An Experience on Probabilistic Model Checking and Stochastic Simulation to Design Self-Organizing Systems

Matteo Casadei *Mirko Viroli*

Alma Mater Studiorum – Università di Bologna

{m.casadei,mirko.viroli}@unibo.it

The 2009 IEEE Congress on Evolutionary Computation (IEEE CEC 2009)

Session on *Cooperation and Self-Organisation*

Trondheim, Norway

May 20, 2009

Preface

Self-organizing distributed systems

- *Self-organization*: features and motivations
  - local design vs. global emergence of desired behaviour
  - promote system adaptation in open and highly dynamic environments
  - exploit stochastic behaviour to tackle unpredictability
- Example systems (generally nature-inspired ones)
  - Field-based retrieval of agents in mobile networks
  - Pheromone-based formation of paths to resources
  - Ant-inspired relocation of items in networks
  - Ecology-inspired management of pervasive services

Can we design emergence?

- How often the desired property emerge?
- Any tool to verify this during design?

Goal of the paper

1. Discuss a hybrid (H) technique of approximate model checking, combining:
   - stochastic simulation (SS)
   - probabilistic model checking (PMC)
2. Present a problem called collective sort as a running example
3. Analyze it with the PRISM tool
4. Draw some conclusion about this experience

Approach 1: Stochastic Simulation

Reference formal models

Systems modelled as (stochastic) transition systems

- Discrete-time Markov Chains (DTMC): probability of a transition
- Continuous-time Markov Chains (CTMC): probability + average time

Which analysis?

- Experiments by so-called stochastic simulation algorithms (i.e. the Monte-Carlo approach)
- A rather standard approach, with pros and cons
  - Analysis of few cases, to detect coherence w.r.t. intended behaviour
  - But what about completeness and confidence?
- After initial validation, can we formally verify its behavior?

Approach 2: Probabilistic Model Checking (PMC)

An extension of Model Checking

1. System to be verified modelled as a DTMC or CTMC
2. Properties to check expressed by some probabilistic temporal logics (e.g. PCTL or STL)
   - Prob. of reaching a `fault` state: $\mathbb{P}_{\leq 0.01}(t\, U\, \texttt{fault})$
   - Exact prob. of reaching `final` state within time k: $\mathbb{P}_{=?}(t\, U^{\leq k}\, \texttt{final})$
3. A model checker "builds" the model (complete transition graph) and performs an exhaustive search for the property

Traditional applications of PMC

- Randomised algorithms (leader election, byzantine protocols,..)
- Communication/Multimedia protocols (CSMA/CD, IPv4 Zeroconf, ..)
- Security protocols (PIN cracking schemes, contract signing, ..)

Is PMC feasible for Self-Organizing Systems?

The case of self-organizing systems

Main goal:

- exploit PMC for verifying the emergence of desired behavior in a SOS

Issues:

- Efficiently model self-organizing systems by DTMC/CTMC
- State-space explosion is the norm, since SOS are not "small" systems
  - this may make checking impractical (in space & time)

Approach 3: A Hybrid Approach

Approximate PMC [Hérault et.al. 2004]

Tackle the state-space explosion, probabilistically:

- Explore a subset of state-space through a (possibly high) number of stochastic simulations (requires less time and less space)
- Result: probability for the property to hold, with known confidence

Three key parameters

1. Number of simulation runs N
2. Approximation ϵ: the desired precision on the obtained probability
3. Confidence factor δ: probability that the result is not reliable

   Definition of ϵ and δ: $Prob[|M_{exact} - M_{approx}| \leq \epsilon] \geq 1 - \delta$

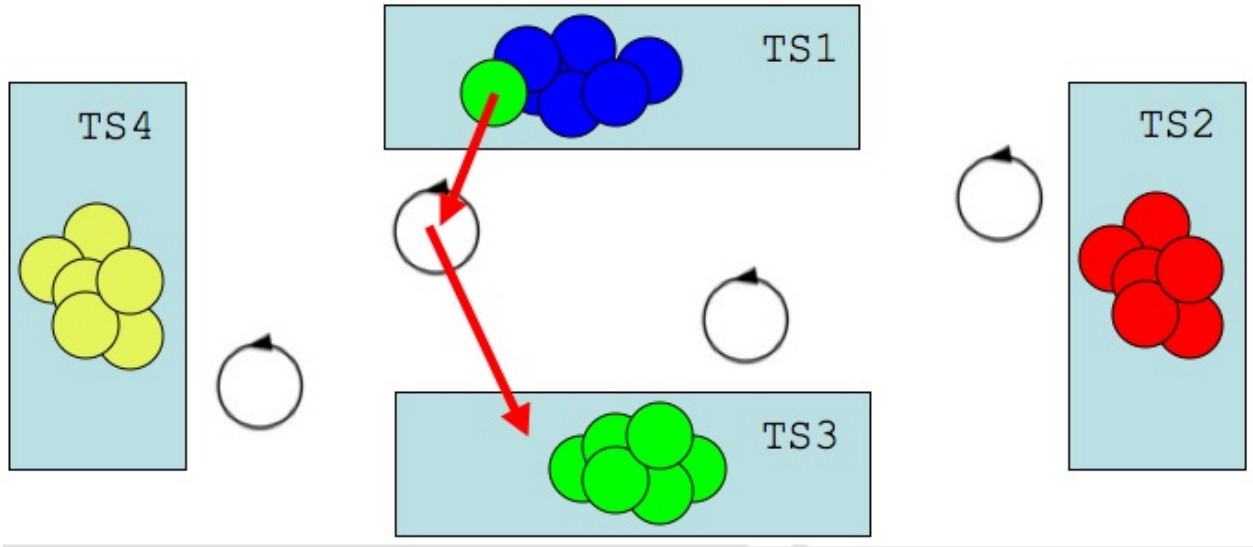
- Parameters are linked: $N \geq 4log(\frac{2}{\delta})/\epsilon^2$
- Example choice: $(\epsilon = 0.01,\ \delta = 0.01,\ N \simeq 90'000)$.

A case study: The Collective Sort Problem

A simple paradigmatic case of self-organization

- Inspired by brood and larvae sorting by ants
- Useful to adaptively relocate data-items in a distributed system
- See: *Casadei et.al., Science of Computer Programming, 2009*

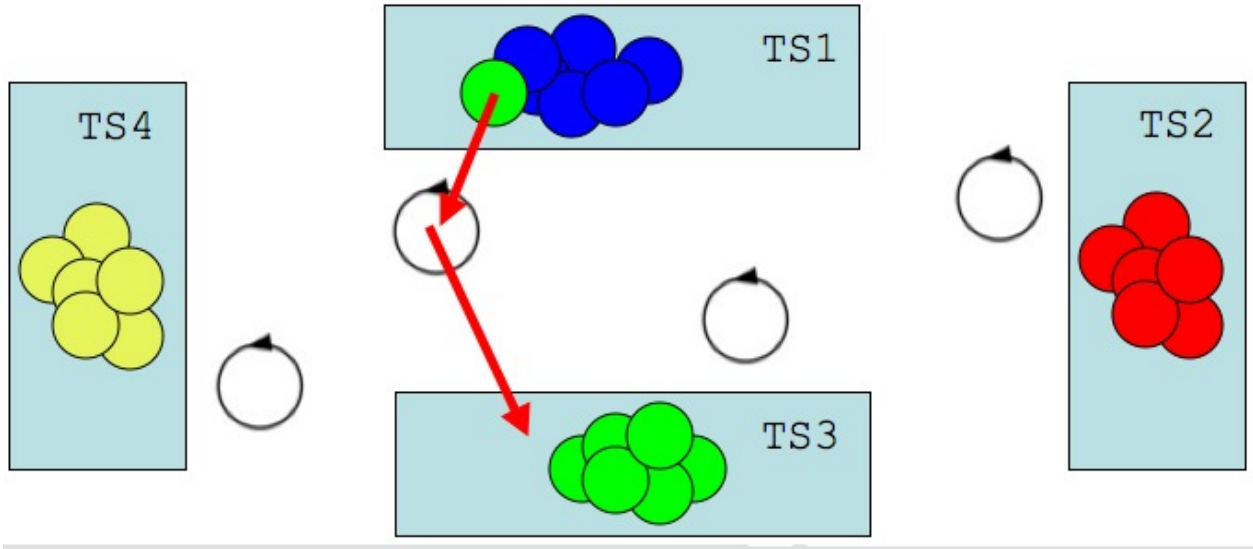
Architecture



Problem Definition

- Take a set of n tuple spaces $(S_1, \ldots, S_n)$..
- ..holding a dynamic set of tuples of n different kinds $(K_1, \ldots, K_n)$
- Design a self-organizing solution where:
  - Locally: a tuple is moved from S_i to S_j due to pointwise observations
  - Globally: each tuple kind gets collected into a different space

Solution Strategy



A sorting service transparent to "user agents"

One sorter agent for each space S

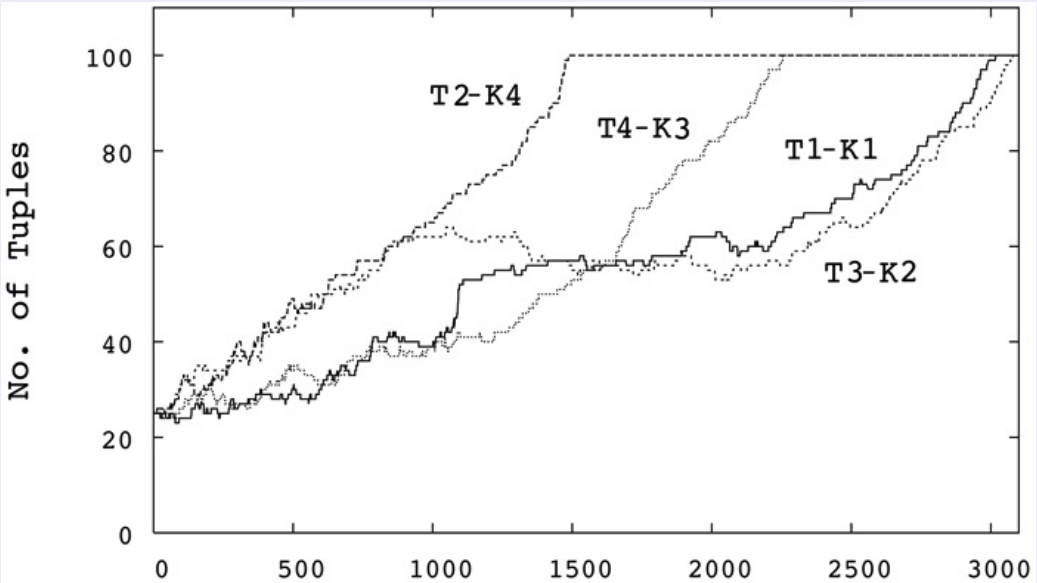
- Its goal is to move unwanted tuples away from S, at a certain rate
- Decisions based on pointwise and probabilistic read operations
  - $T_K := read(S_1)$, $T_{K'} := read(S_3)$, if $K \neq K'$ move $T_{K'}$ from S_1 to S_3
- Avoiding local/global counting operations

Collective Sort: Example

An example simulation scenario

- 4 spaces, 25 tuples per kind in each space

Result: Complete ordering is eventually reached



The problem we want to study

Is it always converging? Let's use model checking..

Collective Sort in PRISM (www.prismmodelchecker.org)

Explicitly writing down all transitions

Chunk of code ($N = 3$):

```
//***S1 --> S2****,
// k1 drawn in S1,  k2 drawn in S2
[] k11>0 & k22>0 & k21>0 & k22<n -> (1/6 * k11/tot1 * k22/tot2)
   : (k21'=k21-1) & (k22'=k22+1);
// k1 drawn in S1,  k3 drawn in S2
[] k11>0 & k32>0 & k31>0 & k32<n -> (1/6 * k11/tot1 * k32/tot2)
   : (k31'=k31-1) & (k32'=k32+1);
```

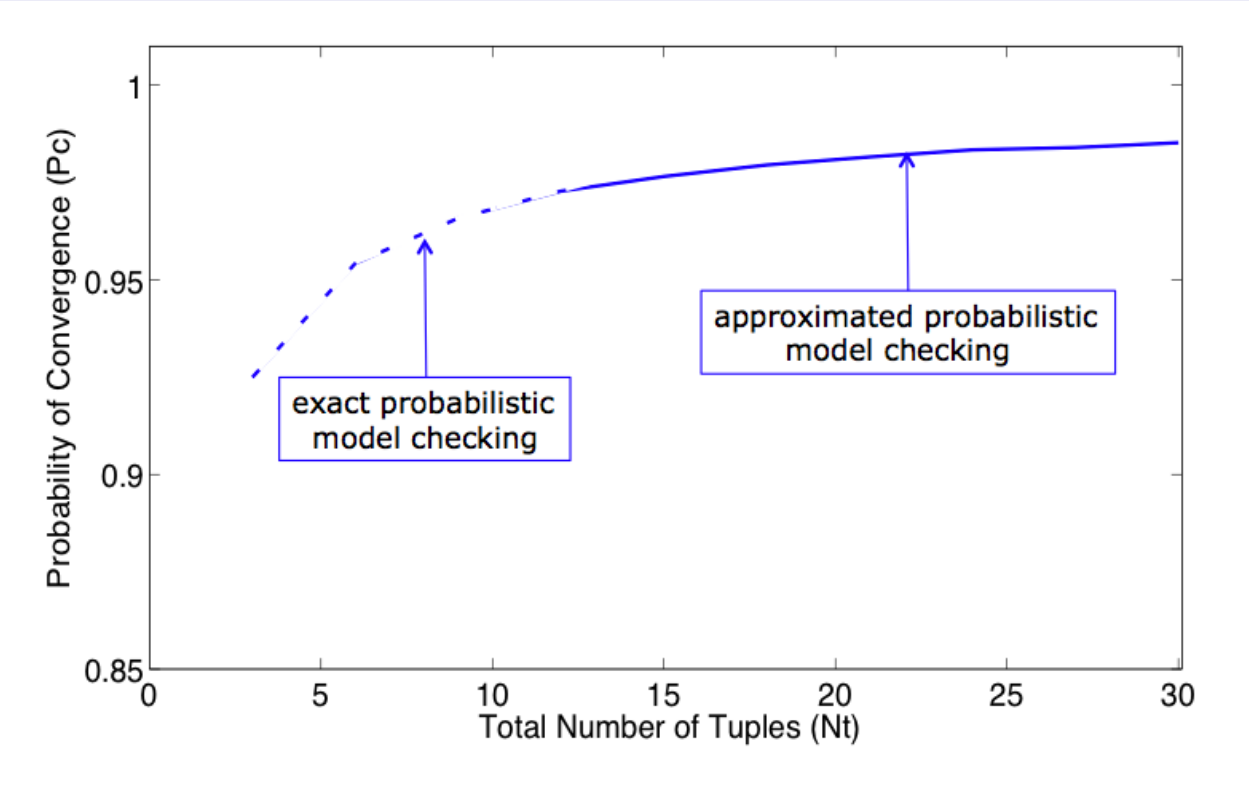
Verifying complete sorting

```
P=? [true U "complete-sorting"]

label "complete-sorting" =  ok1 & ok2 & ok3 & tot1=n & tot2=n & tot3=n;
formula ok1 = k11=n | k21=n | k31=n;
formula ok2 = k12=n | k22=n | k32=n;
formula ok3 = k13=n | k23=n | k33=n;

formula tot1 = k11 + k21 + k31;
formula tot2 = k12 + k22 + k32;
formula tot3 = k13 + k23 + k33;
```

Experiment 1: qualitative understanding

Case $N = 3$: from exact PMC to approximated PMC

$$\epsilon = 0.05,\ \delta = 10^{-10} \rightarrow N_{samples} = 16,481$$



Falling into local minima

Why full-convergence is not always reached?

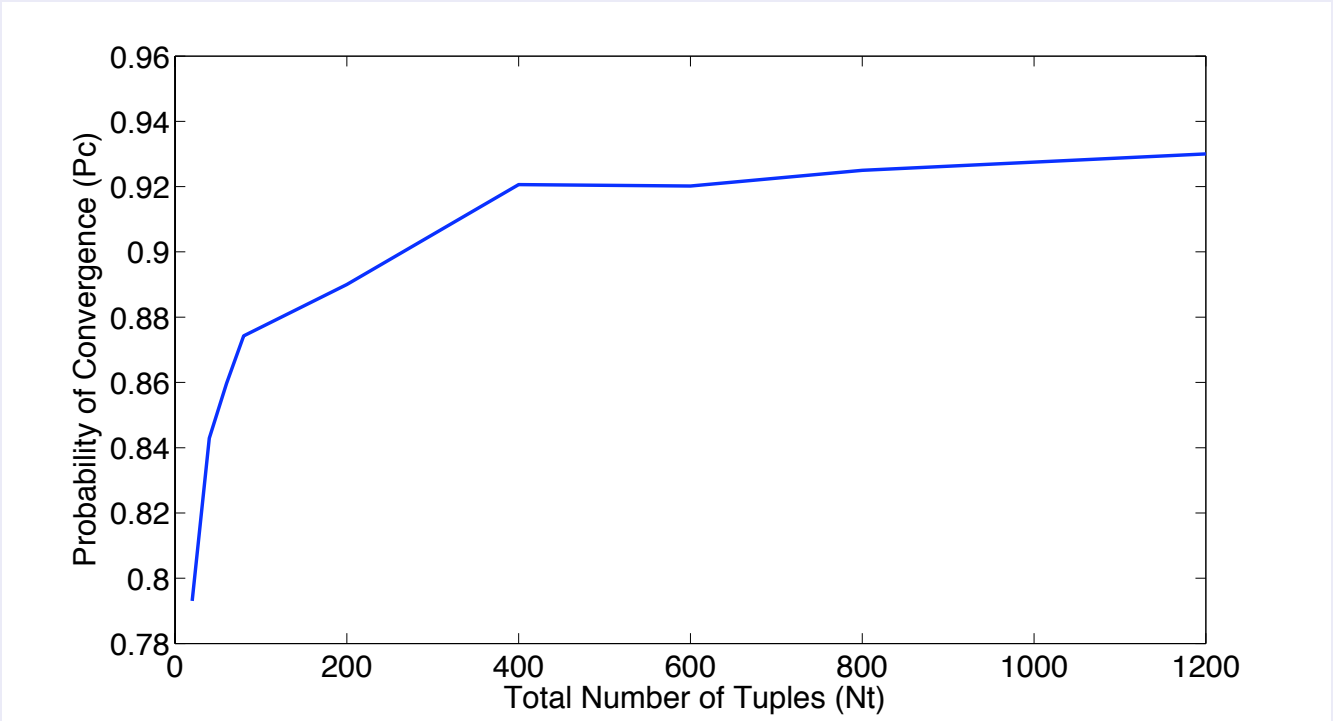
The evolution might be attracted towards states of partial order, e.g. for $N = 4$:

```
S0: K1[100] K2[  0] K3[  0] K4[  0])
S1: K1[  0] K2[ 69] K3[  0] K4[  0])
S2: K1[  0] K2[ 31] K3[  0] K4[  0])
S3: K1[  0] K2[  0] K3[100] K4[100])
```

Experiment 2: more precise quantitative measures

Case $N = 4$: larger sets of data

$$\epsilon = 0.01, \quad \delta = 0.01, \quad \rightarrow N_{samples} = 92'042$$

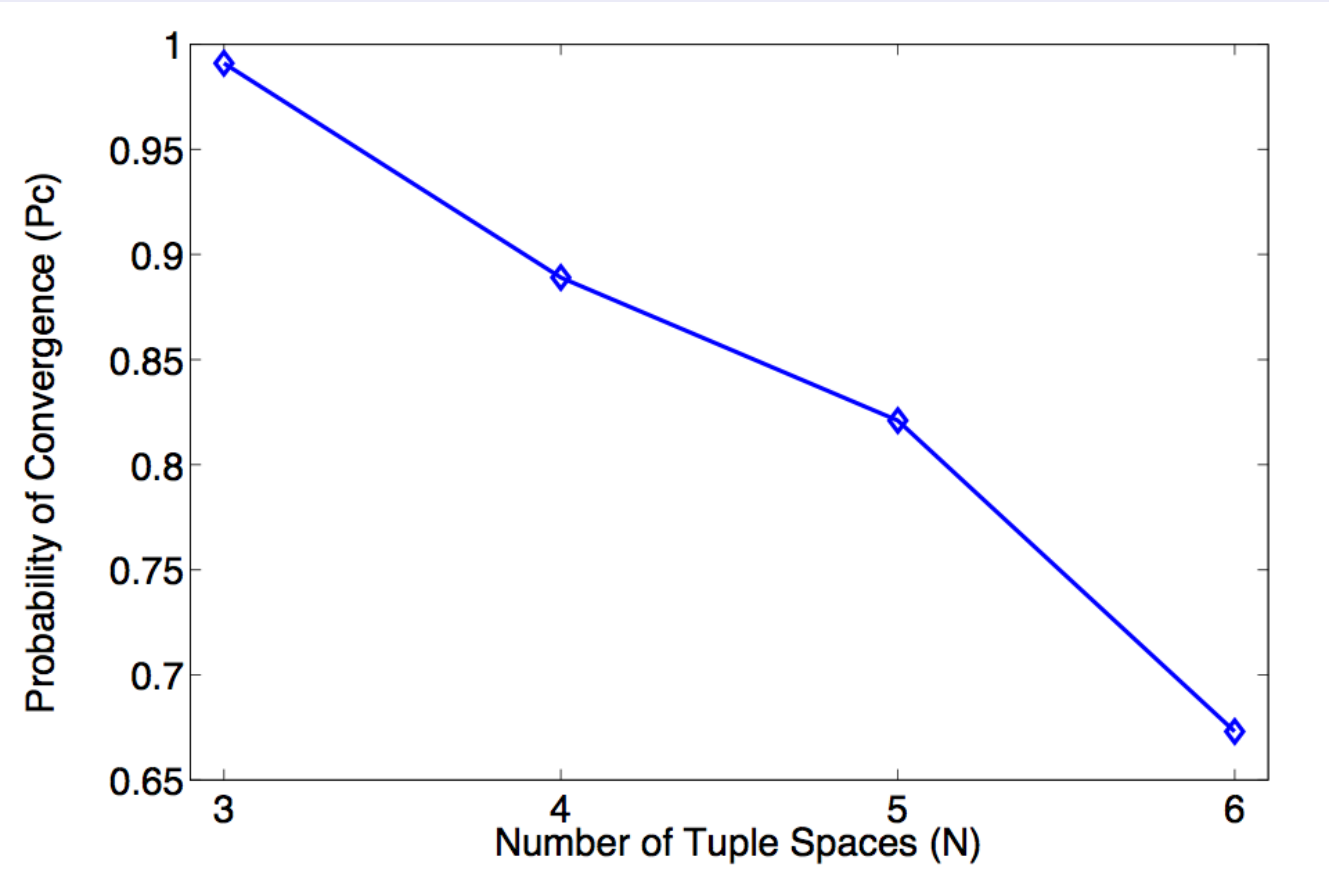


Case Nt=100: Simulation time$\simeq$3h, Average conv. time=3850 steps

Higher scales (n. of data items) imply better convergence!

Experiment 3: other simulation scenarios

Trend over increasing N, with 1000 tuples per kind



The higher N, the higher probability of reaching a local minimum!

Discussion

Collective sort

The proposed approach may incur in states of non-complete order

- With low number of data-items / With high numbers of spaces
- (Note that even partial ordering is usefull)
- A solution based on annealing is already proposed, not analyzed here

General considerations

1. Keeping the model as compact as possible promotes efficiency
2. Building a generator for your PRISM specificiation is often mandatory
3. Buy approx. PMC if you need a "simulation methodology"
4. Approx. PMC easily supports distribution, differently from exact PMC
5. Appl. to "emergence in self-organization" is worth being pursued
6. More easily applicabile if the single simulation run is short enough

Future Works

Increasing our experience with more self-org patterns

- Computational-fields diffusing dynamics
- Clustering in medium-sized nets
- Cellular automata models

Developing a methodological approach

- Building translators towards PRISM (see stochastic π-calculus)
- Automatizing the identification of proper parameters
- Identifying patterns of properties related to emergence

An Experience on Probabilistic Model Checking and Stochastic Simulation to Design Self-Organizing Systems

Matteo Casadei *Mirko Viroli*

Alma Mater Studiorum – Università di Bologna

{m.casadei,mirko.viroli}@unibo.it

The 2009 IEEE Congress on Evolutionary Computation (IEEE CEC 2009)

Session on *Cooperation and Self-Organisation*

Trondheim, Norway

May 20, 2009

Collective Sort: Space Complexity

How big is the transition graph?

Size of a problem

- Let N be the number of tuple spaces/kind
- T_i the overall number of tuples of kind K_i

Size of the graph

- #transitions in each node is $N^3(N-1)$
- #states in the graph is $\prod_{i=1}^{n} \binom{N+T_i-1}{T_i}$
- With $N = 4$, $\forall i\ T_i = 20$: #transitions $= 192$, #states $> 10^{15}$

Which analysis?

- Complete model checking shortly becomes unfeasible
- Might worth trying with approximate model checking..

Approx. PMC scenarios

Remember formula: $N \geq 4log(\frac{2}{\delta})/\epsilon^2$

Some example of parameters setting

$\epsilon = 0.1$, (δ, N) can be:

- $(0.01, 921)$, $(0.001, 1321)$, $\ldots$, $(10^{-10}, 4121)$

$\epsilon = 0.05$, (δ, N) can be:

- $(0.01, 3682)$, $(0.001, 5282)$, $\ldots$, $(10^{-10}, 16482)$

$\epsilon = 0.01$, (δ, N) can be:

- $(0.01, 92042)$, $(0.001, 132042)$, $\ldots$, $(10^{-10}, 412042)$