PRISM and CTMC

Mario Binder
WS 2017/2018

Outline

- Introduction
- CTMC
- PRISM
- PRISM and CTMC
- Summary

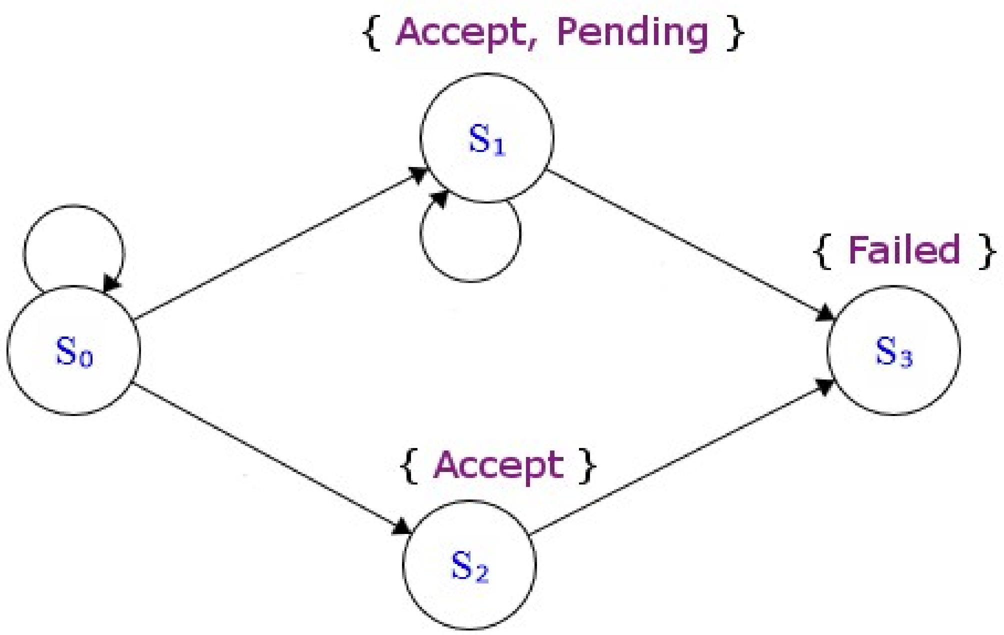
Introduction

- A model checker is a program that decides if a set of given **properties** hold in every state of a **model**
- A model could be anything. It could be a simple FSM or an electronic circuit
- A property is usually described with a set of logical operators and variables. Every property has a **truth value**
- There are also different languages for describing properties. A few examples:
  - CTL
  - LTL
  - PCTL

Computational Tree Logic

- **Computational Tree Logic(CTL)** has all operators from predicate logic plus two quantifiers and a few temporal operators. It operates on **paths**
- A path or trace is a sequence of states, representing a execution of a model
- Additional Quantifiers:
  - $A\phi$: ϕ has to hold in all subsequent paths
  - $E\phi$: there exists one or more paths where ϕ holds
- Temporal Operators:
  - Next (X)
  - Globally(G) and Finally(F)
  - Until(U) and Weak Until(W)

Example for a model(Kripke Structure):



Example properties in CTL:

- (AX ***Accept***)(S_0) → False
- (EF ***Failed***)(S_1) → True
- (AG ***Pending***)(S_2) → False
- (A (***Accept*** U ***Failed***))(S_1) → True

Stochastic Model Checking

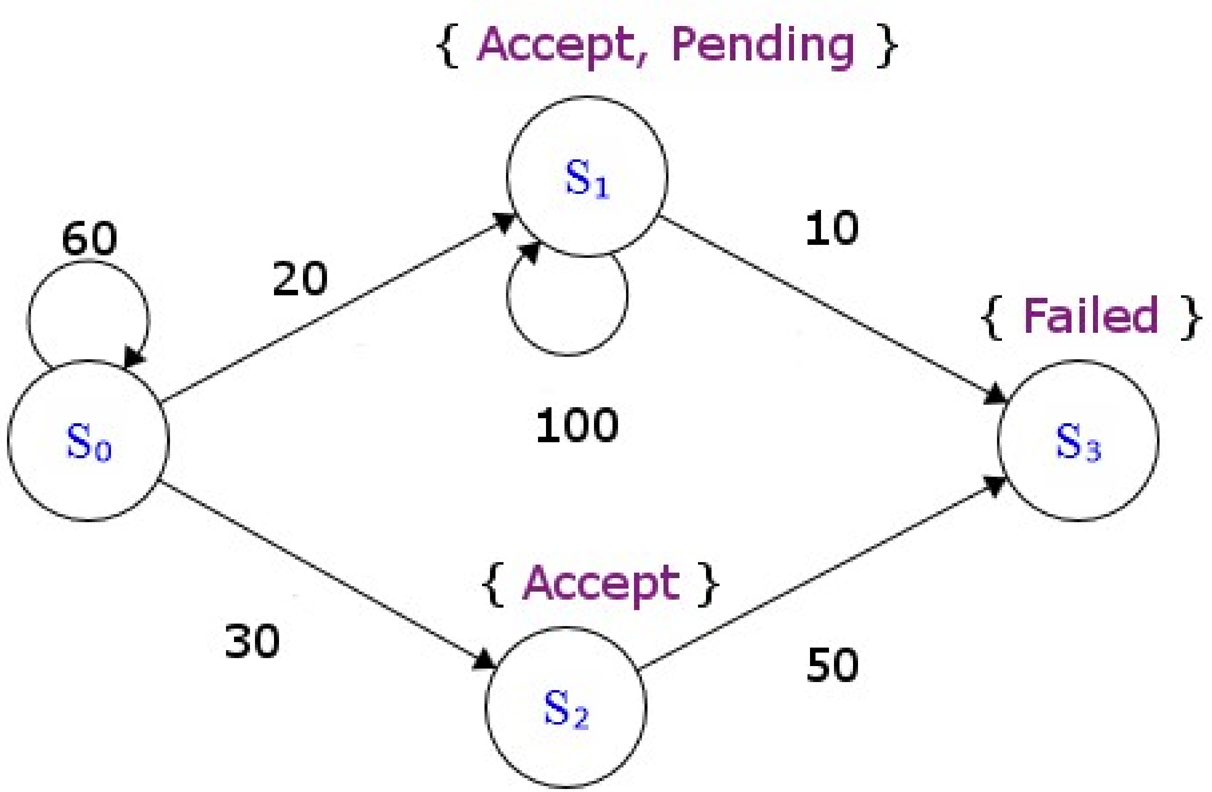
- **Stochastic** model checking is a part of probabilistic model checking, meaning that the model has probabilistic behaviour
- In addition to check if a model satisfies some properties, it also can determine the **likelihood** of reaching a specific state
- Examples for model types, where we can describe probabilistic behaviour:
  - **Discrete-time Markov chains**
  - **Continuous-time Markov chains**
- Furthermore, the property language has to be probabilistic as well

Outline

- Introduction
- CTMC
- PRISM
- PRISM and CTMC
- Summary

Continuous-time Markov chain

- Continuous-time Markov chains(ctmc) are essentially Kripke structures with transition **firing rates**
- Example from before:



- A firing rate determines how often per time interval **t**(e.g. seconds) the transition is made

- The probability of a transition to be triggered is $1 - e^{-R(s,s')*t}$ where R(s, s') is the rate

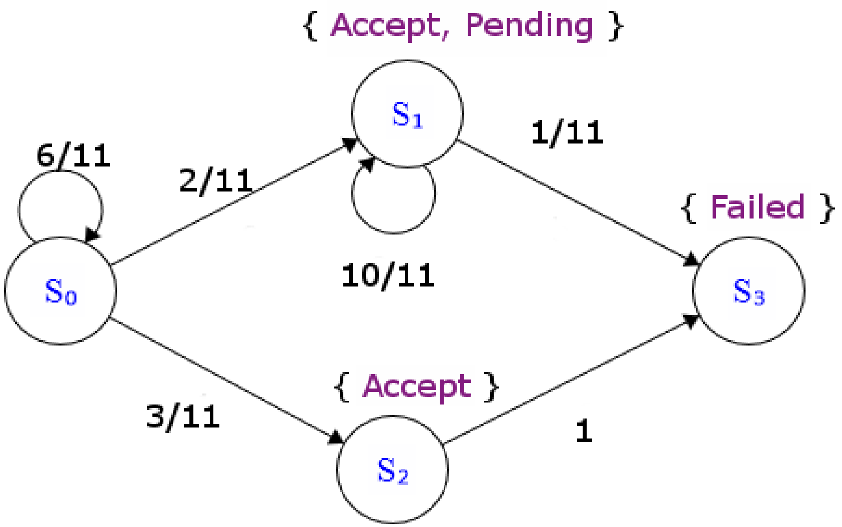
- E.g. R($\mathbf{S_0}$, $\mathbf{S_1}$) in our example was 20. The probability of this transition to be triggered within one second is therefore $1 - e^{-20}$

- If there is more than one transition with R(s, s') > 0 → the next state is decided by a **race condition**

- This means that the first transition that is triggered determines the next state

CTMC and Probabilities

- We can also calculate the probabilities of the next states, producing another DTMC
- The **total rate** or **exit rate** E(s) of a state is the sum of all transition rates
- The probability of a transition is then $\frac{R(s,s')}{E(s)}$
- Example from before:



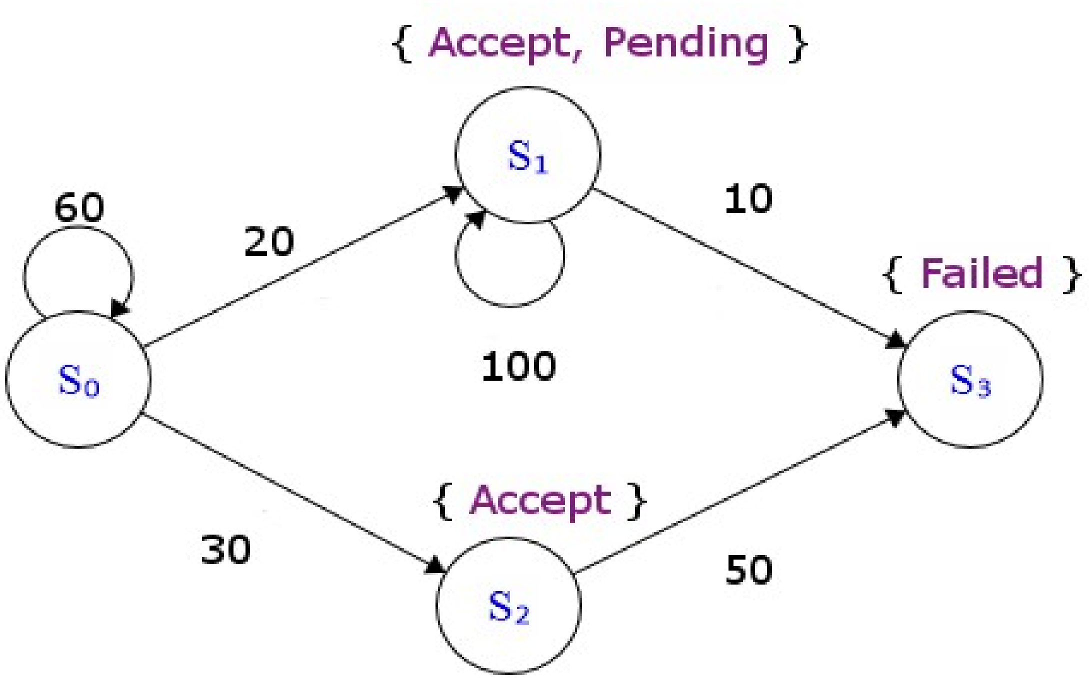
Continuous Stochastic Logic

- **Continuous Stochastic Logic(CSL)** is an extension of CTL
- Removed:
  - All **temporal logic operators**(Global, Finally, etc.) except Next(X)
  - **Quantifiers**
- Added:
  - Two probabilistic operators → see **next slide**
  - **Time-bounded Until**(e.g. a $U^{[0.3,\ 1.5]}$ b)
  - All other temporal logic operators can be derived from the bounded until and are consequently also **time-bounded**

Transient-State and Steady-State Behaviour

- To understand the two probabilistic operators, we first have to cover **transient-state** and **steady-state behaviour**
- Transient-state behaviour:
  - Behaviour/State at instant **t**
  - E.g. ($F^{[10, 10]}$ ***Accept***) → **t** = 10
  - Probability-Operator P: $P_{>0.8}$($F^{[10, 10]}$ ***Accept***)
  - Can be used for bounds too: $P_{>0.8}$($F^{[0, 10]}$ ***Accept***)
- Steady-state behaviour:
  - Behaviour/State at **t** → **∞**
  - Steady-state operator S: $S_{<0.2}$(***Accept***)

Example from before:



Example properties in CSL:

- $(P_{>0.3}(\ S_0 \vee \textit{Pending}\ U^{[0,\ 0.2]}\ \textit{Failed})) \rightarrow$ true
- $(P_{>0.8}(\ S_0 \vee \textit{Pending}\ U^{[0,\ 0.2]}\ \textit{Failed})) \rightarrow$ false
- $(S_{>0.99}(\textit{Failed})) \rightarrow$ true
- $(P_{<0.8}(F^{[0,\ 0.1]}\ \textit{Failed})) \rightarrow$ true

How do we know this? → ***PRISM***

Outline

- Introduction
- CTMC
- PRISM
- PRISM and CTMC
- Summary

PRISM

- **PRISM** is a free open-source probabilistic model checker released under GPL
- Current version is 4.4
- Besides checking, it is also a tool to describe models and properties(with GUI)
- Additionally, it is capable of simulating models using a discrete-event simulation engine

A Quick Introduction to Syntax

- Models are described using the **PRISM Language**
- There are several types of models:
    - Markov decision processes(**mdp**)
    - discrete-time Markov chains(**dtmc**)
    - continuous-time Markov chains(**ctmc**)
    - probabilistic timed automata(**pta**)

- Properties are defined in the **PRISM Property Specification Language**

PRISM Language

- A model in PRISM consists of **modules**
- Each module has **variables** and **commands**
- Variables can have a standard data type, as seen in many other programming languages(**int**, **bool** and **double**) but may have to be **bounded**
- Commands are comprised of **guards** and **updates**
- A guard is a condition of the local variables when the update should be executed
- An update is a transition function

Example from the PRISM manual:

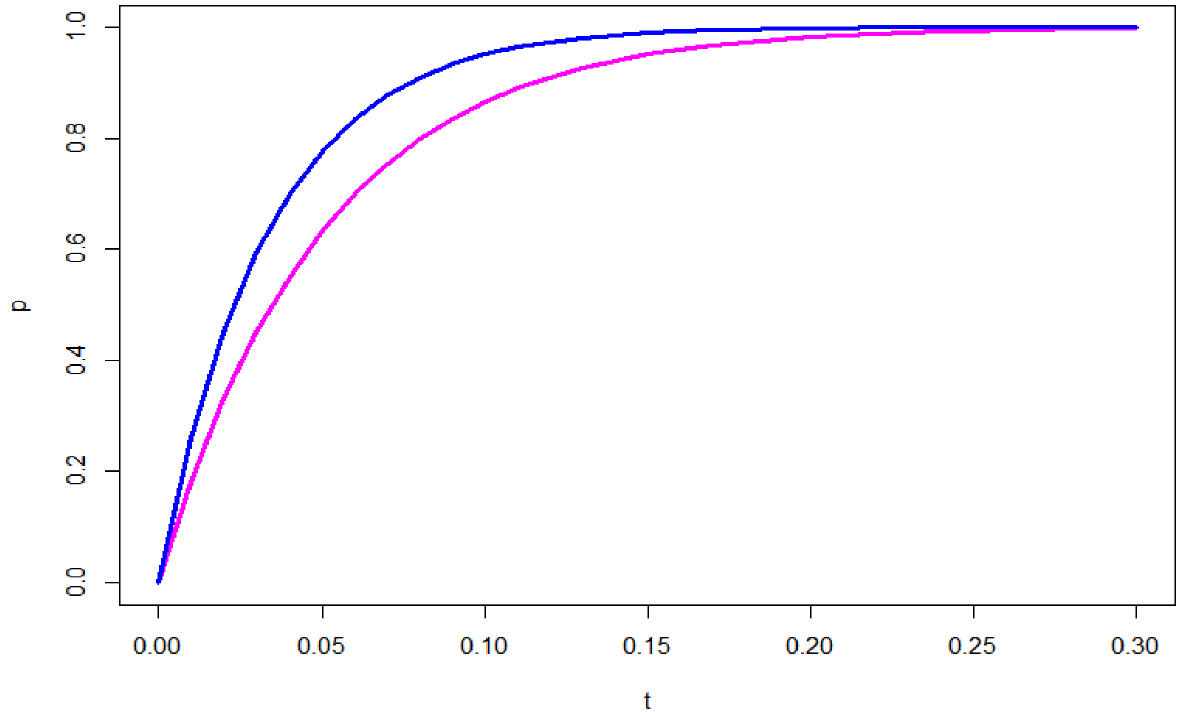
```
mdp

module M1

    x : [0..2] init 0;

    [] x=0 -> 0.8:(x'=0) + 0.2:(x'=1);
    [] x=1 & y!=2 -> (x'=2);
    [] x=2 -> 0.5:(x'=2) + 0.5:(x'=0);

endmodule

module M2

    y : [0..2] init 0;

    [] y=0 -> 0.8:(y'=0) + 0.2:(y'=1);
    [] y=1 & x!=2 -> (y'=2);
    [] y=2 -> 0.5:(y'=2) + 0.5:(y'=0);

endmodule
```

← Modules

← Variables

← Commands

- Modules are executed in **parallel**
- The scheduling is depending on the model type:
    - mdp: non-deterministic
    - dtmc: probabilistic
    - ctmc: race condition
- **Race condition**:
    - The first transition to trigger determines the next state
    - Exponentially distributed $1-e^{-R(s,s')*t}$
    - Consider two rates **R1=20** and **R2=30**. Distributions:



Rewards

- **Rewards** are a way to get additional information about the model
- An example would be “the average path time” of the model:

```
rewards “steps”
   true : 1;
endrewards
```

- Here we assign a reward of 1 to each state where the condition holds. In this case every state
- We can later **accumulate** the rewards

Labels

- **Labels** are a way of identifying a set of states by names instead of numbers or conditions
- **Pending**, **Accept**, **Failed** from our ctmc are examples for such labels
- We could define them in the following way:

```
label "accept" = s=1 | s=2;
label "pending" = s=2;
label "failed" = s=3;
```

PRISM Property Specification Language

- Combines the syntax of several logics(CTL, LTL, PCTL, CSL, etc.)
- Furthermore, it is possible to define properties about **rewards**
- The syntax is generally very similar to the definitions we already saw. E.g.:

```
P<0.1[G[0,10] !"failed"]
```

- We can also let PRISM calculate actual **probabilities** instead of boolean value. E.g.:

```
P=?[G[0,10] !"failed"]
```

- In a sentence: What is the probability that the program does not fail the first ten seconds?

Verification and Simulation

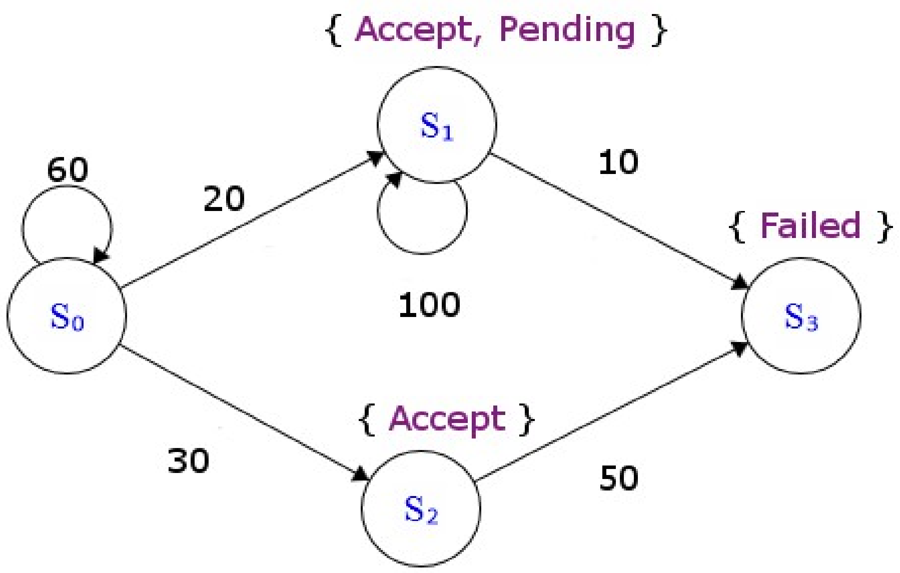
- We can calculate the truth value or probabilities of our models either through **verification** or **simulation**
- Both methods will be covered in more detail in the next presentation
- Verification is using one of four computation engines:
  - **Sparse**(small models)
  - **Explicit**(even smaller models)
  - **MTBDD**(often used in MDP)
  - **Hybrid**(combination of explicit and symbolic)
- Simulation is using PRISM's discrete-event simulation engine

Outline

- Introduction
- CTMC
- PRISM
- PRISM and CTMC
- Summary

Running Example

- Remember our example from before?



- We will now write a model in PRISM and query our previous properties + the average path length

Example in PRISM:

```
ctmc

module running_example

   s : [0..3] init 0;

   [] s=0 -> 60:(s'=0) + 20:(s'=1) + 30:(s'=2);
   [] s=1 -> 100:(s'=1) + 10:(s'=3);
   [] s=2 -> 50:(s'=3);

endmodule

rewards "steps"
   true : 1;
endrewards

label "accept" = s=1 | s=2;
label "pending" = s=2;
label "failed" = s=3;
```

Properties:

- $(P_{>0.3}(\ \boldsymbol{S_0} \vee \textbf{\textit{Pending}}\ U^{[0,\ 0.2]}\ \textbf{\textit{Failed}}))$
- $(P_{>0.8}(\ \boldsymbol{S_0} \vee \textbf{\textit{Pending}}\ U^{[0,\ 0.2]}\ \textbf{\textit{Failed}}))$
- $(S_{>0.99}(\textbf{\textit{Failed}}))$
- $(P_{<0.8}(F^{[0,\ 0.1]}\ \textbf{\textit{Failed}}))$

Properties in PRISM Property Specification Language:

- P>0.3[s=0 | "pending" U[0, 0.2] "failed"]
- P>0.8[s=0 | "pending" U[0, 0.2] "failed"]
- S>0.99 ["failed"]
- P<0.8[F[0, 0.1] "failed"]

Average path time:

- R=?[F “failed”]

Working with properties

- Sample paths can be created in the simulator
- A path can be chosen automatically or be created manually
- Properties are created using the Property Editor
- They can be either verified by calculation or simulation
- Changing the property **epsilon** in the properties, the precision of the simulation can be changed
- Changing the number of samples, precision can be increased

Circadian Clock Example

Outline

- Introduction
- CTMC
- PRISM
- PRISM and CTMC
- Summary

Summary

- Stochastic Model Checking is used to analyse models with probabilistic behaviour
- Continuous-time Markov chains are models with probabilistic behaviour
- PRISM provides a language to describe CTMC
- We can simulate our model and create traces/paths
- We can also query a CTMC with the PRISM Property Specification Language, similar to CSL

References

- Kwiatkowska, M., Norman, G., & Parker, D. (2007, May). Stochastic model checking. In SFM (Vol. 7, pp. 220-270).
- Kwiatkowska, M., Norman, G., & Parker, D. (2010, September). Advances and challenges of probabilistic model checking. In Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on (pp. 1691-1698). IEEE.
- Barkai, N., & Leibler, S. (2000). Biological rhythms: Circadian clocks limited by noise. Nature, 403(6767), 267-268.
- http://www.prismmodelchecker.org/manual/

Questions?