Computational Systems Biology

… Biology X – Lecture 10…

Bud Mishra

Professor of Computer Science, Mathematics, & Cell Biology

Wittgenstein: Brown Book

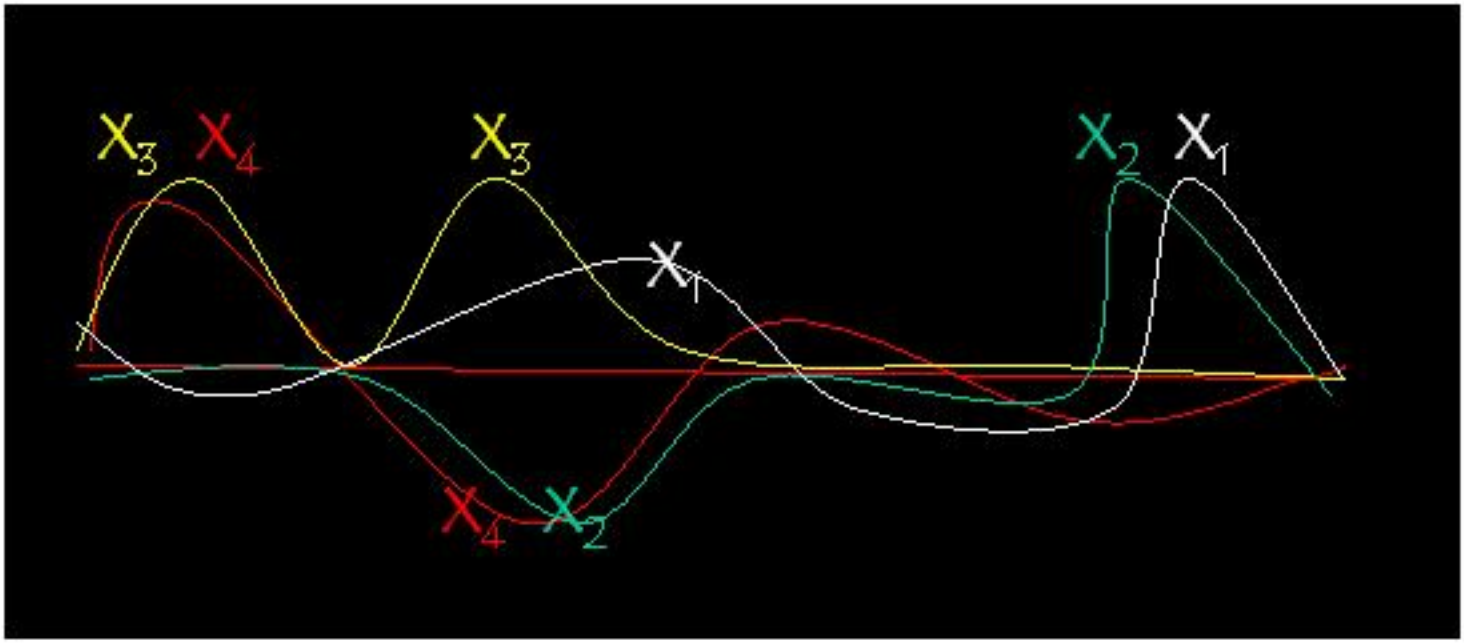
- "Augustine, in describing his learning of language, says that he was taught to speak by learning the names of things…
- "Suppose a man describes a game of chess, without mentioning the existence and operations of the pawns. His description of the game as a natural phenomenon will be incomplete. On the other hand, we may say that he has completely described a simpler game."

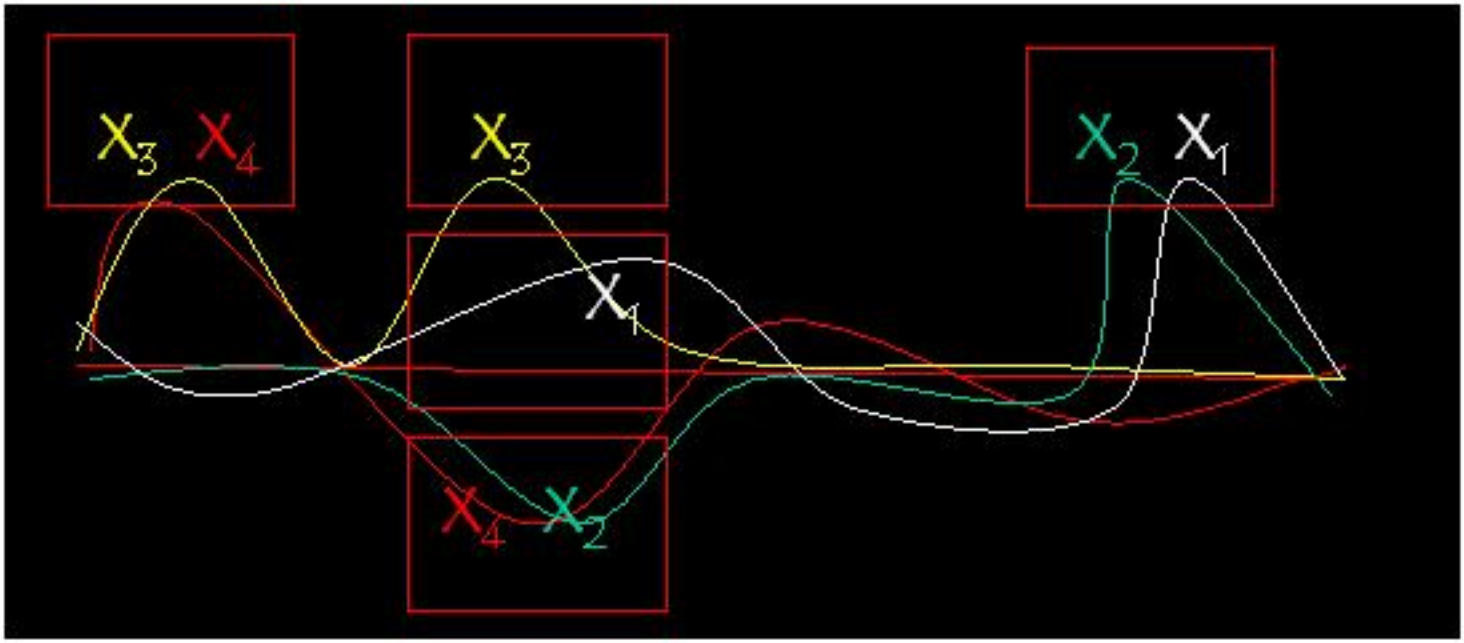
X_1 Female, Sings, Overweight

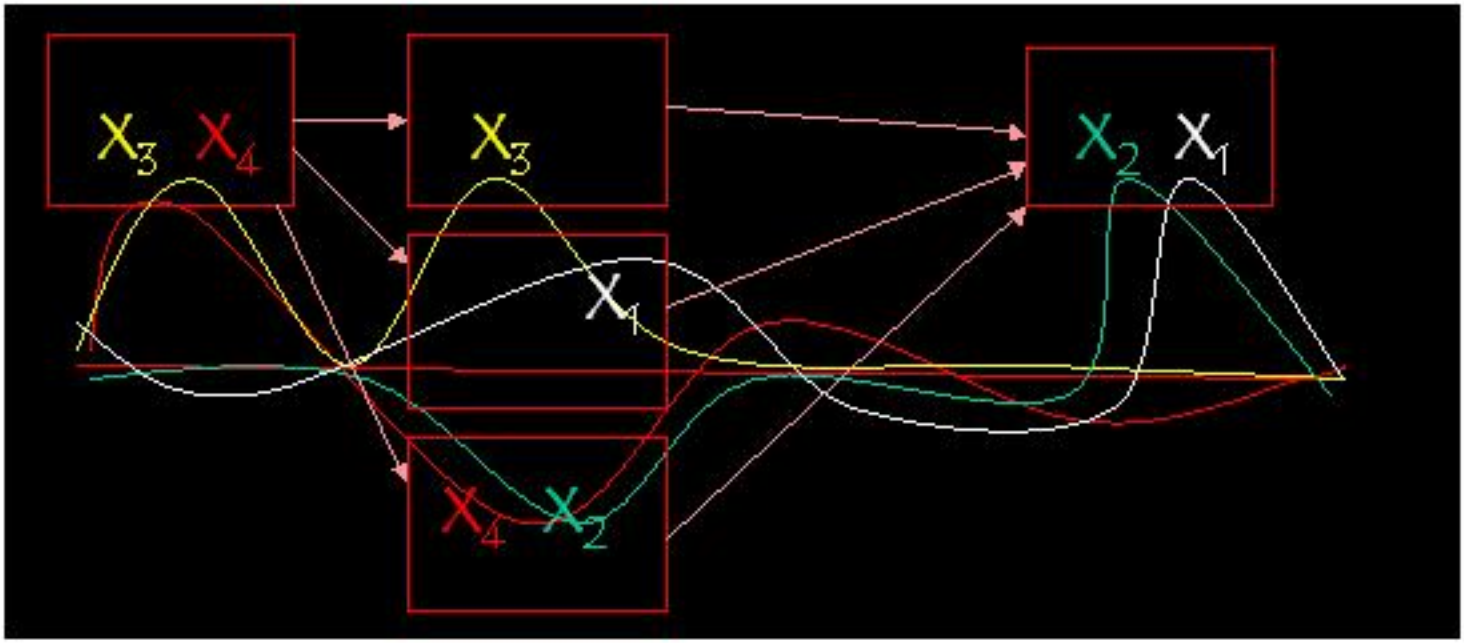
X_2 Male, Talks, Thin

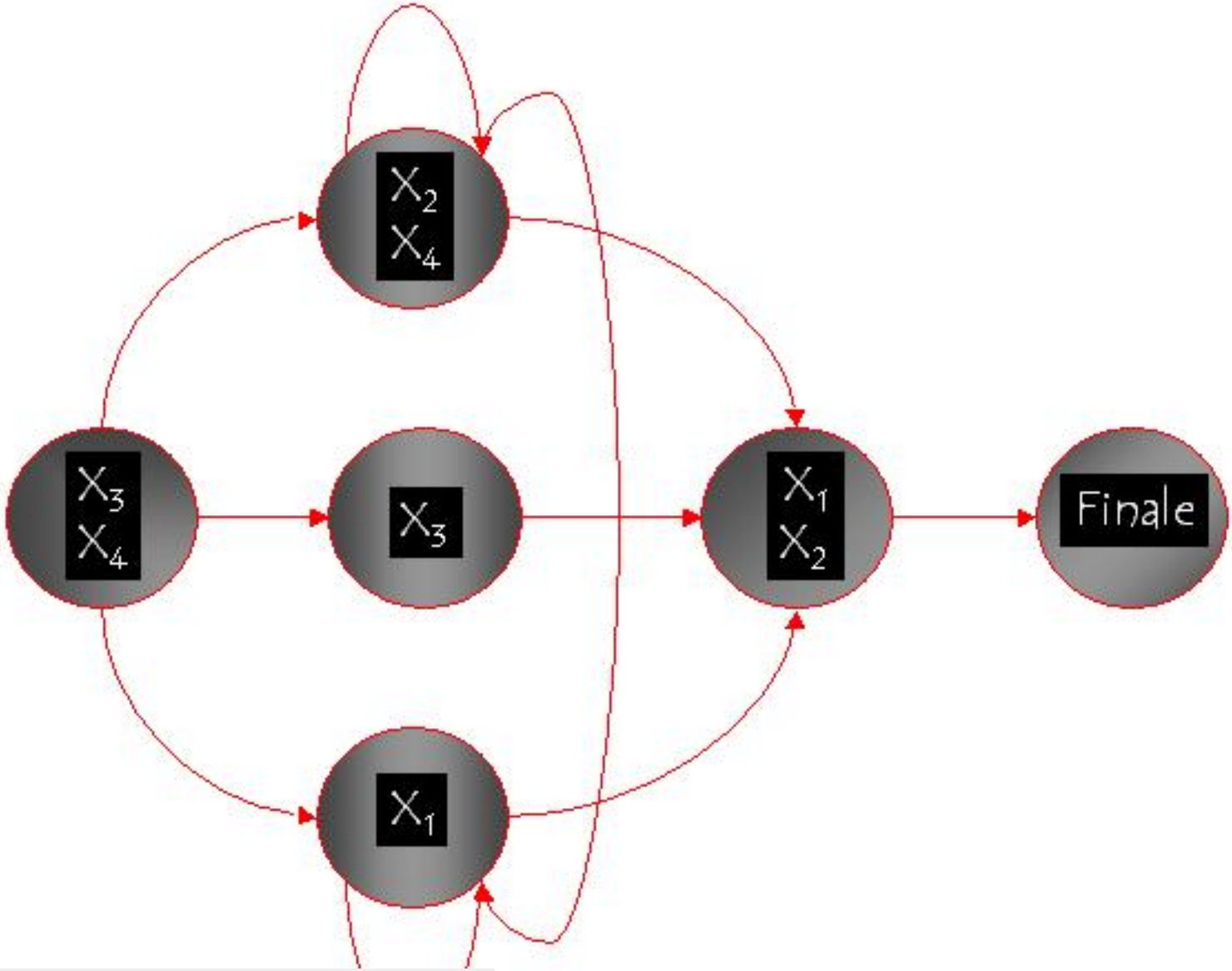
X_3 Male, Sings, Overweight

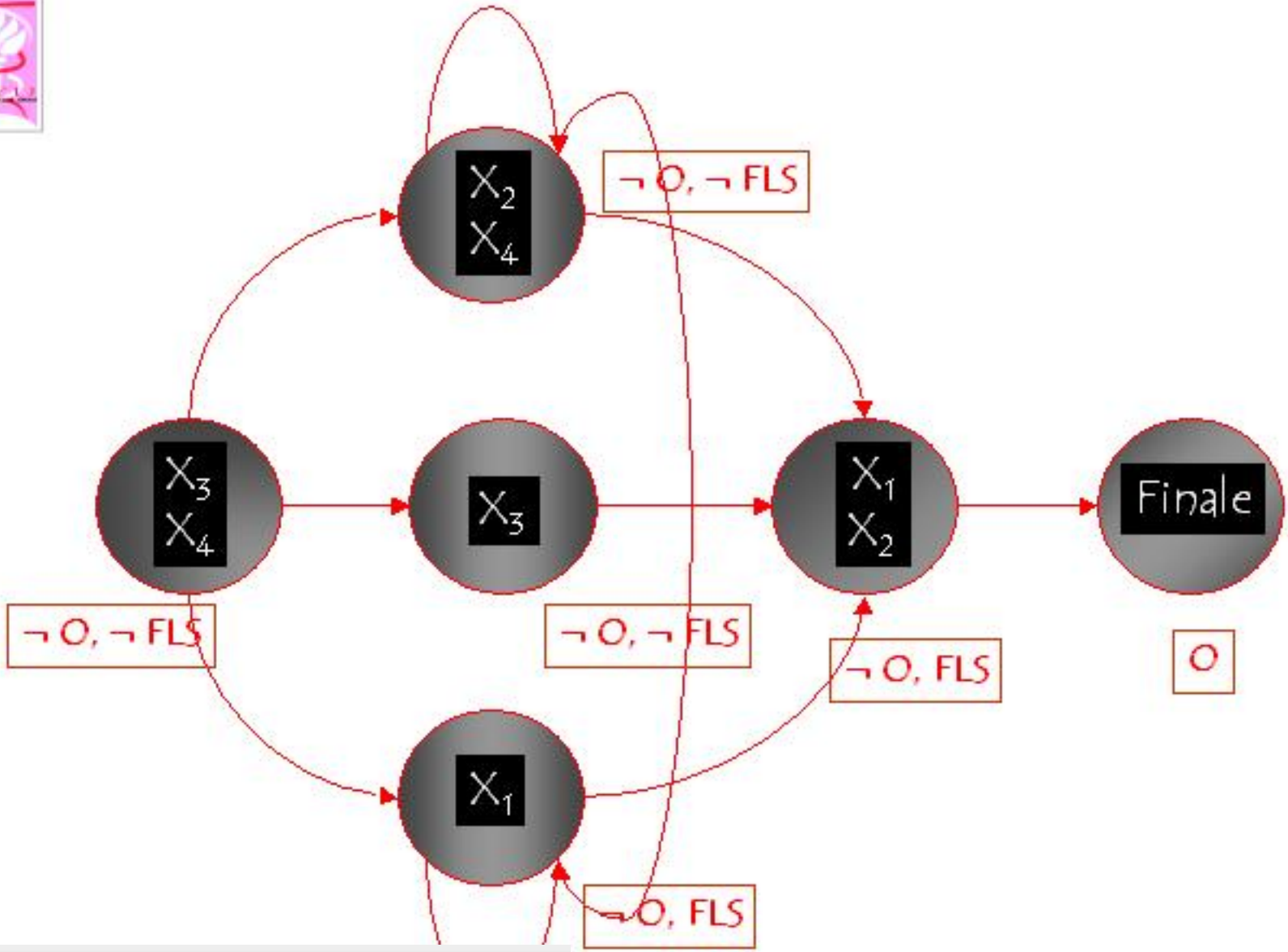
X_4 Female, Sings, Underweight

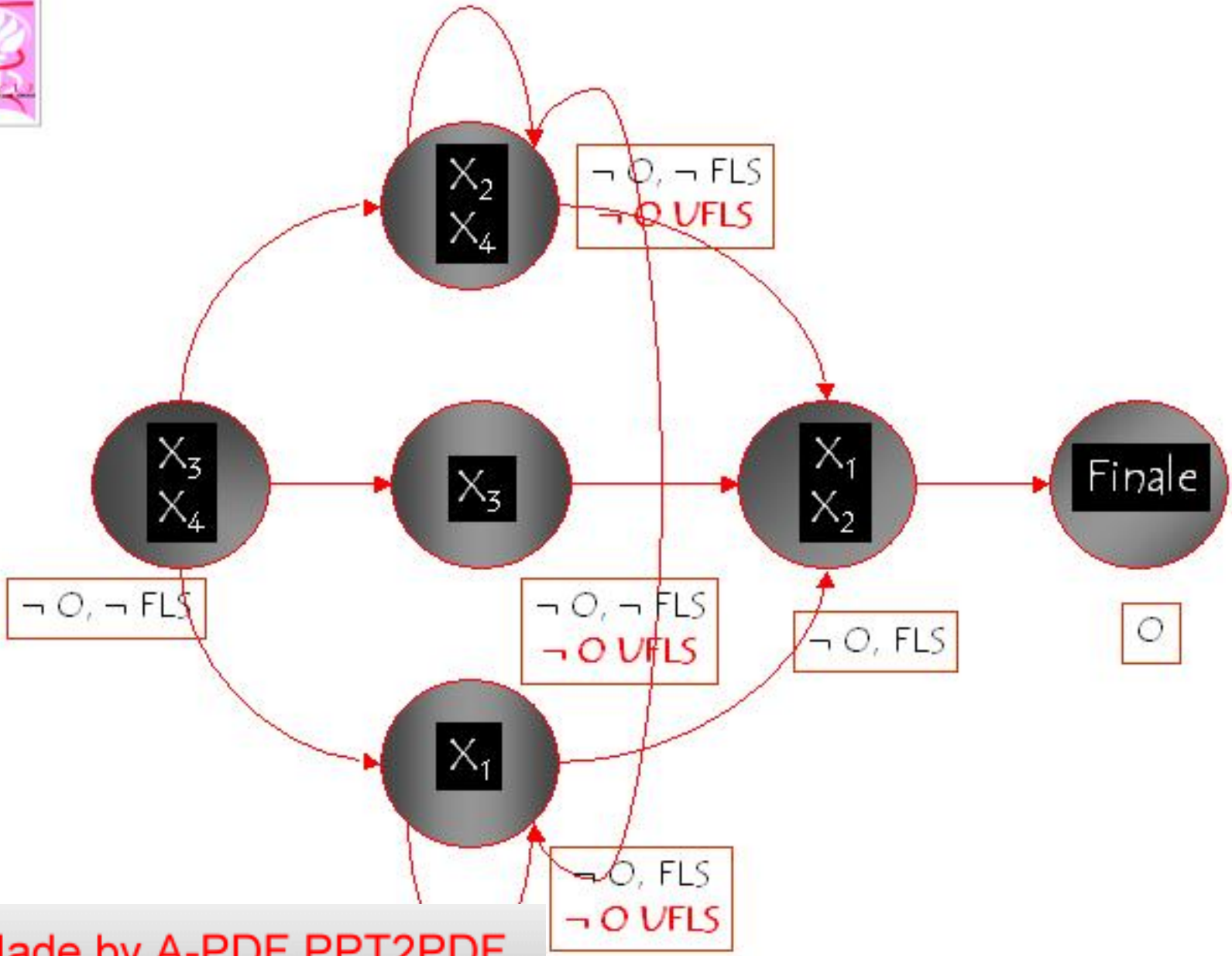


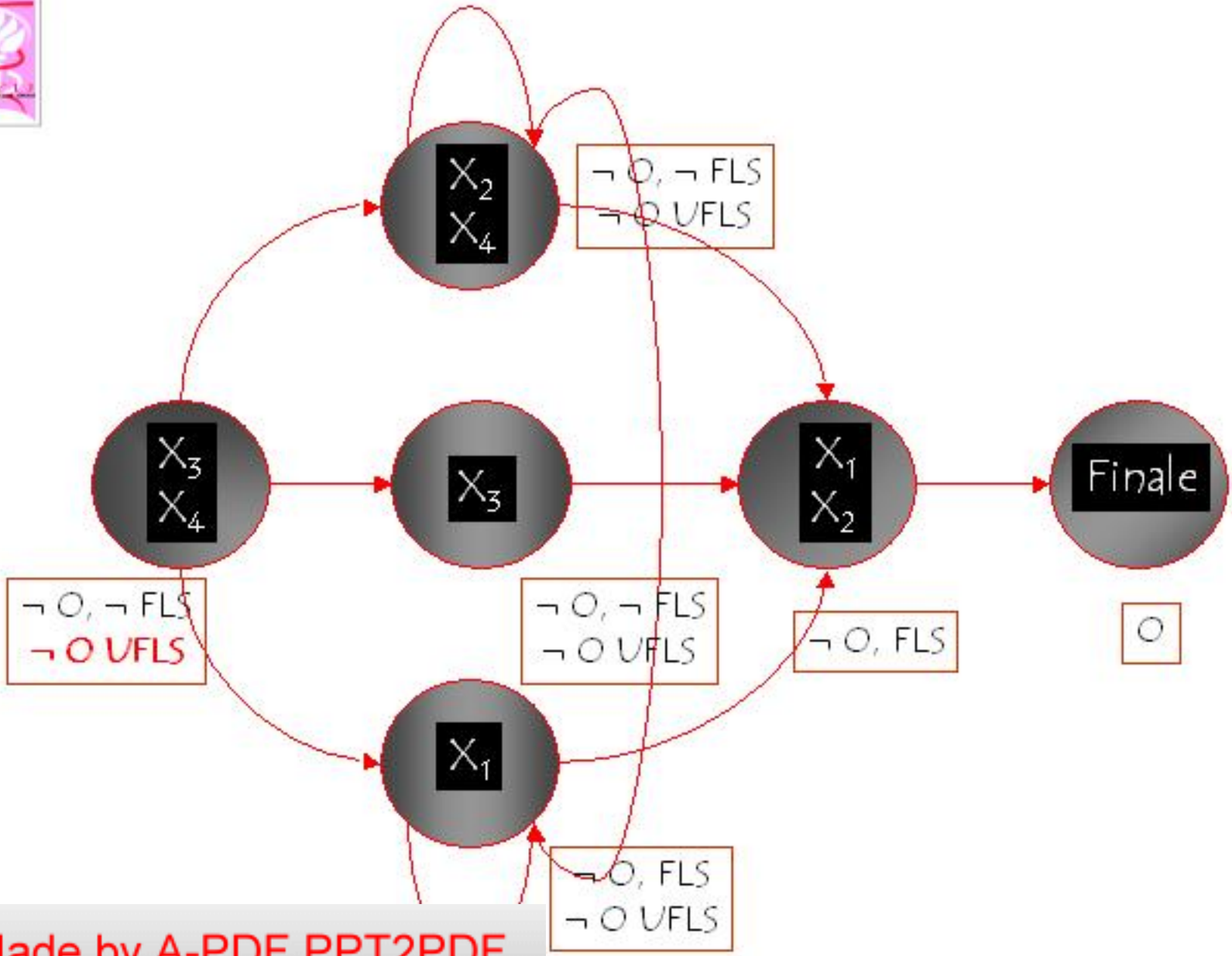


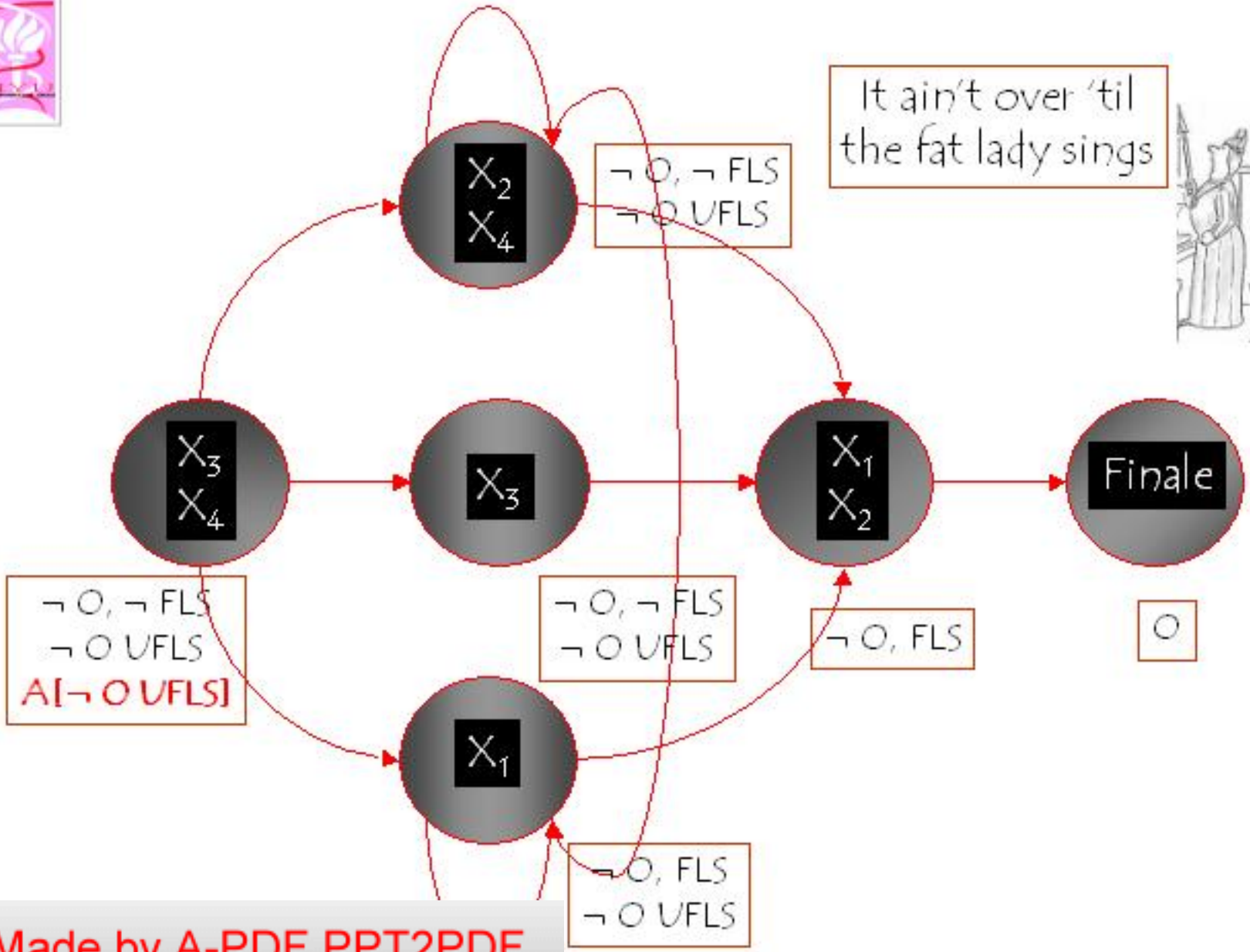










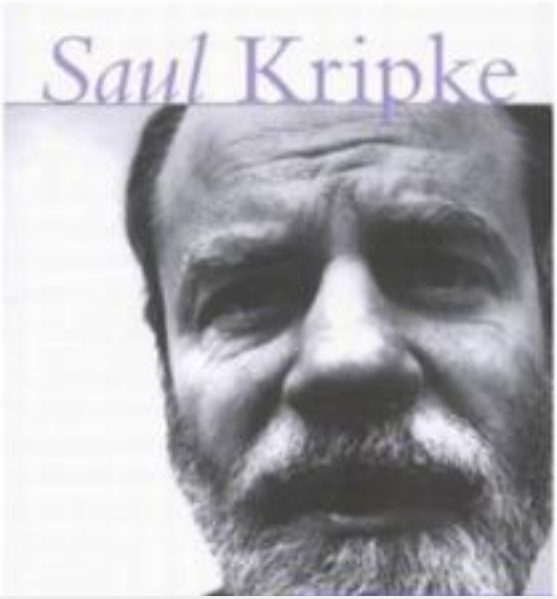


Task: Infer Temporal Invariants

- Select a Language of Discourse
- Formally encode the behavior of the system
- Formally encode the properties of interest
- Automate the process of checking if the formal model of the system satisfies the formally encoded properties

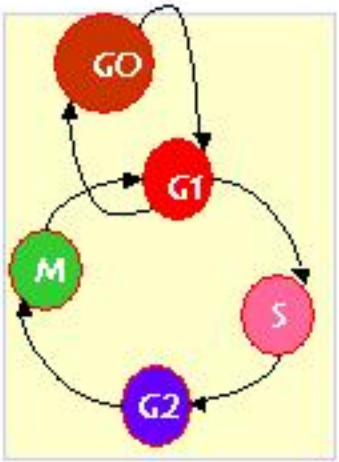
Model Checking

Kripke Structure



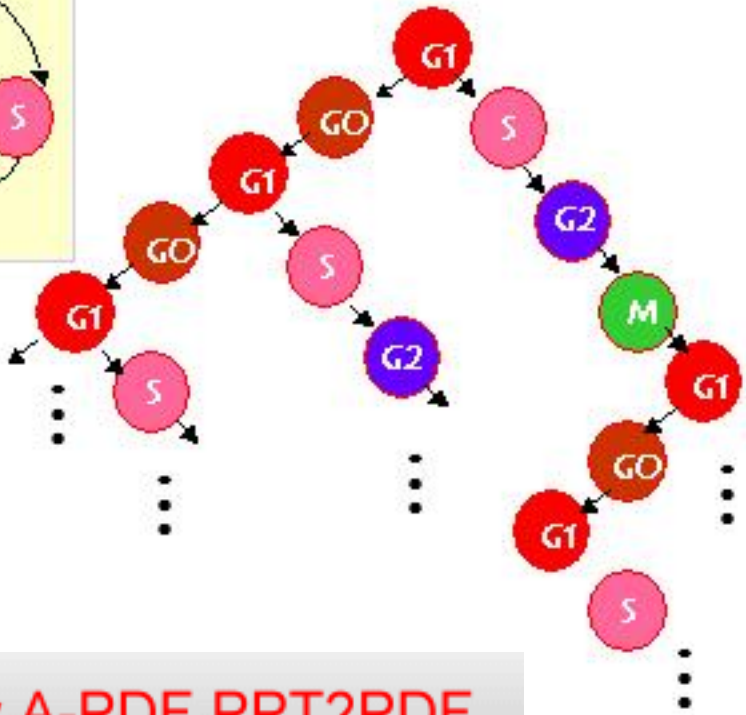
- Formal Encoding of a Dynamical System:
- Simple and intuitive pictorial representation of the behavior of a complex system
  - A **Graph** with **nodes** representing **system states** labeled with information true at that state
  - The **edges** represent **system transitions** as the result of some action

Computation Tree



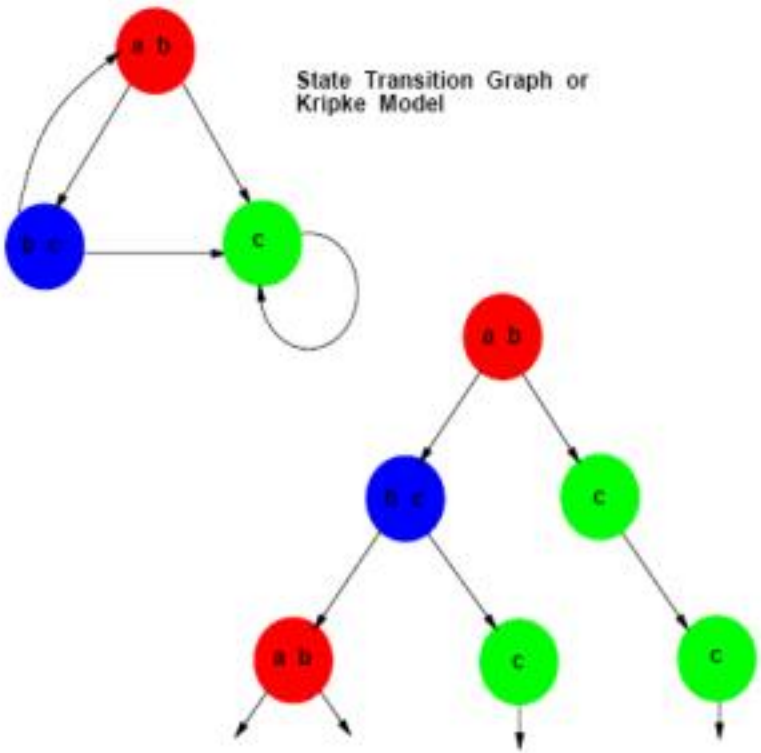
[a]

[b]



- Finite set of states; Some are initial states
- **Total** transition relation: every state has at least one next state i.e. infinite paths
- There is a set of basic environmental variables or features ("atomic propositions")
- In each state, some atomic propositions are true

Discrete Models



- Labeled Finite State Transition Systems
- Formally, a Kripke structure is a triple $M = \langle S, R, L \rangle$, where
- S is the set of **states**,
- $R \subseteq S \times S$ is the **transition relation**, and
- $L : S \rightarrow \mathcal{P}(AP)$ gives **the set of atomic propositions true in each state.**
- We assume that R is total (i.e., for all states $s \in S$ there exists a state $s' \in S$ such that $(s, s') \in R$).

Model Checking

- A path in M is an infinite sequence of states, $\pi = s_0, s_1, \ldots$ such that for $i \geq 0$, $(s_i, s_{i+1}) \in R$.
- We write π^i to denote the suffix of π starting at s_i.
- Unless otherwise stated, all of our results apply only to finite Kripke structures.

The Logic CTL*

- The computation tree logic CTL* combines both branching-time and linear-time operators.
- In this logic a path quantifier can prefix an assertion composed of arbitrary combinations of the usual linear-time operators.

1. Path quantifier:
   - A—"for every path"
   - E—"there exists a path"
2. Linear-time operators:
   - Xp—p holds next time.
   - Fp—p holds sometime in the future
   - Gp—p holds globally in the future
   - pUq—p holds until q holds

Path Formulas and State Formulas

- The syntax of **state formulas** is given by the following rules:
  - If $p \in AP$, then p is a state formula.
  - If f and g are state formulas, then $\neg f$ and $f \vee g$ are state formulas.
  - If f is a path formula, then $E(f)$ is a state formula.
- Two additional rules are needed to specify the syntax of path formulas:
  - If f is a state formula, then f is also a **path formula**.
  - If f and g are path formulas, then $\neg f$, $f \vee g$, Xf, and $(f \mathrm{U} g)$ are path formulas.

State Formulas (Cont.)

- If f is a **state formula**, the notation $M, s \models f$ means that f holds at state s in the Kripke structure M.
- Assume f_1 and f_2 are state formulas and g is a path formula. The relation $M, s \models f$ is defined inductively as follows:

1. $s \models p \quad \Leftrightarrow \quad p \in L(s)$.
2. $s \models \neg f_1 \quad \Leftrightarrow \quad s \not\models f_1$.
3. $s \models f_1 \vee f_2 \quad \Leftrightarrow \quad s \models f_1$ or $s \models f_2$.
4. $s \models E(g) \quad \Leftrightarrow \quad$ there exists a path π starting with s such that $\pi \models g$.

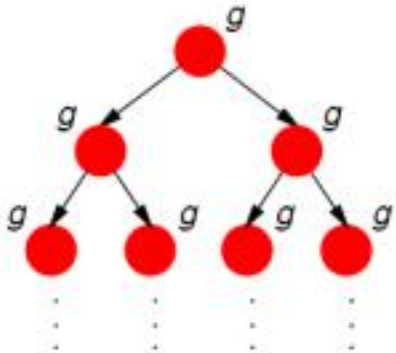
Path Formulas (Cont.)

- If f is a **path formula**, $M, \pi \models f$ means that f holds along path in Kripke structure M.
- Assume g_1 and g_2 are path formulas and f is a state formula. The relation $M, \pi \models f$ is defined inductively as follows:

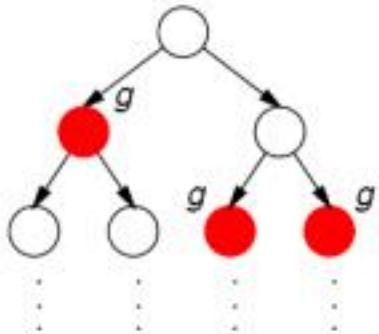
1. $\pi \models f \quad \Leftrightarrow \quad$ s is the 1st state of π and $s \models f$.
2. $\pi \models \neg g_1 \quad \Leftrightarrow \quad \pi \not\models g_1$.
3. $\pi \models g_1 \vee g_2 \quad \Leftrightarrow \quad \pi \models g_1$ or $\pi \models g_2$.
4. $\pi \models X g_1 \quad \Leftrightarrow \quad \pi^1 \models g_1$.
5. $\pi \models (g_1 \, U \, g_2) \Leftrightarrow$ there exists a $k \geq 0$ such that $\pi^k \models g_2$ and for $0 \leq j < k$, $\pi^j \models g_1$.

Basic CTL Operators

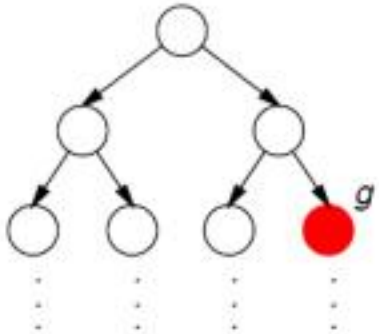
- The four most widely used CTL operators are illustrated below. Each computation tree has the state s_0 as its root.



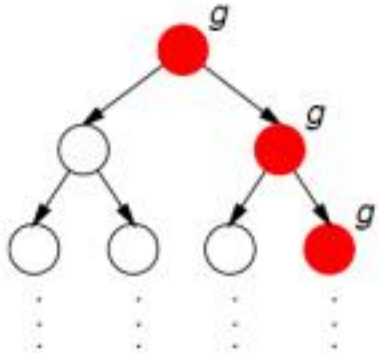
$M, s_0 \models \mathbf{AG}\, g$



$M, s_0 \models \mathbf{AF}\, g$



g



$M, s_0 \models \mathbf{EG}\, g$

S-System

Definition 1 (S-system). *An S-system is a quadruple* $S = (DV, IV, DE, C)$ *where:*

- $DV = \{X_1, \ldots, X_n\}$ *is a finite non empty set of* dependent variables *ranging over the domains* $D_1, \ldots, D_n$, *respectively;*
- $IV = \{X_{n+1}, \ldots, X_{n+m}\}$ *is a finite set of* independent variables *ranging over the domains* $D_{n+1}, \ldots, D_{n+m}$, *respectively;*
- DE *is a set of* differential equations, *one for each dependent variable, of the form*

$$\dot{X}_i = \alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}}$$

  with $\alpha_i, \beta_i \geq 0$ *called* rate constants;
- C *is a set of* algebraic constraints *of the form*

$$C_j(X_1, \ldots, X_{n+m}) = \sum (\gamma_j \prod_{k=1}^{n+m} X_k^{f_{jk}}) = 0$$

  with γ_j *called* rate constants

Trace

Definition 2 (Trace). *Let $S = (DV, IV, DE, C)$ be an S-system. Let $\vec{f}(t) = \langle f_1(t), \ldots, f_{n+m}(t) \rangle$ be a (approximated) solution for the S-system S in the time interval $[t_0, t_f]$ starting with initial values $\vec{X}(t_0)$ in t_0. Let $s > 0$ be a time step such that $t_f = t_0 + j * s$. The sequence of vectors of values*

$$tr(S, t_0, \vec{X}(t_0), s, t_f) = \langle \vec{f}(t_0), \vec{f}(t_0 + s), \ldots, \vec{f}(t_0 + (j-1) * s), \vec{f}(t_0 + j * s) \rangle$$

is a trace *of S. When we are not interested in the parameters defining the trace we use the notation tr.*

Kripke Structure

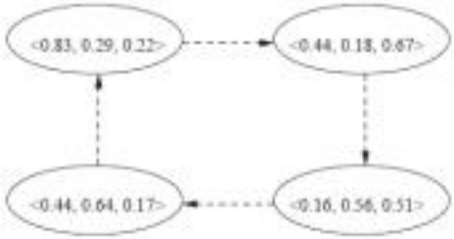
Definition 4 (S-system Automaton). *Let S be an S-system and Tr be a set of traces on S. An* S-system automaton *is $\mathcal{A}(S, Tr) = (V, \Delta, I, F)$, where*

- $V = \{\vec{v} = \langle v_1, \ldots, v_{n+m} \rangle \mid \exists tr \in Tr : \vec{v} \text{ is in } tr\} \subseteq D_1 \times \ldots \times D_{n+m}$ *is the set of states;*
- $\Delta = \{(\vec{v}, \vec{w}) \mid \exists tr \in Tr : \vec{v}, \vec{w} \text{ are consecutive in } tr\}$ *is the transition relation;*
- $I = \{\vec{v} \mid \exists tr \in Tr : \vec{v} \text{ is initial in } tr\} \subseteq V$ *is the set of initial states;*
- $F = \{\vec{v} \mid \exists tr \in Tr : \vec{v} \text{ is final in } tr\} \subseteq V$ *is the set of final states.*

Projection

Definition 5 (Projection). *Let S be an S-system and U be a subset of the set of variables of S. Given a trace $tr = \langle \vec{a}_0, \ldots, \vec{a}_j \rangle$ of S the* projection over U *of tr is the sequence $tr \restriction U = \langle \vec{a}_0 \restriction U, \ldots, \vec{a}_j \restriction U \rangle$. Given a set of traces Tr the* projection over U *of Tr is the set of projected traces $Tr \restriction U = \{tr \restriction U \mid tr \in Tr\}$. The U-*projected S-system automaton *from Tr and S is $\mathcal{A}(S, Tr \restriction U)$.*

$$\dot{X}_1 = X_4 X_3^{-1} - X_1^{0.5}$$
$$\dot{X}_2 = X_5 X_1^{-1} - X_2^{0.578151}$$
$$\dot{X}_3 = X_6 X_2^{-1} - X_3^{0.5}$$



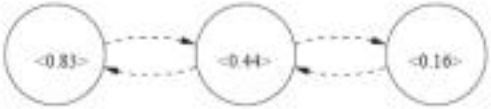
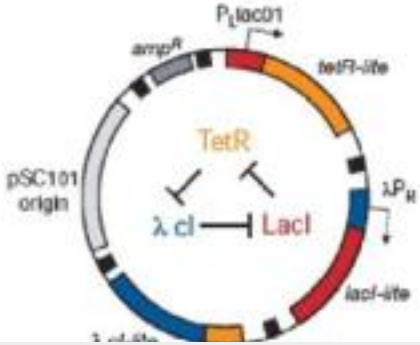


Fig. 1. Repressilator: automaton and projected automaton.



Example

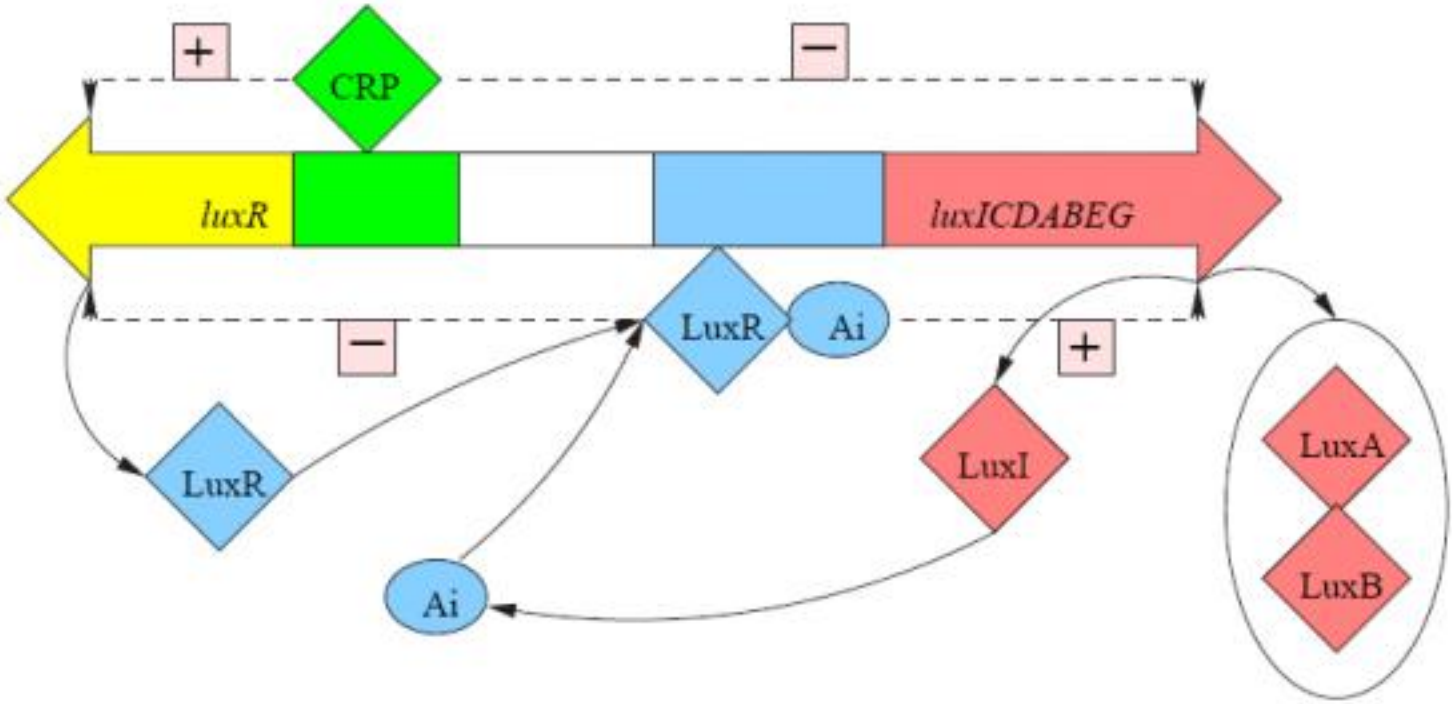


Fig. 5. The *lux* region of *Vibrio fischeri*.

Collapsed Model

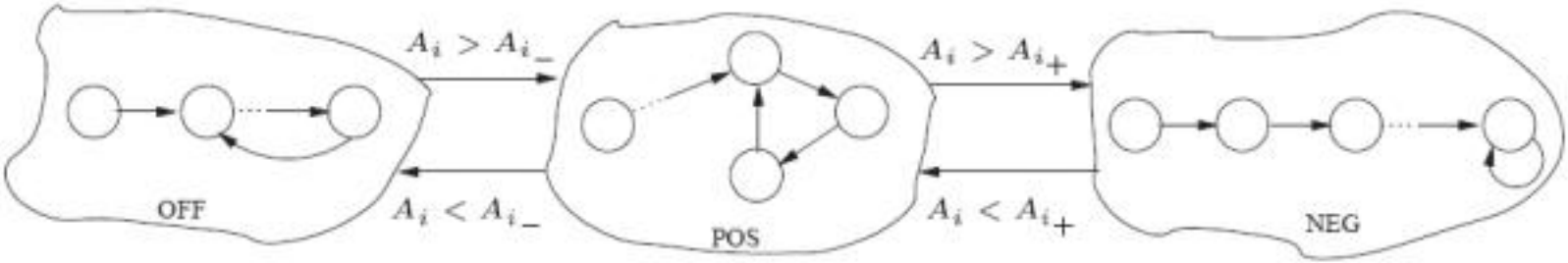


Fig. 7. Strcture of the *Vibrio fischeri* final model.

CTL Model-Checking

- Straight-forward approach: **Recursive descent on the structure of the query formula**
- Label the states with the terms in the formula:
  - Proceed by marking each point with the set of valid sub-formulas
- **"Global" algorithm:**
  - Iterate on the structure of the property, traversing the whole of the model in each step
  - Use fixed point unfolding to interpret Until:

$$\mathbf{E}(\psi_2\,\mathbf{U}^+\,\psi_1) \leftrightarrow \mathbf{E}\,\mathbf{X}(\psi_1 \vee \psi_2 \wedge \mathbf{E}(\psi_2\,\mathbf{U}^+\,\psi_1))$$

$$\mathbf{A}(\psi_2\,\mathbf{U}^+\,\psi_1) \leftrightarrow \mathbf{A}\,\mathbf{X}(\psi_1 \vee \psi_2 \wedge \mathbf{A}(\psi_2\,\mathbf{U}^+\,\psi_1))$$

Naïve CTL Model-Checker

```
procedure CTL_check (Model (U, I, w0), Formula φ) =
    if w0 ∈ eval(φ)
    then print("φ is satisfied at w0 in (U, I)")
    else print("φ not satisfied at w0 in (U, I)");

function eval (Formula φ): Pointset =
    case φ of
        p : return I(p);
        ⊥ : return {};
        (ψ1 → ψ2) : return U \ eval(ψ1) ∪ eval(ψ2);
        E(ψ2 U+ ψ1) :  E1 := eval(ψ1);  E2 := eval(ψ2);  E := {};
            repeat until stabilization
                E := E ∪ {w | (succ(w) ∩ (E1 ∪ (E2 ∩ E))) ≠ {}};
            return E;
        A(ψ2 U+ ψ1) :  E1 := eval(ψ1);  E2 := eval(ψ2);  E := {};
            repeat until stabilization
                E := E ∪ {w | {} ≠ succ(w) ⊆ E1 ∪ (E2 ∩ E)};
            return E;
                                    intset = return {w' | (w, w') ∈ I(≺)};
```

Complexity Comparison

Size of transition system: **n**

Size of temporal logic formula: **m**

- Worst Case Comparison:
  - CTL: **linear** - $O(nm)$
  - LTL: **exponential** - $n\, 2^{O(m)}$
- For an LTL formula that can also be expressed in vCTL, **LTL model-checking can be done in a time linear in the size of the formula**
- LTL is PSPACE complete: Hamiltonian Path problem can be reduced to an LTL Model Checking problem:

$$\mathbf{F}p_1 \wedge \mathbf{F}p_2 \wedge \mathbf{F}p_3 \wedge ..$$

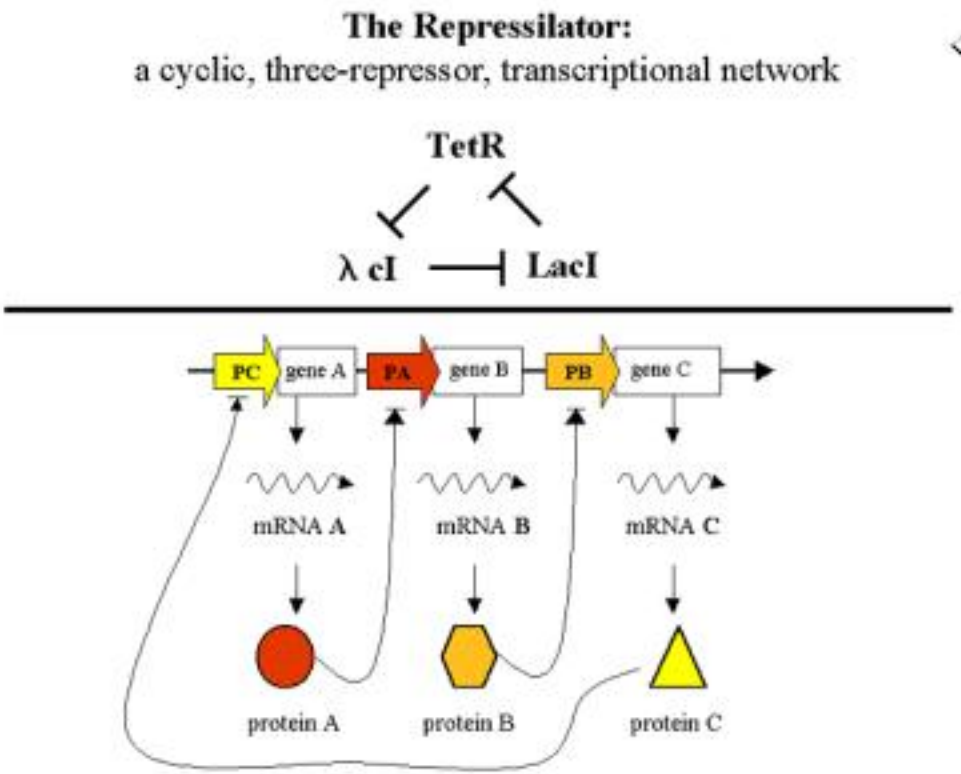
$$\mathbf{G}\,(p_1 \rightarrow \mathbf{XG} \neg p_1) \wedge \mathbf{G}(p_2 \rightarrow \mathbf{XG} \neg p_2) \wedge ....$$

Other Model Checking Algorithms

- LTL Model Checking: Tableu-based…
- CTL* Model Checking: Combine CTL and LTL Model Checkers…
- Symbolic Model Checking
  - Binary Decision Diagram
  - OBDD-based model-checking for CTL
  - Fixed-point Representation
  - Automata-based LTL Model-Checking
- SAT-based Model Checking
- Algorithmic Algebraic Model Checking
- Hierarchical Model Checking

Artificial Gene Networks

An Artificial Clock



- Three proteins:
  - LacI, tetR & λ cI
  - Arranged in a cyclic manner (logically, not necessarily physically) so that the protein product of one gene is rpressor for the next gene.

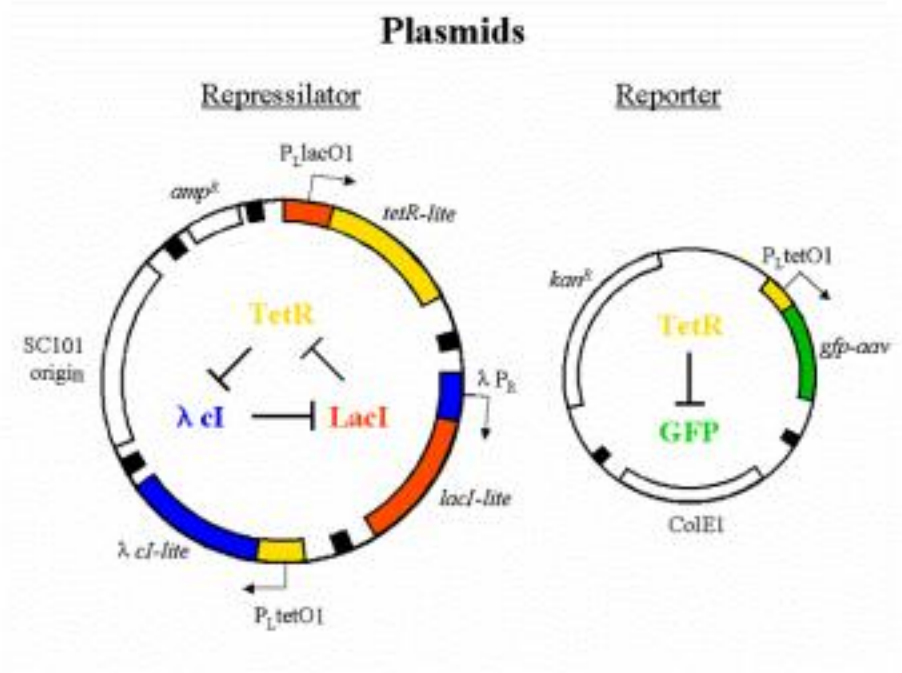
LacI→ ¬ *tetR*; *tetR*→ TetR

TetR→ ¬ λ *cI*; λ *cI* → λ cI

λ cI→ ¬ *lacI*; *lacI*→ LacI

uet et al., Antoniotti et al., Wigler & Mishra

Biological Model

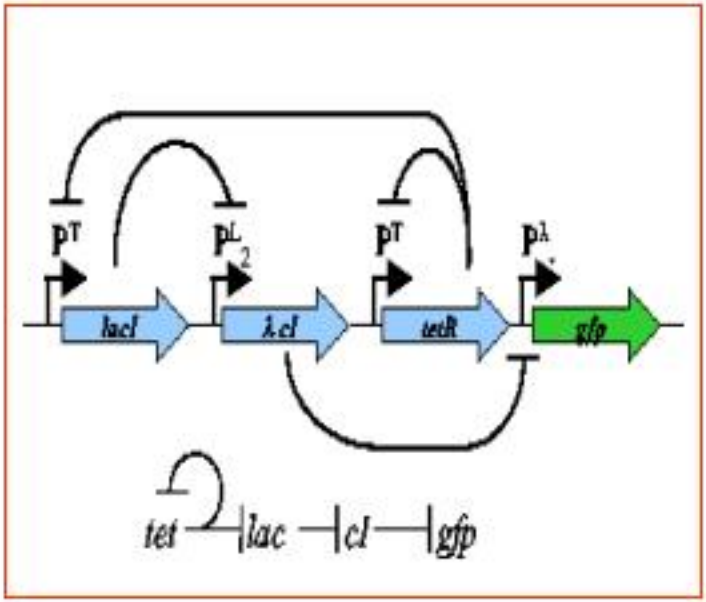


- Standard molecular biology: Construct
  - A low-copy plasmid encoding the repressilator and
  - A compatible higher-copy reporter plasmid containing the tet-repressible promoter PLtetO1 fused to an intermediate stability variant of gfp.

Artificial Gene Network

- Network of interacting biomolecules:
  - "Combinatorial Synthesis of Genetic Networks," Guet et al.
- Design principles:
  - Underlying the functioning of such intercellular networks.
  - Not much progress with quantitative analysis of relatively simple systems..

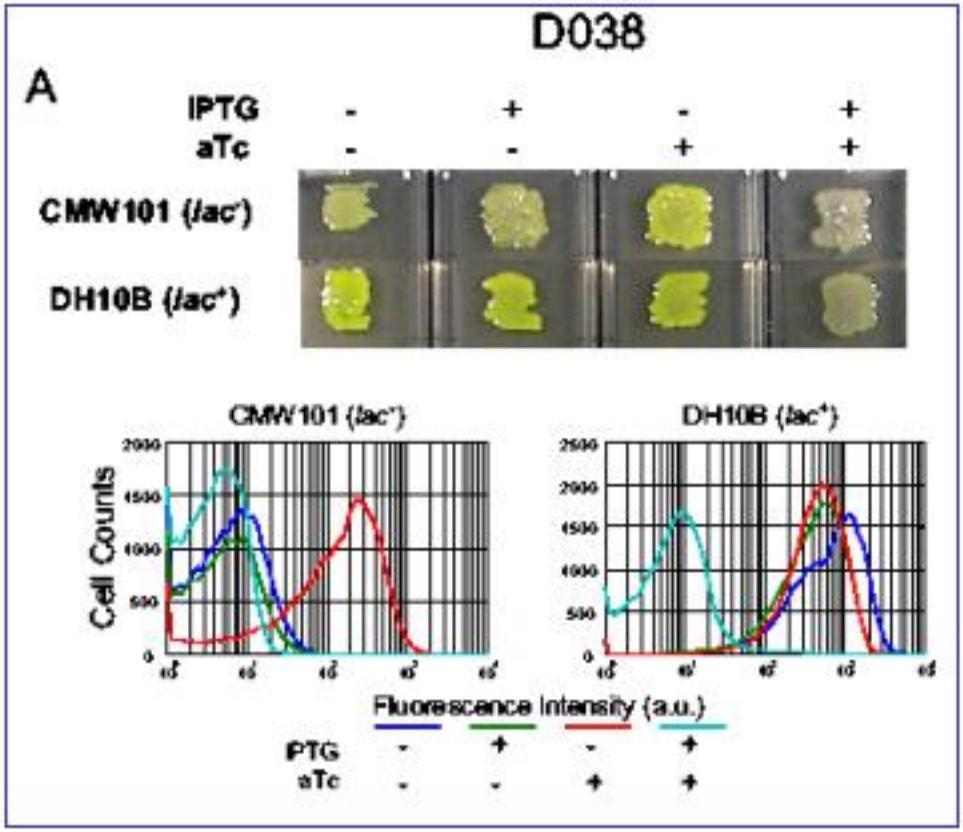
An Example



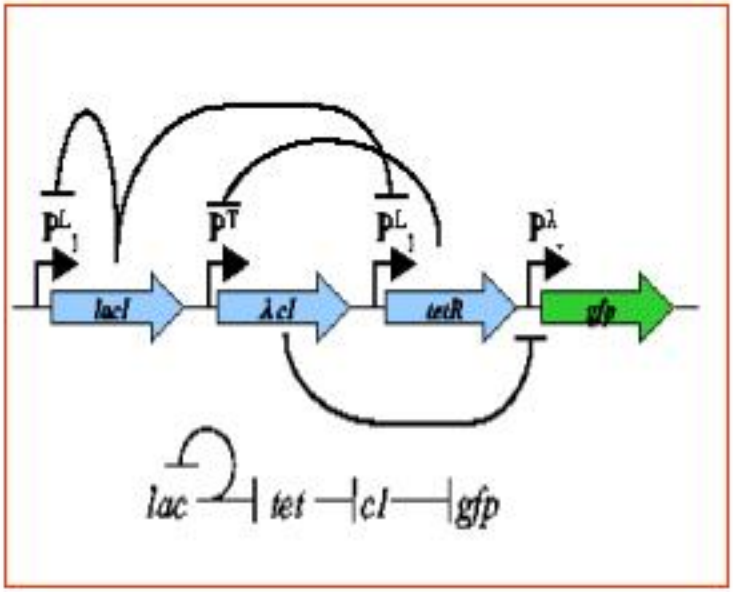
A circuit composed of three genes and three suppressors: D038.

- Four genes: Lac, λ, Tet and GFP.
- Five Operons:
  - Lac-based: PL1, PL2
  - λ CI-based: $P\lambda_-$, $P\lambda_+$
  - Tet-based: PT
- Lac is allosterically suppressible by IPTG
- Tet is allosterically suppressible by aTc
- Total 5^3 = 125 different combinatorial circuits are possible…

Experimental Results



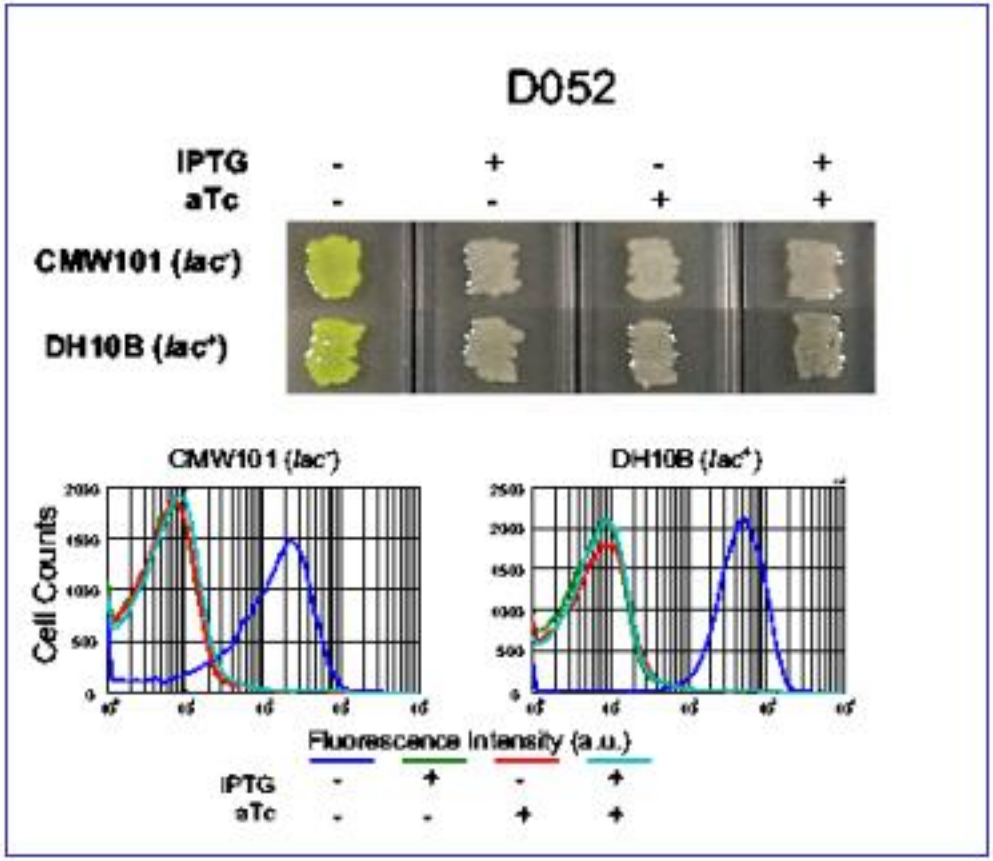
Another Example



Another circuit composed of three genes and three suppressors: D052.

- A different structure:
- Changing the topology changes the circuit behavior…
- The circuit behave differently in the wild-type (Lac_+) and the mutant (Lac_-)…

Experimental Results



ODE Models

- If x denotes a gene and X its corresponding protein, we have the following equation for x's transcription:

$$[\dot{x}] = -[x] + \alpha[\rho + f_x(\theta, [Y], [u_y])]$$

$$\text{where } f_x(\theta, [Y], [u_y]) = \frac{1 + \theta[Y]^n + [u_y]^k}{1 + [Y]^n + [u_y]^k}.$$

- The transcription is activated or repressed by a protein Y and Y itself is modulated by a small molecule u_y.
  - Note that, for small values of $[u_y]$, f_x shows a sharp transition from a value of 1 (when [Y] = 0) to a value of θ (when [Y] = ∞), as Y increases.
  - However, for large values of $[u_y]$, f_x remains at 1 (when [uy] = ∞), thus inactivating the effect of Y.

ODE Models

- Similarly, we have the following equation for X's (corresponding proteins) translation:

$$[\dot{X}] = -\beta([X] - [x]).$$

Example

- For an example circuit $P\lambda_+$-lac-PL1-λ ci-PL1-tet, we can write down in a straightforward manner the corresponding ODE's as shown below:

$$\dot{[lac]} = -[lac] + \alpha\rho + \alpha\frac{1+\theta_a[\lambda\ \mathrm{CI}]^n}{1+[\lambda\ \mathrm{CI}]^n}$$

$$\dot{[\mathrm{LAC}]} = -\beta([\mathrm{LAC}] - [lac])$$

$$\dot{[gfp]} = -[gfp] + \alpha\rho + \alpha\frac{1+\theta_s[\lambda\ \mathrm{CI}]^n}{1+[\lambda\ \mathrm{CI}]^n}$$

$$\dot{[\mathrm{GFP}]} = -\beta([\mathrm{GFP}] - [gfp])$$

$$\dot{[tet]} = -[tet] + \alpha\rho + \alpha\frac{1+\theta_s[\mathrm{LAC}]^n + [\mathrm{IPTG}]^k}{1+[\mathrm{LAC}]^n + [\mathrm{IPTG}]^k}$$

$$\dot{[\mathrm{TET}]} = -\beta([\mathrm{TET}] - [tet])$$

$$\dot{[\lambda\ ci]} = -[\lambda\ ci] + \alpha\rho + \alpha\frac{1+\theta_s[\mathrm{LAC}]^n + [\mathrm{IPTG}]^k}{1+[\mathrm{LAC}]^n + [\mathrm{IPTG}]^k}$$

$$\dot{[\lambda\ \mathrm{CI}]} = -\beta([\lambda\ \mathrm{CI}] - [\lambda\ ci])$$

Application

- Find a circuit that generates an "OR" Gate:

```
eventually(IPTG = 0 and aTc = 0 ==> eventually(always(low(c))))
and eventually(IPTG = 0 and aTc = 3 ==> eventually(always(high(c))))
and eventually(IPTG = 3 and aTc = 0 ==> eventually(always(high(c))))
and eventually(IPTG = 3 and aTc = 3 ==> eventually(always(high(c)))).
```

- Examples:

Boolean Function	*Circuit*
$\neg$ *IPTG*	51 52 56 57 76 77 78 79 80 81 82 83 85
aTc	14 39 64 89 114
aTc $\rightarrow$ *IPTG*	61 62

Table 2

The classification of potential Boolean circuits given a threshold of 1.3 μMol. Each number denotes one of the circuits described in [7].

Applications

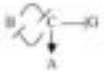
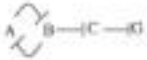
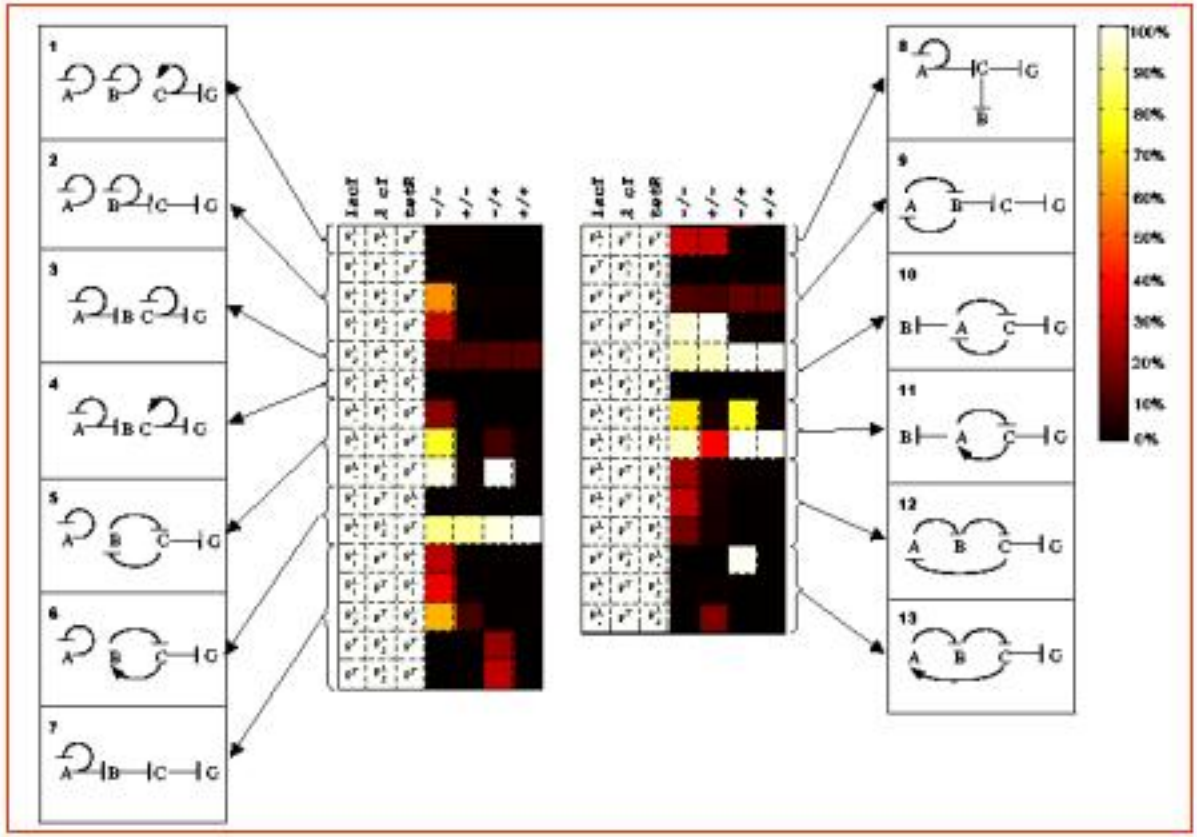
Circuit	Function	Comment
	$\neg\ IPTG$	Circuit 85 $\langle P\lambda_-, PL2, P\lambda_+\rangle$
	aTc	Circuit 114 $\langle P\lambda_+, PT, P\lambda_-\rangle$
	$aTc \Rightarrow IPTG$	Circuit 61 $\langle PT, PT, PL1\rangle$

Table 3

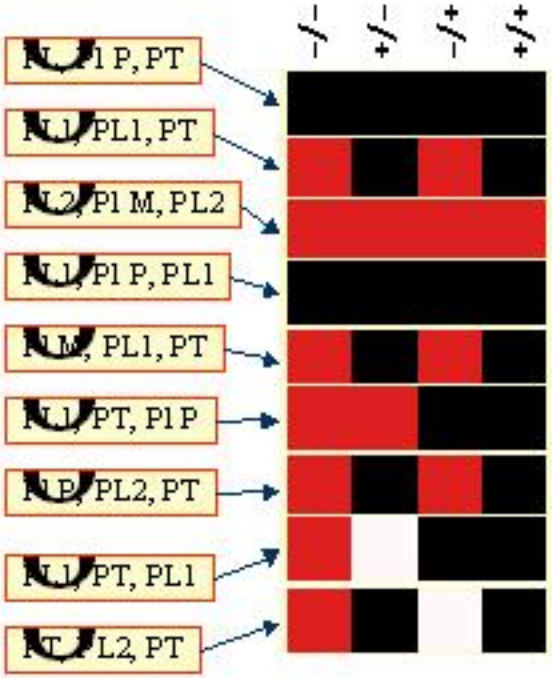
Some of the circuits implementing the logic-combinatorial circuits found with threshold parameter equal to 1.3μMol. Again the triple of promoters denotes the structure of the circuit.

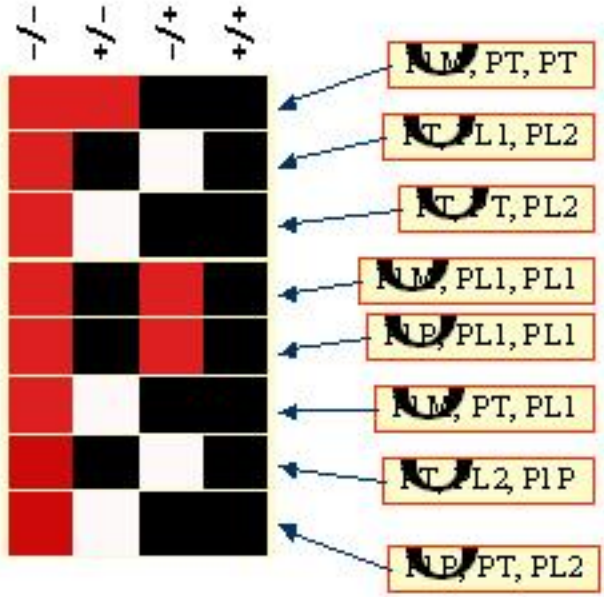
Biological Results



Results

Further Results



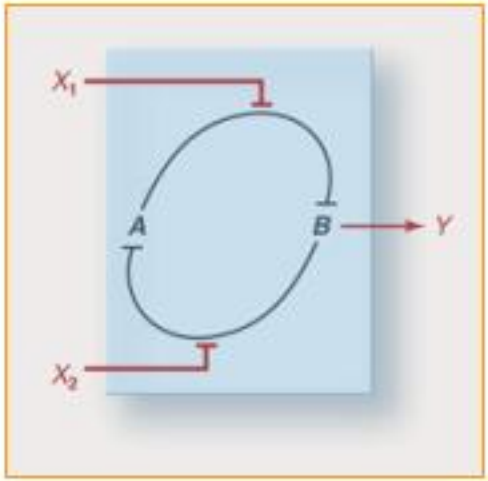


Remaining Questions

- Simulation:
  - Nonlinearity
  - Hybrid Model (Piece-wise linear)
- Stability Analysis
- Reachability Analysis
- Robustness

Metastability

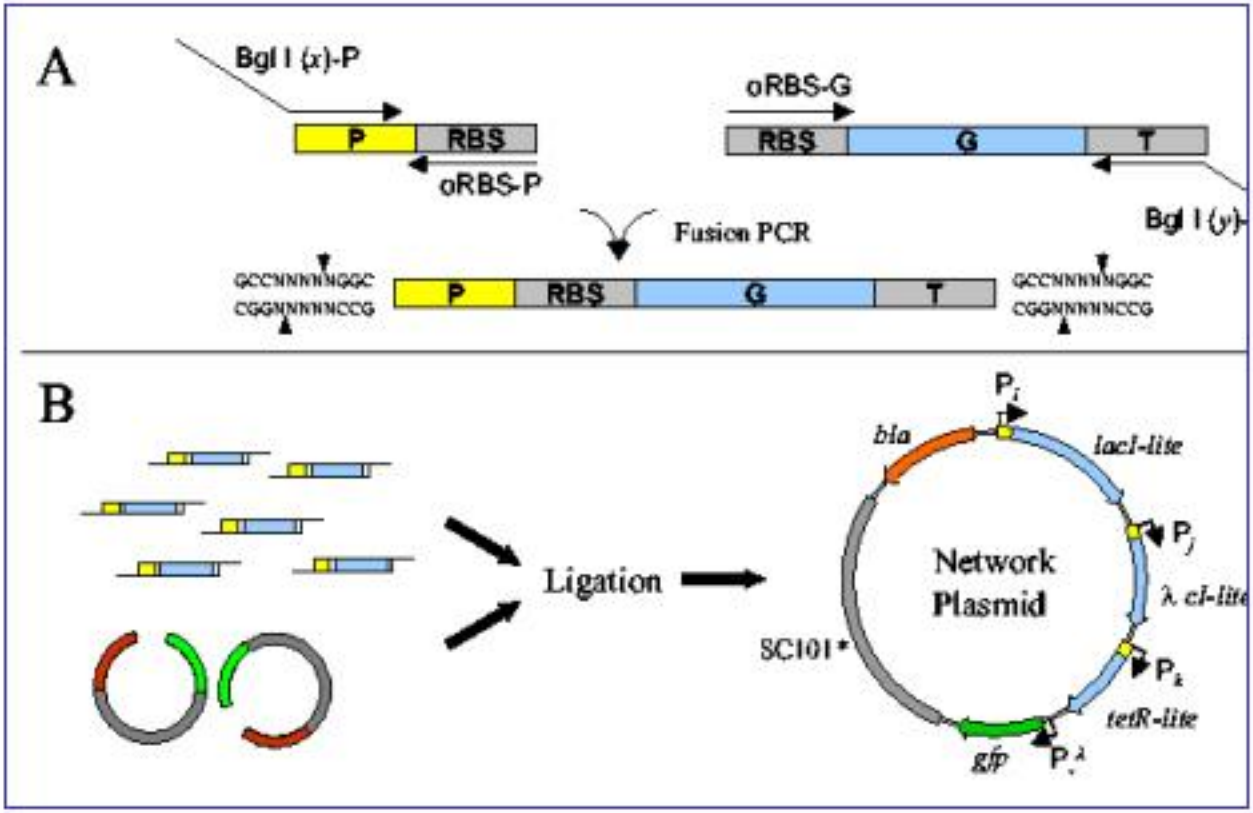
"Feedback Dyad"
A simple metastable biological circuit.



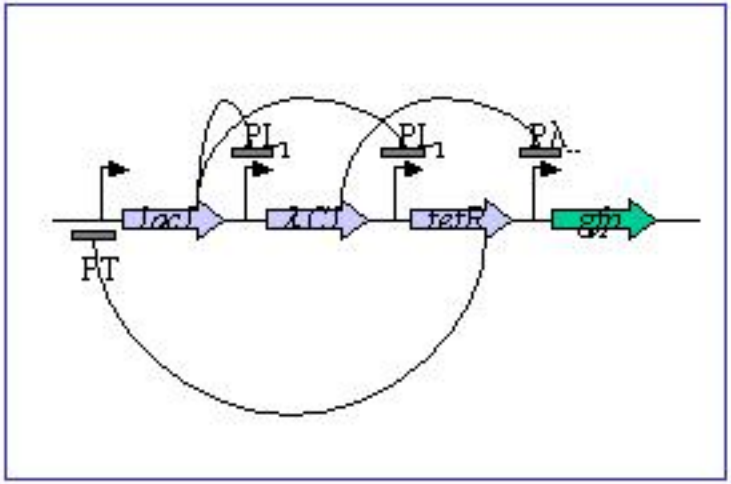
- The combinatorial genetic network depicted consists of
  - Two mutually inhibiting repressor genes, *A* and *B*, which are modulated by two small molecule inducers, X_1 and X_2.
  - The gene products encode the state of the system, and the inducers act as inputs to the network.
  - The state of *B* encodes the output (*Y*).
- This network has two stable states (output is "high" or "low"), but also a metastable state (output assumes an intermediate state between "high" and "low") that is achieved by withdrawing both inputs simultaneously.
- For these reasons, the network is also extremely sensitive to the relative order in which the inputs arrive and, thus, is unpredictable.

Bio-Circuits

Combinatorial Synthesis



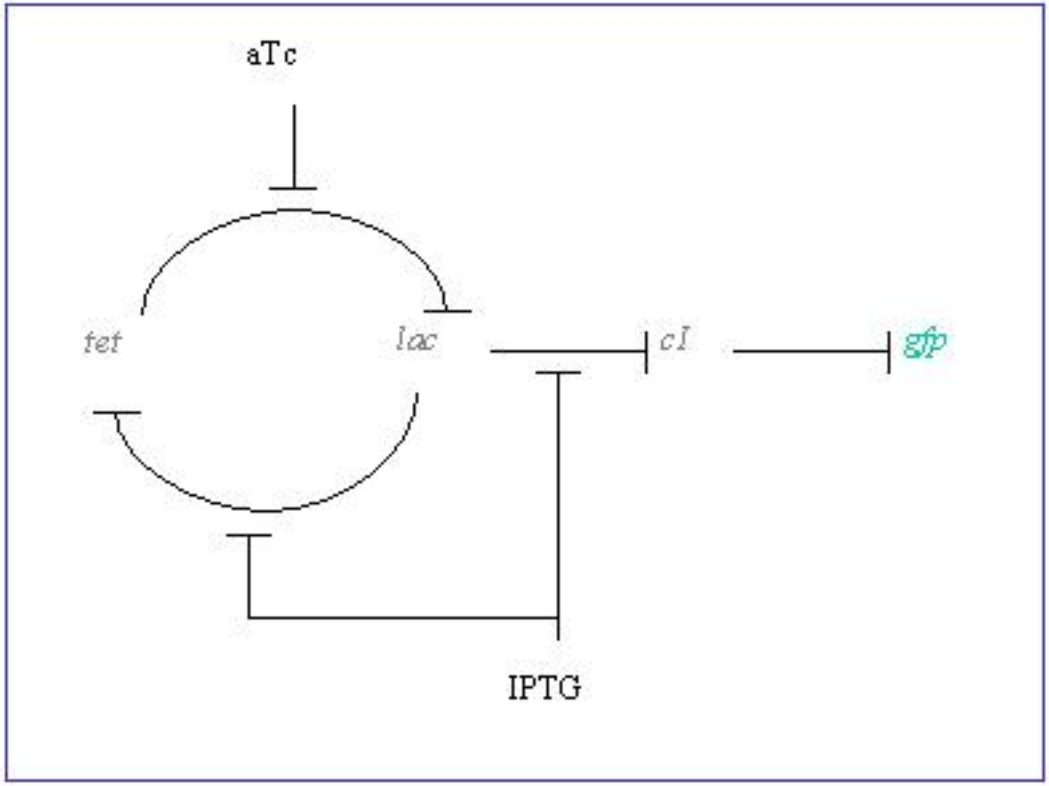
A Circuit with Metastability



A circuit composed of three genes and three suppressors

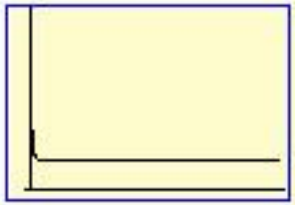
- Four genes: Lac, λ, Tet and GFP.
- Five Operons:
  - Lac-based: PL1, PL2
  - λ CI-based: $P\lambda_-$, $P\lambda_+$
  - Tet-based: PT
- Lac is allosterically suppressible by IPTG
- Tet is allosterically suppressible by aTc

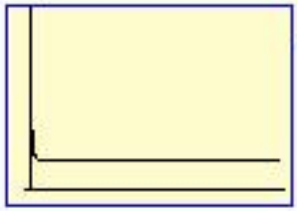
Similar Logic-More Realistic



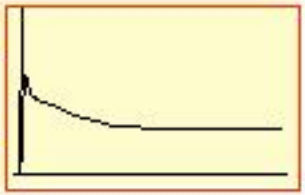
Simulation: IPTG is slightly slower…

IPTG:
iptg(t) = 3;
$d(\text{iptg})/dt = -\text{iptg}(t)e^{[illegible]}$.

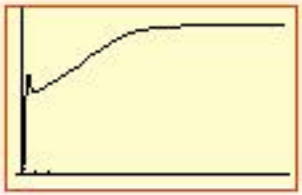




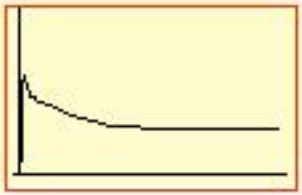
aTc:
atc(t) = 3;
$d(\text{atc})/dt = -\text{atc}(t)e^{[illegible]}$.



tet



lac

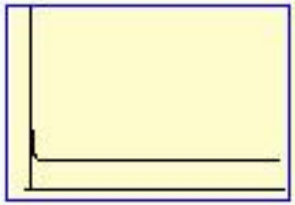


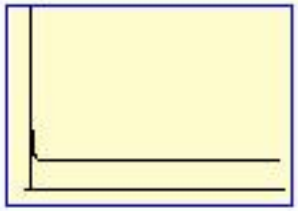
cI

gfp

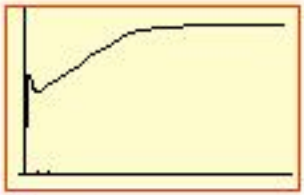
Simulation: IPTG is slightly faster…

IPTG:
$iptg(t) = 3;$
$d(iptg)/dt = -iptg(t)e^{-t/t_i}.$

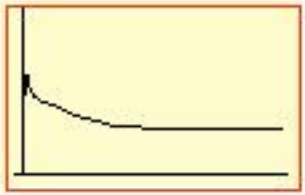




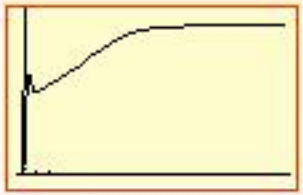
aTc:
$atc(t) = 3;$
$d(atc)/dt = -atc(t)e^{-t/t_a}.$



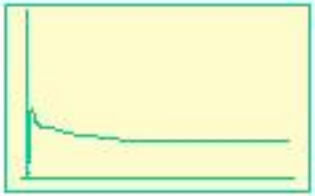
tet



lac

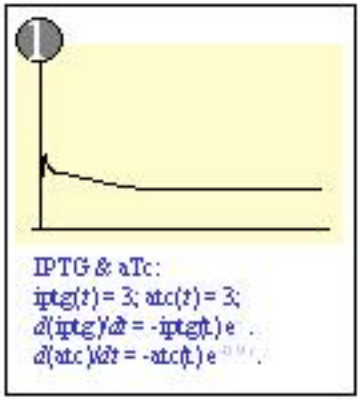


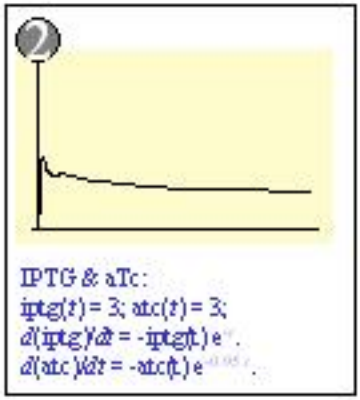
cI

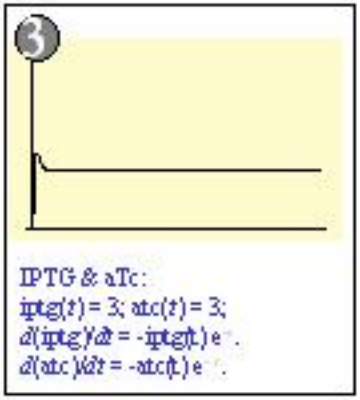


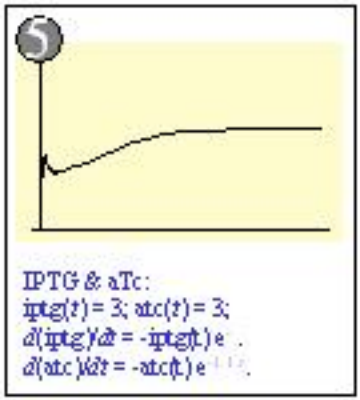
gfp

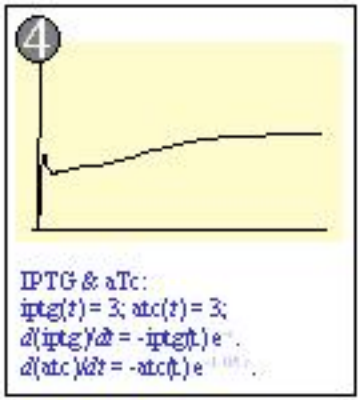
Variation in Speed affects the Cell Behavior











Natural Circuits

- Are these circuits likely to be found in nature?
  - **Most likely yes**, because they are virtually the inevitable consequence of wiring pathways together.
  - **Likely to be highly useful** in a variety of ways.
    - For example, a feedback dyad can function as a simple **memory device**, its state recording which of two present signals arrived first.
    - Such "memory" may be extremely useful when, for example, a free-living microbe is sensing a complex chemical environment, or a cell exposed to a variety of soluble factors during deciding its fate.

Natural Circuits

- Can unpredictability be good?
  - The unpredictability of a metastable circuit may itself be a useful feature
    - In predator-prey evasion, for example, or
    - When an organism scans its environment by random searches.
    - Such systems may be indeterminate at a single-cell level but deterministic in a population at large. For example, to maintain a healthy tissue, it may be advantageous to respond to signals such that some cells divide and some die, maintaining new cells without a net increase in population.
  - Finally, circuits with metastable components may be readily modified by small genetic or biochemical perturbations that bias resolution into one state or another, and thus can be reprogrammed to perform a variety of logical operations.

To be continued…

…