Chapter 6: Regulatory Networks

6.3 Boolean Network Models

Prof. Yechiam Yemini (YY)
Computer Science Department
Columbia University

Overview

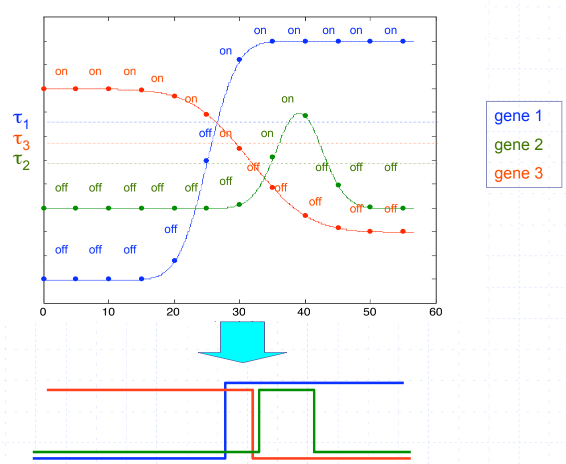
- Boolean network models
- Sample applications
- Kaufmann's theory of evolution
- Learning (reverse engineering) Boolean nets

2

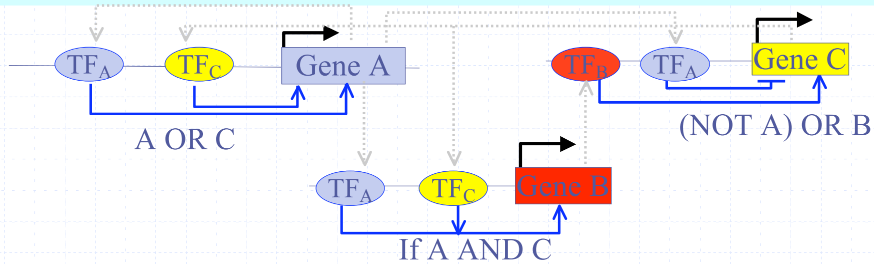
Intro To Boolean Networks

Example: Thresholding Gene Expression

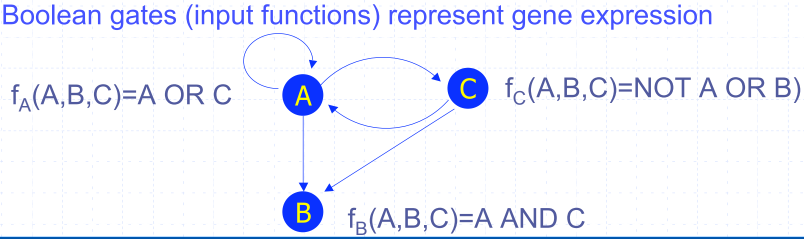
- Boolean model: discretize expressions to on/off model



Boolean Network Model



- A Boolean Network Model:
  - Nodes represent transcription factors
  - Edges represent regulatory input
  - Boolean gates (input functions) represent gene expression



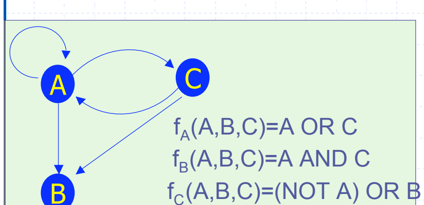
5

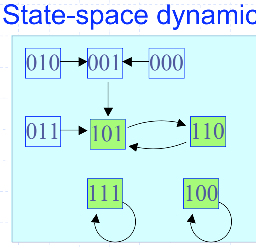
Dynamics

- Network State: X=(A,B,C…) is a Boolean vector
- State evolution: X(t+1)=f(X(t))=(f_A(X(t)),f_B(X(t)), ….)
  - E.g., X(t+1)=(A OR C, A AND C, (NOT A) OR B)
  - (0,1,1)=>(1,0,1)
- This is discrete time synchronous dynamics
  - State transitions occur through concurrent gates firings

X(t)	X(t+1)
000	001
001	101
010	001
011	101
100	100
101	110
110	101
111	111

State-space dynamics





Cycle

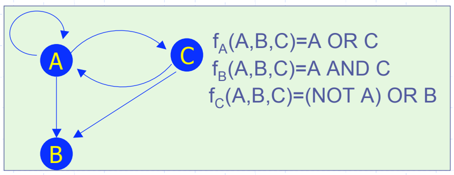
Attractors

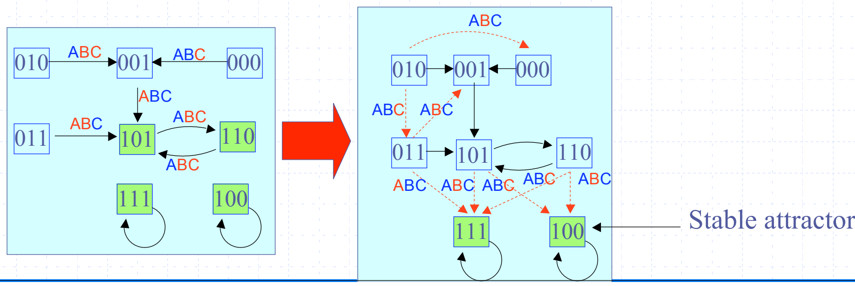
6

Noisy (Stochastic) Dynamics

- If gene-gates "fire" randomly
- The network becomes asynchronous
- The dynamics landscape changes

X(t)	X(t+δ)
000	001
001	101
010	001,000,011
011	101,111,001
100	100
101	110, 111,100
110	101,111,100
111	111

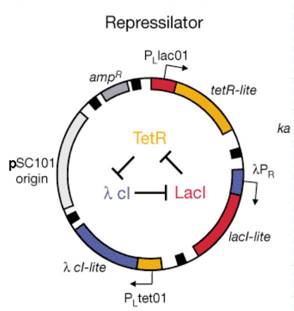


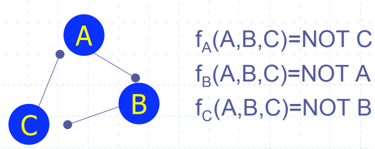


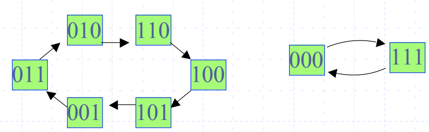
7

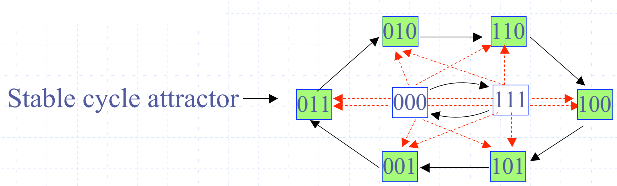
Example: Boolean Repressilator

- Repressilator has three repressors in a loop:





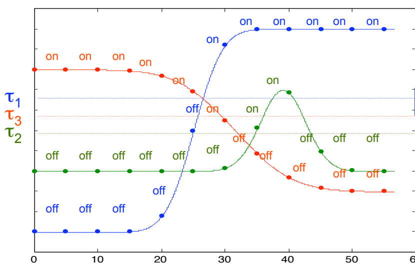




8

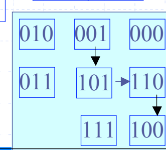
More Generally

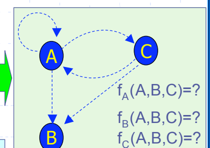
- Boolean Network:
  - A digraph G=<V,E>; nodes = genes, edges=regulation
  - For each node assign a Boolean function over ingress neighbors
- Attractors & cycles describe dynamics of expression
- Learning (reverse engineering; identification):
  - Extract Boolean network model from expression levels



Time	State
1	001
2	101
3	110
4	100

001	101
101	110
110	100

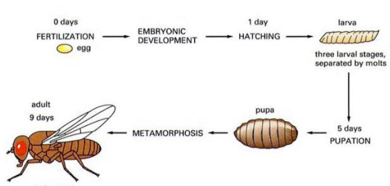


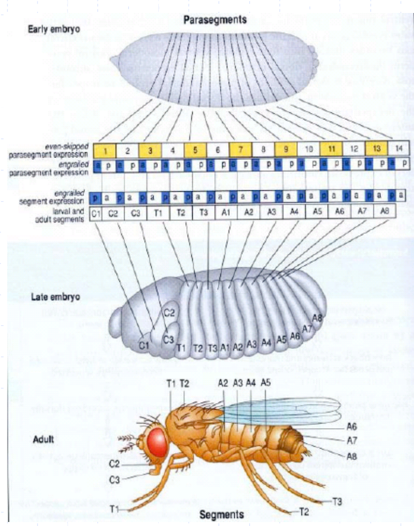


Example: Regulation of Drosophila Patterns

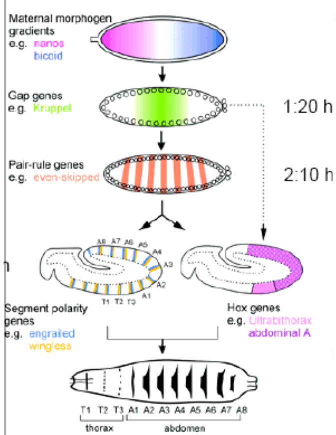
R. Albert & H.G., Othmer; Journal of Theoretical Biology 223 (2003)

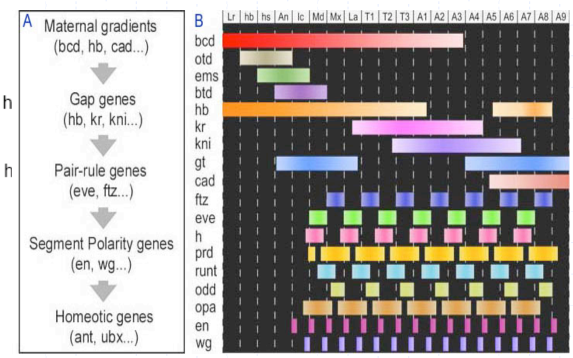
- Fundamental question: how do gene's regulate spatial patterns?





Segmentation Is Regulated By A Cascade

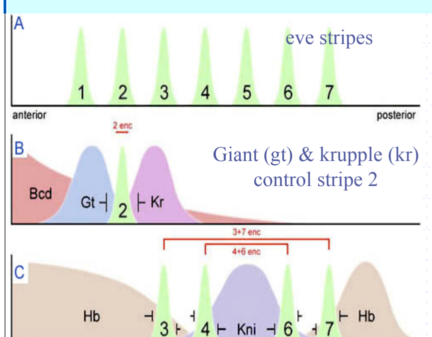




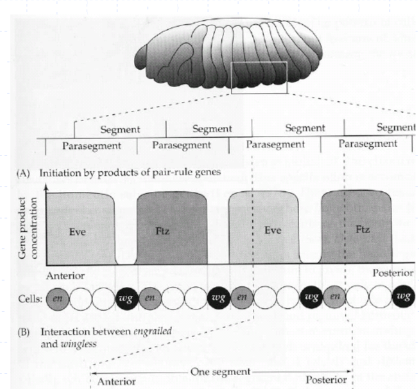
- Genes are activated in precise temporal order
- Use regulatory interaction to coordinate development functions

11

Segmentation Is Regulated By A Cascade



knirps (kni) & hunchback (hb) control stripes 3-7



- Pair rule genes initiate stripes
  - eve, ftz..
- Segment polarity genes control anterior/posterior structure
  - Engrailed, wingless

12

A Boolean Network Model

R. Albert & H.G., Othmer; Journal of Theoretical Biology 223 (2003)

$$en_i^{t+1} = (WG_{i-1}^t \text{ or } WG_{i+1}^t) \text{ and not } SLP_i^t$$

$$hh_i^{t+1} = EN_i^t \text{ and not } CIR_i^t$$

$$ptc_i^{t+1} = CIA_i^t \text{ and not } EN_i^t \text{ and not } CIR_i^t$$

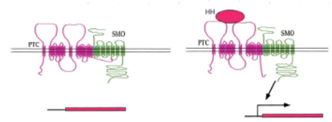
$$ci_i^{t+1} = \text{not } EN_i^t$$

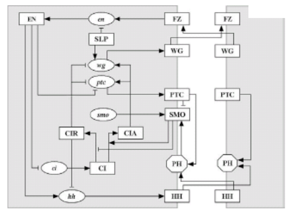
$$EN_i^{t+1} = en_i^t$$

$$WG_i^{t+1} = wg_i^t$$

$$CI_i^{t+1} = ci_i^t$$

$$HH_i^{t+1} = hh_i^t$$





$$PH_i^t = PTC_i^t \text{ and } (HH_{i-1}^t \text{ or } HH_{i+1}^t)$$

$$SMO_i^t = \text{not } PTC_i^t \text{ or } HH_{i-1}^t \text{ or } HH_{i+1}^t$$

$$wg_i^{t+1} = (CIA_i^t \text{ and } SLP_i^t \text{ and not } CIR_i^t) \text{ or } [wg_i^t \text{ and } (CIA_i^t \text{ or } SLP_i^t) \text{ and not } CIR_i^t]$$

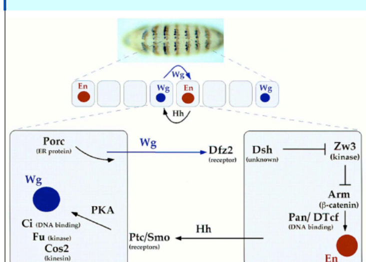
Either of the activators can counter mRNA decay.

$$PTC_i^{t+1} = ptc_i^t \text{ or } (PTC_i^t \text{ and not } HH_{i-1}^t \text{ and not } HH_{i+1}^t)$$

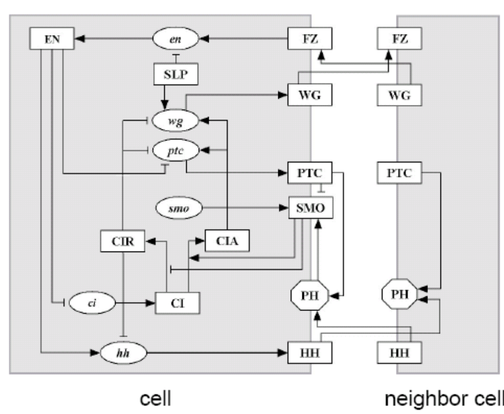
Free PTC does not decay .

13

Coordinated Regulation



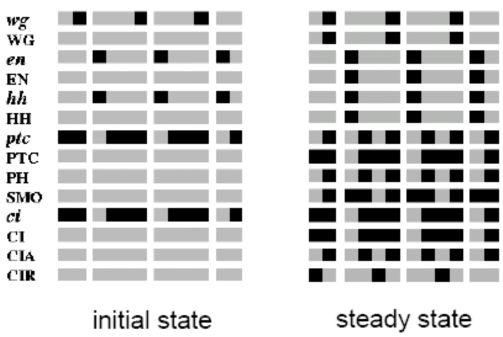
Cadigan, Nusse, Genes & Development 11, 3286 (1997)



14

The Steady State (Attractor)

The model reproduces the wild type steady state

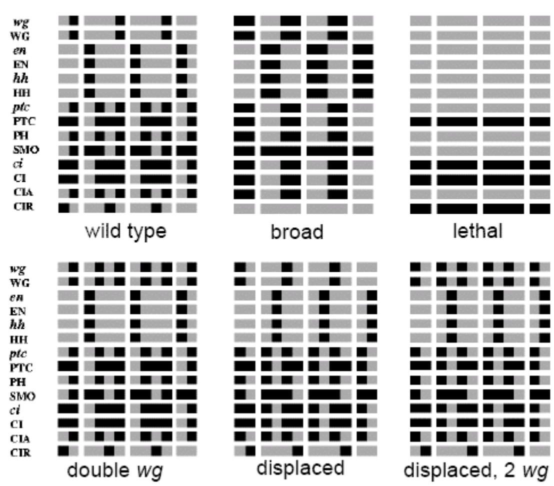


Agrees with the wild type pattern of the segment polarity genes.

15

Compute Attractor Expression

Possible stable patterns

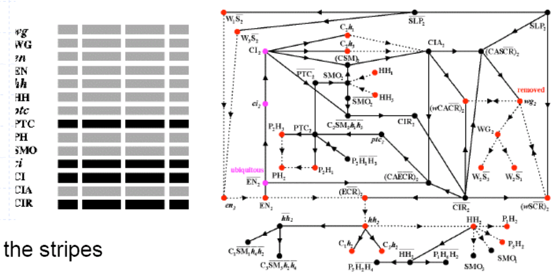


The latter states have very small probability.

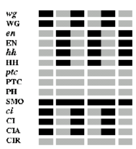
16

Mutations (Perturbations)

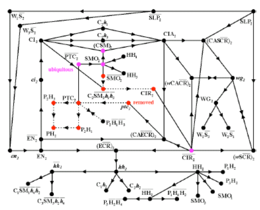
wg, *en* or *hh* mutations are lethal



ptc mutation broadens the stripes



final state



The *wg*, *en* and *hh* stripes broaden, regardless of initial state.

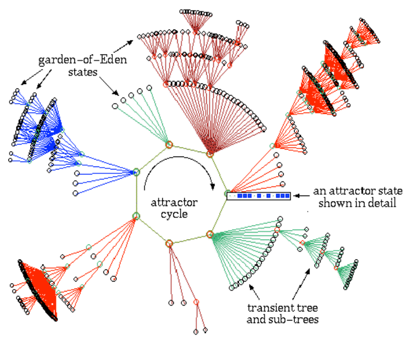
17

Kauffman’s Model

Kauffman’s Model [60’s, 93]

- Study Boolean networks to describe evolution
- BN: a graph of “genes” each with a random Boolean function
  - N=# of nodes; k=connectivity
- BN traverses trajectories over the hypercube $[0,1]^n$
- Converges to best fit response to random inputs

- Trajectories: series of state transitions
- Attractors: repeating trajectories
- Basin of Attraction: all states leading to an attractor

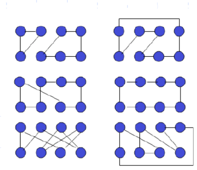


One attractor basin for a BN n=13, k=3. The cycle is of size 7

Evolution of Boolean Networks

- Nature evolves an ensemble of networks
  - Mutations change connectivity/gene-transition-function
- Genes select best-fitness transition functions
- What happens if k is large (e.g., k=N-1)?
  - X(t+1) is uncorrelated with X(t)
  - The number of attractors is very small; cycles are huge with period of some $2^{0.5N}$
  - Most genes would be oscillating
  - Network is very sensitive to small perturbations
- Need to keep k small
  - K=1, too small; gene’s do not interact
  - K=2; large number of attractors ~$N^{0.5}$; avg cycle $N^{0.5}$

An ensemble of N=8, k=2 Networks

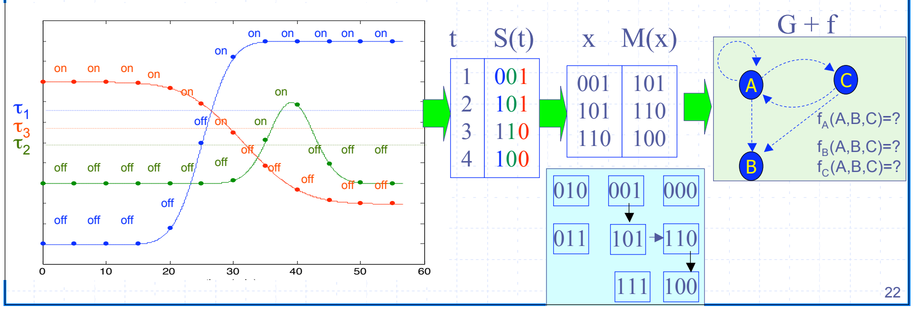


Learning Boolean Nets

(“Reverse Engineering” “Identification”)

The Challenge

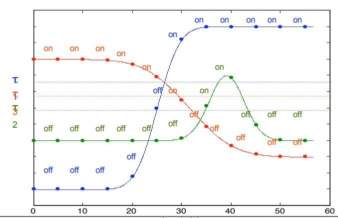
- Discretization: Given expression profiles vector X(t)
  - Set expression thresholds τ (how?)
  - Extract a time-state map S(t); compute state transitions map M(x)
- Learning:
  - Given: state transition map M
  - Compute: a Boolean vector function f such that M(x)=f(x)



22

Akutsu Algorithm (99)

- Brute force search for f
- Fix k, and consider networks of max degree k
  - For each gene i, and for each subset of k ingress genes find all functions f_i that are compatible over this ingress set for all {S(r)}
  - i.e., $S'_i(r)=f_i(S'(r-1))$ where S' is the restriction of S to the ingress set
  - For k fixed: $O(k2^{2^k}n^{k+1}m)$; if k is not fixed, learning is NP complete.
- Notes
  - Works for small k…does not handle noise…
  - Later improvements handle noise



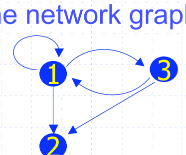
t	S(t)
1	001
2	101
3	110
4	100

23

Suppose the Network Graph is Known

- Given M and G, computing f is simple:
  - The truth table for $f_i(X)$ obtains by projecting M
  - The network graph G guides the projections

G
+
M



Input state			Output state		
X_1	X_2	X_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

X_1	X_3	f_1
0	0	0
0	1	1
1	0	1
1	1	1

$f_1(X) = X_1 \text{ OR } X_3$

24

How Do We Find The Network Graph ?

- An intuitive approach:
  - $f(X_1,X_2,X_3)$ depends on X_1, iff $f(1,X_2,X_3) \neq f(0,X_2,X_3)$ for some (X_2,X_3)
  - We call such values $<X_1,X_2,X_3,f>$ "dependency"

X_1	X_2	X_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

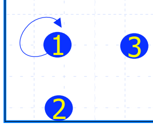
f_1 depends on X_1

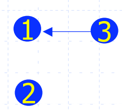
X_1	X_2	X_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

f_1 is independent of X_2

X_1	X_2	X_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

f_1 depends on X_3





25

An Intuitive Algorithm

- Repeat for all X_i and f_k :
  - Scan M to find a dependency of f_k on X_i; if found then add an X_i=>f_k edge to G
  - Else (no dependency found) then f_k is independent of X_i

X_1	X_2	X_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

f_1 depends on X_1

X_1	X_2	X_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

f_1 depends on X_3

X_1	X_2	X_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

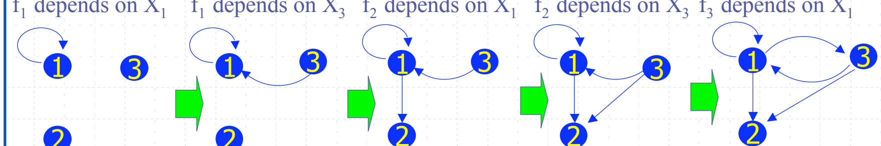
f_2 depends on X_1

X_1	X_2	X_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

f_2 depends on X_3

X_1	X_2	X_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

f_3 depends on X_1



26

REVEAL (98 Liang)

- Compute network graph from mutual information measure
- Base theory:
  - Let <X,Y> be an <input,output> stream
  - Consider H(Y), the entropy of Y, and M(X,Y), the mutual information of X and Y
  - If M(Y,X)=H(Y) then X determines Y uniquely
- $H(X) = - \sum p_i \log(p_i)$
  - p_i is the probability that a random element of data stream X is i
- M(X, Y)= H(X) + H(Y)- H(X,Y)

27

REVEAL Algorithm

- Step 1: compute state transition <input,output> table

Input stream			Output stream		
A_{i-1}	B_{i-1}	C_{i-1}	A_i	B_i	C_i
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

28

Step 2a: Compute Entropies

Input stream value			Output stream value		
A_{i-1}	B_{i-1}	C_{i-1}	A_i	B_i	C_i
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

$P(A_i=0)=2/8=0.25$
$P(A_i=1)=6/8=0.75$

$H(A_i) = -((0.25)\log(0.25) + (0.75)\log(0.75))$
$= 0.81$

29

Step 2a: Compute Entropies

note: $\lim_{x^+\to 0} x^x=1$, therefore in the left-hand limit, $(0)\log(0) = 0$.

$H(A_i) = -((0.25)\log(0.25) + (0.75)\log(0.75)) = 0.81$

$H(B_i) = -((0.75)\log(0.75) + (0.25)\log(0.25)) = 0.81$

$H(C_i) = -((0.5)\log(0.5) + (0.5)\log(0.5)) = 1$

$H(A_{i-1}) = H(B_{i-1}) = H(C_{i-1}) = -((0.5)\log(0.5) + (0.5)\log(0.5)) = 1$

$H(A_{i-1}, C_{i-1}) = -((0.25)\log(0.25) +(0.25)\log(0.25)+(0.25)\log(0.25)+(0.25)\log(0.25)) = 2$

$H(C_i, A_{i-1}) =-((0.5)\log(0.5) + (0.5)\log(0.5) = 1$

$H(A_i, A_{i-1}, C_{i-1})=-((0.25)\log(0.25)+(0.25)\log(0.25)+ (0.25)\log(0.25)+(0.25)\log(0.25))=2$

$H(B_i, A_{i-1}, C_{i-1})=-((0.25)\log(0.25)+(0.25)\log(0.25)+(0.25)\log(0.25)+(0.25)\log(0.25))= 2$

……

……

30

Step 2b: Compute Network

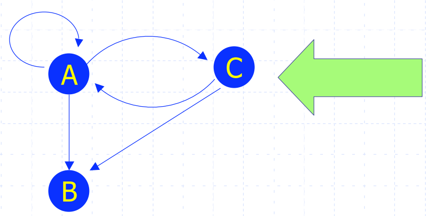
- First compute mutual information

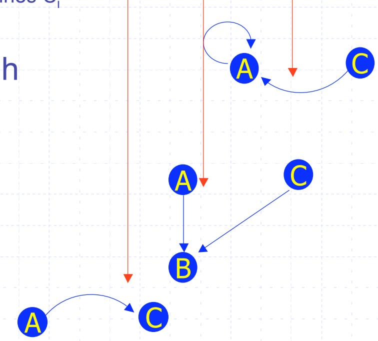
(I) $M(A_i, [A_{i-1}, C_{i-1}]) = H(A_i) + H(A_{i-1}, C_{i-1}) - H(A_i, A_{i-1}, C_{i-1}) = 0.81 + 2 - 2 = 0.81$
$= H(A_i)$, therefore A_{i-1} and C_{i-1} determine A_i

(II) $M(B_i, [A_{i-1}, C_{i-1}]) = H(B_i) + H(A_{i-1}, C_{i-1}) - H(B_i, A_{i-1}, C_{i-1}) = 0.81 + 2 - 2 = 0.81$
$= H(B_i)$, therefore A_{i-1} and C_{i-1} determine B_i

(III) $M(C_i, A_{i-1}) = H(C_i) + H(A_{i-1}) - H(C_i, A_{i-1}) = 1 + 1 - 1 = 1$
$= H(C_i)$, therefore A_{i-1} determines C_i

- Use this to determine network graph





31

Step 3: Compute Boolean Functions

- Consider only network dependencies

A_{i-1}	C_{i-1}	A_i
0	0	0
0	1	1
1	0	1
1	1	1

$A_i = A_{i-1}$ OR C_{i-1}

A_{i-1}	C_{i-1}	B_i
0	0	0
0	1	0
1	0	0
1	1	1

$B_i = A_{i-1}$ AND C_{i-1}

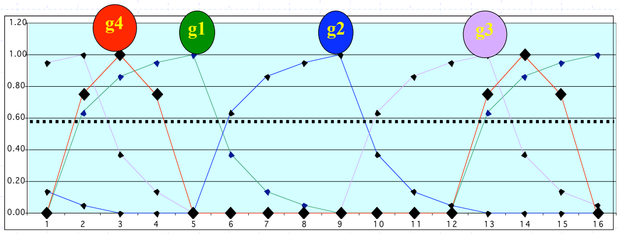
A_{i-1}	C_i
0	1
1	0

$C_i =$ NOT A_{i-1}

32

Example

- Consider the following expression scenario of 4 genes:



g_1	g_2	g_3	g_4	f_1	f_2	f_3	f_4
0	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	0	0	1	1	0	0	0
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0

- A threshold at 0.6 yields: 0010→1011→1001→1000→0100→0010
  - Note: the intermediate state 0000 of the transition 1000=>0000=>0100 is ignored
- The transition map is partial
  - Can admit multiple Boolean net models
  - Exercise: find 2 distinct Boolean net models
- Exercise: use REVEAL to compute a network model

Computing A Partial Dependency Graph

g_1	g_2	g_3	g_4	f_1	f_2	f_3	f_4
0	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	0	0	1	1	0	0	0
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0

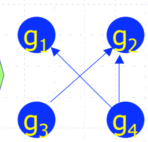
f_1 depends on g_4

g_1	g_2	g_3	g_4	f_1	f_2	f_3	f_4
0	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	0	0	1	1	0	0	0
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0

f_2 depends on g_4

g_1	g_2	g_3	g_4	f_1	f_2	f_3	f_4
0	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	0	0	1	1	0	0	0
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0

f_4 depends on g_3



g_1	g_2	g_3	g_4	f_1	f_2	f_3	f_4
0	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	0	0	1	1	0	0	0
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0

g_1	g_2	g_3	g_4	f_1
0	0	1	0	1
1	0	1	1	1
1	0	0	1	1
1	0	0	0	0
0	1	0	0	0

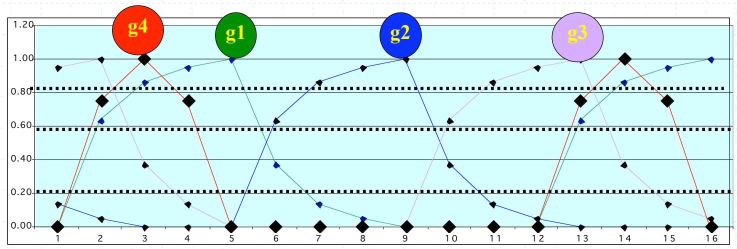
$f_1 = g_3 \vee g_4$

$f_1 = (\neg g_1 \vee g_4) \wedge (\neg g_2)$

Dependency means that g_4 must appear in any expression of f_1;
For a partial map f_1 may require other genes without depending on them.

Sensitivity Considerations & Noisy Maps

- Consider again the 4 genes example



- Different thresholds yield different Boolean dynamics:
  - Tr1/2/3/4=0.6: 0010→1011→1001→1000→0100→0010…
  - Tr1/2/3=0.2, Tr4=0.8: 0010 →1010→1011→1000→1100→0100 →01100→0010
  - Tr1/2/3/4=0.8: 0010 →0011→1000→0000→0100→0000→0010… (non-deterministic)

Research Questions

- Extend the intuitive algorithm to handle partial noisy maps
- Extend REVEAL to handle partial noisy maps
- ?Probabilistic Boolean net models?
  - ?Max likelihood training…EM…?
- ?SVM based models… Boolean kernel machines…?

Final Notes

How Good Are Boolean Models?

Advantages

- Provide good qualitative interpretation of regulation
- Particularly important for switching behaviors
  - Phage lysis...sporulation…Drosophila patterns..
  - Such systems are “robust” wrt exact expression values
- Useful connection with evolutionary behaviors

Disadvantages

- Boolean abstraction is poor fit to real expression data
- Cannot model important features:
  - Amplification of a signal; subtraction and addition of signals
  - Handling smoothly varying environmental parameter (e.g. temperature, nutrients)
  - Temporal performance behavior (e.g. cell cycle period)
  - Negative feedback control (Boolean model oscillates vs. stabilize)

A Variety of Regulatory Network Models

- Finite-field models: $X(t+1)=p(X)$
  - p is a polynomial over finite field
  - Generalizes the Boolean model
- Differential equations models: describe $dX/dt=f(X)$
  - f describes non-linear control of change by neighbors
- Linear model: $X(t+1)=W X+ B$
  - W is a weight matrix; linear approximation near steady state
- Neural network models: $x_i(t)=\sigma(WX_{Neighbors(i)} +B)$
  - Sigmoid non-linearity can be trained through gradient algorithm
  - Comes with a learning algorithm
- Bayesian network models…

39