Parallel and Distributed Methods in Probabilistic Model Checker PRISM

Marta Kwiatkowska
School of Computer Science

UNIVERSITY OF
BIRMINGHAM

www.cs.bham.ac.uk/~mzk
www.cs.bham.ac.uk/~dxp/prism

INRIA, Grenoble, Nov 2005

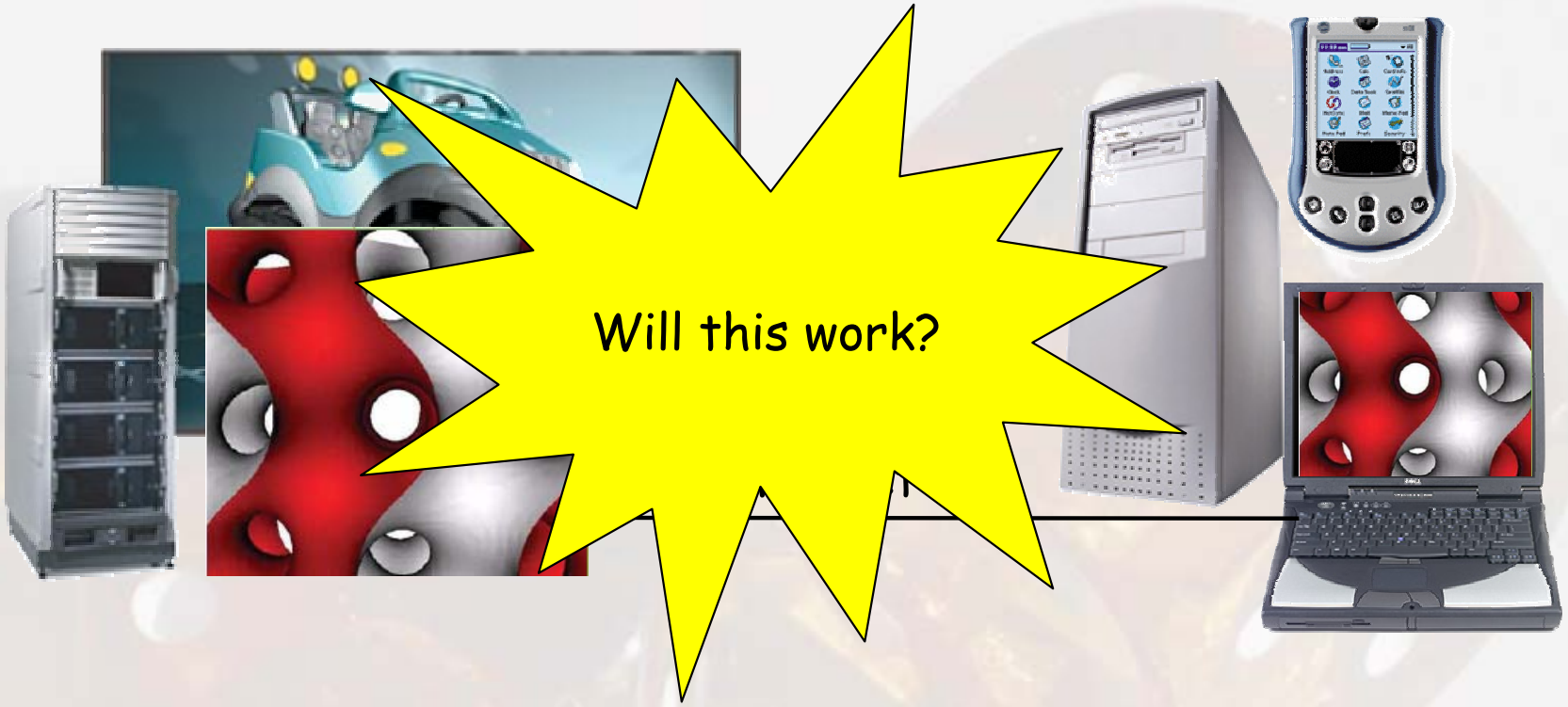
Overview

- Context
  - The PRISM model checker
- Probabilistic model checking
  - What does it involve?
  - Symbolic MTBDD-based techniques
- Parallel symbolic approaches
  - Parallel numerical solution
  - Grid-based techniques
- Simulation and sampling-based model checking
  - Distributed engine
- Conclusion and future work

With thanks to…

- Main collaborators on probabilistic model checking
  - Gethin Norman, Dave Parker, Jeremy Sproston, Christel Baier, Roberto Segala, Michael Huth, Luca de Alfaro, Joost-Pieter Katoen, Markus Siegle, Antonio Pacheco
- PRISM model checker implementation
  - Dave Parker, Andrew Hinton, Rashid Mehmood, Hakan Younes, Stephen Gilmore, Michael Goldsmith, Conrado Daws, Fuzhi Wang
- Parallelisation & Grid-enabling
  - Dave Parker, Yi Zhang, Rashid Mehmood
- And many more…

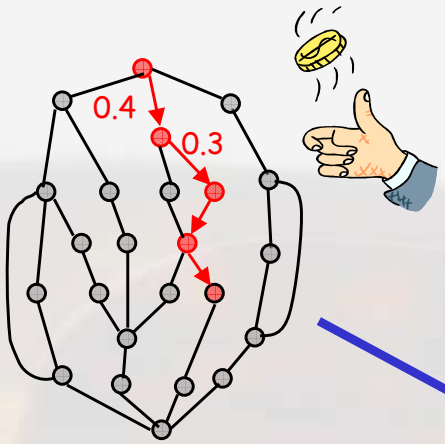
The e-Scientist of the future



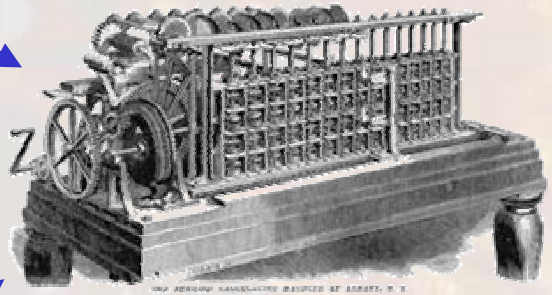
Remote access to high-performance computers, via Internet
Remote access to visualisation facilities, via Internet
Computational steering from anywhere, via PDA
Visualisation, on your laptop
Fast, online, accurate, …

Probabilistic model checking…

in a nutshell



Probabilistic model



Probabilistic Model Checker

send → $\mathbf{P}_{\geq 0.9}(\Diamond \text{deliver})$

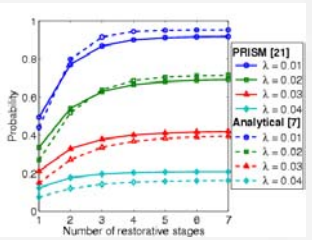
Probabilistic temporal logic specification

or

or

The probability

State 5: 0.6789
State 6: 0.9789
State 7: 1.0
…
State 12: 0
State 13: 0.1245



Probabilistic model checking inputs…

- Models
  - discrete time Markov chains (DTMCs)
  - continuous time Markov chains (CTMCs)
  - Markov decision processes (MDPs)
  - (currently indirectly) probabilistic timed automata (PTAs)
- (Yes/No) temporal logic specification languages
  - Probabilistic temporal logic PCTL (for DTMCs/MDPs)
  - Continuous Stochastic Logic CSL (for CTMCs)
  - Probabilistic timed computation tree logic PTCTL (for PTAs)
- Quantitative specification language variants
  - Probability values for logics PCTL/CSL/PTCTL (for all models)
  - Extension with expectation operator (for all)
  - Extension with costs/rewards (for all)

Probabilistic model checking involves…

- Construction of models:
  - DTMCs/CTMCs, MDPs, and PTAs (with digital clocks)
- Implementation of probabilistic model checking algorithms
  - graph-theoretical algorithms, combined with
    - (probabilistic) reachability
    - qualitative model checking (for 0/1 probability)
  - numerical computation – iterative methods
    - quantitative model checking (plot probability values, expectations, rewards, steady-state, etc, for a range of parameters)
    - exhaustive
  - sampling-based – simulation
    - quantitative model checking as above, based on many simulation runs
    - approximate

The PRISM probabilistic model checker

- History
  - Implemented at the University of Birmingham
  - First public release September 2001, ~7 years development
  - Open source, GPL licence, available freely for research an teaching
  - >2600 downloads, many users worldwide, >100 papers using PRISM, connection to several tools, >40 case studies and 6 flaws, …
- Approach
  - Based on symbolic, BDD-based techniques
  - Multi-Terminal BDDs, first algorithm [ICALP'97]
- www.cs.bham.ac.uk/~dxp/prism/
  - For software, publications, case studies, taught courses using PRISM, etc

Overview of PRISM

- Functionality
  - Implements temporal logic probabilistic model checking
  - Construction of models: discrete and continuous Markov chains (DTMCs/CTMCs), and Markov decision processes (MDPs)
  - High-level model description language, state-based, also PEPA
  - Probabilistic temporal logics: PCTL and CSL
  - Extension with costs/rewards, expectation operator

- Underlying computation combines graph-theoretical algorithms
  - Reachability, qualitative model checking, BDD-based

  with numerical computation – iterative methods
  - Linear equation system solution - Jacobi, Gauss-Seidel, ...
  - Uniformisation (CTMCs)
  - Dynamic programming (MDPs)
  - Explicit and symbolic (MTBDDs, etc.)

PRISM real-world case studies

- CTMCs
  - Dynamic power management [HLDVT'02, FAC 2005]
  - Dependability of embedded controller [INCOM'04]
  - RKIP-inhibited ERK pathway (by Calder et al) [CMSB'01]
  - thinkteam (by ter Beek, Massink & Latella) [DSVIS'05]
- MDPs/DTMCs
  - Bluetooth device discovery [ISOLA'04]
  - Crowds anonymity protocol (by Shmatikov) [CSFW'02, JSC 2003]
  - Randomised consensus [CAV'01,FORTE'02]
  - Contract signing protocols (by Norman & Shmatikov) [FASEC'02]
  - Reliability of NAND multiplexing (with Shukla) [VLSI'04,TCAD 2005]
- PTAs
  - IPv4 ZeroConf dynamic configuration [FORMATS'03]
  - Root contention in IEEE 1394 FireWire [FAC 2003, STTT 2004]
  - IEEE 802.11 (WiFi) Wireless LAN MAC protocol [PROBMIV'02]

PRISM property specifications

- PCTL/CSL (true/false) formula examples:
  - P<0.001 [true U≤100 error]
    "the probability of the system reaching an error state within 100 time units is less than 0.001"

- Can also write query formulae:
  - P=? [true U≤10 terminate]
    "what is the probability that the algorithm terminates successfully within 10 time units?"

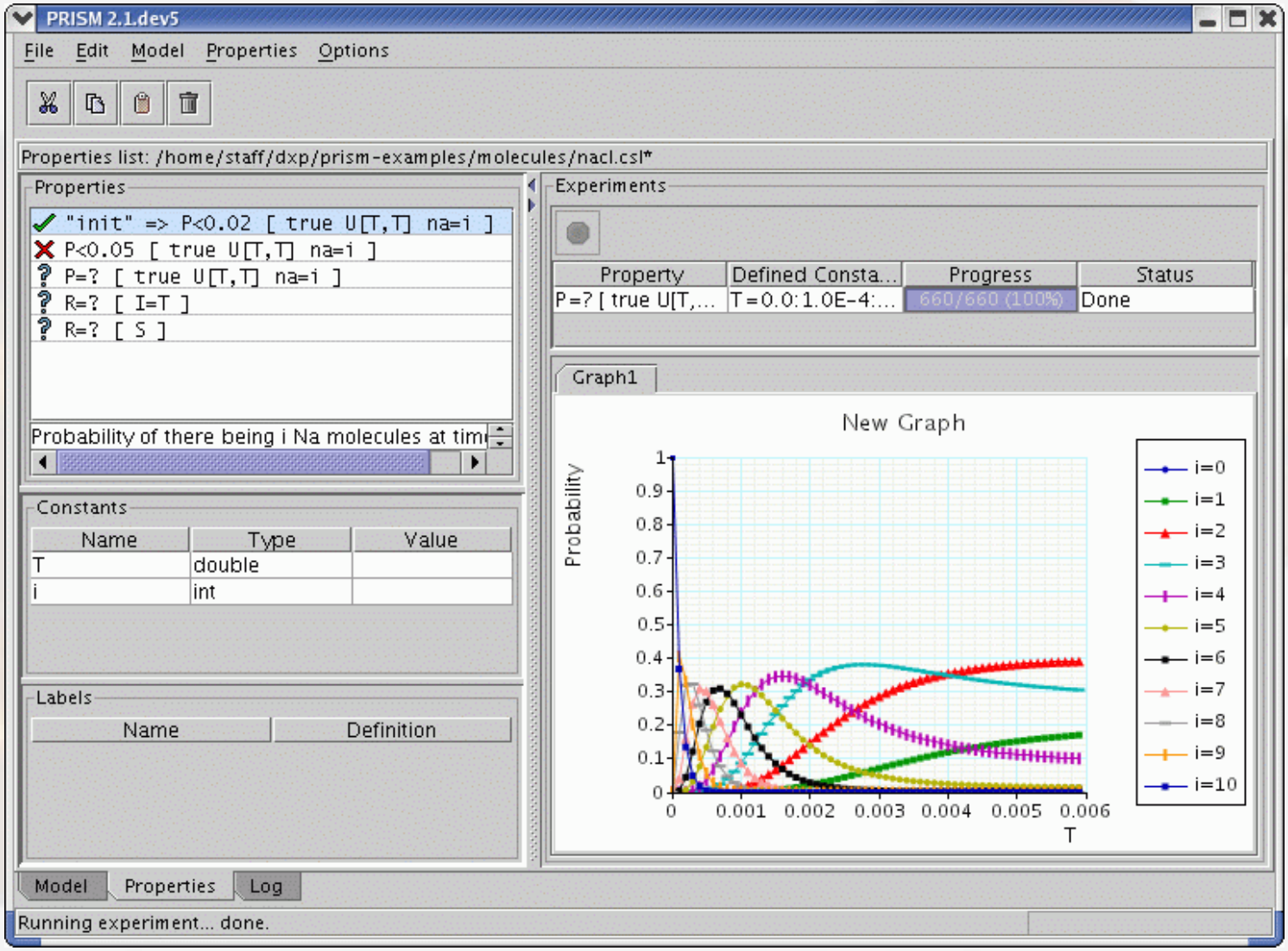
- Instantaneous rewards, state-based, e.g. "queue size":
  - R=? [I=T], expected reward at time instant T?

- Cumulative rewards, state/transition, e.g. "power consumed":
  - R=? [C<=T], expected reward by time T?
  - R=? [S], expected long-run reward per unit time?

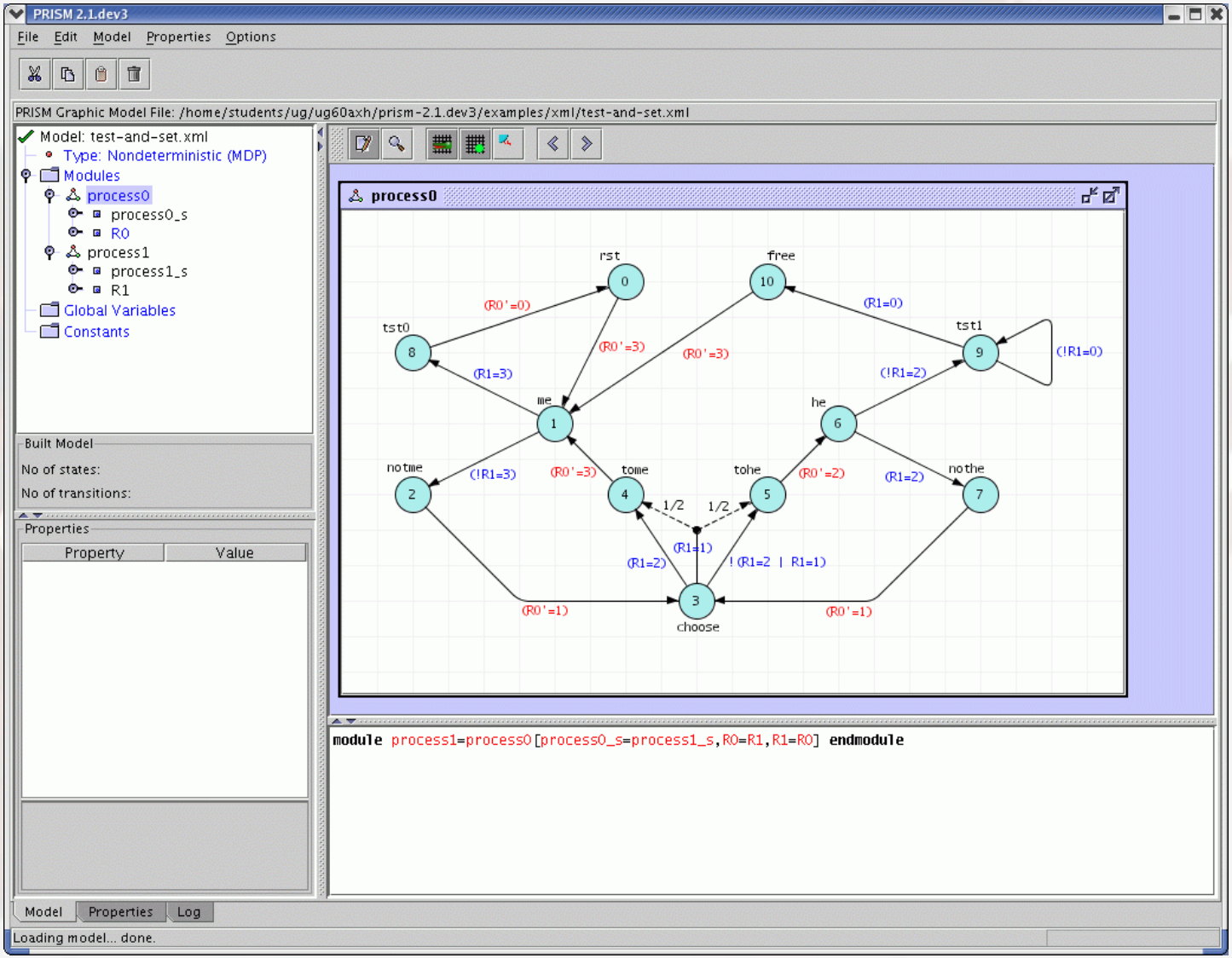
PRISM technicalities

- (New) Simulator and sampling-based model checking
  - allows to "excute" the model step-by-step or randomly
  - avoids state-space explosion, trading off accuracy
- GUI implementation
  - integrated editor for PRISM language
  - automatic graph plotting
- Support for "experiments"
  - e.g. P=? [true U<=T error] for N=1..5,T=1..100
  - repeats model checking for a range of model/formula parameters
  - plots on single graph

Screenshot: Graphs



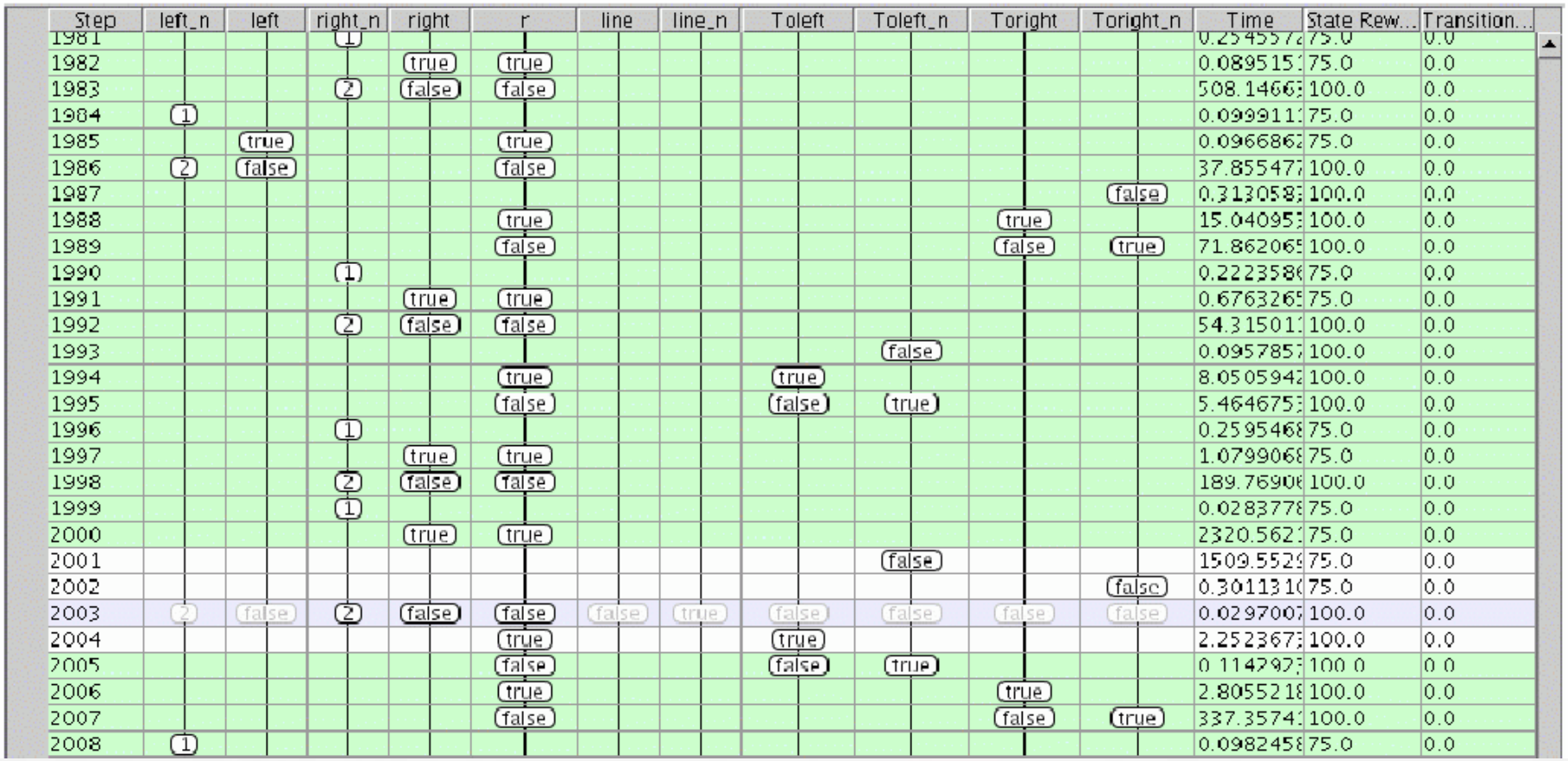
Screenshot: Graphical input language



Simulator output: Workstation cluster

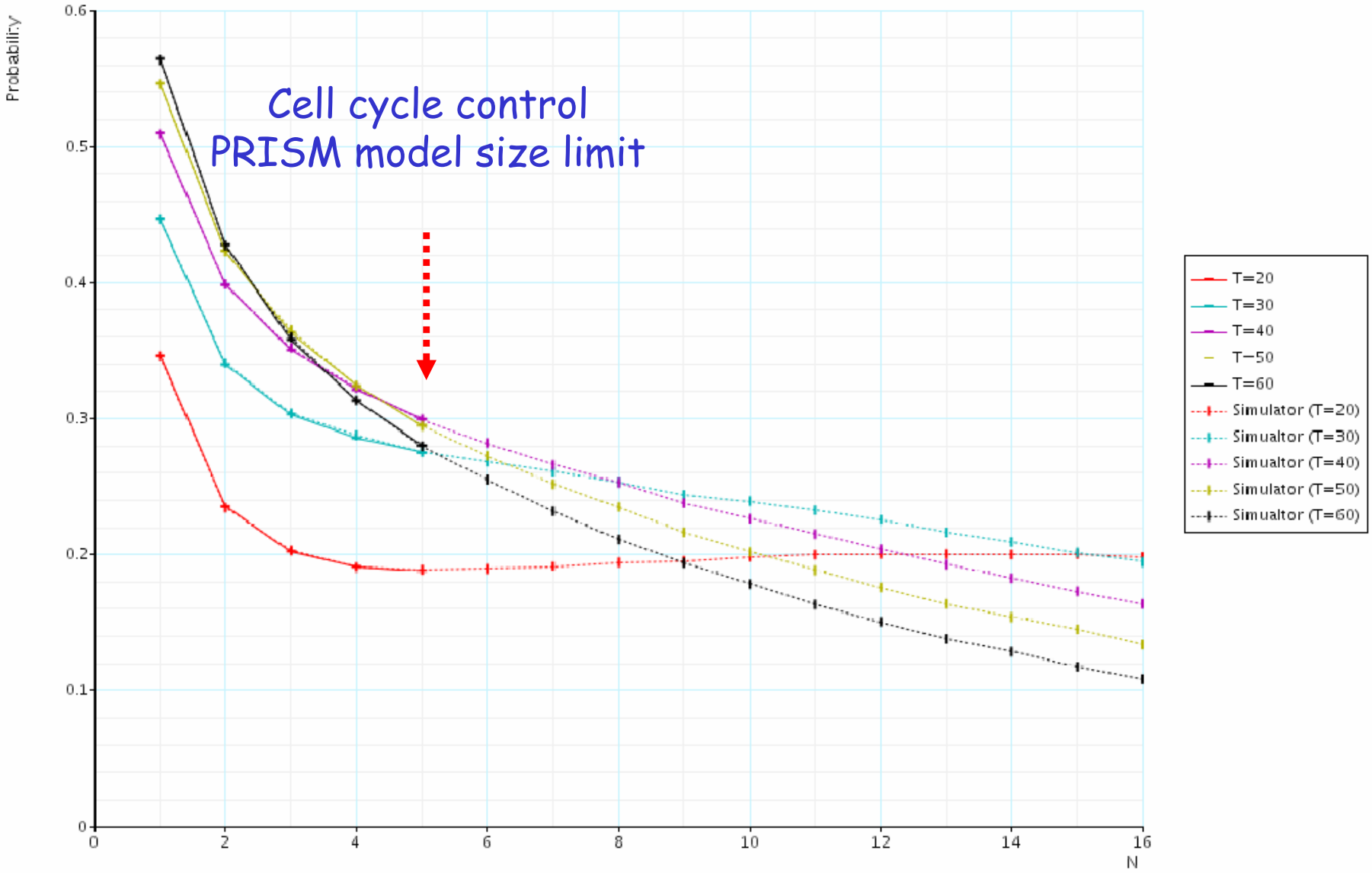
Formula label highlighted:

"premium" = (left_n>=left_mx&Toleft_n)|
(right_n>=right_mx&Toright_n)|
((left_n+right_n)>=left_mx&Toleft_n&line_n&Toright_n)



Step	left_n	left	right_n	right	r	line	line_n	Toleft	Toleft_n	Toright	Toright_n	Time	State Rew...	Transition...
1981			1									0.2545572	75.0	0.0
1982				true	true							0.0895151	75.0	0.0
1983			2	false	false							508.14663	100.0	0.0
1984	1											0.0999111	75.0	0.0
1985		true			true							0.0966862	75.0	0.0
1986	2	false			false							37.855477	100.0	0.0
1987											false	0.3130583	100.0	0.0
1988					true					true		15.040953	100.0	0.0
1989					false					false	true	71.862065	100.0	0.0
1990			1									0.2223586	75.0	0.0
1991				true	true							0.6763265	75.0	0.0
1992			2	false	false							54.315011	100.0	0.0
1993									false			0.0957857	100.0	0.0
1994					true			true				8.0505942	100.0	0.0
1995					false			false	true			5.4646753	100.0	0.0
1996			1									0.2595468	75.0	0.0
1997				true	true							1.0799068	75.0	0.0
1998			2	false	false							189.76906	100.0	0.0
1999			1									0.0283778	75.0	0.0
2000				true	true							2320.5621	75.0	0.0
2001									false			1509.5529	75.0	0.0
2002											false	0.3011310	75.0	0.0
2003	2	false	2	false	false	false	true	false	false	false	false	0.0297007	100.0	0.0
2004					true			true				2.2523673	100.0	0.0
2005					false			false	true			0.1147973	100.0	0.0
2006					true					true		2.8055218	100.0	0.0
2007					false					false	true	337.35741	100.0	0.0
2008	1											0.0982458	75.0	0.0

Scalability with sampling-based method



What we have learnt from practice

- Probabilistic model checking
  - Is capable of finding 'corner cases' and 'unusual trends'
  - Good for worst-case scenarios, for all initial states
  - Benefits from quantitative-style analysis for a range of parameters
  - Is limited by state space size
  - Useful for real-world protocol analysis, power management, performance, biological processes, …
- Simulation and sampling-based techniques
  - Limited by accuracy of the results, not state-space explosion
  - May need to rerun experiments for each possible start state, not always feasible
  - Statistical methods in conjunction with sampling help
  - Nested formulas may be difficult

New directions and challenges

- Often not practical to analyse the full system beforehand
  - Adaptive methods
    - Genetic algorithm-based methods [Jarvis et al, 2005]
  - Online methods
    - Continuous approximations using ODEs [Gilmore et al, 2005]
    - Applicable for sufficiently large numbers of entities
    - Work with PEPA, derivation of ODEs
- Scalability challenge
  - State-space explosion has not gone away…
  - Parallelisation
  - Distributed computation
  - Compositionality
  - Abstraction
  - Approximate methods

Why parallelise?

- Experience with PRISM indicates
  - Symbolic representation very compact, >10^{10} states for CTMCs
  - Extremely large models feasible depending on regularity
  - Numerical computation often slow
  - Sampling-based computation can be even slower
- Can parallelise the symbolic approach
  - Facilitates extraction of the dependency information
  - Compactness enables storage of the full matrix at each node
  - Focus on steady-state solution for CTMCs, can generalise
  - Use wavefront techniques to parallelise
- Easy to distribute sampling-based computation
  - Individual simulation runs are independent

Numerical Solution for CTMC/DTMCs

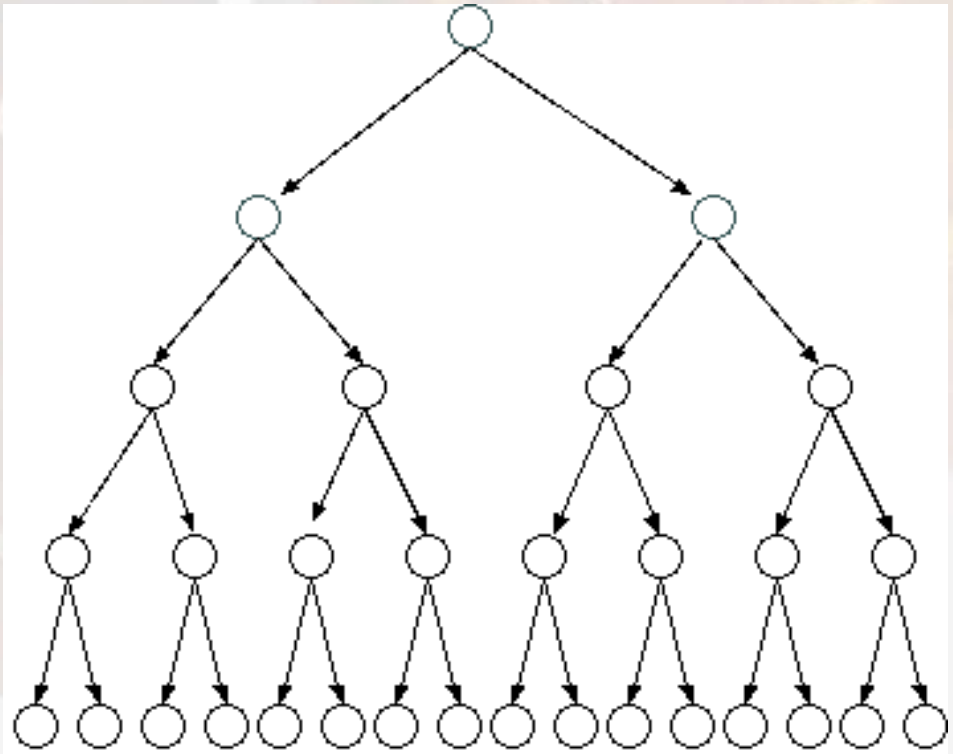
- Steady-state probability distribution can be obtained by solving linear equation system:
  - $\pi \mathbf{Q} = 0$ with constraint $\Sigma_i \pi_i = 1$
- Consider the more general problem of solving:
  - $\mathbf{Ax} = \mathbf{b}$,
  - where $\mathbf{A}$ is $n \times n$ matrix, $\mathbf{b}$ vector of length n
- Numerical solution techniques
  - Direct, not feasible for very large models
  - Iterative stationary (Jacobi, Gauss-Seidel), memory efficient
  - Projection methods (Krylov, CGS, …), fastest convergence, but require several vectors
- Transient probabilities similar
  - Computed via an iterative method (uniformisation)

Symbolic techniques for CTMCs

- Explicit matrix representation
  - Intractable for very large matrices
- Symbolic representations
  - e.g. Multi-Terminal Binary Decision Diagrams (MTBDDs), matrix diagrams and Kronecker representation
  - Exploit regularity to obtain compact matrix storage
  - Also faster model construction, reachability, etc
  - Sometimes also beneficial for vector storage
- This paper uses MTBDDs (Clarke et al) and derived structures
  - Underlying data structure of the PRISM model checker
  - Enhanced with caching-based techniques that substantially improve numerical efficiency

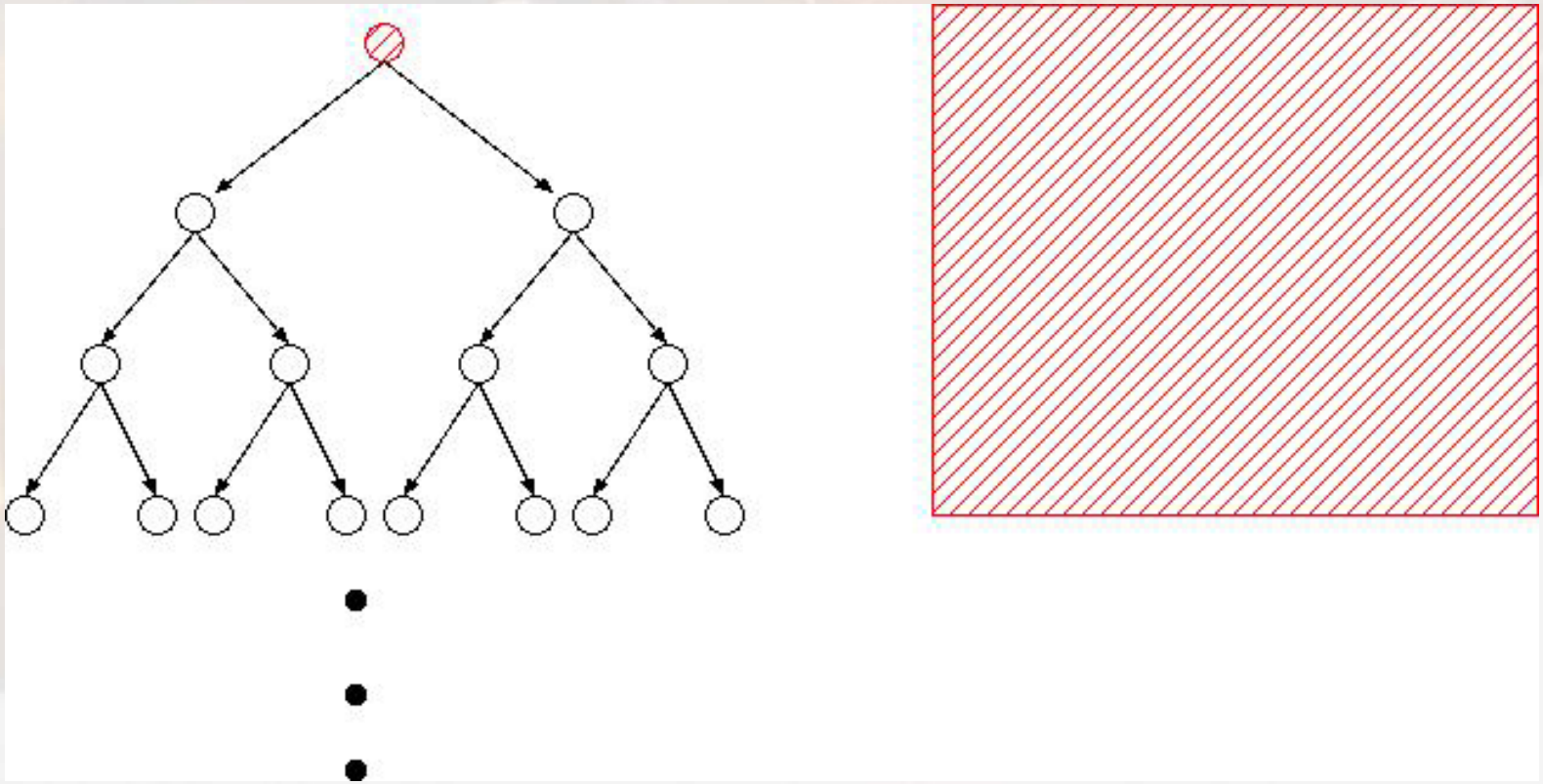
MTBDD data structures

- Recursive, based on Binary Decision Diagrams (BDDs)
  - Stored in reduced form (DAG), with isomorphic subtrees stored only once
  - Exploit regularity to obtain compact matrix storage



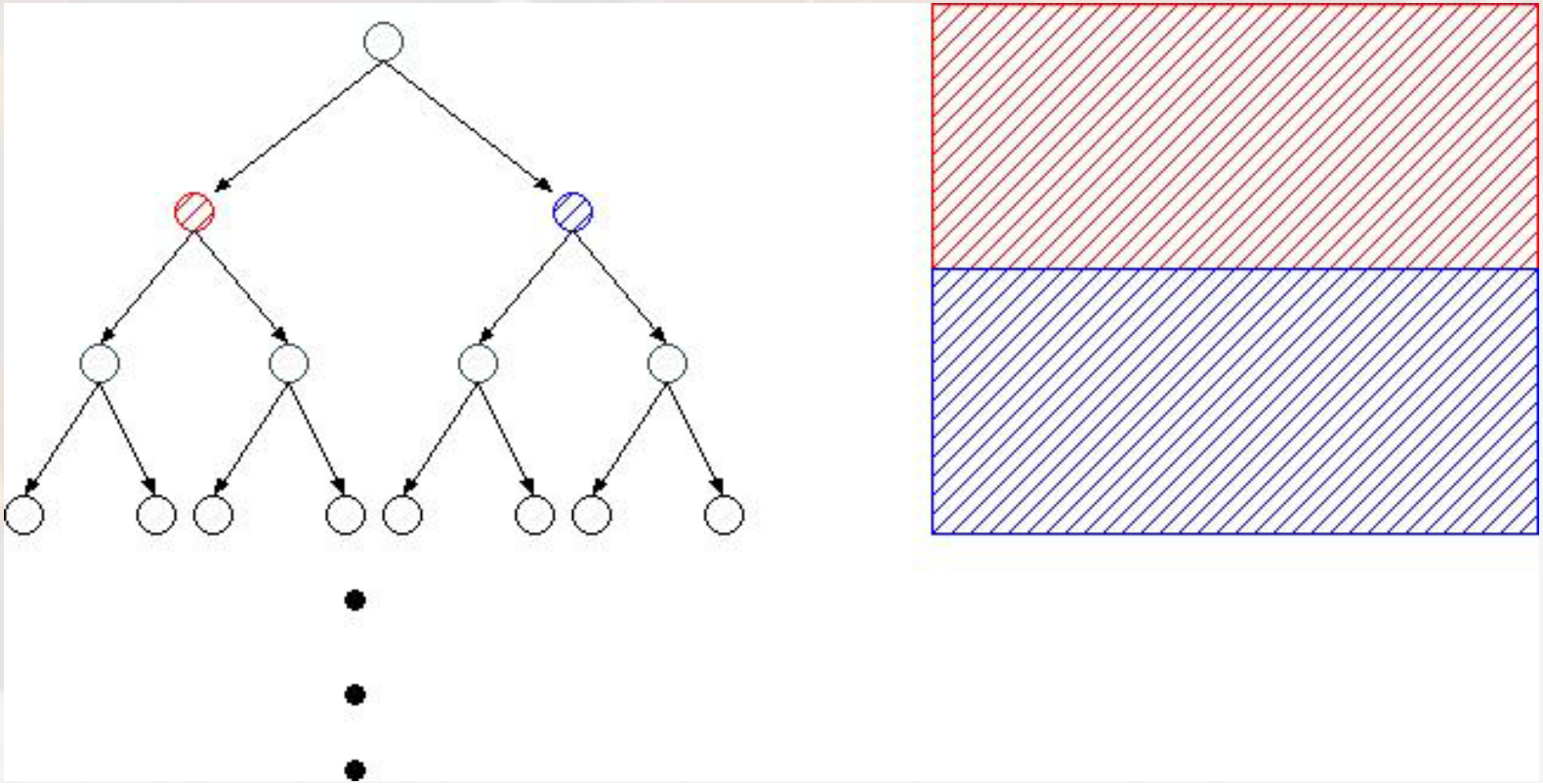
Matrices as MTBDDs

- Representation
  - Root represents the whole matrix
  - Leaves store matrix entries, reachable by following paths from the root node



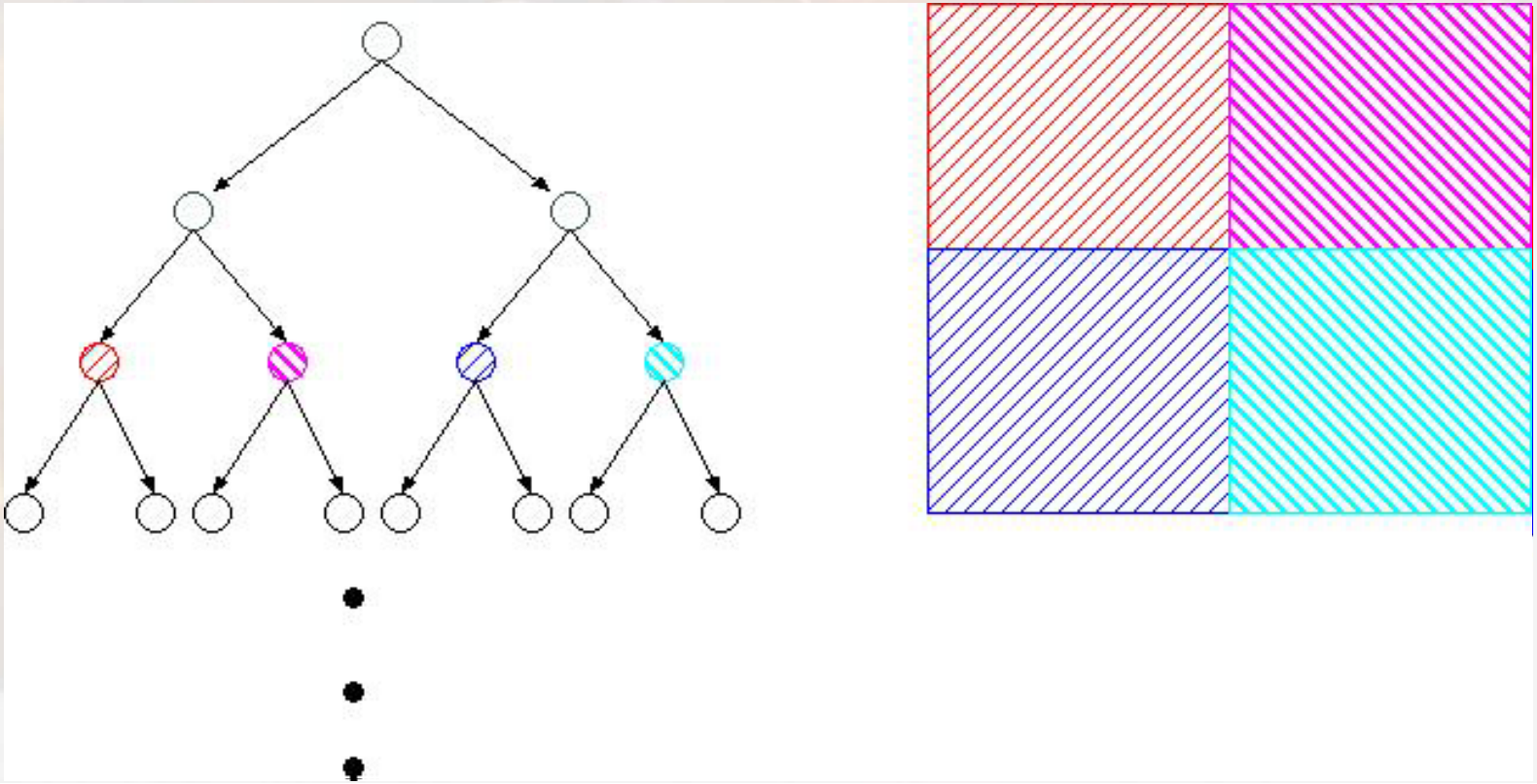
Matrices as MTBDDs

- Recursively descending through the tree
  - Divides the matrix into submatrices
  - One level, divide into two submatrices



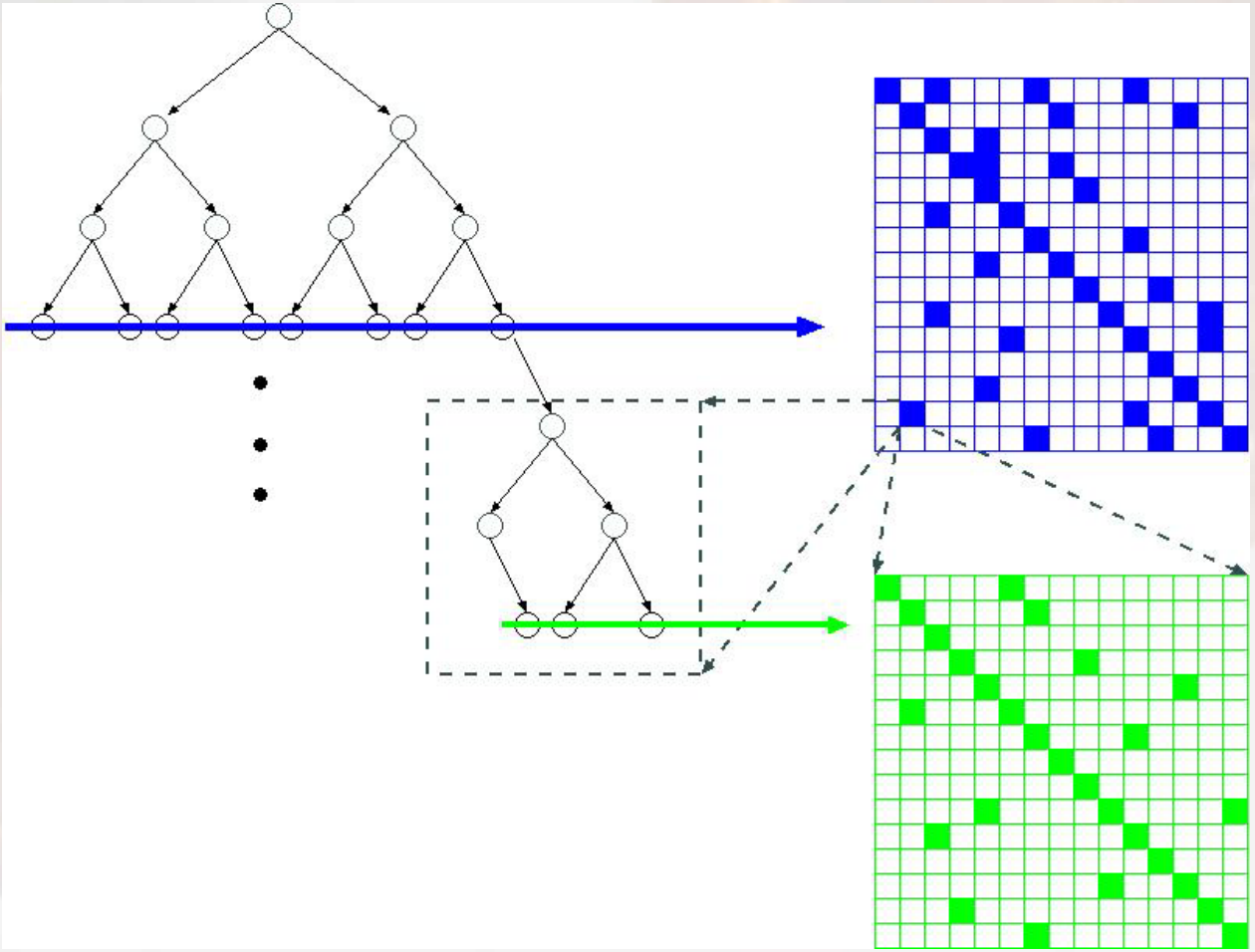
Matrices as MTBDDs

- Recursively descending through the tree
  - Provides a convenient block decomposition
  - Two levels, divide into four blocks



A two-layer structure from MTBDDs

- Can obtain block decomposition, store as two sparse matrices
  - Enables fast row-wise access to blocks and block entries



(Parker'02, Mehmood'05)

Parallel symbolic numerical engine

- MTBDDs
  - Provide a convenient block decomposition of the matrix (computation) into submatrices
- Parallel symbolic solution techniques for $Ax = b$
  - Store full matrix at each node (see also Kemper et al)
  - Solve in block form
  - Tasks, each determined by a matrix block
  - The execution order determined by computational dependency
- Techniques
  - Parallel Jacobi, CGS
  - Gauss-Seidel more difficult
    - Non-symbolic (Joubert et al) relies on permutation, not feasible in the symbolic context
    - Here, first symbolic parallelisation of Gauss-Seidel

Gauss-Seidel

- Computes one matrix row at a time
- Updates i^{th} element using most up-to-date values
- Computation for a single iteration, n×n matrix:
  1. for (0 · i · n-1)
  2. $x_i := (b_i - \sum_{0 \cdot j \cdot n-1, j \neq i} A_{ij} ¢ x_j) / A_{ii}$

- Can be reformulated in block form, N×N blocks, length M
  1. for (0 · p · N-1)
  2. $\mathbf{v} := \mathbf{b}_{(p)}$
  3. for each block $\mathbf{A}_{(pq)}$ with q≠p
  4. $\mathbf{v} := \mathbf{v} - \mathbf{A}_{(pq)} \mathbf{x}_{(q)}$
  5. for (0· i· M-1,i≠j)
  6. $\mathbf{x}_{(p)i} := (\mathbf{v}_i - \sum_{0 \cdot j \cdot M} \mathbf{A}_{(pp)ij} ¢ \mathbf{x}_{(p)j}) / \mathbf{A}_{(pp)ii}$

- Computes one matrix block at a time

Parallelising Gauss-Seidel

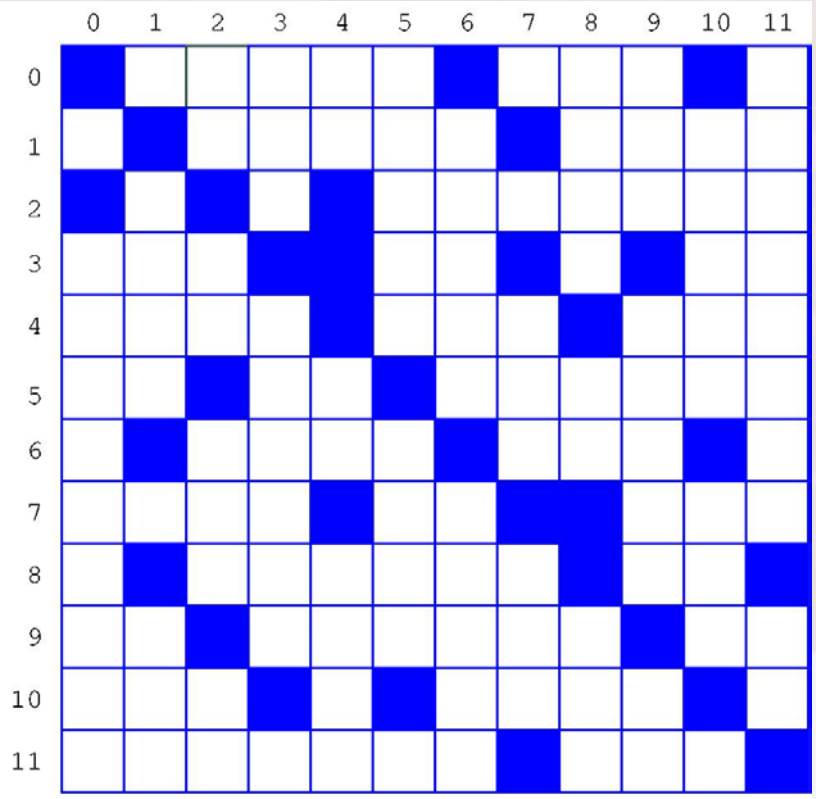
- Inherently sequential for dense matrices
  - Uses results from current and previous iterations
- Permutation has no effect on correctness of the result
  - Can be exploited to achieve parallelisation for certain sparse matrix problems, e.g. Koester, Ranka & Fox 1994
- The block formulation helps, although
  - Requires row-wise access to blocks and block entries
  - Need to respect computational dependencies, i.e. when computing vector block $\mathbf{x}_{(p)}$, use values from current iteration for blocks $q < p$, and from previous iteration for $q > p$
- Idea: propose to use wavefront techniques
  - Extract dependency information and form execution schedule

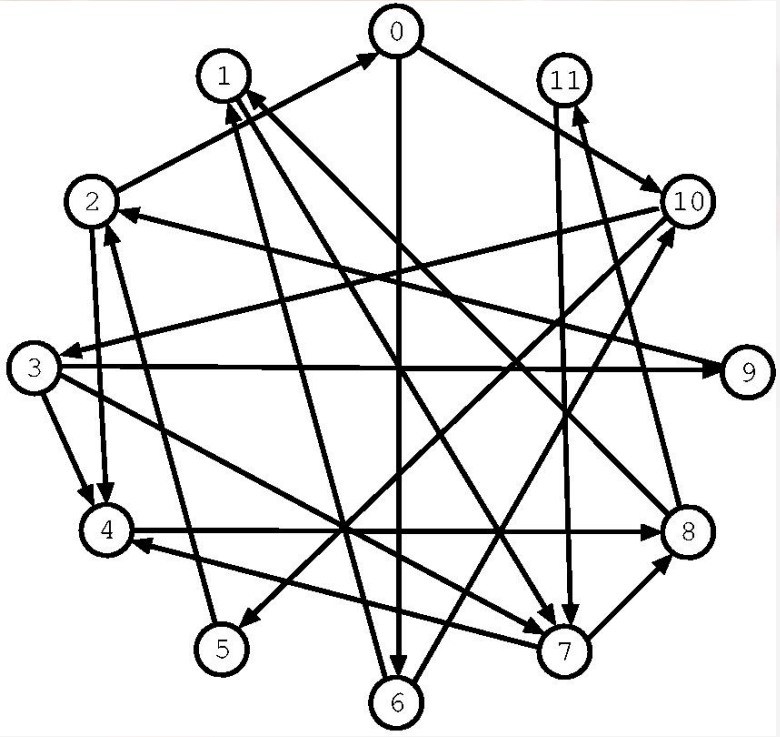
Wavefront techniques

- An approach to parallel programming, e.g. Joubert et al '98
  - Divide a computation into many tasks
  - Form a schedule for these tasks
- A schedule contains several wavefronts
  - Each wavefront comprises tasks that are algorithmically independent of each other
  - i.e. correctness is not affected by the order of execution
- The execution is carried out from one wavefront to another
  - Tasks assigned according to the dependency structure
  - Each wavefront contains tasks that can be executed in parallel

Dependency graph from MTBDD

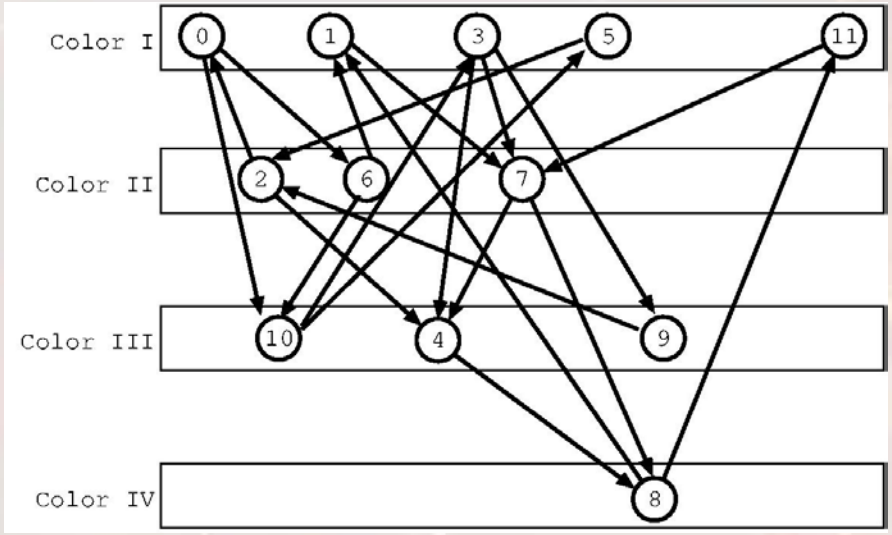
- By traversal of top levels of MTBDD, as for top layer





Generating a Wavefront Schedule

- By colouring the dependency graph



- Can generate a schedule to let the computation perform from one colour to another

Wavefront with MTBDDs

- Our parallelisation of Gauss-Seidel
  - Allows much larger CTMC models to be solved
  - Has good overall speedup
- Symbolic approach particularly well suited to Wavefront parallelisation of Gauss-Seidel
  - Easy to extract task dependency information
  - Reduced memory requirement and communication load
- Gauss-Seidel excellent candidate to solve very large linear equation systems
  - Small memory requirement (only requires one iteration vector, vs 2 for Jacobi and 6 for CGS)
  - Method generalises to other symbolic techniques and application domains

Implementation

- Implemented on a Ethernet and Myrinet-enabled PC cluster
  - Use MPI (the MPICH implementation)
  - Prototype extension for PRISM, uses PRISM numerical engines and CUDD package for MTBDDs (Somenzi)
  - 32 nodes available
- Evaluated on a range of benchmarks
  - Kanban, FMS and Polling system
- Optimisations for PC-cluster environments
  - Non-blocking inter-processor communication is used to interleave communication and computation
  - Load-balancing is implemented to distribute computation load evenly between processors and minimise communication load
  - Cache mechanism is used to reduce communication further

Experimental results: models

- Parameters and statistics of models
  - Include Kanban 9,10 and FMS 13, previously intractable
  - All compact, requiring less than 1GB

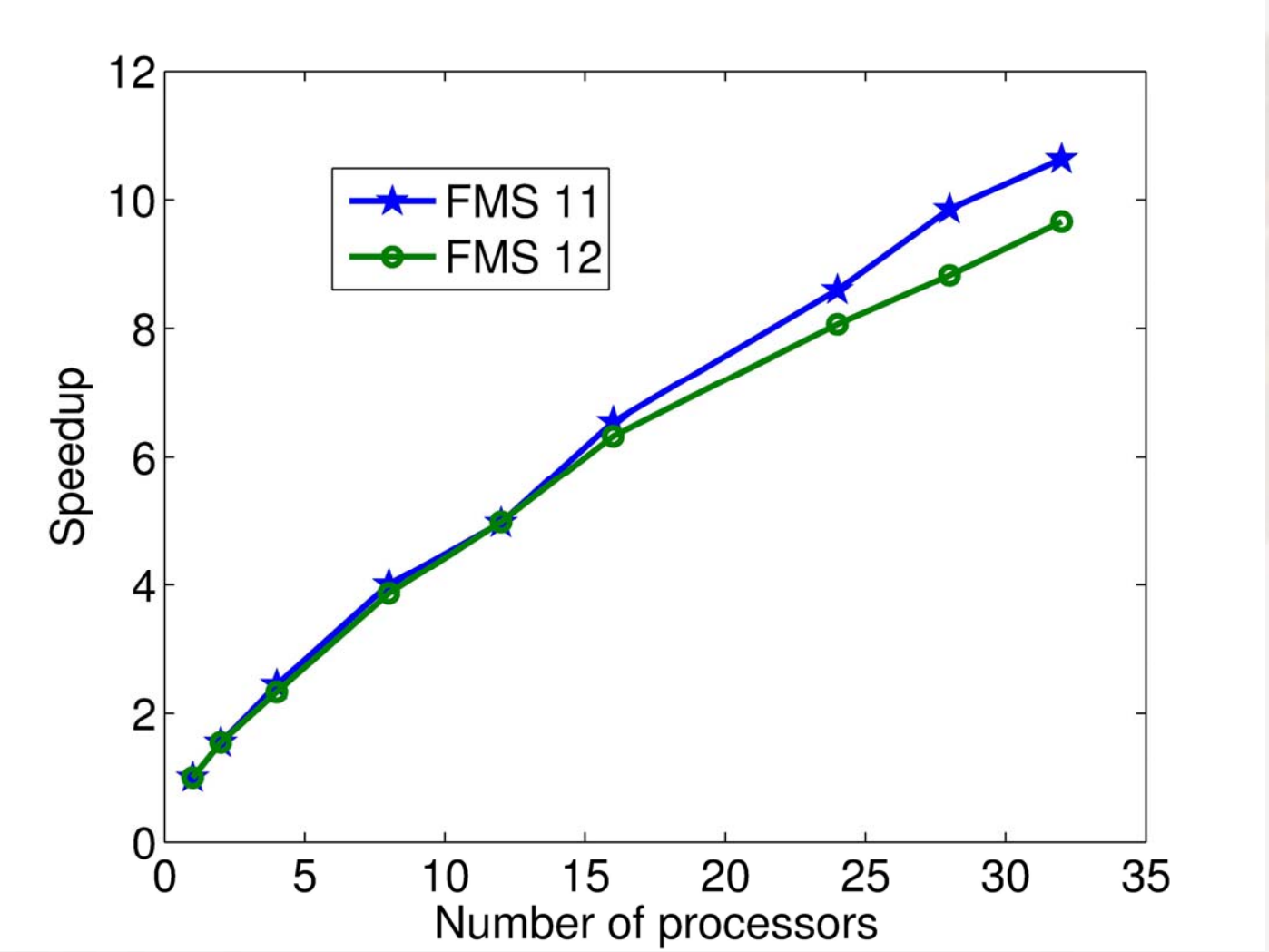
Model	States	Transitions	Blocks (N)	Size (MB)	
				MTBDD	Sparse
FMS (K=11)	54,682,992	518,030,370	1,365	297	6,137
FMS (K=12)	111,414,940	1,078,917,632	1,820	558	12,772
FMS (K=13)	216,427,680	2,136,215,172	2,380	1,005	25,273
Kanban (K=7)	41,644,800	450,455,040	120	18	5,314
Kanban (K=8)	133,865,325	1,507,898,700	165	43	17,767
Kanban (K=9)	384,392,800	4,474,555,800	220	95	52,674
Kanban (K=10)	1,005,927,208	12,032,229,352	286	195	141,535
Polling (K=20)	31,457,280	340,787,200	308	65	4,020
Polling (K=21)	66,060,288	748,683,264	324	141	8,820
Polling (K=22)	138,412,032	1,637,875,712	340	307	19,272

Experimental results: time

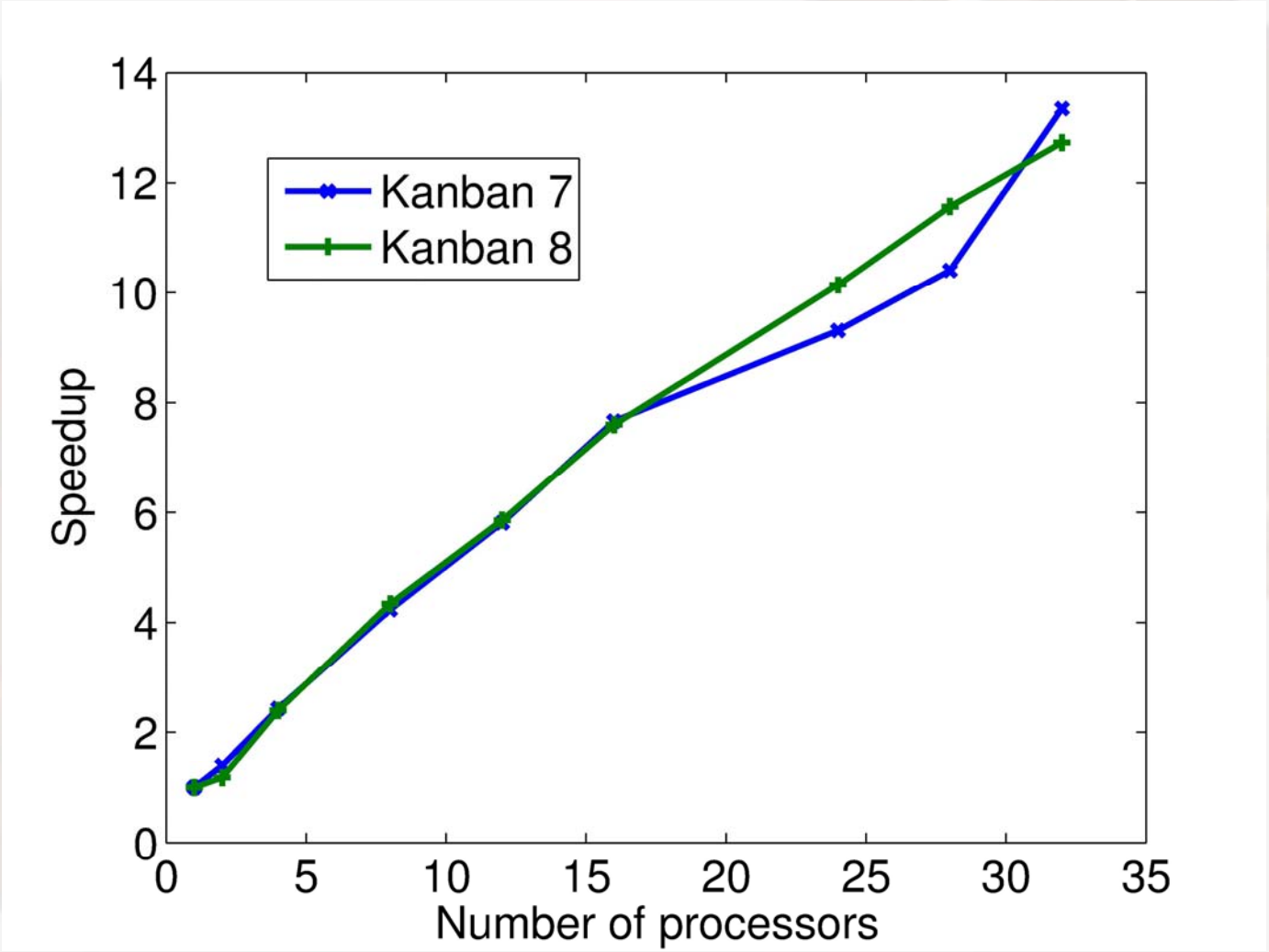
- Total execution times (in seconds) with 1 to 32 nodes
  - Termination condition maximum relative difference 10^{-6}
  - Block numbers selected to minimise storage

Num. nodes	FMS			Kanban				Polling		
	$K=11$	$K=12$	$K=13$	$K=7$	$K=8$	$K=9$	$K=10$	$K=20$	$K=21$	$K=22$
1	15,990	35,637	O/M	4,683	19,417	O/M	O/M	8,764	14,195	45,485
2	10,349	22,986	O/M	3,351	16,419	O/M	O/M	6,451	10,834	37,713
4	6,548	15,264	O/M	1,925	8,099	34,755	O/M	4,906	6,301	21,553
8	3,991	9,212	O/M	1,106	4,474	16,271	O/M	2,123	3,463	11,287
12	3,218	7,148	O/M	806	3,314	11,452	45,206	1,488	2,433	8,338
16	2,446	5,642	12,544	611	2,555	9,522	29,674	1,153	1,807	5,929
24	1,860	4,419	9,657	503	1,915	6,741	20,560	769	1,335	4,546
28	1,623	4,038	8,173	450	1,679	5,753	18,599	736	1,203	3,491
32	1,504	3,689	7,693	351	1,526	5,134	15,750	650	858	3,086

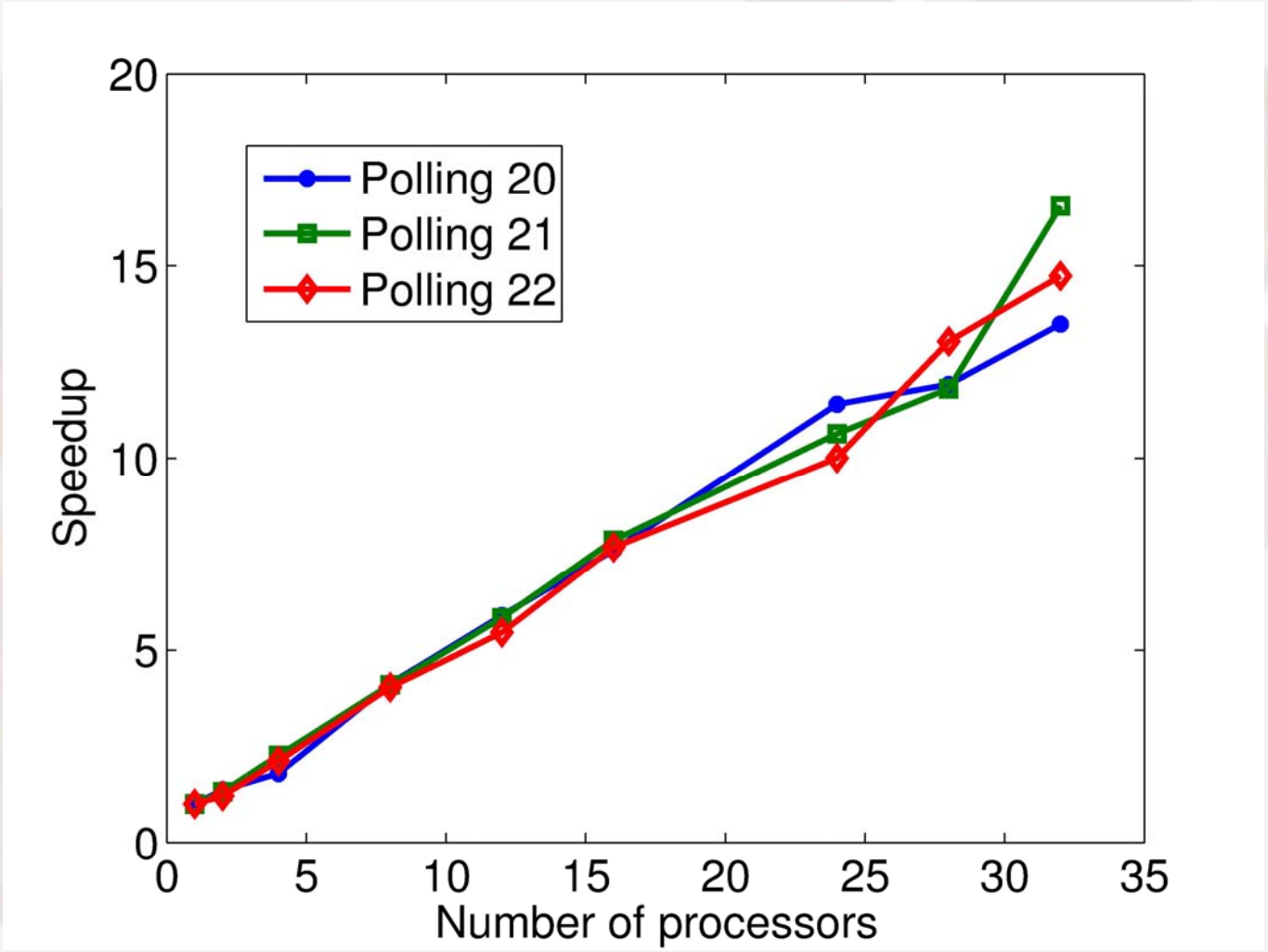
Experimental results: FMS speed-up



Experimental results: Kanban speed-up



Experimental results: Polling speed-up



What we have learnt…

- Experience with PRISM indicates
  - Symbolic representation very compact, >10^{10} states for CTMCs
  - Extremely large models feasible depending on regularity
  - Numerical computation often slow
  - Sampling-based computation can be even slower
- Can parallelise the symbolic approach
  - Facilitates extraction of the dependency information
  - Compactness enables storage of the full matrix at each node
  - Focus on steady-state solution for CTMCs, can generalise
  - Use wavefront techniques to parallelise
- Easy to distribute sampling-based computation
  - Individual simulation runs are independent

A Grid engine

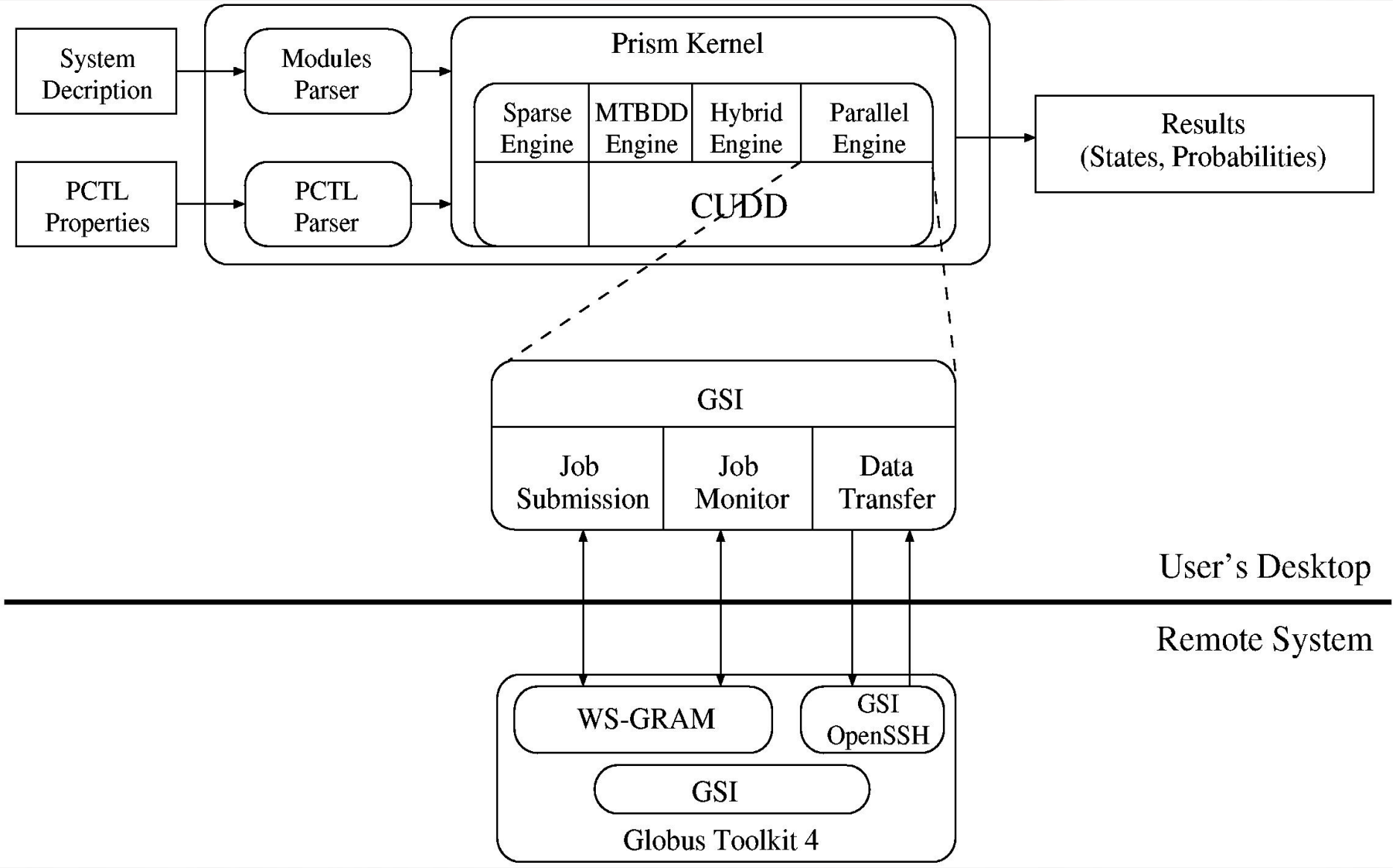
- Integrate a parallel numerical engine into PRISM
  - Access cluster from desktop
  - Manage remote computation resources for end users
  - Free end users from learning remote scheduling systems
  - Handling data transfer on behalf of end users
  - Monitoring job execution on remote computation resources

- Implemented parallel symbolic numerical engine
  - Based on MTBDD data structures
  - Solving linear equation systems for analysis of CTMC and DTMC
  - A Parallel Gauss-Seidel Iterative Method
    - for shared memory machines.
    - for message passing machines.

The Role of Globus Toolkits

- Globus version 4 (GT4)
  - Web-services and HPC standard
- Provide building blocks for our middleware
  - GSI for security
  - GRAM for job management
  - GSI-OpenSSH for file transfer
  - Grid services for data handling and job monitoring
- Offers convenience and high-performance to end users
  - Linear equations generated by PRISM
  - Matrices and vectors transferred to cluster
  - Solution can be queries, transferred back

Structure of Grid-enabled PRISM



PRISM with Globus

- Job Submission Component
  - Based on WS-GRAM
  - Generates job description files
  - Communicates with WS-GRAM services at remote resources
- Data Transfer
  - Using GSI-OpenSSH for file transfer
    - Matrices
    - Vectors
  - Create grid services for fine-grained data access
    - Block by block
- Job Monitoring (under development)
  - Information about job status
  - Runtime information
  - Convergence rate information

Experimental evaluation

Table 2. Total computation time (seconds) for model checking of Kanban models.

Num. nodes	Kanban		
	K=8	K=9	K=10
1	19,868	O/M	O/M
4	8,666	38,451	O/M
8	4,698	18,120	O/M
16	2612	10,172	32,459
24	2,006	7,123	22,971

Summary

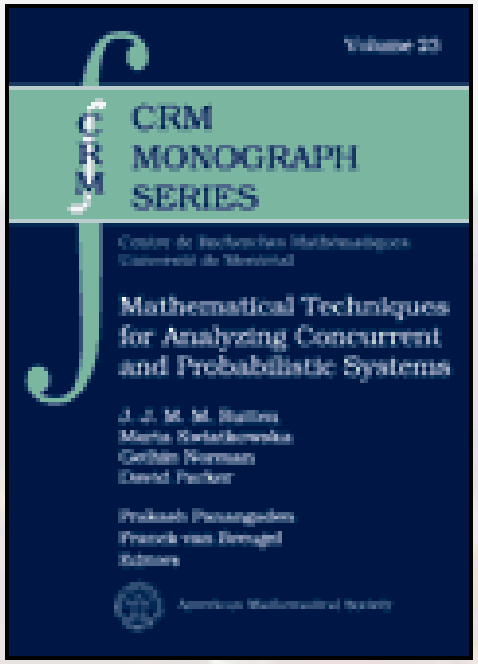
- Grid-based middleware for PRISM
  - Allows much larger CTMC models to be solved
  - Provides easy access of remote parallel computation resources for end users
  - A foundation for future parallelisation work in PRISM
- Symbolic approach particularly well suited to Wavefront parallelisation of Gauss-Seidel
  - Easy to extract task dependency information
  - Reduced memory requirement and communication load
  - Good speed-up
- Sampling-based Monte Carlo
  - Easy to parallelise, simulation traces independent
  - Implementing SSH based scheduling

Challenges for future

- State-space explosion has not gone away…
  - Are Grid techniques the answer?
- Exploiting structure
  - Abstraction, compositionality…
  - Parametric probabilistic verification?
- Efficient methods for continuous models
  - Continuous PTAs? Continuous time MDPs? LMPs?
  - Sampling-based, approximate model checking
- Proof assistant for probabilistic verification?
- Real software, not models!

PRISM collaborators worldwide

For more information…



J. Rutten, M. Kwiatkowska, G. Norman and D. Parker
Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems
P. Panangaden and F. van Breugel (editors), CRM Monograph Series, vol. 23, AMS
March 2004

www.cs.bham.ac.uk/~dxp/prism/

- Case studies, statistics, group publications
- Download, version 2.0 (approx. 2200 users)
- Linux/Unix, Windows, Macintosh versions
- Publications by others and courses that feature PRISM…