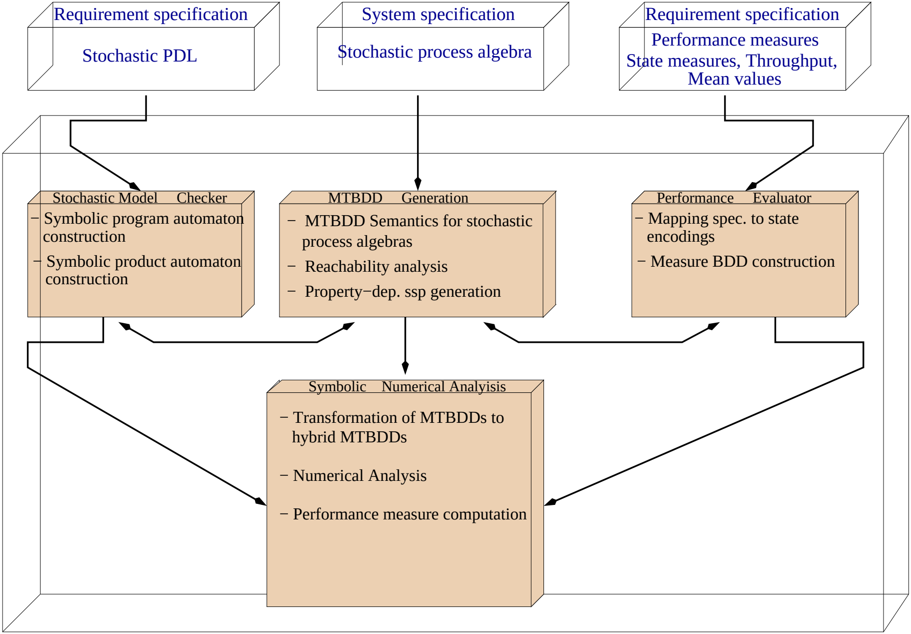
CASPA - A Tool for Symbolic Performance Evaluation and Stochastic Model Checking

Boudewijn R. Haverkort[1], Matthias Kuntz[1], Martin Riedl[2],
Johann Schuster[2], Markus Siegle[2]

[1]: Universiteit Twente
[2]: Universität der Bundeswehr München

Two Decades of Probabilistic Verification
Workshop at Lorentz Center, Leiden
12 - 16 November 2007

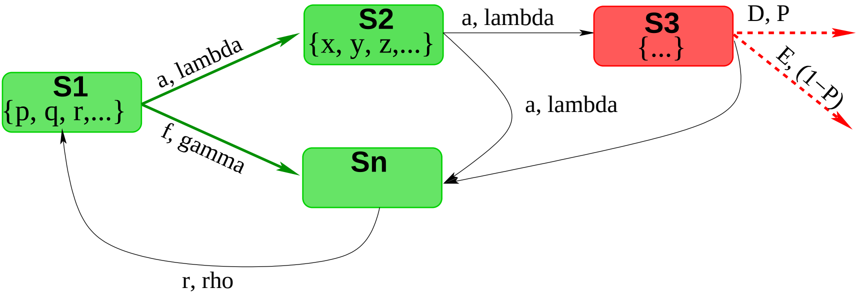
Principle Architecture of CASPA


Requirement specification
Stochastic PDL
System specification
Stochastic process algebra
Requirement specification
Performance measures
State measures, Throughput, Mean values
Stochastic Model Checker
– Symbolic program automaton construction
– Symbolic product automaton construction
MTBDD Generation
– MTBDD Semantics for stochastic process algebras
– Reachability analysis
– Property−dep. ssp generation
Performance Evaluator
– Mapping spec. to state encodings
– Measure BDD construction
Symbolic Numerical Analyisis
– Transformation of MTBDDs to hybrid MTBDDs
– Numerical Analysis
– Performance measure computation

Example Specification

```
int max = 3
Process  := Queue(0)
Queue(n [max]) := [n>=2] -> (serve, mu);Queue(n-2)
                   [n < max] -> (arrival, lambda);Queue(n+1)
                   [*] -> (fail, gamma);Repair
Repair := (repair, rho);Queue(0)
/***Measure specification
statemeasure Fill2 Queue(n > 0) & !Queue(n = max)
meanvalue Occupancy Queue(n)
throughputmeasure Serve serve
spdl P(> 0.9){tt [arrival*;repair;arrival*](4.3) Queue(n=max)}
```

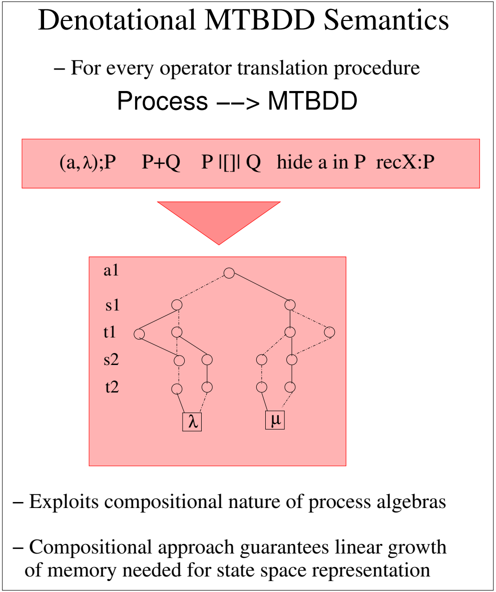
Analysable Models: Extended SLTS



S1 Tangible state, only timed transitions

S3 Vanishing state, at least one untimed transition

State Space Generation



Denotational MTBDD Semantics

– For every operator translation procedure

Process –-> MTBDD

(a,λ);P P+Q P |[]| Q hide a in P recX:P

– Exploits compositional nature of process algebras

– Compositional approach guarantees linear growth of memory needed for state space representation

Numerical Analysis

Three basic types of measures:

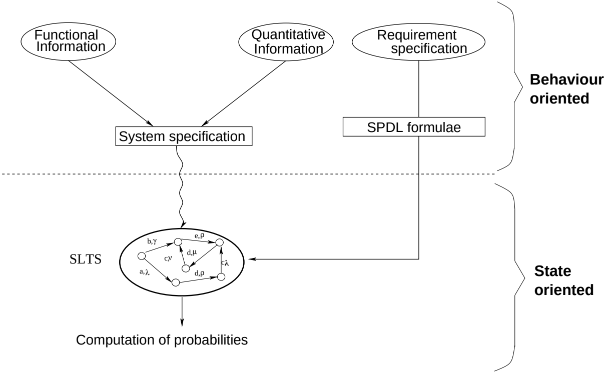
1. State measures
2. Mean values
3. Throughput measures

To compute measures we have to the following things:

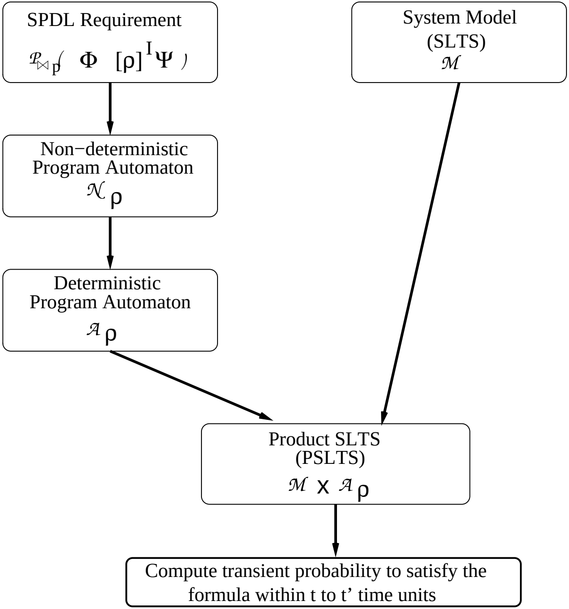
1. Relate measure definition and states that are relevant for the measure at hand
2. Compute state probability vector (transient or steady state)
3. Compute actual value of the measure

Stochastic Propositional Dynamic Logic (SPDL)

- SPDL is based on the Logic PDL (Fisher, Ladner 1979)
- CSL + action sequences
- Action sequences: Extended regular expressions (Programs)



Model checking SPDL path formulae



SPDL Requirement

$\mathcal{P}_{\bowtie p}(\Phi \ [\rho]^{I} \Psi)$

System Model (SLTS)

$\mathcal{M}$

Non−deterministic Program Automaton

$\mathcal{N}_{\rho}$

Deterministic Program Automaton

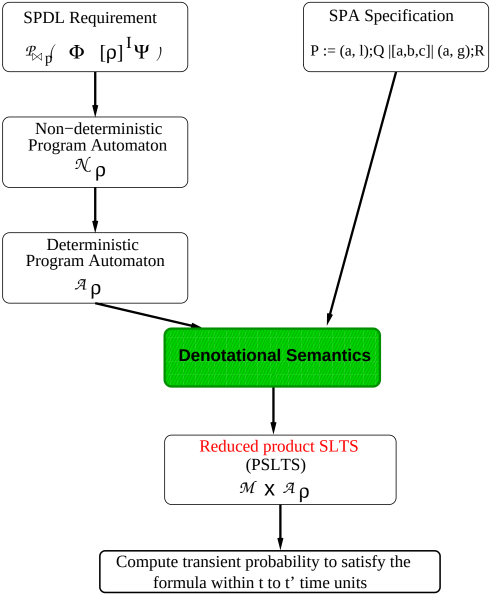
$\mathcal{A}_{\rho}$

Product SLTS (PSLTS)

$\mathcal{M} \times \mathcal{A}_{\rho}$

Compute transient probability to satisfy the formula within t to t’ time units

Property-Driven State Space Generation



SPDL Requirement

$\mathcal{P}_{\bowtie p}(\Phi \ [\rho]^{I} \Psi)$

SPA Specification

P := (a, l);Q |[a,b,c]| (a, g);R

Non−deterministic Program Automaton

$\mathcal{N}_{\rho}$

Deterministic Program Automaton

$\mathcal{A}_{\rho}$

Denotational Semantics

Reduced product SLTS (PSLTS)

$\mathcal{M} \times \mathcal{A}_{\rho}$

Compute transient probability to satisfy the formula within t to t’ time units

Case Studies

We have performed the following case studies:

- Flexible Manufacturing System
- Kanban System
- Polling System
- Fault Tolerant Computer System
- Handover Procedure in a Cellular Mobile Radio Network
- Mainframe System with Failures
- Tandem Queuing System

Empirical Results: Steady State Analysis

N	Reach. States	MTBDD Nodes		MTBDD Gen.	Iterations	Num. Analysis
		peak	final			
CASPA:						
5	2,546,432	25,514	5,392	0.32 sec.	457	2min 47sec
6	11,261,376	47,395	8,086	0.69 sec.	625	18min 30sec
7	41,644,800	76,230	10,389	1.32 sec.	804	1h 27min
10	1,005,927,208	248,461	23,231	6.92 sec.	-	-
12	5,519,907,575	414,719	32,324	12.97 sec.	-	-

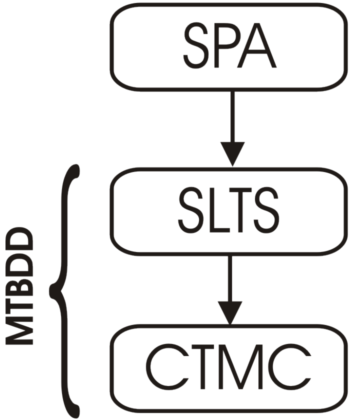
Pentium IV, 3.0 GHz, 1GB RAM, SuSe Linux 10.2

Empirical Results: Transient Analysis

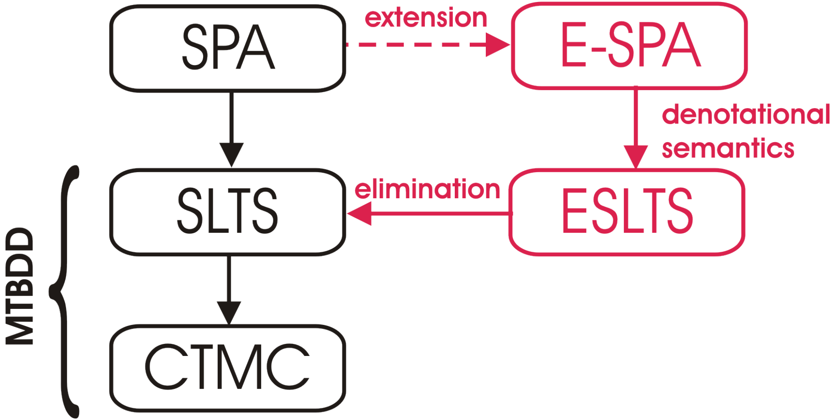
N	Reach. States	MTBDD Nodes		MTBDD Gen.	Iterations	Num. Analysis
		peak	final			
CASPA:						
5	2,546,432	25,514	5,392	0.32 sec.	282	1min 44sec
6	11,261,376	47,395	8,086	0.69 sec.	282	8min 6sec
7	41,644,800	76,230	10,389	1.32 sec.	804	30min 31sec
10	1,005,927,208	248,461	23,231	6.92 sec.	-	-
12	5,519,907,575	414,719	32,324	12.97 sec.	-	-

Pentium IV, 3.0 GHz, 1GB RAM, SuSe Linux 10.2

Extension of CASPA for handling immediate transitions

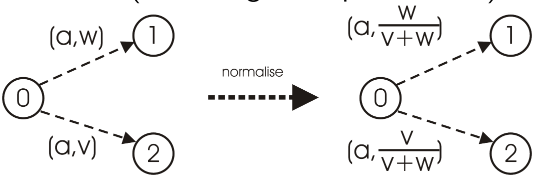


Extension of CASPA for handling immediate transitions



Semantics and Implementation Aspects

- semantics (from weights to probabilities)



- implementation
  - two separate MTBDDs for immediate and Markovian transitions
  - semi-symbolic elimination algorithm (no additional MTBDD variables needed)
  - can handle cycles of immediate transitions

A GUI for CASPA? - Why?

Reasons for a GUI:

- clearness of the graphical representation of the model
- no need to tackle with CASPA syntax and call conventions

Therefore CASPAEdit has

- an import/export mechanism for models in CASPA syntax,
- a panel to support setting the CASPA arguments (e.g. type of numerical algorithm) and its execution

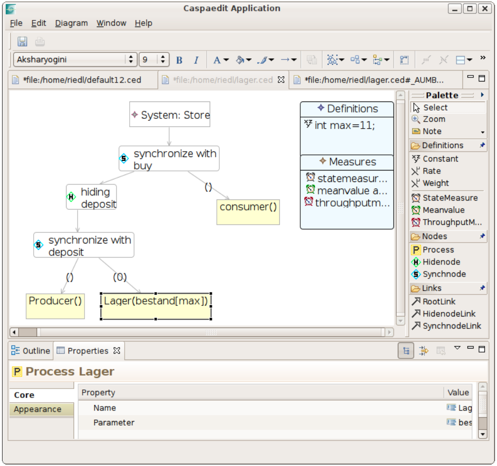
and furthermore it provides

- automatic layouting and
- complete syntactical validation
- partial semantical validation

CASPAEdit - System Layer

A system is defined by a directed graph

- nodes for
  - the root process
  - synchronisation
  - hiding
  - sequential processes
- links between the nodes
- and definitions of constants, measures

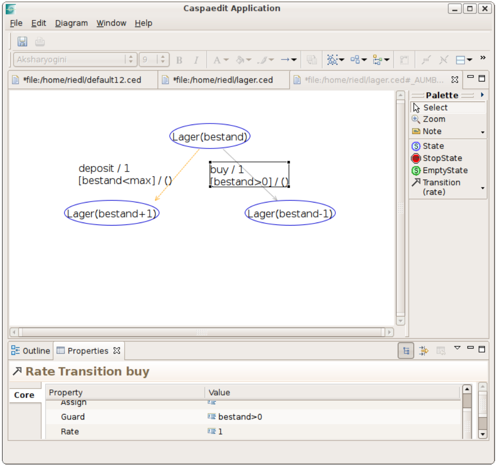


Property	Value
Name	Lag
Parameter	bes

CASPAEdit - Process Layer

Definition of a process by

- states
  - name
  - values of process parameters
- transitions
  - label
  - weight/rate
  - guard
  - assignment



Rate Transition buy

Core

Property	Value
Assign	
Guard	bestand>0
Rate	1

CASPAEdit - Implementation Aspects

The CASPA GUI has been developed in a model driven way using Eclipse

- and its Graphical Modeling Framework (GMF) and
- is usable as an Eclipse Plugin or as a Rich Client Application.

It supports the CASPA syntax and therefore the existing models with an

- import function (JavaCC generated parser: for syntactical Analysis of a CASPA-Model and the instantiation of the internal model), and an
- export function (Java Emitter Templates: for Model-to-Text (M2T) transformations).

CASPA Tool Presentation

Please ask

- Markus Siegle
- Matthias Kuntz
- Martin Riedl
- Johann Schuster