Facing complexity and uncertainty in model checking of biological systems

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa, Italy

SEFM School – CNR Pisa – September 8, 2010

Introduction: Systems Biology

> *"Systems Biology is a comprehensive quantitative analysis of the manner in which all the components of a biological system interact functionally over time."*
>
> Alan Aderem, *Systems Biology: Its Practice and Challenges*. Cell 121, 511-513 (2005)

The aim of current research in Systems Biology is to integrate the knowledge about single constituents of living organisms into *system view*.

The two main approaches to biological systems modelling:

- **Biomath** Models are given as *differential equations* (or recurrence equations), and are studied by applying *analytical* and *numerical* techniques.
- **Bioinfo** Biological systems are modelled as *stochastic concurrent systems* and analyzed by *simulation* and *model checking*.

The application of such tools is limited to small, well known pathways

Introduction: The need of approximations

> *"Biological processes are profoundly complex, containing hundreds or thousands of component interactions. This leads to uncertainty i.e., precise information about probabilities, pathway structure, rate constants and similar parameters, is often unknown. Further, it is often impossible to assign precise point probabilities to each of the myriad constituents of an intricate biological pathway."*
>
> Iyengar M.S., McGuire M.F., *Imprecise and Qualitative Probability in Systems Biology*, ICSB, October 1-6, 2007

The two main problems in biological systems modelling are:

- complexity of the systems
- unavailability of (precise) kinetic parameters

Hence, the need of constructing approximated models

- by means (if possible) of conservative abstractions

Introduction: two examples of approaches

We propose two approaches for the construction and analysis of models with approximations:

- Modular verification
  - PhD thesis (in progress) by Peter Drábik
    Dipartimento di Informatica, Università di Pisa
- Probabilistic model checking with uncertainty on kinetic rates
  - PhD thesis (in progress) by Guido Scatena
    IMT Lucca Institute for Advanced Studies

Outline of the talk

Abstract Interpretation (1)

Abstract Interpretation is:

- a static analysis technique
- aimed at allowing a property of the possible behaviours of a complex system to be verified
- on an abstraction of these behaviours dealing only with aspects related with the considered property

Abstract Interpretation (2)

Verification of a safety property (nothing bad happens) of a computer program by Abstract Interpretation consists in:

- considering an abstract semantics, that is a superset of the (concrete) semantics of the program
- the abstract semantics has to be sound: it must cover all possible concrete executions
- on the other hand it has to allow for a more efficient verification of the property
- if the property holds in the abstract semantics it holds also in the concrete one

Abstract Domains

A typical way of defining an abstract semantics is by means of abstract domains for program variables:

- Example of concrete domain:
  - integers $\ldots, -2, -1, 0, 1, 2, \ldots$
- Examples of abstract domains:
  - signs $\mathit{Neg}, \mathit{Pos}, 0, \mathit{Unknown}$
  - intervals $[-5, -1], [1, 5], \ldots$
  - polyedra $\{x \geq 0, y \geq x + 1\}, \ldots$

An element of an abstract domain represent a set of elements of the concrete domain

Abstract Domains

New semantic rules have to be defined for the considered abstract domain:

- Semantic rule for multiplication of integers: $3 \times -2 \rightarrow -6$
- Semantic rule for multiplication of $\{Neg, 0, Pos, Unknown\}$: $Pos \times Neg \rightarrow Neg$
- Semantic rule for multiplication of intervals: $[1, 5] \times [-5, -1] \rightarrow [-25, -1]$

The use of abstract domains allows the state space of program semantics to be reduced

In many cases this makes it feasible to verify properties

Abstractions and LTS

In context of LTS semantics, an abstraction is a function mapping states and transition of the (concrete) semantics into state and transitions of an abstract LTS semantics

- In the case of programs the function can be inferred from the definition of abstract domains and the corresponding operations
- In the case of models of biological systems it is often necessary to define "ad-hoc" abstractions

Required properties of an abstract LTS semantics:

- soundness: every trace in the concrete semantics must have a corresponding abstract trace
- precision: the abstract semantics should be precise enough to avoid "false alarms"
- simplicity: the abstract semantics should be as abstract as possible to make analysis feasible

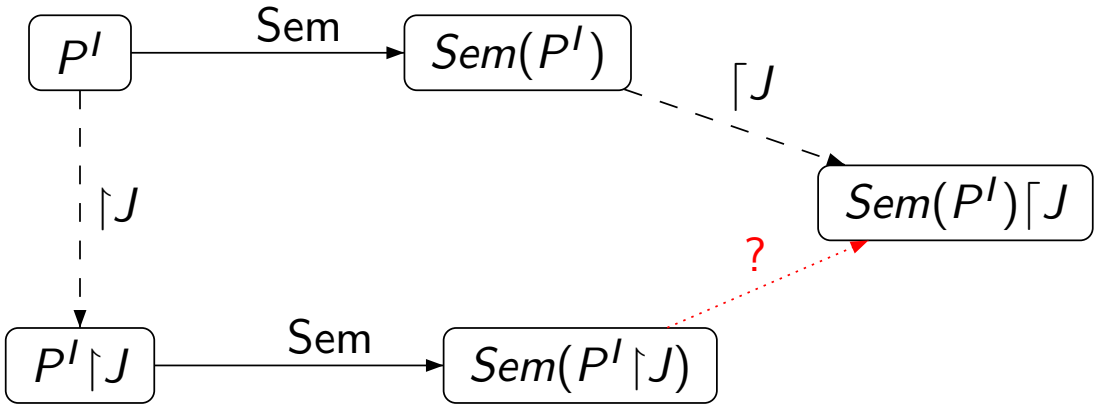
Again: this may allow verification of safety properties (properties of all traces).

Outline of the talk

Motivation

- Goal – verify properties of subsystems, and infer that these hold in the complete system
- Class of properties identified by Grumberg *et al.* as ACTL – the universal fragment of CTL
- Addressed by Attie for verification and synthesis of concurrent programs
  - Synchronisation skeletons – move of a component may depend on the states of other components
  - Not suitable for describing biological systems
- Synchronised moves of more components are crucial to model biological phenomena
- Extension – sync-programs – enable synchronised move of an arbitrary number of automata

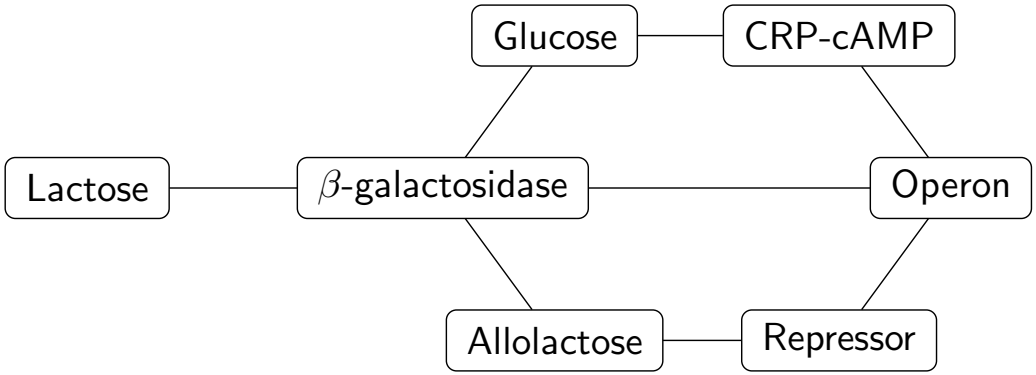
Modular verification – principle



- Define: prog. language, semantics, projections
- Show: $Sem(P^I \restriction J) \sqsupseteq Sem(P^I)\lceil J$

- Computation is preserved (infinite path)
- Properties talking about all computations (ACTL)
- Positive answer carried over to the whole system

Interaction graph



A system is made of components.

Definition

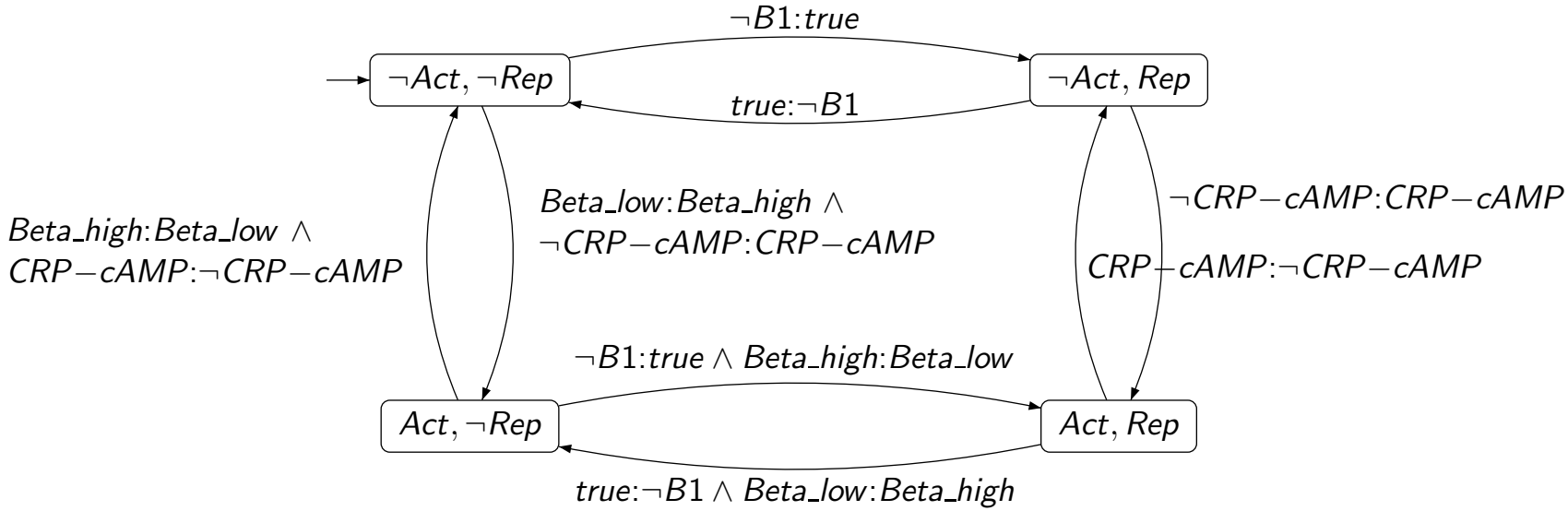
Interaction graph I

- nodes are components of the system
- edges represent possible (direct) interactions

Each component $i \in I$ is associated with a set of atomic propositions AP_i.

- These sets are pairwise disjoint

Sync-automaton



Each component of the system is described by a sort of finite state automaton.

Definition

Sync-automaton $P_i^!$

- states – mappings of AP_i to $\{true, false\}$
- transitions (called moves) with conditions – $s_i \xrightarrow{c_i} t_i$

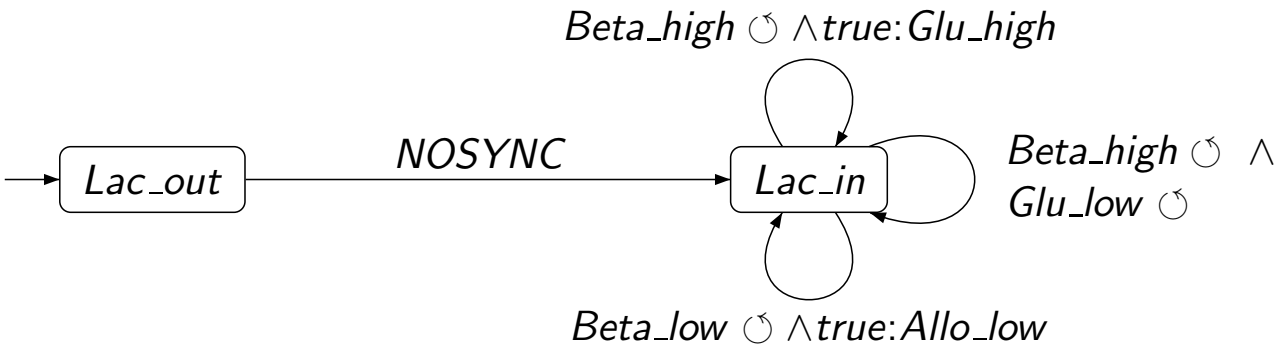
Synchronisation condition c_i is of the form $\wedge_{j \in L} A_j{:}B_j$ with $A_j, B_j \in AP_j$.

Synchronisation conditions

Conjunction of pairs of atomic propositions $\wedge_{j \in L} A_j{:}B_j$

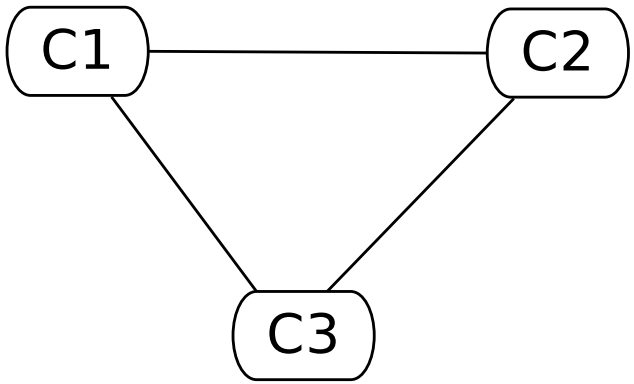
Definition allows

- loops: $\wedge_{j \in L} A_j{:}A_j$ (written $\wedge_{j \in L} A_j \circlearrowleft$)
- n-ary synchronizations
- autonomous ($NOSYNC$) moves: $\wedge_{j \in \emptyset} A_j{:}B_j$



Sync-programs: An example

Consider a system made of three components $C1, C2$ and $C3$ whose interactions are described by the following interaction graph:



Assume the three components to be associated with the following sets of atomic propositions:

$$AP_{C1} = \{A, B, C\} \qquad AP_{C2} = \{a, b\} \qquad AP_{C3} = \{1, 2\}$$

and that the states of interest are the following:

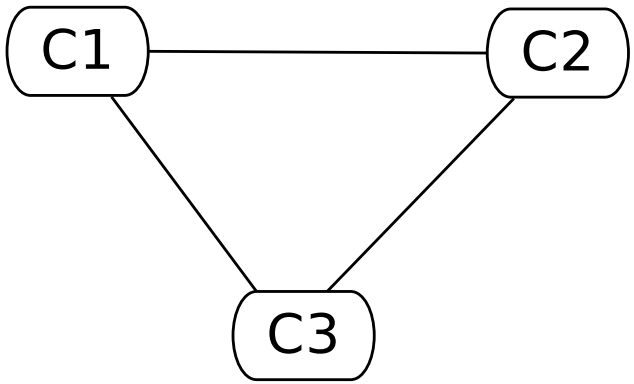
$$A \wedge \neg B \wedge \neg C \qquad \neg A \wedge B \wedge \neg C \qquad \neg A \wedge \neg B \wedge C$$

$$a \wedge \neg b \qquad \neg a \wedge b$$

$$1 \wedge \neg 2 \qquad \neg 1 \wedge 2$$

Sync-programs: An example

Consider a system made of three components $C1, C2$ and $C3$ whose interactions are described by the following interaction graph:



Assume the three components to be associated with the following sets of atomic propositions:

$$AP_{C1} = \{A, B, C\} \qquad AP_{C2} = \{a, b\} \qquad AP_{C3} = \{1, 2\}$$

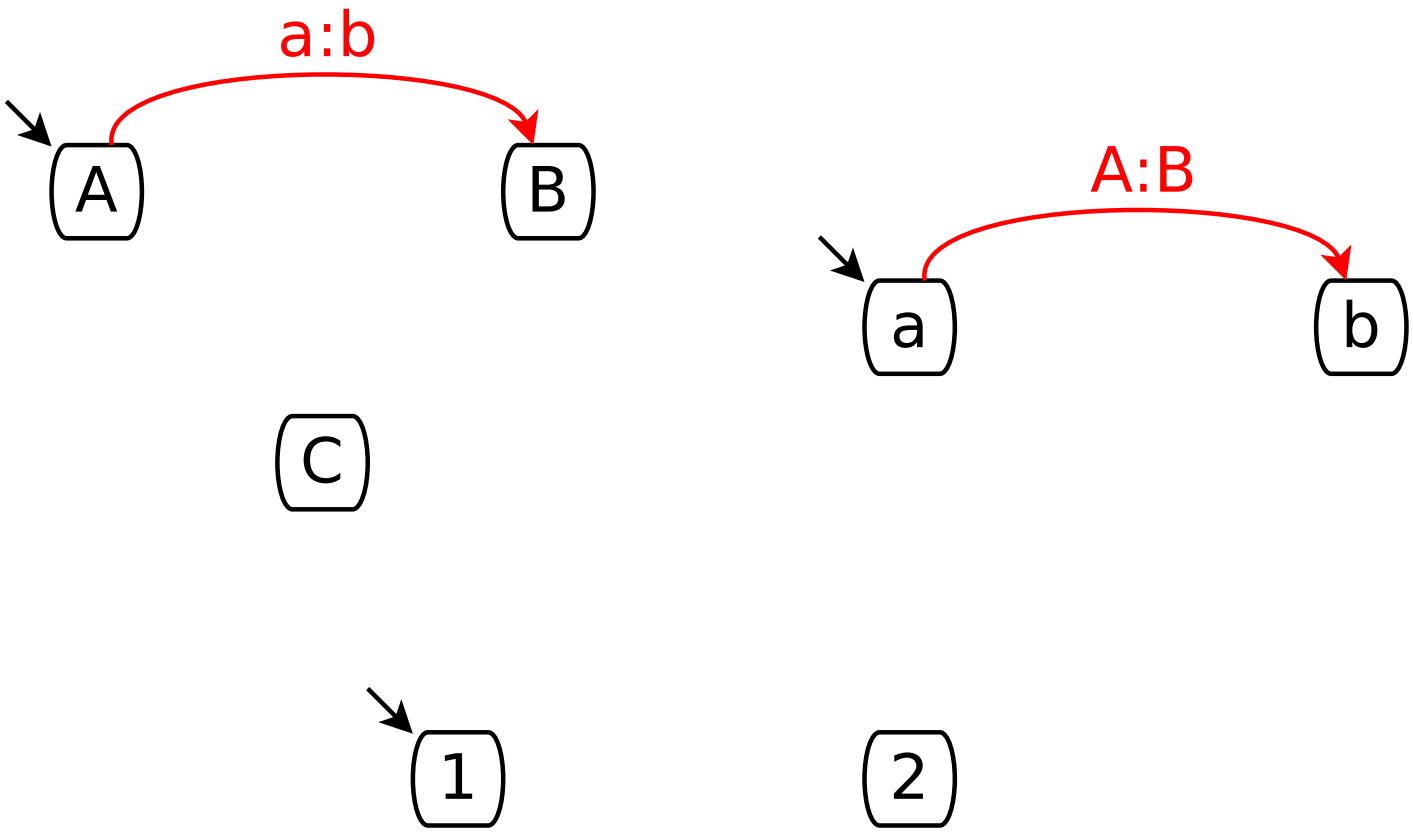
and that the states of interest are the following:

$$A \wedge \neg B \wedge \neg C \equiv A \qquad \neg A \wedge B \wedge \neg C \equiv B \qquad \neg A \wedge \neg B \wedge C \equiv C$$

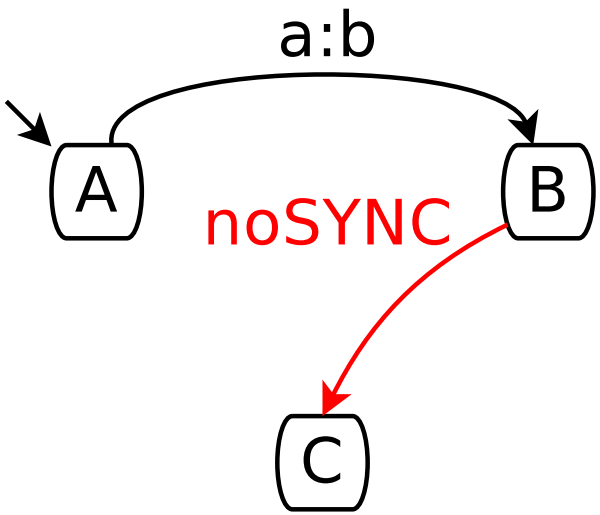
$$a \wedge \neg b \equiv a \qquad \neg a \wedge b \equiv b$$

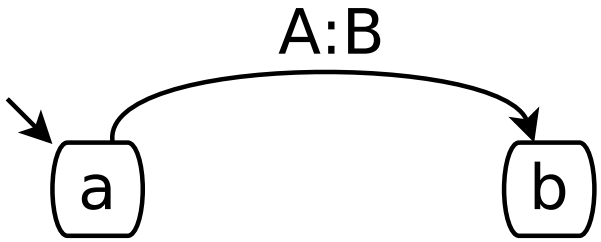
$$1 \wedge \neg 2 \equiv 1 \qquad \neg 1 \wedge 2 \equiv 2$$

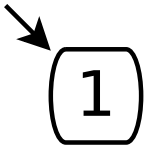
Sync-programs: An example



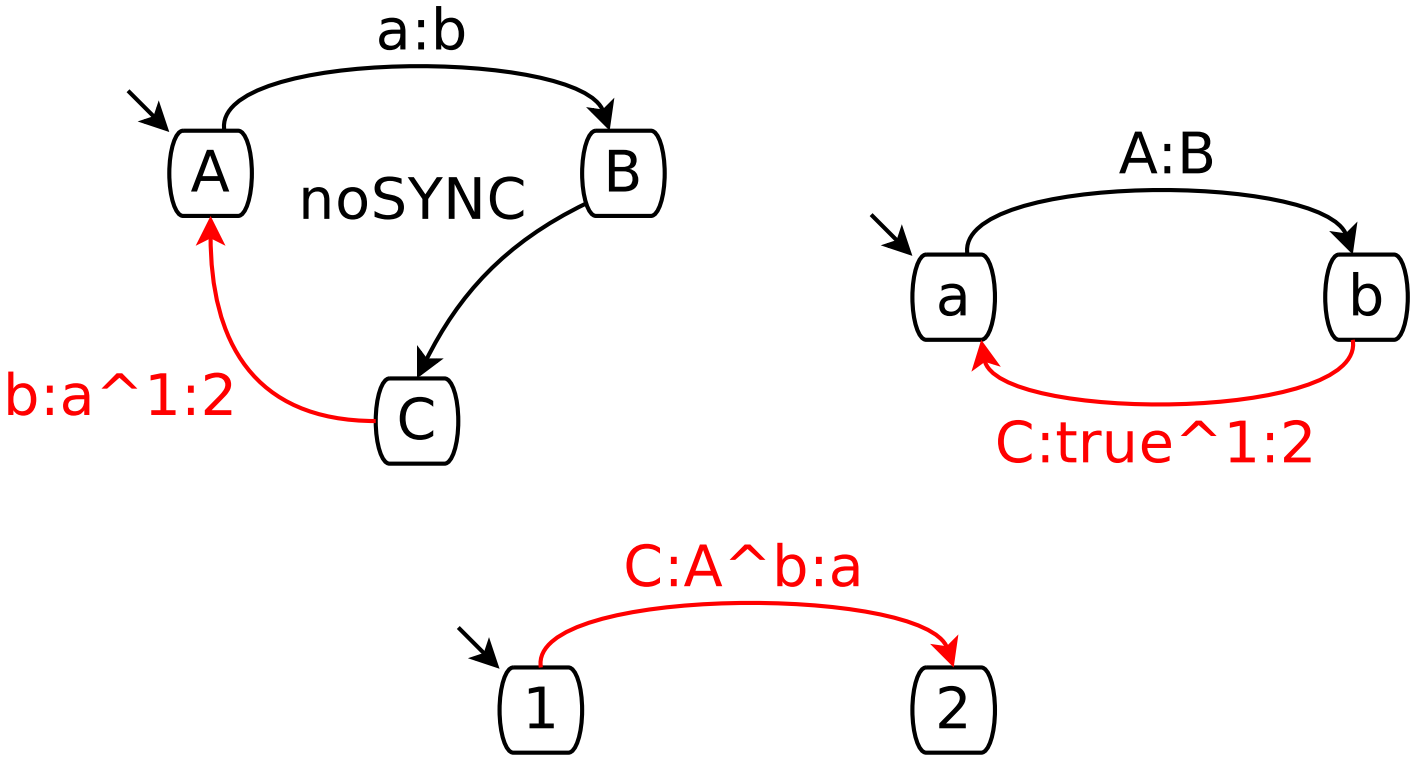
Sync-programs: An example



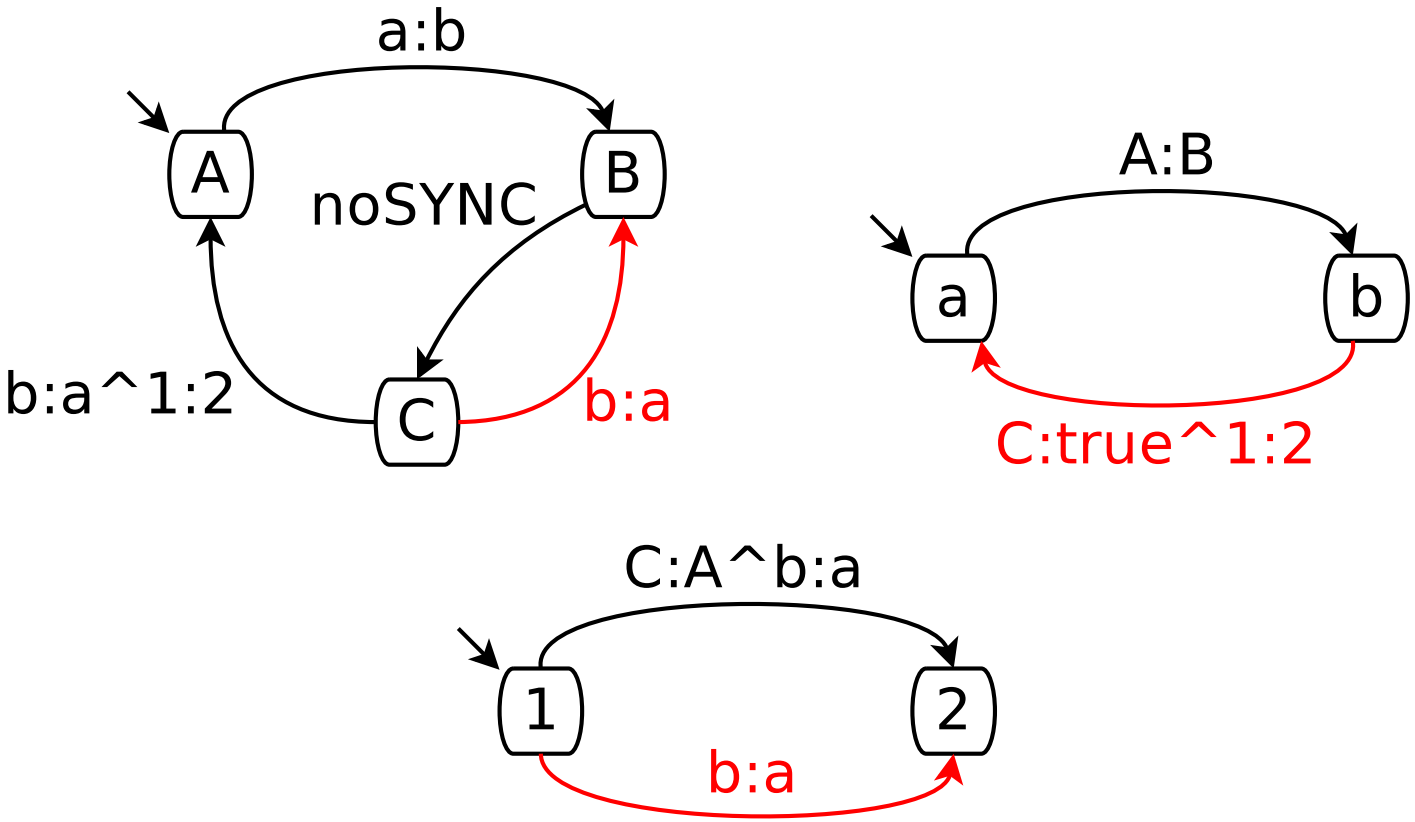




Sync-programs: An example



Sync-programs: An example



Syntax of Sync-programs

Let I be an interaction graph consisting of n nodes.

Parallel composition of sync-automata related by I.

Definition

A sync-program is a tuple

$$P^I = (S_0^I, P_1^I || \ldots || P_n^I),$$

where each P_i^I is a sync-automaton. Set $S_0^I = S_1^0 \times \ldots \times S_n^0$ is the set of initial states of the sync-program.

Semantics of Sync-programs

Definition

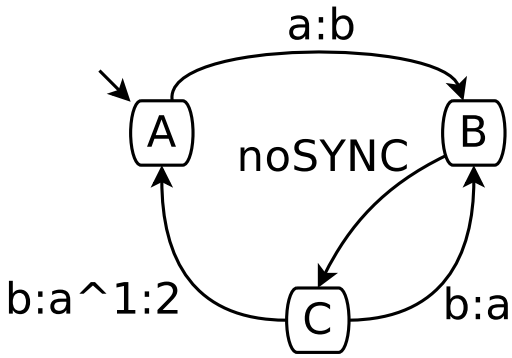
The semantics of $P^I = (S_I^0, P_1^I || \ldots || P_n^I)$ is a labelled transition system on I-states.

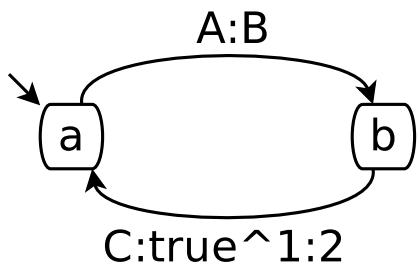
A I-state is a union of states of sync-automata $P_1^I, \ldots, P_n^I$.

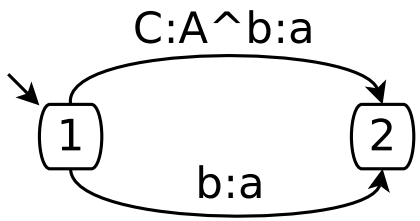
There is a transition (s, ℓ, t) iff

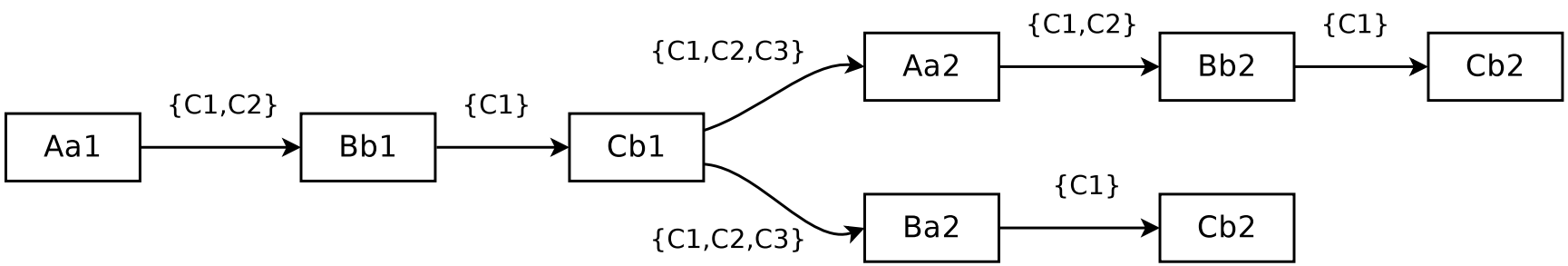
- label ℓ contains indices of all automata that perform a move, with mutually satisfied synchronisation conditions
- ℓ is minimal

Semantics – Example



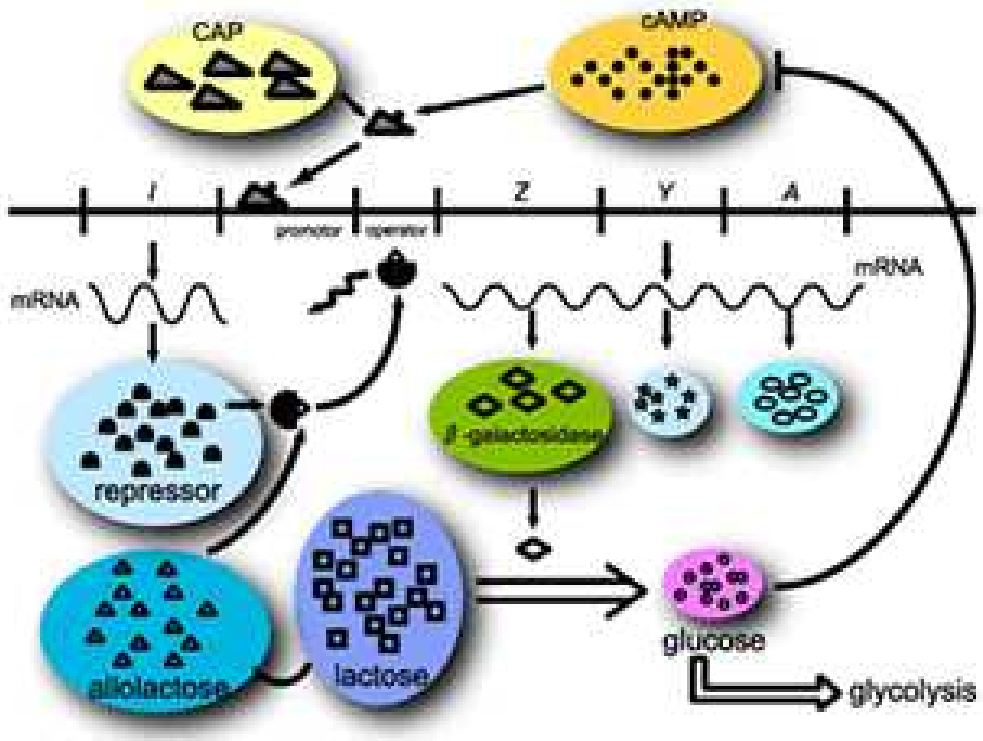




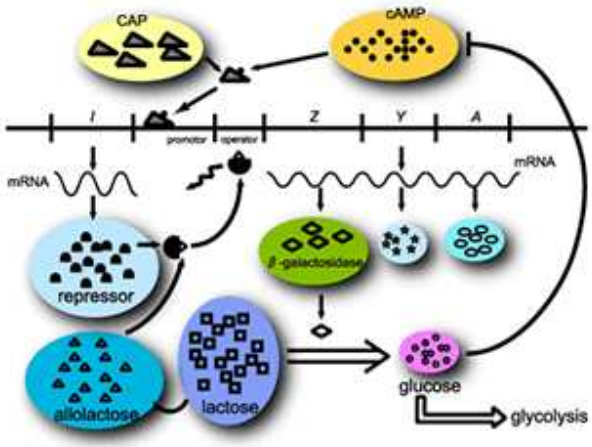


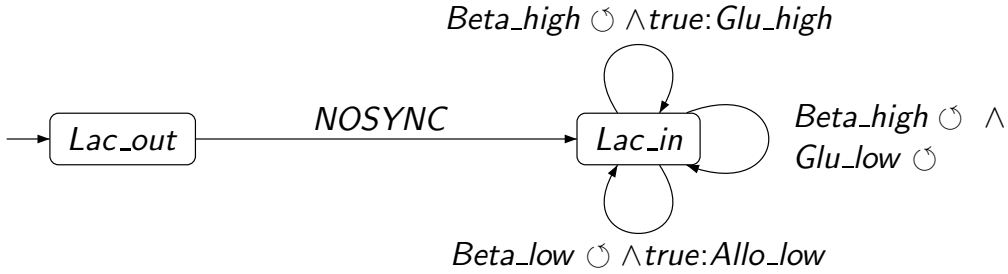
Outline of the talk

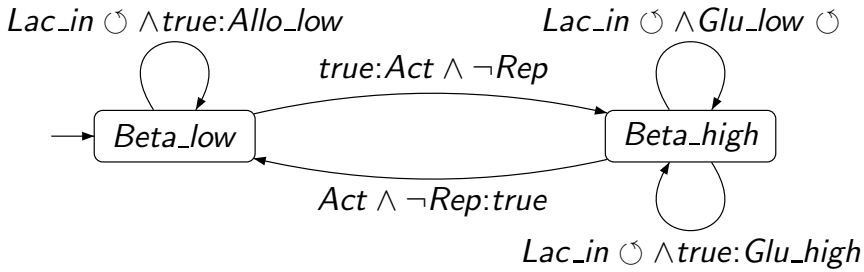
Lac operon regulation

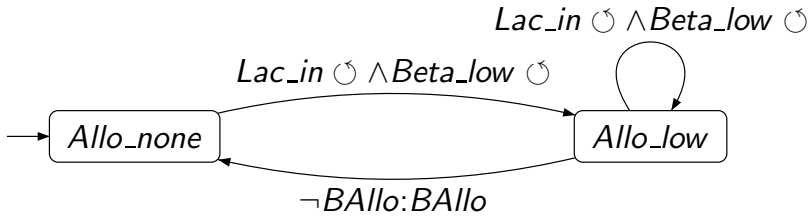


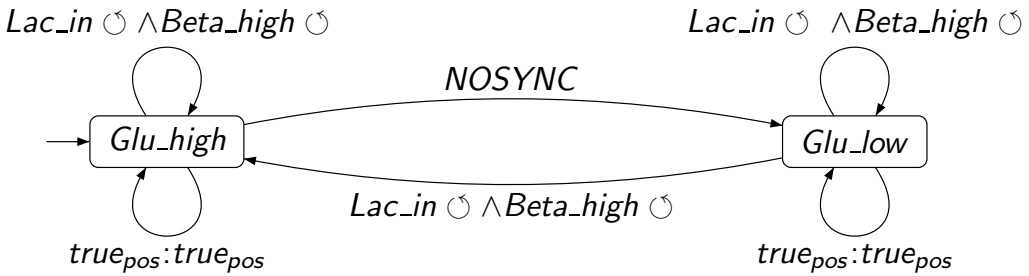
The model (1)


CAP
cAMP
I
promotor
operator
Z
Y
A
mRNA
mRNA
repressor
β-galactosidase
allolactose
lactose
glucose
glycolysis

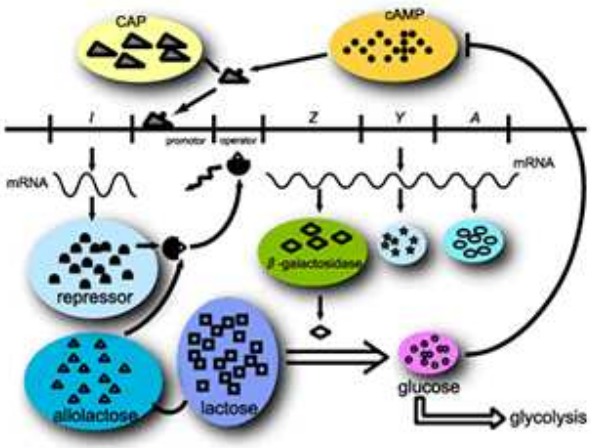

Beta_high ↺ ∧true:Glu_high
Lac_out
NOSYNC
Lac_in
Beta_high ↺ ∧
Glu_low ↺
Beta_low ↺ ∧true:Allo_low

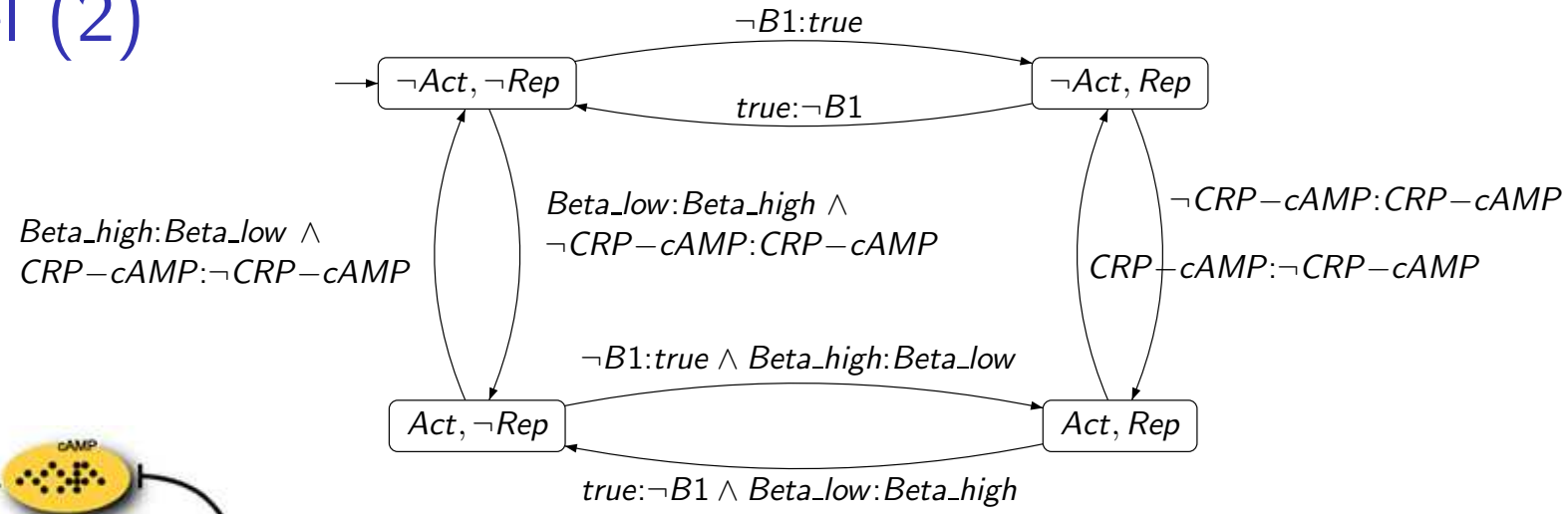

Lac_in ↺ ∧true:Allo_low
Lac_in ↺ ∧Glu_low ↺
true:Act ∧ ¬Rep
Beta_low
Beta_high
Act ∧ ¬Rep:true
Lac_in ↺ ∧true:Glu_high

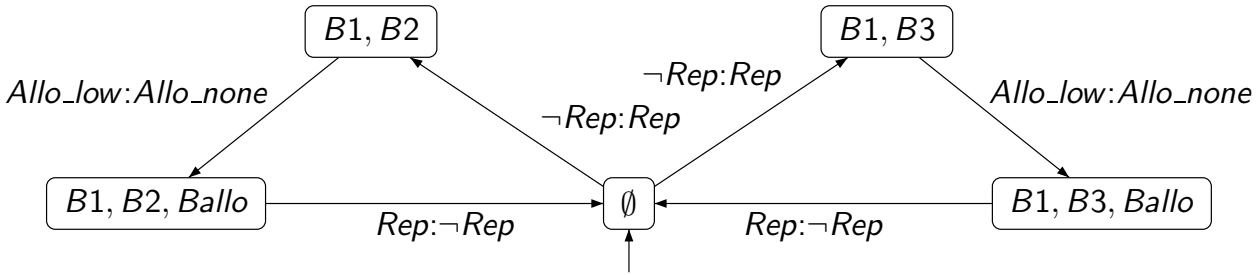

Lac_in ↺ ∧Beta_low ↺
Lac_in ↺ ∧Beta_low ↺
Allo_none
Allo_low
¬BAllo:BAllo

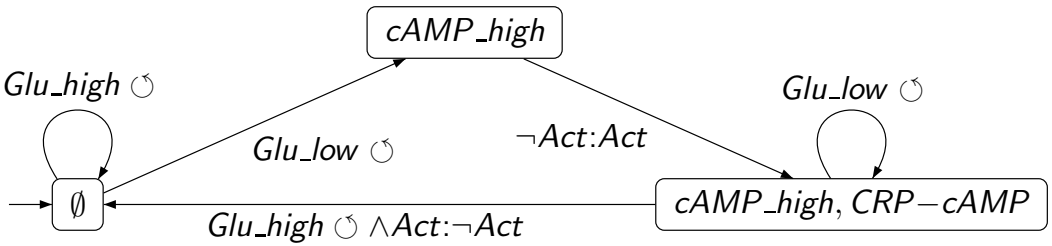

Lac_in ↺ ∧Beta_high ↺
Lac_in ↺ ∧Beta_high ↺
NOSYNC
Glu_high
Glu_low
Lac_in ↺ ∧Beta_high ↺
true_pos:true_pos
true_pos:true_pos

The model (2)

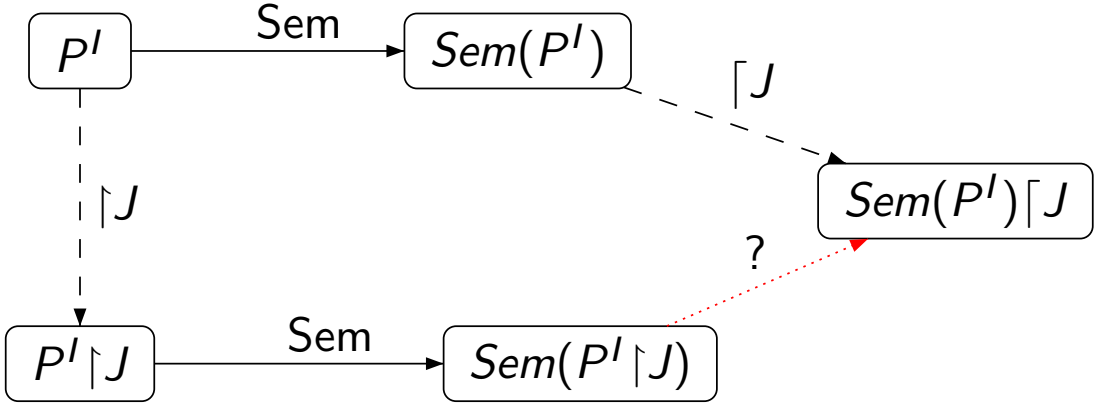

CAP
cAMP
I
promotor
operator
Z
Y
A
mRNA
mRNA
repressor
β-galactosidase
allolactose
lactose
glucose
glycolysis


¬B1:true
¬Act, ¬Rep
¬Act, Rep
true:¬B1
Beta_high:Beta_low ∧
CRP−cAMP:¬CRP−cAMP
Beta_low:Beta_high ∧
¬CRP−cAMP:CRP−cAMP
¬CRP−cAMP:CRP−cAMP
CRP−cAMP:¬CRP−cAMP
¬B1:true ∧ Beta_high:Beta_low
Act, ¬Rep
Act, Rep
true:¬B1 ∧ Beta_low:Beta_high


B1, B2
B1, B3
Allo_low:Allo_none
¬Rep:Rep
¬Rep:Rep
Allo_low:Allo_none
B1, B2, Ballo
∅
B1, B3, Ballo
Rep:¬Rep
Rep:¬Rep


cAMP_high
Glu_high ↻
Glu_low ↻
Glu_low ↻
¬Act:Act
∅
cAMP_high, CRP−cAMP
Glu_high ↻ ∧Act:¬Act

Modular verification – principle

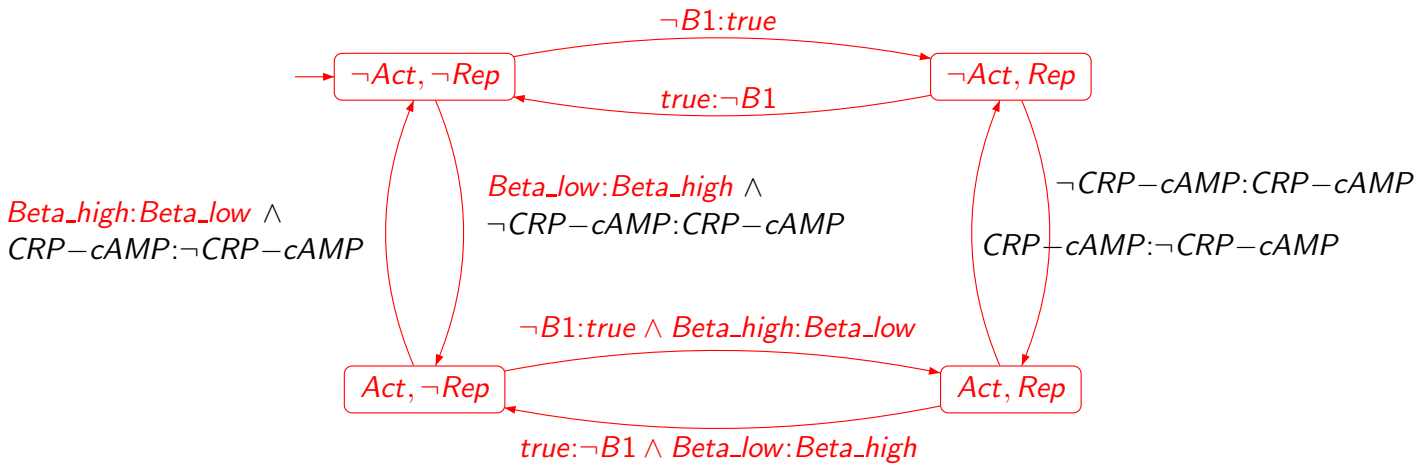


- Define: prog. language, semantics, projections
- Show: $Sem(P^I \restriction J) \sqsupseteq Sem(P^I)\lceil J$

Syntactical projection

Syntactical projection – subprogram $P^I \restriction J$

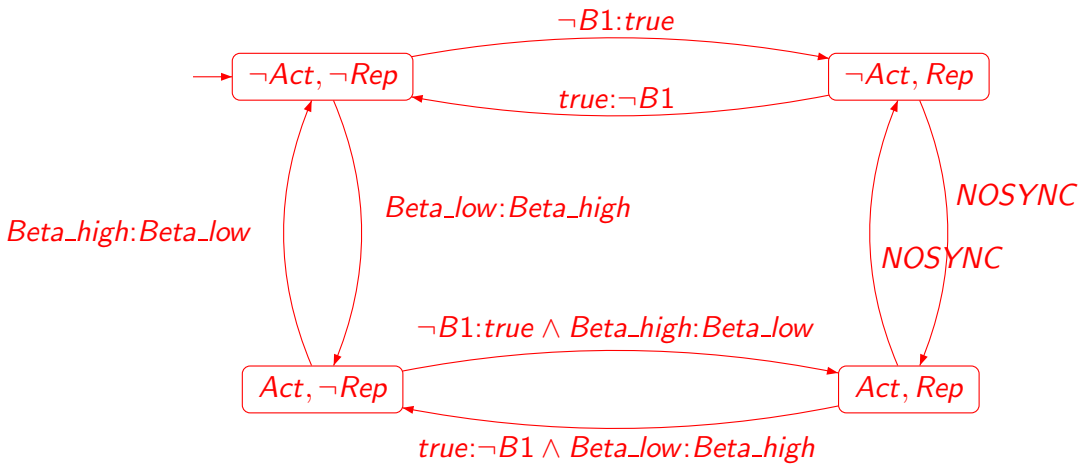
- only sync-automata from J
- sync-automata from J remain, synchronisation conditions change



Syntactical projection

Syntactical projection – subprogram $P^I \restriction J$

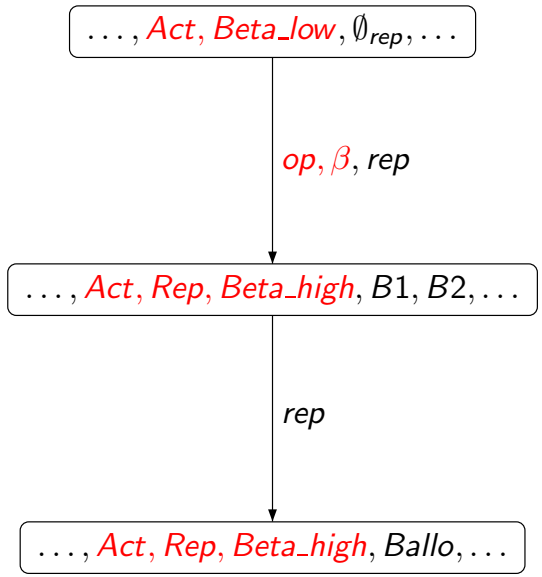
- only sync-automata from J
- sync-automata from J remain, synchronisation conditions change



Semantical projection

Semantical projection – $\mathcal{M}^I \lceil J$

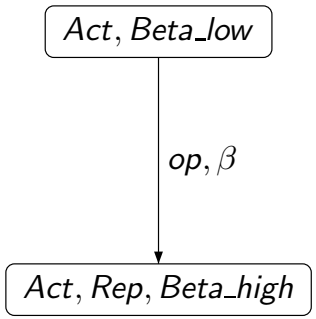
- I-states projected
- transitions projected



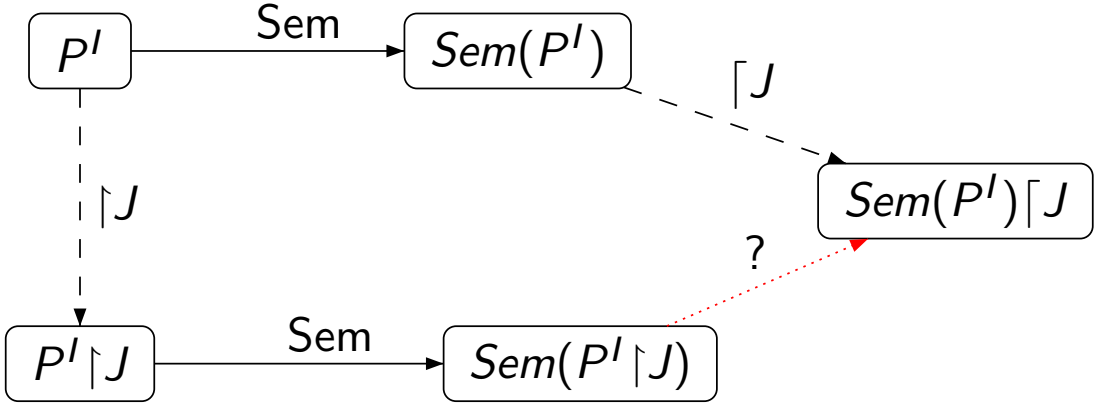
Semantical projection

Semantical projection – $\mathcal{M}^I \lceil J$

- I-states projected
- transitions projected



Modular verification – principle



- Define: prog. language, semantics, projections
- Show: $Sem(P^I \restriction J) \sqsupseteq Sem(P^I)\lceil J$

Principle (1)

- Verification of the properties of the computation
- Computation = maximal path (fullpath)

Lemma (Path projection)

Let $\mathcal{M}_I$ be semantics of sync-program P^I. For every $J \subseteq I$
if π is a path in $\mathcal{M}_I$ then $\pi\lceil J$ is a path in $\mathcal{M}_J$,
where $\mathcal{M}_J$ is the semantics of sync-program $P^J = P^I \restriction J$.

Principle (2)

Possible problem – by projecting we may loose path maximality

Definition

A path $\pi = (s^1, I^1, s^2, I^2, \ldots)$ in $\mathcal{M}_I$ is fair iff for all $i \in |I|$ we have that $\{m \mid i \in I^m\}$ is infinite.

Lemma (Fullpath projection)

Let $J \subseteq I$ be an interaction graph. If π is a fair fullpath in $\mathcal{M}_I$, then $\pi \lceil J$ is a fair fullpath in $\mathcal{M}_J$.

ACTL logic

Definition (ACTL logic)

- $true, false$
- $p, \neg p$ for $p \in AP$
- $f \wedge g$, $f \vee g$
- AXf and $A[fUg]$

Features

- Includes: AFf, AGf and $AG[p \rightarrow AFq]$.
- ACTL_J – atomic propositions are from $\{AP_i \mid i \in J\}$
- Can express: exclusion, necessary consequence, necessary persistence, oscillatory behaviour
- Semantics – on LTSs, needs fullpaths

Property preservation theorem

Theorem (Property preservation)

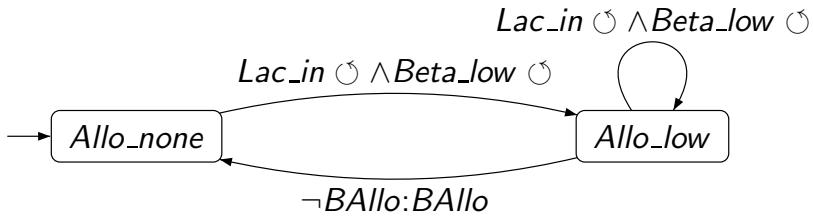
Let $J \subseteq I$ be an interaction graph, s an I-state and f an $ACTL_J$ property. If $\mathcal{M}_J, s\lceil J \models_\Phi f$ then $\mathcal{M}_I, s \models_\Phi f$.

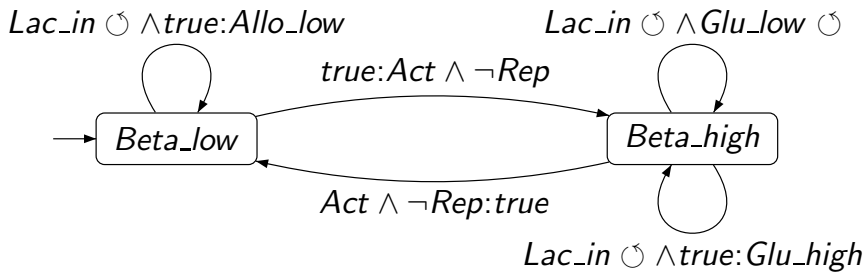
- proved by induction on the formula

Experiments (1)

"The increase of allolactose concentration can only be mediated by β-galactosidase in low concentration".

- formula $AG(Allo\_none \wedge Beta\_high \rightarrow A(\neg Allo\_low U Beta\_low))$
- true in the semantics of $P^{allo,\beta}$.

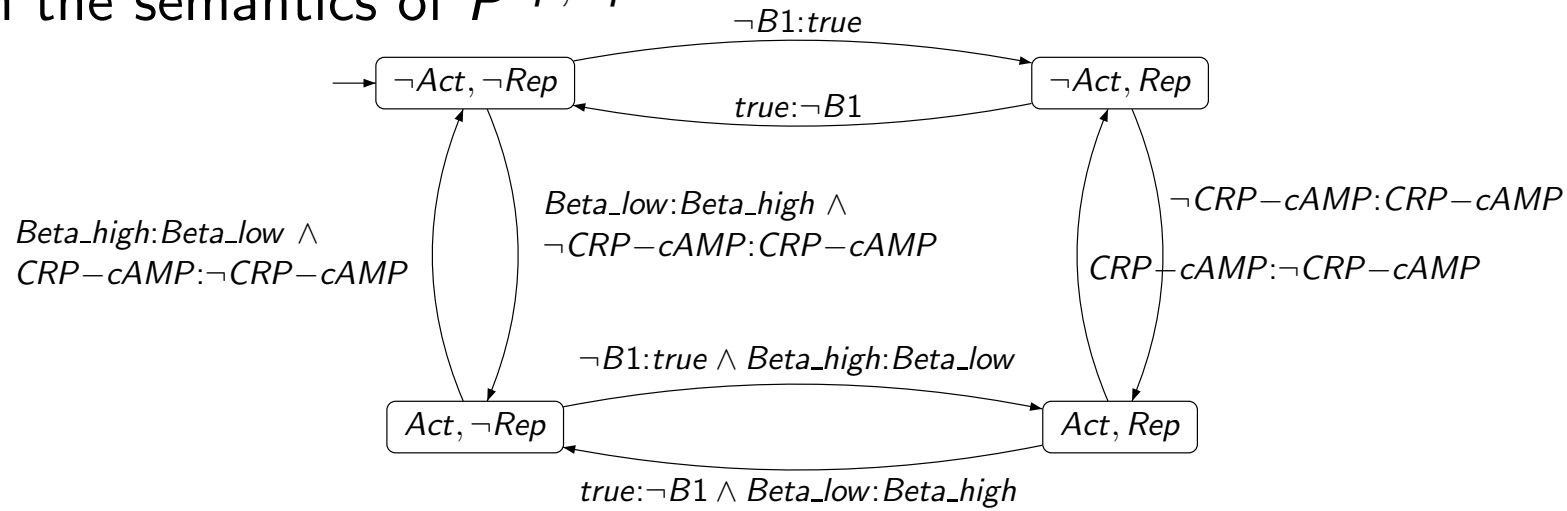


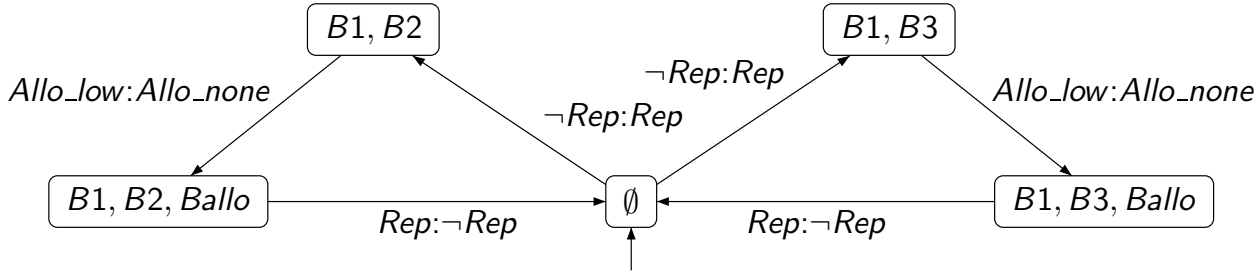


Experiments (2)

"The operon will oscillate between repressed and unrepressed state".

- formula $AG((rep \rightarrow AF\neg rep) \wedge (\neg rep \rightarrow AFrep))$
- true in $\mathcal{M}_I$, but the verification in semantics of P^{op} fails
- by inspecting the model, we enlarge the fragment needed
- true in the semantics of $P^{op,rep}$





Further developments

Further work

- Dynamic systems
- Abstract interpretation – towards full logic preservation
- Relations with other formalisms (e.g. process calculi)
- Schocastic extension

Outline of the talk

Uncertain kinetic rates

Kinetic parameters of (bio)chemical reactions are often very difficult to estimate precisely

- the rate of a reaction depends many physical parameters: temperature, pH, volumes, etc. . .

Moreover, some parameters cannot be measured at all in laboratory

- inferred (with rough approximations) from similar reactions

The approach we propose consists in:

- replacing kinetic constants with intervals of possible values
- applying probabilistic model checking to obtain conservative upper and lower bounds for probabilistic reachability properties

We expolit abstract interpretation techniques to prove the correctness of our approach

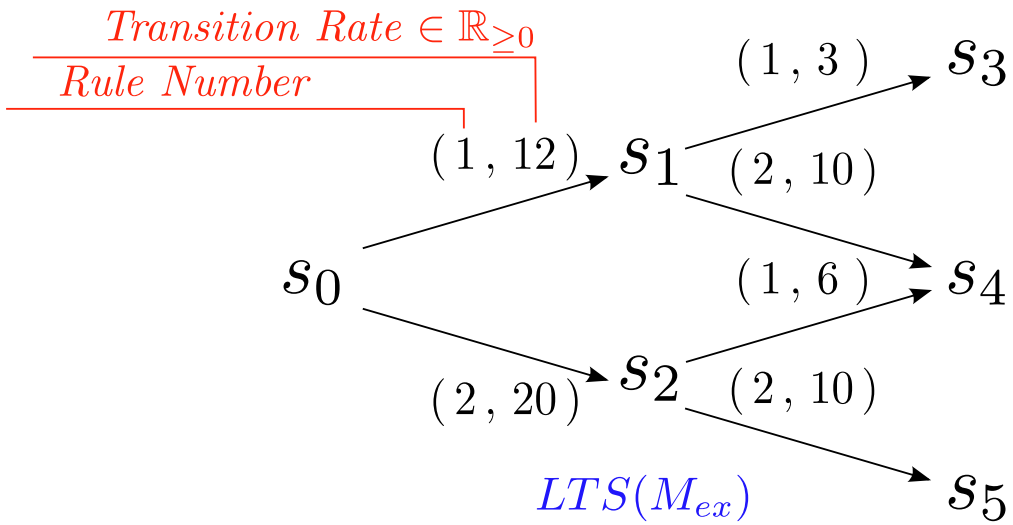
Probabilistic Reachability without Uncertainty

Let us consider the following simple example:

$$M_{ex} = \{ \quad R_1 : X\ Y \xrightarrow{3} Z \qquad R_2 : X\ W \xrightarrow{1} W \quad \}$$

with initial state $s_0 = 2X\,2Y\,10W$.

We can easily construct the following Labelled Transition System (LTS):



where the transition rate is computed as in Gillespie's algorithm.

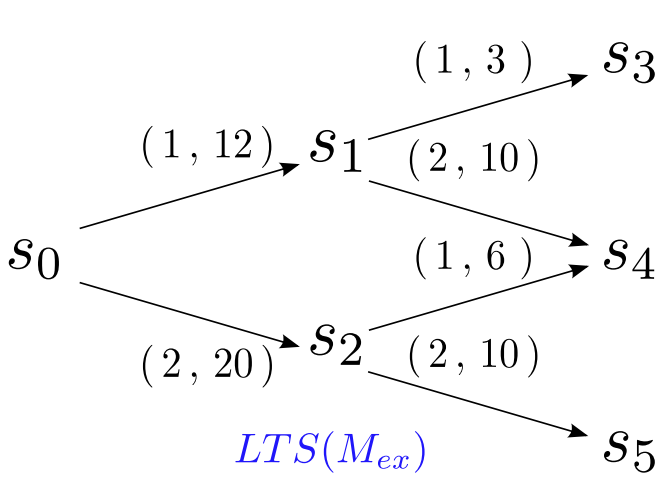
Probabilistic Reachability without Uncertainty

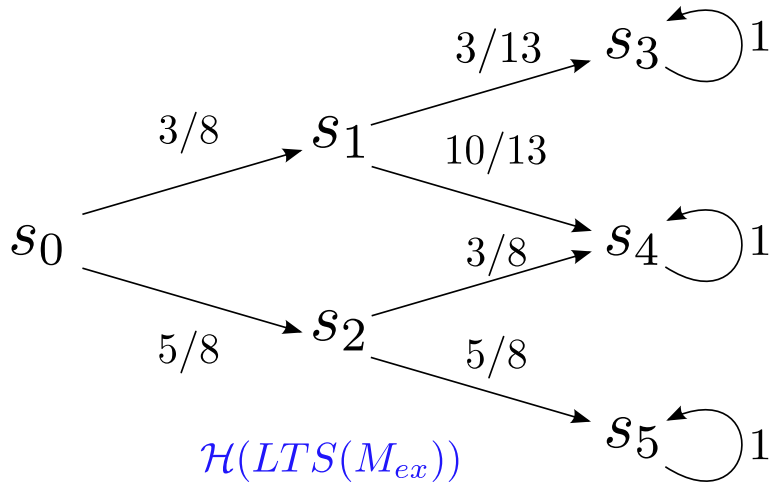
Let us consider the following simple example:

$$M_{ex} = \{ \quad R_1 : X\ Y \xrightarrow{3} Z \qquad R_2 : X\ W \xrightarrow{1} W \quad \}$$

with initial state $s_0 = 2X\,2Y\,10W$.

We can translate the LTS into a Discrete Time Markov Chain (DTMC):





We consider only sequentiality of events and we loose information on the elapsing of time.

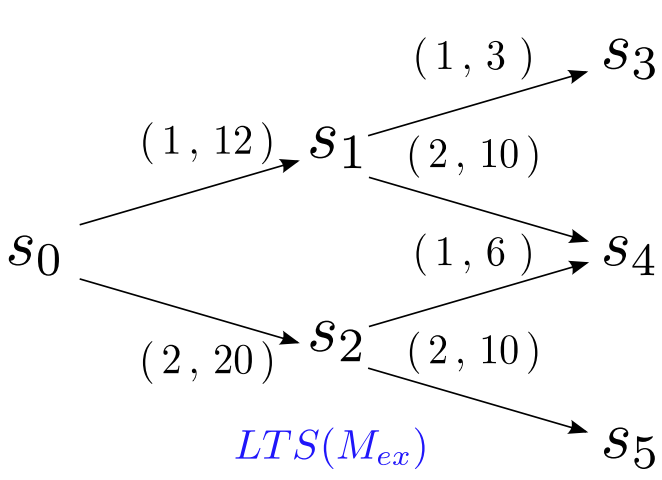
Probabilistic Reachability without Uncertainty

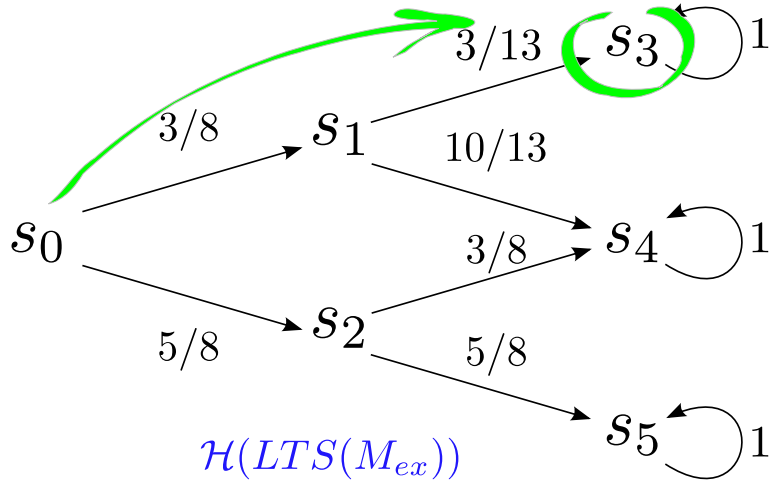
Let us consider the following simple example:

$$M_{ex} = \{ \quad R_1 : X\ Y \xrightarrow{3} Z \qquad R_2 : X\ W \xrightarrow{1} W \quad \}$$

with initial state $s_0 = 2X\,2Y\,10W$.

The DTMC can be used for probabilistic reachability analysis:





Example: $P(\text{obtaining two } Z) = Reach(s_3) = 3/8 \times 3/13 = 9/104$

Probabilistic Reachability with Uncertainty

Our approach:

- we allow intervals of possible values to be used in place of kinetic constants
- a model of chemical reactions with intervals (abstract model) represents an infinite set of models of reactions with kinetic constants (concrete models)

For example, the following abstract model

$$M^{\circ}_{ex} = \{ \quad R^{\circ}_1 : \; X \; Y \xrightarrow{[1,5]} Z \qquad R^{\circ}_2 : \; X \; W \xrightarrow{[1,5]} W \quad \}$$

includes the previously considered concrete model

$$M_{ex} = \{ \quad R_1 : \; X \; Y \xrightarrow{3} Z \qquad R_2 : \; X \; W \xrightarrow{1} W \quad \}$$

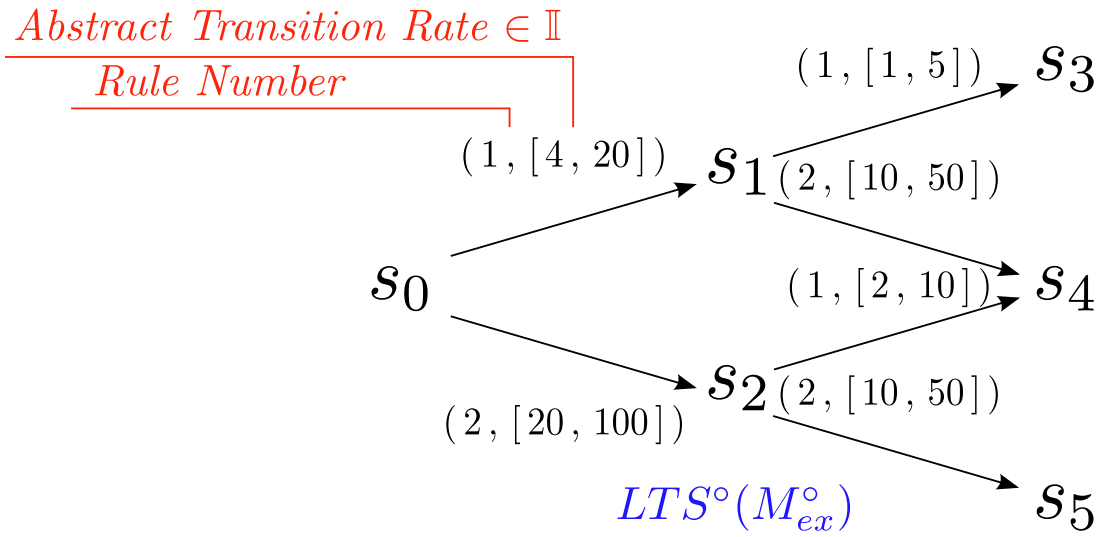
Probabilistic Reachability with Uncertainty

Let us consider the following simple example:

$$M^{\circ}_{ex} = \{ \quad R^{\circ}_1 : \; X \; Y \xrightarrow{[1,5]} Z \qquad R^{\circ}_2 : \; X \; W \xrightarrow{[1,5]} W \quad \}$$

with initial state $s_0 = 2X\,2Y\,10W$.

We can easily construct the following Labelled Transition System (LTS):



where the abstract transition rate is computed as in Gillespie's algorithm on the interval endpoints.

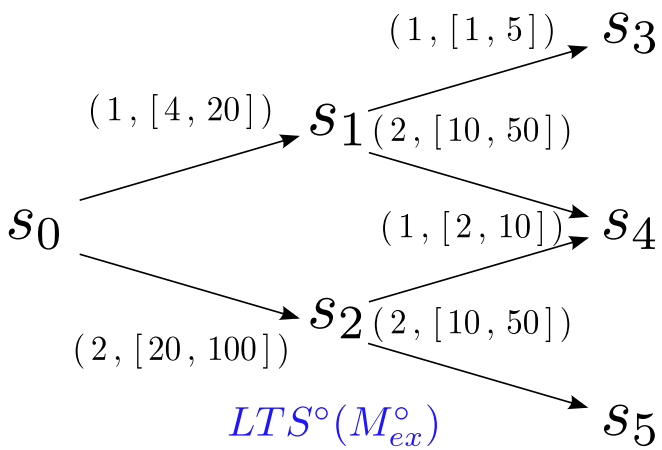
Probabilistic Reachability with Uncertainty

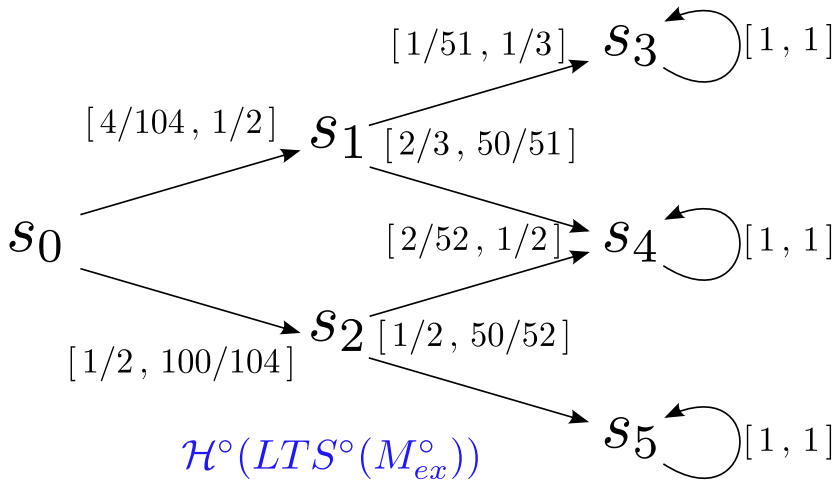
Let us consider the following simple example:

$$M^{\circ}_{ex} = \{ \quad R^{\circ}_1 : \; X \; Y \xrightarrow{[1,5]} Z \qquad R^{\circ}_2 : \; X \; W \xrightarrow{[1,5]} W \quad \}$$

with initial state $s_0 = 2X\,2Y\,10W$.

We can translate the LTS into a Interval Markov Chain (IMC):





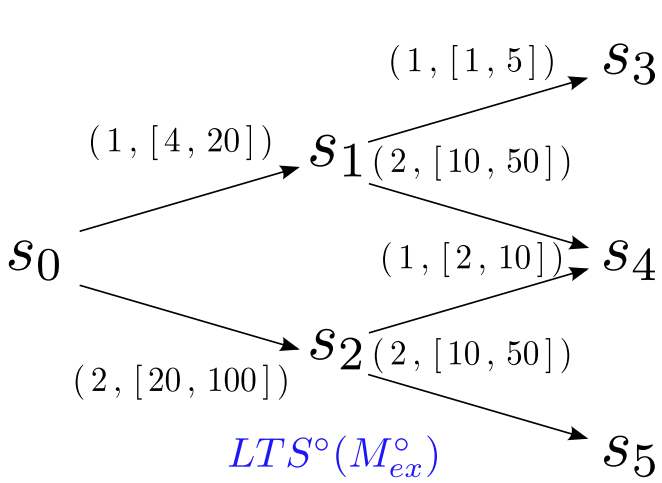
Probabilistic Reachability with Uncertainty

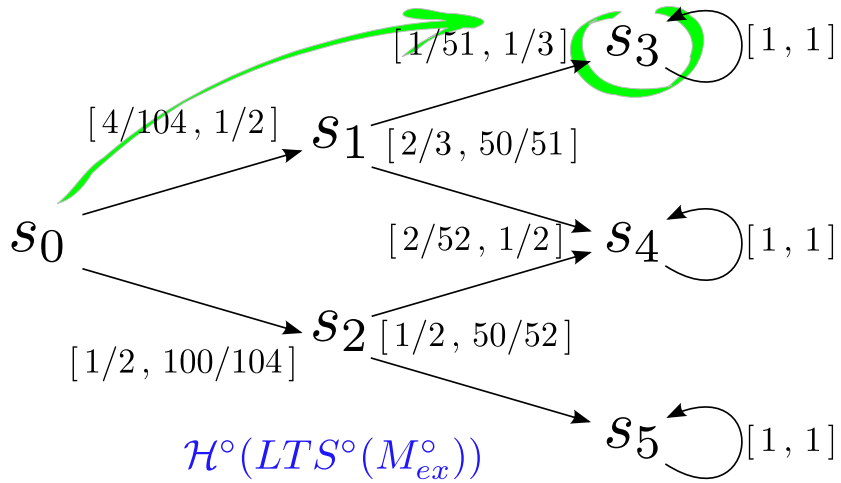
Let us consider the following simple example:

$$M_{ex}^{\circ} = \{ \quad R_1^{\circ} : X\ Y \xrightarrow{[1,5]} Z \qquad R_2^{\circ} : X\ W \xrightarrow{[1,5]} W \quad \}$$

with initial state $s_0 = 2X\,2Y\,10W$.

The IMC can be used for probabilistic reachability analysis:



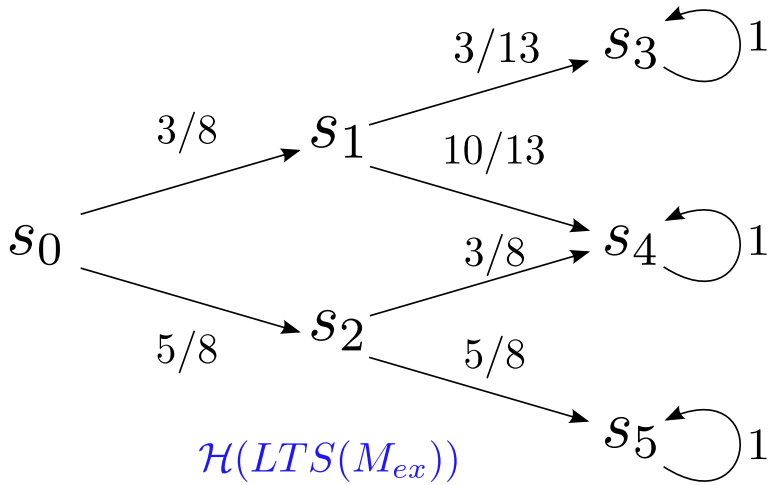


Example: $P(\text{obtaining two } Z) = Reach(s_3) =$

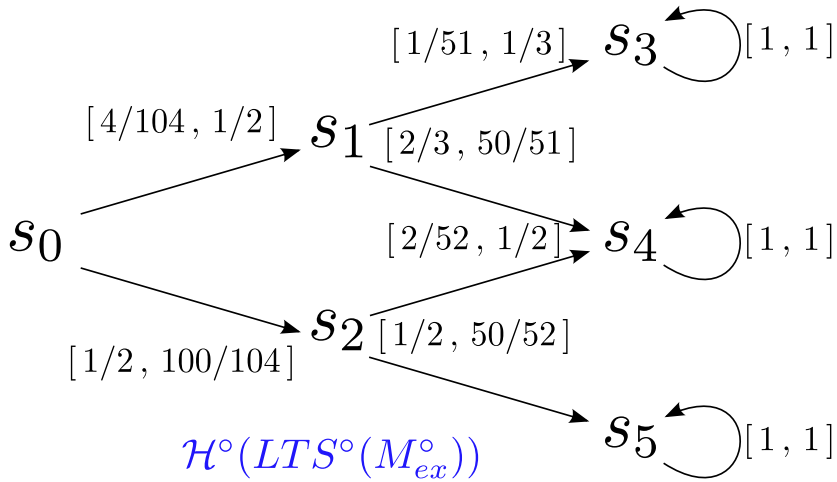
$$= [4/104, 1/2] \times^{Int} [1/51, 1/3] = [1/1326, 1/6]$$

Probabilistic Reachability with Uncertainty

In a DTMC the outgoing transitions of each state are associated with a probability distribution



In a IMC the outgoing transitions of each state may be associated with a infinite number of probability distributions



Probabilistic Reachability with Uncertainty

We have proved that the probability distributions of states of a concrete model M are included in those of the corresponding abstract model M°

- abstract probabilistic reachability gives correct upper- and lower-bounds

We have applied standard abstract interpretation techniques:

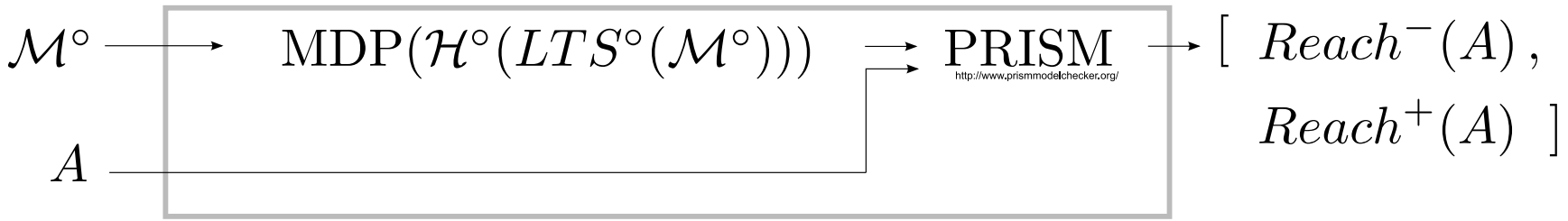
$$\begin{array}{ccccc} \mathcal{M}^\circ & \xrightarrow{LTS^\circ} & \mathcal{LTS}^\circ & \xrightarrow{\mathcal{H}^\circ} & \mathcal{IMC} \\ \uparrow \alpha & & \uparrow \alpha_{\mathcal{LTS}} & & \uparrow \alpha_{\mathcal{MC}} \\ \mathcal{M} & \xrightarrow{LTS} & \mathcal{LTS} & \xrightarrow{\mathcal{H}} & \mathcal{DTMC} \end{array}$$

Probabilistic Reachability with Uncertainty

Probabilistic reachability analysis becomes more complex when the model consists of more than two chemical reactions

- We have followed a standard extreme distributions approach (Fecher et Al.) that requires translation of the IMC into a Markov Decision Process (MDP)

We have developed a translator from chemical reactions with uncertain rates into PRISM input language

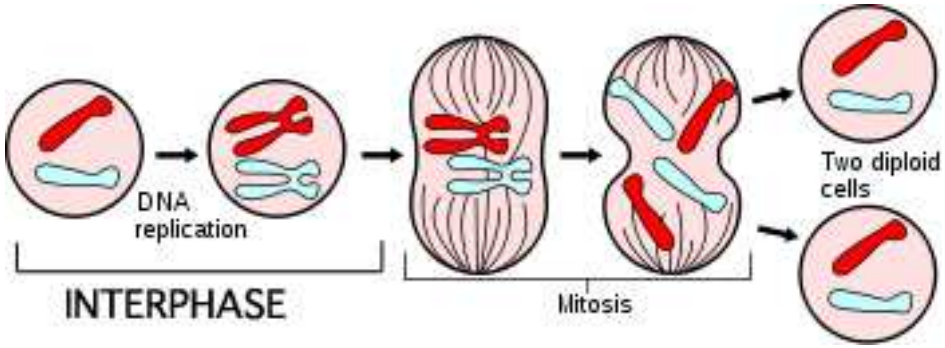


- AMSR2PRISM translator,
  `http://www.di.unipi.it/msvbio/`
- PRISM model checker,
  `http://www.prismmodelchecker.org`

An example: Tumor growth (cell cycle)

Tumor growth is based on cell divisions (or *mitosis*).

The cell cycle, the process between two mitosis, consists of 4 phases :



- I : interphase
  - G_1: pre-synthetic phase
  - S : replication of DNA
  - G_2: post-synthetic phase
- M : mitosis phase

An example: Tumor growth (cell cycle)

We consider a ODE model of tumor growth proposed by *Villasana and Radunskaya*.

Tumor cells are classified in two populations:

- T_I: cells in the interphase (phases G_1, S and G_2);
- T_M: cells in the mitotic phase (M).

The model includes the following events:

1. cell death in any phase (apotosis)
2. interphase $\longrightarrow$ mitosis (one cell in T_I moves to T_M)
3. mitosis $\longrightarrow$ interphase (one cell in T_M becomes two in T_M)

The passage from interphase to mitosis takes much more time than the other events.

An example: Tumor growth (cell cycle)

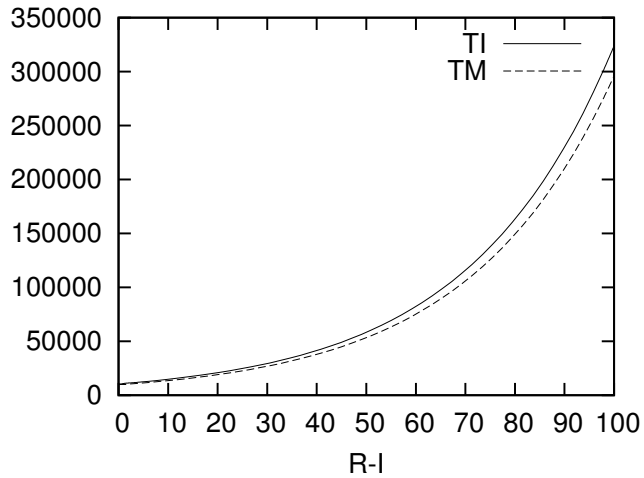
The ODE model by *Villasana and Radunskaya* is:

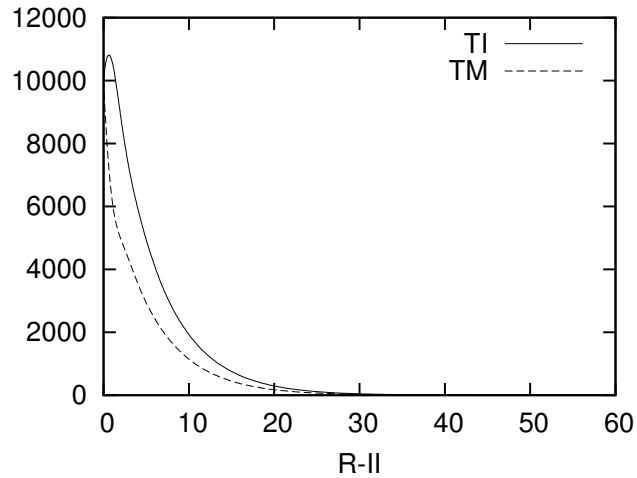
$$\frac{dT_I}{dt} = 2a_4 T_M - d_2 T_I - a_1 T_I$$
$$\frac{dT_M}{dt} = a_1 T_I - d_3 T_M - a_4 T_M$$

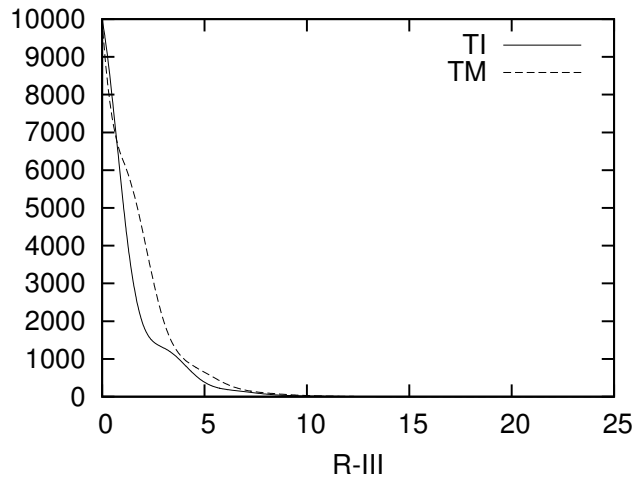
Let $d = d_3 + a_4$, namely d is the rate at which mitotic cells disappear.

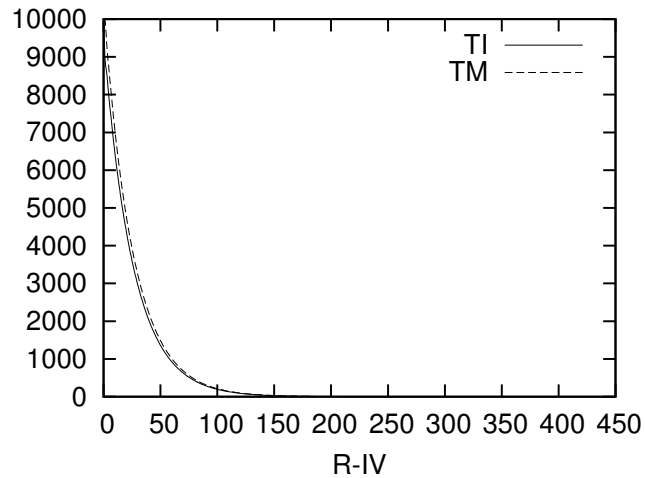
An example: Tumor growth (cell cycle)

These are some results of numerical simulation.









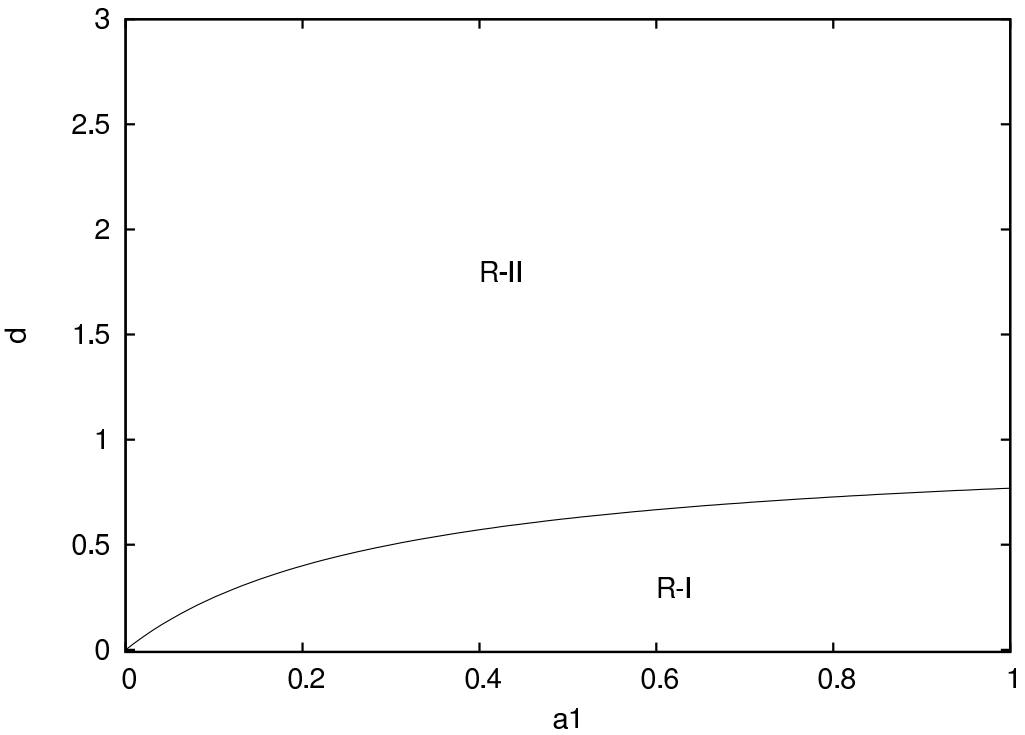
Probabilistic Reachability in the Tumor Growth Model

Let us reformulate the tumor growth example as a set of reactions.

Reactions:

$$T_M \xrightarrow{a_4} 2\,T_I \qquad T_I \xrightarrow{d_2}$$

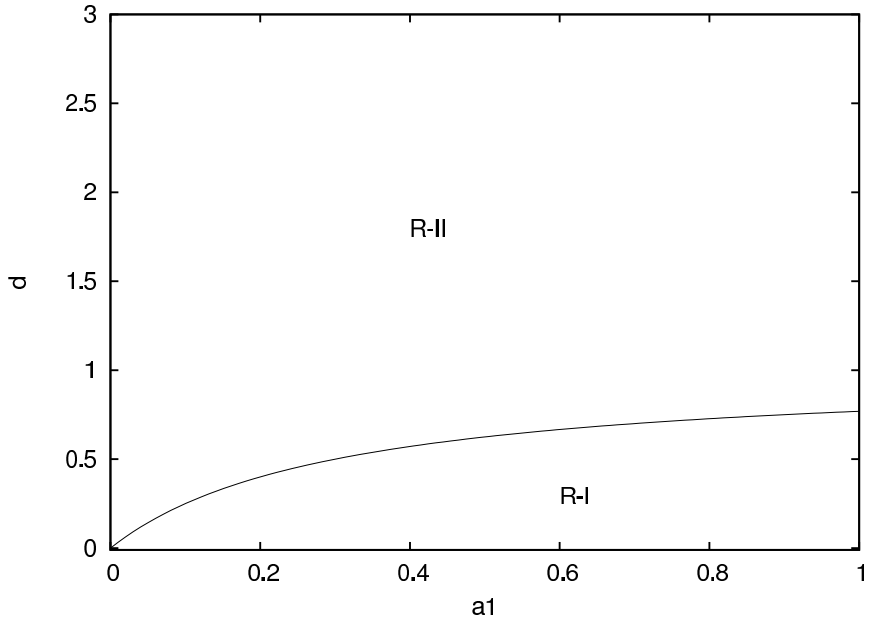
$$T_I \xrightarrow{a_1} T_M \qquad T_M \xrightarrow{d_3}$$



In this case we have only two parameter regions:

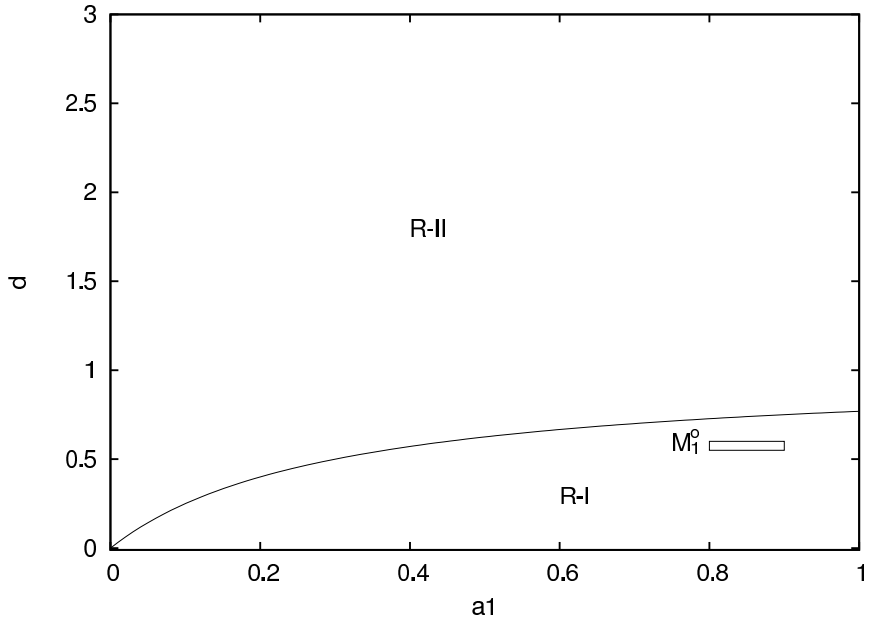
- In R-I the tumor grows
- In R-II the tumor decays

Probabilistic Reachability in the Tumor Growth Model



We consider three abstract models of tumor growth.

Probabilistic Reachability in the Tumor Growth Model



We consider three abstract models of tumor growth.

Abstract model M_1° :

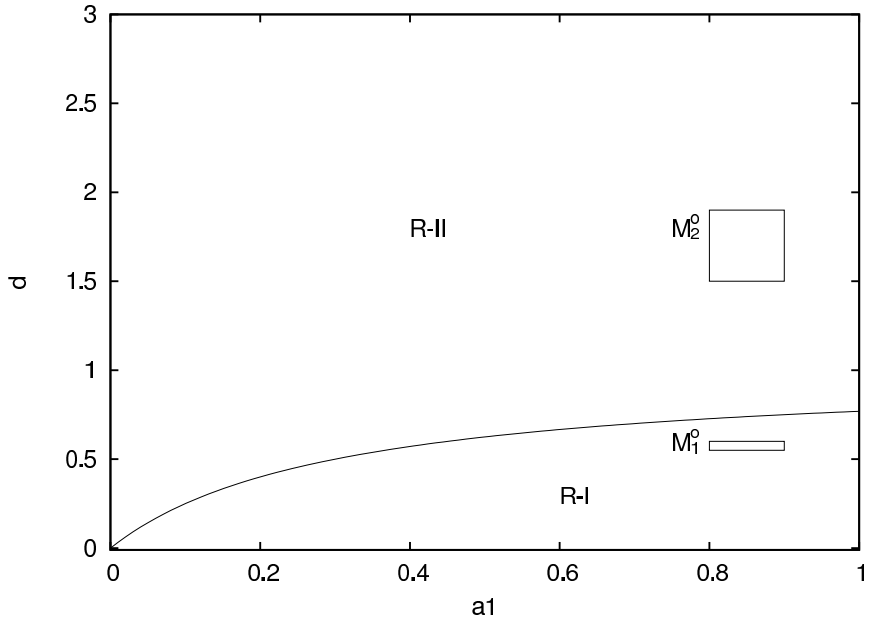
$T_M \xrightarrow{0.5} 2\,T_I$

$T_I \xrightarrow{[0.8,0.9]} T_M$

$T_I \xrightarrow{0.3}$

$T_M \xrightarrow{[0.05,0.1]}$

Probabilistic Reachability in the Tumor Growth Model



We consider three abstract models of tumor growth.

Abstract model M_1° :

$T_M \xrightarrow{0.5} 2T_I$

$T_I \xrightarrow{[0.8,0.9]} T_M$

$T_I \xrightarrow{0.3}$

$T_M \xrightarrow{[0.05,0.1]}$

Abstract model M_2° :

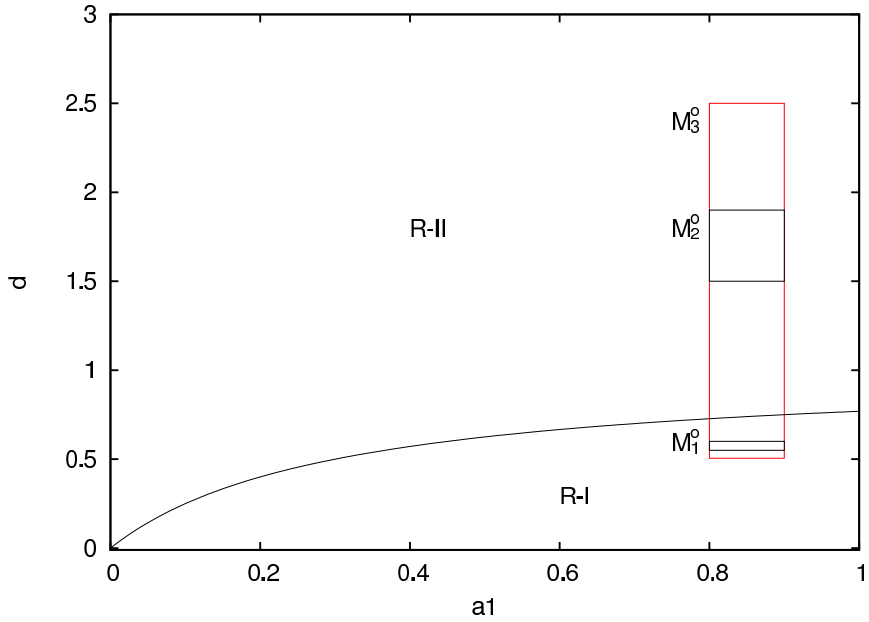
$T_M \xrightarrow{0.5} 2T_I$

$T_I \xrightarrow{[0.8,0.9]} T_M$

$T_I \xrightarrow{0.3}$

$T_M \xrightarrow{[1,1.4]}$

Probabilistic Reachability in the Tumor Growth Model



We consider three abstract models of tumor growth.

Abstract model M_1° :

$T_M \xrightarrow{0.5} 2T_I$

$T_I \xrightarrow{[0.8,0.9]} T_M$

$T_I \xrightarrow{0.3}$

$T_M \xrightarrow{[0.05,0.1]}$

Abstract model M_2° :

$T_M \xrightarrow{0.5} 2T_I$

$T_I \xrightarrow{[0.8,0.9]} T_M$

$T_I \xrightarrow{0.3}$

$T_M \xrightarrow{[1,1.4]}$

Abstract model M_3° :

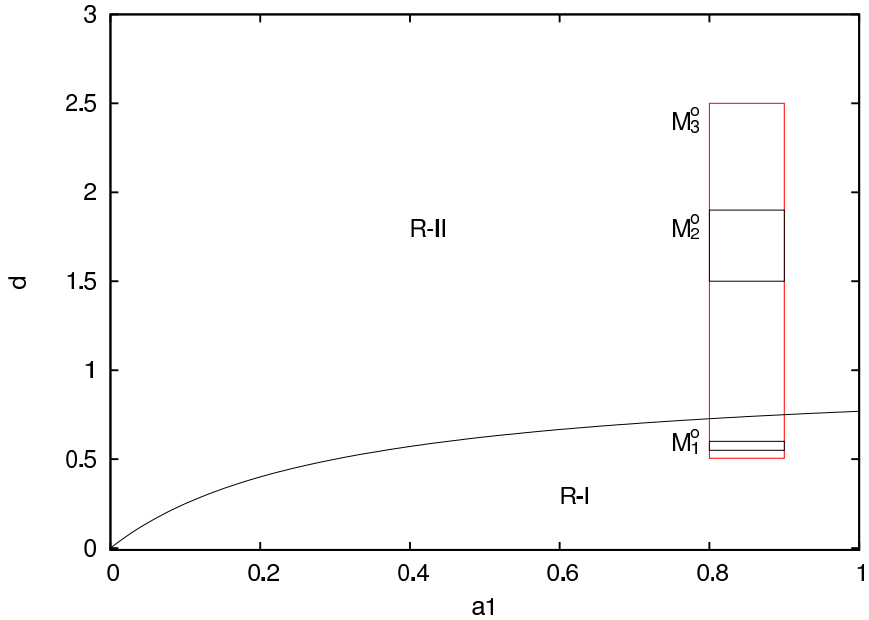
$T_M \xrightarrow{0.5} 2T_I$

$T_I \xrightarrow{[0.8,0.9]} T_M$

$T_I \xrightarrow{0.3}$

$T_M \xrightarrow{[0.005,2]}$

Probabilistic Reachability in the Tumor Growth Model



We consider three abstract models of tumor growth.

We consider an initial population consisting of $10\,T_M$ and $10\,T_I$.

Abstract model M_1° :

$T_M \xrightarrow{0.5} 2\,T_I$

$T_I \xrightarrow{[0.8,0.9]} T_M$

$T_I \xrightarrow{0.3}$

$T_M \xrightarrow{[0.05,0.1]}$

Abstract model M_2° :

$T_M \xrightarrow{0.5} 2\,T_I$

$T_I \xrightarrow{[0.8,0.9]} T_M$

$T_I \xrightarrow{0.3}$

$T_M \xrightarrow{[1,1.4]}$

Abstract model M_3° :

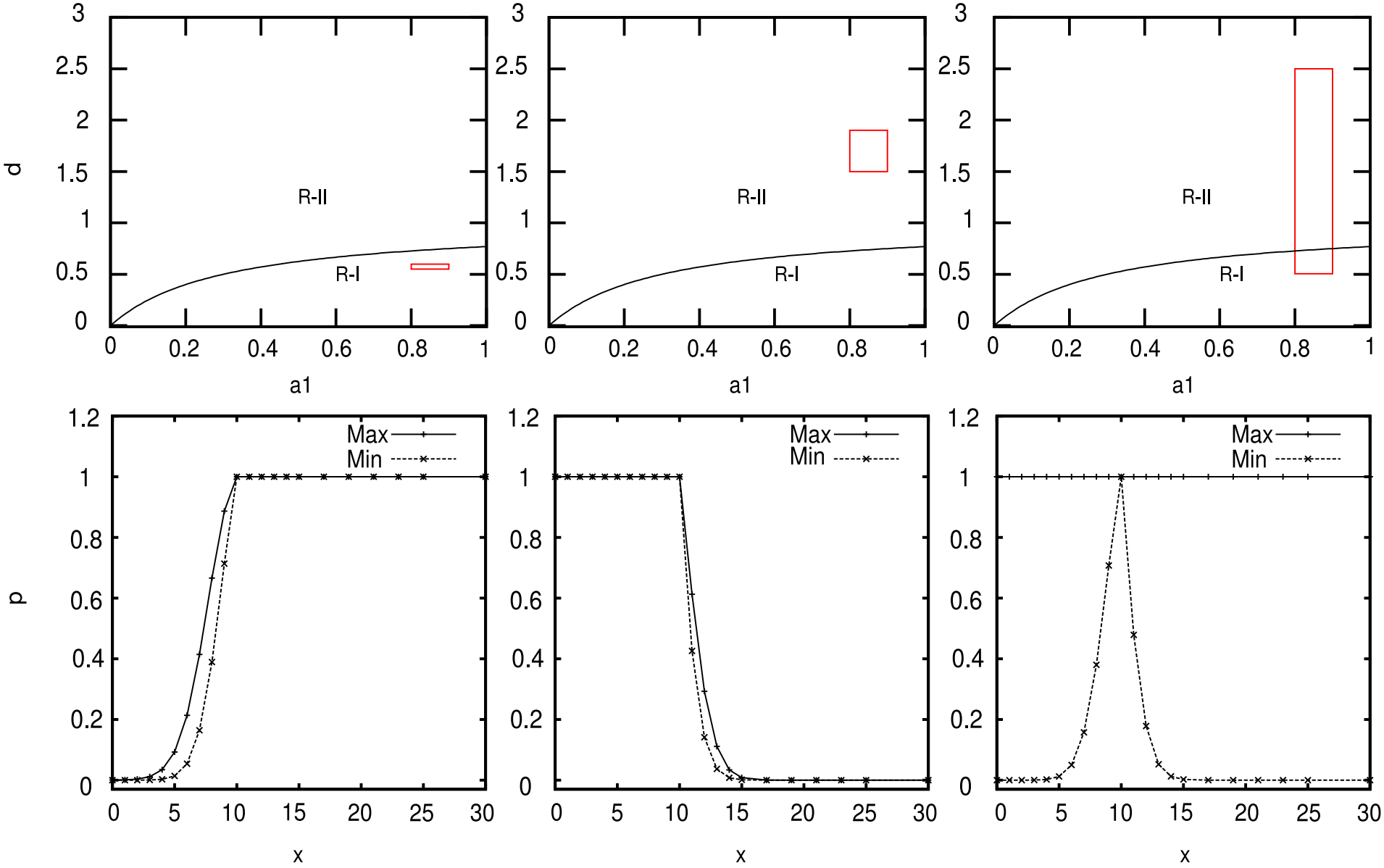
$T_M \xrightarrow{0.5} 2\,T_I$

$T_I \xrightarrow{[0.8,0.9]} T_M$

$T_I \xrightarrow{0.3}$

$T_M \xrightarrow{[0.005,2]}$

Probabilistic Reachability in the Tumor Growth Model



$Reach(T_M = x)$ on $M_1^\circ, M_2^\circ, M_3^\circ$

Some Considerations

Our approach gives meaninful answers when the sensitivity of the system on variation of the uncertain parameters is not too high

The approach can also be used for parameter estimation by iteratively

1. constructing an abstract model with wide intervals
2. checking properties known to hold
3. refine the model until model checking gives [1,1] as result

The efficiency of the approach depends very much on the number of uncertain parameters

- the translation of an IMC into a MDP is exponential in the number of parameter intervals

Further Developments

We are working at a continuous time approach, in which the elapsing of time is taken into account

References

P. Drábik, A. Maggiolo-Schettini and P. Milazzo. **Modular Verification of Interactive Systems with an Application to Biology**. Int. Workshop CS2Bio'10, in press.

P. Drábik, A. Maggiolo-Schettini and P. Milazzo. **Dynamic Sync-programs for Modular Verification of Biological Systems**. Proc. of Int. Workshop NCMA'10.

R. Barbuti, F. Levi, P. Milazzo and G. Scatena. **Probabilistic Model Checking of Biological Systems with Uncertain Kinetic Rates**. Int. Conference on Reachability Problems (RP'06), LNCS 5797, pp. 64-78, 2009.