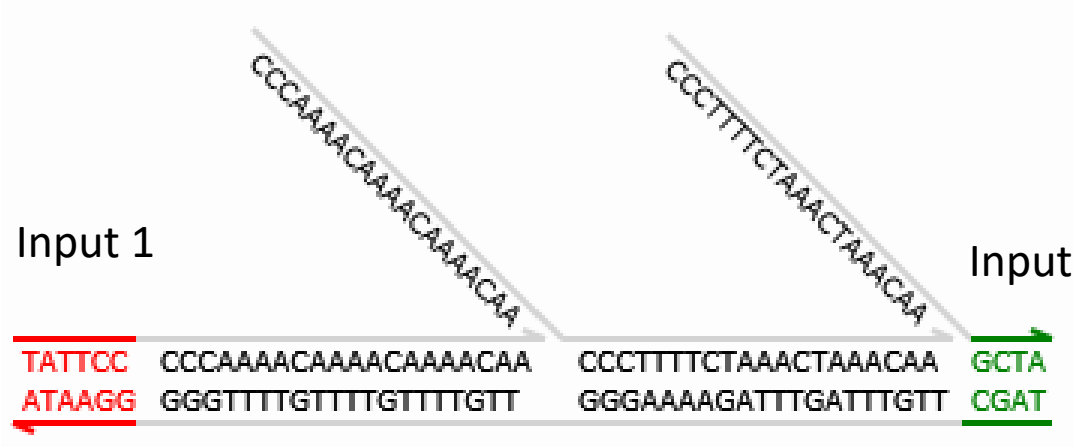
DNA Strand Displacement

DSD Logic Gate [Output = Input1 AND Input2]



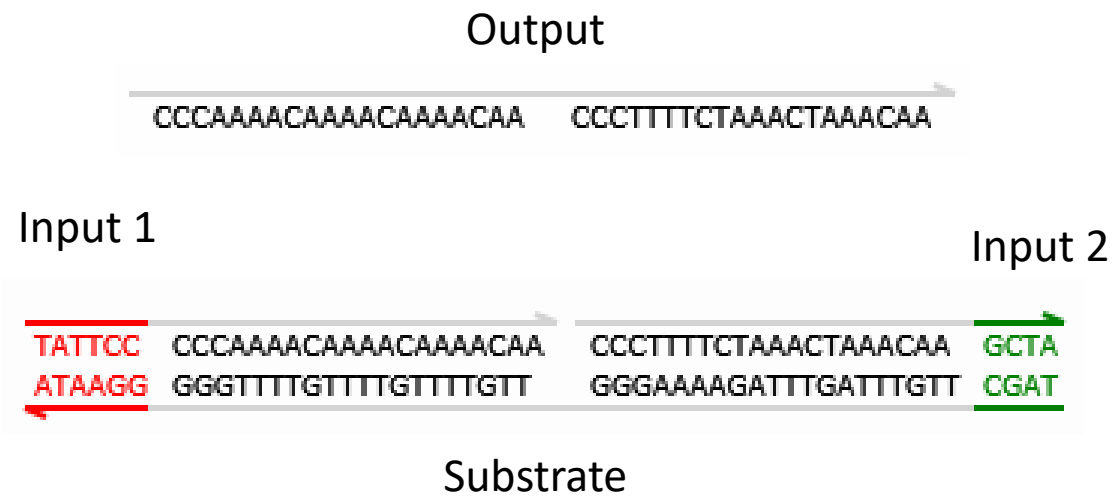
DNA Strand Displacement

DSD Logic Gate [Output = Input1 AND Input2]



DNA Strand Displacement

DSD Logic Gate [Output = Input1 AND Input2]



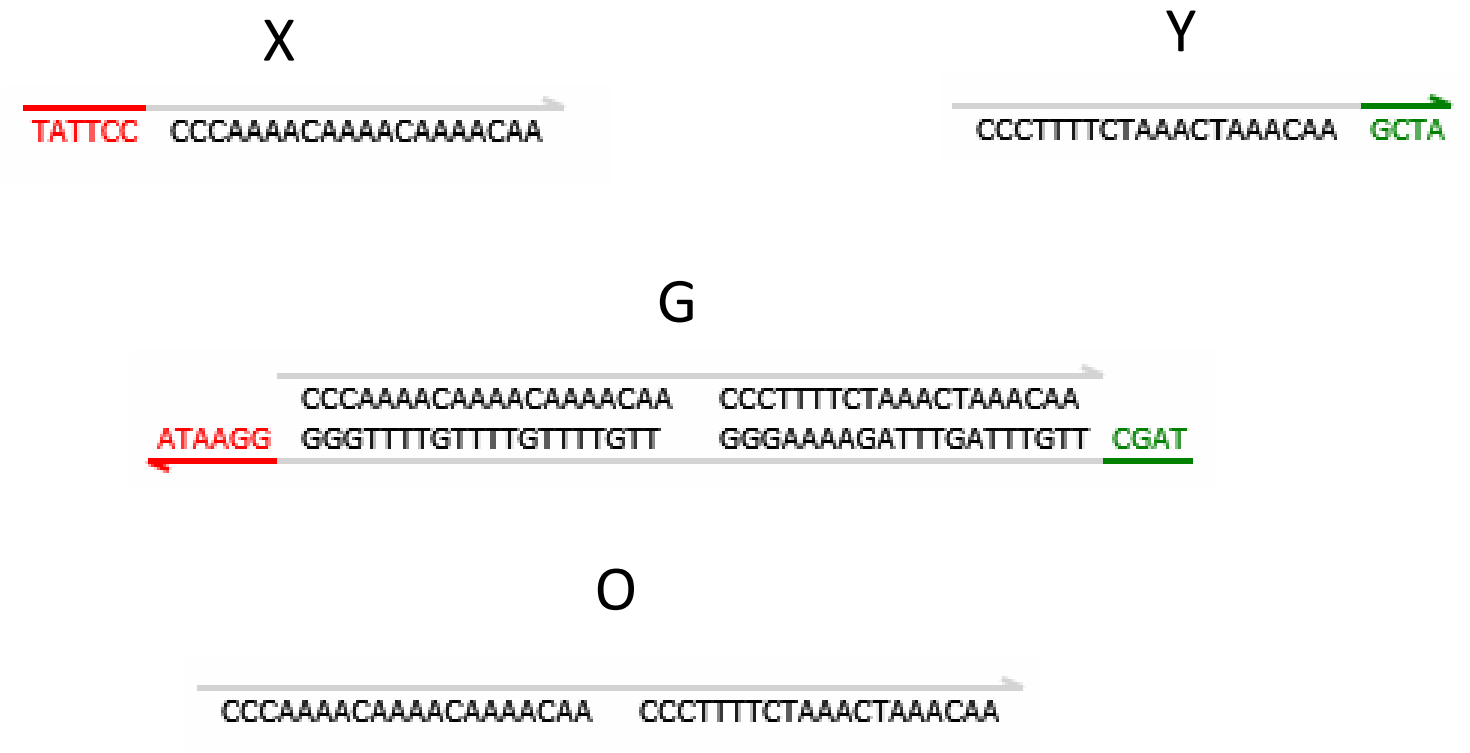
Chemical Reaction Networks (CRNs)

X + G <-> XG

Y + G <-> GY

XG + Y -> XGY + O

GY + X -> XGY + O



DNA Strand Displacement

Programming Examples

Specification	Program
Y := 2 X	X -> Y + Y
Y := ⌊X/2⌋	X + X -> Y
Y := X1 + X2	X1 -> Y X2 -> Y
Y := min (X1,X2)	X1 + X2 -> Y

Programming Examples

Specification

Y := max (X1,X2)

Program

X1 -> L1 + Y

X2 -> L2 +Y

L1 + L2 -> K

Y + K ->

max (X1,X2) := X1 + X2 – min(X1,X2)

Luca Cardelli, 2019

Computing with CRNs

What does the following CRN compute?

X + Y -> X + B

Y + X -> Y + B

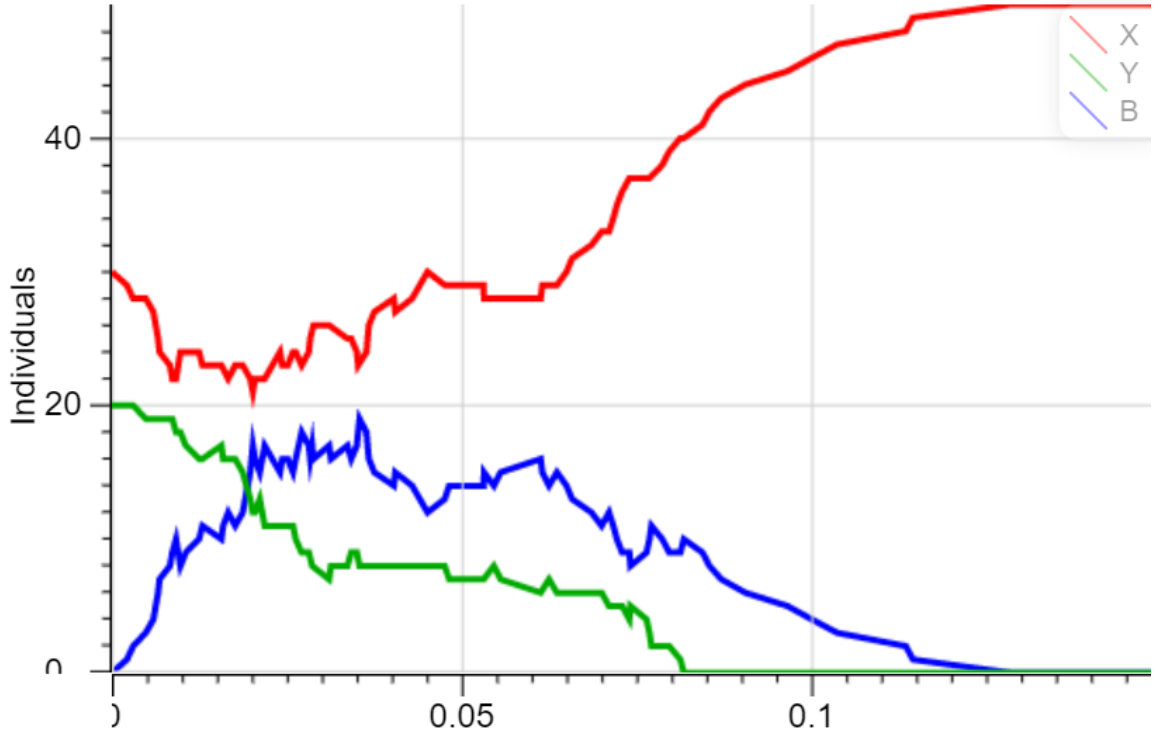
B + X -> X + X

B + Y -> Y + Y

Approximate Majority

Approximate Majority – Visual DSD

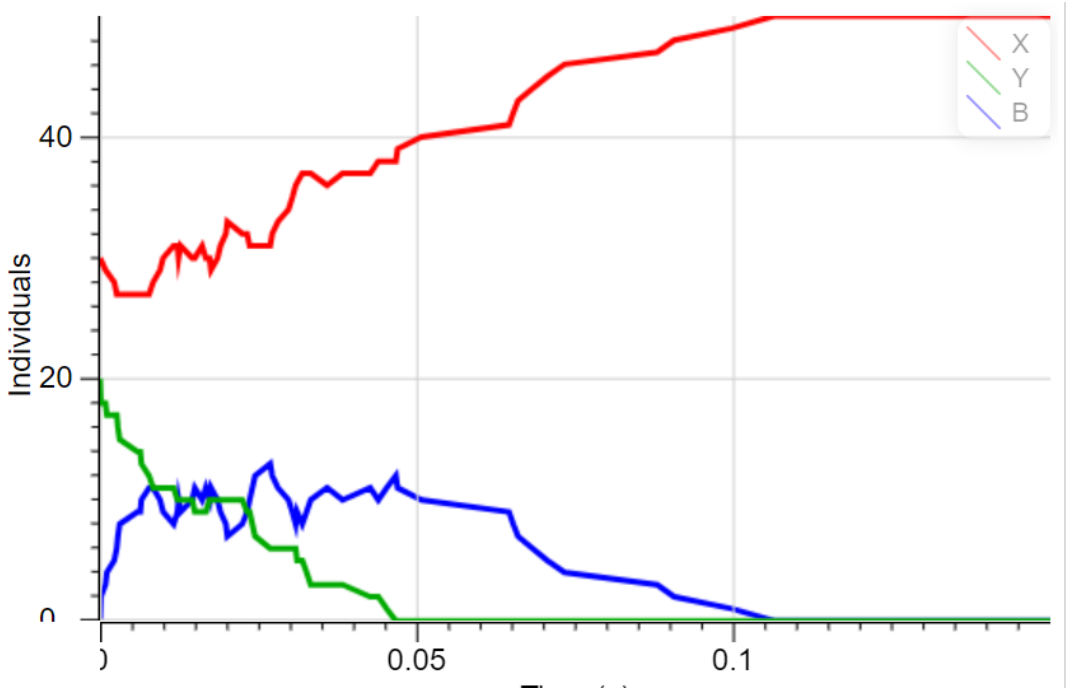
```
directive simulation { final=0.15; points=1000 }
directive parameters [r = 1.0]

| X + Y ->{r} X + B
| Y + X ->{r} Y + B
| X + B ->{r} X + X
| Y + B ->{r} Y + Y

| 30 X
| 20 Y
```



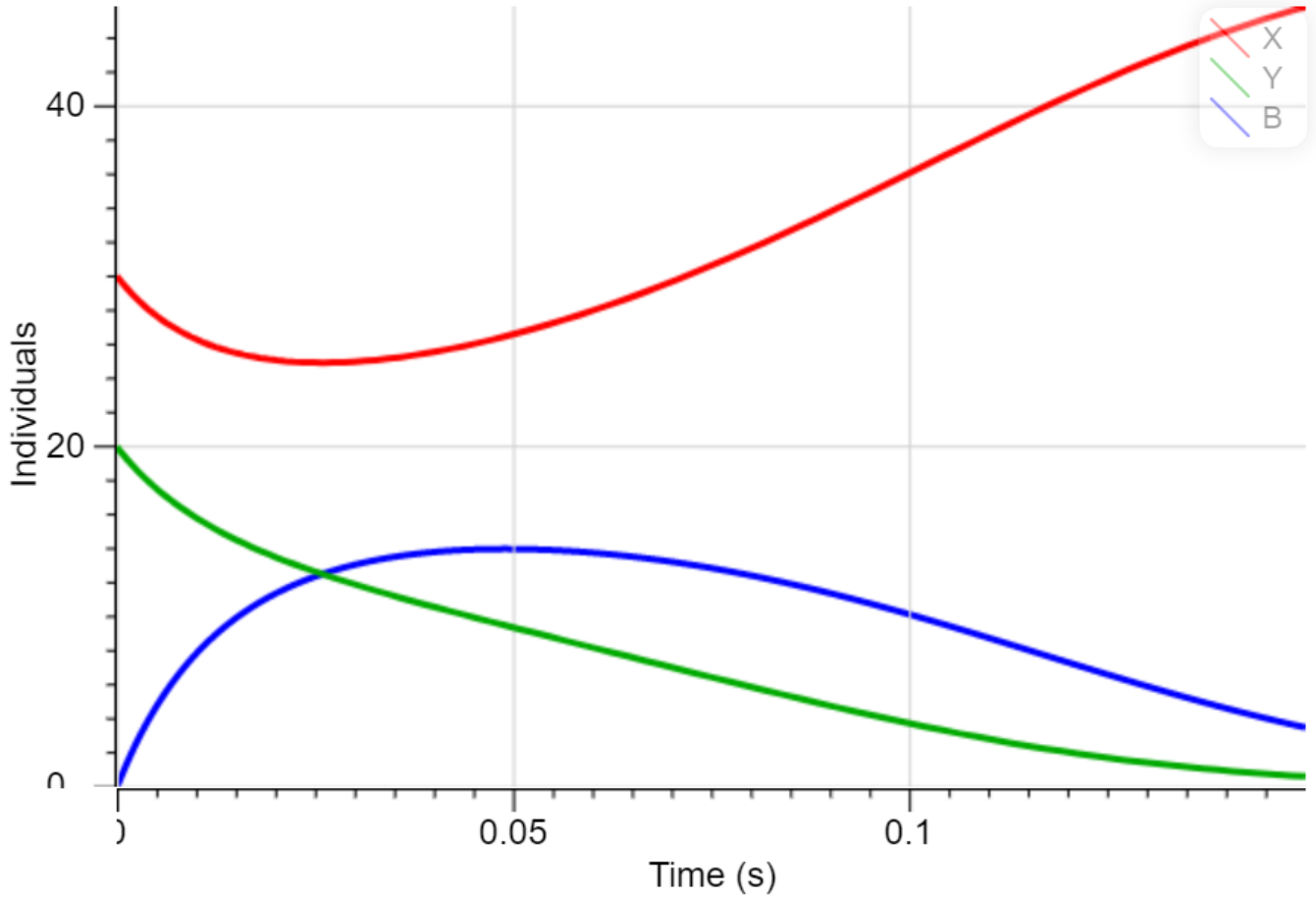
Phillips and Cardelli *RSIF* 2009 Laikin et al. *Bioinformatics* 2011

Approximate Majority – Visual DSD

```
directive simulation { final=0.15; points=1000 }
directive parameters [r = 1.0]

| X + Y ->{r} X + B
| Y + X ->{r} Y + B
| X + B ->{r} X + X
| Y + B ->{r} Y + Y

| 30 X
| 20 Y
```



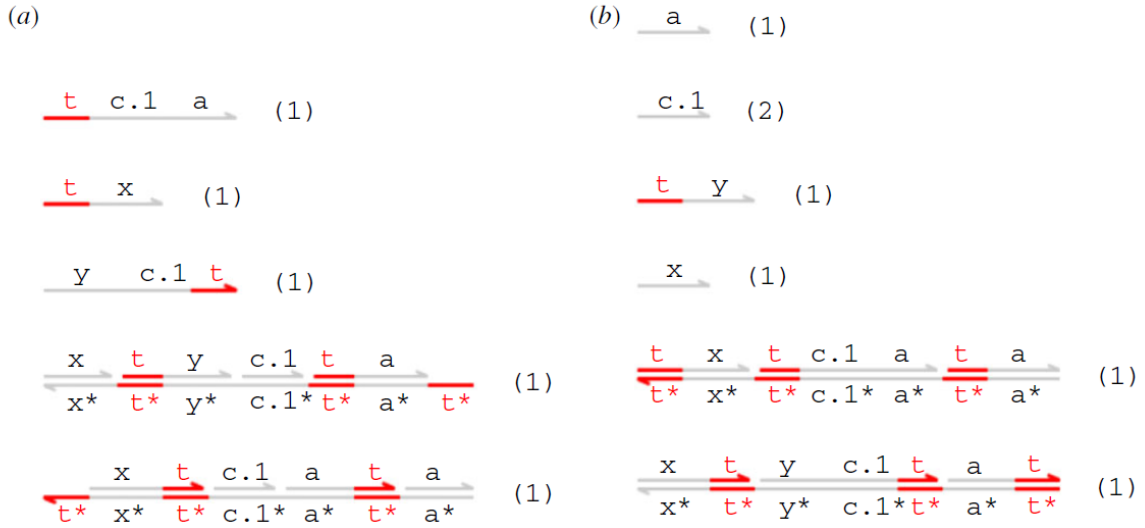
Approximate Majority – Continuous Semantics



Formal Verification of Strand Displacement Systems – Discrete Semantics

```
(* Signal strand *)
def S(N, x) = N * <t^ x>

(* Transducer gate *)
def T(N, x, y) = new c
  ( N * {t^*}:[x t^]:[c]:[a t^]:[a
  | N * [x]:[t^ y]:[c]:[t^ a]:{t^*
  | N * <t^ c a>
  | N * <y c t^> )
```

DSD Code - Transducer



Initial and expected final state

```
label "all_done"
  = strands_reactive=output &
    output=N & gates_reactive=0;
```

```
A [ G "deadlock" => "all_done" ]
E [ F "all_done" ]
```

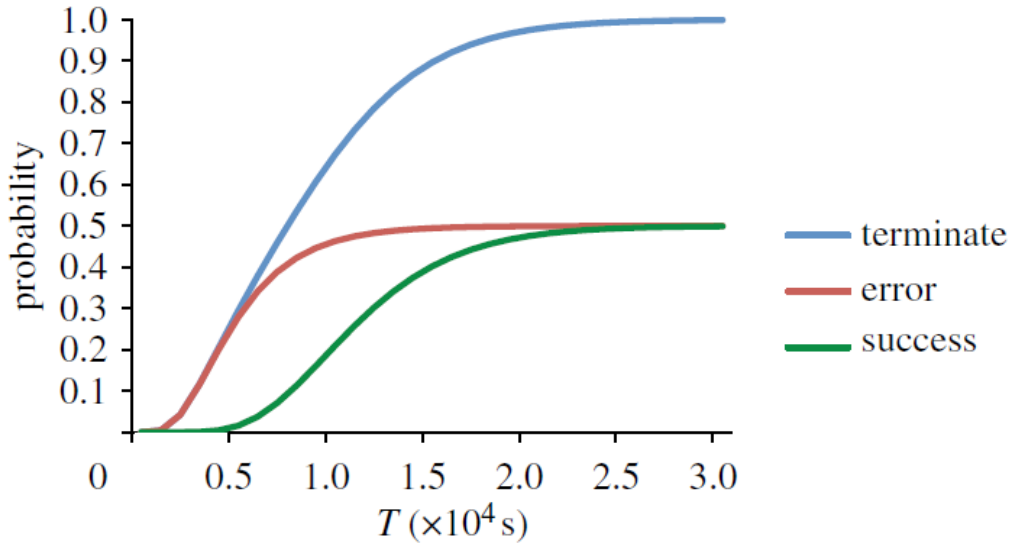
CTL property checked by PRISM

Lakin, Parker, Cardelli, Kwiatkowska, Phillips *RSIF* 2009

Probabilistic Verification – CTMC Semantics

```
(* Signal strand *)
def S(N, x) = N * <t^ x>

(* Transducer gate *)
def T(N, x, y) = new c
  ( N * {t^*}:[x t^]:[c]:[a t^]:[a]
  | N * [x]:[t^ y]:[c]:[t^ a]:{t^*}
  | N * <t^ c a>
  | N * <y c t^> )
```

DSD Code - Transducer



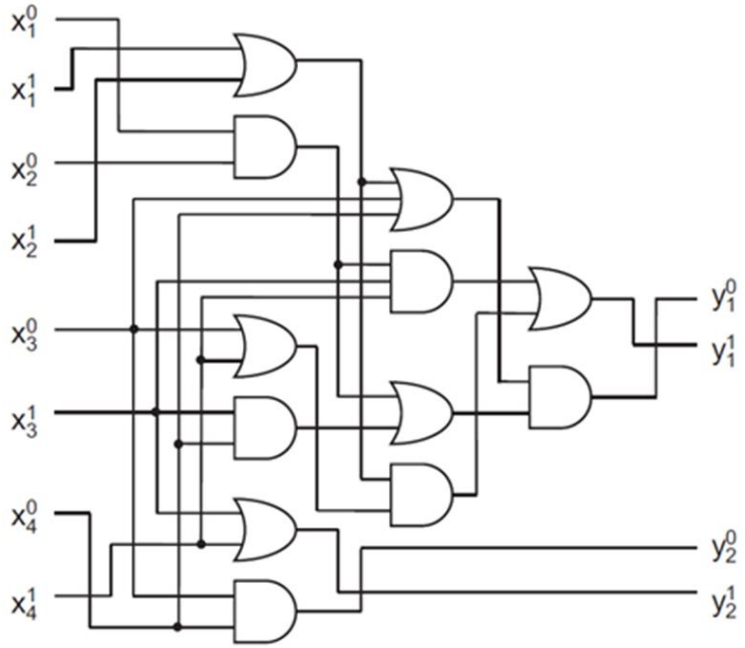

```
P=? [ F[T,T] "deadlock" ]
P=? [ F[T,T] "deadlock" & "all_done" ]
P=? [ F[T,T] "deadlock" & !"all_done" ]
```

PCTL property checked by PRISM

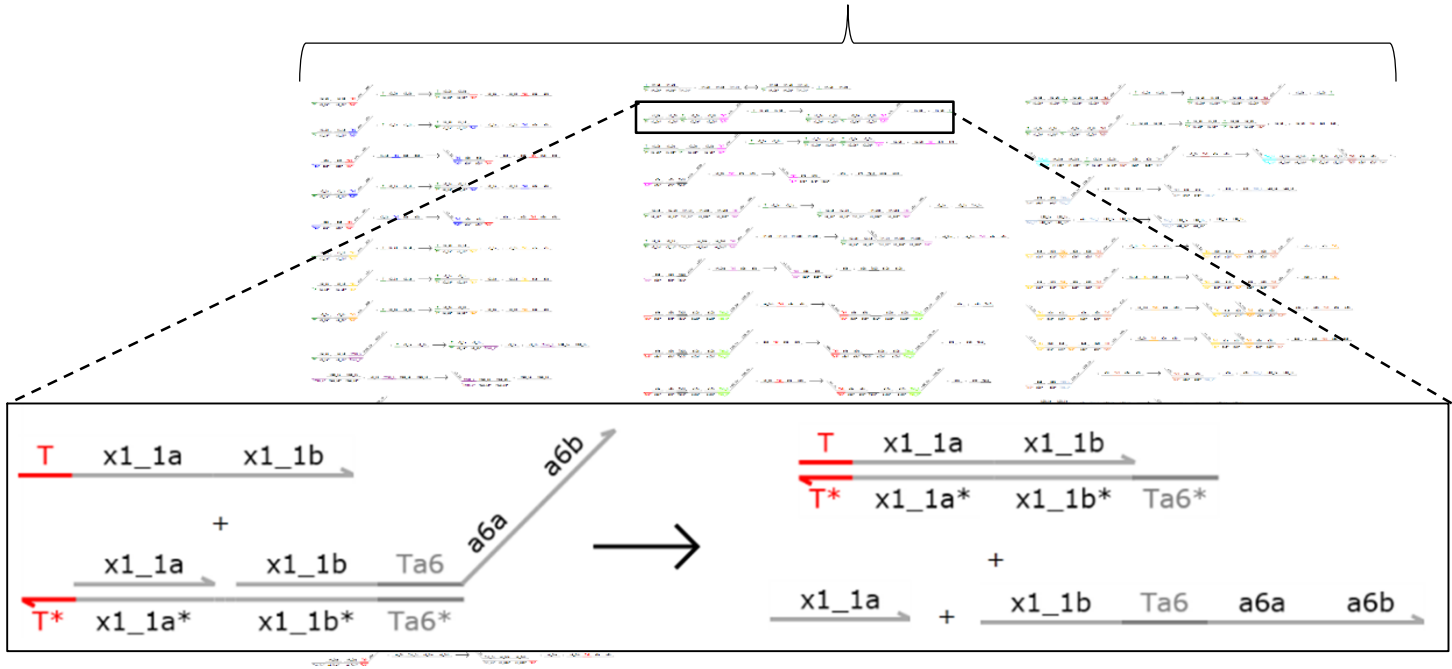
Lakin, Parker, Cardelli, Kwiatkowska, Phillips *RSIF* 2009

DNA device verification
Among DNA circuit constructed experimentally

[Qian, Winfree, Science, 2011; Chandran, Gopalkrishnan, Phillips, Reif, *DNA17*, 2011]



$$Output = \lfloor \sqrt{Input} \rfloor$$

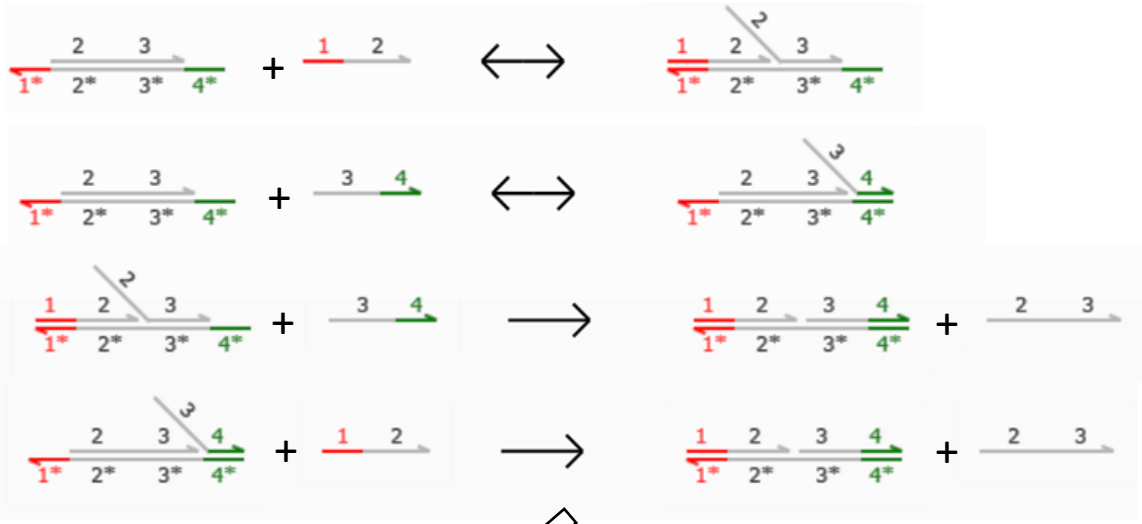


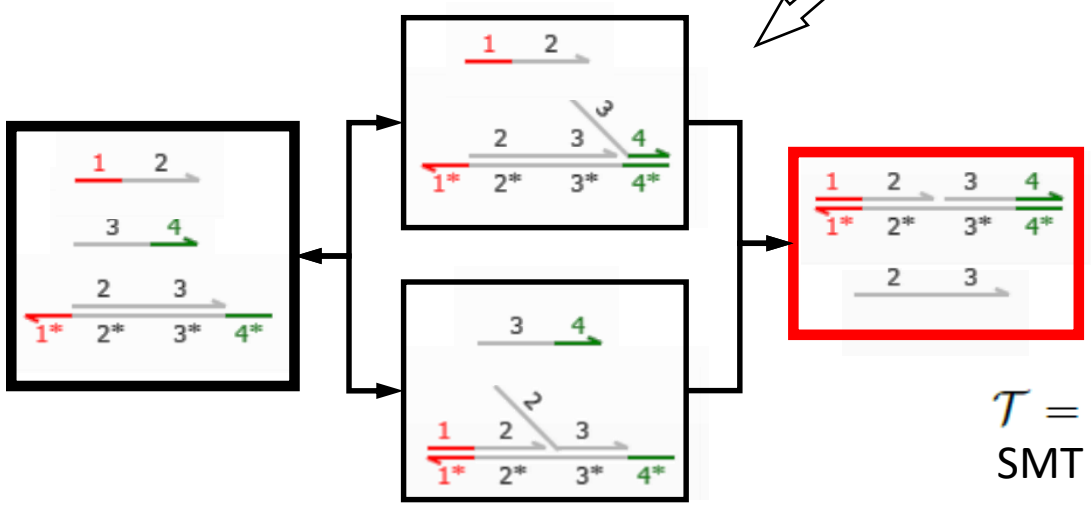
Yordanov, Wintersteiger, Hamadi, Phillips, Kugler. *DNA19, 2013*

Model Generation

```
def Input1() = <1^ 2>
def Input2() = <3 4^>
def Output() = <2 3>
def AND() = {1^*}[2 3]{4^*}
( N1*Input1()
| N2*Input2()
| N*AND()
| 0*Output()
)
```

Visual DSD



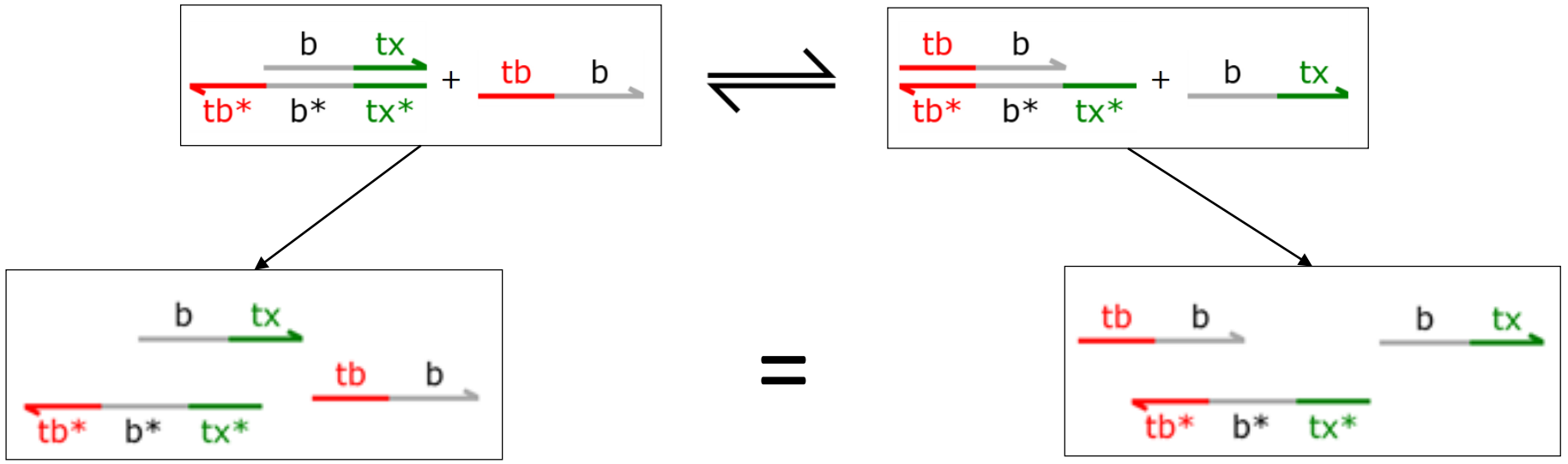


$\mathcal{T} = (Q, q_0, T)$

SMT encoding

DNA Verification Strategies

- Inductive invariants - conservation of strands



- Acceleration- multiple reactions firing

Yordanov, Wintersteiger, Hamadi, Phillips, Kugler. *DNA'19*
Yordanov, Wintersteiger, Hamadi, Kugler. *NFM'13*

Network Based Biocomputation

Bio4Comp Horizon2020 European Project:

Lund University, Technische Universität Dresden, Linnaeus University, Molecular Sense Ltd, Bar-Ilan University, Fraunhofer Gesellschaft

Develop Network Based Biocomputation:

- Speed up solution of complex problems
- Low energy
- Interface with Biological material

Formal Verification of Biocomputation Circuits

Bio4Comp

Network Based Biocomputation (NBC)

Build Nanofabricated device (using Electron Beam Lithography)

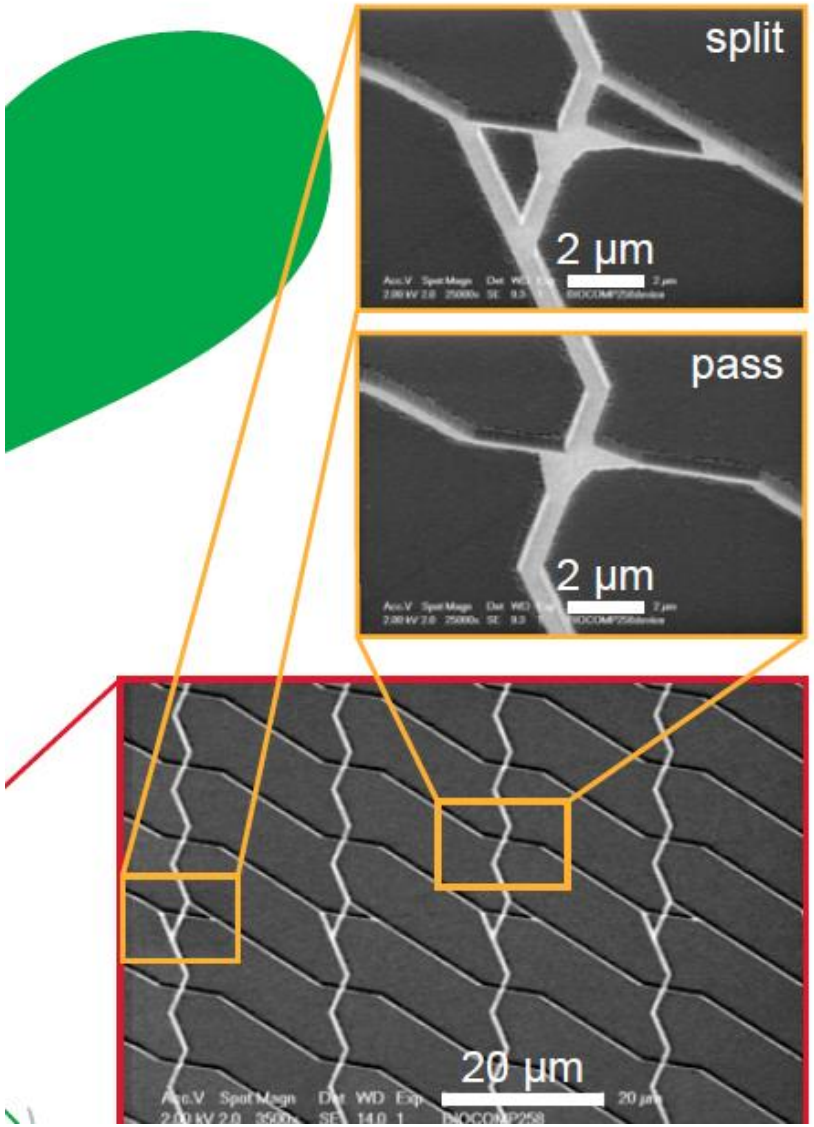
Molecular agents can travel through network exploring it in parallel

Use actin-myosin filaments or microtubles-kinesin (speeds 5-10 μms^{-1}, 0.5-1 μms^{-1})

Two Types of Junctions: Pass, Split

Measure agents exiting device

Deduce the answer using exit information



Nicolau et al. PNAS 2016

The Subset Sum Problem (SSP)

SSP : Given $S = \{ a_1, a_2 .. a_n \}$ each a_i an integer and integer T decide if there is a subset $S^* \subseteq S$ whose some is T, i.e. $\sum a_{i_k} = T$ where $a_{i_k} \in S^*$

Decision Problem – only need to answer:

Yes if there is such a subset
No if there isn't

Example $S = \{2,5,9\}$ $T = 11$ Yes $T = 13$ No

SSP is known to be NP Complete (NPC)
$\Rightarrow$ No polynomial algorithm unless P = NP
$\Rightarrow$ Can reduce many other interesting problems to SSP

Network Based Biocomputation (NBC)

Solving SSP using NBC:

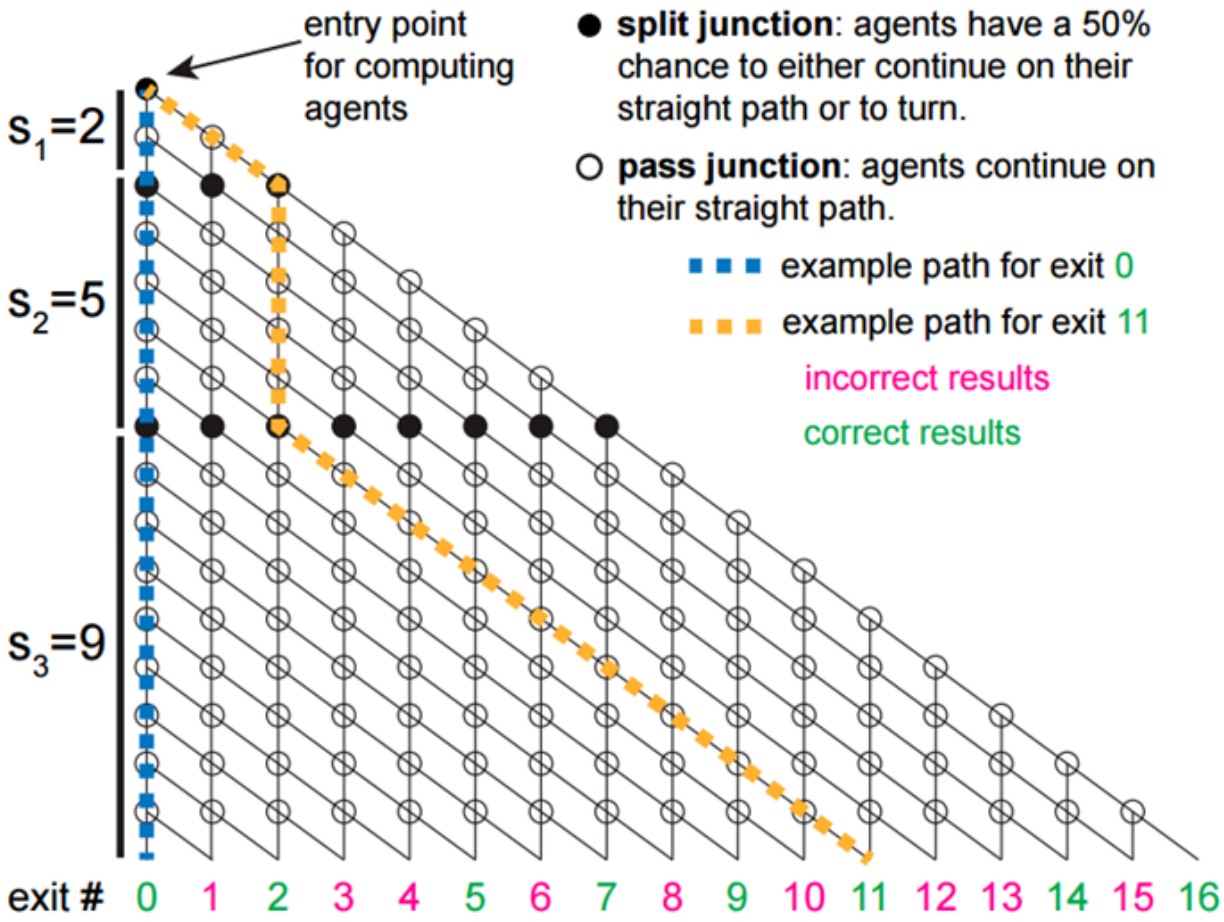
Agents start at top left entry and exit at bottom row

At split junctions ● agents can decide if to move down or diagonally

At pass junctions ○ agents continue in the direction it was travelling

A choice to move diagonally in split junction -> adding a number to the sum

Agent traversing yellow path 11 = 2 + 9



Network Based Biocomputation

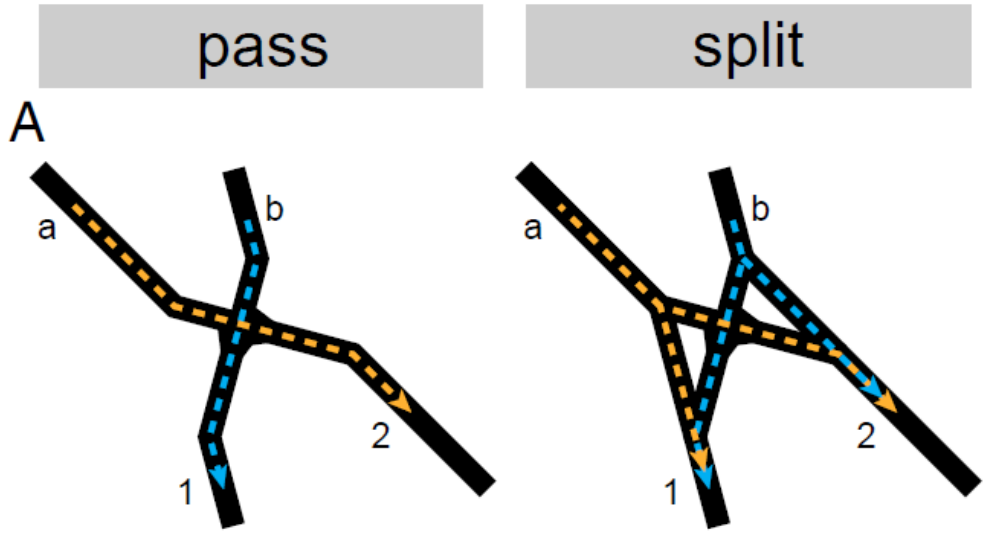
Some experimental challenges:

Distribution in split junction
not symmetric

Errors in pass junctions

Agents get stuck in junction

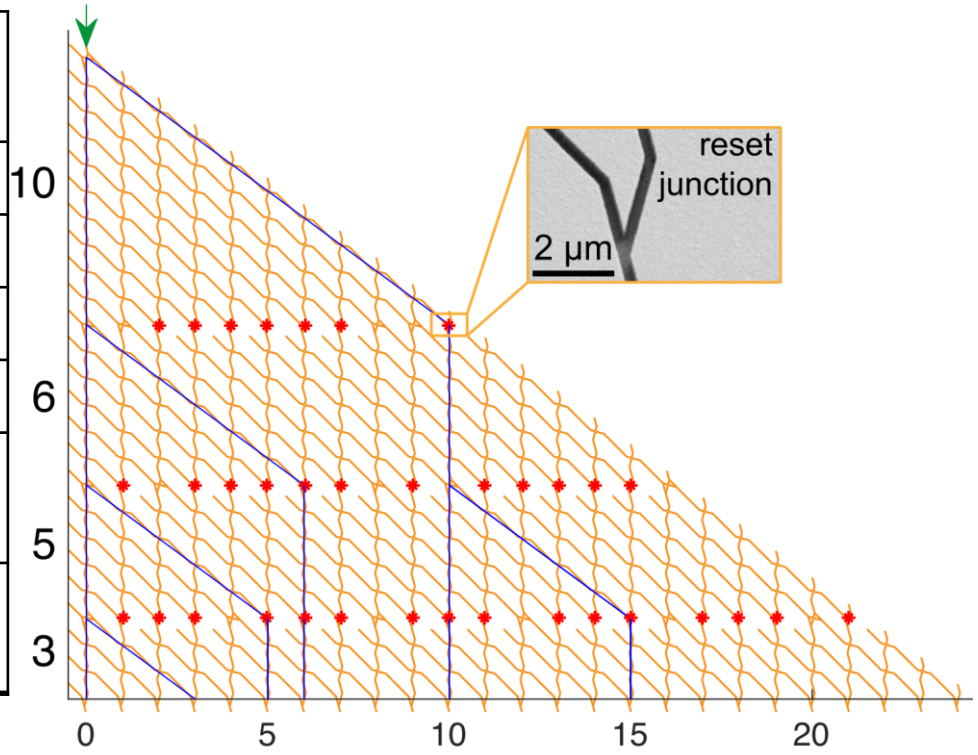
Agents “climb out” of tracks and
land in wrong positions



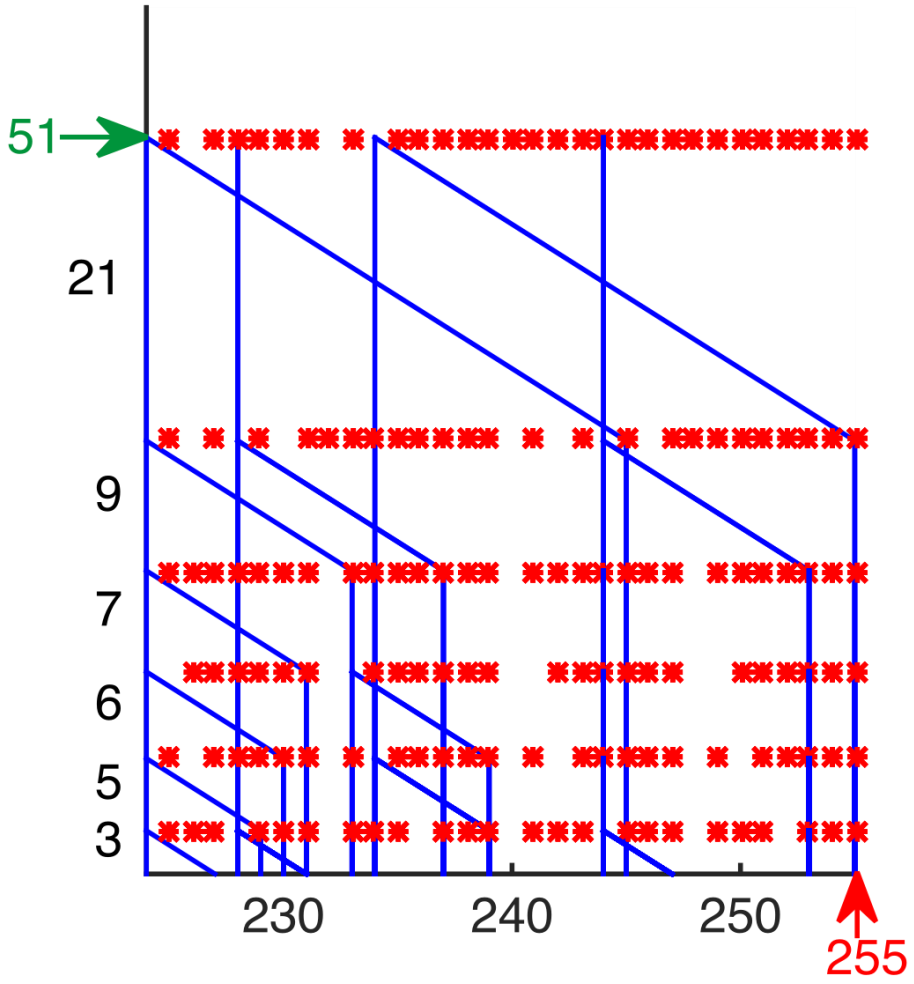
Network Encoding of ExCov (Exact Cover)

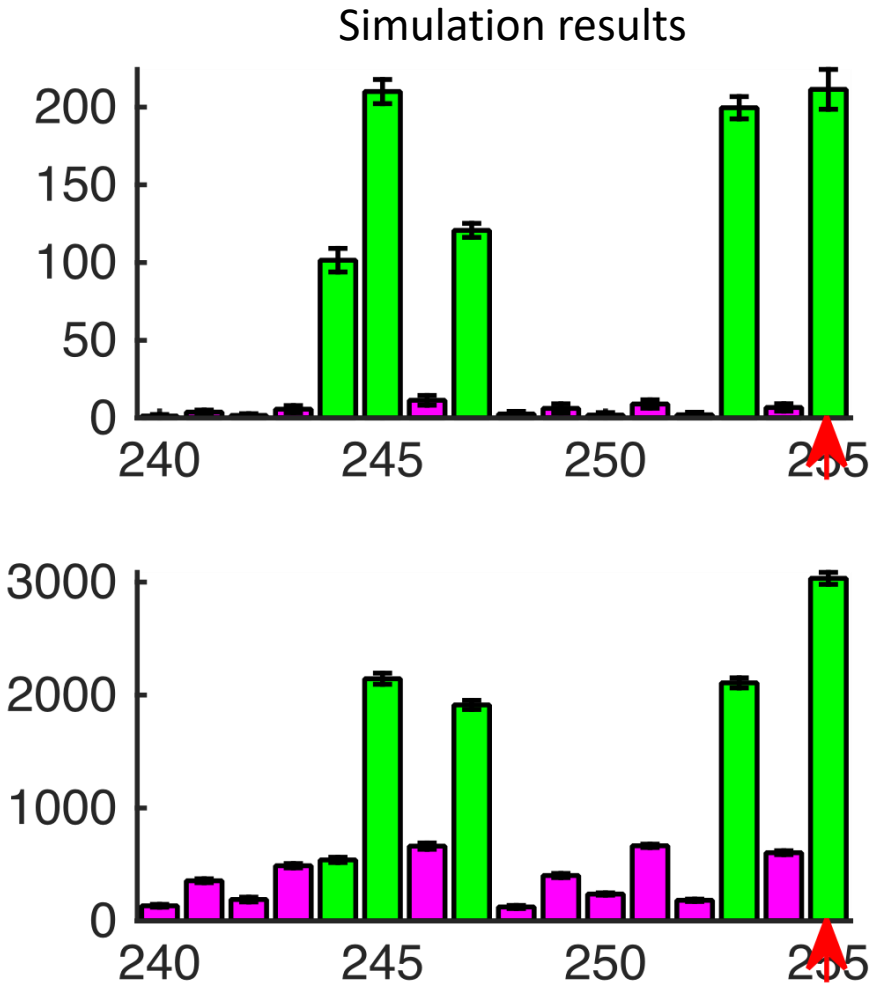
- EXCOV sets represented as binary numbers
- Each EXCOV Set encoded into the network as one decimal number
- RESET junctions prevent addition of colliding sets

	Set 1 {2;4}	**Set 2 {2;3}**	**Set 3 {1;3}**	**Set 4 {1;2}**
2^0	0	0	1	1
2^1	1	1	0	1
2^2	0	1	1	0
2^3	1	0	0	0
Decimal Numbers	10	6	5	3
Target Sum	15			



10-set EXCOV: One Solution





Formal Verification of NBC Circuits

Eliminate logical errors before manufacturing circuits

Prototype new NBC ideas, complementing simulation tools

Identify faulty junctions using experimental measurements of exits

Formal Verification of SSP Network

Define Transition System:

Variables x, y, dir

x, y : 1 .. $(\sum a_i)$

dir : {0,1} (0 – down, 1 – diagonally)

Transition Relation

$y' = y + 1$

$(x' = x \ \wedge dir = 0) \ \vee (x' = x + 1 \ \wedge dir = 1)$

dir allowed to non-deterministically choose 0 or 1 at split junction, no change in pass junction.

Initial Condition

$(x' = 1 \ \wedge y = 1 \wedge (dir = 0 \vee dir = 1))$

Formal Verification of SSP Network

$$\text{CTLSPEC NAME csum} := !(EX(AG((flag = FALSE)|(!(column = sum)))));$$

$$\text{CTLSPEC NAME nsum} := !(EF((flag = TRUE)\&(column = xsum)));$$

Table 8 SSP general sum verification runtimes in minutes.

SSP						
ID	Set Size	Set	Spec	Spec Validity	Tag Runtime	No Tag Runtime
0	3	[2, 3, 5]	csum	VALID	0.0011	0.0011
0	3	[2, 3, 5]	nsum	VALID	0.0009	0.0009
1	4	[2, 3, 5, 7]	csum	VALID	0.0013	0.0012
1	4	[2, 3, 5, 7]	nsum	VALID	0.0009	0.0009
2	5	[2, 3, 5, 7, 11]	csum	VALID	0.0018	0.0013
2	5	[2, 3, 5, 7, 11]	nsum	VALID	0.0009	0.0009
3	6	[2, 3, 5, 7, 11, 13]	csum	VALID	0.0037	0.0018
3	6	[2, 3, 5, 7, 11, 13]	nsum	VALID	0.0009	0.0009
4	7	[2, 3, 5, 7, 11, 13, 17]	csum	VALID	0.0092	0.0025
4	7	[2, 3, 5, 7, 11, 13, 17]	nsum	VALID	0.0009	0.0009
5	8	[2, 3, 5, 7, 11, 13, 17, 19]	csum	VALID	0.0260	0.0042
5	8	[2, 3, 5, 7, 11, 13, 17, 19]	nsum	VALID	0.0009	0.0009
6	9	[2, 3, 5, 7, 11, 13, 17, 19, 23]	csum	VALID	0.0821	0.0074
6	9	[2, 3, 5, 7, 11, 13, 17, 19, 23]	nsum	VALID	0.0010	0.0009

Future Outlook

Formal Verification tools used as mainstream approach in Genetic Network Inference and Analysis

Whole Tissue models – Verification and Reasoning

Industrial applications for biodevices will require certification opening key role for FV tools

Thanks for Listening !

Til Korten, Stefan Diez - Technische Universität Dresden
Dan Nicolau Jr. - Molecular Sense Ltd.

Sara Jane Dunn, Boyan Yordanov, Andrew Phillips – Microsoft Research Cambridge

Michelle Aluf-Medina, Tamar Viclizky, Ani Amar, Amit Schussheim, Avraham Raviv – Bar Ilan University

Jane Hubbard NYU

David Harel Weizmann

All Bio4Comp members

Bio4Com

Funding: European Commission Horizon 2020
Israeli Science Foundation (ISF)