PRISM and CTMC

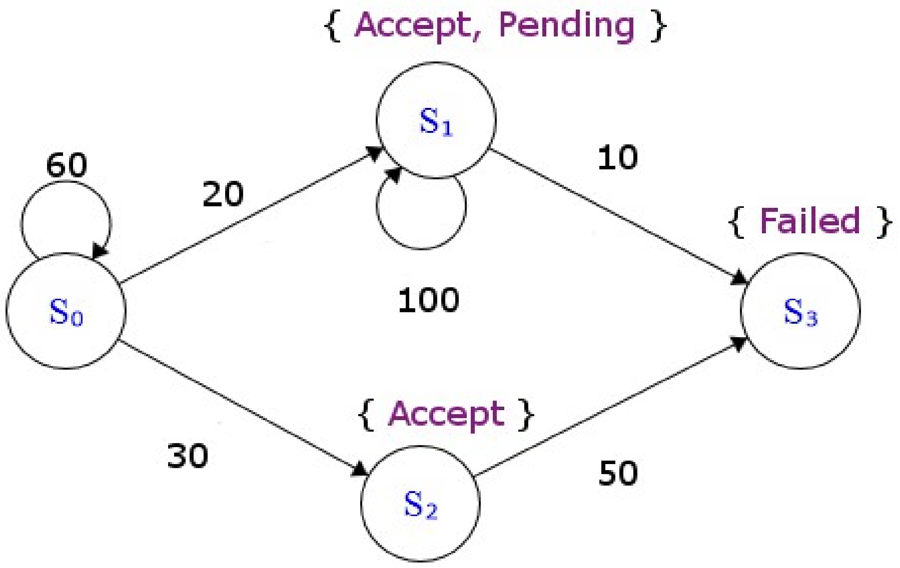
Mario Binder
WS 2017/2018

Outline

- Revision
- Matrices
- Paths
- Uniformisation
- Model Checking – Putting it all together
- Summary

CTMC

- Continuous-time Markov Chains
- **Rates** instead of probabilities
- Transition is chosen by **race condition**
- Example from last time:

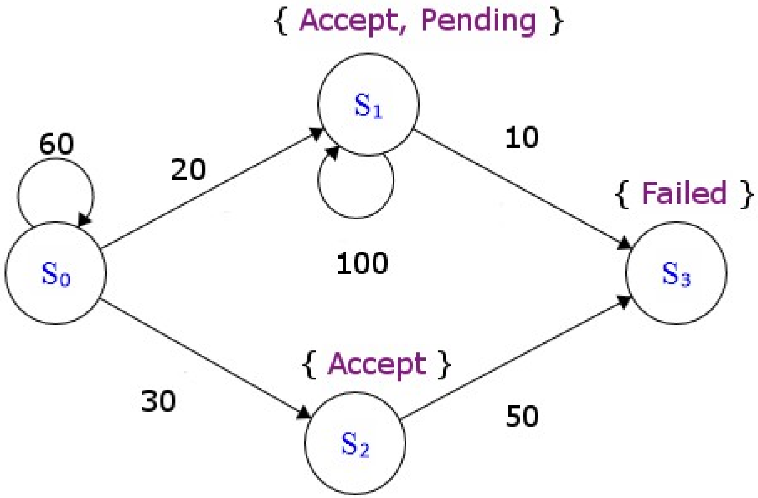


Outline

- Revision
- Matrices
- Paths
- Uniformisation
- Model Checking – Putting it all together
- Summary

CTMC as Matrix

- We can create a transition matrix **R** of a CTMC
- The entries are **0** if there is no connection between states and the **transition rates** otherwise



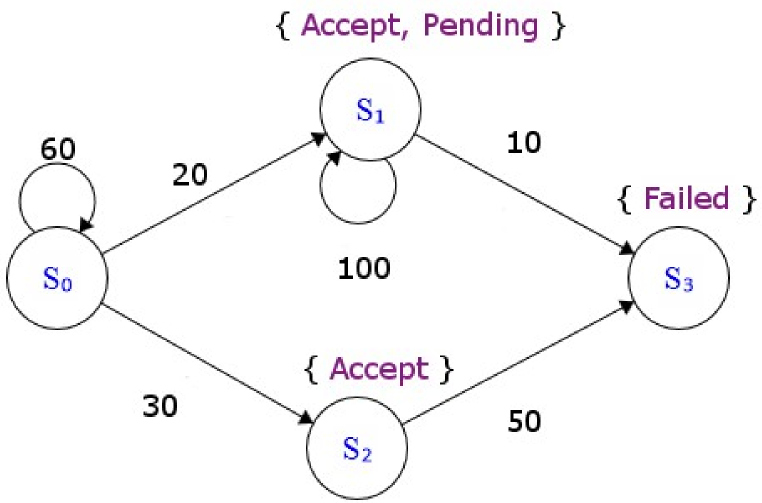
R	S_0	S_1	S_2	S_3
S_0	60	20	30	0
S_1	0	100	0	10
S_2	0	0	0	50
S_3	0	0	0	0

Probability Matrix from a CTMC

- We can also create a matrix of probabilities(similar to DTMC) from a CTMC
- A little bit of terminology(and revision):
  - The **Exit Rate** of a state is defined as $E(s) = \sum_{s' \in S} R(s, s')$ i.e. the sum of all outgoing transition rates
  - A state s is called **absorbing** if E(s) = 0
  - The probability of a transition from s to s' is then:
    - **1** if s=s' and s is absorbing
    - $\frac{R(s,s')}{E(s)}$ if s is not absorbing
    - **0** otherwise

Probability Matrix from a CTMC

- We can now build a probability matrix **P**

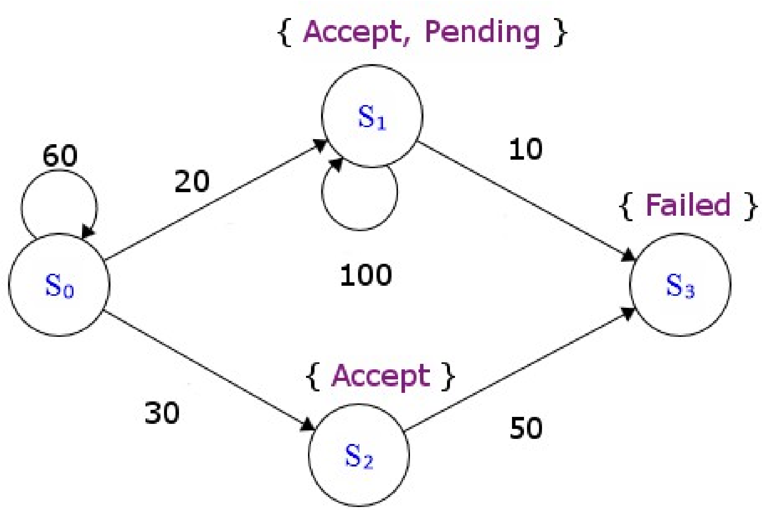


P	S_0	S_1	S_2	S_3
S_0	6/11	2/11	3/11	0
S_1	0	10/11	0	1/11
S_2	0	0	0	1
S_3	0	0	0	1

- Note that if we wouldn't have extra rules for absorbing states, we would have $\frac{0}{0}$ for elements in the last row
- Therefore $P(S_3, S_3) = 1$

Infinitesimal Generator Matrix

- The infinitesimal generator matrix **Q** is essentially the same as the transition rate matrix **R**, except that the elements of the main diagonal are now $-((\sum_{s'\in S} R(s,s')) - R(s,s))$ i.e. the negative matrix row sum without the diagonal element



→

Q	S_0	S_1	S_2	S_3
S_0	-50	20	30	0
S_1	0	-10	0	10
S_2	0	0	-50	50
S_3	0	0	0	0

- We will need this matrix later when talking about **uniformisation**

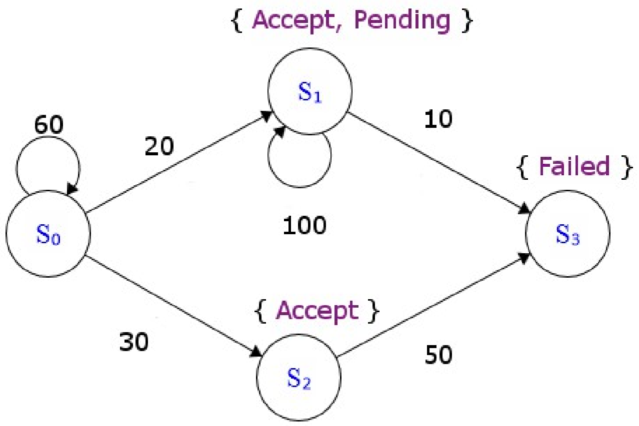
Outline

- Revision
- Matrices
- Paths
- Uniformisation
- Model Checking – Putting it all together
- Summary

Paths

- An **infinite path** ω is a sequence $s_0 t_0 s_1 t_1 \ldots$
- The t-values specify the amount of time spent in a state
- Notation:
  - $\omega(i)$ is the i-th state of the path
  - time(ω, i) is the same as t_i
  - ω@t is the state in the path at time t
- A **finite path** ω is a sequence $s_0 t_0 s_1 t_1 \ldots s_{k-1} t_{k-1} s_k$, where s_k is an absorbing state
  - time(ω, i) is the same as with infinite paths as long as $i \leq k$; otherwise time(ω, i) = ∞

Example



- Finite path $\omega = S_0$-0.5 S_2-0.8 S_3
- $\omega(1) = S_2$
- $\text{time}(\omega, 0) = 0.5$
- $\text{time}(\omega, 2) = \infty$
- $\omega@1 = S_2$
- $\omega@0.3 = S_0$
- $\omega@1000 = S_3$

Set of Paths

- The next thing we want to do, is to span a probability space starting from a start state s_0
- This means we are searching for a function **μ**, that takes a start state $\mathbf{s_0}$, a CTMC **C** and some set of paths **S** starting from s_0 and returns the following:
  - 0 if S = ∅(the empty set)
  - 1 if S = All possible paths in all possible intervals from s_0
  - The cardinality of **S** divided by the cardinality of all possible paths from s_0, otherwise

Important Observations

- The set of all possible paths starting from s_0 is uncountable if $E(s_0) > 0$, because time $t \in \mathbb{R}_{\geq 0}$
- If we have found a function **μ** and we give it a path ω,

  μ(ω) = 0 for all valid paths
- For this reason, we do not only have to pass a path in the sense of a DTMC to **μ** but also **time intervals I**

Notations

- The **cylinder set $C(\omega_{fin})$** is as in DTMC the set of all paths starting with ω_{fin}. However, ω_{fin} is now a sequence

  $s_0 I_0 s_1 I_1 \ldots s_{k-1} I_{k-1} S_k$, where I_i is a non-empty interval in $\mathbb{R}$

- Using cylinder sets, we can recursively define our function **μ**, which we will now call **Pr**:
  - $Pr_s(C(s)) = 1$
  - $Pr_s(C(s, I, \ldots, I_{k-2}, S_{k-1}, I_{k-1}, S_k)) =$

    $Pr_s(C(s, I, \ldots, I_{k-2}, S_{k-1})) * P(S_{k-1}, S_k) *$

    $e^{-E(s_{k-1}) * \inf I_{k-1}} - e^{-E(s_{k-1}) * \sup I_{k-1}}$

Example

- $Pr_{S0}(C(S_0, [0, 2], S_1, [0, 4], S_3)$

  $\rightarrow Pr_{S0}(C(S_0, [0, 2], S_1) *$

  $P(S_1, S_3) * (1 - e^{-440})$

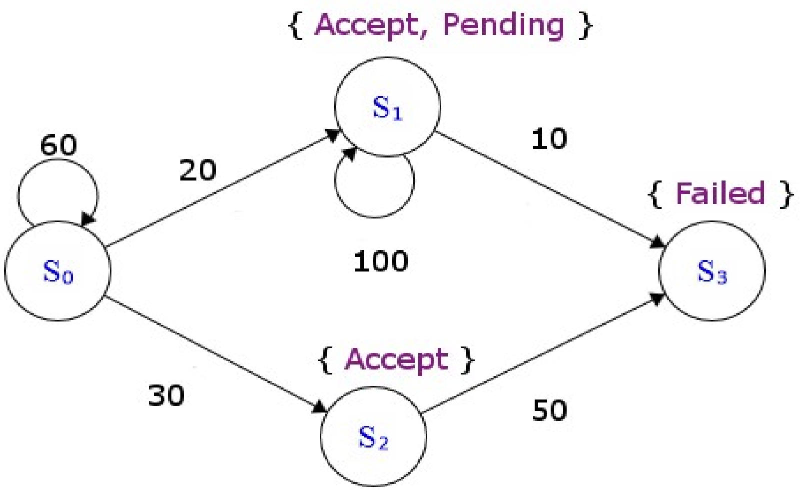
  $\rightarrow Pr_{S0}(C(S_0, [0, 2], S_1) * \frac{1}{11}$

  $\rightarrow Pr_{S0}(C(S_0)) * P(S_0, S_1) *$

  $(1 - e^{-220}) * \frac{1}{11}$

  $\rightarrow 1 * \frac{2}{11} * 1 * \frac{1}{11}$

  $\rightarrow \frac{2}{11} * \frac{1}{11} \rightarrow \frac{2}{121}$



P	S_0	S_1	S_2	S_3
S_0	6/11	2/11	3/11	0
S_1	0	10/11	0	1/11
S_2	0	0	0	1
S_3	0	0	0	1

Transient and Steady-state Behaviour

- **Transient behaviour** = state at time instant t
- **Steady-state behaviour** = state at time t → ∞
- We can assign probabilities to each state **s'** at specific times(e.g. Probability of being in S_3 at time t = 2) starting from a state **s** using the following definitions:
  - $\pi^{C}_{s,t}(s') = Pr_s(\{\omega \in C(s) \mid \omega @ t = s'\})$ for **transient behaviour**
  - $\pi^{C}_{s}(s') = \lim_{t \to \infty} \pi^{C}_{s,t}(s')$ For **steady-state behaviour**
- But how do we compute those probabilities/sets?

Outline

- Revision
- Matrices
- Paths
- Uniformisation
- Model Checking – Putting it all together
- Summary

Uniformisation

- Because calculating uncountably infinite sets is unpractical, we have to resort to numerical solutions
- We can calculate the transient probability matrix Π_t^C by a technique called **Uniformisation**
- **Poisson distribution**:
  - The poisson distribution models the probability of k events occurring at time rate λ
  - Intuitively very useful for CTMC
  - $f(k;\lambda)=\dfrac{\lambda^k e^{-\lambda}}{k!}$

Exponential Matrix

- Quick generating functions reminder:
  - $\exp(x) = \sum_{n=0}^{\infty} \frac{1}{n!} x^n$
- We can express Π_t^C as a power series(using **Master Equation** and **Chapman-Kolmogorov** Equation → see References):
  - $\Pi_t^C = \exp(Q * t) = \sum_{n=0}^{\infty} \frac{1}{n!} (Q * t)^n$
- However, the computation of Π_t^C can be unstable

  (round-off errors)

Better Approach

- We can **uniformise** a CTMC by creating a normalized probability matrix from the infinitesimal generator matrix Q:
  - $P^{unif(C)} = I + \frac{Q}{q}$
  - q is the maximal exit rate occurring in the states of the CTMC(in our example q = 110)

Q	S_0	S_1	S_2	S_3
S_0	-50	20	30	0
S_1	0	-10	0	10
S_2	0	0	-50	50
S_3	0	0	0	0

$\mathbf{P^{unif(C)}}$	S_0	S_1	S_2	S_3
S_0	6/11	2/11	3/11	0
S_1	0	10/11	0	1/11
S_2	0	0	6/11	5/11
S_3	0	0	0	0

Better Approach

- Using the normalized matrix we can now express Π_t^C in the following way(using the poisson distribution):

$$\Pi_t^C = \sum_{n=0}^{\infty} \frac{(qt)^n * e^{-qt}}{n!} * \left(P^{unif(C)}\right)^n$$

- Advantages:
  - $P^{unif(C)}$ is stochastic and does not have negative numbers in contrast to Q → more stable
  - Poisson distribution can be calculated efficently
  - The matrix multiplication can be rewritten

    as a vector-matrix multiplication

    → less computation

Outline

- Revision
- Matrices
- Paths
- Uniformisation
- Model Checking – Putting it all together
- Summary

CTL – Reminder

- Operators:
  - Propositional Logic
  - Probability-Operator P + Steady-state Operator S
  - Next(unary, X Φ)
  - Until with a time interval I(binary, $\Phi\ U^{I}\ \Psi$)

- All other temporal logic operators can be built from the until-operator

General Process

- We want to find the set of states that satisfy a CTL formula Φ
- We can use the normal **inference rules** for all parts of

  a property that only contain propositional logic
- We can use normal **model checking** for all other temporal operators
- We only have to do things differently(also to DTMC) when using the probability operator or the steady-state operator

Probability Operator

- We can can build all our temporal operators from the Until-operator, except the Next-operator
- Therefore two cases:
  - $P_{\sim p}[X\ \Phi] \rightarrow$ see DTMC
  - $P_{\sim p}[\Phi\ U^{I}\ \Psi]$ which we can again split into three cases:
    - $I = [0, t]$
    - $I = [t, t']$ where $t \leq t'$
    - $I = [t, \infty)$

Case I = [0, t]

- To calculate the probability for a specific start state s we can use the following formula:

$$\mathrm{Prob}^{C}(s, \Phi\, U^{[0,t]}\, \Psi) = \sum_{s' \models \Psi} \pi_{s,t}^{C[\neg\Phi \vee \Psi]}(s')$$

- Here, $C[\neg\Phi \vee \Psi]$ is the CTMC where we remove all outgoing connections from states where $\Phi\ U^{I}\ \Psi$ is either true or false but not pending/unresolved
- Calculating $\pi_{s,t}^{C[\neg\Phi \vee \Psi]}(s')$ can be done with uniformisation
- This can be interpreted as the probability of reaching a satisfying state from start state s *within* t time units

Other Cases

- For I = [t, t'] we can split the calculation into two parts:
  - The probability of satisfying Φ until time t
  - The probability of satisfying $\Phi\ U^{[0,\ t'-t]}\ \Psi$
- The two conditions are multiplied, and we get as expected:

$$\mathrm{Prob}^{C}(s, \Phi U^{[t,t']}\Psi) = \sum_{s' \models \Phi} \pi_{s,t}^{C[\neg\Phi]}(s') \sum_{s' \models \Psi} \pi_{s,t'-t}^{C[\neg\Phi \vee \Psi]}(s')$$

- Although more complicated, we can do a similar thing with I = [t, ∞)

Steady-state Operator

- Again, we have to consider two cases:
  - The CTMC is strongly connected
  - The CTMC is not strongly connected
- In the first case we can solve the following equation system:

  $\pi^C * Q = 0$ and $\sum_{s \in S} \pi^C(s) = 1$
- In the second case, we first have to identify all **bottom strongly connected components(BSCC)** and then calculate the probability of reaching each component

Model Checking in PRISM

- PRISM transforms a model in specified in the PRISM language into an internal representation(discarding unreachable states), according to the chosen **computation engine:**
  - **MTBDD:** Multi-terminal binary decision diagrams. BDD with e.g. real numbers as terminals(the bdd is encoded with rows and columns from the transition matrix)
  - **Sparse:** Uses sparse matrices(I couldn't find out which exact technique PRISM uses)
  - **Hybrid:** Combination of the above
  - **Explicit:** Uses the transition matrix

Solving Linear Equations in PRISM

- As seen with the steady-state operator, we need to be able to solve linear equations
- PRISM provides many methods/options:
  - Power method
  - Jacobi method
  - Gauss-Seidel method
  - ...
- All those methods and uniformisation are iterative methods and are therefore terminated when they converge below an **epsilon threshold**

Statistical Model Checking

- As shown last time, PRISM is also capable of solving model checking problems using its built-in discrete event simulator
- Currently, it only supports the operators P and R
- The process is analogously to **hypothesis testing** in statistics
- Different supported methods in PRISM are:
  - **CI Method:** Testing against Student's t-distribution
  - **Asymptotic CI Method:** Uses central limit theorem
  - **Approximate Probabilistic Model Checking**
  - **Sequential Probability Ratio Test**

Outline

- Revision
- Matrices
- Paths
- Uniformisation
- Model Checking – Putting it all together
- Summary

Summary

- We looked at the creation of various embedded matrices in CTMC(**transition rate matrix**, **probability matrix**, **infinitesimal matrix**, **uniformised matrix**)
- Various operators were defined for finite and infinite paths
- We discussed the process of calculating probabilities for transient and steady-state behaviour called **uniformisation**
- Using CTL, the calculation behind model checking, which is very similar to normal model checking extended by transient and steady-state operators, was shown

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.
- Stewart, W. J. (1994). Introduction to the numerical solution of Markov chains. Princeton University Press.

Questions?