MARCIE - Model checking And Reachability analysis done effiCIEntly

[1] Martin Schwarick [2] Christian Rohr [1]Monika Heiner

[1]Brandenburg University of Technology Cottbus (BTU)

[2]Magdeburg Centre for Systems Biology (MaCS)

September 07, 2011

M odel checking A and R eachability done effi CIE ntly

- is a tool to validate generalized stochastic Petri nets
- supports qualitative and quantitative analysis of standard properties
- offers qualitative and quantitative model checking capabilities
- implements therefore state of the art techniques,
  most of them in a multi-threaded fashion

Features

- Qualitative analysis of bounded Petri nets
  - symbolic state space computation
  - behavioural properties
  - Computation Tree Logic (CTL) model checker
- "Exact" quantitative analysis of bounded generalized stochastic Petri nets
  - symbolic steady state/transient analysis
  - Continuous Stochastic Logic (CSL) model checker
  - reward computation
- Approximative quantitative analysis of generalized stochastic Petri nets
  - Gillespie's stochastic simulation
  - Fast adaptive uniformization
  - restricted CSL model checker

Features

- Qualitative analysis of bounded Petri nets
  - symbolic state space computation
  - behavioural properties
  - Computation Tree Logic (CTL) model checker
- "Exact" quantitative analysis of bounded generalized stochastic Petri nets
  - symbolic steady state/transient analysis
  - Continuous Stochastic Logic (CSL) model checker
  - reward computation
- Approximative quantitative analysis of generalized stochastic Petri nets
  - Gillespie's stochastic simulation
  - Fast adaptive uniformization
  - restricted CSL model checker

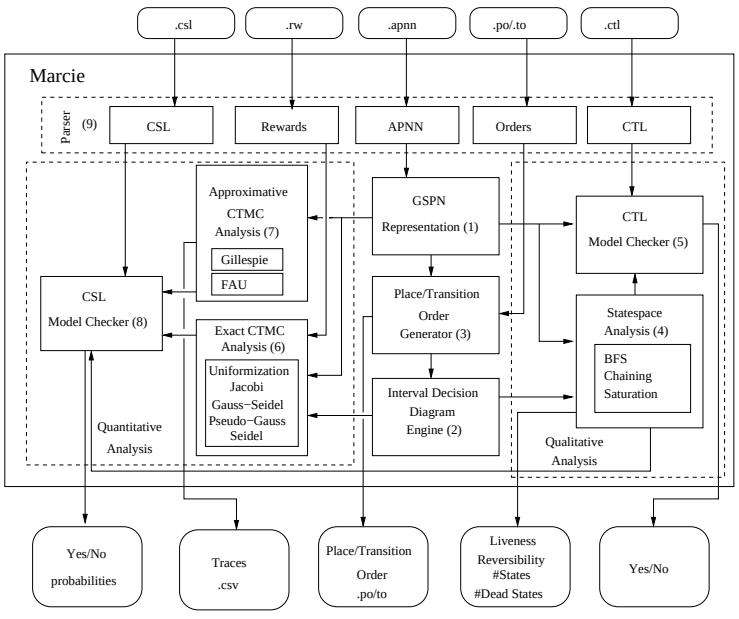
Features

- Qualitative analysis of bounded Petri nets
  - symbolic state space computation
  - behavioural properties
  - Computation Tree Logic (CTL) model checker
- "Exact" quantitative analysis of bounded generalized stochastic Petri nets
  - symbolic steady state/transient analysis
  - Continuous Stochastic Logic (CSL) model checker
  - reward computation
- Approximative quantitative analysis of generalized stochastic Petri nets
  - Gillespie's stochastic simulation
  - Fast adaptive uniformization
  - restricted CSL model checker

Features

- Qualitative analysis of bounded Petri nets
  - symbolic state space computation
  - behavioural properties
  - Computation Tree Logic (CTL) model checker
- "Exact" quantitative analysis of bounded generalized stochastic Petri nets
  - symbolic steady state/transient analysis
  - Continuous Stochastic Logic (CSL) model checker
  - reward computation
- Approximative quantitative analysis of generalized stochastic Petri nets
  - Gillespie's stochastic simulation
  - Fast adaptive uniformization
  - restricted CSL model checker

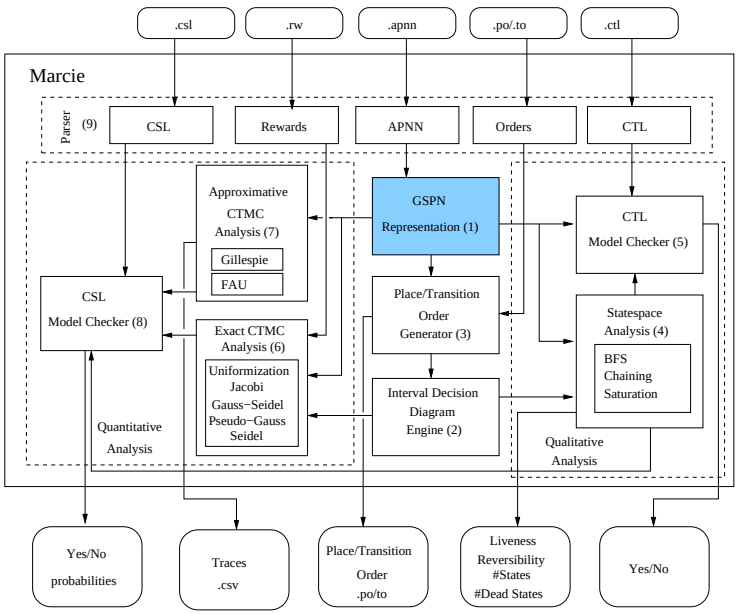
Architecture



Characteristics

- command line tool
- entirely written in C++
- about 40,000 LOC
- libraries: *flex*/*bison*, *GMP*, *phtreads*, *boost*
- available for Linux and MAC OS

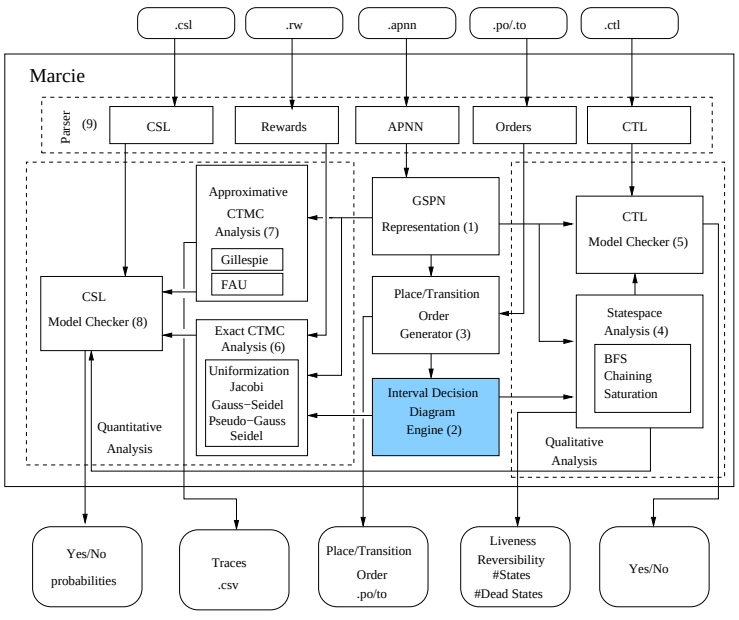
Architecture



GSPN Representation

- stochastic transitions with marking-dependent rate functions
- immediate transitions with weights
- extended arcs (read arcs, inhibitor arcs, reset arcs)

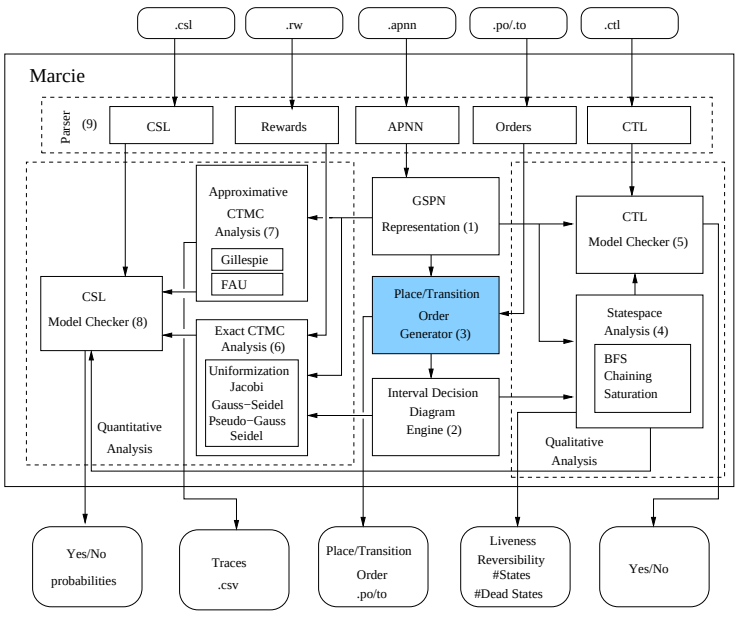
Architecture



IDD Engine

- Interval Decision Diagram (BDD generalization)
- implementation characteristics
  - shared IDDs
  - fast unique tables and operation caches
  - garbage collection
  - dedicated operations to fire *PN* transitions

Architecture



Order Generator

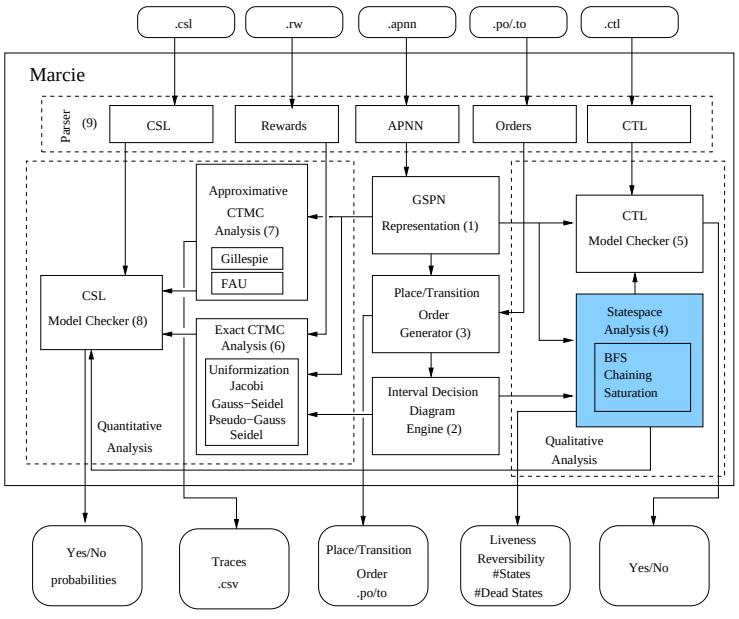
Pre-computation of static

- variable orders
  - small sized IDDs
- transition orders
  - reducing the number of computation steps for Transition Chaining and Saturation

based on the net structure

(heuristics)

Architecture



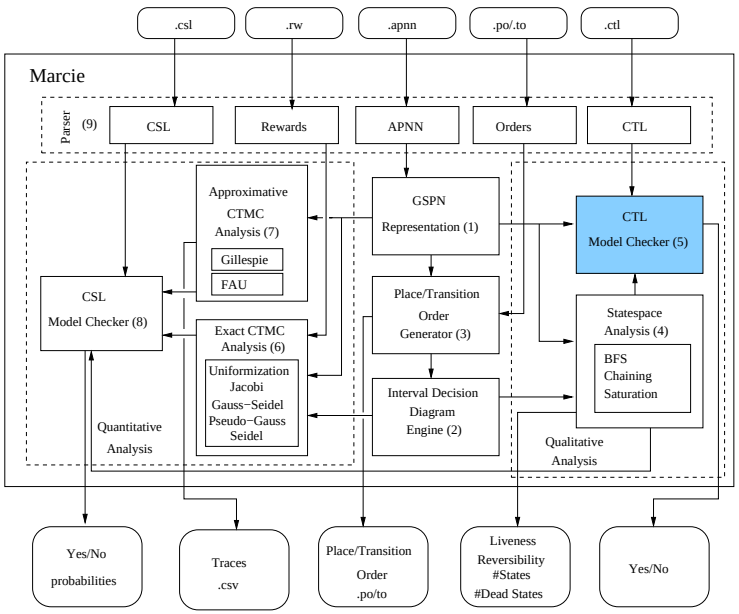
State Space Analysis
of bounded Petri nets
generation algorithms:

- Breath-First-Search
- Transition Chaining
- Saturation

behavioural properties:

- dead states
- reversibility
- liveness

Architecture



CTL Model Checker

efficient symbolic implementation of the standard CTL model checking algorithm

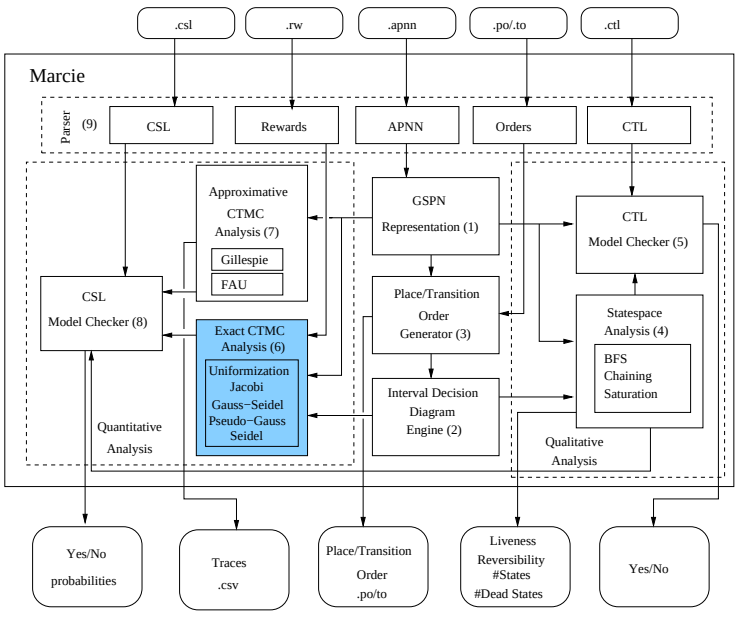
- atomic propositions: arithmetic and boolean expressions over the marking of places
- state formulas:

$$\phi :\equiv \phi \vee \phi \mid \neg\phi \mid \phi \wedge \phi \mid \phi \Leftrightarrow \phi \mid \phi \Rightarrow \phi \mid A\psi \mid E\psi$$

- path formulas:

$$\psi :\equiv X\phi \mid \phi U\phi \mid F\phi \mid G\phi$$

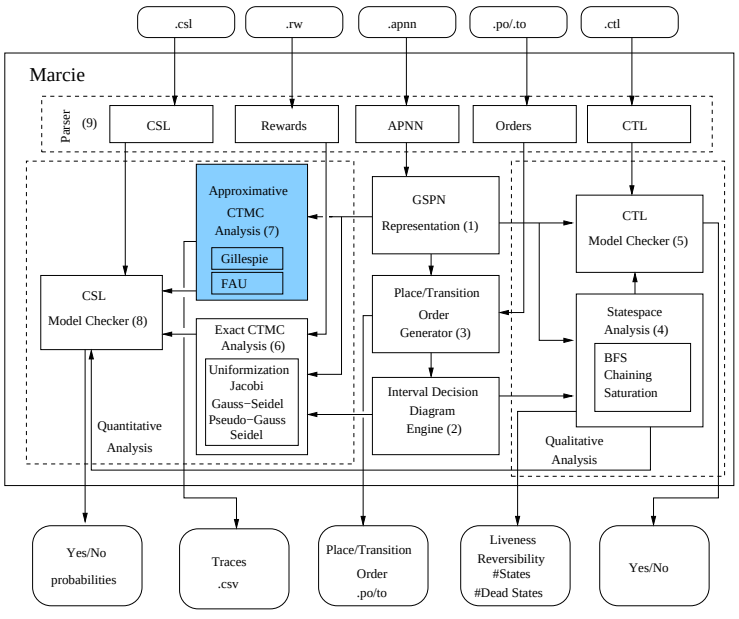
Architecture



"Exact" CTMC Analysis

- computation of probability distributions
  - transient analysis - uniformization method
  - steady state analysis - iterative methods – Jacobi and (Pseudo-) Gauss-Seidel
- basic operation is a matrix-vector multiplication
  - "on-the-fly" computation of the matrix based on the symbolic state space representation (IDD)
  - *multi-threaded*

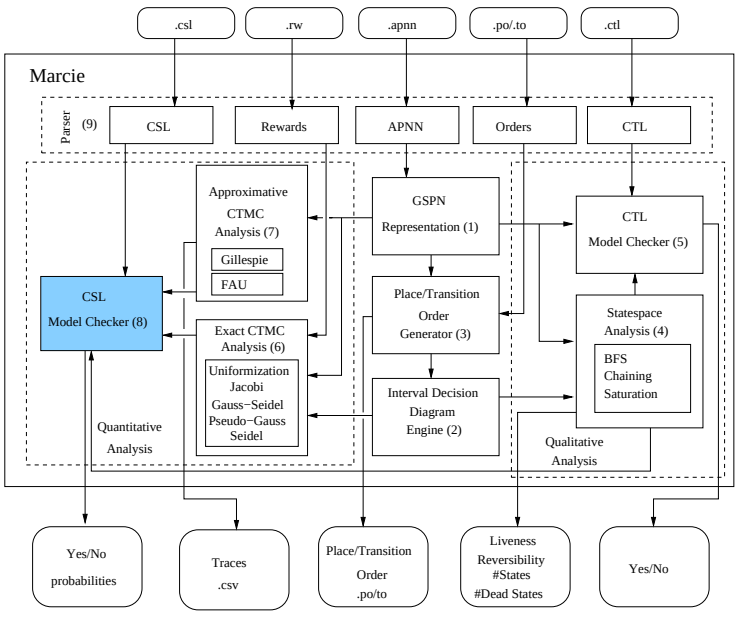
Architecture



Approximative CTMC Analysis

1. Stochastic Simulation
   - single trajectories
   - modified Gillespie algorithm
   - confidence intervals
   - *multi-threaded*
2. Fast Adaptive Uniformization
   - adaptive uniformization
   - state pruning
   - produces often a very small part of the state space

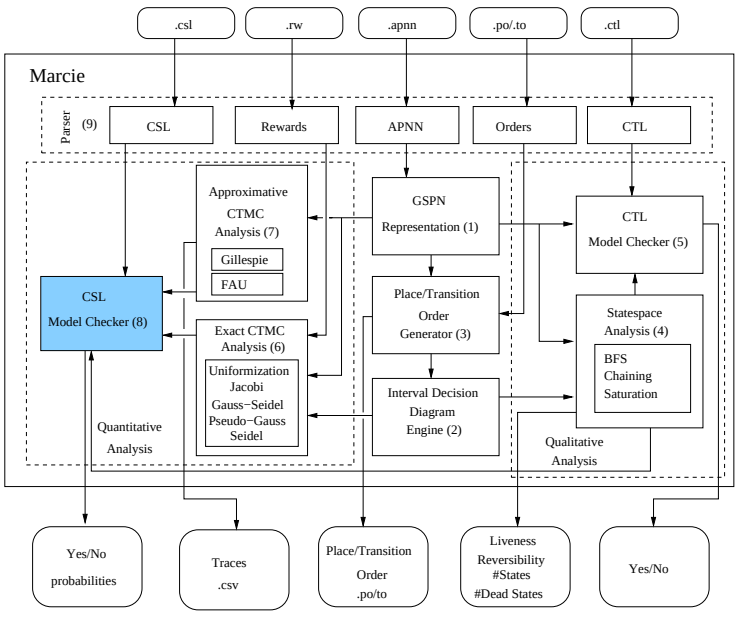
Architecture



CSL Model Checker

- CSL is a stochastic extension of CTL
- $\mathcal{P}_{\circ p}$ replaces E and A
- $\mathcal{S}_{\circ p}$ allows to reason about steady state behaviour
- temporal operators are decorated with time intervals

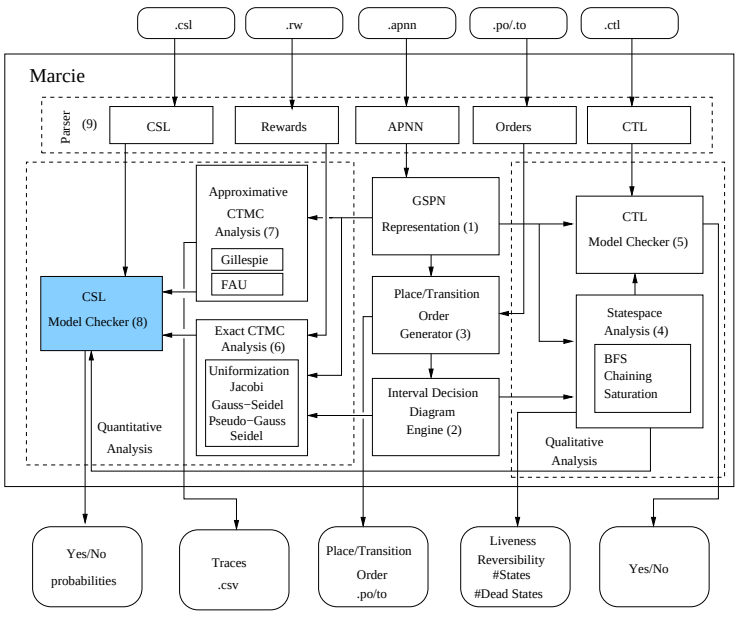
Architecture



CSL Model Checker

- CSL model checking algorithm is similar to CTL
- but the evaluation of $\mathcal{P}$ and $\mathcal{S}$ requires the computation of a probability distribution
- standard CTMC analysis techniques can be applied
- for unnested and time-bounded formulas approximation or simulation can be used
- *multi-threaded*

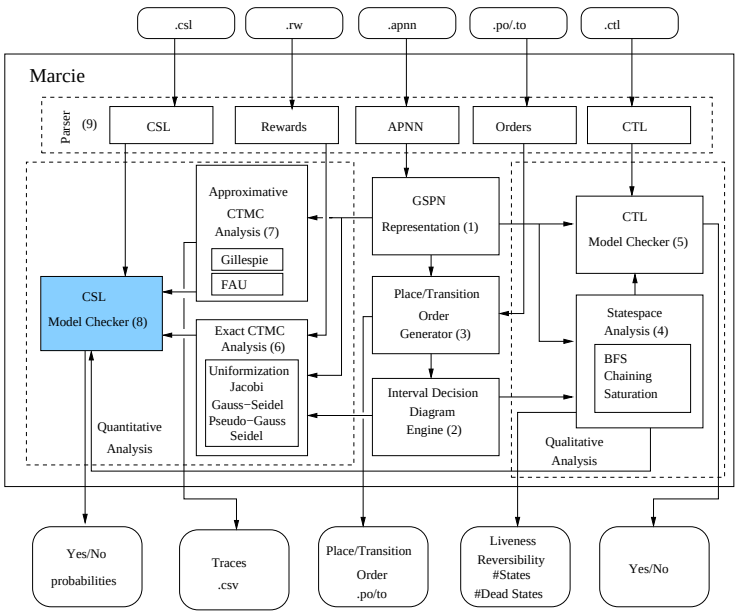
Architecture



CSL Model Checker

- extensions to reason about expectations of rewards
- new operator $\mathcal{R}_{\circ\nabla}$ with
  - $[C_{<=t}]$ the cumulative reward
  - $[I_{=t}]$ the instaneous reward
  - $[F(sp)]$ the cumulative reward until the first occurence of a state satisfying sp
  - $[\mathcal{S}]$ expected reward at steady state

Architecture

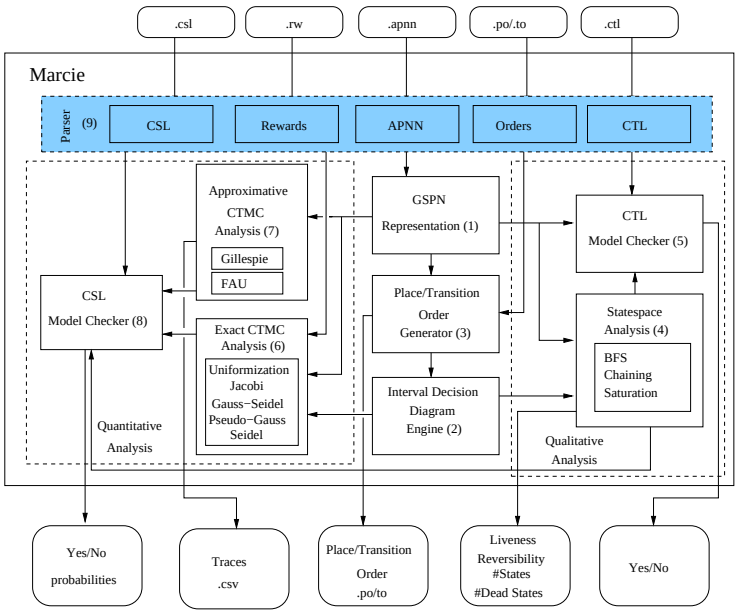


CSRL Model Checker

- Continuous Stochastic Reward Logic
- temporal operators are decorated with a time *and* a reward interval
- performability:

$$\mathcal{P}_{=?}[true\mathbf{U}_{[t,t][0,r]}\psi]$$

Architecture



Parser

component to allow the input of

- the GSPN in an adaption of *APNN*
- reward structures as in the PRISM language
- CSL/CTL formulas
- place/transition orders in simple text format

Related Tools

short selection of widely used tools

- PRISM - University of Oxford
  `http://www.prismmodelchecker.org`
- MRMC - RWTH Aachen
  `http://www.mrmc-tool.org`
- SMART - University of California at Riverside
  `http://www.cs.ucr.edu/~ciardo/SMART/`

Related Tools

Features

	MARCIE	PRISM	MRMC	SMART

Related Tools

Features

	MARCIE	PRISM	MRMC	SMART
high level description	GSPN	GCL		GSPN*

Related Tools

Features

	MARCIE	PRISM	MRMC	SMART
high level description	GSPN	GCL		GSPN*
formalism		DTMC	DTMC/DMRM	DTMC
	CTMC	CTMC	CTMC/CMRM	CTMC
		MDP	MDP	

Related Tools

Features

	MARCIE	PRISM	MRMC	SMART
high level description	GSPN	GCL		GSPN*
formalism		DTMC	DTMC/DMRM	DTMC
	CTMC	CTMC	CTMC/CMRM	CTMC
		MDP	MDP	
model checking	CTL			CTL
		LTL		
	CS(R)L	PCTL/CSL	PCTL/CS(R)L	

Related Tools

Features

	MARCIE	PRISM	MRMC	SMART
high level description	GSPN	GCL		GSPN*
formalism		DTMC	DTMC/DMRM	DTMC
	CTMC	CTMC	CTMC/CMRM	CTMC
		MDP	MDP	
model checking	CTL			CTL
		LTL		
	CS(R)L	PCTL/CSL	PCTL/CS(R)L	
exact engine		explicit	explicit	explicit
	IDD	(MT)BDD		(MT)MDD,MD,EVDD

Related Tools

Features

	MARCIE	PRISM	MRMC	SMART
high level description	GSPN	GCL		GSPN*
formalism		DTMC	DTMC/DMRM	DTMC
	CTMC	CTMC	CTMC/CMRM	CTMC
		MDP	MDP	
model checking	CTL			CTL
		LTL		
	CS(R)L	PCTL/CSL	PCTL/CS(R)L	
exact engine		explicit	explicit	explicit
	IDD	(MT)BDD		(MT)MDD,MD,EVDD
simulative engine	yes	yes	yes	yes

Related Tools

Features

	MARCIE	PRISM	MRMC	SMART
high level description	GSPN	GCL		GSPN*
formalism		DTMC	DTMC/DMRM	DTMC
	CTMC	CTMC	CTMC/CMRM	CTMC
		MDP	MDP	
model checking	CTL			CTL
		LTL		
	CS(R)L	PCTL/CSL	PCTL/CS(R)L	
exact engine		explicit	explicit	explicit
	IDD	(MT)BDD		(MT)MDD,MD,EVDD
simulative engine	yes	yes	yes	yes
multi-threading	yes	no	no	no

PRISM vs. MARCIE

Exact Quantitative Analysis

Similarities

- based on Decision Diagrams (DD)
- multiplication by DD traversal,
  state indices are computed using offsets during the traversal
- explicit storage of probability vectors and the diagonal entries
- similar caching strategies

PRISM vs. MARCIE

Exact Quantitative Analysis

Differences

	PRISM	MARCIE

PRISM vs. MARCIE

Exact Quantitative Analysis

Differences

	PRISM	MARCIE
state space	BDD	IDD

PRISM vs. MARCIE

Exact Quantitative Analysis

Differences

	PRISM	MARCIE
state space	BDD	IDD
rate matrix	MTBDD or BDD+SM	IDD + GSPN

PRISM vs. MARCIE

Exact Quantitative Analysis

Differences

	PRISM	MARCIE
state space	BDD	IDD
rate matrix	MTBDD or BDD+SM	IDD + GSPN
prior knowledge of the boundedness degree	yes	no

PRISM vs. MARCIE

Exact Quantitative Analysis

Differences

	PRISM	MARCIE
state space	BDD	IDD
rate matrix	MTBDD or BDD+SM	IDD + GSPN
prior knowledge of the boundedness degree	yes	no
#(variables)	depends on the boundedness degree	#(places)

PRISM vs. MARCIE

Exact Quantitative Analysis

Differences

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

PRISM vs. MARCIE

Exact Quantitative Analysis

Differences

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

PRISM vs. MARCIE

Exact Quantitative Analysis

Differences

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

Experimental Results

Case studies

- Flexible Manufacturing System
- Kanban
- Polling Server System
- AKAP scaffold-mediated crosstalk
- Angiogenetic process

Test system

- $8 \times 2.2 Ghz$ MAC Pro
- 32GB RAM
- Cent OS 5.5

Experiments

- transient analysis
- steady state analysis
- simulative analysis

State space size

Size of the reachability graphs for different model configurations.

model	N	states	transitions
FMS	2	810	3,699
	4	35,910	237,120
	6	537,768	4,205,670
	8	4,459,455	38,533,968
	10	25,397,658	234,523,289
	12	111,414,940	1,078,917,632
	14	403,259,040	4,047,471,180
Kanban	2	4,600	28,120
	4	454,475	3,979,850
	6	11,261,376	115,708,992
	8	133,865,325	1,507,898,700
PSS	5	240	800
	10	15,360	89,600
	15	737,280	6,144,000
	20	31,457,280	340,787,200
AKAP	3	1,632,240	12,691,360
	4	15,611,175	141,398,580
	5	74,612,328	734,259,344
	6	386,805,104	4,116,788,172
ANG	2	5,384	26,193
	4	2,413,480	21,810,412
	6	277,789,578	24,813,347,031

Transient analysis

Total run-time including state space generation, initialization and probability computation.
The column *iter* gives the number of required iterations.

model			MARCIE				PRISM	
	N	iter	1	2	4	8	hybrid	sparse
FMS	8	202	2m27s	2m28s	1m51s	1m28s	3m47s	2m50s
	10	202	13m11s	9m44s	7m51s	8m22s	27m39s	15m57s
	12	208	55m54s	42m41s	31m56s	29m57s	2h10m47s	1h39m05s
	14	208	4h07m51s	3hm35ms07s	2h48m59s	2h41m41s	8h26m38s	‡
Kanban	4	181	6s	3s	2s	2s	12s	8s
	6	181	1m57s	1m34s	48s	34s	5m16s	3m09s
	8	181	23m59	17m53s	9m18s	6m41s	1h08m15s	43m23s
PSS	10	377	$\ll$1s	$\ll$1s	$\ll$1s	$\ll$1s	2s	2s
	15	377	17s	7s	4s	3s	33s	17s
	20	377	9m02s	5m32s	2m59s	2m06s	29m18s	14m22s
AKAP	4	1,532	25m15s	17m09s	7m49s	5m13s	52m20s	24m12s
	5	1,850	2h40m14s	1h34m36s	47m54s	30m26s	–	–
	6	2,218	–	–	5h44m53s	3h49m45s	†	†
ANG	4	547	1m53s	1m18s	54s	44s	2m27s	7m24s
	5	794	31m08s	20m50s	13m45s	11m46s	34m06s	4h25m34s
	6	1,093	7h36m57s	4h39m52s	3h17m47s	3h00m16s	†	†

– means that the experiment was canceled after 12 hours
† means that the CTMC could not be created using the default tool settings ‡ means that we ran out of memory

Steady state analysis

Total run-time including state space generation, initialization, reward computation - if necessary, and probability computation. The number of required iterations is given in the columns *iter*. If possible we used the Jacobi solvers. For the Kanban and ANG models we used the Gauss-Seidel solvers. The number of iterations required by the Gauss-Seidel solvers differ between MARCIEand PRISM because of different variable orders.

model		MARCIE					PRISM		
	N	iter	1	2	4	8	iter	hybrid	sparse
FMS	2	378	$\ll$1s	$\ll$1s	$\ll$1s	$\ll$1	378	2s	3s
	6	1,084	59s	42s	57s	34s	1,084	41s	30s
	10	1,812	1h11m15s	45m07s	30m31s	24m47s	1,812	2h25m42s	37m15s
	12	2,193	6h13m53s	3h52m32s	2h26m17s	2h04m45s	2,193	–	3h32m11s
	14	2,589	–	–	–	9h10m04s	†	†	†
Kanban	2	48	$\ll$1s	Gauss-Seidel			189	2	2s
	4	122	15s				323	12s	10s
	6	224	11m07s	no multi-threading support			622	9m48s	6m18s
	8	356	3h57m23s				999	3h27m27s	1h59m59s
PSS	10	406	$\ll$1s	$\ll$1s	$\ll$1s	$\ll$1s	406	2s	2s
	15	657	14s	9s	5s	3s	657	24s	21s
	20	920	16m16s	10m58s	6m21s	3m08s	920	31m28s	26m31s
AKAP	4	1,433	21m17s	13m07s	7m04s	4m40s	1,433	17m37s	17m08s
	5	1,348	1h46m52s	1h05m12s	37m39s	23m03s	–	–	–
	6	1,659	–	8h00m02s	3h53m29s	2h43m48s	–	–	†
ANG	2	*n.a.*	48s	Gauss-Seidel			*n.a.*	4m05s	28s
	3	*n.a.*	1s24m55s				†	†	†
	4	*n.a.*	–	no multi-threading support			†	†	†

– means that the experiment was canceled after 12 hours
† means that the CTMC could not be created using the default tool settings

Simulative analysis

Total run-time for simulations done with MARCIEand PRISM.
The confidence level is 99%, the desired accuracy 1×10^{-5}, which leads to 66, 348, 303 simulation runs.

model	N	t	Threads	MARCIE	PRISM
FMS	14	1	1	40m43s	3h24m13s
			8	5m4s	*n.a.*
Kanban	10	10	1	13m48s	2h23m58s
			8	1m58s	*n.a.*
PSS	20	1	1	4h10m4s	62h32m33s
			8	31m53s	*n.a.*
AKAP	6	1	1	2m37s	10m23s
			8	25s	*n.a.*
ANG	6	1	1	38m48s	8h26m57s
			8	5m20s	*n.a.*

Conclusions

MARCIE is an analysis tool for generalized stochastic Petri nets which

- is easy to use
  - flat Petri net models
  - no prior knowledge of the boundedness degree
  - automatic computation of variable orders
- assures high performance
  - efficient IDD engine
  - efficient simulation engine
- offers model checking of branching time logic
  - qualitative: Computation Tree Logic
  - quantitative: Continuous Stochastic Logic

Conclusions

MARCIE is an analysis tool for generalized stochastic Petri nets which

- is easy to use
  - flat Petri net models
  - no prior knowledge of the boundedness degree
  - automatic computation of variable orders
- assures high performance
  - efficient IDD engine
  - efficient simulation engine
- offers model checking of branching time logic
  - qualitative: Computation Tree Logic
  - quantitative: Continuous Stochastic Logic

Conclusions

MARCIE is an analysis tool for generalized stochastic Petri nets which

- is easy to use
  - flat Petri net models
  - no prior knowledge of the boundedness degree
  - automatic computation of variable orders
- assures high performance
  - efficient IDD engine
  - efficient simulation engine
- offers model checking of branching time logic
  - qualitative: Computation Tree Logic
  - quantitative: Continuous Stochastic Logic

Conclusions

MARCIE is an analysis tool for generalized stochastic Petri nets which

- is easy to use
  - flat Petri net models
  - no prior knowledge of the boundedness degree
  - automatic computation of variable orders
- assures high performance
  - efficient IDD engine
  - efficient simulation engine
- offers model checking of branching time logic
  - qualitative: Computation Tree Logic
  - quantitative: Continuous Stochastic Logic

Work in Progress

- PLTLc model checking
- steady state simulation
- parallel simulative model checking on Graphical Processing Units
- parallelization of the (Pseudo-)Gauss-Seidel solver
- completion of the CSRL model checker

Thanks for your attention.

Questions?

Demonstration?

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