

Multi-rooted *Zero-suppressed MTBDDs* and the symbolic, quantitative verification of systems

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

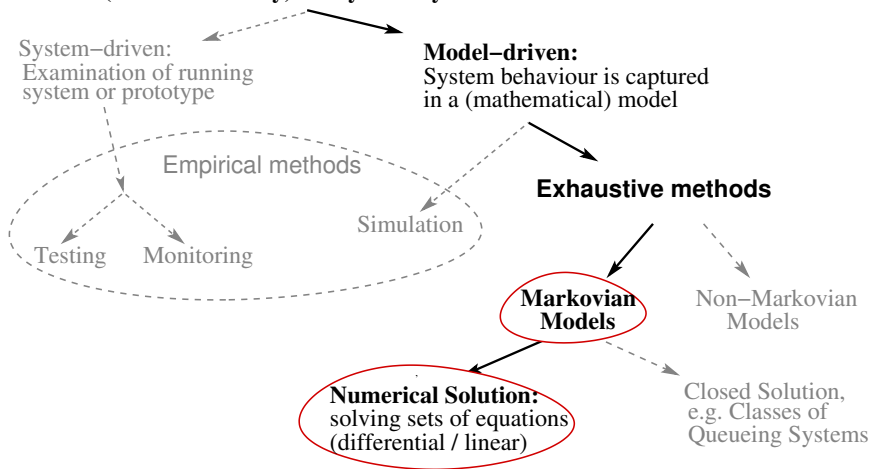
Computer Engineering and Networks Laboratory

Kai Lampka
www.tik.ee.ethz.ch

Workshop on Boolean Problems 2008

Context: Quantitative evaluation of systems

Performance and Dependability (= Performability) Analysis of Systems



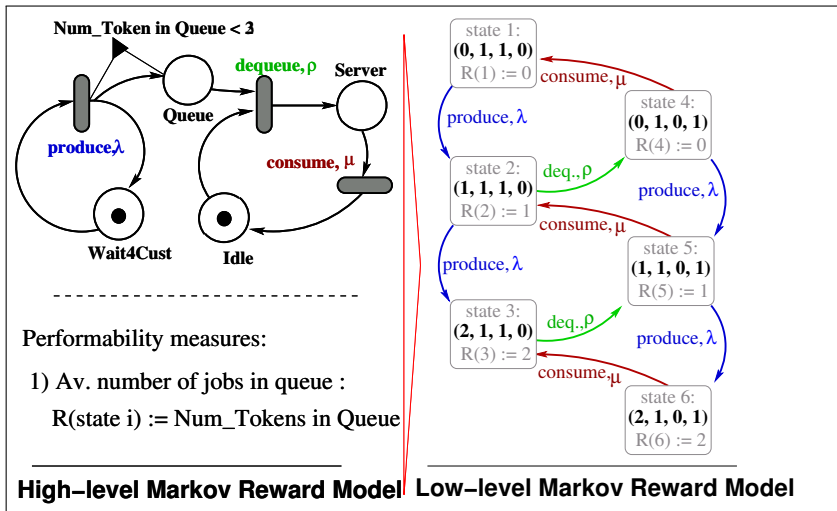
Complexity and size of systems enforces the use of

High-level (formal) model description techniques

Examples of high-level model description techniques

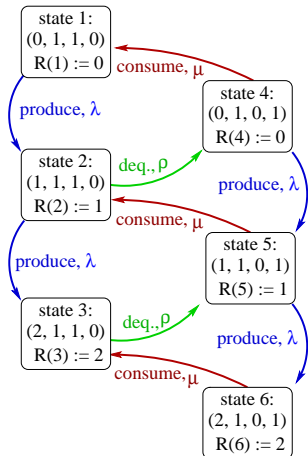
- Stochastic process algebras (*Tipp, Pepa, Empa, ...*)
- *Generalized stochastic Petri net* (GSPN)
- Stochastic activity networks (*SAN*)
- ...

High and Low-level Markov Reward Models (MRMs)



State Graph based Evaluation of High-level MRMs

(A) State graph (SG)



(B) Derive required information

1 Generator matrix Q

$$\begin{pmatrix} -\lambda & \lambda & 0 & 0 & 0 & 0 \\ 0 & -(\lambda + \rho) & \lambda & \rho & 0 & 0 \\ \vdots & \dots & & & \dots & \vdots \\ 0 & 0 & \rho & 0 & 0 & -\rho \end{pmatrix}$$

2 Vector for each reward:

- $\vec{R}_{Queue} := (0, 1, 2, 0, 1, 2)$

(C) Compute performability measures

- 1 State probabilities.
- 2 Moments of reward functions (over all states).

State Graph based Evaluation of High-level MRMs

- 1 Apply well-known iterative solution techniques to solve Chapman-Kolomogoroff system of

- differential equations for *transient state* probabilities:

$$\frac{\delta \vec{\pi}(t)}{\delta t} = \vec{\pi}(t)Q$$

- linear equations for *steady-state* probabilities ($0 = \vec{\pi}Q$).

⇒ **Vector of state probabilities $\vec{\pi}(t)$**

- 2 Evaluated reward functions:

- instant-of-time rate reward r :

$$\mathcal{R}_r(t) = \sum_{i \in \mathcal{S}} \mathcal{R}_r(i) \cdot \pi_i(t)$$

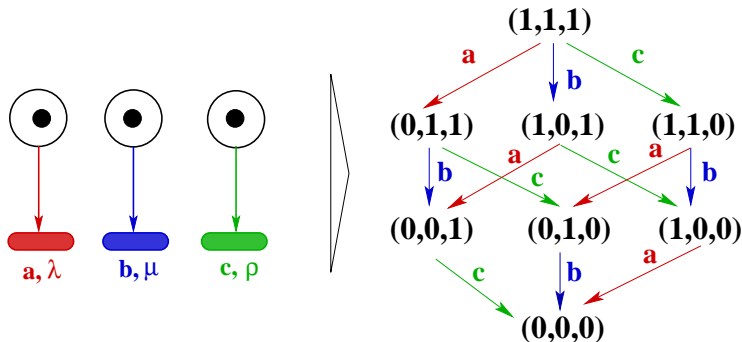
- interval-of-time impulse reward a :

$$\mathcal{I}^a(t, t + \Delta_t) = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} \bar{\pi}_i(t, t + \Delta_t) \cdot \Delta_t \cdot \mathcal{I}^a(i, j) \cdot \lambda_{ij}$$

State Space Explosion Problem

Independent activities + interleaving semantics

⇒ **Exponential growth of state graph**

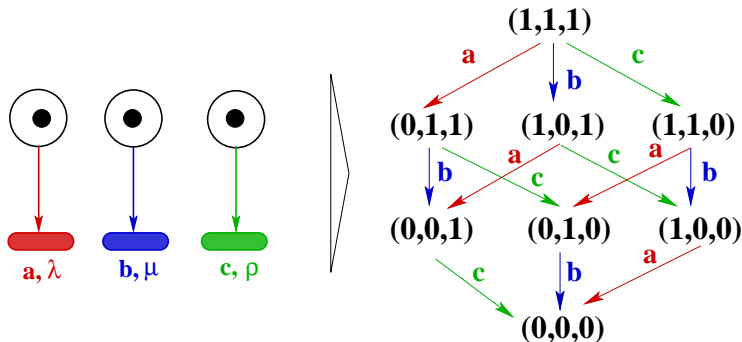


Traditional state graph (SG) exploration (bfs/dfs, hash-table) can only handle systems in the range of $\sim E6$ states (standard PCs)

State Space Explosion Problem

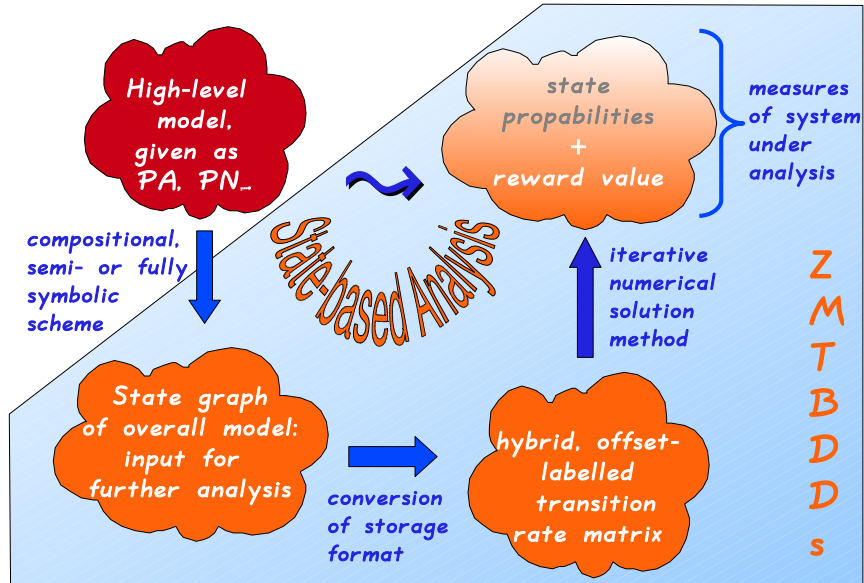
Independent activities + interleaving semantics

⇒ **Exponential growth of state graph**



Decision diagrams(DDs) ease this, system sizes in the range of $\sim E8$ states, numerical analysis is the bottleneck.

DD-based quantitative analysis of systems



Agenda

- 1 Context ✓
- 2 Multi-rooted Zero-suppressed Multi-terminal Decision Diagrams
- 3 Application: DD-based analysis of high-level, stochastic models
- 4 Empirical Evaluation
- 5 Conclusion

Part II

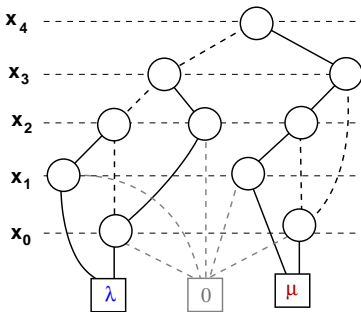
Shared DD-environments & Zero-suppressed Multi-terminal Decision Diagrams

Zero-suppressed MTBDDs (ZDDs) [LS06]

For deriving ZDDs from Minato's standard ZBDDs one allows function values from an arbitrary (finite) set, e.g. $\mathbb{D} \subset \mathbb{R}$.

⇒

ZDDs are representations of pseudo-Boolean functions
($f : \mathbb{B}^{|\mathcal{V}|} \rightarrow \mathbb{D}$)



Canonicity

In a **shared ZDD-environment**, ZDD-nodes are unique as long all functions are defined on the same set of input variables!

- If node $n \in \mathcal{K}_T$ then

$$f(n, \mathcal{F}) := \left(\prod_{x_i \in \mathcal{F}} (1 - x_i) \right) * \text{value}(n)$$

- if node $n \in \mathcal{K}_{NT}$ and $\text{var}(n) \notin \mathcal{F} \Rightarrow f(n, \mathcal{F}) = \perp$
- if node $n \in \mathcal{K}_{NT}$ and $\text{var}(n) \in \mathcal{F}$

$$\begin{aligned} f(n, \mathcal{F}) := & \prod_{x_i \in \mathcal{F}_n^{\text{before}}} (1 - x_i) \\ & * [\text{var}(n) * f(\text{then}(n), \mathcal{F}_n^{\text{after}}) + \\ & (1 - \text{var}(n)) * f(\text{else}(n), \mathcal{F}_n^{\text{after}})] \end{aligned}$$

Theorem (Existence)

For each pseudo-Boolean function f defined on \mathcal{F} and a strict total ordering π on \mathcal{V} there exists a ZDD-based representation (proof by induction over size of \mathcal{F}).

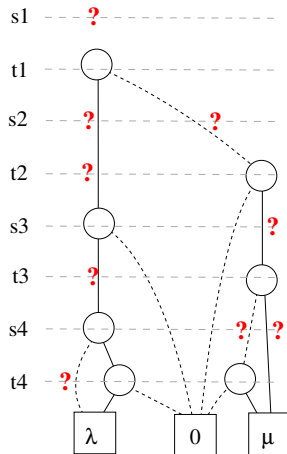
Theorem (Canonicity)

Let B and C be reduced ZDDs representing the functions f_B and f_C . Let $\mathcal{B} = \mathcal{C} = \mathcal{F}$ be their sets of input variables with the total strict ordering π defined on. If $f_B = f_C$, then ZDDs B and C are isomorphic (proof by induction over size of \mathcal{F}).

Problem

Weakly canonical representation

- Interpretation of graph depends on the associated set of input variables
- ZDD-nodes within shared DD-environments are not unique!
- Sharing of graphs is a major key to efficiency



How to handle multi-rooted ZDDs within shared DD-environments

- ZDD-object consists of root node and private set of input variables
- ZDD-manipulating algorithms are analogously implemented to Bryant's Apply-algorithm, but **they additionally iterate over set of input variables**:

Semantics of skipped variable x

Let n be the current node with $\text{var}(n) > x$:

- 1 $x \notin \mathcal{N} \Rightarrow$ *don't care* branching: $1\text{-succ} := n$ and $0\text{-succ} := n$
- 2 $x \in \mathcal{N} \Rightarrow$ *0-sup* branching: $1\text{-succ} := 0$ and $0\text{-succ} := n$

Why all this fuss?

Contemporary symbolic SG generation relies on

- compositional SG construction (DD-based cross-product computations) and
- DD-based reachability analysis

This requires DD-manipulating algorithms operating on DDs which not necessarily share the same set of input variables.

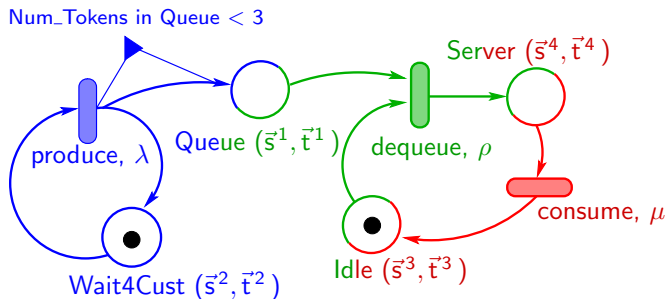
$$\text{newStates}(\vec{s}, \vec{t}) := \text{unex}(\vec{s}) \times \text{Trans}(\vec{s}, \vec{t})$$

$$\text{newStates}(\vec{t}) := \text{Abstract}(\text{newStates}, \vec{s}, \vee)$$

$$\text{newStates}(\vec{s}) := \text{newStates}(\vec{t}) \{ \vec{s} \leftarrow \vec{t} \}$$

Part III

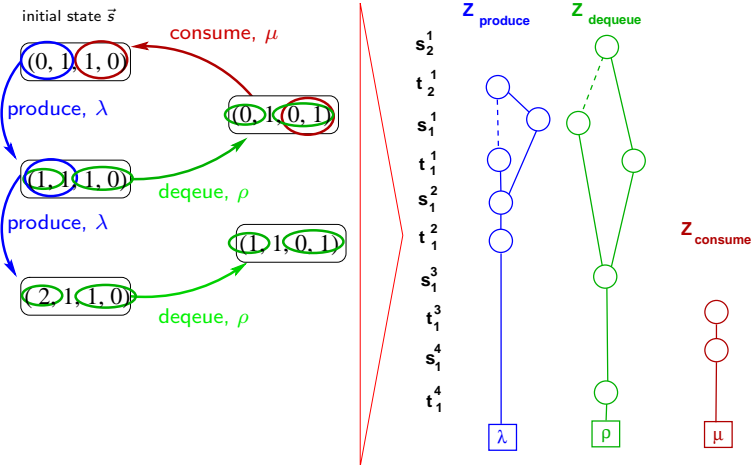
DD-based analysis of high-level models



Exploiting model's structures

$l \in \mathcal{A}ct$	dependent SVs (\mathcal{G}_l^D)
produce	{Queue, Wait4Cust}
dequeue	{Queue, Idle, Server}
consume	{Idle, Server}

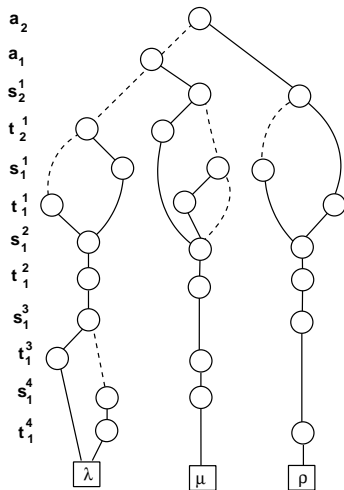
- \mathcal{G}_l^D -oriented encoding
- $\mathcal{S} \setminus \mathcal{G}_l^D$ are non-essential
- dep. relation on $\mathcal{A}ct$
 \Rightarrow partial SG expl.



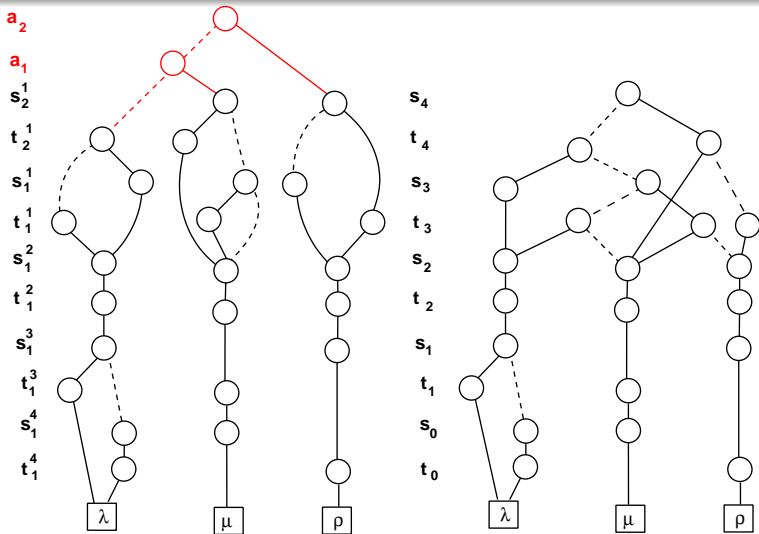
Selective breadth-first search explicit exploration and **activity-dependent encoding** of transitions, **until a fixed point is reached (local)**.

- 1 **Symbolic composition:** Compute potential SG from act.-local symbolic structures by cross-product computations and summation
- 2 Execute **symbolic reachability analysis** for eliminating unreachable transitions

⇒ **Symbolic representation of the model's underlying SG**



A symbolic scheme delivers the transition rate matrix Z_T



s-variables \mapsto row-indices; t-variables \mapsto column-indices

Computing measures of the model

Solve Chapman-Kolomogoroff system of equations

$$\boxed{\frac{\delta \vec{\pi}(t)}{\delta t} = \vec{\pi}(t)Q} \text{ or } \boxed{0 = \vec{\pi}Q}.$$

- Generator matrix Q is a ZDD obtained as follows:

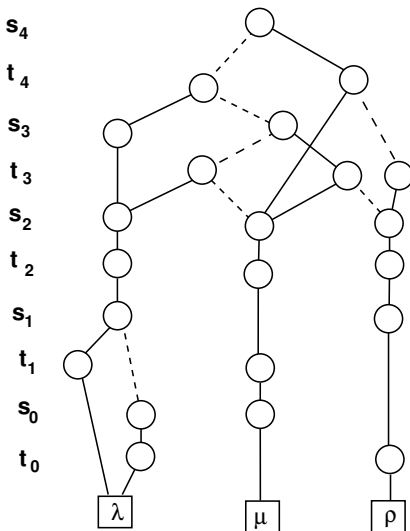
$$Q := \frac{1}{\max_{\forall i} \left(\sum_{i \neq j} Z_T(i, j) \right)} \cdot Z_T - \sum_{i \neq j} Z_T(i, j)$$

- for efficiency the DD representing Q is somehow customized!

Solver iterates over block-struct. hybrid ZDDs

Analogously to ADDs [Par02]:

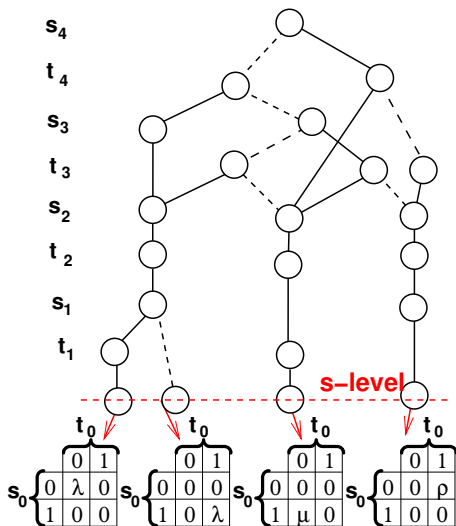
- Elements of diagonal are stored in a separate array ($q[i] := \sum_{i \neq j} Z_T(i, j)$)
- Remove upper b levels: Ordered access on the level of blocks (Pseudo Gauss-Seidel).
- Replace levels $> s$: All sub-matrix representations rooted at level s are converted into *sparse matrix format*.



Solver iterates over block-struct. hybrid ZDDs

Analogously to ADDs [Par02]:

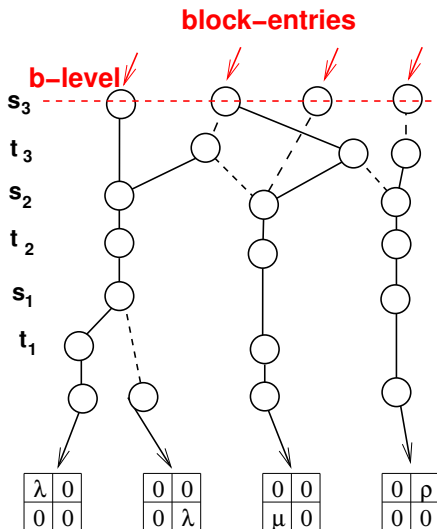
- Elements of diagonal are stored in a separate array ($q[i] := \sum_{i \neq j} Z_T(i, j)$)
- Remove upper b levels: Ordered access on the level of blocks (Pseudo Gauss-Seidel).
- Replace levels $> s$: All sub-matrix representations rooted at level s are converted into sparse matrix format.



Solver iterates over block-struct. hybrid ZDDs

Analogously to ADDs [Par02]:

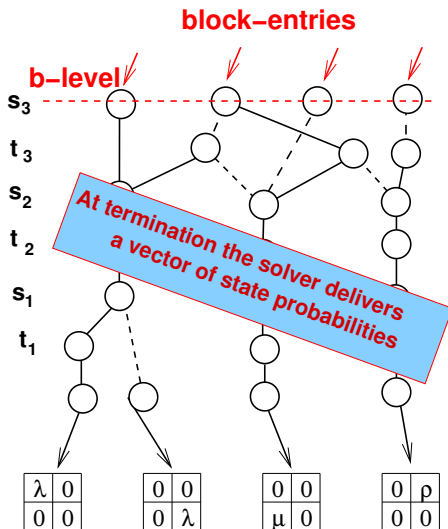
- Elements of diagonal are stored in a separate array ($q[i] := \sum_{i \neq j} Z_T(i, j)$)
- Remove upper b levels:
Ordered access on the level of blocks (Pseudo Gauss-Seidel).
- Replace levels $> s$:
All sub-matrix representations rooted at level s are converted into *sparse matrix format*.



Solver iterates over block-struct. hybrid ZDDs

Analogously to ADDs [Par02]:

- Elements of diagonal are stored in a separate array ($q[i] := \sum_{i \neq j} Z_T(i, j)$)
- Remove upper b levels: Ordered access on the level of blocks (Pseudo Gauss-Seidel).
- Replace levels $> s$: All sub-matrix representations rooted at level s are converted into *sparse matrix format*.



Part IV

Empirical evaluation

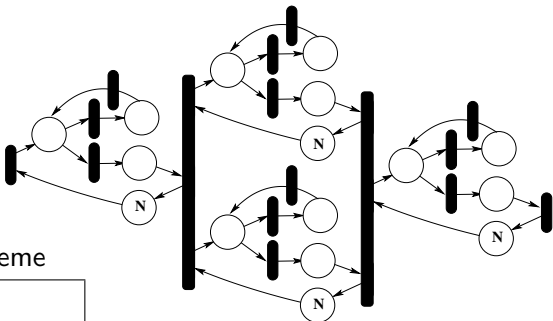
Benefits of employing ZDDs: Kanban System [CT96]

Model data

N	$\#states$	$\#trans$
10	$1.0 E9$	$12 E9$

(A) ZDD-based scheme

# nodes		
Z_R	Z_T	$peak$
497	8933	$0.66096 E6$



(B) Ratios to ADDs [ADD97]

r_R	r_T	r_{peak}
1.92	2.23	2.60

Steady state analysis for Kanban model (ZDD / ADD)

N	JAC with $b := 0.35$			
	# iter	t_{iter} in sec.		r_{iter}
		ADD	ZDD	
5	1,977	0.68	0.32	2.12
6	2,785	3.23	1.49	2.16
7	3,724	10.95	5.06	2.16

PGS $b := 0.35, s := 0.35$			
# iter	t_{iter} in sec.		r_{iter}
	ADD	ZDD	
1,542	0.83	0.29	2.90
2,176	3.88	1.37	2.84
2,913	15.08	5.15	2.93

Remark

- 1 ZDDs speed-up numerical analysis, (factor 2 - 3)
- 2 This can be further increased, when choosing larger values for s and b , which is possible due to lower memory requirements of ZDDs over ADDs

Steady state analysis for Kanban model (ZDD / ADD)

N	JAC with $b := 0.35$			
	# iter	t_{iter} in sec.		r_{iter}
		ADD	ZDD	
5	1,977	0.68	0.32	2.12
6	2,785	3.23	1.49	2.16
7	3,724	10.95	5.06	2.16

# iter	PGS $b := 0.35, s := 0.35$		
	t_{iter} in sec.		r_{iter}
	ADD	ZDD	
1,542	0.83	0.29	2.90
2,176	3.88	1.37	2.84
2,913	15.08	5.15	2.93

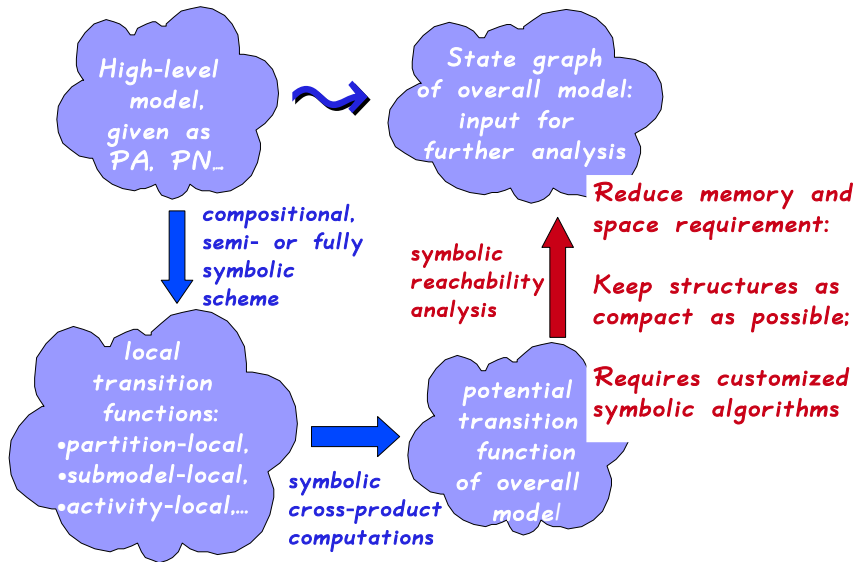
Remark

- 1 ZDDs speed-up numerical analysis, (factor 2 - 3)
- 2 This can be further increased, when choosing larger values for s and b , which is possible due to lower memory requirements of ZDDs over ADDs

Part V

Conclusion

DD-based analysis of systems




Lessons learned

- 1 Run-time-wise matrix/vector multiplication is the bottleneck of analysis: convert as much as possible into sparse matrix format for gaining speed.
- 2 Memory-wise the probability vector is the limiting factor, e.g. for $1.34 \cdot 10^8$ one already requires $\sim 1\text{GByte}$ RAM (pure symbolic solvers are not an option).

Future steps: Speeding-up numerical computations

- Adapt fast converging solution methods
- Reduce systems by exploiting symmetries
- Approximate solution methods (state aggregation)

 *Formal Methods in System Design: Special Issue on Multi-terminal Binary Decision Diagrams*, Volume 10, No. 2-3, April - May 1997.

 G. Ciardo and M. Tilgner.

On the use of Kronecker operators for the solution of generalized stochastic Petri nets.

Technical Report 96-35, Institute for Computer Applications in Science and Engineering, 1996.

 K. Lampka and M. Siegle.

Activity-Local State Graph Generation for High-Level Stochastic Models.

In *Measuring, Modelling, and Evaluation of Systems 2006*, pages 245–264, April 2006.

 D. Parker.

Implementation of Symbolic Model Checking for Probabilistic Systems.

PhD thesis, University of Birmingham, Birmingham (U.K.),
2002.