Computational Systems Biology

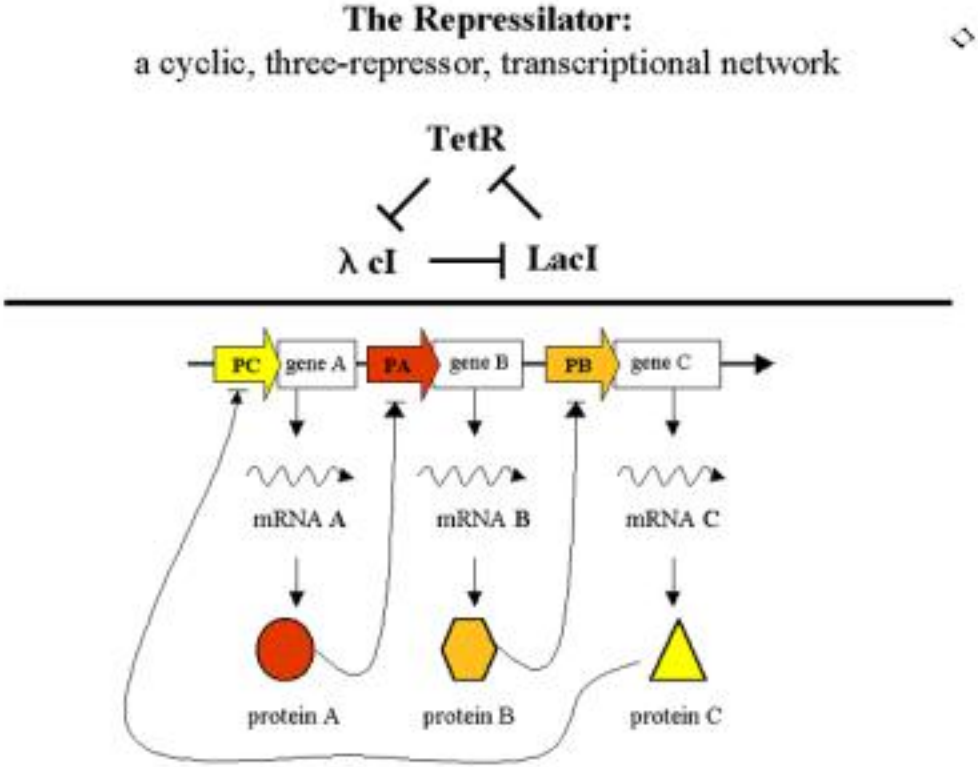
. . . Biology X – Lecture 9 . . .

Bud Mishra

Professor of Computer Science, Mathematics, & Cell Biology

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.

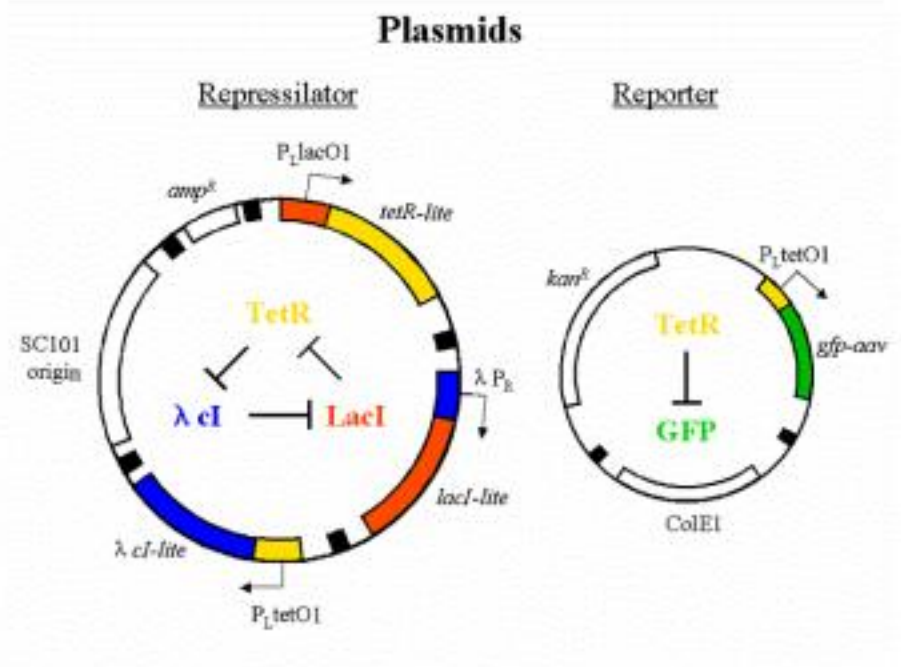
LacI→ ¬ *tetR*; *tetR*→ TetR

TetR→ ¬ λ *cI*; λ *cI* → λ cI

λ cI→ ¬ *lacI*; *lacI*→ LacI

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

Biological Model

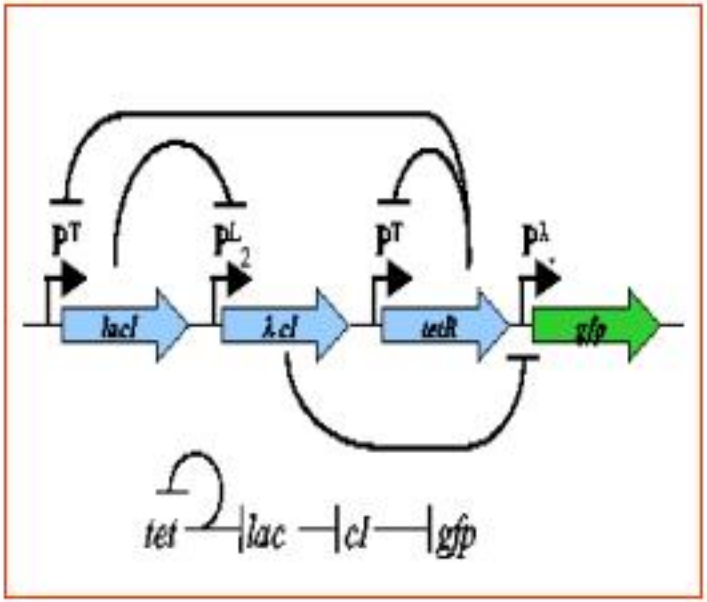


- Standard molecular biology: Construct
  - A low-copy plasmid encoding the repressilator and
  - A compatible higher-copy reporter plasmid containing the tet-repressible promoter PLtetO1 fused to an intermediate stability variant of gfp.

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..

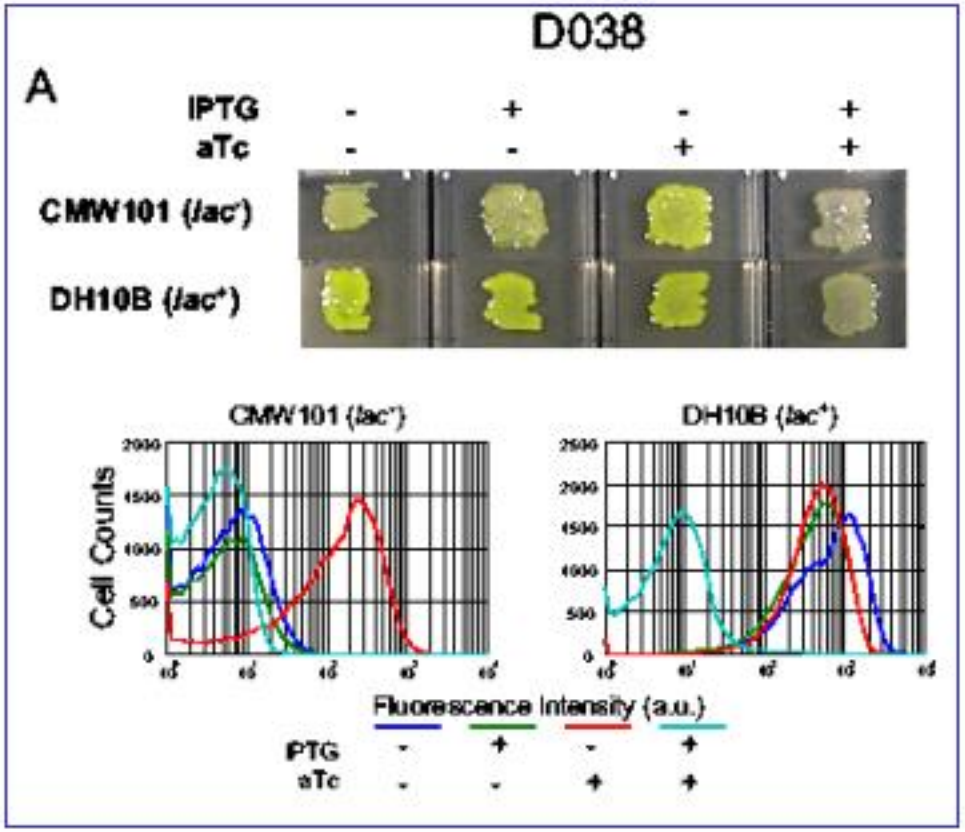
An Example



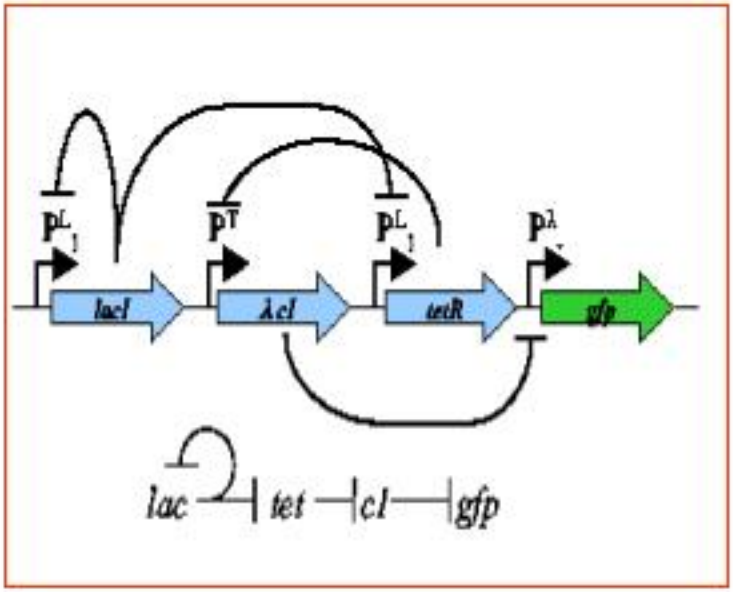
A circuit composed of three genes and three suppressors: D038.

- Four genes: Lac, λ, Tet and GFP.
- Five Operons:
  - Lac-based: PL1, PL2
  - λ CI-based: Pλ_, Pλ+
  - 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



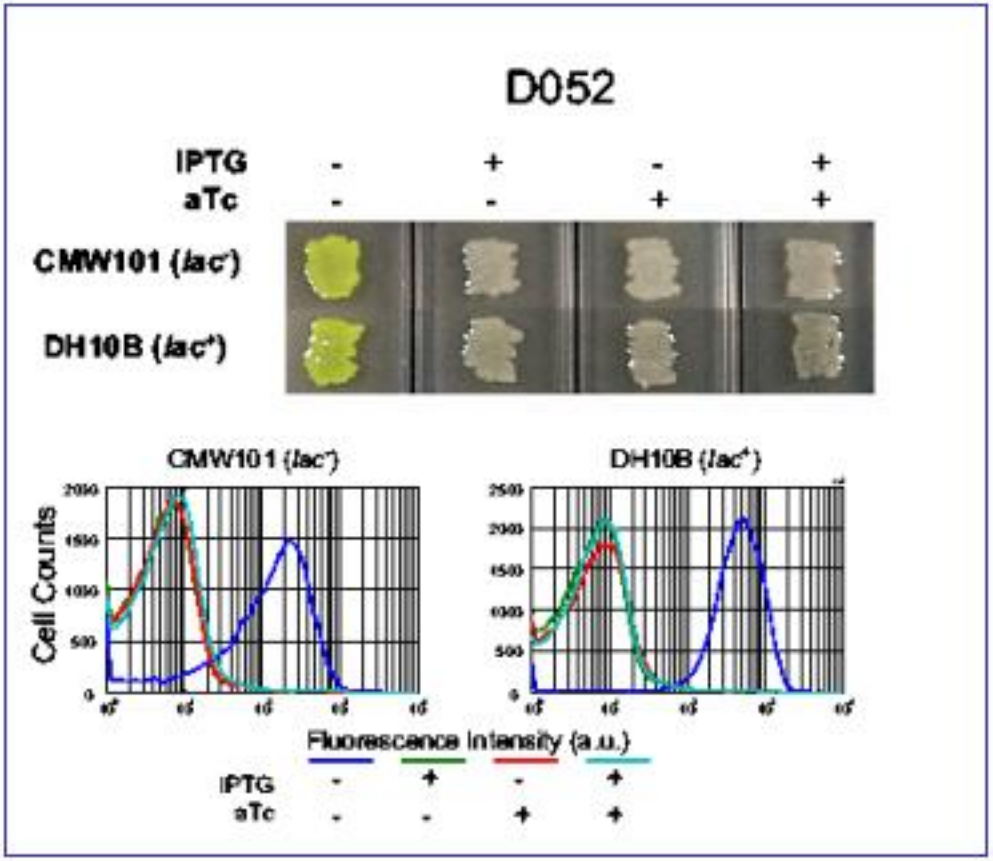
Another Example



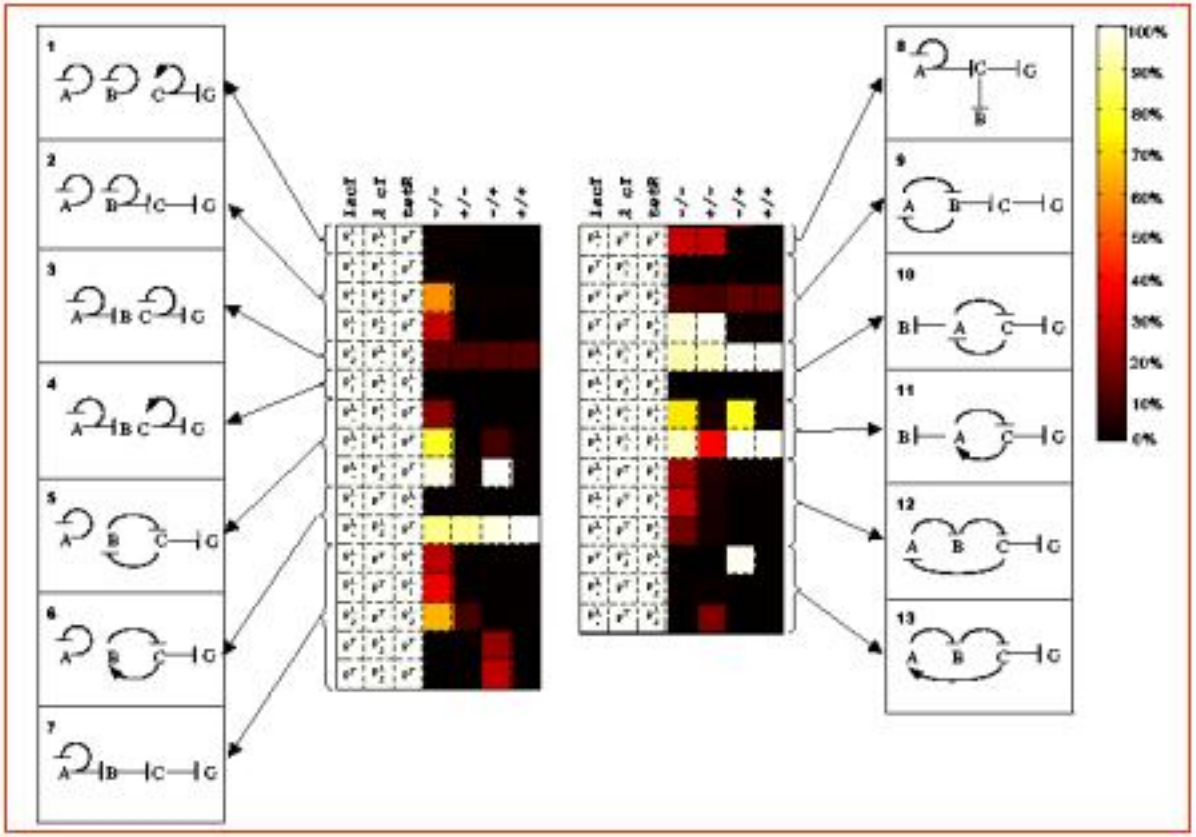
Another circuit composed of three genes and three suppressors: D052.

- 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

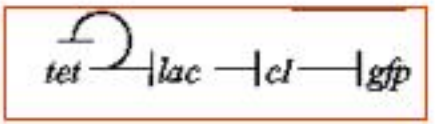


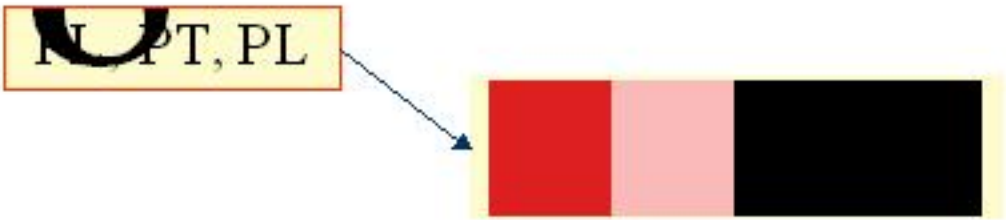
Biological Results



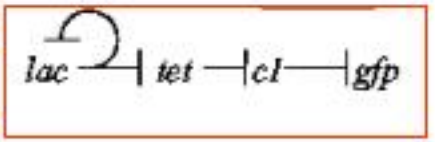
Results

D038

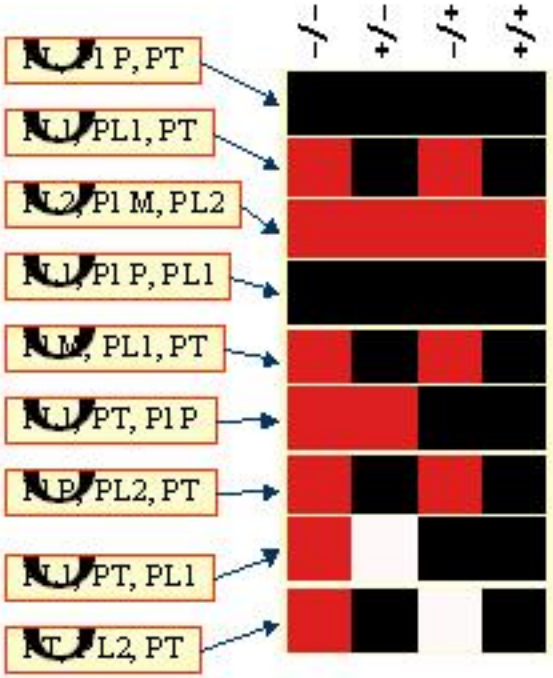


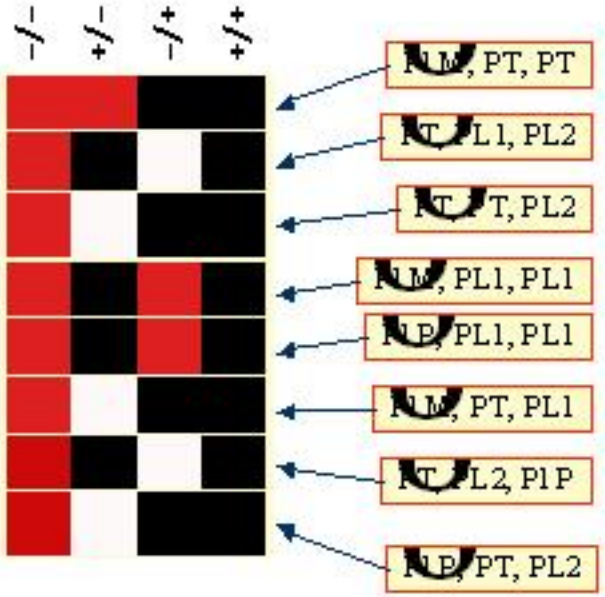


D052

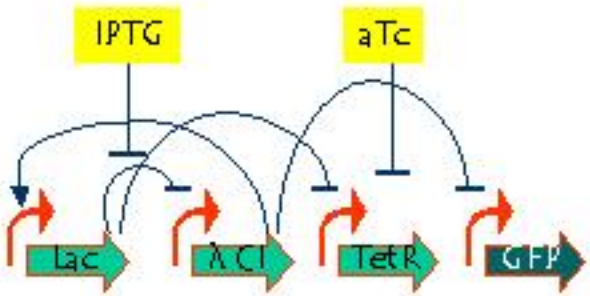


Further Results



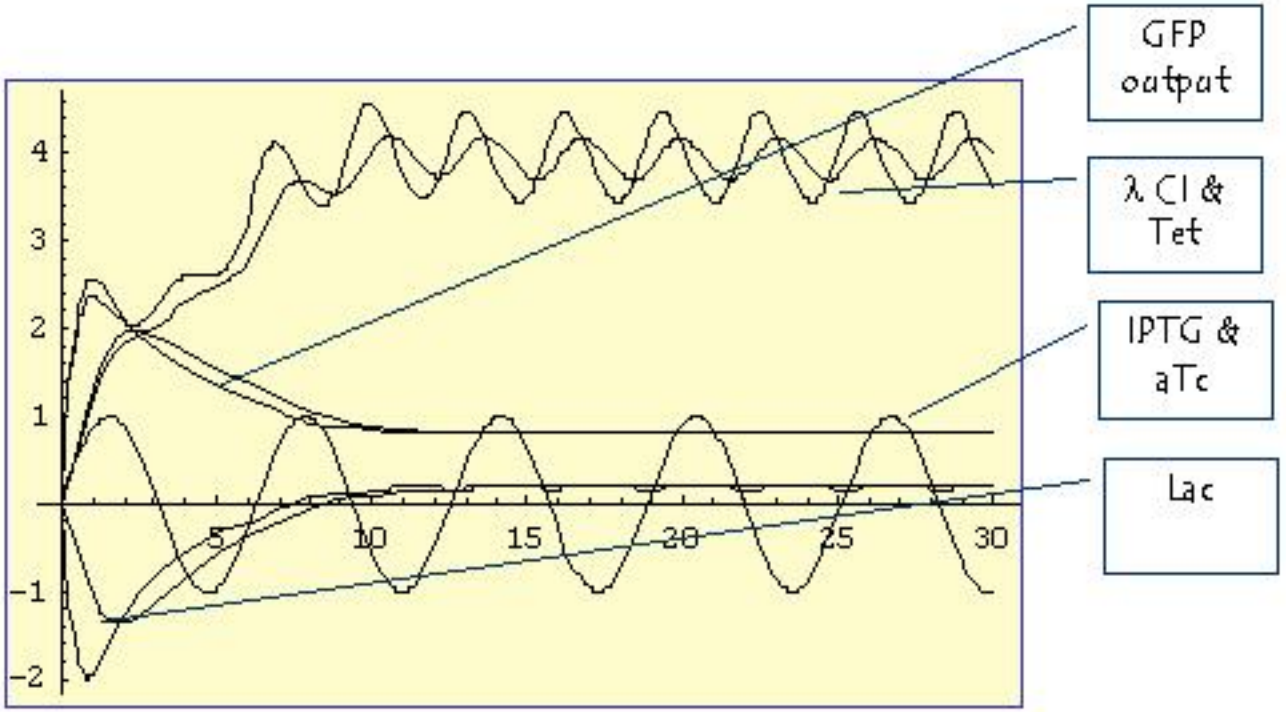


Continuous Time Model



- $d/dt(m_{Lac}) = -m_{Lac} + \alpha\rho - \alpha/(1+p_{\lambda}^{n})$
- $d/dt(p_{Lac}) = -\beta(p_{Lac} - m_{Lac})$
- $d/dt(m_{\lambda}) = -m_{\lambda} + \alpha\rho + \alpha/(1+p_{Lac}^{n}+IPTG^{k})$
- $d/dt(p_{\lambda}) = -\beta(p_{\lambda} - m_{\lambda})$
- $d/dt(m_{Tet}) = -m_{Tet} + \alpha\rho + \alpha/(1+p_{Lac}^{n}+IPTG^{k})$
- $d/dt(p_{Tet}) = -\beta(p_{Tet} - m_{Tet})$
- $d/dt(m_{GFP}) = -m_{GFP} + \alpha\rho + \alpha/(1+p_{\lambda}^{n})$

Simulated Output

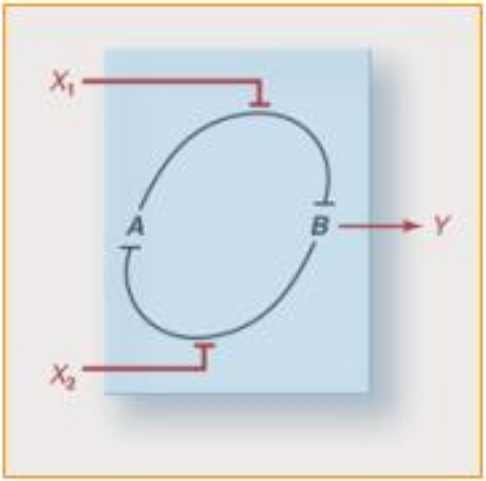


Remaining Questions

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

Metastability

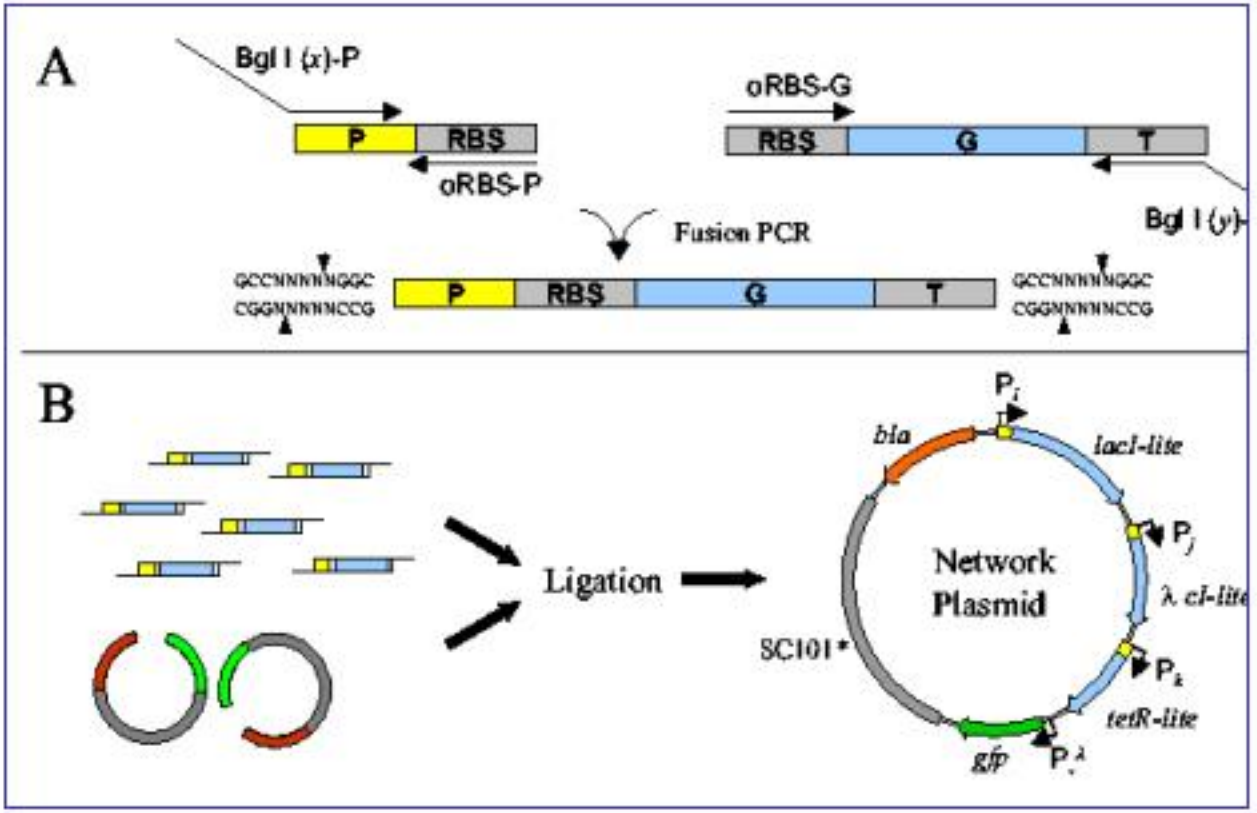
"Feedback Dyad"
A simple metastable biological circuit.



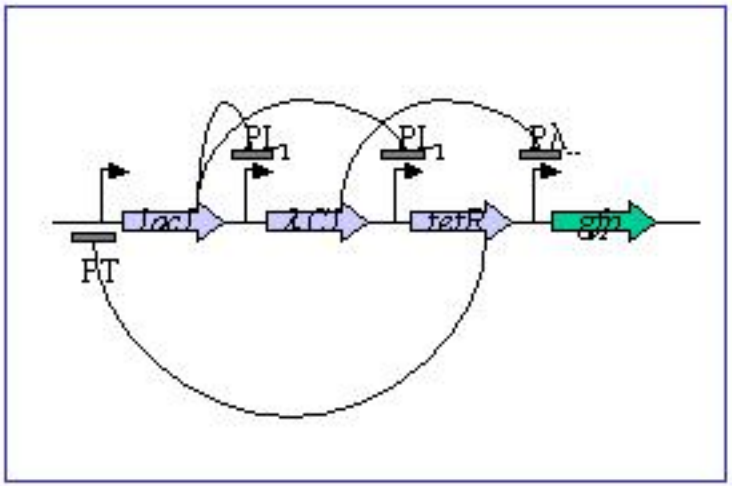
- The combinatorial genetic network depicted consists of
  - Two mutually inhibiting repressor genes, *A* and *B*, which are modulated by two small molecule inducers, X_1 and X_2.
  - The gene products encode the state of the system, and the inducers act as inputs to the network.
  - The state of *B* 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



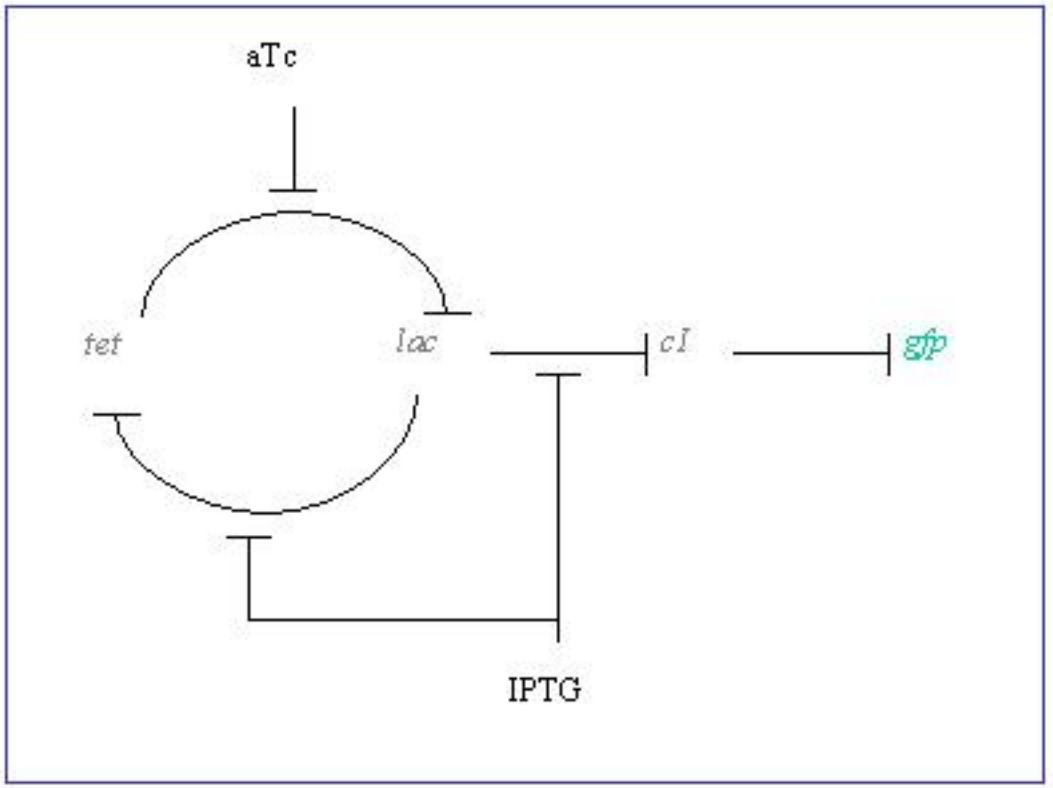
A Circuit with Metastability



A circuit composed of three genes and three suppressors

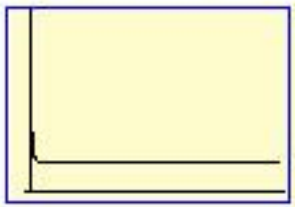
- Four genes: Lac, λ, Tet and GFP.
- Five Operons:
  - Lac-based: PL1, PL2
  - λ CI-based: $P\lambda_-$, $P\lambda_+$
  - Tet-based: PT
- Lac is allosterically suppressible by IPTG
- Tet is allosterically suppressible by aTc

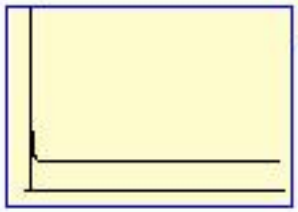
Similar Logic-More Realistic



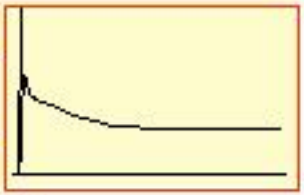
Simulation: IPTG is slightly slower…

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

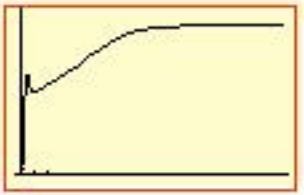




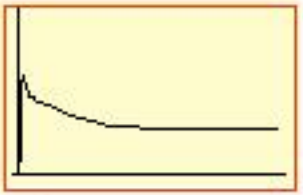
aTc:
$atc(t) = 3;$
$d(atc)/dt = -atc(t)e^{-1.1t}.$



tet



lac

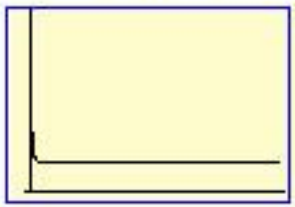


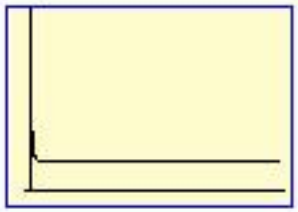
cI

gfp

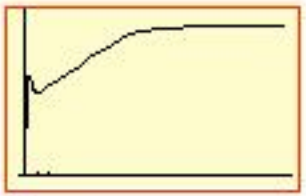
Simulation: IPTG is slightly faster…

IPTG:
$iptg(t) = 3;$
$d(iptg)/dt = -iptg(t)e^{-1.1t}.$

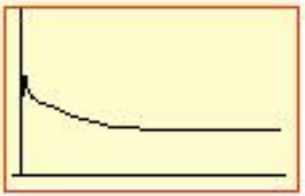




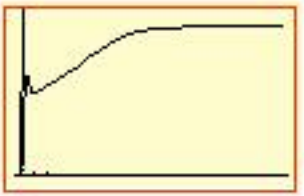
aTc:
$atc(t) = 3;$
$d(atc)/dt = -atc(t)e^{-t}.$



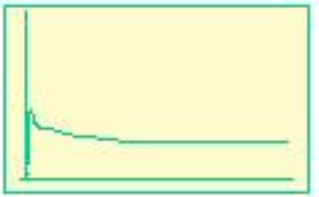
tet



lac

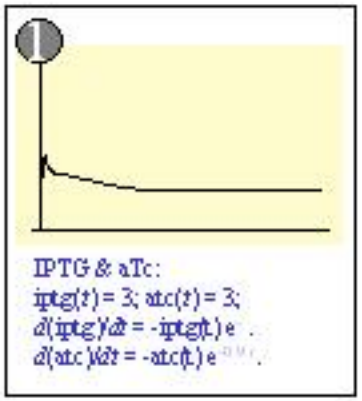


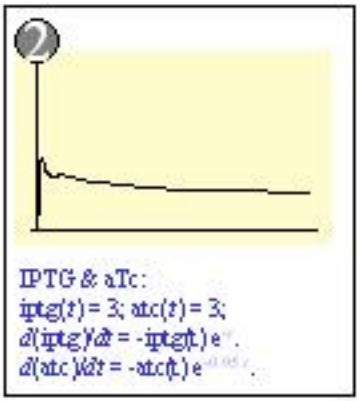
cI

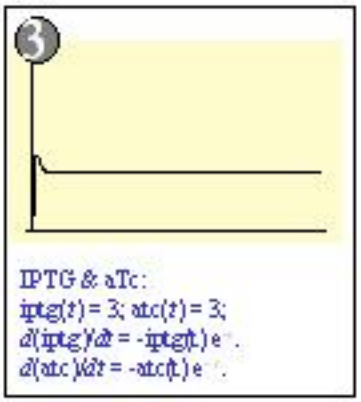


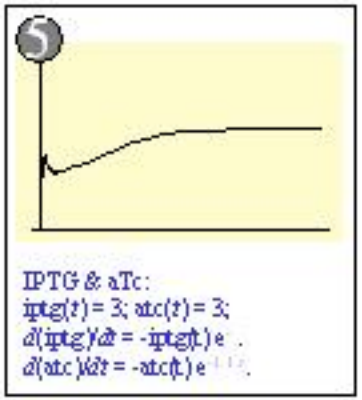
gfp

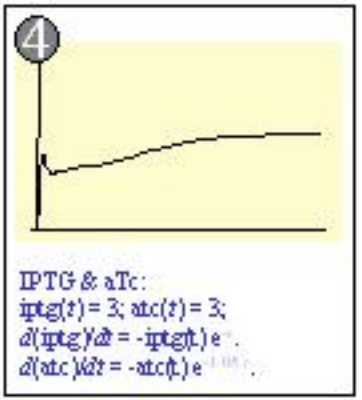
Variation in Speed affects the Cell Behavior











Natural Circuits

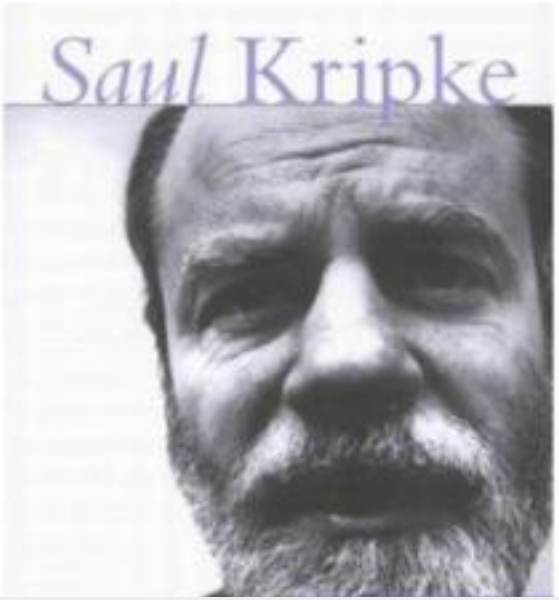
- Are these circuits likely to be found in nature?
  - **Most likely yes**, because they are virtually the inevitable consequence of wiring pathways together.
  - **Likely to be highly useful** in a variety of ways.
    - For example, a feedback dyad can function as a simple **memory device**, its state 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.

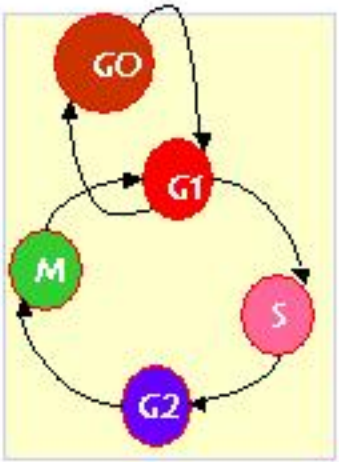
Model Checking

Kripke Structure



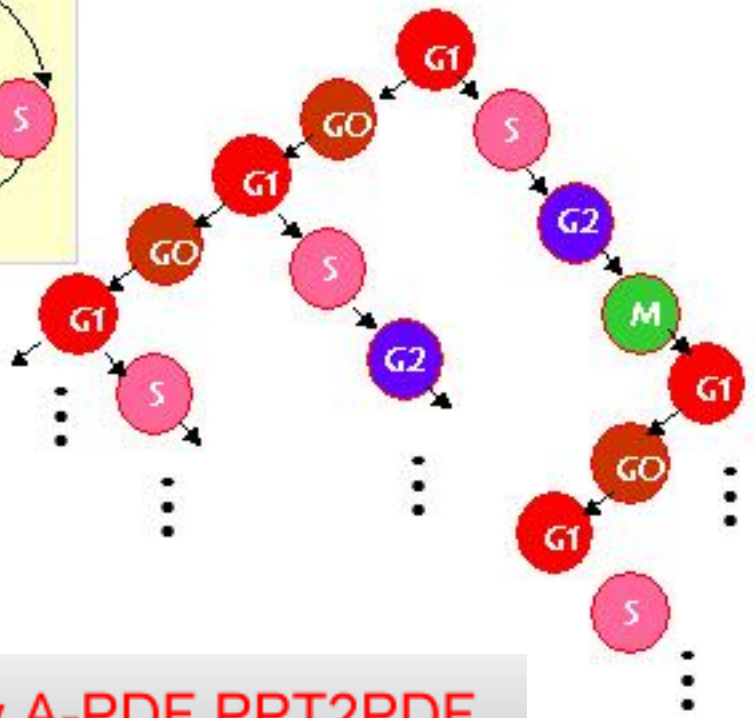
- Formal Encoding of a Dynamical System:
- Simple and intuitive pictorial representation of the behavior of a complex system
  - A **Graph** with **nodes** representing **system states** labeled with information true at that state
  - The **edges** represent **system transitions** as the result of some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

Computation Tree



[a]

[b]

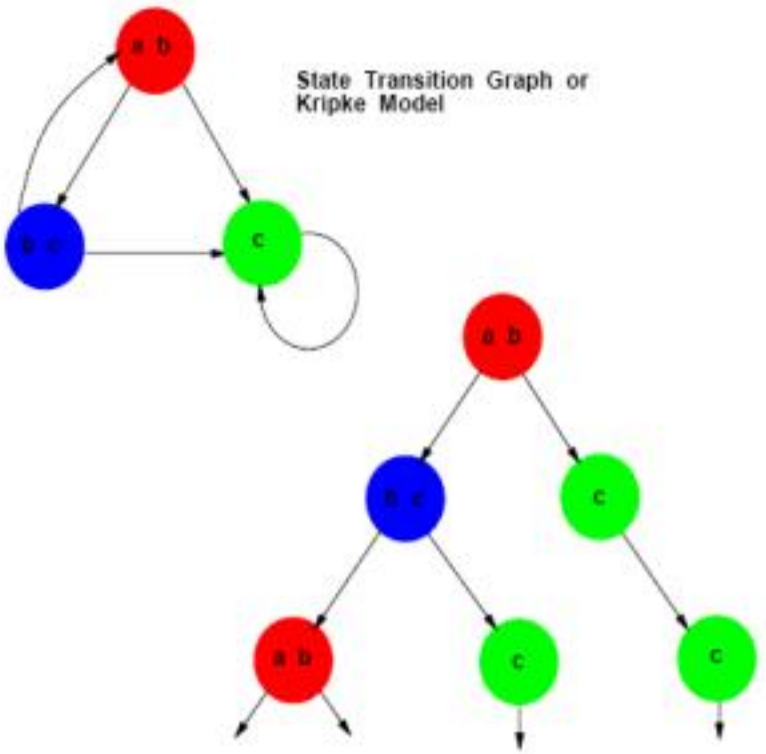


- Finite set of states; Some are initial states
- **Total** transition relation: every state has at least one next state i.e. infinite paths
- There is a set of basic environmental variables or features ("atomic propositions")
- In each state, some atomic propositions are true

Logic

- G_0 is not entered until G_1 holds.
- If S is entered then eventually M is entered.
- It is always true that eventually G_0 is entered

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
- $L : S \to \mathcal{P}(AP)$ 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.

Computation Tree Logics

- Temporal logics may differ according to how they handle branching in the underlying computation tree.
- In a **linear temporal logic**, operators are provided for describing events along a single computation path.
- In a **branching-time logic** the temporal operators quantify over the paths that are possible from a given state.

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

Standard Abbreviations

- The **customary abbreviations** will be used for the connectives of propositional logic.
- In addition, we will use the following abbreviations in writing temporal operators:

- $A(f) \equiv \neg E(\neg f)$
- $f \equiv (\text{true } U\ f)$
- $Gf \equiv \neg F \neg f$

CTL and LTL

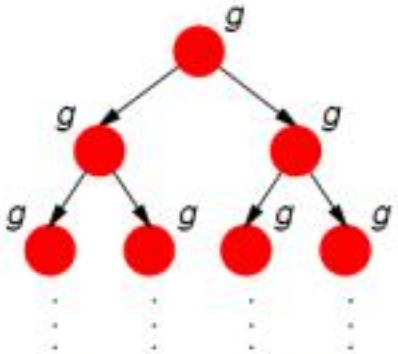
- CTL is a restricted subset of CTL* that permits only branching-time operators—each of the linear-time operators G, F, X, and U must be immediately preceded by a path quantifier.
  - **Example: AG(EF p)**
- LTL consists of formulas that have the form Af where f is a path formula in which the only state subformulas permitted are atomic propositions.
  - **Example: A(FGp)**

Basic CTL Operators

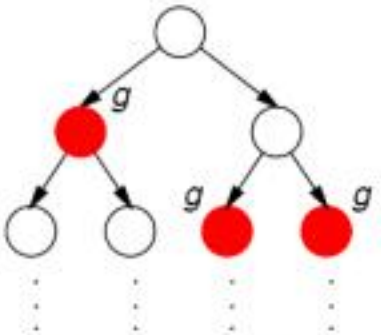
- There are eight basic CTL operators:
  - AX and EX,
  - AG and EG,
  - AF and EF,
  - AU and EU
- Each of these can be expressed in terms of EX, EG, and EU:
  - $AXf = \neg EX(\neg f)$
  - $AGf = \neg EF(\neg f)$
  - $AFf = \neg EG(\neg f)$
  - $EF\,f = E[true\ U\ f]$
  - $\ldots U \neg f \wedge \neg g] \wedge \neg EG \neg g$

Basic CTL Operators

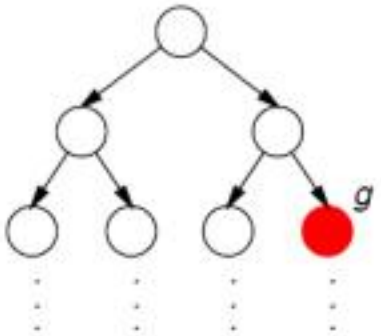
- The four most widely used CTL operators are illustrated below. Each computation tree has the state s_0 as its root.



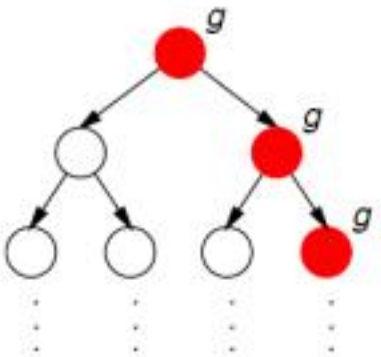
$M, s_0 \models \mathbf{AG}\, g$



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



g



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

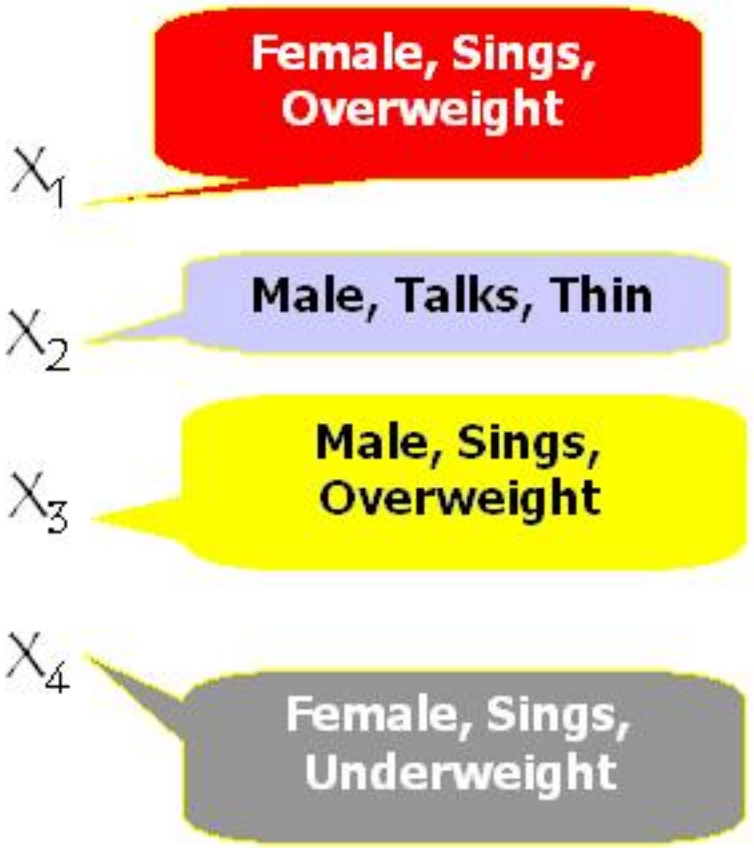
Johann Wittgenstein

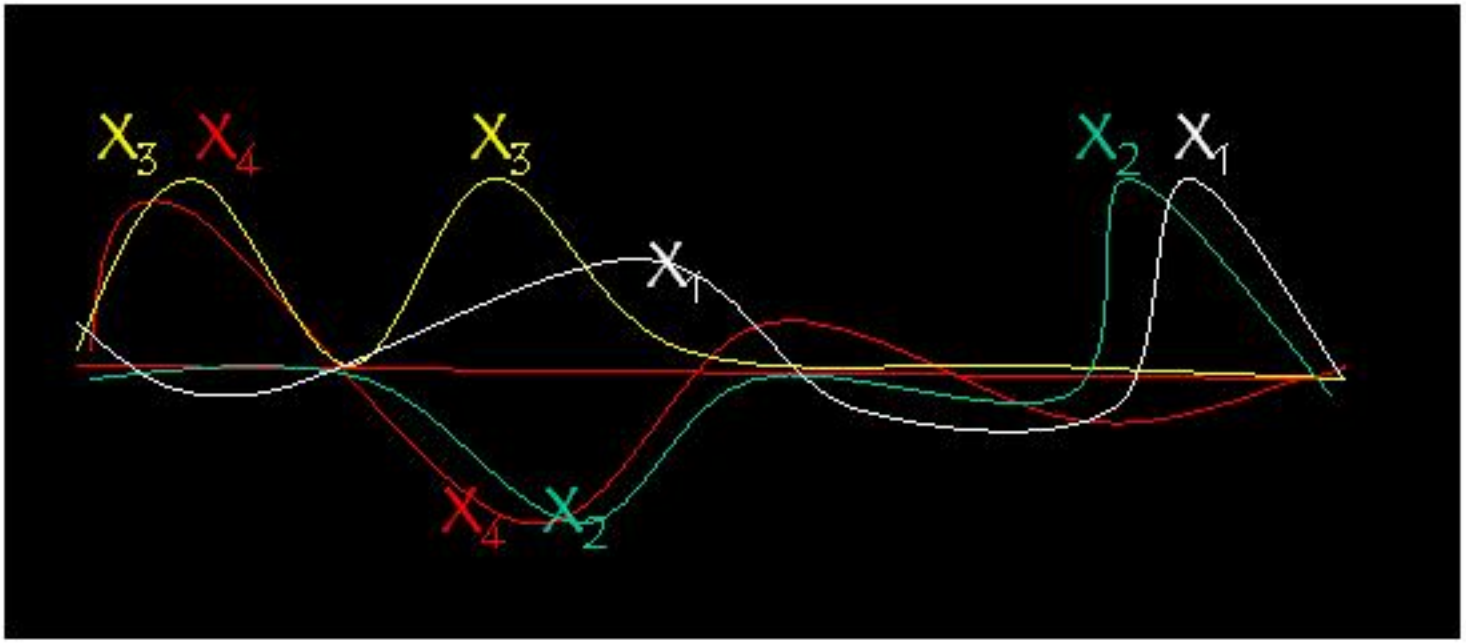
Wittgenstein: Brown Book

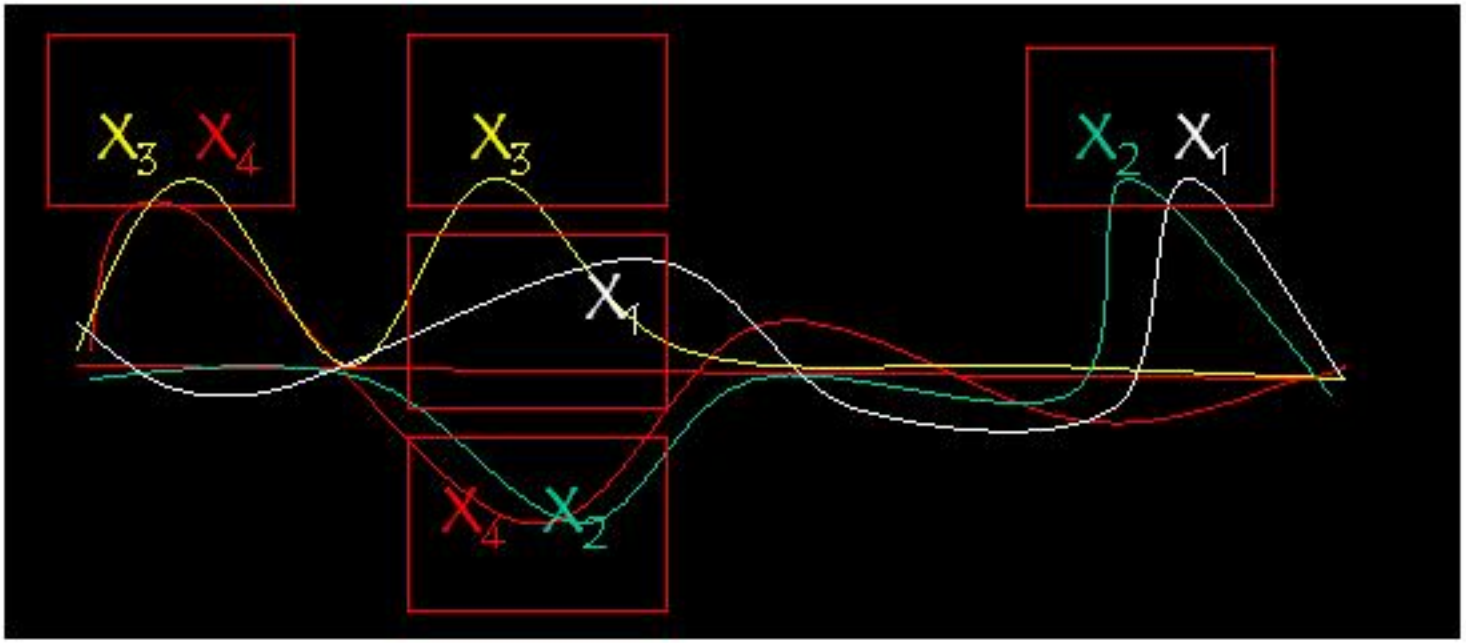
- "Augustine, in describing his learning of language, says that he was taught to speak by learning the names of things…
- "Suppose a man describes a game of chess, without mentioning the existence and operations of the pawns. His description of the game as a natural phenomenon will be incomplete. On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

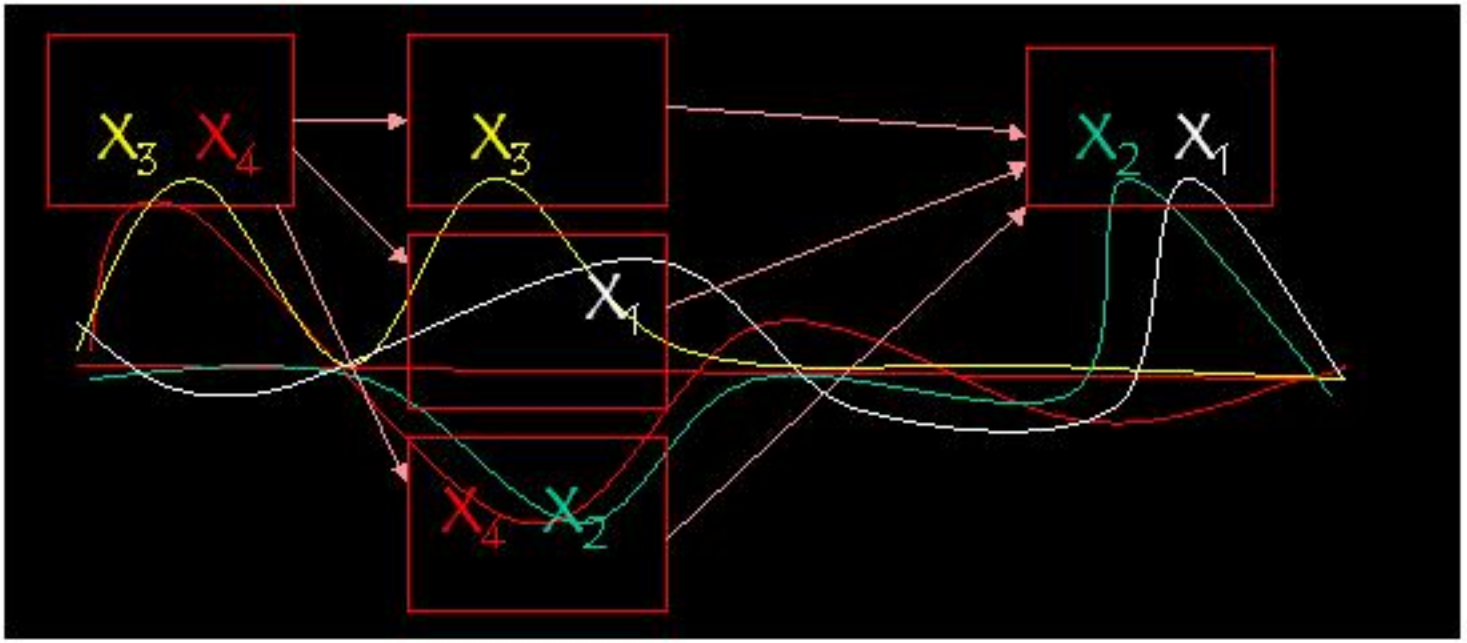
An Augustinian Game...

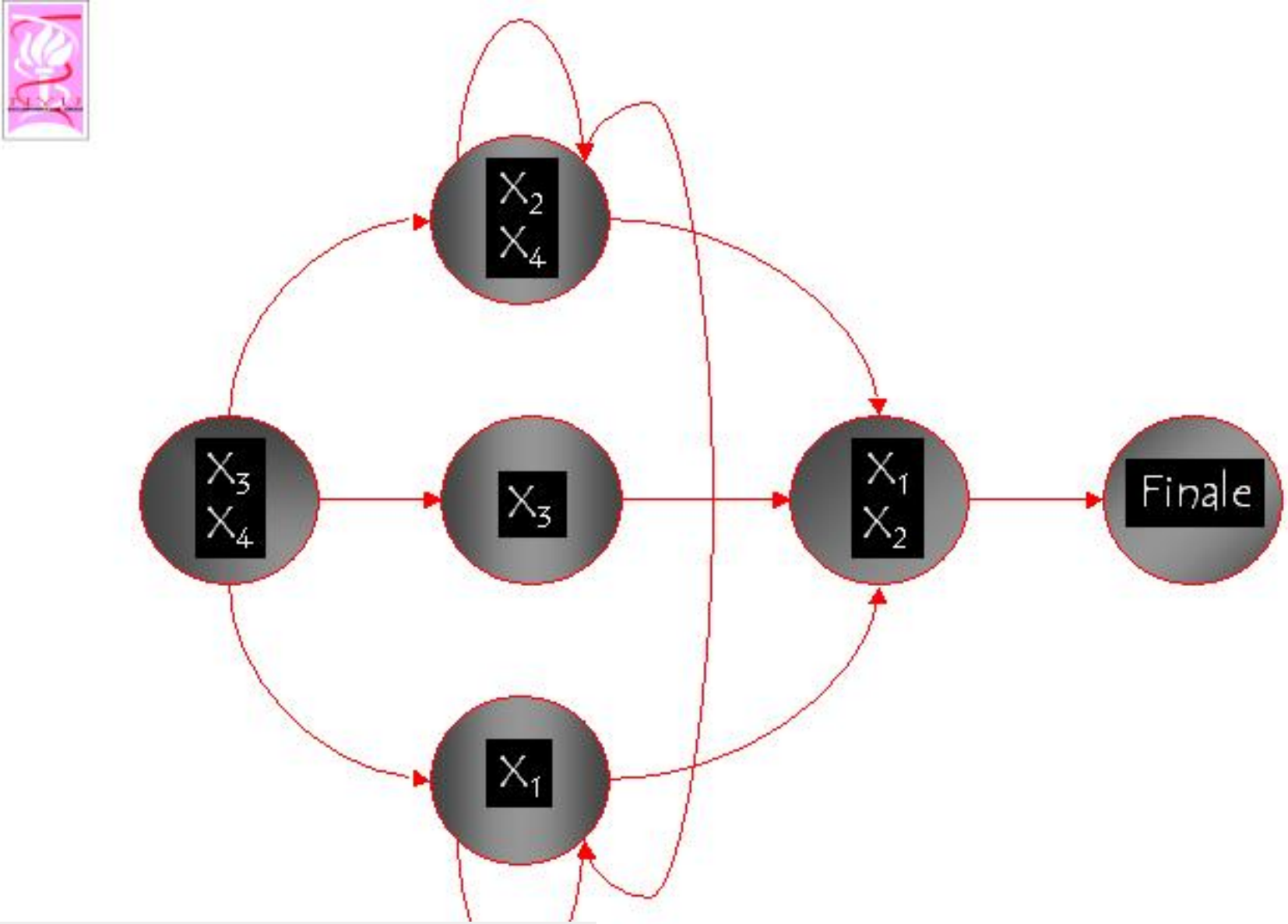
Four Characters Analyzed using Mathematical Logic

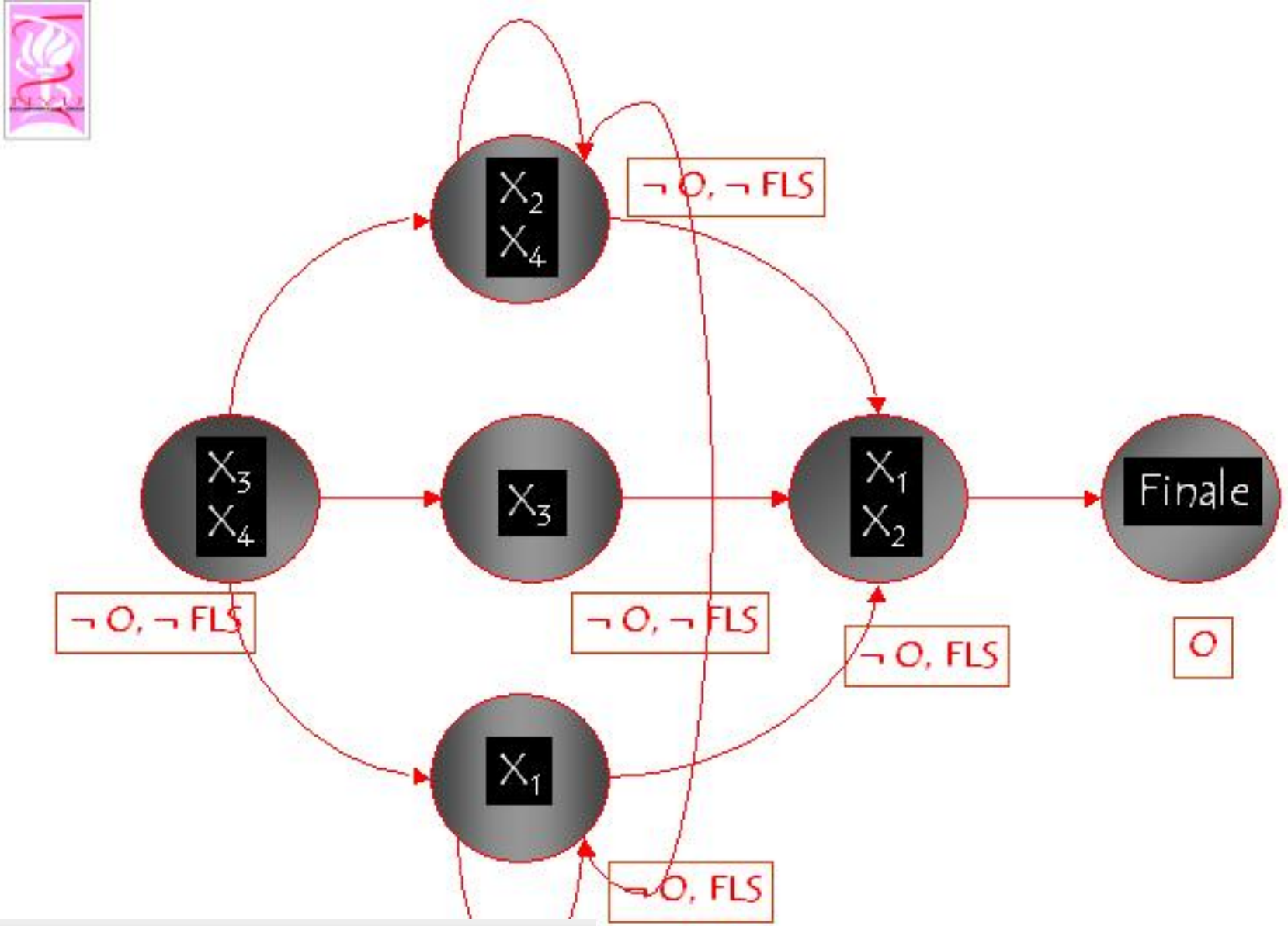


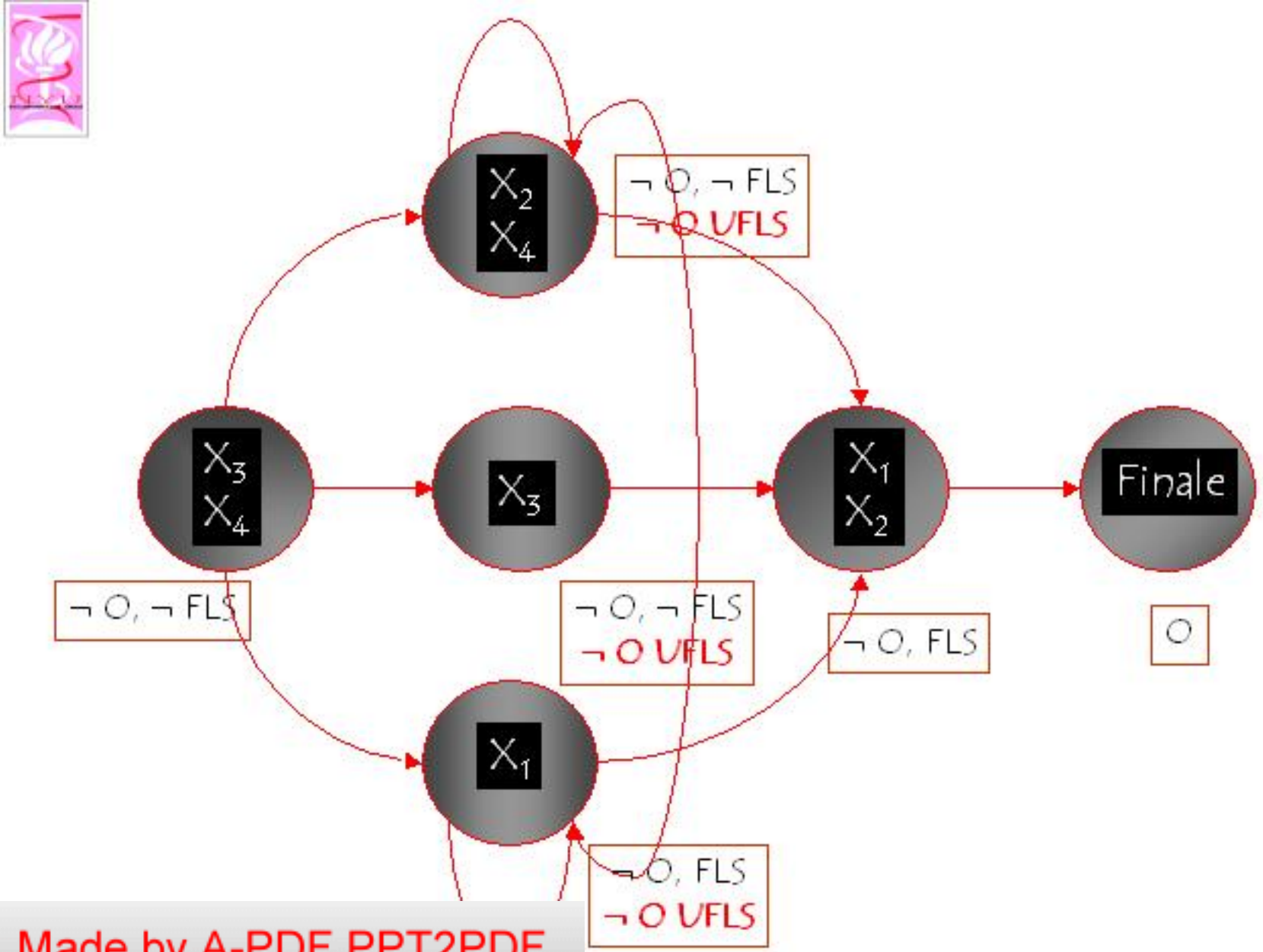


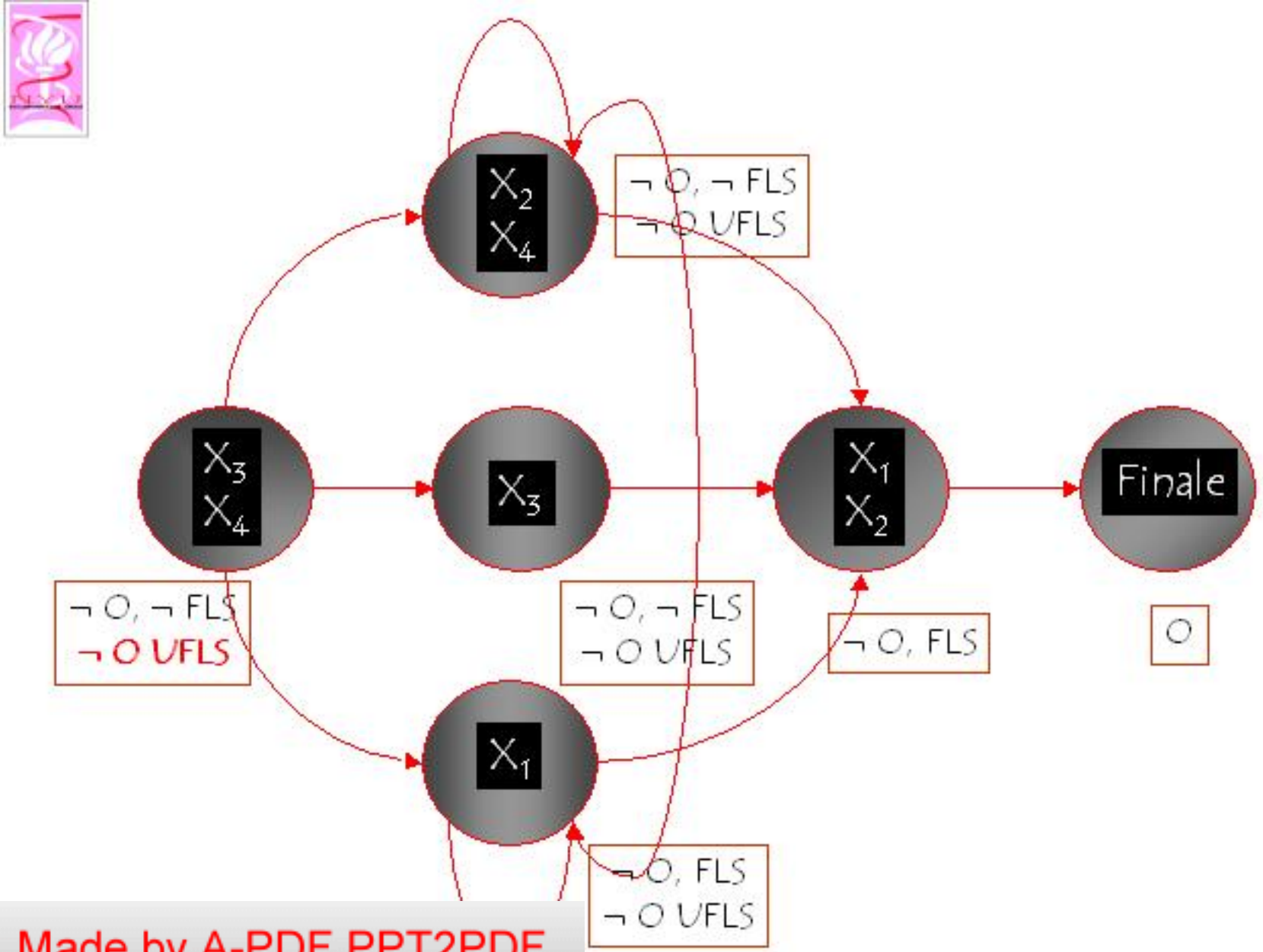












It ain't over 'til the fat lady sings

X_2 X_4

¬ O, ¬ FLS
¬ O UFLS

X_3 X_4

X_3

X_1 X_2

Finale

¬ O, ¬ FLS
¬ O UFLS
A[¬ O UFLS]

¬ O, ¬ FLS
¬ O UFLS

¬ O, FLS

O

X_1

¬ O, FLS
¬ O UFLS

Augustine

"The good Christian should beware of mathematicians and all those who make empty prophecies. The danger already exists that mathematicians have made a covenant with the devil to darken the spirit and confine man in the bonds of Hell."

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

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 $(U, \mathcal{I}, w_0)$, Formula φ) =
  if $w_0 \in$ eval(φ)
  then print("φ is satisfied at w_0 in $(U, \mathcal{I})$")
  else print("φ not satisfied at w_0 in $(U, \mathcal{I})$");

function eval (Formula φ): Pointset =
  case φ **of**
    p : **return** $\mathcal{I}$(p);
    $\bot$: **return** {};
    $(\psi_1 \rightarrow \psi_2)$: **return** $U \setminus$ eval(ψ_1) $\cup$ eval(ψ_2);
    $\mathbf{E}(\psi_2 \, \mathbf{U}^+ \, \psi_1)$: $E1 :=$ eval(ψ_1); $E2 :=$ eval(ψ_2); $E := \{\}$;
      repeat until stabilization
        $E := E \cup \{w \mid (\text{succ}(w) \cap (E1 \cup (E2 \cap E))) \neq \{\}\}$;
      return E;
    $\mathbf{A}(\psi_2 \, \mathbf{U}^+ \, \psi_1)$: $E1 :=$ eval(ψ_1); $E2 :=$ eval(ψ_2); $E := \{\}$;
      repeat until stabilization
        $E := E \cup \{w \mid \{\} \neq \text{succ}(w) \subseteq E1 \cup (E2 \cap E)\}$;
      return E;

intset = **return** $\{w' \mid (w, w') \in \mathcal{I}(\prec)\}$;

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:

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

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

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

To be continued…

. . .