CSL model checking of biochemical networks with Interval Decision Diagrams

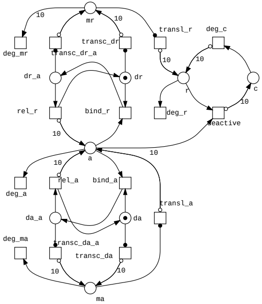
Martin Schwarick, Monika Heiner

Brandenburg University of Technology Cottbus
Computer Science Department

http://www-dssz.informatik.tu-cottbus.de/software/mc.html

September 01, 2009

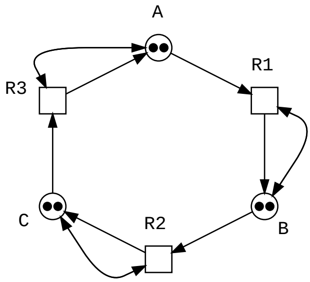
Contents


mr
10
10
deg_mr
transc_dr_a
transc_dr
transl_r
deg_c
dr_a
dr
r
c
rel_r
bind_r
deg_r
deactive
a
rel_a
bind_a
deg_a
transl_a
da_a
da
deg_ma
transc_da_a
transc_da
ma

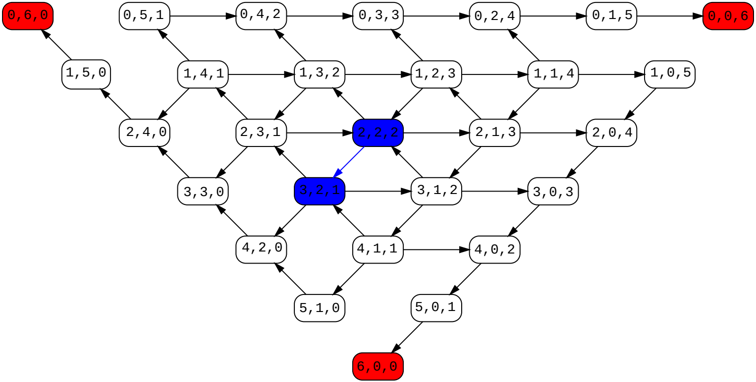
Stochastic Petri net (SPN)

3way-oscillator model [Ballarini2009]

$$R_1 : A + B \xrightarrow{r_B \cdot A \cdot B} B + B$$
$$R_2 : B + C \xrightarrow{r_C \cdot B \cdot C} C + C$$
$$R_3 : C + A \xrightarrow{r_A \cdot C \cdot A} A + A$$



Continuous Time Markov Chain (CTMC)



marking-dependent state transition rates: e.g. $(2,2,2) \xrightarrow{r_A \cdot 2 \cdot 2} (3,2,1)$

CTMC analysis

- Transient analysis
  - computation of $\underline{\pi}(\alpha, \tau)$
  - uniformisation method
- Steady state analysis
  - computation of $\underline{\pi}(\alpha)$
  - iterative methods (Jacobi, Gauss-Seidel)
- Model checking
  - Continuous Stochastic Logic (CSL)
  - can be reduced to transient and steady state analysis

CTMC analysis

- Transient analysis
  - computation of $\underline{\pi}(\alpha, \tau)$
  - uniformisation method
- Steady state analysis
  - computation of $\underline{\pi}(\alpha)$
  - iterative methods (Jacobi, Gauss-Seidel)
- Model checking
  - Continuous Stochastic Logic (CSL)
  - can be reduced to transient and steady state analysis

In any case, basic operation:
multiplication of a matrix and a vector over real values.

CTMC analysis

- Problem
  - possibly huge matrix and vector
  - explicit (sparse) matrix representation techniques fail
- Solutions
  - Kroneker algebra
  - Multi Terminal Binary Decision Diagrams (MTBDD)
  - matrix-free
  - sophisticated swapping (disc-based)

CTMC analysis

- Kronecker based techniques (SMART)
  - require regular models (partitioning of the place set)
- MTBDD based techniques (PRISM)
  - require prior knowledge of the boundedness degree
  - become inefficient when
    - high amount of variables $\Rightarrow$ must be doubled to encode a matrix
    - boundedness degree increases (binary encoding) $\Rightarrow$ #(BDD variables) increases
    - #(different matrix entries) increases $\Rightarrow$ #(terminal nodes) increases, destroys BDD compression effect

CTMC analysis

Biochemical network models

- often non-regular structure
- accurate analysis requires an high amount of tokens (molecules, concentration levels)
- many different real-valued matrix entries because of the marking-dependent rate functions

CTMC analysis

Biochemical network models

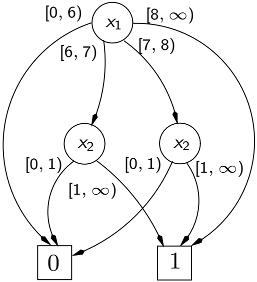
- often non-regular structure
- accurate analysis requires an high amount of tokens (molecules, concentration levels)
- many different real-valued matrix entries because of the marking-dependent rate functions

Our contribution:
a matrix-free technique based on Interval Decision Diagrams (IDD)

Interval Decision Diagrams

Characteristics

- #(variables) = #(places) = IDD height
- always two terminal nodes
- arcs are label with intervals over $\mathbb{N}$
- intervals of the outgoing arcs of a nonterminal node are a partitioning of $\mathbb{N}$

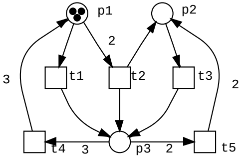


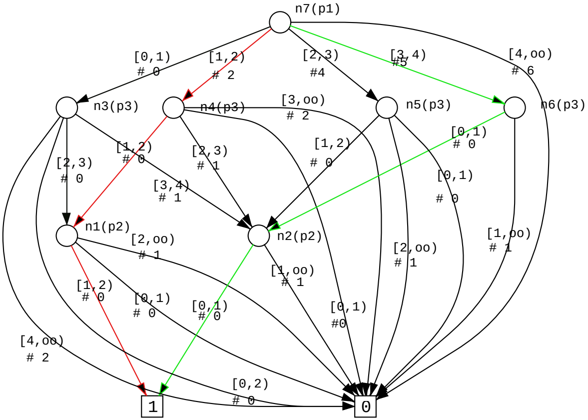
Matrix-free IDD-based approach

Basic ideas

- no storage of the rate matrix
- the SPN and its state space (symbolically encoded by an IDD) represent the CTMC
  CTMC = state space + net structure + rate functions
- for a state s all state transitions (s, s') and their rates are computed when needed
- defined by the firing of the transitions of the net
- forward (backward) firing means:
  extracting the rows (columns) of the CTMC

Matrix-free IDD-based approach





IDD-CSL

Basics

- prototype implemented in C++
- command line tool

Features implemented so far

- subset of CSL (time bounded operators)
- Jacobi-based steady state analysis
- ordinary Place/Transition nets

Variable order

- variable order affects the size of decision diagrams
- finding an optimal order is NP complete
- solution: net structure based heuristics;
  dependent places should be close together

$$W(p) := \frac{\sum_{t \in {}^\bullet p} \frac{|{}^\bullet t \cap Q|}{|{}^\bullet t|} + \sum_{t \in p^\bullet} \frac{|t^\bullet \cap Q|}{|t^\bullet|}}{|\ {}^\bullet p \cup p^\bullet\ |}$$

Q is the set of already used places

in each step select $p \in P \setminus Q$ such that $\forall p' \in P \setminus Q, p \neq p' : W(p) \geq W(p')$

Further techniques

- Caching (like PRISM)
  - no traversal of the whole IDD
  - introducing a cache layer
  - stopping traversal at the cache layer and processing the cached data (index pairs and rates)
  - problem: memory consumption for the cached data $\Rightarrow$ finding the best layer
- Parallelisation
  - parallelised multiplication
  - partitioning of the state space
  - applying the traversal to the subsets

Benchmarks

- comparison with the probabilistic model checker PRISM
- 4× 2.8GHz Intel XEON, 4 GB RAM, 64bit Linux
- *only runtime*
- maximal runtime 24 h
- three biochemical networks
- different levels by multipilying the initial marking with a constant

PRISM vs. IDD-CSL

IDD-CSL is inspired by PRISM's hybrid engine, there are several

Similarities

- based on Decision Diagrams
- multiplication by DD traversal,
  state indices are computed using offsets during the traversal
- explicit storage of probability vectors and the diagonal entries
  $\Rightarrow$ **for IDD-CSL the limiting factor**
- similar caching strategies

PRISM vs. IDD-CSL

Differences

	PRISM	IDD-CSL

PRISM vs. IDD-CSL

Differences

	PRISM	IDD-CSL
state space	BDD	IDD

PRISM vs. IDD-CSL

Differences

	PRISM	IDD-CSL
state space	BDD	IDD
rate matrix	MTBDD	IDD + SPN

PRISM vs. IDD-CSL

Differences

	PRISM	IDD-CSL
state space	BDD	IDD
rate matrix	MTBDD	IDD + SPN
prior knowledge of the boundedness degree	yes	no

PRISM vs. IDD-CSL

Differences

	PRISM	IDD-CSL
state space	BDD	IDD
rate matrix	MTBDD	IDD + SPN
prior knowledge of the boundedness degree	yes	no
#(variables)	depends on the boundedness degree	#(places)

PRISM vs. IDD-CSL

Differences

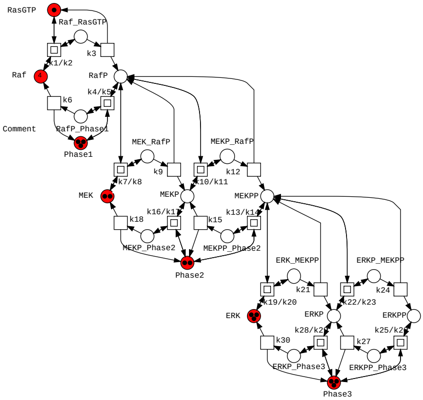
	PRISM	IDD-CSL
state space	BDD	IDD
rate matrix	MTBDD	IDD + SPN
prior knowledge of the boundedness degree	yes	no
#(variables)	depends on the boundedness degree	#(places)
variable order	plain	heuristics

PRISM vs. IDD-CSL

Differences

	PRISM	IDD-CSL
state space	BDD	IDD
rate matrix	MTBDD	IDD + SPN
prior knowledge of the boundedness degree	yes	no
#(variables)	depends on the boundedness degree	#(places)
variable order	plain	heuristics
rates	terminal nodes	on-the-fly computation

Mitogen Activated Protein Kinase [Levchenko2000]



CTMC size for different levels

levels	states	edges[a)]
4	24,065	206,007
8	6,110,643	78,948,888
12	315,647,600	4,958,809,056
16	6,920,337,880	122,381,517,819
20	88,125,763,956	1,689,018,298,500
24	769,371,342,640	15,635,976,824,982
28	5,084,605,436,988	108,065,356,604,208
32	27,124,071,792,125	597,236,499,605,178

[a)] *i.e., #(non-zero entries) in rate matrix;*

For level 520 the MAPK model exhibits 3.40e+30 states (IDD-CTL).

Influence of variable order

MTBDD size (PRISM) for different levels

levels	terminal nodes[a)]	original order time	original order nodes	good order time	good order nodes
4	30				
8	76				
12	140				
16	219				
20	320				
24	453				
28	697				
32	770				

[a)] *i.e., #(different entries) in the matrix;*

Influence of variable order

MTBDD size (PRISM) for different levels

levels	terminal nodes[a)]	original order time	original order nodes	good order time	good order nodes
4	30	2.47	123,730		
8	76	401.68	*3,881,914*		
12	140	-	-		
16	219	-	-		
20	320	-	-		
24	453	-	-		
28	697	-	-		
32	770	-	-		

[a)] *i.e.,#(different entries) in the matrix; – exceeds physical memory;*

Influence of variable order

MTBDD size (PRISM) for different levels

levels	terminal nodes[a)]	original order time	original order nodes	good order time	good order nodes
4	30	2.47	123,730	0.12	8,672
8	76	401.68	*3,881,914*	1.56	60,452
12	140	-	-	22.99	199,496
16	219	-	-	71.25	542,339
20	320	-	-	296.87	953,146
24	453	-	-	635.92	2,029,598
28	697	-	-	928.45	*3,771,617*
32	770	-	-	1847.597	6,015,521

[a)] *i.e., #(different entries) in the matrix; – exceeds physical memory;*

$\mathcal{P}_{>0.0}[\mathbf{F}_{[0,1]} RafP = 2]$, for 8 levels

PRISM[a]		
cl[b]	total[c]	iter[d]
65	208.35	**140.82**
60	222.67	169.76
55	201.23	154.94
50	200.13	158.99
45	**195.83**	159.90
40	198.43	163.03
35	214.64	179.90
30	226.16	191.22
25	218.51	184.06
20	230.66	195.92
1	2318.86	2275.71

IDD-CSL				
cl	1 thread		2 threads	
	total	iter	total	iter
3	440.23	170.06	432.77	157.08
5	158.65	110.02	158.71	99.75
7	93.84	79.48	72.71	55.01
9	84.62	75.61	62.57	50.55
10	**84.49**	**75.04**	**60.81**	**49.17**
11	90.65	81.73	64.08	52.18
13	100.40	91.39	67.47	55.97
15	127.72	118.09	81.40	69.71
17	253.60	243.05	147.85	134.09
19	692.84	676.05	387.62	368.08
21	1808.66	1771.00	957.26	917.07

[a] *using a good variable order, determined by the network structure;*

[b] *cache layers;* [c] *includes time for state space construction, initialisation, computation and determining the satisfying states;* [d] *effective probability computation time;*

$\mathcal{P}_{>0.0}[\mathbf{F}_{[1,1]} RafP = 2]$, for 8 levels

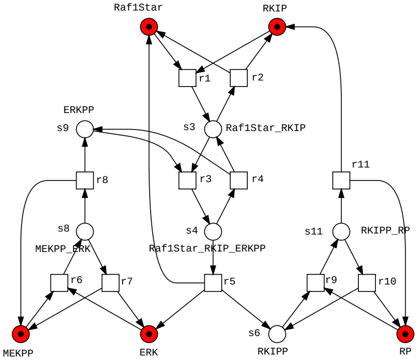
PRISM[a]		
cl[b]	total[c]	iter[d]
65	242.80	**163.05**
60	231.65	170.69
55	275.53	219.89
50	222.23	173.32
45	210.13	167.48
40	**209.91**	168.83
35	212.23	171.38
30	211.43	170.39
25	221.78	181.28
20	231.90	192.46
1	2745.47	2691.70

IDD-CSL				
cl	1 thread		2 threads	
	total	iter	total	iter
3	382.00	148.05	365.91	129.08
5	135.06	94.01	124.50	74.27
7	88.74	75.84	67.32	50.43
9	82.22	73.47	60.25	48.34
10	**81.58**	**72.82**	**58.55**	**47.21**
11	87.67	78.98	62.42	51.11
13	97.53	88.98	65.55	54.34
15	124.67	116.00	78.85	67.56
17	260.95	250.09	150.66	137.09
19	782.22	766.00	428.38	409.02
21	2128.20	2087.00	1150.41	1106.00

[a] *using a good variable order, determined by the network structure;*

[b] *cache layers;* [c] *includes time for state space construction, initialisation, computation and determining the satisfying states;* [d] *effective probability computation time;*

RKIP inhibited ERK pathway [Cho2003]



$\mathcal{P}_{>0.0}[\mathbf{F}_{[1,1]} Raf1Star = 1]$

level	CTMC size		PRISM[a)]		IDD-CSL	
	states	edges	hybrid	sparse	1 thread	2 threads
5	1,974	12,236				
10	47,047	372,372				
15	368,220	3,213,408				
20	1,696,618	15,609,594				
25	5,723,991	54,438,930				
30	15,721,464	152,964,146				

a) using a good variable order, determined by the network structure;

$\mathcal{P}_{>0.0}[\mathbf{F}_{[1,1]} Raf1Star = 1]$

level	CTMC size		PRISM[a)]		IDD-CSL	
	states	edges	hybrid	sparse	1 thread	2 threads
5	1,974	12,236	0.73	0.65		
10	47,047	372,372	5.52	3.97		
15	368,220	3,213,408	†	†		
20	1,696,618	15,609,594	†	†		
25	5,723,991	54,438,930	†	†		
30	15,721,464	152,964,146	†	†		

[a)] *using a good variable order, determined by the network structure;*
† *time for initialisation exceeds 24 hours;*

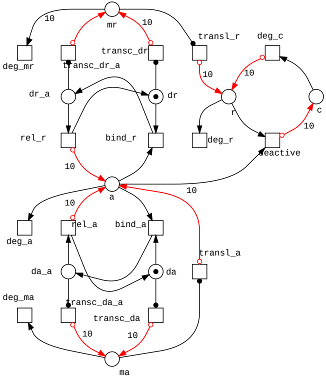
$\mathcal{P}_{>0.0}[\mathbf{F}_{[1,1]} \mathit{Raf1Star} = 1]$

level	CTMC size		PRISM[a]		IDD-CSL	
	states	edges	hybrid	sparse	1 thread	2 threads
5	1,974	12,236	0.73	0.65	0.20	0.17
10	47,047	372,372	5.52	3.97	1.27	1.05
15	368,220	3,213,408	†	†	20.73	16.67
20	1,696,618	15,609,594	†	†	148.92	118.59
25	5,723,991	54,438,930	†	†	740.28	581.39
30	15,721,464	152,964,146	†	†	3,005.62	2,455.57

[a] *using a good variable order, determined by the network structure;*

† *time for initialisation exceeds 24 hours;*

Circadian clock [Barkai2000]



For the experiments an ordinary SPN was used, the capacities being modelled by complementary places.

$\mathcal{P}_{>0.0}[\mathbf{F}_{[1,1]} a = 1]$

level	CTMC size		PRISM[a]		IDD-CSL	
	states	edges	hybrid	sparse	1 thread	2 threads
5	31,104	290,160				
10	644,204	6,766,320				
15	4,194,304	45,972,480				
20	16,336,404	183,032,640				
25	47,525,504	539,650,800				
30	114,516,604	1,312,110,960				

[a] *original model from the PRISM case study suite*

$\mathcal{P}_{>0.0}[\mathbf{F}_{[1,1]} a = 1]$

level	CTMC size		PRISM[a)]		IDD-CSL	
	states	edges	hybrid	sparse	1 thread	2 threads
5	31,104	290,160	3.90	2.36		
10	644,204	6,766,320	122.55	64.94		
15	4,194,304	45,972,480	1,090.44	570.43		
20	16,336,404	183,032,640	5,569.65	2,835.89		
25	47,525,504	539,650,800	†	-		
30	114,516,604	1,312,110,960	†	-		

[a)] *original model from the PRISM case study suite*

† *time for initialisation exceeds 24 hours; — exceeds the physical memory;*

$\mathcal{P}_{>0.0}[\mathbf{F}_{[1,1]} a = 1]$

level	CTMC size		PRISM[a]		IDD-CSL	
	states	edges	hybrid	sparse	1 thread	2 threads
5	31,104	290,160	3.90	2.36	1.76	1.16
10	644,204	6,766,320	122.55	64.94	44.65	26.10
15	4,194,304	45,972,480	1,090.44	570.43	466.47	312.83
20	16,336,404	183,032,640	5,569.65	2,835.89	2,471.70	1,684.96
25	47,525,504	539,650,800	†	-	8,595.37	6,027.33
30	114,516,604	1,312,110,960	†	-	26,085.95	17,314.66

[a] *original model from the PRISM case study suite*

† *time for initialisation exceeds 24 hours;* — *exceeds the physical memory;*

Summary

- new symbolic matrix-free multiplication technique for the CTMC of an SPN and a probability vector
- very efficient prototype implementation of a CSL model checker
- efficiency is achieved by
  - IDD-based encoding of the state space
  - considering the Petri net structure
    $\Rightarrow$ variable order computation
    $\Rightarrow$ transition firing to extract matrix entries
  - parallelisation
- our variable order helps tuning PRISM

Outlook

- non-ordinary stochastic Petri nets with extended arcs
- steady states analysis based on Gauss-Seidel
- increasing performance
  - caching
  - parallelisation
  - controlled swapping
- porting to a cluster environment with distributed memory

Thanks for your attention.

Questions?
Demonstration?

http://www-dssz.informatik.tu-cottbus.de/software/mc.html