Using Standard AADL for COMPASS

Thomas Noll (`noll@cs.rwth-aachen.de`)

AADL Standards Meeting

Aachen, Germany; July 25–28, 2016

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Overview

Introduction

SLIM Language Updates

COMPASS Development Roadmap

Fault Injections

Parametric Error Models

Timed Failure Propagation Graphs

Outline

Introduction

SLIM Language Updates

COMPASS Development Roadmap

Fault Injections

Parametric Error Models

Timed Failure Propagation Graphs

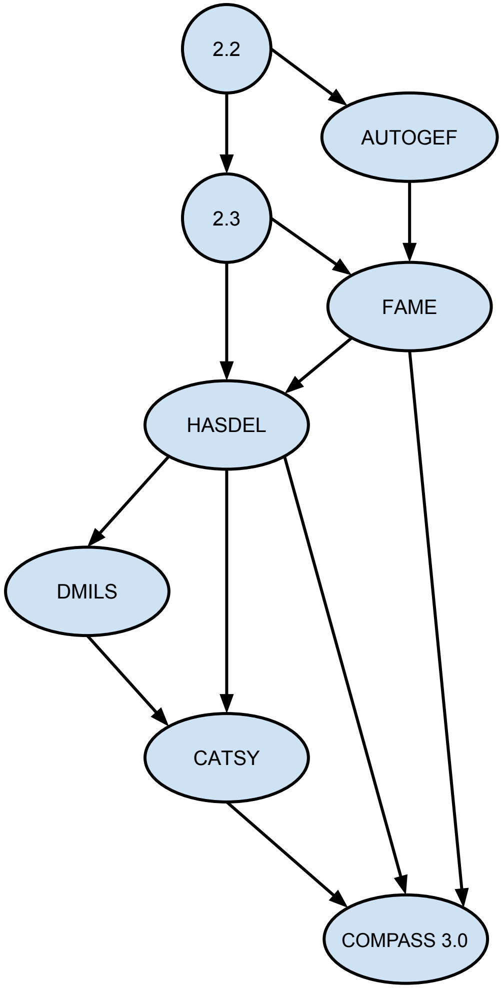
COMPASS Sub-projects and Extensions

On RWTH side:

1. COMPASS started in 2008
2. HASDEL in 2013
3. D-MILS in 2013
4. CATSY in 2015

Other projects (FBK):

1. AUTOGEF (2011)
2. FAME (2012)



Introduction

Issues

Many previous projects and many ideas. This leads to some issues:

- Legacy code
- Outdated tools
- As many code repositories as there are projects

Issues

Many previous projects and many ideas. This leads to some issues:

- Legacy code
- Outdated tools
- As many code repositories as there are projects

To address this, there is now: COMPASS 3

COMPASS 3

Project goals:

- Update SLIM language
- Update tools
- Improve examples and tutorials
- Set up development roadmap (also for ESA)

Introduction

COMPASS 3

Project goals:

- Update SLIM language
- Update tools
- Improve examples and tutorials
- Set up development roadmap (also for ESA)

Outline

Caveat: Properties Everywhere

SLIM properties: formalization of requirements as component annotations

```
system Simple
 features
  input: in event port;
  output: out event port;
  {OCRA: CONTRACT myContract
   assume:  in the future {input};
   guarantee:  always ({input} implies in the future {output});}
end Simple;
```

Caveat: Properties Everywhere

SLIM properties: formalization of requirements as component annotations

```
system Simple
 features
  input: in event port;
  output: out event port;
  {OCRA: CONTRACT myContract
   assume:  in the future {input};
   guarantee:  always ({input} implies in the future {output});}
end Simple;
```

AADL properties: generic mechanism for providing information about elements

```
process implementation MyProcess.Impl
 subcomponents
  t: thread MyThread.Impl;
 properties
  Example::Stack_Size => 2 KB;
end MyProcess.Impl;
```

SLIM Language Changes

Main goal of SLIM update: re-integration with AADL V2 [V3]

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

SLIM Language Changes

Main goal of SLIM update: re-integration with AADL V2 [V3]

Multiple benefits:

- Easier to get industry acceptance: AADL is a known language
- Possible to use existing AADL tooling (e.g., no editor specifically for SLIM)
  - OSATE2 development environment can be used for AADL V2

SLIM Language Changes

Main goal of SLIM update: re-integration with AADL V2 [V3]

Multiple benefits:

- Easier to get industry acceptance: AADL is a known language
- Possible to use existing AADL tooling (e.g., no editor specifically for SLIM)
  - OSATE2 development environment can be used for AADL V2

Approach: use AADL properties mechanism to represent SLIM-specific features

`SLIMdatatypes`: embedding of SLIM data types in AADL

`SLIMpropset`: AADL properties for specifying special-purpose attributes of architectural elements such as ports

`CSSP`: AADL properties for specifying SLIM properties in *Catalogue of System and Software Properties* (cf. Harold's presentation on CATSY project)

SLIM Data Types

SLIM predefines a few basic data types:

- `bool`
- `int`
- `real`
- `enum`
- `clock`
- `continuous`

AADL Data Types

AADL has a data type system that allows the definition of (custom) data types. For example:

Data type package

```
package SLIMdatatypes
  public
  data bool
  end bool;
  data int
  end int;
  data real
  end real;
  ...
end SLIMdatatypes;
```

Also useful for (nested) data types defined by the user.

SLIM Language Updates

AADL Properties for SLIM Attributes

Examples of port attributes:

`Alarm`: port is an alarm for FDIR analysis

`Blocking`: for blocking communication if incoming event (data) port not enabled

`Default`: for defining default values of data elements

`Observable`: for indicating observable parameters (FDIR analysis)

SLIM attributes as AADL properties

```
system sys
 features
  input: in event port
   blocking;
  output: out data port
   int default 1;
end sys;
```

$\rightsquigarrow$

```
system sys
 features
  input: in event port
   {Blocking = true;};
  output: out data port int
   {Default => "1";};
end sys;
```

SLIM Language Updates

Storing Contracts

SLIM contracts as AADL properties

```
system simple
 features input: in event port; output: out event port;
  {OCRA: CONTRACT myContract
   assume:  in the future {input};
   guarantee:  always ({input} implies in the future {output});}
end simple;
```

$\rightsquigarrow$

```
system simple
 features input: in event port; output: out event port;
 properties
  Contracts => [
   {Name => "myContract";
    Assumption => "in the future input";
    Guarantee =>
     "always ({input} implies in the future {output})";}];
end simple;
```

SLIM Language Updates

AADL Property Set for SLIM

SLIM properties

```
property set SLIMpropset is
  Blocking: aadlboolean => true
    applies to (event port, event data port);
  TimeUnit: ClockTimeUnit applies to (data);
  ClockTimeUnit: type enumeration (Milliseconds, Seconds, ...);
  Contracts: list of SLIMpropset::Contract
    applies to (system, process, ...);
  Contract: type record (
    Name: aadlstring;
    Assumption: aadlstring;
    Guarantee: aadlstring;);
  ContractRefinement: type record (
    Contract: aadlstring;
    SubContracts: list of aadlstring;);
  ContractRefinements: list of SLIMpropset::ContractRefinement
   applies to (system, process, ...);
  ...
end SLIMpropset;
```

Outline

COMPASS Development Roadmap

- Deliverable of COMPASS 3 project
- To identify future steps of COMPASS and prepare corresponding development and release plan

COMPASS Development Roadmap

- Deliverable of COMPASS 3 project
- To identify future steps of COMPASS and prepare corresponding development and release plan
- Strategic aims:
  - extend usability and applicability of toolset
  - identify and remove limitations that prevent “market” penetration and industrial usage

  $\Rightarrow$ reach higher levels of technology readiness
- Derived goals:
  - improving system development process
  - advancing COMPASS technology, mainly w.r.t. error modeling and analysis

COMPASS Development Roadmap

Improving the System Development Process

- Main concern: integration of COMPASS modeling, analysis, and validation activities with design and implementation steps supported by other (AADL) tools (TASTE, Simulink, ...)
- Requires methods for checking conformance between hardware/software implementation and AADL model: model-based testing
  - steer automated generation of test cases by AADL model

COMPASS Development Roadmap

Advancing Error Modeling and Analysis

- More expressive fault injections
- Parametric error models
  - parameter synthesis
  - model repair
- Timed failure propagation graphs
- Formal validation of requirement specifications
- Contract-based design
- More efficient analysis of dynamic fault trees

COMPASS Development Roadmap

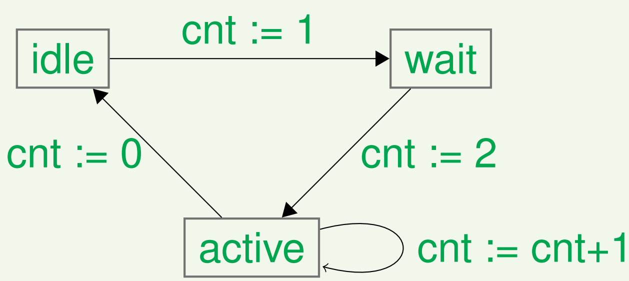
Advancing Error Modeling and Analysis

- More expressive fault injections
- Parametric error models
  - parameter synthesis
  - model repair
- Timed failure propagation graphs
- Formal validation of requirement specifications
- Contract-based design
- More efficient analysis of dynamic fault trees

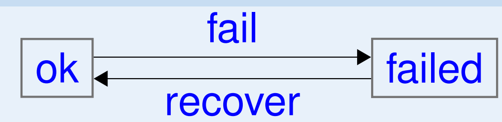
Outline

SLIM Error Modeling

Nominal behavior



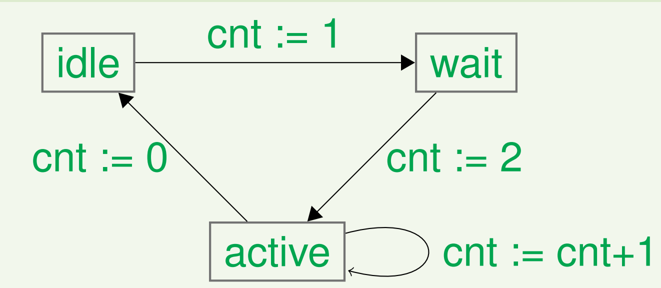
Error behaviour



Fault Injections

SLIM Error Modeling

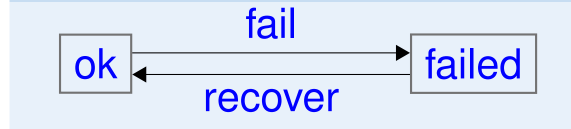
Nominal behavior



Fault injection

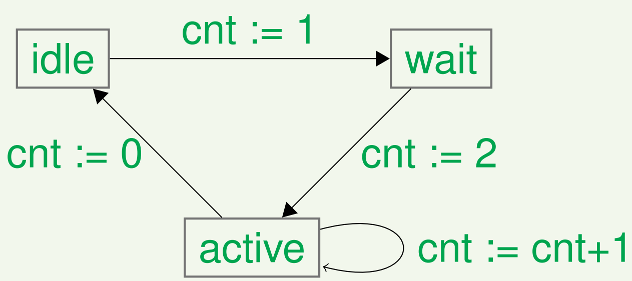
failed: cnt := -1

Error behaviour



SLIM Error Modeling

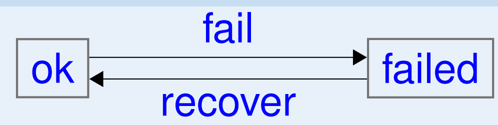
Nominal behavior



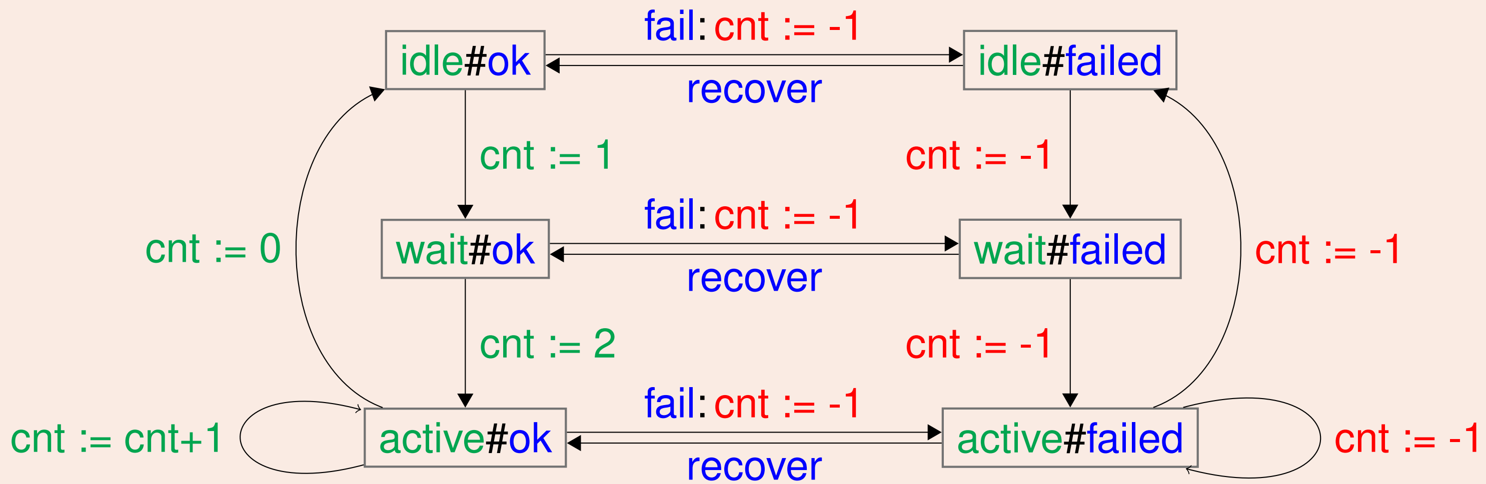
Fault injection

failed: cnt := -1

Error behaviour



Extended model



Fault Injections

Current Fault Injection Mechanisms

`SLIMpropset` properties for fault injection

```
-- Modify nominal data value while in error state
FaultEffect: type record (
  State: aadlstring;
  Target: reference(data port, data);
  Effect: SlimExpr;);

-- Restrict possible modes while in error state
ForcedMode: type record (
  State: aadlstring;
  Modes: list of reference(mode););

-- Disable event ports while in error state
Inhibit: type record (
  State: aadlstring;
  Ports: list of reference(event port, event data port););
```

Possible Enhancements of Fault Injection

Possible enhancements to increase expressiveness and usability

- Transient fault effects
  - only when *entering* state s, the assignment $d := e$ is performed
- Error-triggered nominal transitions
  - error transition $s \rightarrow s'$ causes mode transition $m \rightarrow m'$

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Outline

Introduction

SLIM Language Updates

COMPASS Development Roadmap

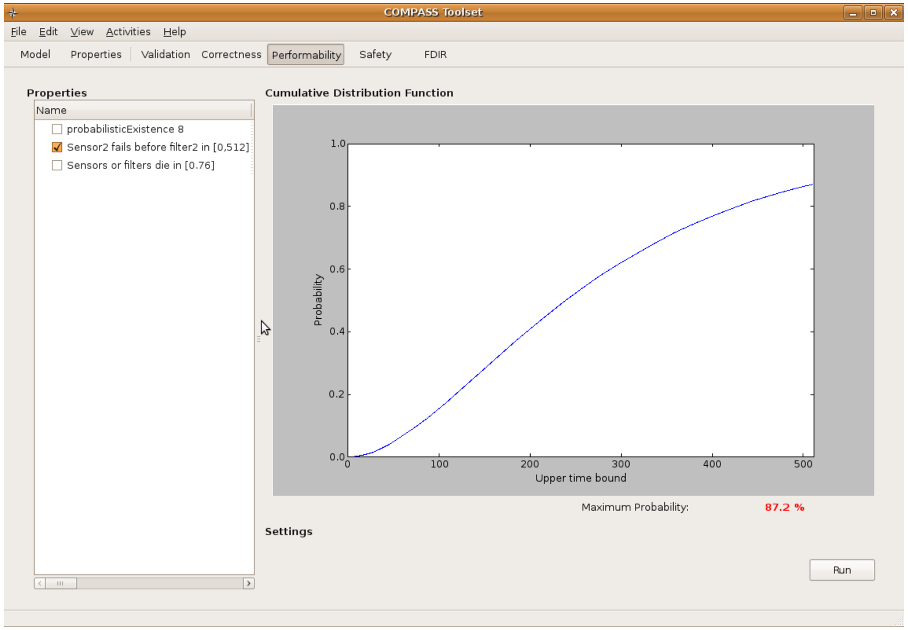
Fault Injections

Parametric Error Models

Timed Failure Propagation Graphs

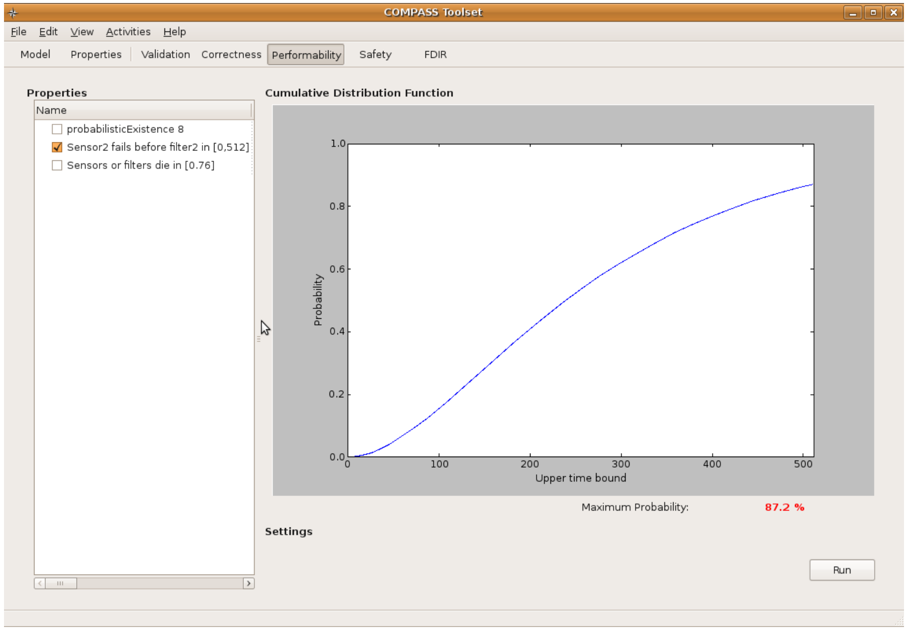
Parametric Error Models

- Performability evaluation: determine likelihood of system failure within given deadline



Parametric Error Models

- Performability evaluation: determine likelihood of system failure within given deadline



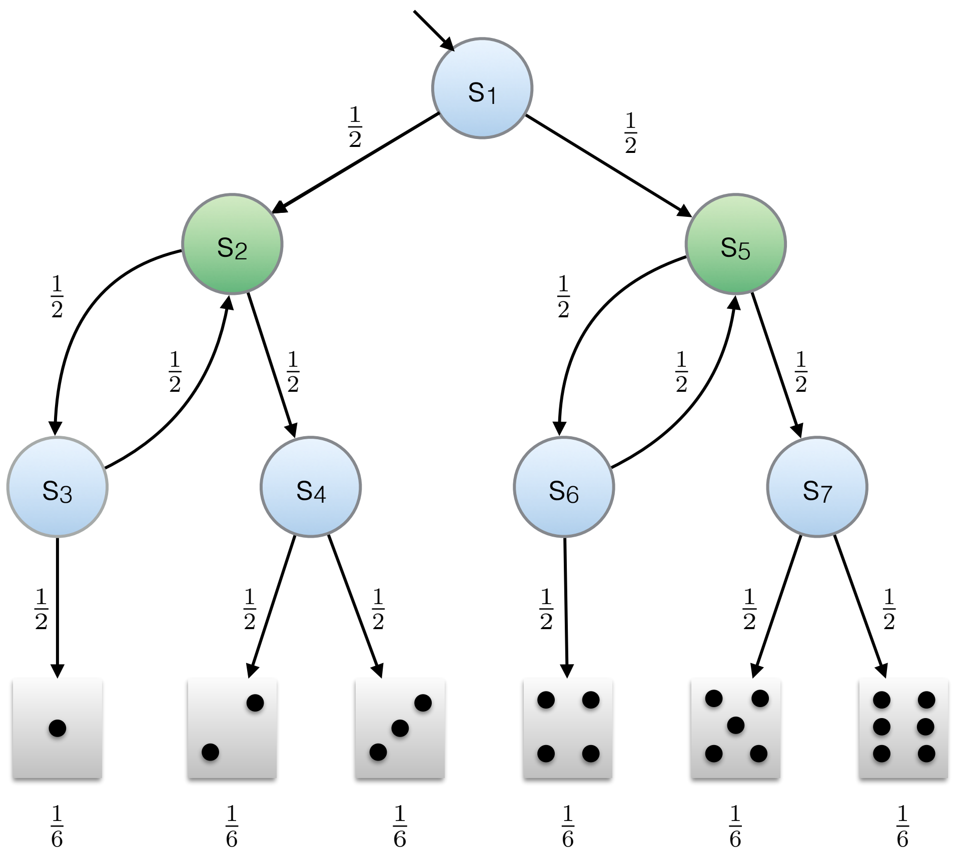
- But: probabilities of basic faults often not (exactly) known

$\Longrightarrow$ Parametric error models

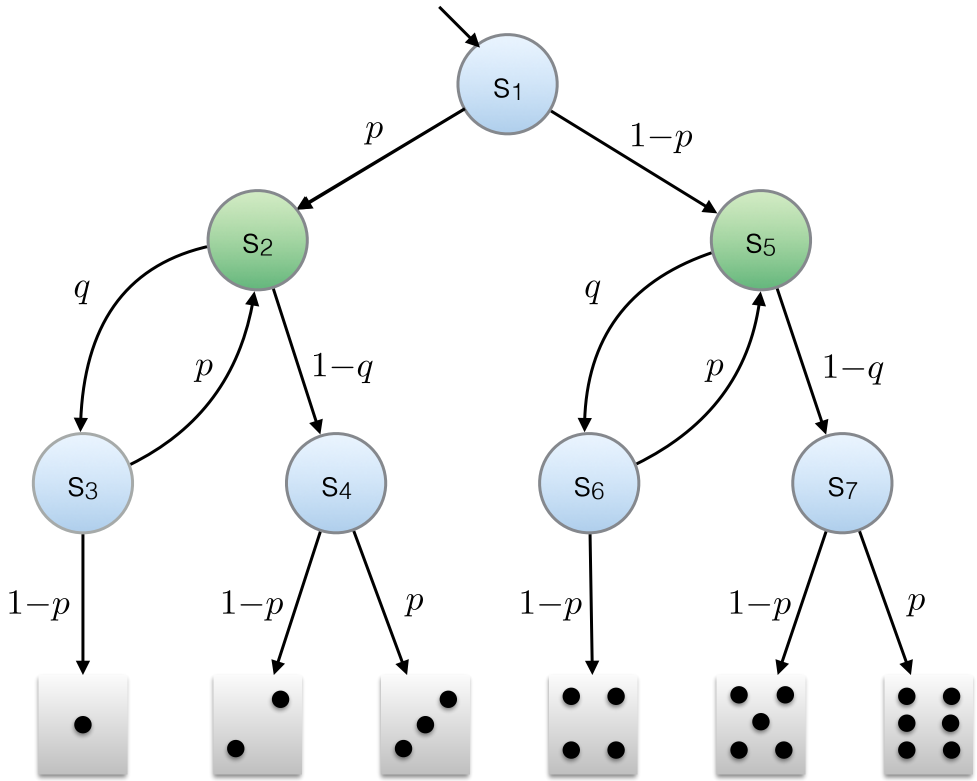
- automatically compute maximal tolerable fault probabilities

Parametric Error Models

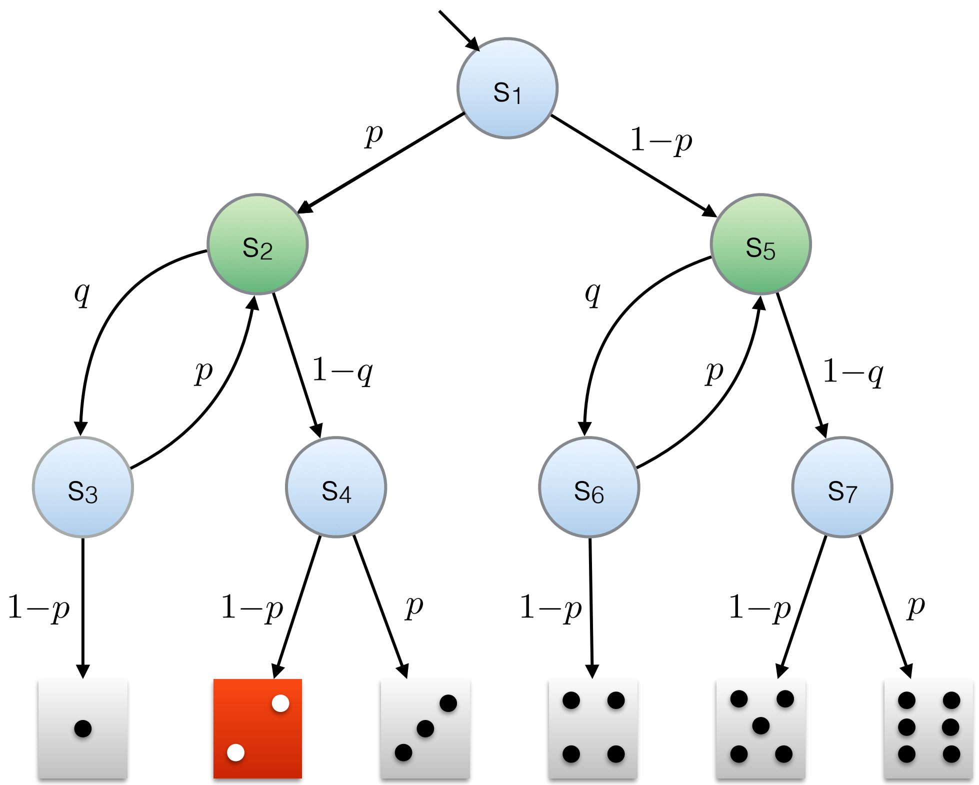
Knuth-Yao Die



Knuth-Yao Die



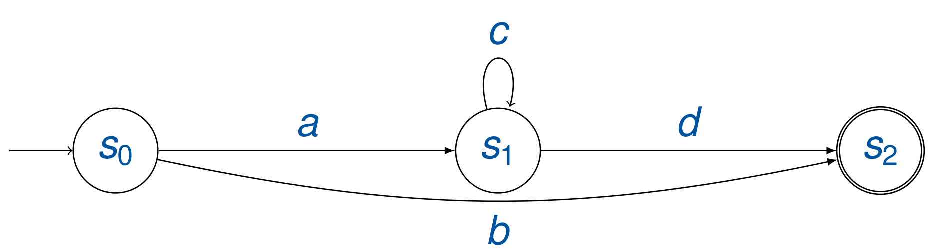
Knuth-Yao Die



PARAMETRIC REACHABILITY PROBABILITY

State elimination

To calculate the regular expression which represents $\mathcal{L}(s_0, \mathcal{A})$ state elimination is used



$\Rightarrow$ Eliminate state s_1:

$\Rightarrow$ We receive the accepted Language $\mathcal{L}(s_0, \mathcal{A}) = a.c^{\star}.d \mid b$

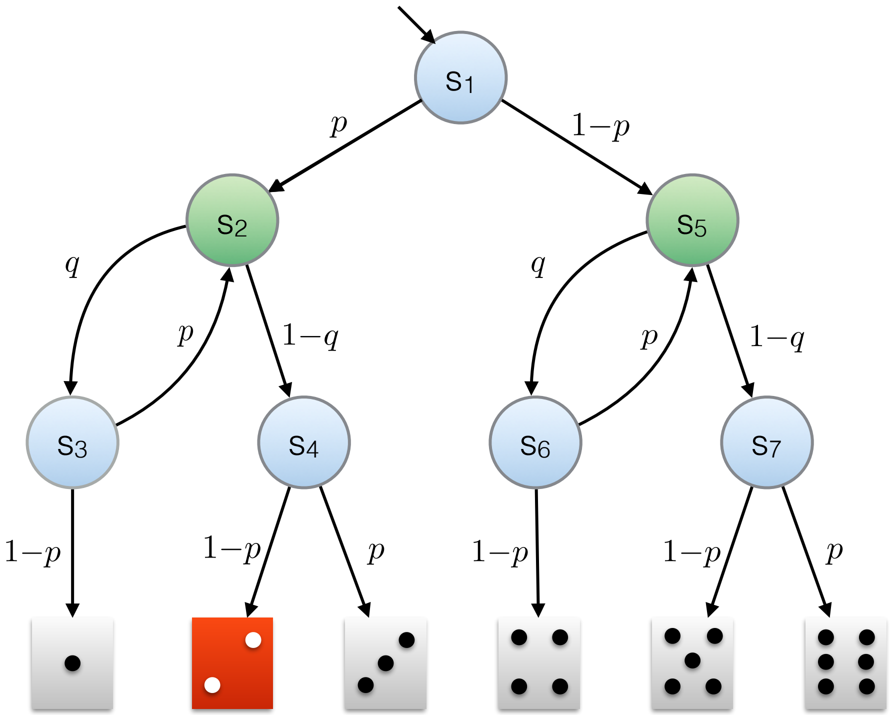
From Languages to Probabilities

Probabilistic interpretation of regular operations

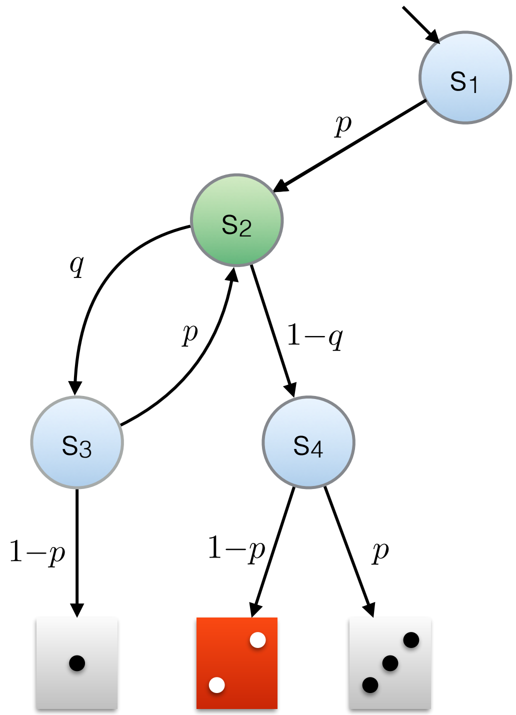
- Concatenation . corresponds to multiplication
- Union | corresponds to addition
- Kleene star $x^* = \varepsilon \mid x \mid xx \mid \ldots$ corresponds to limit of geometric series $\sum_{k\in\mathbb{N}} x^k = \frac{1}{1-x}$

Parametric Error Models

State Elimination

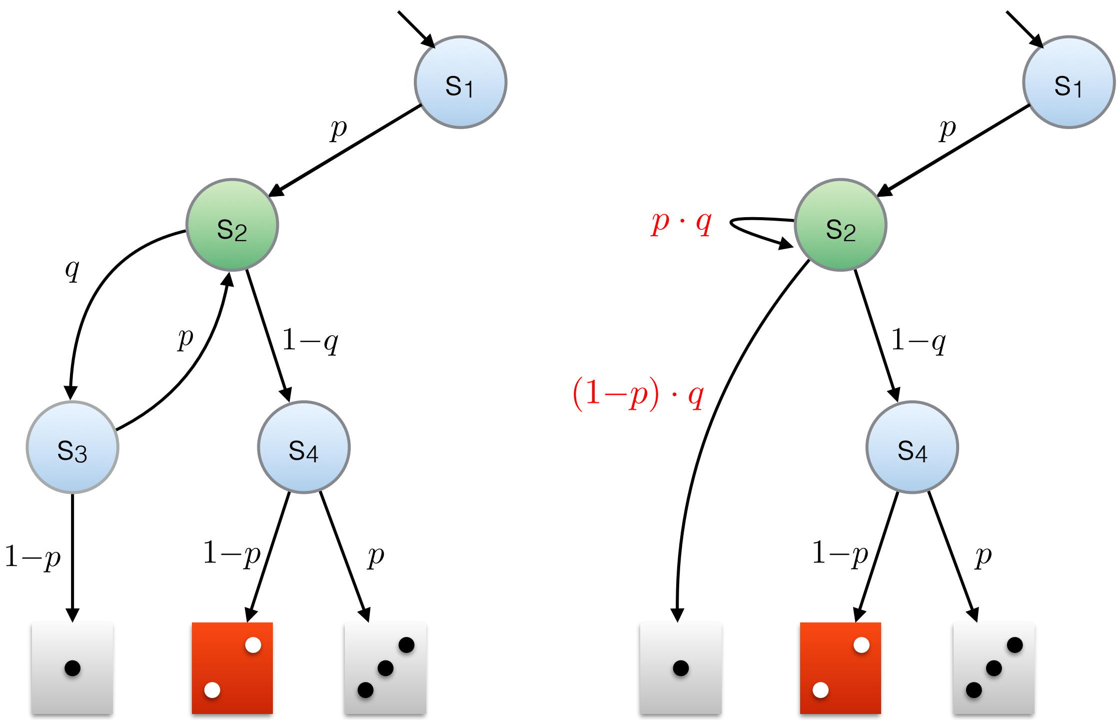


State Elimination



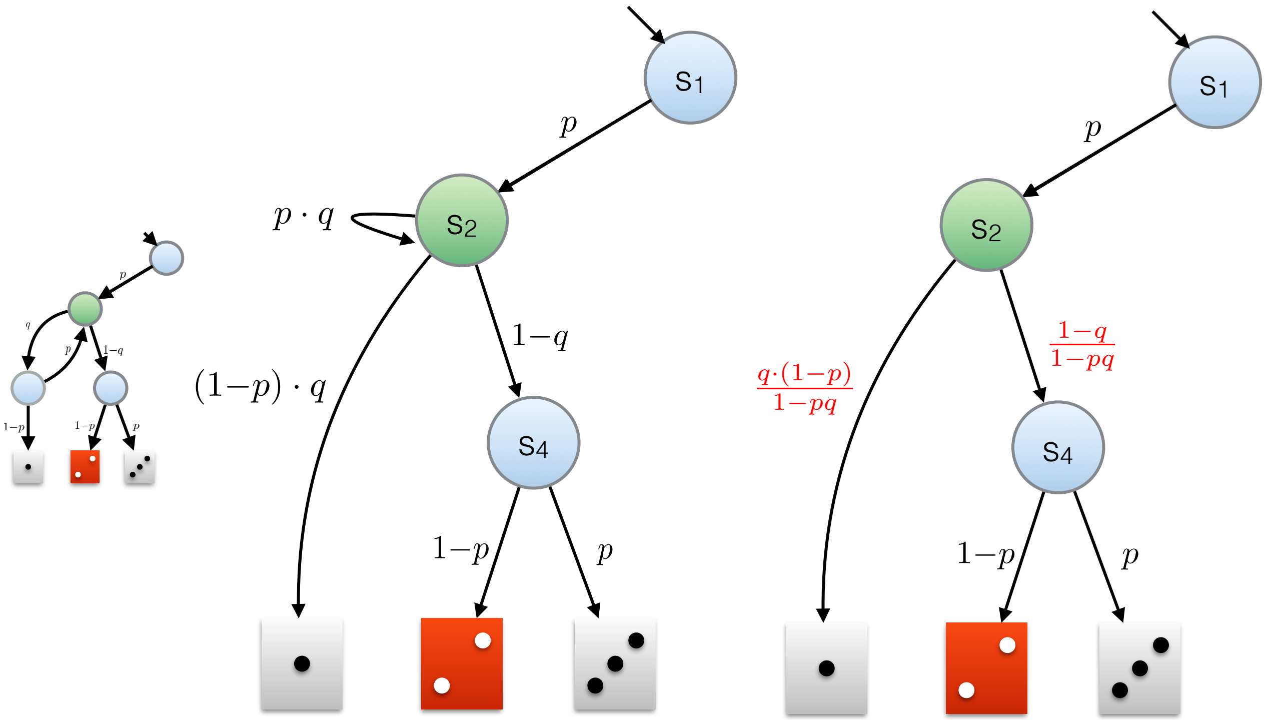
Parametric Error Models

State Elimination

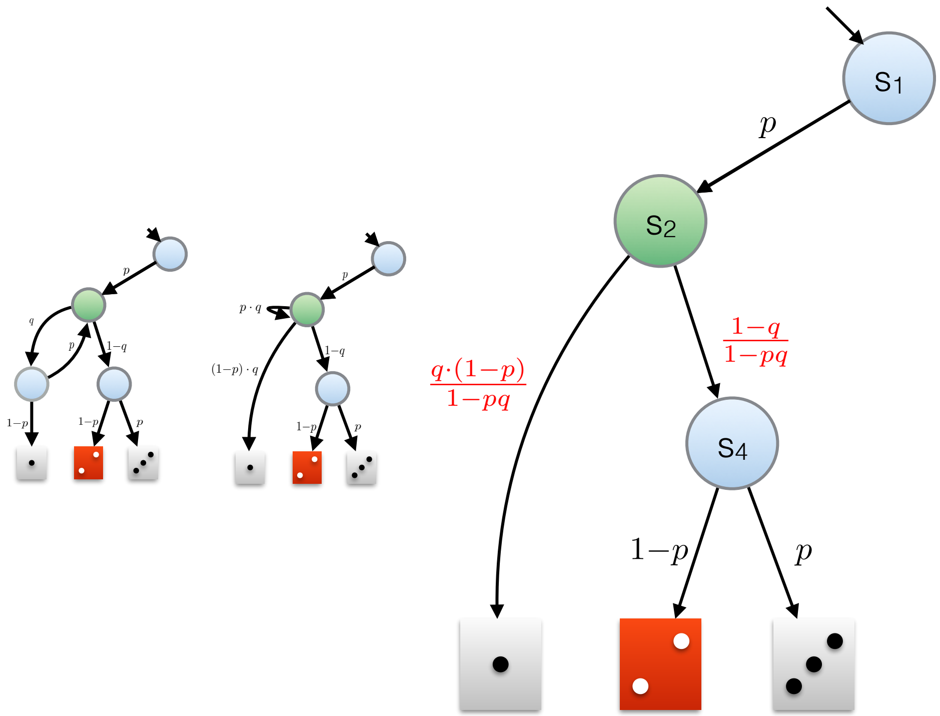


Parametric Error Models

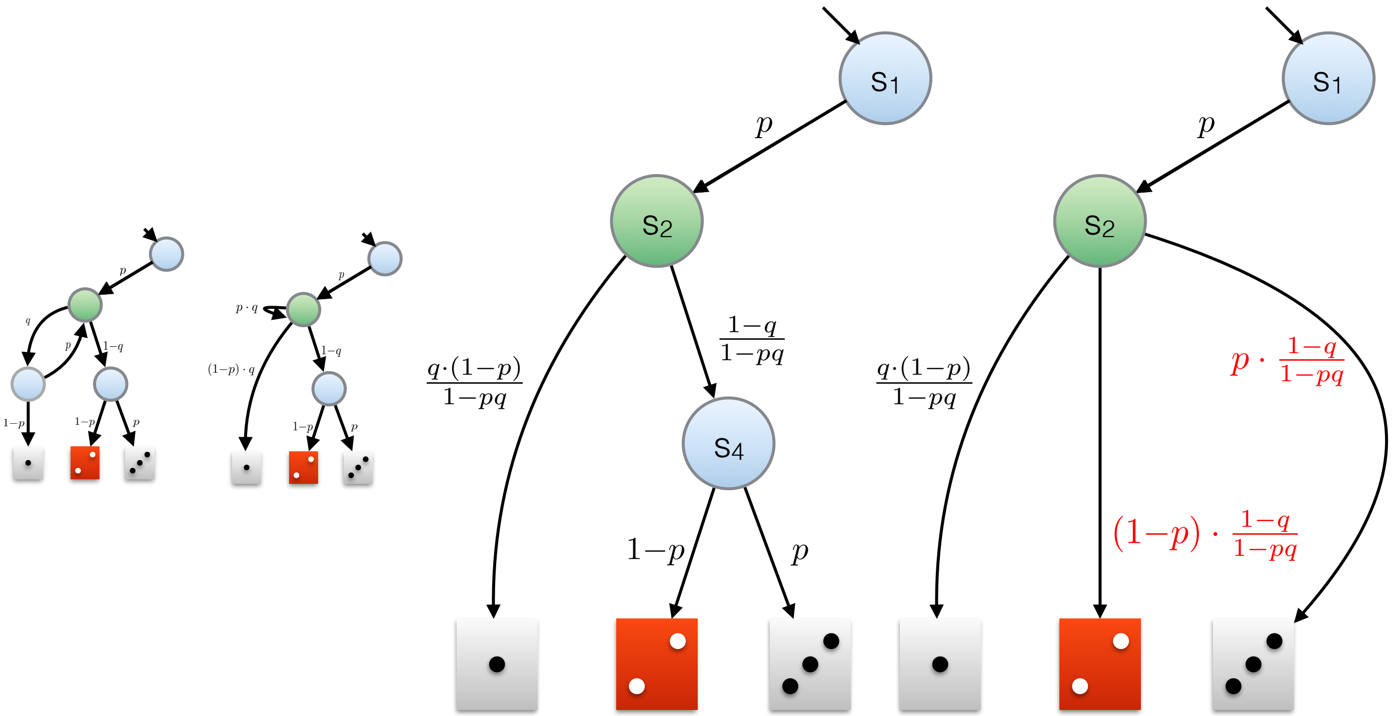
State Elimination



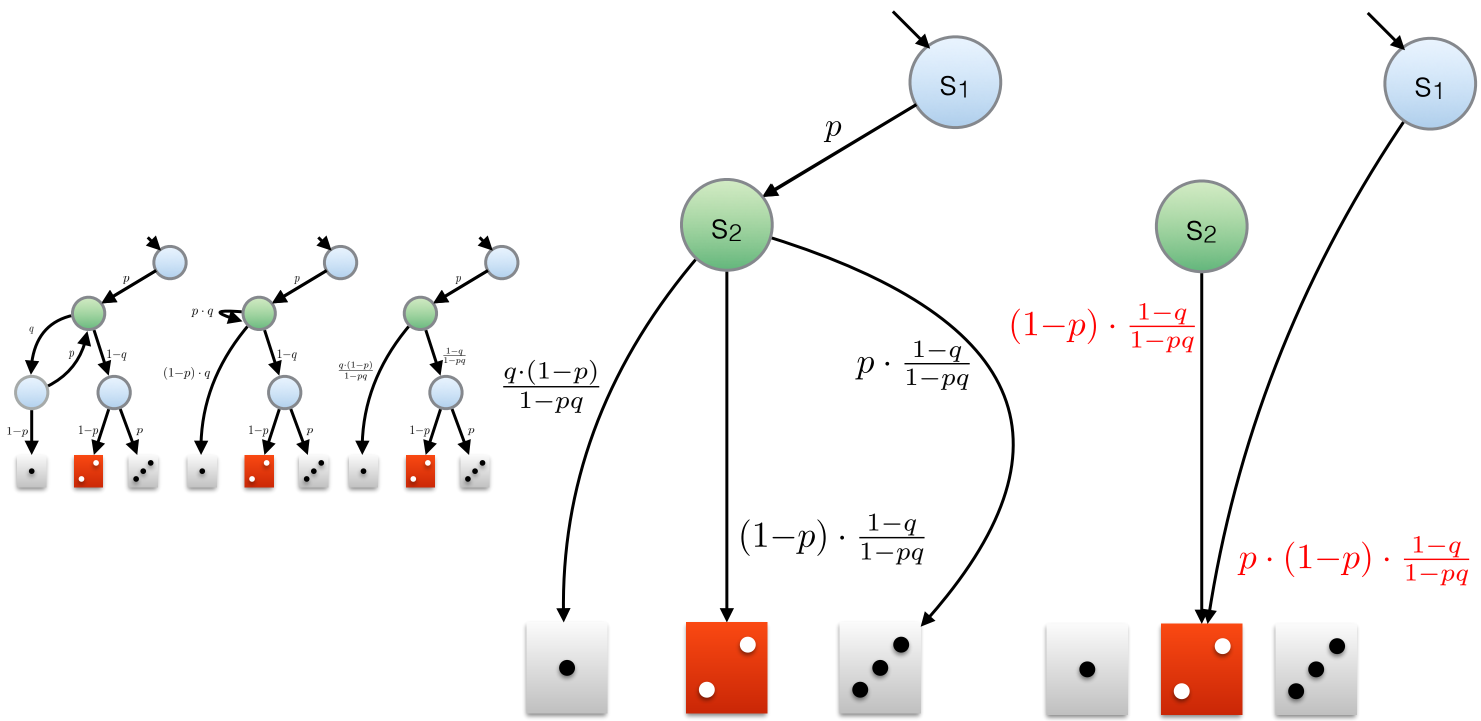
State Elimination



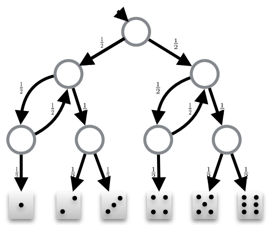
State Elimination

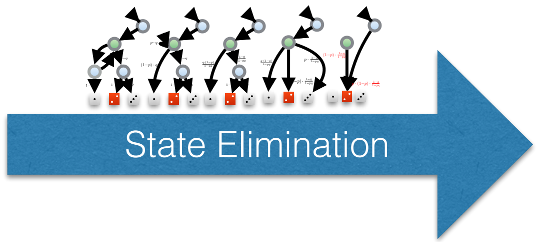


State Elimination



Result of State Elimination





?

Rational function over the parameters

Numerically instable

large

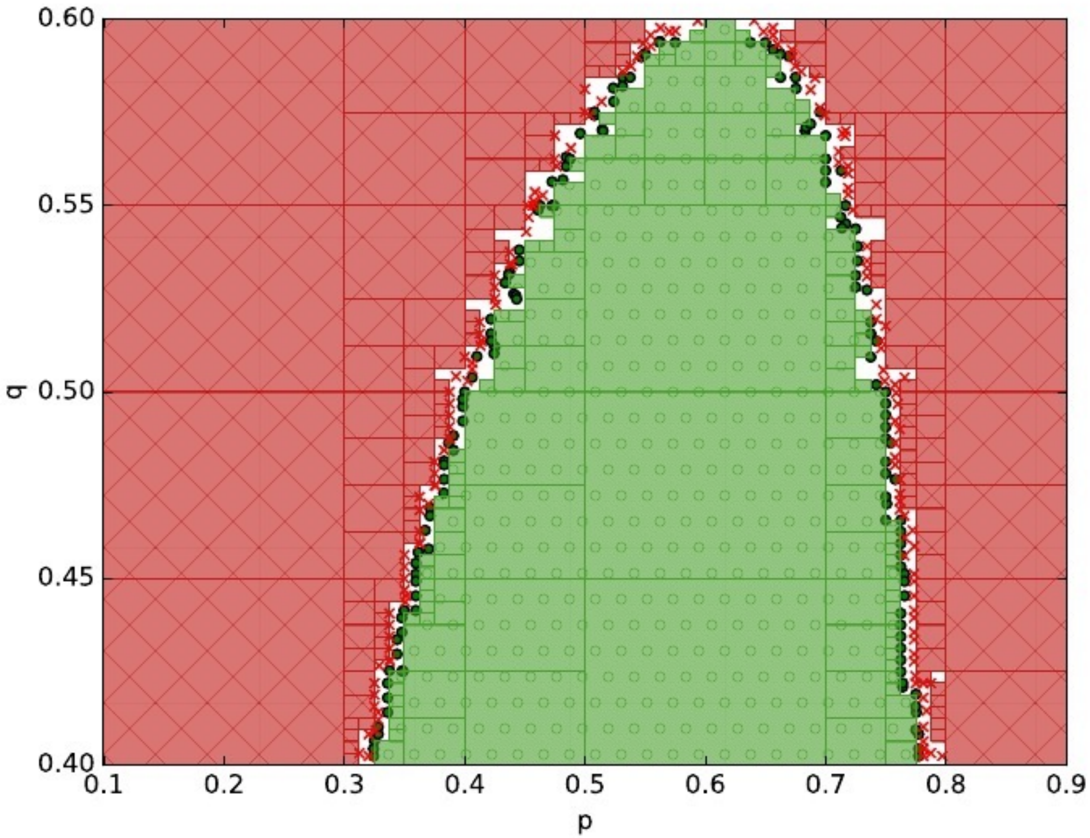
Used by:

[Bartocci et al., 2013]

[Calinescu et al., 2016]

[Rosenblum et al., 2016]

Parameter Space Partitioning

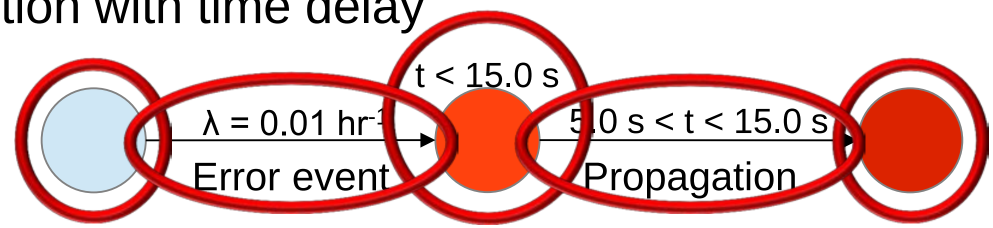


Outline

SLIM: Timed Failure models

An example: modelling error propagation

- First transition with probabilistic rate
- Next transition with time delay




```
error model SubError
features
  err_prop : out error propagation;
end SubError;

error model implementation SubError.Impl
events
  err_evt : error event occurrence poisson 0.01 per hour;
states
  e_nominal : initial state;
  e_triggered : error state urgent in 15 sec;
  e_propagated : error state;
transitions
  e_nominal -[err_evt]-> e_triggered;
  e_triggered -[err_prop between 5 sec and 15 sec]-> e_propagated;
end SubError.Impl;
```

Timed failure propagation graphs (TFPG)