# On Using Temporal Logic with Constraints to express Biological Properties of Cell Processes

François Fages,
Constraint Programming Group,
INRIA Rocquencourt, France

Fages F., Soliman S., Chabrier-Rivier N. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of biological physics and chemistry* 4:64-73. 2004

Calzone L., Chabrier-Rivier N., Fages F., Soliman S., Machine learning biochemical networks from temporal logic properties. *Trans. in Computational Systems Biology* Oct. 06.

Software: BIOCHAM `http://contraintes.inria.fr/BIOCHAM`

# Transposing Programming Language Notions to Systems Biology

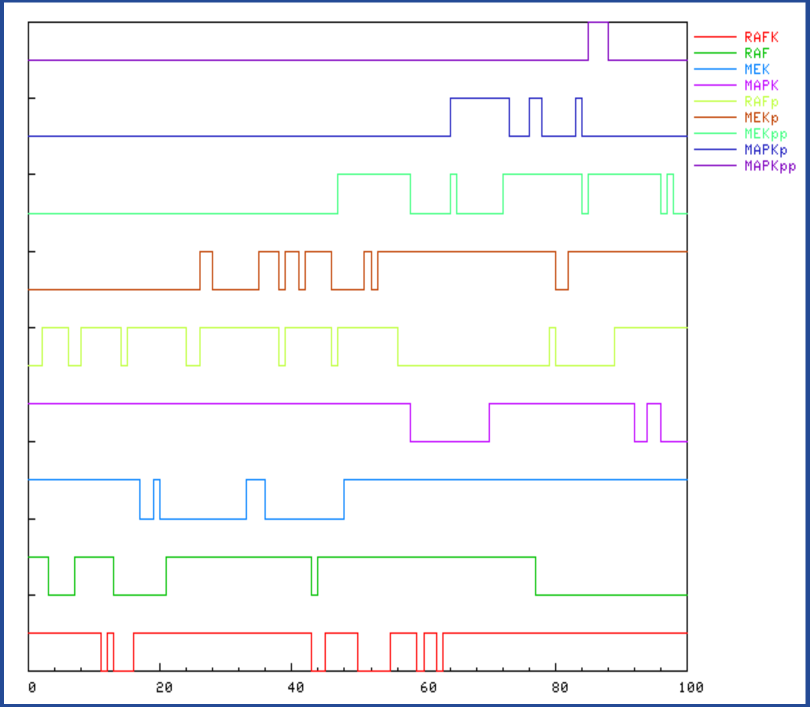
Formally, “the” behavior of a system depends on *our choice* of observables.

# Transposing Programming Language Notions to Systems Biology

Formally, “the” behavior of a system depends on *our choice* of observables.

presence/absence

of molecules

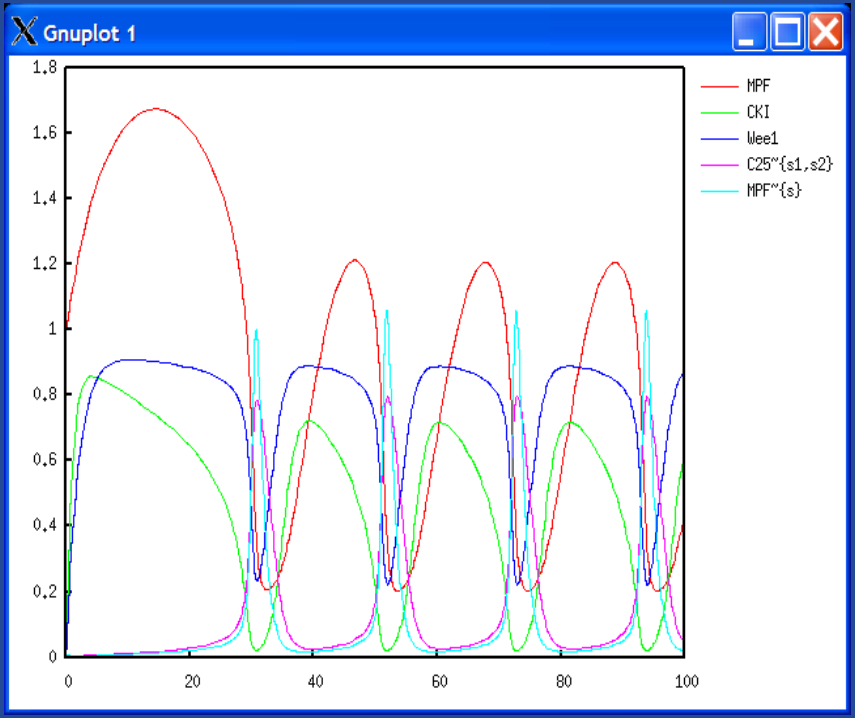


# Transposing Programming Language Notions to Systems Biology

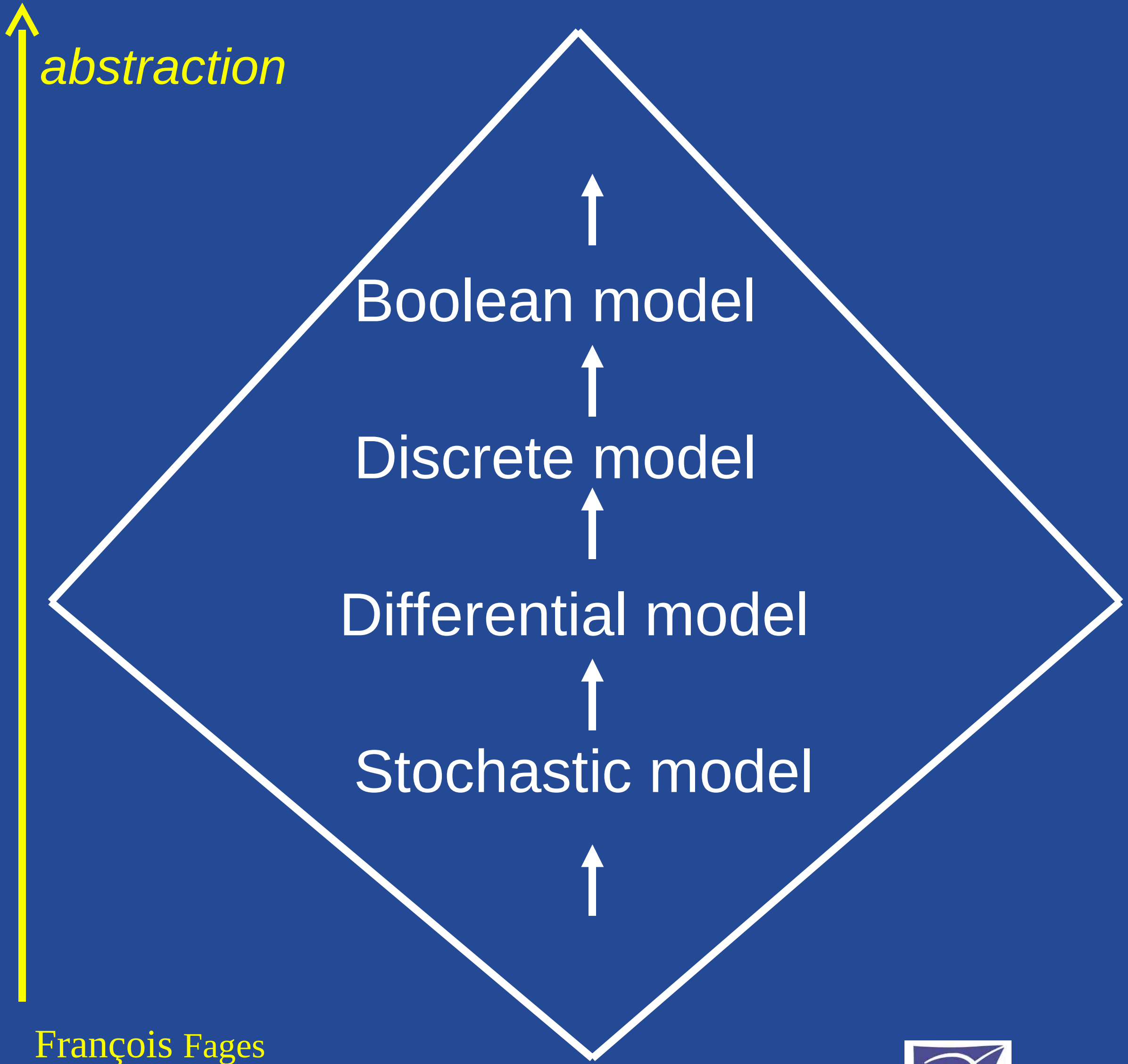
Formally, “the” behavior of a system depends on *our choice* of observables.

molecule

concentrations



# Hierarchy of Biochemical Models



*abstraction*

Boolean model

Discrete model

Differential model

Stochastic model

Models for answering queries

The more abstract the better

# A Logical Paradigm for Systems Biology

Biochemical model = Transition system

Biological property = Temporal Logic formula

Biological validation = Model-checking

[Lincoln et al. 02] [Chabrier Fages 03] [Bernot et al. 04] [Alon et al. 04] …

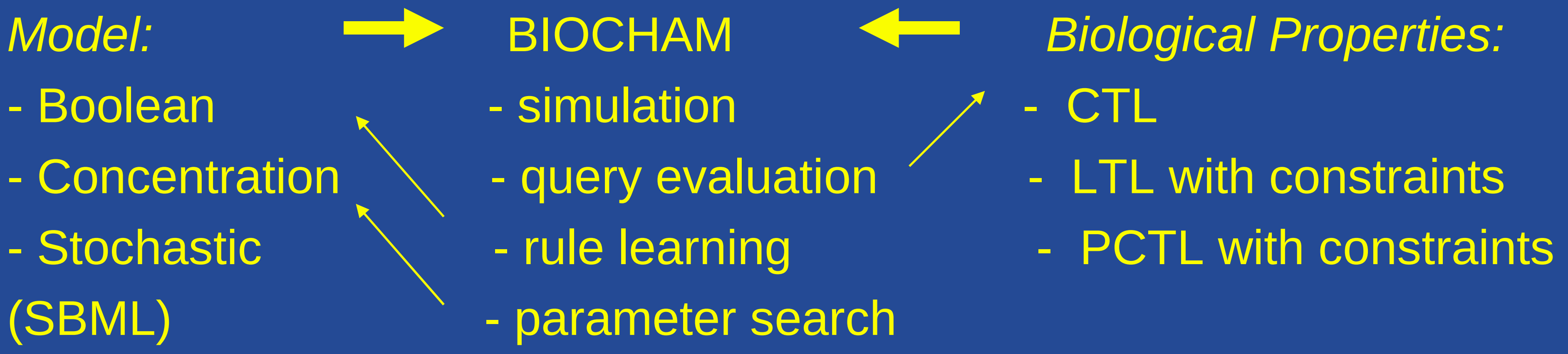
# A Logical Paradigm for Systems Biology

Biochemical model = Transition system

Biological property = Temporal Logic formula

Biological validation = Model-checking

Biochemical Abstract Machine environment



*Model:*

- Boolean
- Concentration
- Stochastic

(SBML)

BIOCHAM

- simulation
- query evaluation
- rule learning
- parameter search

*Biological Properties:*

- CTL
- LTL with constraints
- PCTL with constraints

# Plan of my talk

- Language of reaction rules for modeling biochemical systems
  - Boolean semantics
  - Differential semantics
  - Stochastic semantics
- Temporal logic language for formalizing biological properties
  - CTL
  - LTL with constraints over the reals
  - PCTL with constraints over integers
- Machine learning from temporal properties
  - Searching parameter values
  - Learning rules and model revision
- Conclusions and collaborations

# 1. Language of Reaction Rules

| | | |
|---|---|---|
| Complexation | A + B => A-B | *Cell cycle control model [Tyson 91]* |
| Decomplexation | A-B => A + B | k6*[Cdc2-Cyclin~{p1}] for<br>Cdc2-Cyclin~{p1} => Cdc2+Cyclin~{p1} |
| Phosphorylation | A =[B]=> A~{p} | k8*[Cdc2] for Cdc2 => Cdc2~{p1} |
| Dephosphorylation | A~{p} =[B]=> A | |
| Synthesis | _ =[B]=> A | k1 for _ => Cyclin. |
| Degradation | A =[B]=> _ | k2*[Cyclin] for Cyclin => _. |
| Transport | A::L1 => A::L2 | |

# BIOCHAM Semantics of a Rule Set {$e_i$ for $S_i$ => $S'_i$}

1. Boolean Semantics: presence-absence of molecules

   Concurrent Transition System (asynchronous, non-deterministic)

   A reaction A+B=>C+D is translated into 4 transition rules considering the possible consumption of the reactants by the reaction:

   A+B→A+B+C+D

   A+B→¬A+B +C+D

   A+B→A+¬B+C+D

   A+B→¬A+¬B+C+D

# BIOCHAM Semantics of a Rule Set $\{e_i$ for $S_i \Rightarrow S'_i\}$

1. Boolean Semantics: presence-absence of molecules

   Concurrent Transition System (asynchronous, non-deterministic)

2. Concentration Semantics: number / volume

   Ordinary Differential Equations (deterministic)

$$dx_k/dt = \Sigma_{Xi=1}^{n} r_i(x_k) * e_i - \Sigma_{Xj=1}^{n} l_j(x_k) * e_j$$

where $r_i$ (resp. $l_j$) is the stochiometric coefficient of $x_k$ in $S'_i$ (resp. $S_i$) multiplied by the volume ratio of the location of $x_k$.

# BIOCHAM Semantics of a Rule Set $\{e_i \text{ for } S_i \Rightarrow S'_i\}$

1. Boolean Semantics: presence-absence of molecules

   Concurrent Transition System (asynchronous, non-deterministic)

2. Concentration Semantics: number / volume

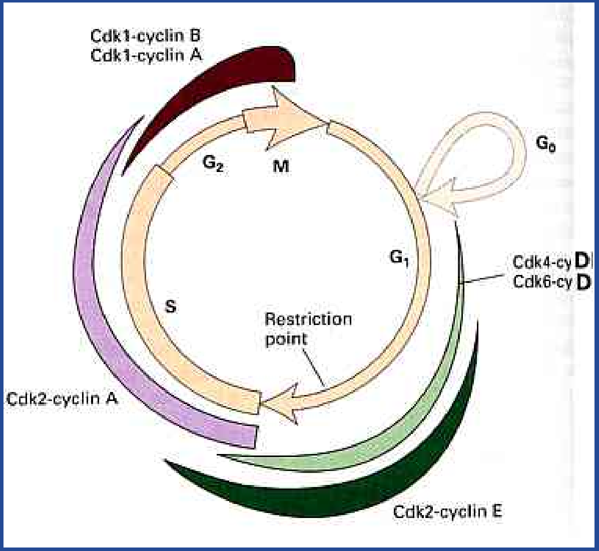
   Ordinary Differential Equations (deterministic)

3. Stochastic Semantics: number of molecules

   Continuous time Markov chain the $e_i$'s giving transition probabilities

$$r_i = c_i \times (V_i \times K)^{(1-\sum_{k=1}^{m} l_i(x_k))} \times \prod_{k=1}^{m} (l_i(x_k))$$

# Example: Cell Cycle Control Model [Tyson 91]



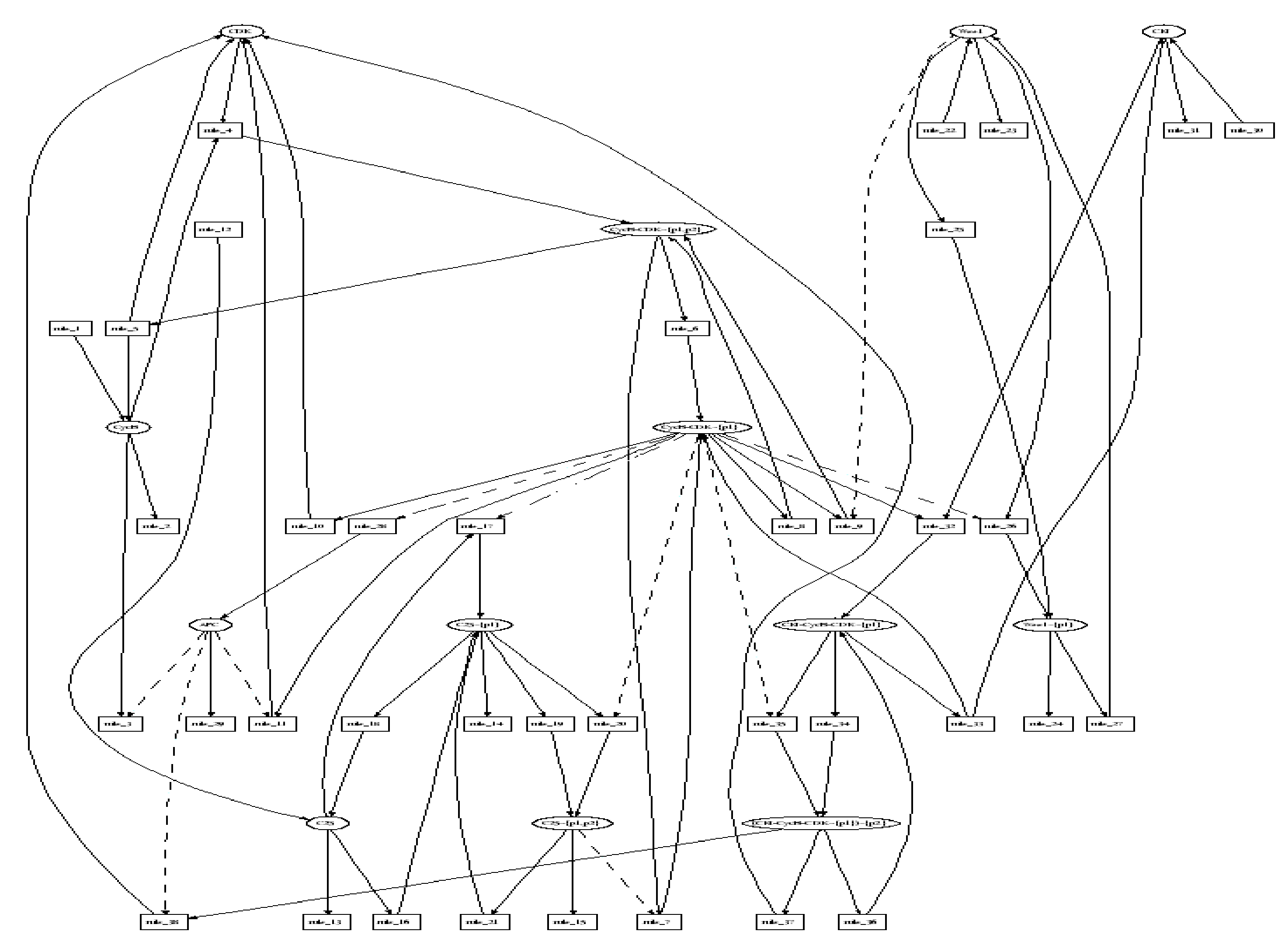
k1 for _ => Cyclin.
k2*[Cyclin] for Cyclin => _.
k7*[Cyclin~{p1}] for Cyclin~{p1} => _.

k8*[Cdc2] for Cdc2 => Cdc2~{p1}.
k9*[Cdc2~{p1}] for Cdc2~{p1} =>Cdc2.

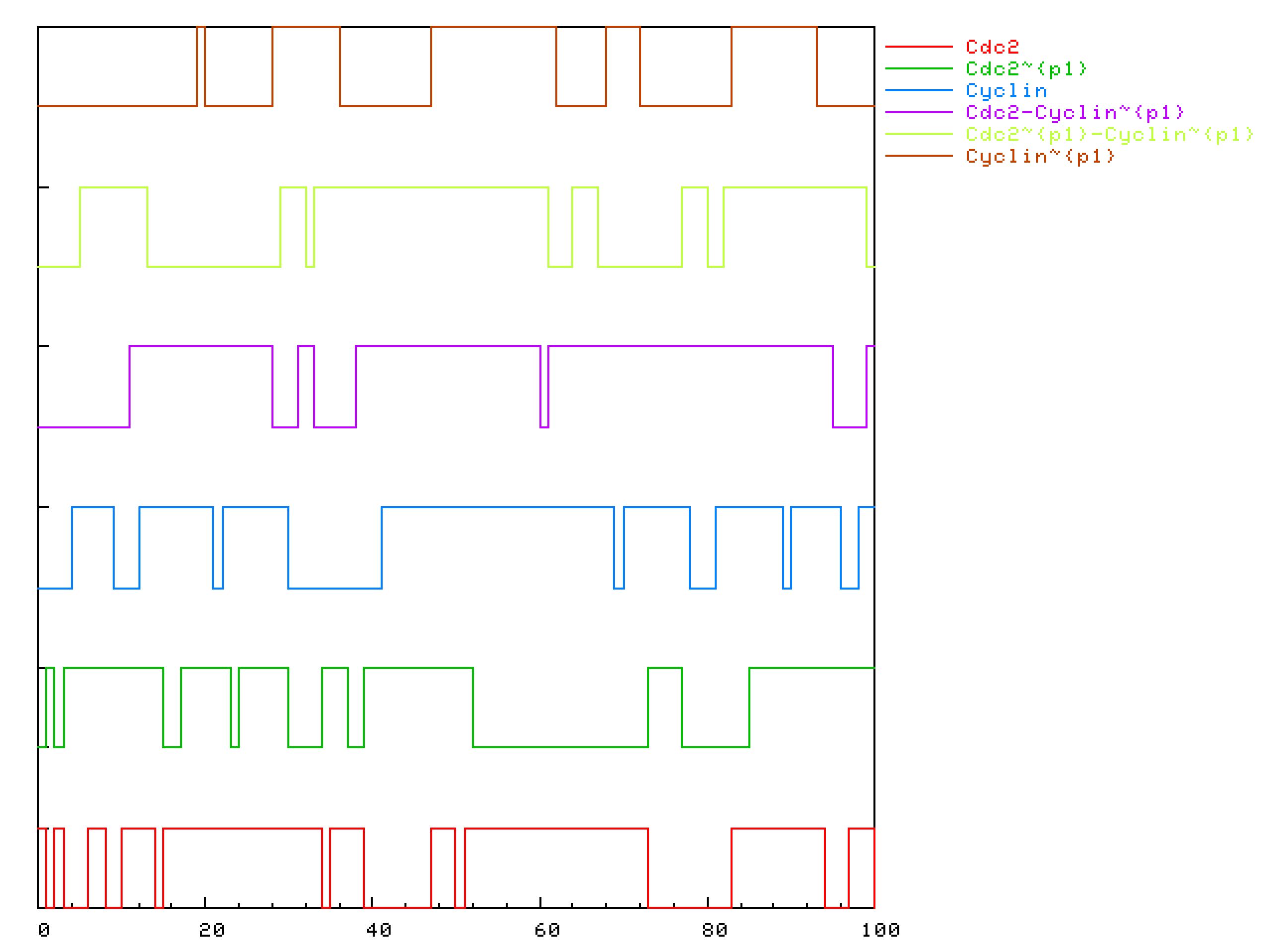
k3*[Cyclin]*[Cdc2~{p1}] for Cyclin+Cdc2~{p1} => Cdc2~{p1}-Cyclin~{p1}.
k4p*[Cdc2~{p1}-Cyclin~{p1}] for Cdc2~{p1}-Cyclin~{p1} => Cdc2-Cyclin~{p1}.
k4*[Cdc2-Cyclin~{p1}]^2*[Cdc2~{p1}-Cyclin~{p1}] for
Cdc2~{p1}-Cyclin~{p1} =[Cdc2-Cyclin~{p1}] => Cdc2-Cyclin~{p1}.

k5*[Cdc2-Cyclin~{p1}] for Cdc2-Cyclin~{p1} => Cdc2~{p1}-Cyclin~{p1}.
k6*[Cdc2-Cyclin~{p1}] for Cdc2-Cyclin~{p1} => Cdc2+Cyclin~{p1}.

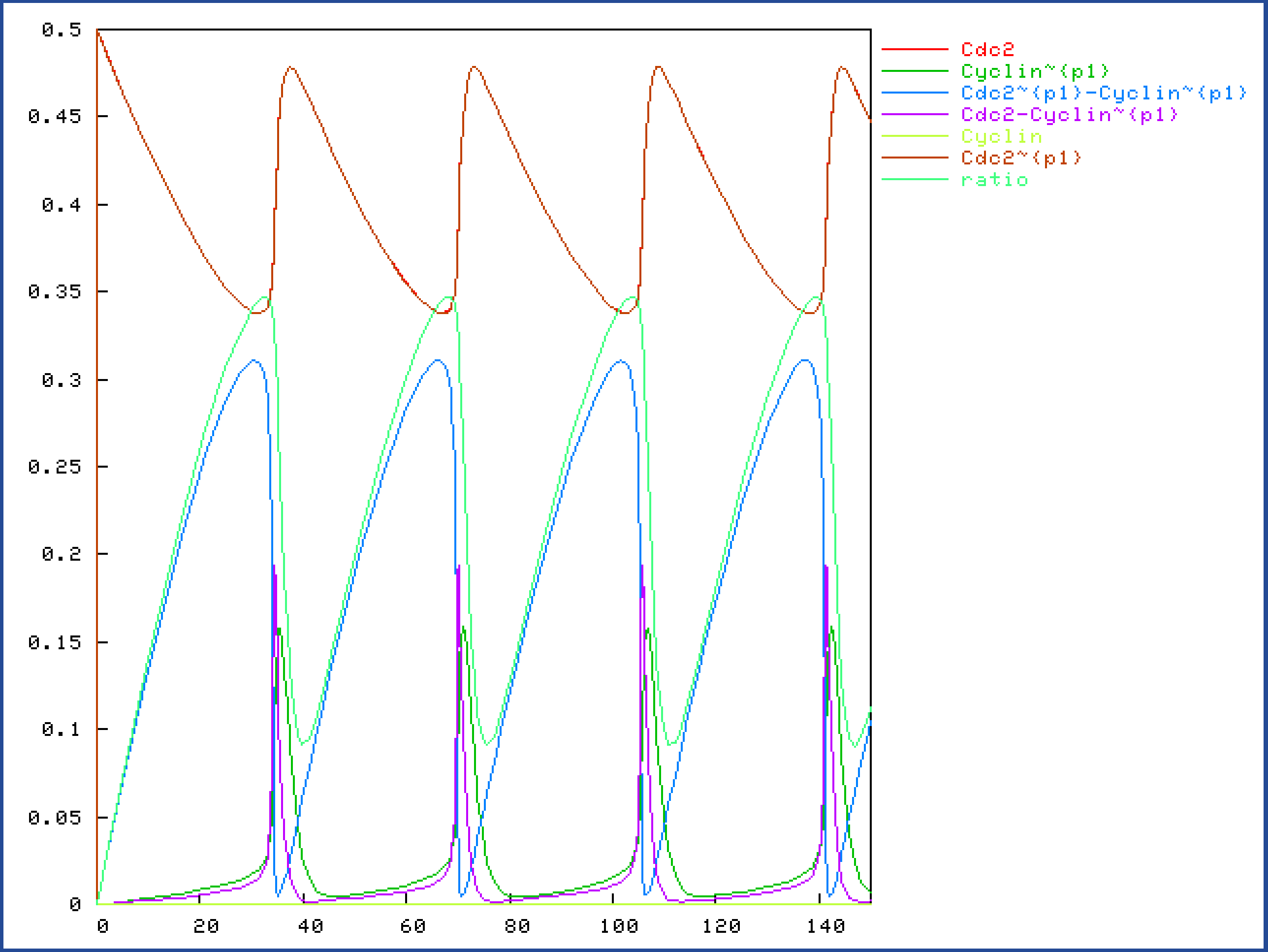
# Interaction Graph



# Boolean Simulation



# Concentration Simulation



# 2. Formalizing Biological Properties in Temporal Logics

Biological property = Temporal Logic Formula

Biological validation = Model-checking

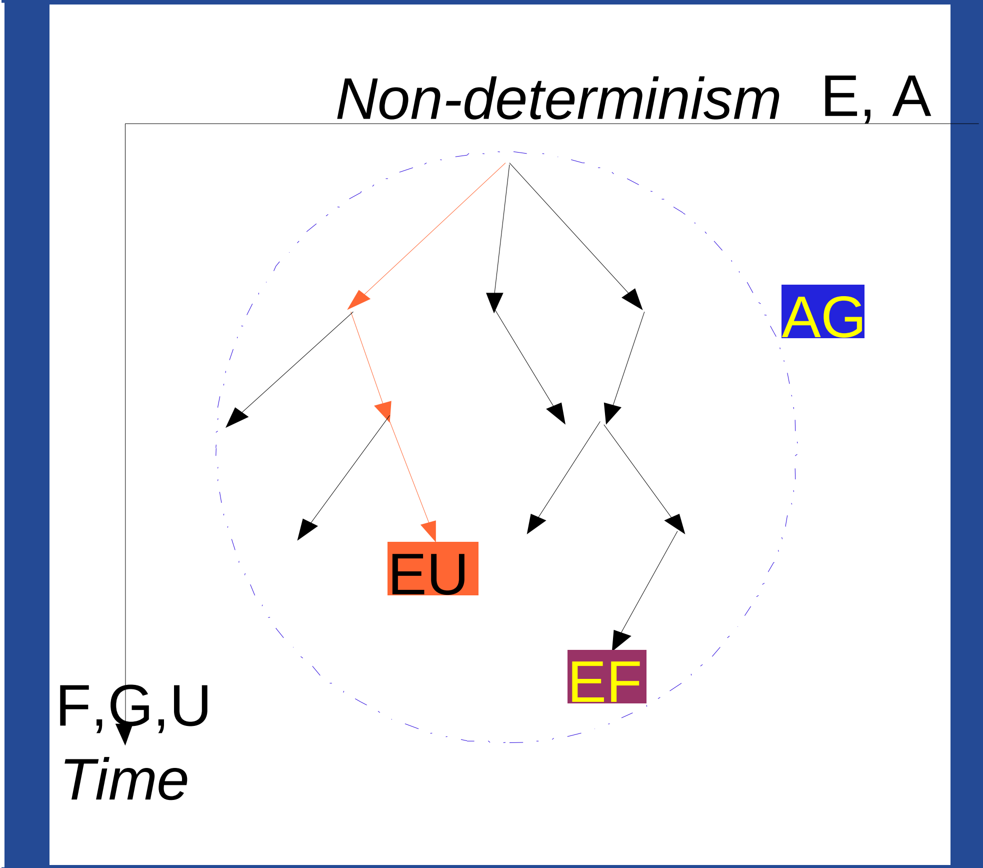
Express properties in:

- Computation Tree Logic CTL for the boolean semantics
- Linear Time Logic with numerical constraints for the concentration semantics
- Probabilistic CTL with numerical constraints for the stochastic semantics

# Computation Tree Logic CTL

Extension of propositional (or first-order) logic with operators for time and choices

| Choice<br>Time | E<br>exists | A<br>always |
|---|---|---|
| X<br>next time | EX(f)<br>$\neg$ *AX($\neg$ f)* | AX(f) |
| F<br>finally | EF(f)<br>$\neg$ *AG($\neg$ f)* | AF(f) |
| G<br>globally | EG(f)<br>$\neg$ *AF($\neg$ f)* | AG(f) |
| U<br>until | E ($f_1$ U $f_2$) | A ($f_1$ U $f_2$) |



# Biological Properties formalized in CTL (1/3)

About *reachability*:

- Can the cell produce some protein P? **`reachable(P)==EF(P)`**

# Biological Properties formalized in CTL (1/3)

About *reachability*:

- Can the cell produce some protein P? **`reachable(P)==EF(P)`**
- Can the cell produce P, Q and not R? **`reachable(P^Q^¬R)`**

# Biological Properties formalized in CTL (1/3)

About *reachability*:

- Can the cell produce some protein P? `reachable(P)==EF(P)`
- Can the cell produce P, Q and not R? `reachable(P^Q^¬R)`
- Can the cell always produce P? `AG(reachable(P))`

# Biological Properties formalized in CTL (1/3)

About *reachability*:

- Can the cell produce some protein P? **reachable(P)==EF(P)**
- Can the cell produce P, Q and not R? **reachable(P^Q^¬R)**
- Can the cell always produce P? **AG(reachable(P))**

About *pathways*:

- Can the cell reach a (partially described) set of states s while passing by another set of states $s_2$? **EF($s_2$^EFs)**

# Biological Properties formalized in CTL (1/3)

About *reachability*:

- Can the cell produce some protein P? **`reachable(P)==EF(P)`**
- Can the cell produce P, Q and not R? **`reachable(P^Q^¬R)`**
- Can the cell always produce P? **`AG(reachable(P))`**

About *pathways*:

- Can the cell reach a (partially described) set of states s while passing by another set of states $s_2$? $\mathbf{EF(s_2\text{^}EFs)}$
- Is it possible to produce P without Q? **`E(¬Q U P)`**

# Biological Properties formalized in CTL (1/3)

About *reachability*:

- Can the cell produce some protein P? **reachable(P)==EF(P)**
- Can the cell produce P, Q and not R? **reachable(P^Q^¬R)**
- Can the cell always produce P? **AG(reachable(P))**

About *pathways*:

- Can the cell reach a (partially described) set of states s while passing by another set of states $s_2$? **EF(**$s_2$**^EFs)**
- Is it possible to produce P without Q? **E(¬Q U P)**
- Is (set of) state $s_2$ a necessary checkpoint for reaching (set of) state s?

**checkpoint(**$s_2$**,s)== ¬E(¬**$s_2$**U s)**

# Biological Properties formalized in CTL (1/3)

About *reachability*:

- Can the cell produce some protein P? **`reachable(P)==EF(P)`**
- Can the cell produce P, Q and not R? **`reachable(P^Q^¬R)`**
- Can the cell always produce P? **`AG(reachable(P))`**

About *pathways*:

- Can the cell reach a (partially described) set of states s while passing by another set of states $s_2$? $\mathtt{EF(s_2\text{^}EFs)}$
- Is it possible to produce P without Q? **`E(¬Q U P)`**
- Is (set of) state $s_2$ a necessary checkpoint for reaching (set of) state s?

  $\mathtt{checkpoint(s_2,s)==\ \neg E(\neg s_2 U\ s)}$

- Is $s_2$ always a checkpoint for s? $\mathtt{AG(\neg s\ \text{->}\ checkpoint(s_2,s))}$

# Biological Properties formalized in CTL (2/3)

About *stationarity*:

- Is a (set of) state s a stable state? `stable(s)== AG(s)`

# Biological Properties formalized in CTL (2/3)

About *stationarity*:

- Is a (set of) state s a stable state? `stable(s)== AG(s)`
- Is s a steady state (with possibility of escaping) ? `steady(s)==EG(s)`

# Biological Properties formalized in CTL (2/3)

About *stationarity*:

- Is a (set of) state s a stable state? `stable(s)== AG(s)`
- Is s a steady state (with possibility of escaping) ? `steady(s)==EG(s)`
- Can the cell reach a stable state s? `EF(stable(s))` `not in LTL`

# Biological Properties formalized in CTL (2/3)

About *stationarity*:

- Is a (set of) state s a stable state? `stable(s)== AG(s)`
- Is s a steady state (with possibility of escaping) ? `steady(s)==EG(s)`
- Can the cell reach a stable state s? `EF(stable(s))` `not in LTL`
- Must the cell reach a stable state s? `AG(stable(s))`

# Biological Properties formalized in CTL (2/3)

About *stationarity*:

- Is a (set of) state s a stable state? **`stable(s)== AG(s)`**
- Is s a steady state (with possibility of escaping) ? **`steady(s)==EG(s)`**
- Can the cell reach a stable state s? **`EF(stable(s))`** **`not in LTL`**
- Must the cell reach a stable state s? **`AG(stable(s))`**
- What are the stable states? Not expressible in CTL. Needs to combine CTL with search

# Biological Properties formalized in CTL (3/3)

About *oscillations*:

- Can the system exhibit a cyclic behavior w.r.t. the presence of P ?

  ```
  oscil(P)== EG((P ⇒ EF ¬P) ^ (¬P ⇒ EF P))
  ```

  (necessary but not sufficient condition without strong fairness)

# Biological Properties formalized in CTL (3/3)

About *oscillations*:

- Can the system exhibit a cyclic behavior w.r.t. the presence of P ?

  ```
  oscil(P)== EG((P ⇒EF ¬P) ^ (¬P ⇒EF P))
  ```

  (necessary but not sufficient condition, needs CTL*)
- Can the system loops between states s and s2 ?

  ```
  loop(P,Q)== EG((s ⇒EF s2) ^ (s2 ⇒EF s))
  ```

# Examples of Properties in the Cell Cycle Model

reachable(Cdc2~{p1})

reachable(Cyclin)

reachable(Cyclin~{p1})

reachable(Cdc2-Cyclin~{p1})

reachable(Cdc2~{p1}-Cyclin~{p1})

oscil(Cdc2)

oscil(Cdc2~{p1})

oscil(Cdc2~{p1}-Cyclin~{p1})

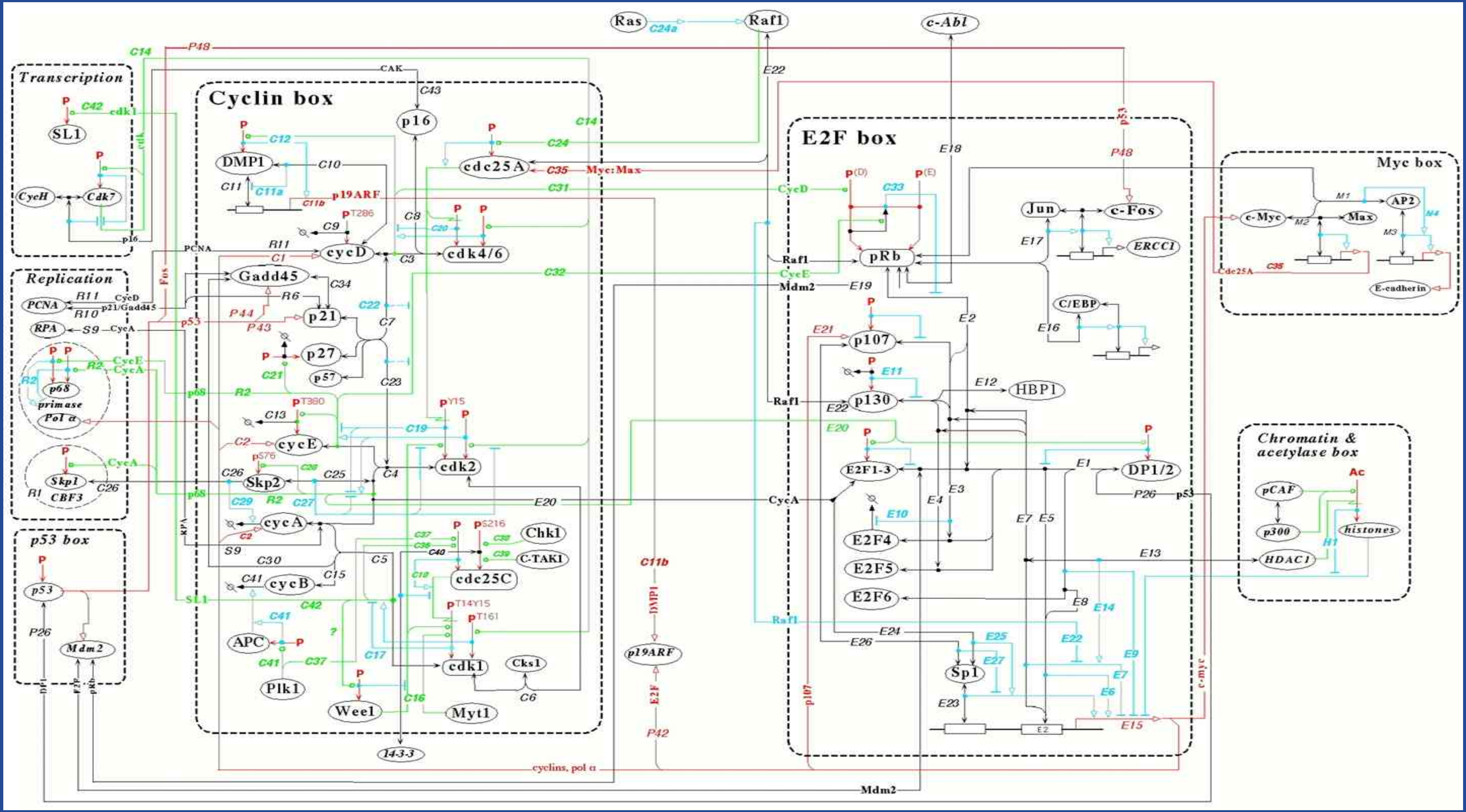
oscil(Cyclin)

AG((!(Cdc2-Cyclin~{p1}))->checkpoint(Cdc2~{p1}-Cyclin~{p1},Cdc2-Cyclin~{p1}))

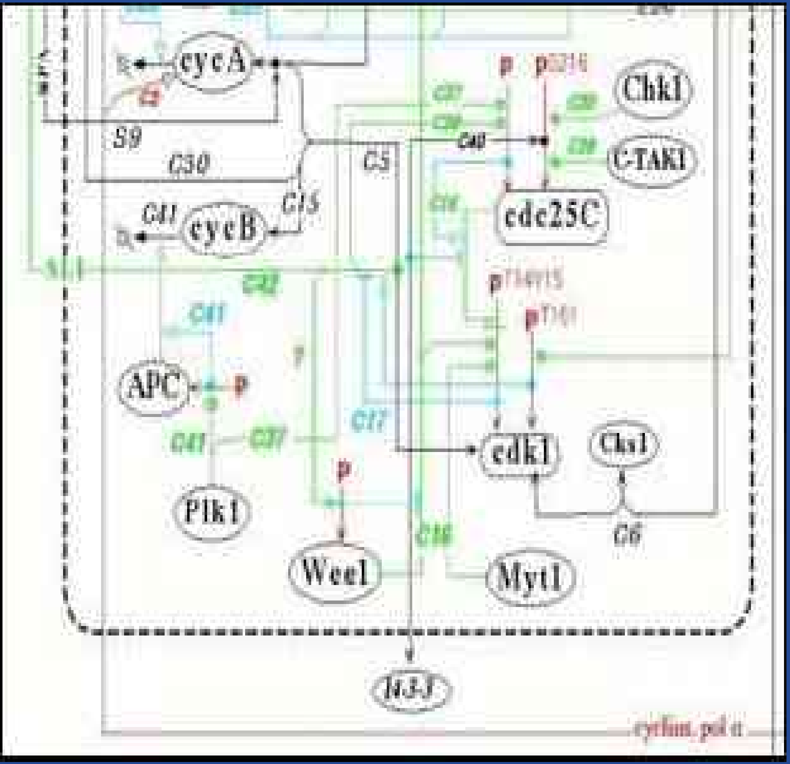
…

Automatically checked / generated by *model-checking* techniques (NuSMV BDD)

# Mammalian Cell Cycle Control Map [Kohn 99]



# Transcription of Kohn’s Map



```
_ =[ E2F13-DP12-gE2 ]=> cycA.
...
cycB =[ APC~{p1} ]=>_.
cdk1~{p1,p2,p3} + cycA => cdk1~{p1,p2,p3}-cycA.
cdk1~{p1,p2,p3} + cycB => cdk1~{p1,p2,p3}-cycB.
...
cdk1~{p1,p3}-cycA =[ Wee1 ]=> cdk1~{p1,p2,p3}-cycA.
cdk1~{p1,p3}-cycB =[ Wee1 ]=> cdk1~{p1,p2,p3}-cycB.
cdk1~{p2,p3}-cycA =[ Myt1 ]=> cdk1~{p1,p2,p3}-cycA.
cdk1~{p2,p3}-cycB =[ Myt1 ]=> cdk1~{p1,p2,p3}-cycB.
...
cdk1~{p1,p2,p3} =[ cdc25C~{p1,p2} ]=> cdk1~{p1,p3}.
cdk1~{p1,p2,p3}-cycA =[ cdc25C~{p1,p2} ]=> cdk1~{p1,p3}-cycA.
cdk1~{p1,p2,p3}-cycB =[ cdc25C~{p1,p2} ]=> cdk1~{p1,p3}-cycB.
```

165 proteins and genes, 500 variables, 800 rules

# Cell Cycle Model-Checking (with BDD NuSMV)

```
biocham: check_reachable(cdk46~{p1,p2}-cycD~{p1}).
   Ei(EF(cdk46~{p1,p2}-cycD~{p1})) is true
biocham: check_checkpoint(cdc25C~{p1,p2}, cdk1~{p1,p3}-cycB).
   Ai(!(E(!(cdc25C~{p1,p2}) U cdk1~{p1,p3}-cycB))) is true
biocham: nusmv(Ai(AG(!(cdk1~{p1,p2,p3}-cycB) -> checkpoint(Wee1, cdk1~{p1,p2,p3}-cycB)))).
   Ai(AG(!(cdk1~{p1,p2,p3}-cycB)->!(E(!(Wee1) U cdk1~{p1,p2,p3}-cycB)))) is false
biocham: why.
-- Loop starts here
   cycB-cdk1~{p1,p2,p3} is present
   cdk7 is present
   cycH is present
   cdk1 is present
   Myt1 is present
   cdc25C~{p1} is present
rule_114 cycB-cdk1~{p1,p2,p3}=[cdc25C~{p1}]=>cycB-cdk1~{p2,p3}.
   cycB-cdk1~{p2,p3} is present
   cycB-cdk1~{p1,p2,p3} is absent
rule_74 cycB-cdk1~{p2,p3}=[Myt1]=>cycB-cdk1~{p1,p2,p3}.
   cycB-cdk1~{p2,p3} is absent
   cycB-cdk1~{p1,p2,p3} is present
```

# Mammalian Cell Cycle Control Benchmark

500 variables, $2^{500}$ states.

BIOCHAM NuSMV model-checker time in sec. [Chabrier et al. TCS 04]

| Initial state G2 | Query: | Time: |
|---|---|---|
| | compiling | 29 |
| Reachability G1 | EF CycE | 2 |
| Reachability G1 | EF CycD | 1.9 |
| Reachability G1 | EF PCNA-CycD | 1.7 |
| Checkpoint<br>for mitosis complex | ¬EF (¬ Cdc25~{Nterm}<br>U Cdk1~{Thr161}-CycB) | 2.2 |
| Oscillation | EG ( (CycA ⇒ EF ¬ CycA) ∧<br>(¬ CycA ⇒ EF CycA)) | 31.8 |

# Quantitative Properties in LTL with Constraints

- Constraints over *concentrations* and *derivatives* as FOL formulae over the reals:
  - [M] > 0.2
  - [M]+[P] > [Q]
  - d([M])/dt < 0

# Quantitative Properties in LTL with Constraints

- Constraints over *concentrations* and *derivatives* as FOL formulae over the reals:
  - [M] > 0.2
  - [M]+[P] > [Q]
  - d([M])/dt < 0
- Linear Time Logic LTL operators for time X, F, U, G
  - F([M]>0.2)
  - FG([M]>0.2)
  - F ([M]>2 & F (d([M])/dt<0 & F ([M]<2 & d([M])/dt>0 & F(d([M])/dt<0))))
  - oscil(M,n) defined as at least n alternances of sign of the derivative
  - Period(A,75)= ∃ t ∃v F(T = t & [A] = v & d([A])/dt > 0 & X(d([A])/dt < 0)
    & F(T = t + 75 & [A] = v & d([A])/dt > 0 & X(d([A])/dt < 0)))

# How to Evaluate a Constraint LTL Formula ?

- Consider the ODE’s of the concentration semantics dX/dt = f(X)

# How to Evaluate a Constraint LTL Formula ?

- Consider the ODE’s of the concentration semantics dX/dt = f(X)

- Numerical integration methods produce a (clever) discretization of time (adaptive step size Runge-Kutta or Rosenbrock method for stiff syst.)

- The trace is a linear Kripke structure:

  $(t_0,X_0,dX_0/dt), (t_1,X_1,dX_1/dt), \ldots, (t_n,X_n,dX_n/dt).$

  over concentrations and their derivatives at discrete time points

- Evaluate the formula on that Kripke structure with a model checking alg.

# Simulation-Based Constraint LTL Model Checking

Hypothesis 1: the initial state is completely known

Hypothesis 2: the formula can be checked over a finite period of time [0,T]

4. Run the numerical integration from 0 to T
   producing values at a finite sequence of time points
5. Iteratively label the time points with the *sub-formulae* of $\phi$ that are true:

   Add $\phi$ to the time points where a FOL formula $\phi$ is true,

   Add F $\phi$ (X $\phi$) to the (immediate) previous time points labeled by $\phi$,

   Add $\phi 1$ U $\phi 2$ to the predecessor time points of $\phi 2$ while they satisfy $\phi 1$,

   (Add G $\phi$ to the states satisfying $\phi$ until T)

Model checker and numerical integration methods implemented in Prolog

# 3. Learning Parameters from Temporal Properties

```
biocham: learn_parameter([k3,k4],[(0,200),(0,200)],20,
                    oscil(Cdc2-Cyclin~{p1},3),150).
```

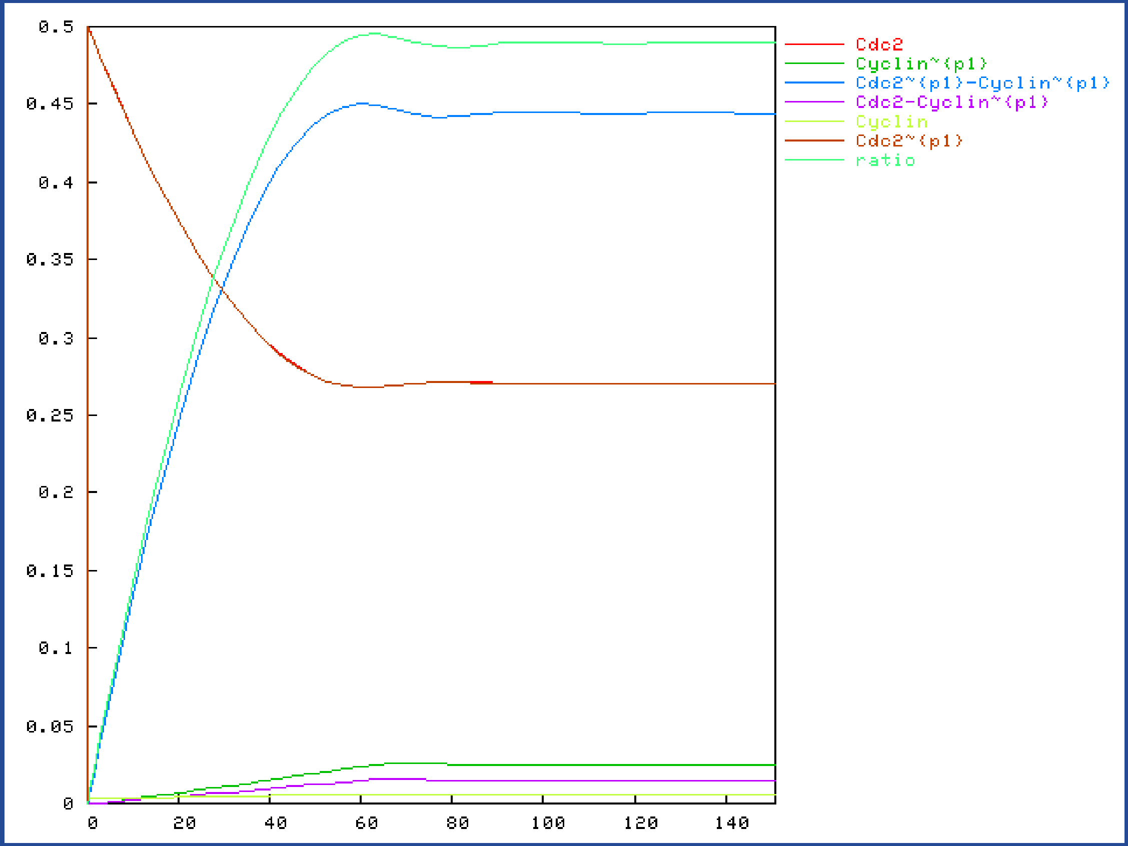
# Learning Parameters from Temporal Properties

biocham: learn_parameter([k3,k4],[(0,200),(0,200)],20,
oscil(Cdc2-Cyclin~{p1},3),150).

First values found :

parameter(k3,10).

parameter(k4,70).



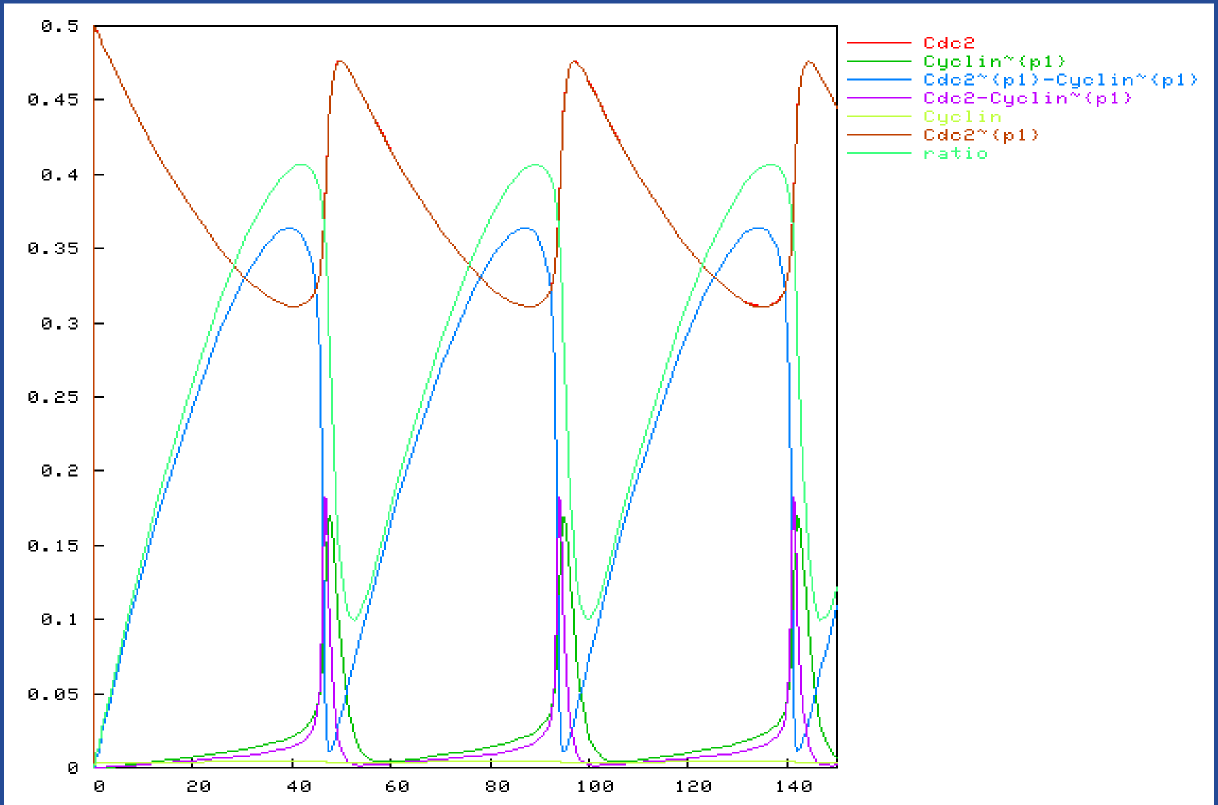
# Learning Parameters from Temporal Properties

biocham: learn_parameter([k3,k4],[(0,200),(0,200)],20,
oscil(Cdc2-Cyclin~{p1},3) & F([Cdc2-Cyclin~{p1}]>0.15), 150).

First values found :

parameter(k3,10).

parameter(k4,120).



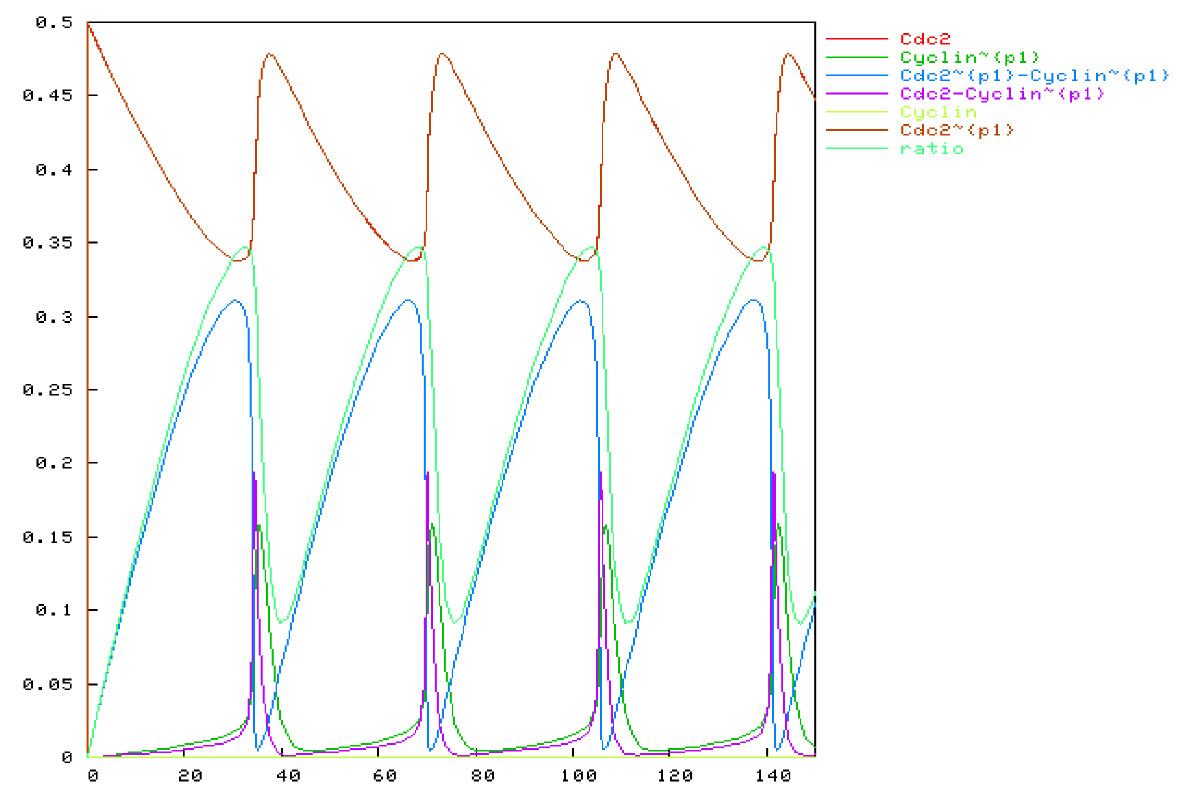
# Learning Parameters from LTL Specification

biocham: learn_parameter([k3,k4],[(0,200),(0,200)],20,
period(Cdc2-Cyclin~{p1},35), 150).

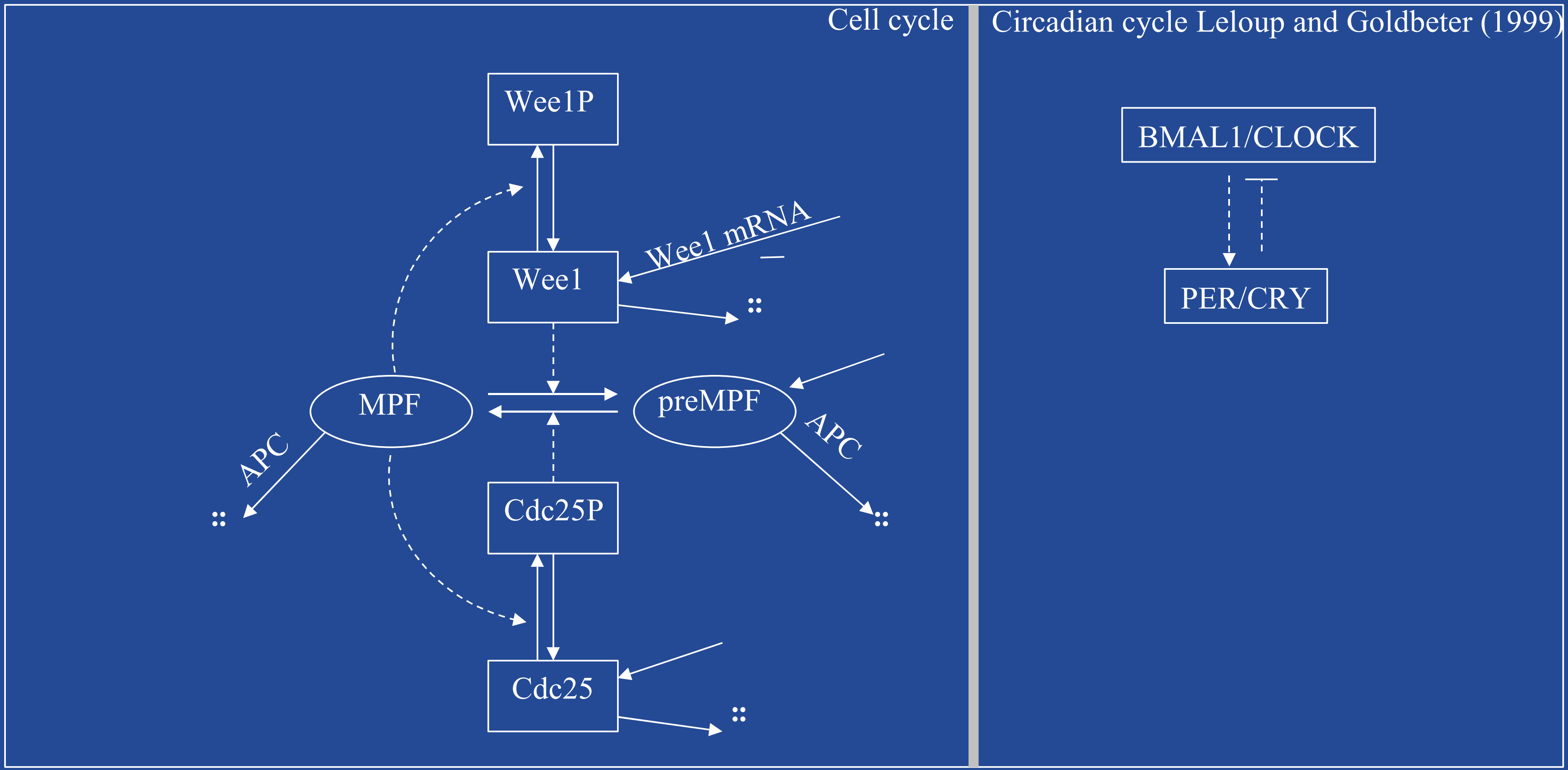
First values found:

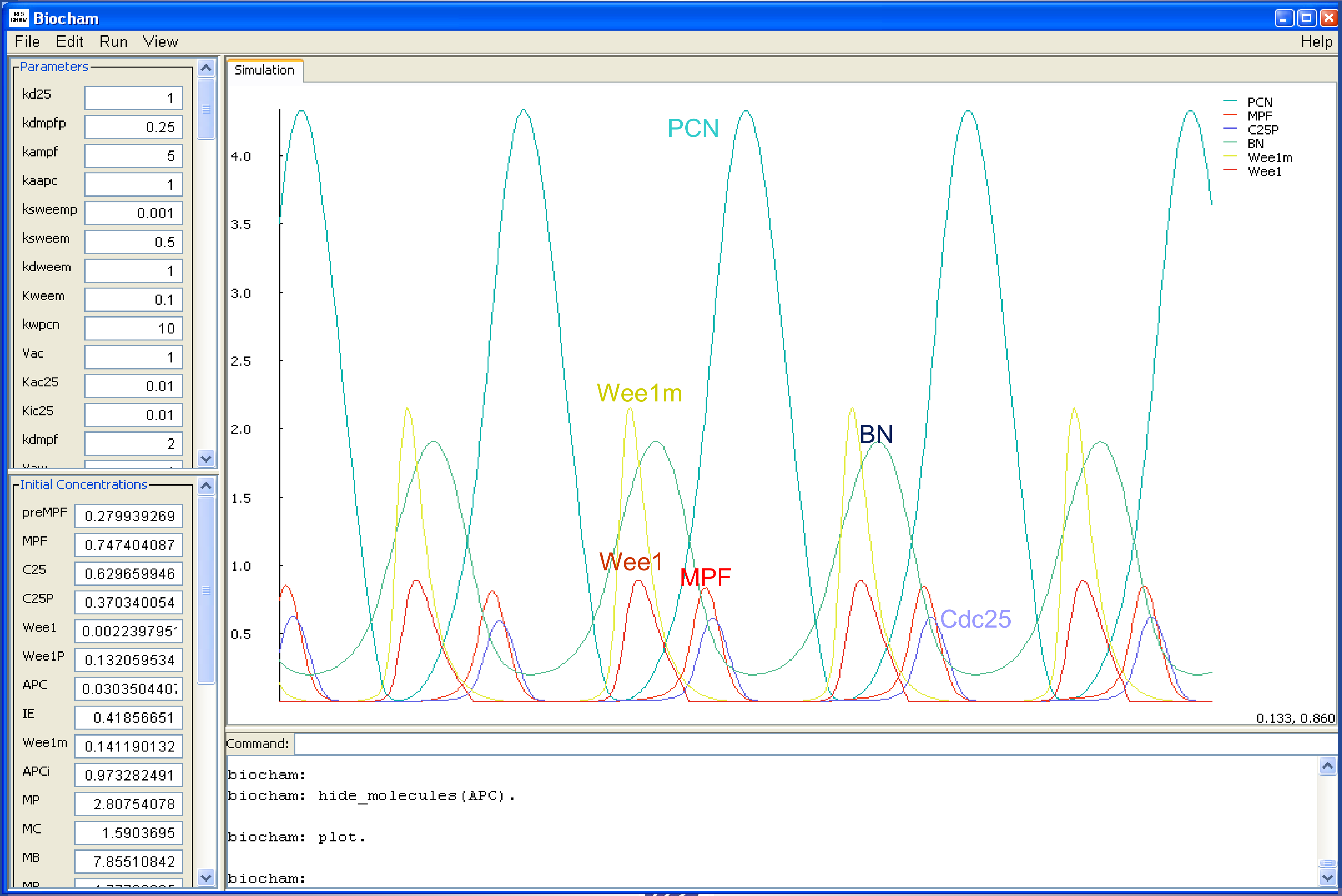
parameter(k3,10).

parameter(k4,280).



# Linking the Cell and Circadian Cycles through Wee1





Parameters

| | |
|---|---|
| kd25 | 1 |
| kdmpfp | 0.25 |
| kampf | 5 |
| kaapc | 1 |
| ksweemp | 0.001 |
| ksweem | 0.5 |
| kdweem | 1 |
| Kweem | 0.1 |
| kwpcn | 10 |
| Vac | 1 |
| Kac25 | 0.01 |
| Kic25 | 0.01 |
| kdmpf | 2 |

Initial Concentrations

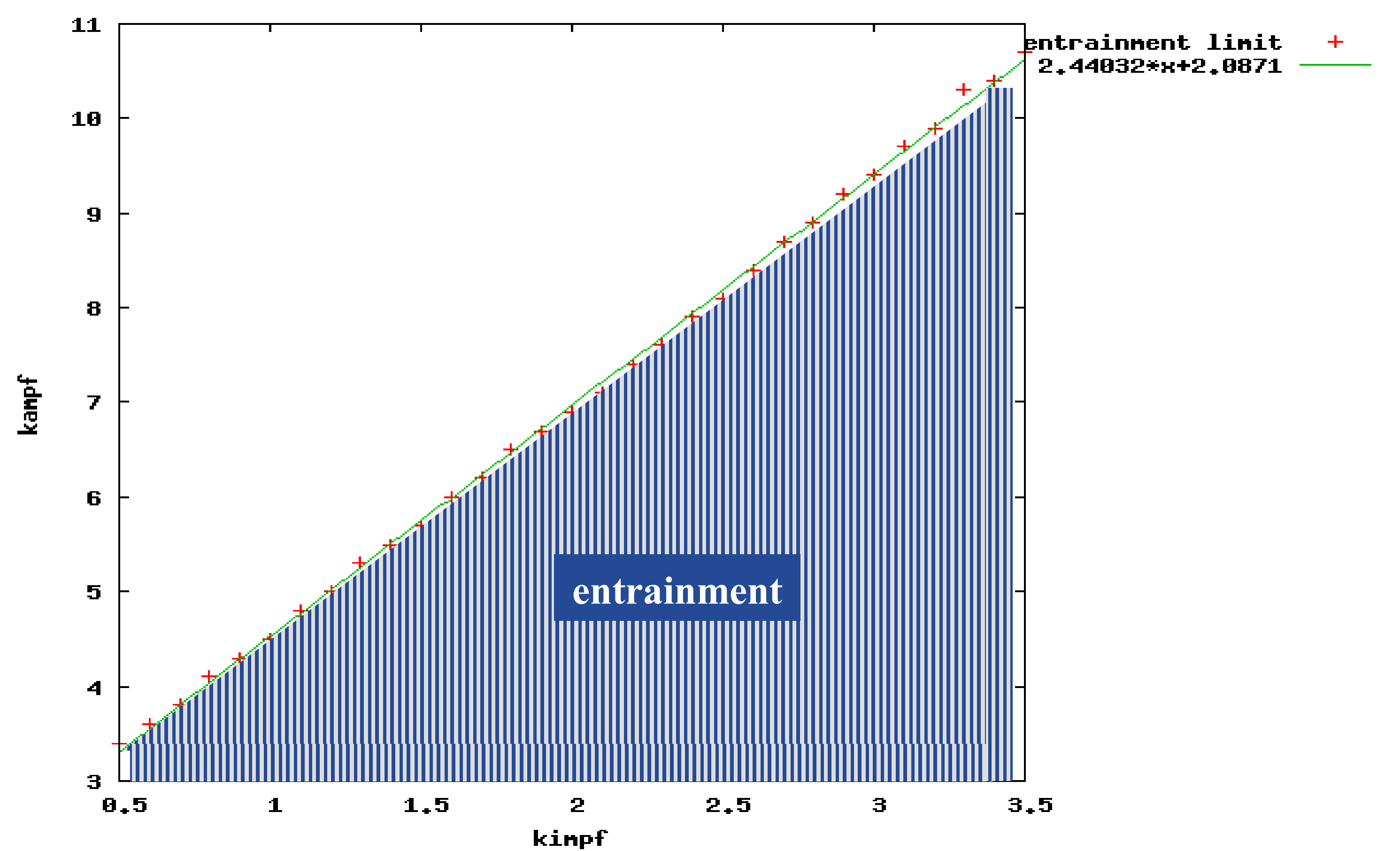
| | |
|---|---|
| preMPF | 0.279939269 |
| MPF | 0.747404087 |
| C25 | 0.629659946 |
| C25P | 0.370340054 |
| Wee1 | 0.002239795 |
| Wee1P | 0.132059534 |
| APC | 0.030350440 |
| IE | 0.41856651 |
| Wee1m | 0.141190132 |
| APCi | 0.973282491 |
| MP | 2.80754078 |
| MC | 1.5903695 |
| MB | 7.85510842 |

```
biocham:
biocham: hide_molecules(APC).

biocham: plot.

biocham:
```

# Condition on Wee1/Cdc25 for the Entrainment in Period



# 4. Learning Rules from Temporal Properties

Given

- a BIOCHAM model (background knowledge)
- a set of properties formalized in temporal logic

learn

- revisions of the reaction model, i.e. rules to delete and rules to add such that the revised model satisfies the properties

# Model Revision from Temporal Properties

- *Background knowledge T*: BIOCHAM model
  - reaction rule language: complexation, phosphorylation, …
- *Examples* φ: biological properties formalized in temporal logic language
  - Reachability
  - Checkpoints
  - Stable states
  - Oscillations
- *Bias R*: Reaction rule patterns or parameter ranges
  - Kind of rules to add or delete

Find a revision T' of T such that *T'* |= φ

# Positive and Negative CTL Formulae

Def. An ECTL (positive) formula is a CTL formula with no occurrence of A (nor negative occurrence of E).

Def. An ACTL (negative) formula is a CTL formula with no occurrence of E (nor negative occurrence of A).

Let K = (S,R,L) and K’ = (S,R’,L) be two Kripke structures such that $R \subset R'$

Proposition For any ECTL formula $\phi$, if K’ $|\neq \phi$ then K $|\neq \phi$.

Proposition For any ACTL formula $\phi$, if K $|\neq \phi$ then K’ $|\neq \phi$.

# Model Revision Algorithm

General idea of constraint programming: replace a *generate-and-test* algorithm by a *constrain-and-generate* algorithm.

Anticipate whether one has to *add or remove a rule*?

- *Positive* ECTL formula: if false, remains false after removing a rule
  - EF(φ) where φ is a boolean formula (pure state description)
  - Oscil(φ)
- *Negative* ACTL formula: if false, remains false after adding a rule
  - AG(φ) where φ is a boolean formula,
  - Checkpoint(a,b): ¬E(¬aUb)
  - Remove a rule on the path given by the model checker (why command)
- Unclassified CTL formulae

# Model Revision Algorithm Steps

Initial state: $<(0, 0, 0), (E,U,A), R>$

E transition: $<(\underline{E},\underline{U},\underline{A}), (E\cup\{e\},U,A), R> \rightarrow <(\underline{E}\cup\{e\},\underline{U},\underline{A}), (E,U,A),R>$ if $R \models e$

E' transition: $<(\underline{E},\underline{U},\underline{A}), (E \cup\{e\},U,A), R> \rightarrow <(\underline{E} \cup\{e\},\underline{U},\underline{A}), (E,U,A),R \cup \{r\}>$

if $R \mid\neq e$ and $\forall\ f \in\{e\} \cup \underline{E} \cup \underline{U} \cup \underline{A}, K \cup \{r\} \models f$

# Model Revision Algorithm Steps

Initial state: $<(0, 0, 0), (E,U,A), R>$

E transition: $<(\underline{E},\underline{U},\underline{A}), (E\cup\{e\},U,A), R> \rightarrow <(\underline{E}\cup\{e\},\underline{U},\underline{A}), (E,U,A),R>$ if $R \models e$

E’ transition: $<(\underline{E},\underline{U},\underline{A}), (E \cup\{e\},U,A), R> \rightarrow <(\underline{E} \cup\{e\},\underline{U},\underline{A}), (E,U,A),R \cup \{r\}>$

if $R \not\models e$ and $\forall f \in \{e\} \cup \underline{E} \cup \underline{U} \cup \underline{A}, K \cup \{r\} \models f$

U transition: $<(\underline{E},\underline{U},\underline{A}), (0,U \cup\{u\},A), R > \rightarrow <(\underline{E},\underline{U} \cup \{u\},\underline{A}), (0,U,A),R>$ if $R \models u$

U’ transition: $<(\underline{E},\underline{U},\underline{A}), (0,U \cup\{u\},A), R > \rightarrow <(\underline{E},\underline{U} \cup\{u\},\underline{A}), (0,U,A),R \cup \{r\}>$

if $R \not\models u$ and $\forall f \in \{u\} \cup \underline{E} \cup \underline{U} \cup \underline{A}, R \cup \{r\} \models f$

U” transition: $<(\underline{E},\underline{U},\underline{A}), (0,U \cup \{u\},A), R \cup Re > \rightarrow <(\underline{E},\underline{U} \cup\{u\},\underline{A}),(0,U,A), R>$

if $K, si \not\models u$ and $\forall f \in \{u\} \cup \underline{E} \cup \underline{U} \cup \underline{A}, R \models f$

# Model Revision Algorithm Steps

Initial state: $<(0, 0, 0), (E,U,A), R>$

E transition: $<(\underline{E},\underline{U},\underline{A}), (E\cup\{e\},U,A), R> \rightarrow <(\underline{E}\cup\{e\},\underline{U},\underline{A}), (E,U,A),R>$ if $R \models e$

E’ transition: $<(\underline{E},\underline{U},\underline{A}), (E \cup\{e\},U,A), R> \rightarrow <(\underline{E} \cup\{e\},\underline{U},\underline{A}), (E,U,A),R \cup \{r\}>$

if $R \not\models e$ and $\forall f \in\{e\} \cup \underline{E} \cup \underline{U} \cup \underline{A}, K \cup \{r\} \models f$

U transition: $<(\underline{E},\underline{U},\underline{A}), (0,U \cup\{u\},A), R > \rightarrow <(\underline{E},\underline{U} \cup \{u\},\underline{A}), (0,U,A),R>$ if $R \models u$

U’ transition: $<(\underline{E},\underline{U},\underline{A}), (0,U \cup\{u\},A), R > \rightarrow <(\underline{E},\underline{U} \cup\{u\},\underline{A}), (0,U,A),R \cup \{r\}>$

if $R\not\models u$ and $\forall f \in \{u\} \cup \underline{E} \cup \underline{U} \cup \underline{A}, R \cup \{r\} \models f$

U” transition: $<(\underline{E},\underline{U},\underline{A}), (0,U \cup \{u\},A), R \cup Re > \rightarrow <(\underline{E},\underline{U} \cup\{u\},\underline{A}),(0,U,A), R>$

if $K, si\not\models u$ and $\forall f \in\{u\} \cup \underline{E} \cup \underline{U} \cup \underline{A}, R \models f$

A transition: $<(\underline{E},\underline{U},\underline{A}), (0, 0,A \cup\{a\}), R > \rightarrow <(\underline{E},\underline{U},\underline{A} \cup\{a\}), (Ep,Up,A),R>$ if $R \models a$

A’ transition: $<(\underline{E}\cup\underline{Ep},\underline{U}\cup\underline{Up},\underline{A}),(0,0,A\cup\{a\}), R\cup Re>\rightarrow<(\underline{E},\underline{U},\underline{A}\cup\{a\}),(Ep,Up,A),R>$
if $R\not\models a$, $\forall f \in\{u\}$ [ $E \cup U \cup A, R \models f$ and $Ep \cup Up$ is the set of formulae no longer satisfied after the deletion of the rules in Re.

# Termination and Correctness

Proposition The model revision algorithm terminates.

Proposition If the terminal configuration is of the form <(E,U,A),(0,0,0),R> then the model R satisfies the initial CTL specification.

Proof The termination of the algorithm is proved by considering the lexicographic ordering over the couple < a, n > where a is the number of unsatisfied ACTL formulae, and n is the number of unsatisfied ECTL and UCTL formulae. Each transition strictly decreases either a, or lets a unchanged and strictly decreases n.

The correction of the algorithm comes from the fact that each transition maintains only true formulae in the satisfied set, and preserves the complete CTL specification in the union of the satisfied set and the untreated set.

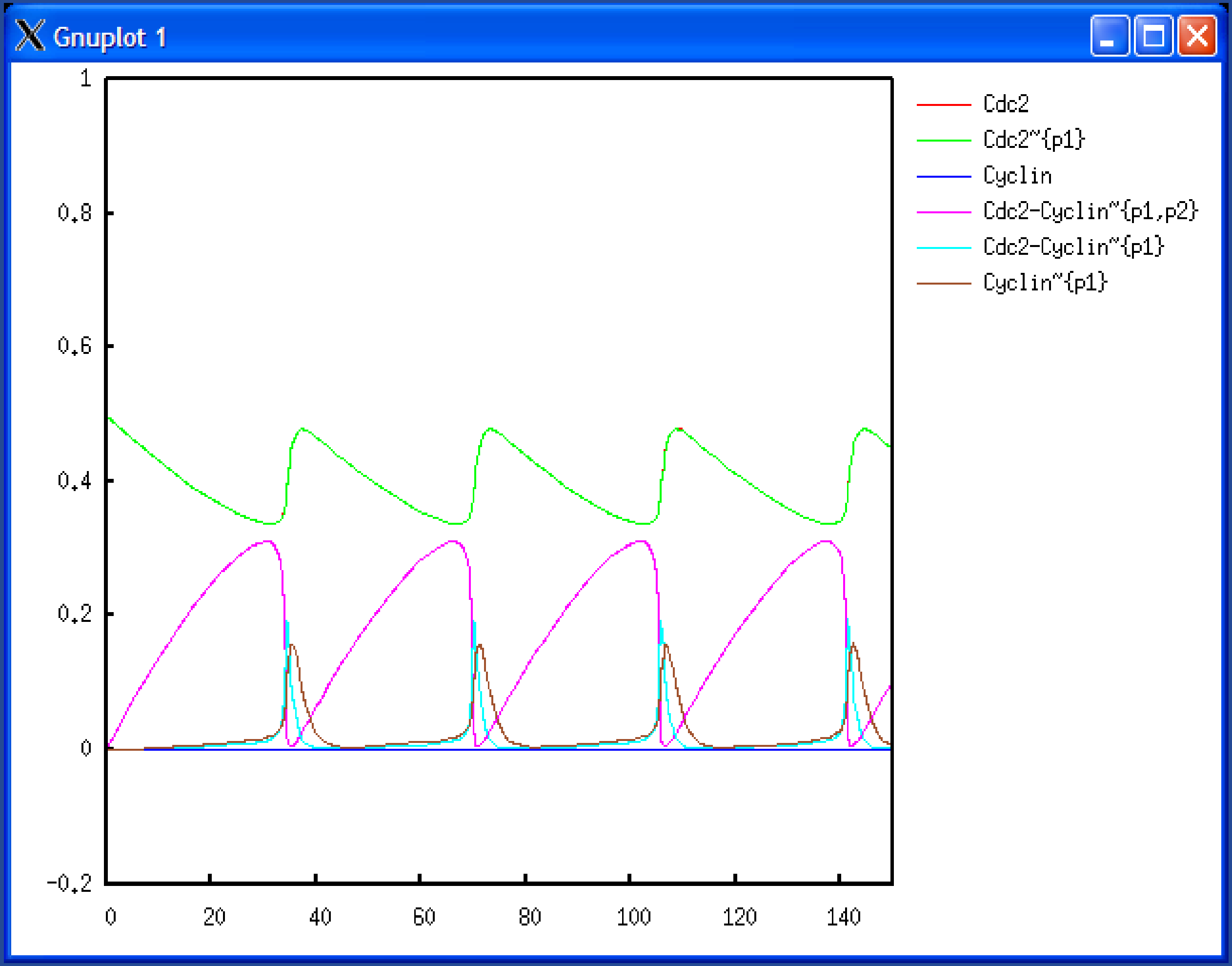
# Incompleteness

Two reasons:

3) The satisfaction of ECTL and UCTL formula is searched by adding only one rule to the model (transition E' and U')

5) The Kripke structure associated to a Biocham set of rules adds loops on terminal states. Hence adding or removing a rule may have an opposite deletion or addition of the loops.

# Cell Cycle Kinetic Model [Tyson 91]



# Generation of True CTL Properties

```
Ei(EF(Cdc2)).
Ei(EF(!(Cdc2))).
Ai(AG(Cdc2->EF(!(Cdc2))&(!(Cdc2)->EF(Cdc2)))).
Ei(EF(Cdc2~{p1})).
Ei(EF(!(Cdc2~{p1}))).
Ai(AG(Cdc2~{p1}->EF(!(Cdc2~{p1}))&(!(Cdc2~{p1})->EF(Cdc2~{p1})))).
Ai(AG(!(Cdc2~{p1})->!(E(!(Cdc2) U Cdc2~{p1})))).
Ei(EF(Cyclin)).
Ei(EF(!(Cyclin))).
Ai(AG(Cyclin->EF(!(Cyclin))&(!(Cyclin)->EF(Cyclin)))).
Ei(EF(Cdc2-Cyclin~{p1,p2})).
Ei(EF(!(Cdc2-Cyclin~{p1,p2}))).
Ai(AG(Cdc2-Cyclin~{p1,p2}->EF(!(Cdc2-Cyclin~{p1,p2}))&(!(Cdc2-Cyclin~{p1,p2})->EF(Cdc2-Cyclin~{p1,p2})))).
Ei(EF(Cdc2-Cyclin~{p1})).
Ei(EF(!(Cdc2-Cyclin~{p1}))).
Ai(AG(Cdc2-Cyclin~{p1}->EF(!(Cdc2-Cyclin~{p1}))&(!(Cdc2-Cyclin~{p1})->EF(Cdc2-Cyclin~{p1})))).
Ai(AG(!(Cdc2-Cyclin~{p1})->!(E(!(Cdc2-Cyclin~{p1,p2}) U Cdc2-Cyclin~{p1})))).
Ei(EF(Cyclin~{p1})).
Ei(EF(!(Cyclin~{p1}))).
Ai(AG(Cyclin~{p1}->EF(!(Cyclin~{p1}))&(!(Cyclin~{p1})->EF(Cyclin~{p1})))).
Ai(AG(!(Cyclin~{p1})->!(E(!(Cdc2-Cyclin~{p1}) U Cyclin~{p1})))).
```

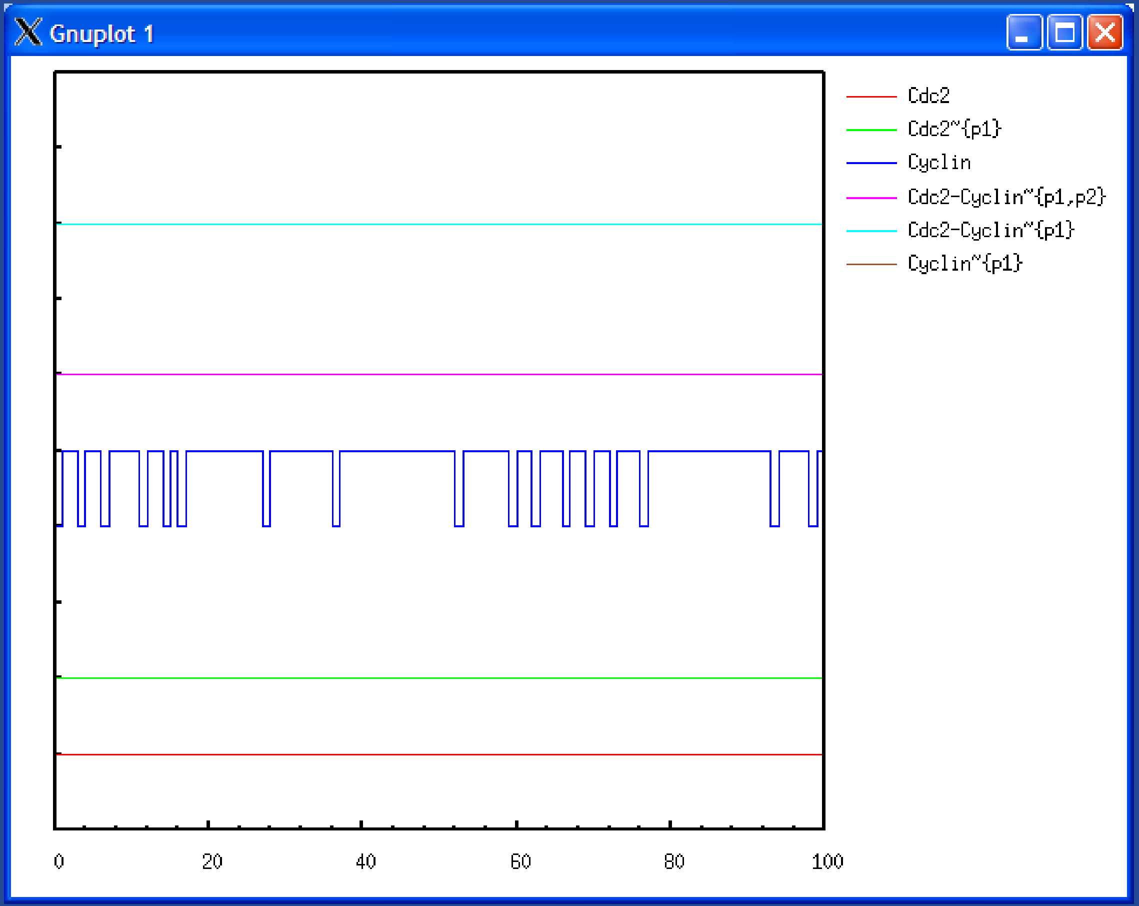
# Model Revision

biocham: delete_rules(Cdc2 => Cdc2~{p1}).

biocham: check_all.

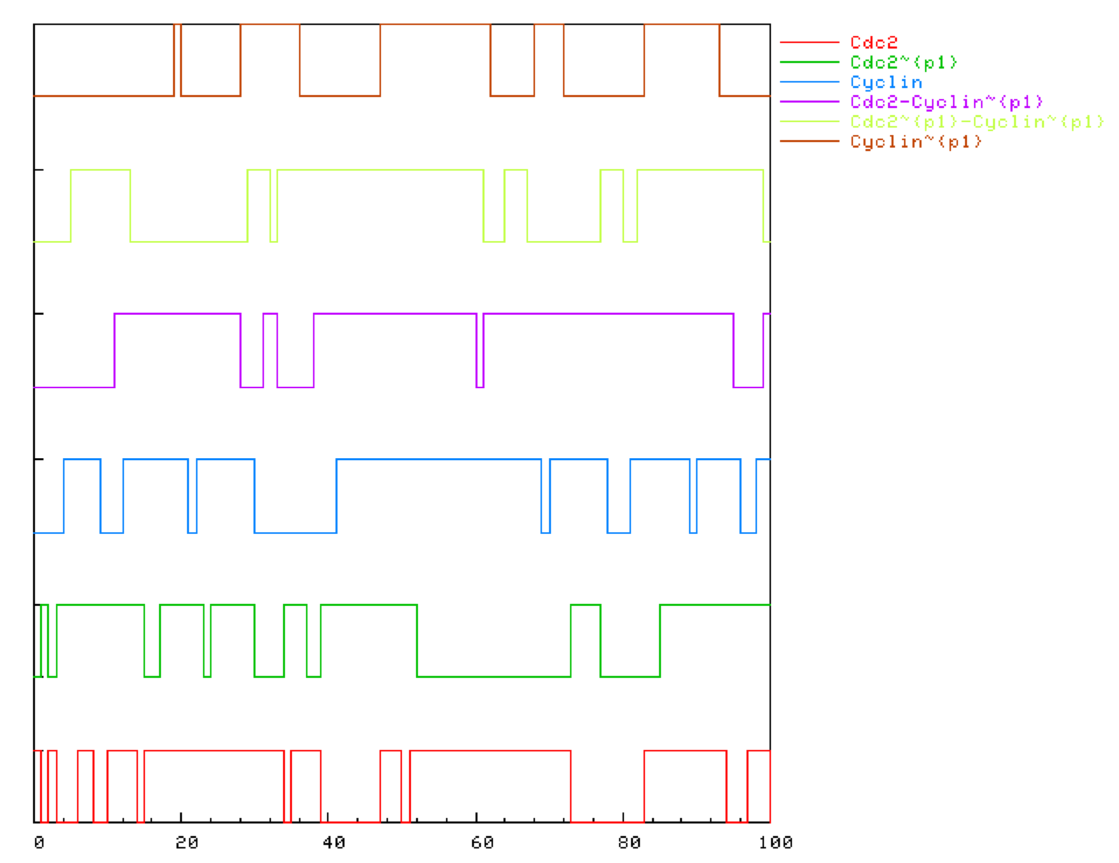
First formula not satisfied

Ei(EF(Cdc2-Cyclin~{p1}))



# Model Revision

```
biocham: revise_model.
Rules to delete:
Rules to add:
Cdc2 => Cdc2~{p1}.
```



```
biocham: learn_one_addition.
(1) Cdc2 => Cdc2~{p1}.
(2) Cdc2 =[Cdc2]> Cdc2~{p1}.
(3) Cdc2 =[Cyclin]> Cdc2~{p1}.
```

# Conclusions

- Temporal logic with constraints is powerful enough to express both *qualitative and quantitative* biological properties of systems

- Three levels of abstraction implemented in BIOCHAM :

| | |
|---|---|
| Boolean semantics | CTL formulas *(rule learning)* |
| Differential semantics | LTL with constraints over reals *(parameter search)* |
| Stochastic semantics | Probabilistic CTL with integer constraints *(inefficient)* |

- Learning from entailment in temporal logic (by model checking)

Theory revision H(M) |= e

# Collaborations

STREP APrIL2 : Applications of probabilistic inductive logic programming

Luc de Raedt, Univ. Freiburg, Stephen Muggleton, IC London,…

- Learning in a probabilistic logic setting

NoE REWERSE : semantic web, François Bry, Münich, Rolf Backofen,

- Connecting Biocham to gene and protein ontologies

STREP TEMPO : Cancer chronotherapies, INSERM Villejuif, Francis Lévi

- Coupled models of cell cycle, circadian cycle, cytotoxic drugs.

INRA Tours : FSH signalling, Eric Reiter