Towards stochastic inference driven SOA testing

Ariele-Paolo Maesano Fabio De Rosa

UPMC/LIP6 - Simple Engineering

Service Oriented Architecture (SOA) Testing

- SOA = design & implementation style that allows organizations to put into operation distributed architectures of loosely coupled systems in order to achieve flexible business automation
- Service contracts are public - system internals are private
- Only available verification methods:
  - black-box testing of individual systems
  - grey-box testing of interactions among systems
- SOA testing is a means for:
  - failure discovering
  - troubleshooting

SOA Testing Problem

- SOA Testing is a heavy task - complex networks, complex exchange scenarios, reduced control, reduced observability, reduced trust, ...
- Services Architecture Under Test (SAUT) is a collection of SUTs (Systems under test) connected by channels conveying communicative actions (message sending, remote procedure call, rpc reply)
- from the standpoint of SOA testing:
  - SUTs implementations are hidden - black-box testing
  - (some or all) channels are observable - grey-box testing
  - only information available to the Tester: match/mismatch between actual communicative actions and expected ones

Solution: automated SOA testing environment

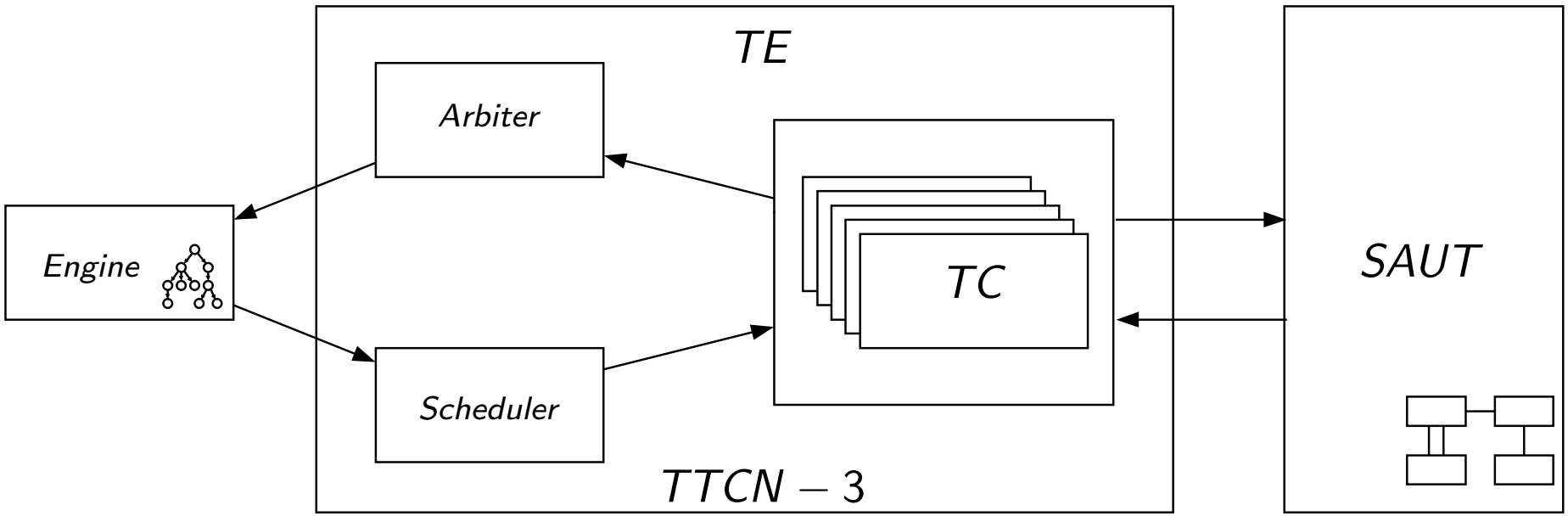
SOA testing automation concerns:

- test case generation
- test run execution and evaluation
- **test campaign dynamic planning (optimization) - reliability computation (Bayesian network inference)**

Solution: automated SOA testing environment

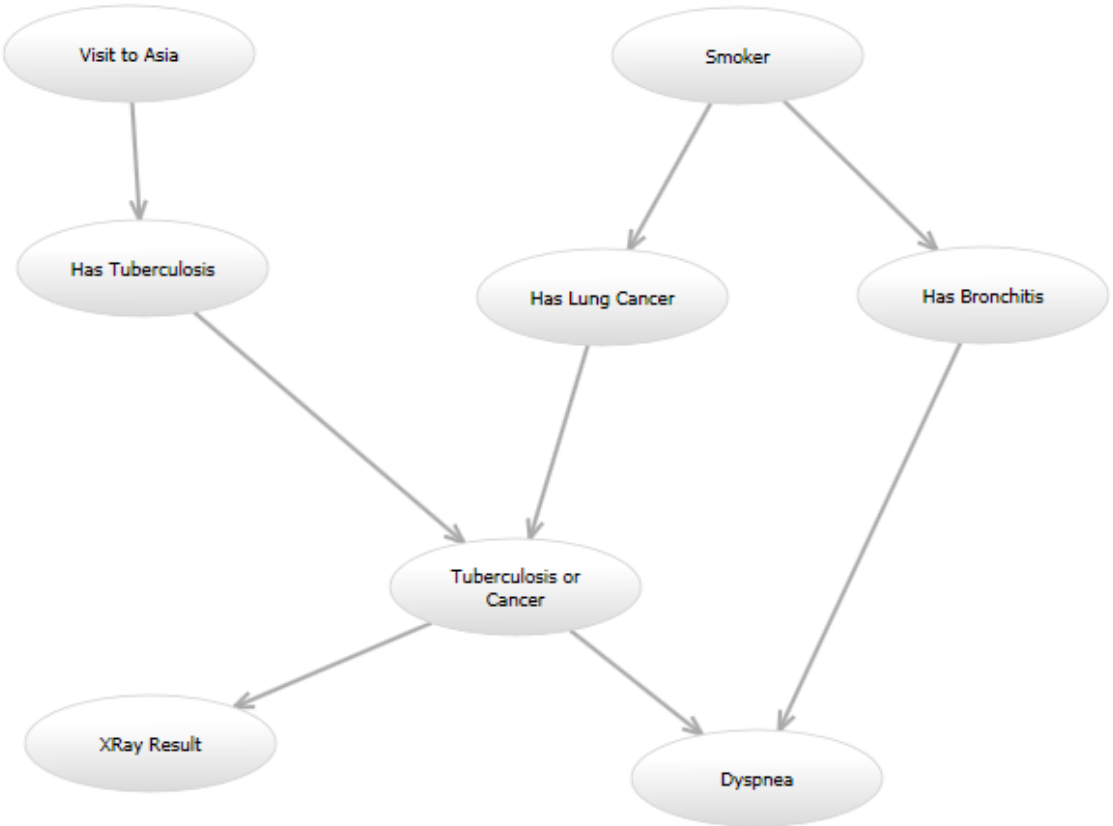
SOA testing automation concerns:

- test case generation
- test run execution and evaluation
- **test campaign dynamic planning (optimization) - reliability computation (Bayesian network inference)**

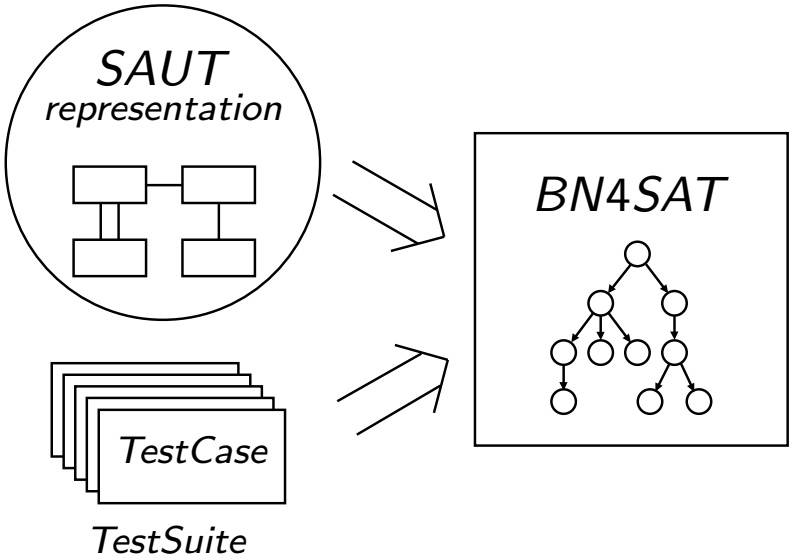


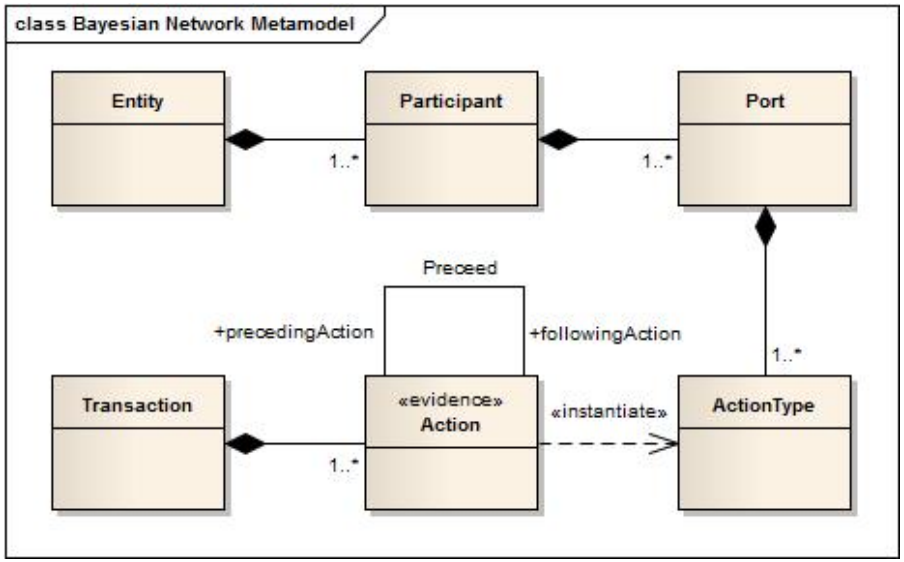
Bayesian network and inference

A Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG) - it can be used to find out updated knowledge of the state of a subset of variables when other variables (the evidence variables) are observed (probabilistic inference)

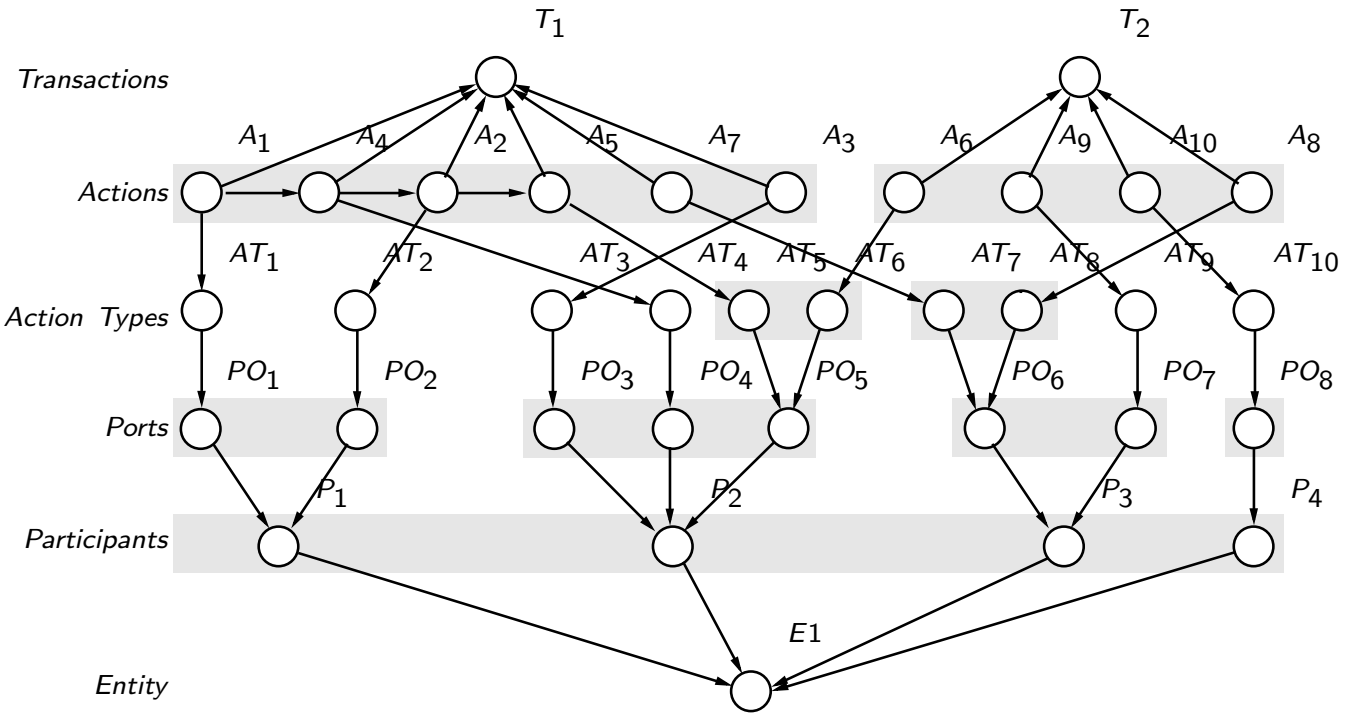


BN4SAT

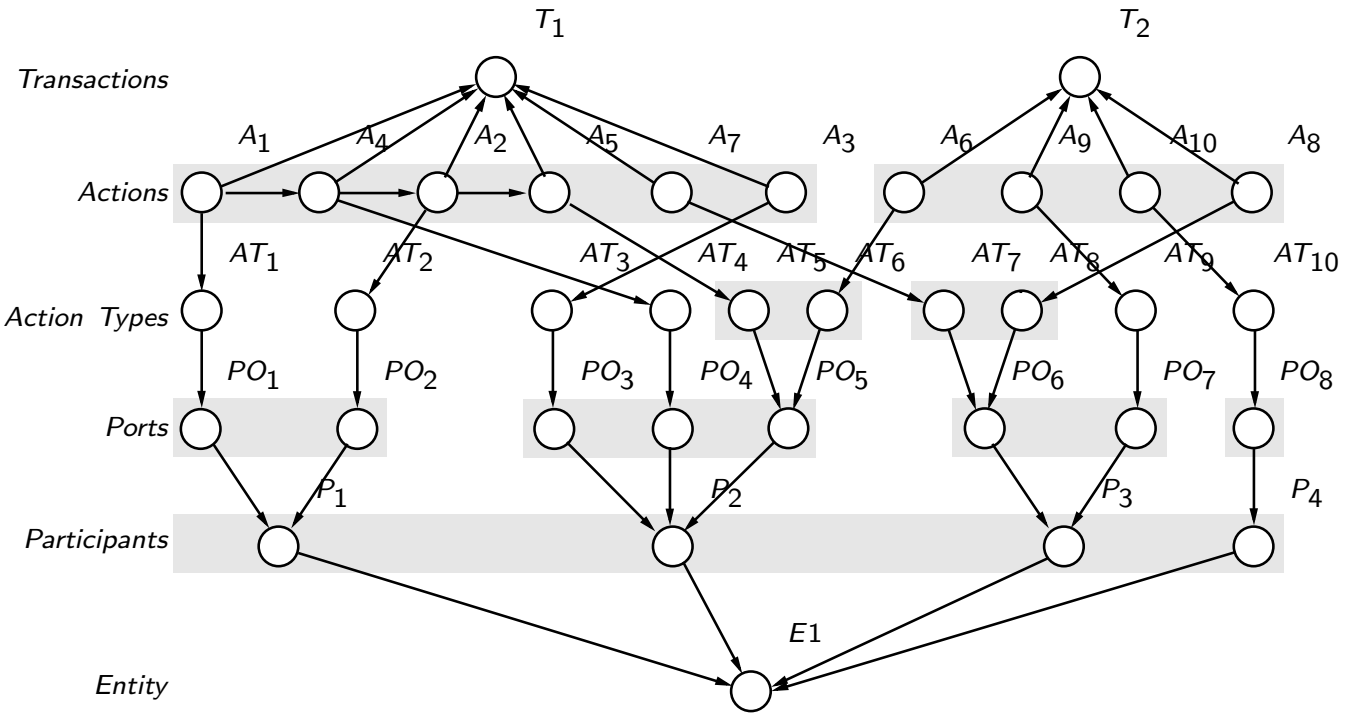

SAUT
representation
TestCase
TestSuite
BN4SAT


class Bayesian Network Metamodel
Entity
Participant
Port
1..*
1..*
Preceed
+precedingAction
+followingAction
1..*
Transaction
«evidence»
Action
«instantiate»
ActionType
1..*

BN4SAT example



BN4SAT example

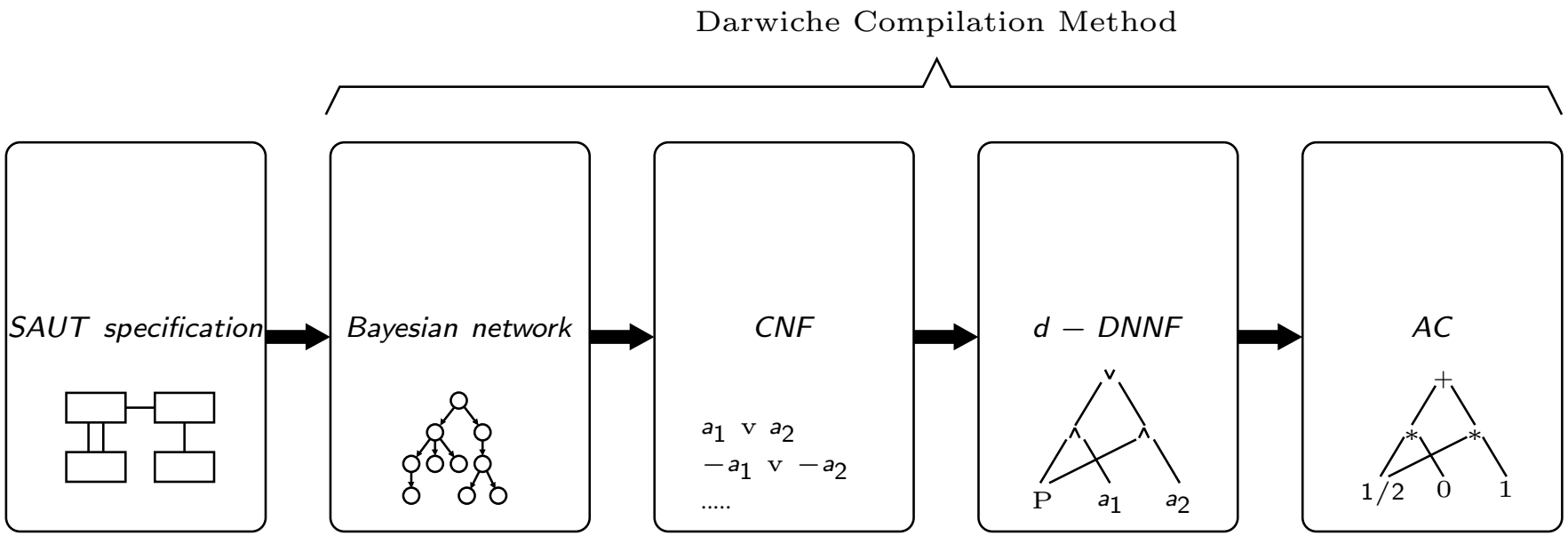


Entity: business organization {notFaulty,faulty}
Participant: architecture participant {notFaulty,faulty}
Port: participant required interface {notFaulty,faulty}
ActionType: type of communicative action {notFaulty,faulty}
Action (evidence): test case action {pass,fail}
Transaction: test case {pass,fail}

Bayesian network inference problem

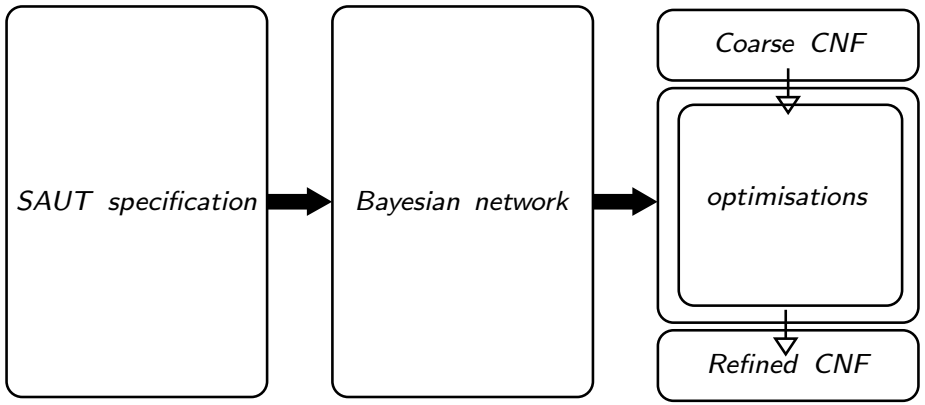
- A BN for SOA testing can exhibit a very large size, and can be very dense (large conditional probability tables)
- Space & time computational complexity,
- Classical BN inference methods (lazy propagation, variable elimination, Shafer-Shenoy,..) are inadequate

Solution - 1st step: Inference by compilation

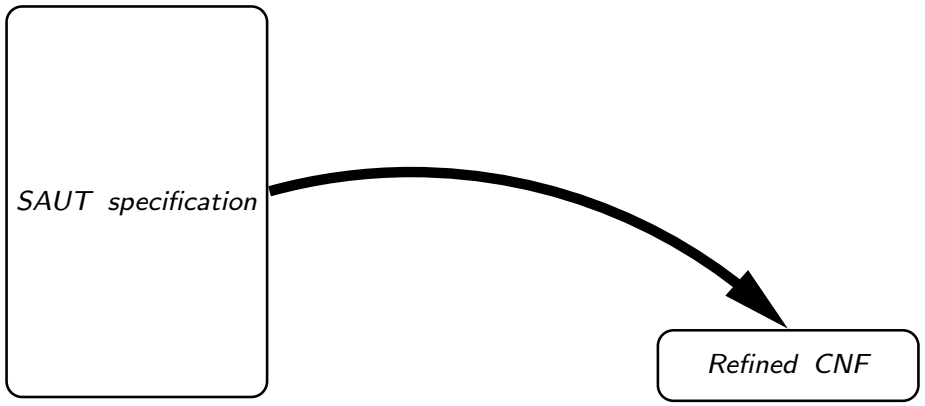


CNF: Conjunctive Normal Form
d-DNNF: deterministic Disjunction Negation Normal Form
AC: Arithmetic Circuit.

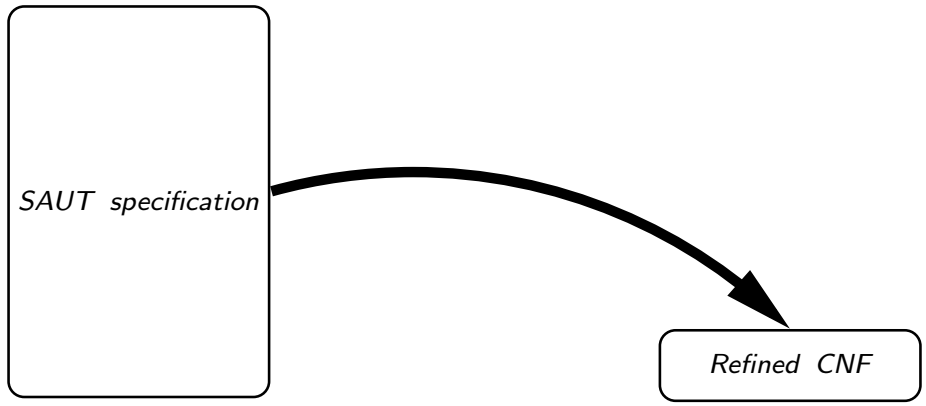
Solution - 2nd step: Model driven compilation method



Solution - 2nd step: Model driven compilation method



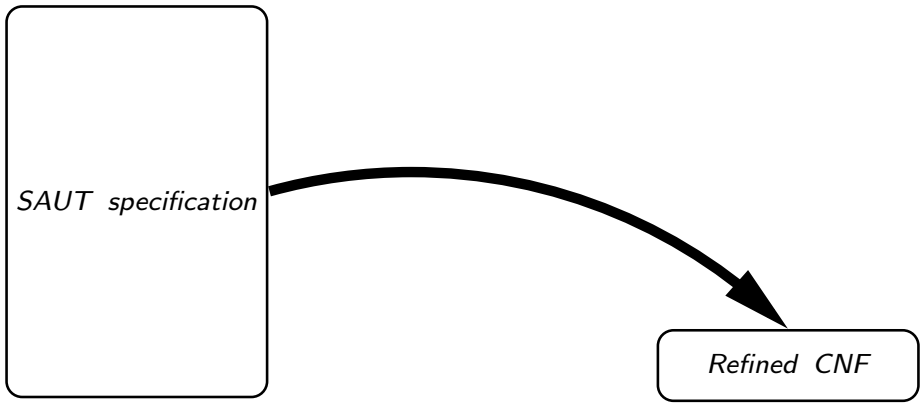
Solution - 2nd step: Model driven compilation method



BN4SAT method

- No intermediary generation of BN
- No CNF optimization step

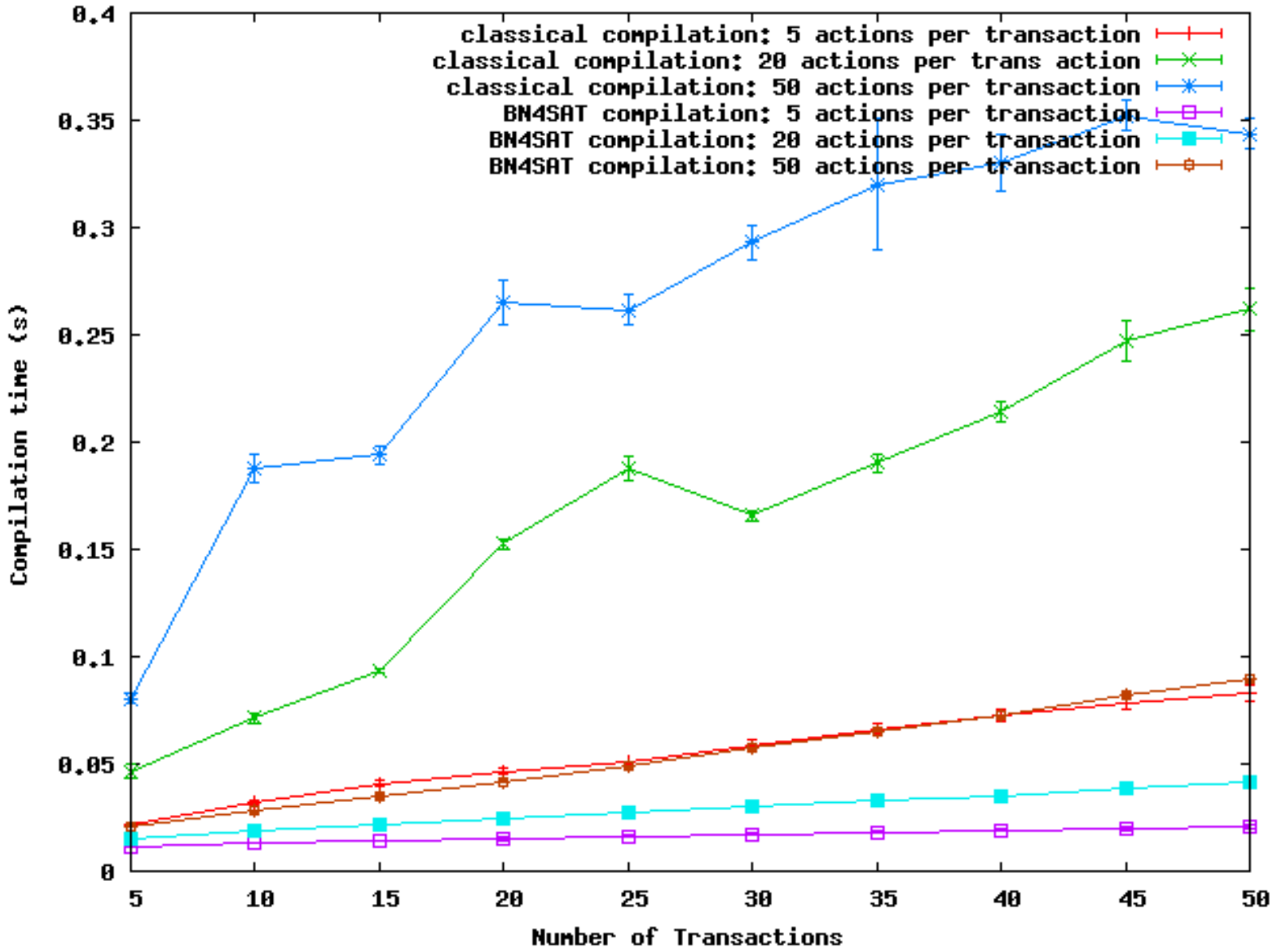
Solution - 2nd step: Model driven compilation method



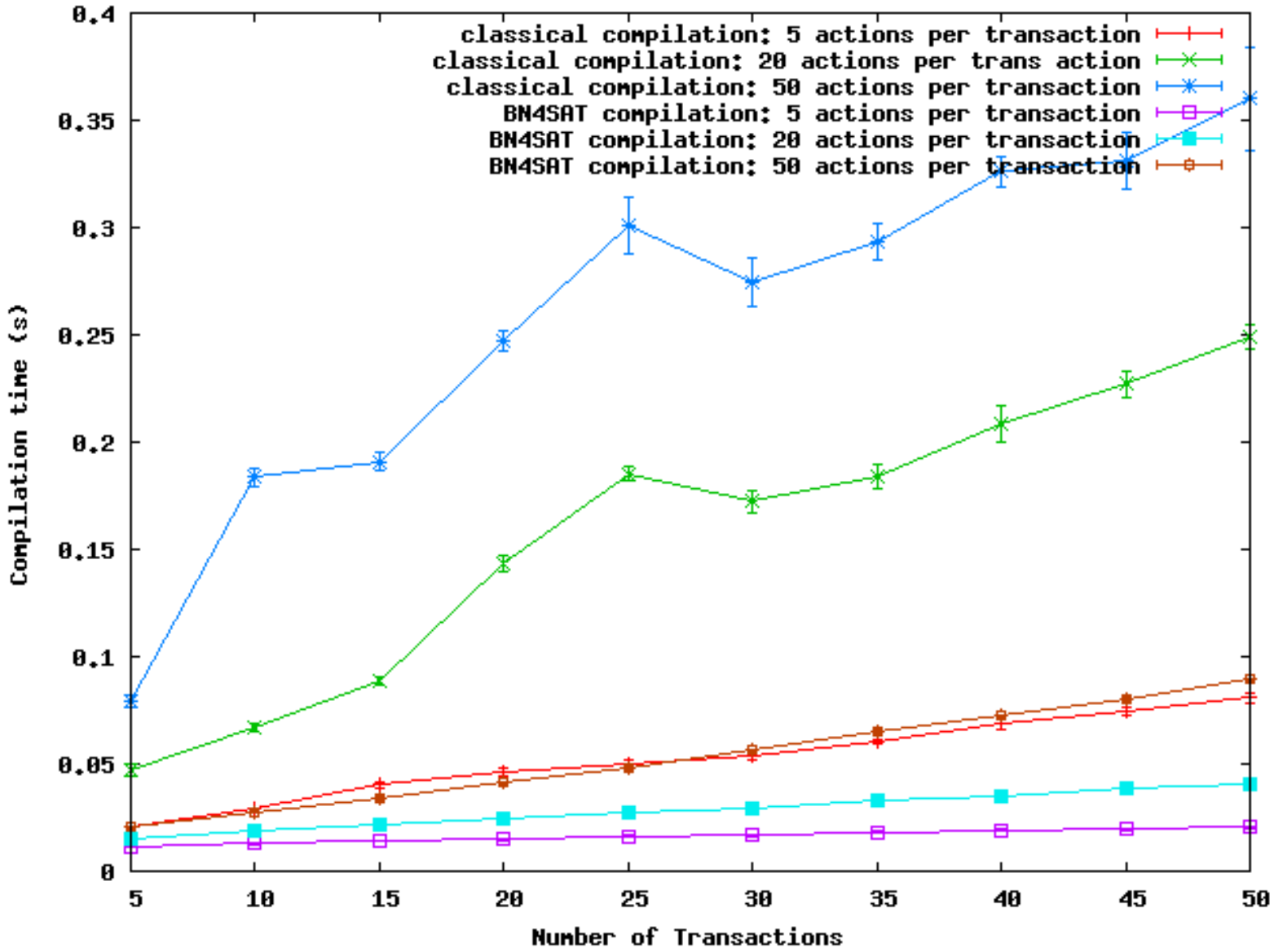
BN4SAT method

- No intermediary generation of BN
- No CNF optimization step
- Builds CNF in linear time
- Push the boundaries concerning size limitation

Experimental results 1



Experimental results 2



Future work

- Evaluation of BN4SAT on realistic services architectures (Healthcare sector)
- Evaluation of the suitability of the developed methods and tools to non functional testing
- Evaluation of alternative models for the opaque (non observable) regions of the services architecture (fault trees)
- Improvement of model driven inference by compilation