Accelerated Simulation of Hybrid Biological Models with Quasi-disjoint Deterministic and Stochastic Subnets

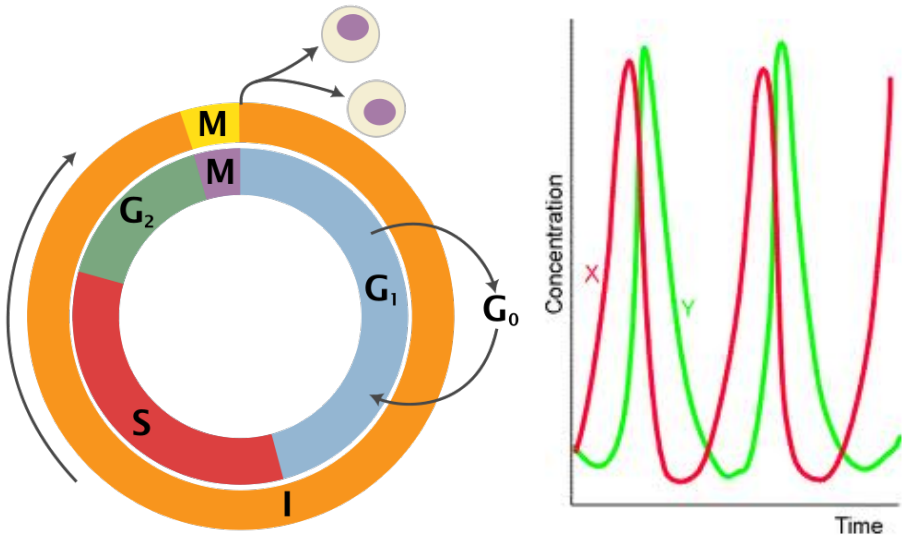
Mostafa Herajy [1] and Monika Heiner [2]

[1] Department of Mathematics and Computer Science,
Faculty of Science, Port Said University, Egypt

[2] Computer Science Institute,
Brandenburg University of Technology, Cottbus, Germany

Grenoble, October 20th 2016

Outline

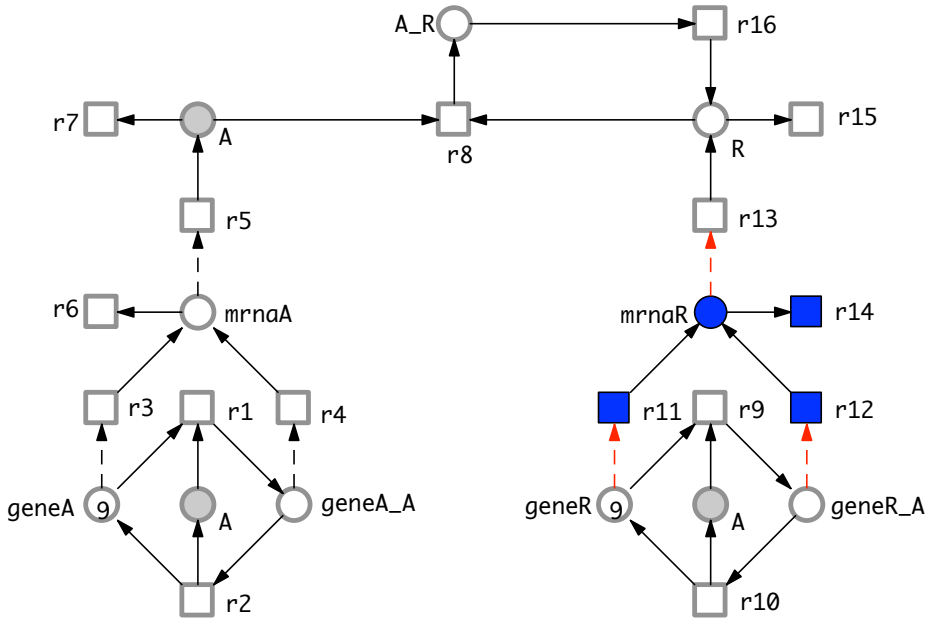


Prolog

We use Generalised Hybrid Petri Nets ($\mathcal{GHPN}$) to

- construct and
- graphically visualise

the models.



$\mathcal{GHPN}$ Elements

Places

Transitions

Arcs

$\mathcal{GHPN}$ Elements

Places

Continuous

Discrete

Transitions

Arcs

$\mathcal{GHPN}$ Elements

Places

Continuous

Discrete

Transitions

Continuous

Stochastic

Immediate

Deterministic

Scheduled

Arcs

$\mathcal{GHPN}$ Elements

Places

Continuous

Discrete

Transitions

Continuous

Stochastic

Immediate

<1>

Deterministic

[_SimStart,1,_SimEnd]

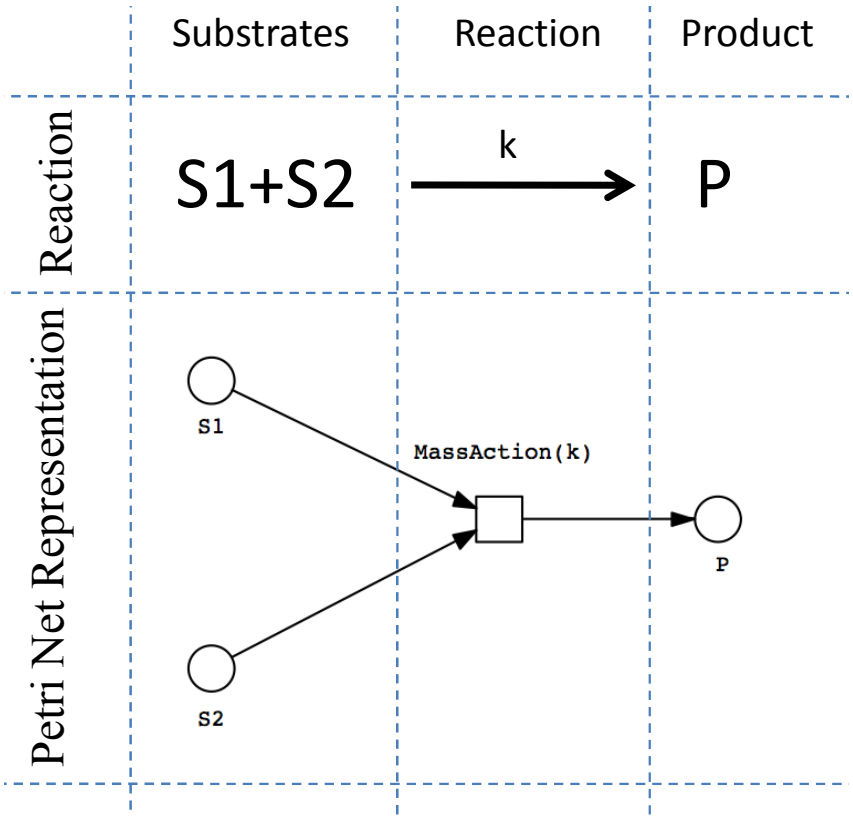
Scheduled

Arcs

Standard Read Inhibitor Equal Reset Modifier

M. Herajy and M. Heiner, NAHS (2012)

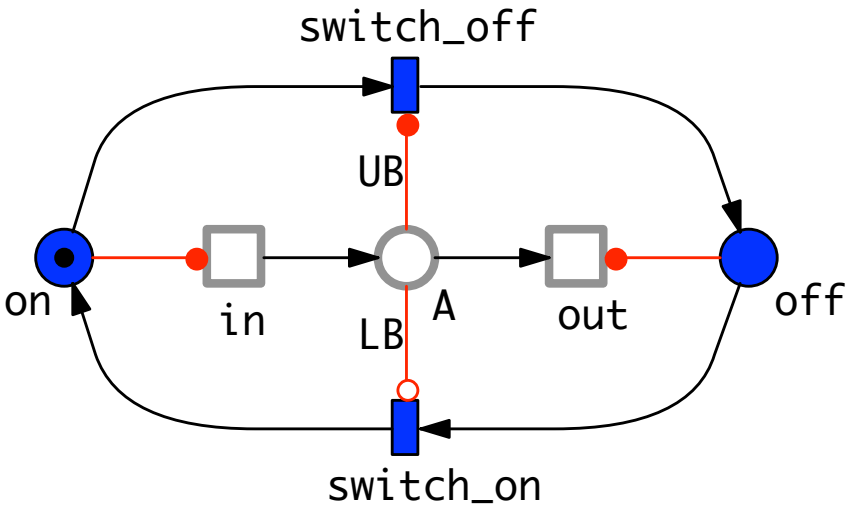
From Reactions to Petri Nets

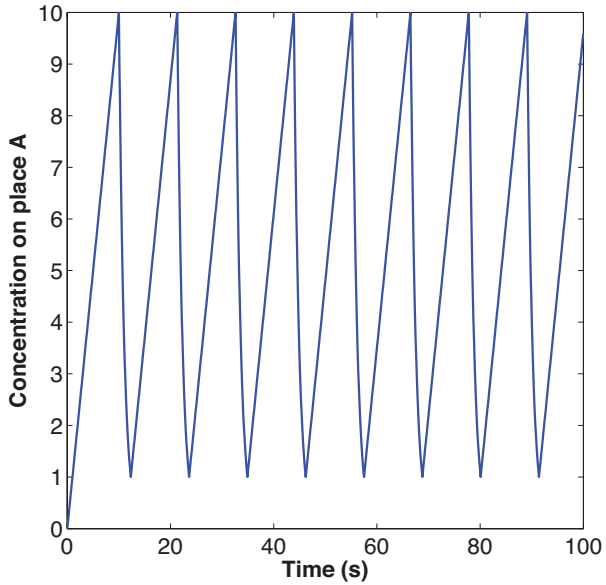


Notations

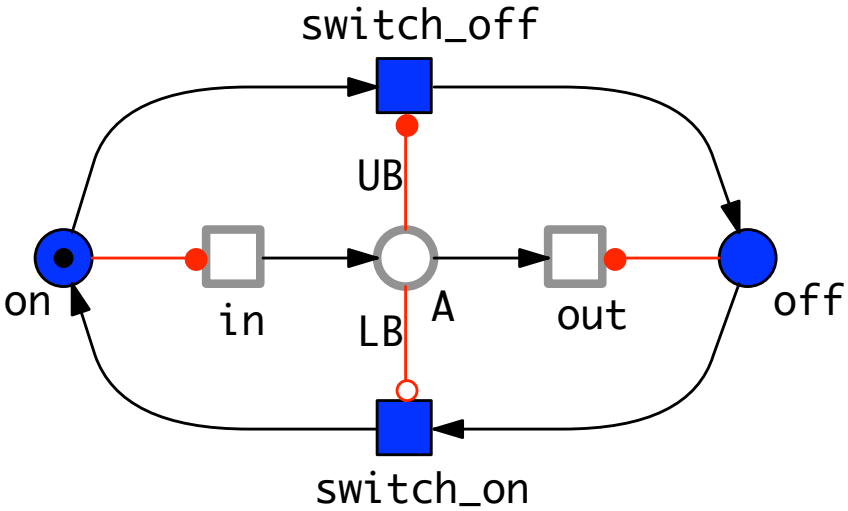
- Species $=\{S_1, S_2, P\}$
- Substrates $=\{S_1, S_2\}$
- Product $=\{P\}$
- Rate Constant $=\{k\}$
- Reaction Rate: $MassAction(k) = k \cdot S_1 \cdot S_2$

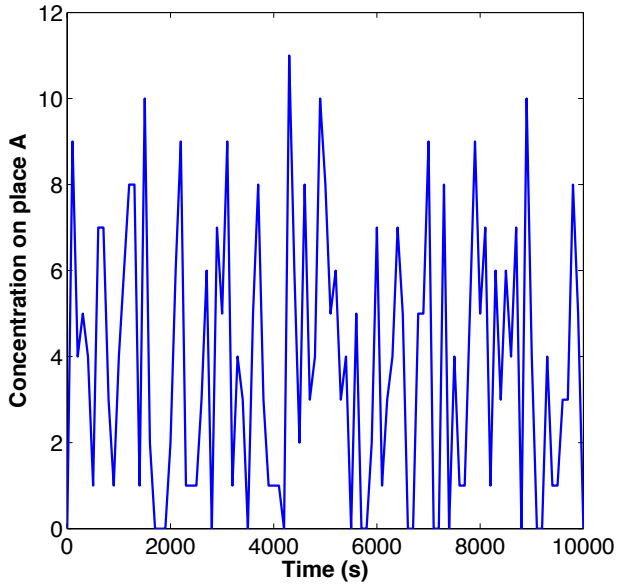
$\mathcal{GHPN}$ Example with Immediate Transition





$\mathcal{GHPN}$ Example with Stochastic Transition





Introduction

- Hybrid simulation (of biological reaction networks) combines deterministic and stochastic regimes to execute a model.
- It can be used as an alternative approach when it is not possible to perform a pure stochastic simulation.
- It provides a trade-off between accuracy and efficiency of model execution.

Introduction: How Does It Work?

Repeat the following steps:

- (Re)initialise the ODE solver
- Numerically integrate the system of ODEs until a stochastic event is to occur
- Find the index of the stochastic reaction to occur
- Fire this stochastic reaction
- Update the propensities of the stochastic reactions

Introduction: How Does It Work?

- Crucial for hybrid simulation performance: calculation of exact time when next stochastic reaction is to occur
- Can be done, e.g., using

$$\int_{t}^{t+\tau} a_0^s(\mathbf{x})dt + log(p_1) = 0\,, \tag{1}$$

where

- $\mathbf{x}$ – state vector of the model at time t,
- a_0^s – cumulative propensity of stochastic reactions,
- p_1 – random number generated from $U(0,1)$.

Haseltine, Rawlings, J. Chem. Phys. (2002)

Introduction: How Does It Work?

- Similarly, the index of the next reaction to fire can be selected as the first index μ satisfying

$$\sum_{j=1}^{\mu} a_j^s(\mathbf{x}) > p_2 a_0^s(\mathbf{x}) \,, \tag{2}$$

where

- a_j^s – propensity of the j^{th} slow reaction,
- p_2 – random number generated from $U(0,1)$.

Haseltine, Rawlings, J. Chem. Phys. (2002)

Introduction: a Performance Issue

- Each time a stochastic event takes place, a discontinuity in the system of ODEs may occur.
- To deal with discontinuities, the ODE solver must be reinitialised after the firing of each stochastic reaction.
- Frequent reinitialisation of the ODE solver introduces additional computational overhead.
- We assume a modular design of the ODE solver and the stochastic simulator;
  this requires adaptive step-size ODE solvers, recording accuracy and history information.

Introduction: a Performance Issue

Do we need to accept

all these reinitialisations?

Our Proposal

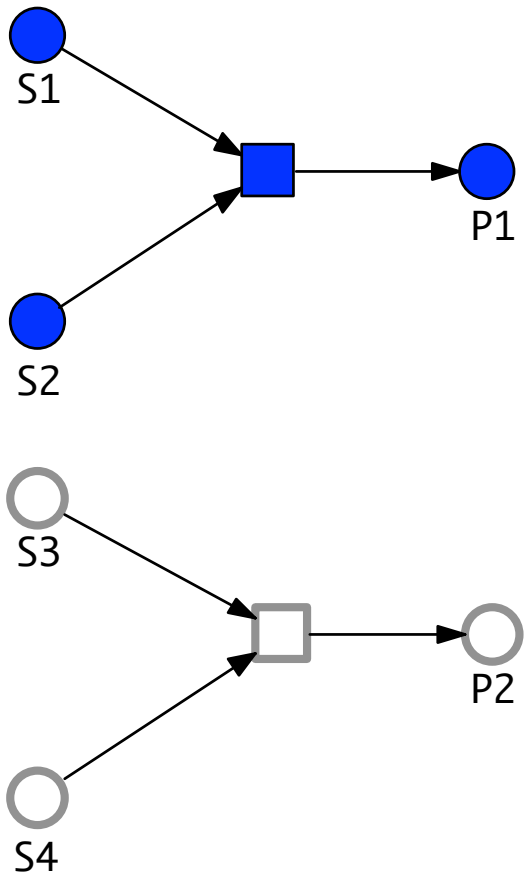
To overcome this problem, we classify the set of stochastic reactions into three groups according to their relation to the deterministic regime:

1. reactions with no dependency
2. reactions with direct dependency – interface reactions
3. reactions with indirect dependency

Completely Independent Reactions

- Such reactions do not share any species with the reactions in the deterministic regime.
- Also, their substrates and products are not used to define reaction rates in the deterministic regime.

Completely Independent Reactions: (Example)



The two reactions:

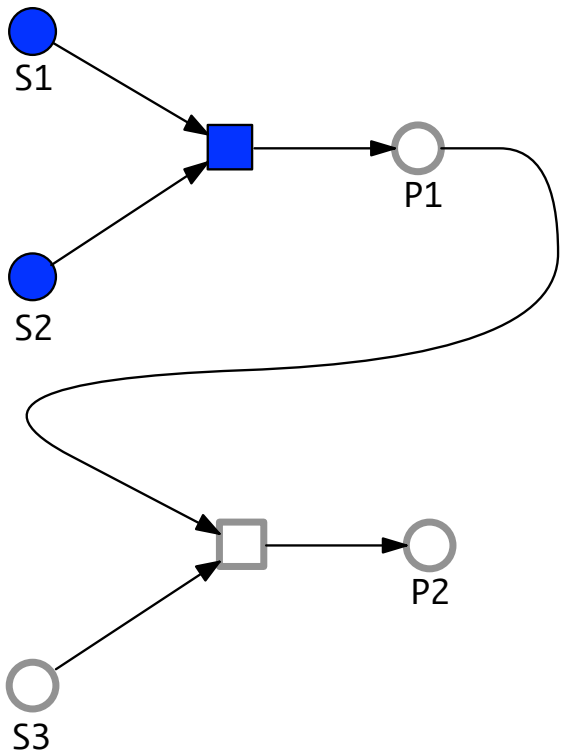
$$S_1 + S_2 \xrightarrow{k_1} P_1$$
$$S_3 + S_4 \xrightarrow{k_2} P_2$$

are independent of one another.

Reactions with Direct Dependency

- Such reactions share certain species with the continuous regime.
- Their substrates and products may be used to define reaction rates in the deterministic regime.

Reactions with Direct Dependency (Example)



The two reactions:

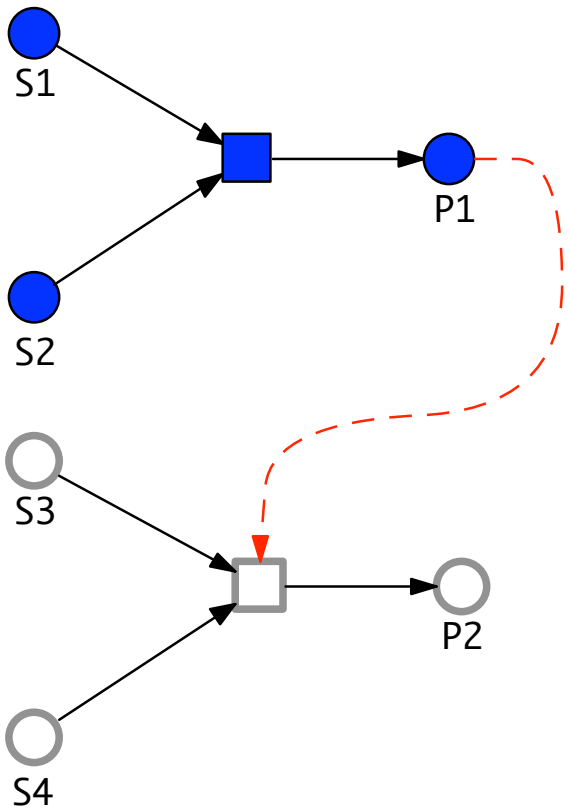
$$S_1 + S_2 \xrightarrow{k_1} P_1$$
$$P_1 + S_3 \xrightarrow{k_2} P_2$$

have a direct dependency.

Reactions with Indirect Dependency

- Such reactions do not share any species with the continuous regimes.
- Their substrates and products are used to define reaction rates in the deterministic regime.

Reactions with Indirect Dependency (Example)



The two reactions:

$$S_1 + S_2 \xrightarrow{k_1} P_1$$
$$S_3 + S_4 \xrightarrow{k_2 * P_1} P_2$$

have an indirect dependency.

Our Proposal, Idea 1

Following this classification, we reinitialise the ODE solver in case of:

- a stochastic reaction with direct dependency fires,
- for some special cases of stochastic reactions with indirect dependency.
- We use an idea similar to the dependency graph.

Our Proposal, Idea 2

- However, Eq. (1) still requires the reinitialisation of the ODE solver each time a stochastic event occurs.
- One workaround is to approximate (1):

$$\int_t^{t+\tau} a_0^s(\mathbf{x})dt + log(p_1) = 0$$

by

$$a_0^s(\mathbf{x}) \cdot \Delta\tau + log(p_1) = 0\,, \tag{3}$$

where

- $\Delta\tau$ – time difference between occurrence time of the previous event and the current event,
- p_1 – random number generated from $U(0, 1)$.

Dependency Graph

- Originally been proposed by Gibson, Bruck (2000)
- Reduces the number of propensity updates following every reaction firing
- Stores for each reaction the set of other reactions that will be affected by every occurrence

Algorithm1: Extracting the Dependency Information

- We assume that the dependency graph of all reactions is already constructed
- Manipulated species: the set of species that are altered when a reaction takes place
- E.g., the set of manipulated species for the reaction $S_1 + S_2 \xrightarrow{k_1} P_1$ is $M = \{S_1, S_2, P_1\}$

Algorithm1: Extracting the Dependency Information (Cont.)

- Input: the sets of slow and fast reactions (G_s, G_f)
- Output: the set of interface reactions
- Steps:
  - Find the intersections of manipulated species for each pair of reactions ($r_1 \in G_s$, $r_2 \in G_f$)
  - If intersection set is not empty, mark r_1 as an interface reactions

Algorithm1: Extracting the Dependency Information

Algorithm 1 Finding Interface Reactions

Require: G_s the set of slow reactions;
Require: G_f the set of fast reactions;
1: $R^* = \phi$ {the set of marked interface reactions is initially empty}
2: **for each** r_i in G_s **do**
3: let S_i denotes the set of manipulated species when r_i fires;
4: **for each** s_{ij} in S_i **do**
5: Find the set of other reactions, R_{ij}, that manipulate s_{ij} when they fire;
6: **if** $\exists r_j \in R_{ij}$ **and** $r_j \in G_f$ **then**
7: Add r_i to R^*; {Mark r_i as an interface reaction}
8: **end if**
9: **end for**
10: **end for**
11: **return** R^*;

Algorithm 1: Example

#	Slow Reactions	M. Species
r_1	$\phi \xrightarrow{s} A$	$\{A\}$
r_2	$A \xrightarrow{d} \phi$	{A}
r_3	$\phi \xrightarrow{s} B$	{B}
r_4	$B \xrightarrow{d} \phi$	{B}
r_5	$A + B \xrightarrow{k_1} B + C$	{A, B,C}

#	Fast Reactions	M. Species
r_6	$C + E \xrightarrow{k_2} D$	{C,D,E}
r_7	$D \xrightarrow{k_3} C + E$	{C, D, E}
r_8	$D \xrightarrow{dd} \phi$	{D}

- $\{r_1 - r_4\}$ – independent reactions
- $R^* = \{r_5\}$ – dependent/interface reaction

Algorithm 2: Accelerated Hybrid Simulation

- Makes use of the information collected from Algorithm 1
- Initialises the ODE solver only when the firing stochastic reaction belongs to the marked list

Algorithm 2: Accelerated Hybrid Simulation (Cont.)

Algorithm 2 Accelerated Hybrid Simulation

Require: G_s and G_f: the sets of slow and fast reactions respectively;
Require: R^* the set of reactions marked as interface reactions;
1: Initialise the ODE solver with the initial concentration of the variables in G_f;
2: set $\tau = \tau_{old} = 0$;
3: **while** we did not reach end simulation time **do**
4: Generate two random numbers p_1 and p_2 from the uniform distribution;
5: **repeat**
6: Numerically integrate the system of ODEs;
7: **until** $a_0^s(\mathbf{x}) \cdot (\tau - \tau_{old}) + log(p_1) = 0$ {cf., Eq., (5)}
8: **Update**$(a(r_i), a_0^s)$, $\forall r_i \in G_s, \forall r_j \in G_f$: **Base**$(r_i) \cap$ **Manipulated**$(r_j) \neq \phi$;
9: Find the reaction r_μ that satisfies (2) using p_2;
10: Fire r_μ and update the system state as well as the current time τ;
11: **Update**$(a(r_i), a_0^s)$, $\forall r_i$: **Base**$(r_i) \cap$ **Manipulated**$(r_\mu) \neq \phi$;
12: Set $\tau_{old} = \tau$
13: **if** $r_\mu \in R^*$ **then**
14: Reinitialise the ODE solver

Algorithm 2: Accelerated Hybrid Simulation (Cont.)

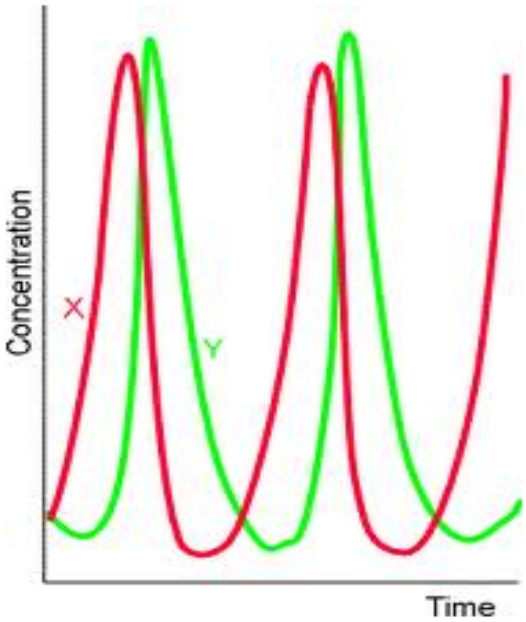
- In some cases, reactions with indirect dependency can cause discontinuity
- For example, when an enzyme value goes from 0 to 1, or vice versa
- The state of such a species can be monitored for such behaviour
- The ODE is reinitialised only when a species values flipping form 0 to 1, or vice versa

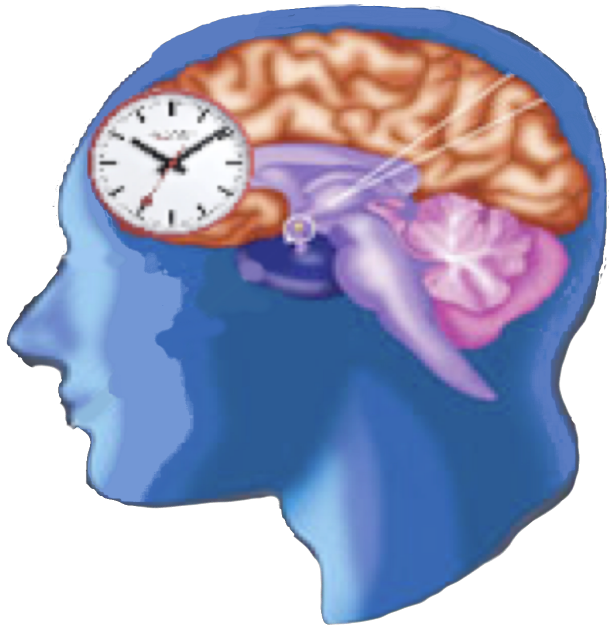
Case Studies

We use three case studies to test the proposed method:

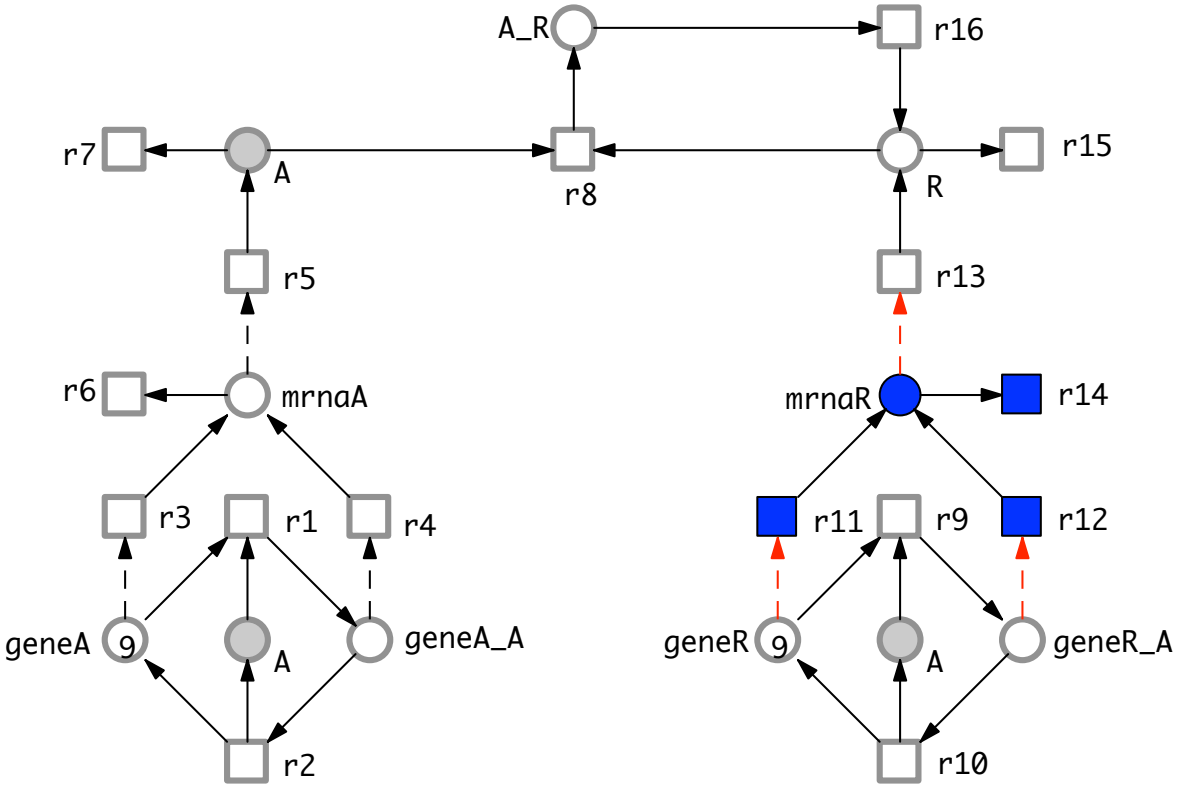
1. Circadian Oscillation
2. Eukaryotic Cell Cyle
3. Yeast Cell Cyle

Case Study: Circadian Oscillation





Case Study: Circadian Oscillation



Vilar et al., PNAS (2002)

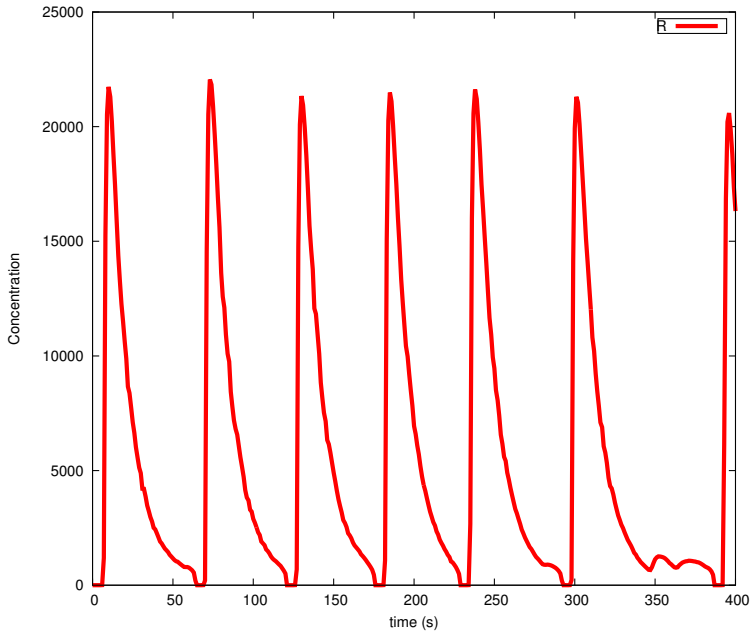
Blätke, Heiner, Marwan, BioModel engineering with Petri nets (2015)

Case Study: Circadian Oscillation

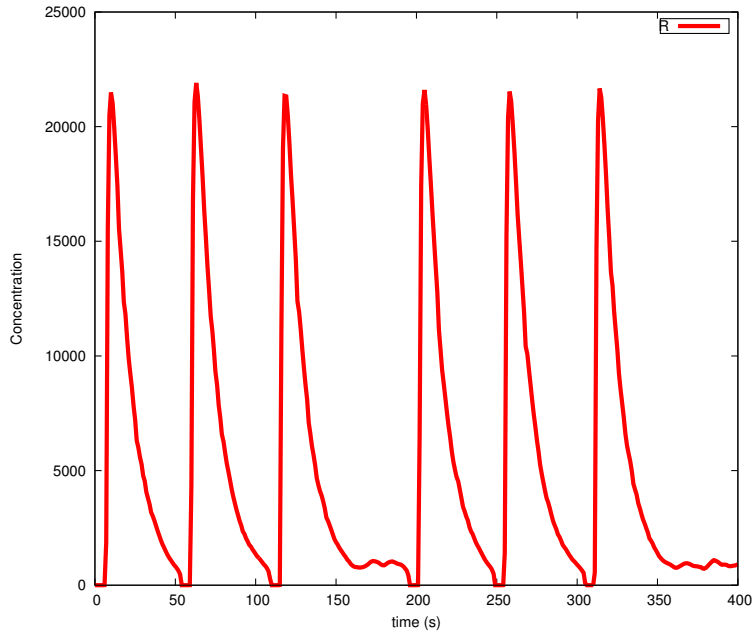
- The model consists of 9 species and 16 reactions.
- The number of generated stochastic events for both simulators are comparable.
- The accelerated simulation algorithm is about **three times faster** than the exact method.

Case Study: Circadian Oscillation

Simulation results of the circadian oscillation model (single runs):



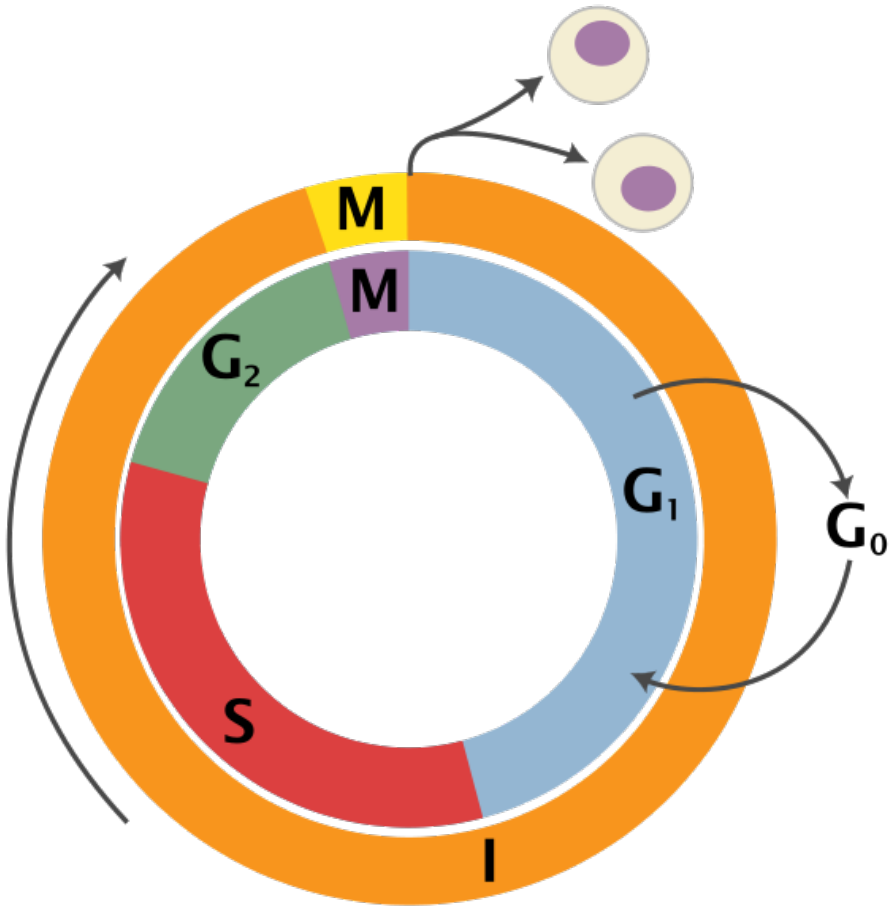
exact



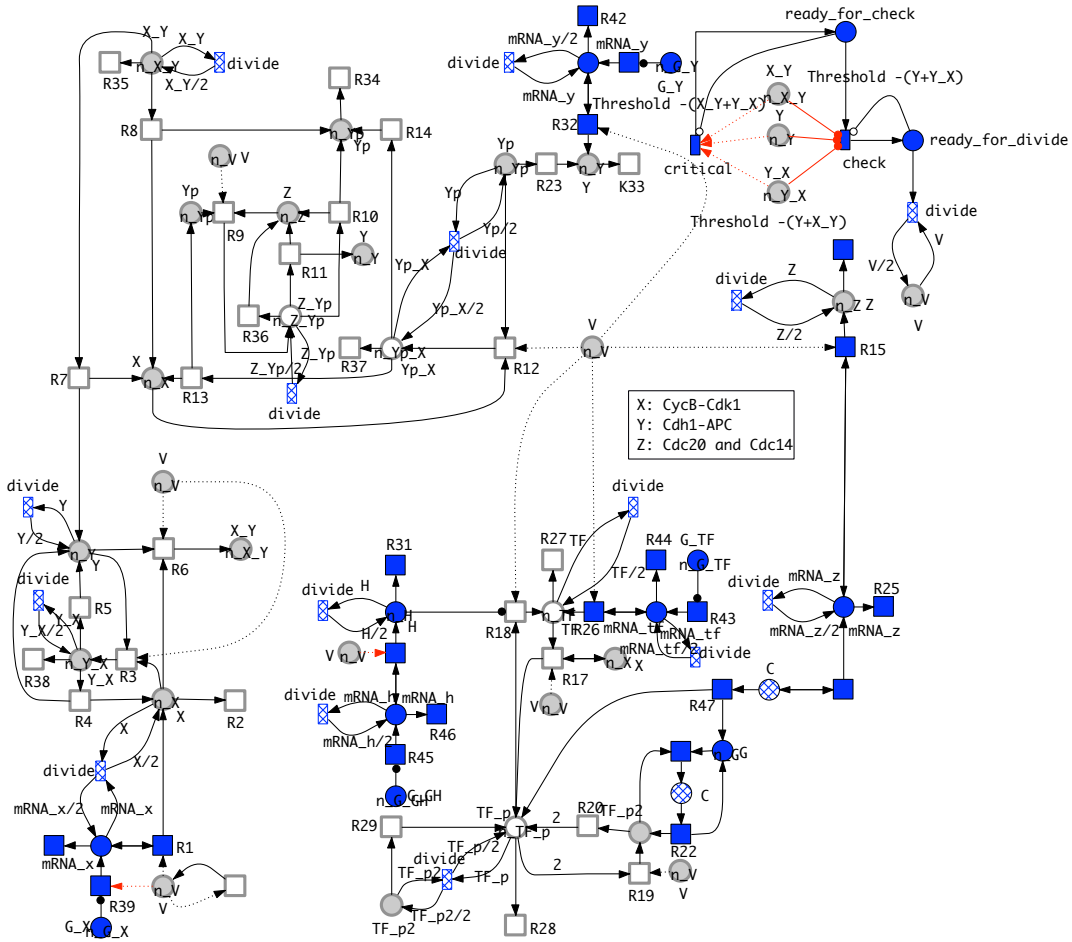
accelerated

Case Study: Cell Cycle Regulation

- S phase (synthesis)
- G2 gap
- M phase (mitosis)
- G1 gap



Case Study: Eukaryotic Cell Cycle



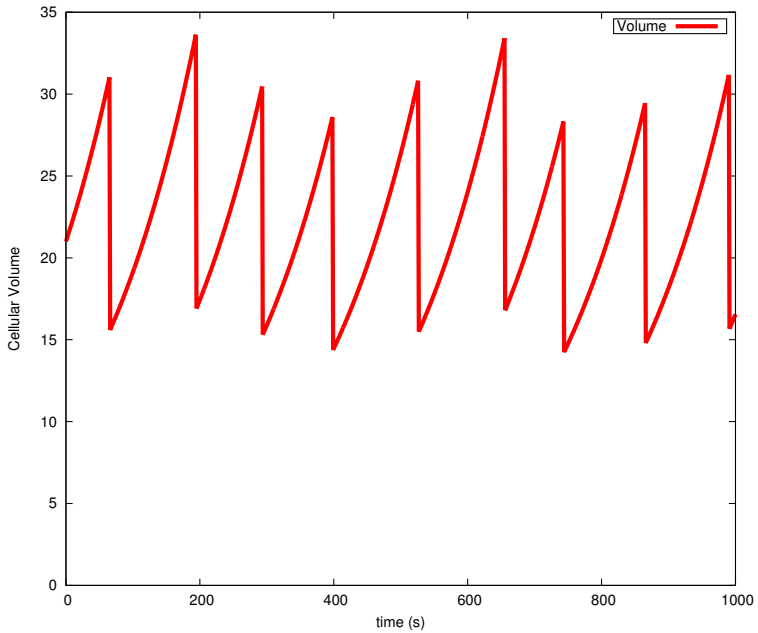
Herajy, Schwarick, Heiner, ToPNoC (2013)

Case Study: Eukaryotic Cell Cycle

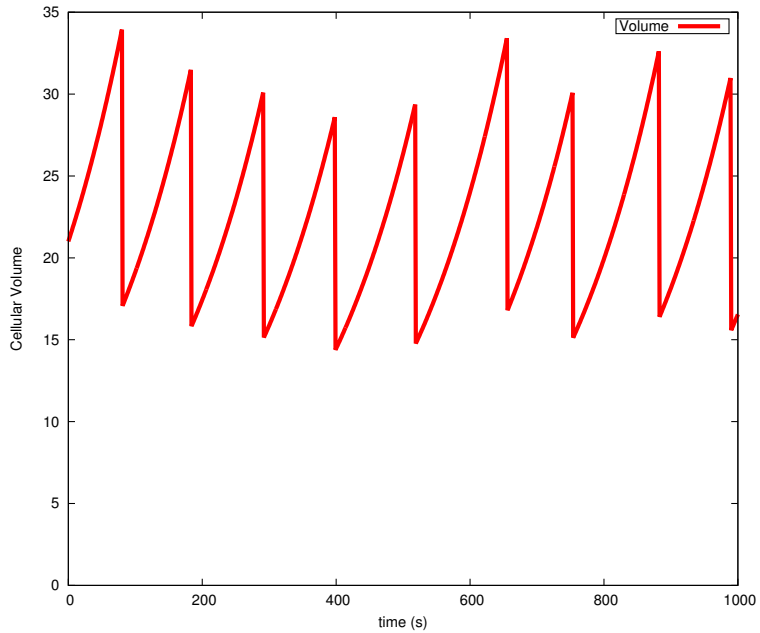
- The model consists of 26 species and 48 reactions.
- The number of generated stochastic events for both simulators are comparable.
- The accelerated simulation algorithm is about **two times faster** than the exact method

Case Study: Eukaryotic Cell Cycle

Simulation results of the eukaryotic cell cycle (single runs):



exact



accelerated

Case Studies: Yeast Cell Cycle

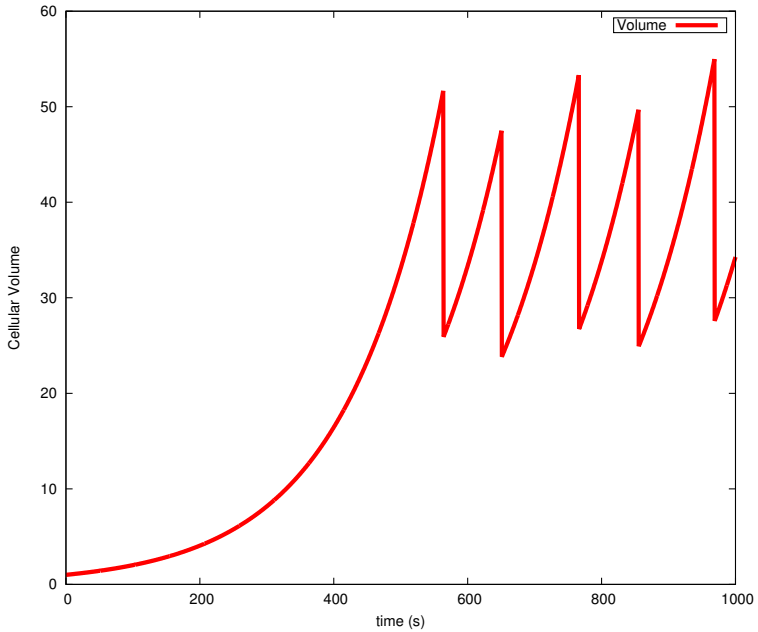
- This hybrid model is based on the stochastic one by Barik et al., 2010
- The model consists of:
  - A set of phosphorylation and dephosphorylation reactions of many proteins
  - Synthesis and degradation of mRNAs
- Phosphorylation and dephosphorylation reactions are simulated deterministically, and
- Reactions related to mRNAs are simulated stochastically

Case Study: Yeast Cell Cycle

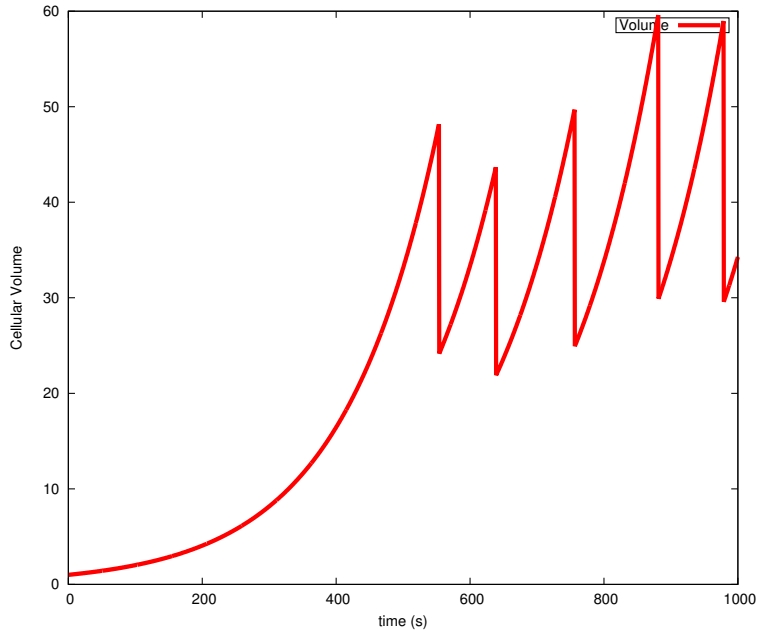
- We use coloured hybrid Petri nets to construct and simulate this model.
- The model consists of 60 species and 190 reactions.
- The number of generated stochastic events for both simulators are comparable.
- The accelerated simulation algorithm is about **10 times faster** than the exact method.

Case Study: Yeast Cell Cycle

Simulation results of the yeast cell cycle (single runs):



exact



accelerated

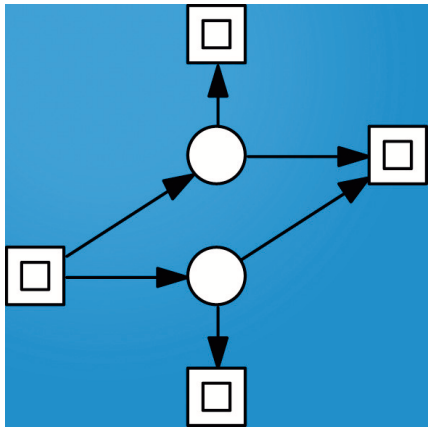
Case Studies: Summary

Criteria/models	Circadian Oscillation	Eukaryotic Cell Cycle	Yeast Cell Cycle
# species	9	26	60
# reactions	16	48	190
# stochastic reactions	3	19	19
# deterministic reactions	13	29	171
# stochastic events (exact)	35,650	780,318	112,908
# stochastic events (accelerated)	35,533	776,192	112,789
# interface reactions	0	8	0
# indirect dependent reactions	3	4	15
Run time (exact) (s)	3.8	731	495
Run time (accelerated) (s)	1.278	445	53

Implementation

Snoopy

A tool for animating and simulating Petri nets.



S^4

Snoopy Simulation and Steering Server

Heiner et al. Petri nets (2012)

Herajy, Heiner, Petri nets (2014)

Conclusions 1

- We have presented an approach for improving the performance of hybrid simulation algorithms.
- The suggested improvements will be useful to cope with the rapid growth of (biological) models.
- For smaller models, the accelerated algorithm is about three times faster than the exact method.
- For larger models, there is a substantial improvement in terms of runtime (10 times faster).

Conclusions 2

- This result is due to saving the ODE solver from repeating the work required to build accuracy and history information.
- Therefore, as soon as the model size is increased, the simulator performance is also improved.

Future Work

- Applying the proposed algorithms to more examples.
- Investigating dynamic partitioning in combination with the presented algorithm.

Thank You