02246 Mandatory Assignment
Part 2: Stochastic Modelling and Verification in Discrete Time

Flemming Nielson, Michael Smith, Lijun Zhang

Set: October 29, 2013

Due: 08:00, December 3, 2013

This assignment is concerned with Part II of the course — stochastic modelling and verification in discrete time. The problems are divided into three parts, with increasing levels of difficulty. All the problems in the first two parts are compulsory, however you may choose between focussing on the practical or the theoretical aspects of the course for the more advanced problems in Part C.

You are expected to spend between 20 and 25 hours on this assignment. Of this, you should expect to spend roughly 6–8 hours on Part A, 9–11 hours on Part B, and 5–6 hours on Part C. You are encouraged to work in groups of two (or at most three) people, but you must clearly identify the contributions of each group member, and you will be jointly responsible for the finished report.

Answers to all parts should be typed up and submitted electronically as a report, although drawings and formulae may be handwritten and scanned. More detailed instructions as to the style of answer we expect for each part are included below.

Part A: Introductory Problems

You are required to answer all *of the problems in Parts A1 and A2. These are short problems, aimed at reinforcing your basic understanding of the course material. Answers should be self-contained, but concise, and you should clearly specify which question you are answering.*

Part A1: Practical Problems

1. In this problem, we will add probabilities to the FCFS scheduler from the previous assignment, so that we construct a discrete time Markov chain.

   (a) Identify all the sources of non-determinism in the model, and explain whether they are due to local non-determinism between the commands in a module, or due to the concurrent execution of two or more modules.

   (b) Resolve the local non-deterministic choices by modifying the modules to include probabilistic commands. Recall that the syntax of this, in PRISM, is as follows:

   $$[\langle ACTION \rangle] \ \langle GUARD \rangle \ \rightarrow \ p_1 : \langle UPDATE_1 \rangle + \cdots + p_n : \langle UPDATE_n \rangle \ ;$$

   where $\sum_{i=1}^{n} p_i = 1$. For now, you should use a *uniform distribution* (i.e. one where all the probabilities are the same) for each probabilistic command.

   (c) Change the first line of the PRISM file from '`mdp`' to '`dtmc`', and save the model in a new file, using the `.pm` extension. This tells PRISM that the model describes a DTMC. Build the model in PRISM, and check that there are no error messages. How many states does your model have?

   (d) The model you have constructed is purely probabilistic. What has happened to the non-determinism that we had due to the concurrent execution of the modules?

   (e) Use PRISM to verify that the following PCTL formula — stating that a job will almost certainly complete — holds of your new model in all reachable states:

   $$(task_1 > 0) \Rightarrow \mathbb{P}_{\geq 1}(F(task_1 = 0))$$

2. This question is about using PRISM to compute numerical properties of your model.

   (a) Calculate the steady state distribution of the model. What is the probability that there are no jobs in the queue of the scheduler?

   (b) By looking at the steady state probabilities, calculate the expected length of a job for $Client_1$. You should do this computation manually. *Hint: you will first need to sum the probabilities of the states where* $Client_1$ *has a job of length* i*, for* $0 \leq i \leq 5$.

   (c) Calculate the transient distribution of the model at time $t = 10$. What is the probability that $Client_1$ does not currently have a job at this time?

   (d) Plot a graph of the probability that $Client_1$ does not have a job, against time t, for $0 \leq t \leq 10$. You should use the same method as part (c) for computing the probabilities, but then plot the graph by hand (or by using a software package such as a spreadsheet). Can you given an intuitive explanation of your graph?

3. This question is about PCTL model checking in PRISM. For each of the following, write down a PCTL formula that captures the query, and use PRISM to determine whether the *initial state* of the model satisfies it:

(a) Is the probability greater than 0.2 that $Client_1$ will have have an active job of length greater than 2 in the next time unit?

(b) Is the probability less than 0.5 that $Client_2$ will create a job of length 5 within 10 time units?

(c) Is the probability greater than zero that $Client_1$ will always have an active job?

(d) For each of the above properties, use the '`P=?`' notation in PRISM, to calculate the actual probability of the path formula holding.

4. In Part I of the course, we used the syntax '`P>=1`' and '`!P<=0`' to encode the quantifiers A and E, when writing CTL formulae in PRISM. If we want to model check CTL formulae on a *DTMC*, however, these do not encode the A and E quantifiers as we would expect.

   (a) We say that a DTMC satisfies a CTL formula if the transition system obtained by throwing away the probabilities in the DTMC satisfies the formula. Write this down formally.

   (b) Consider the CTL formula $EG\ \Phi$, and the PCTL formula $\neg\mathbb{P}_{\leq 0}(G\ \Phi)$, where Φ is an atomic proposition. Describe a DTMC in which one formula holds, but not the other. *Hint: you can do this using a DTMC with only two states.*

   (c) Explain why this is the case.

   (d) What feature of CTL would you require in order for the semantics of these formulae to be the same? *Note: you do not need to go into any detail of how to do this.*

Part A2: Theoretical Problems

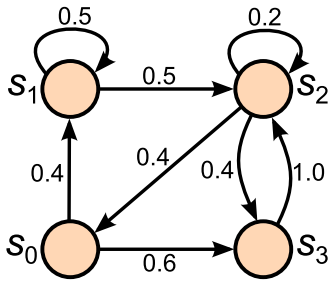

Figure 1: DTMC for questions A2.1 and A2.2

1. Consider the DTMC in Figure 1, which is shown graphically. The initial state is s_0. Because this question is just concerned with numerical properties of the DTMC, we haven't labelled the states with atomic propositions.

   (a) Write down the probability transition matrix corresponding to the DTMC, and its initial distribution as a row vector. The state s_i should correspond to the index $i+1$.

   (b) Compute the transient distribution Θ_3 of the DTMC after 3 time steps. *Hint: you may want to start by computing* Θ_1 *and then* Θ_2.

   (c) Write down the matrix equation whose fixed point is the steady state solution of the DTMC. Solve this equation system analytically.

2. Encode the DTMC in Figure 1 as a PRISM module, using a variable s to represent the state, such that $0 \leq s \leq 3$. Use PRISM to compute the steady state distribution, and check that it agrees with your answer to question 1(c).

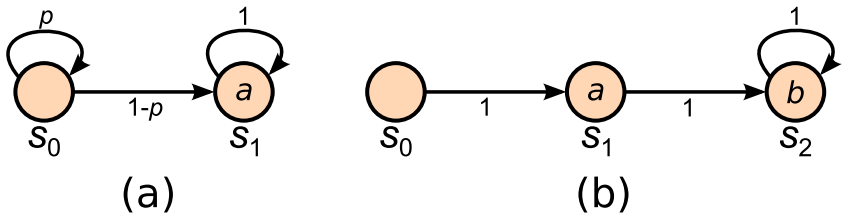

Figure 2: DTMCs for question A2.3

3. Consider the DTMC in Figure 2(a), where the initial state is s_0, and the labels are shown on the states.

   (a) For what values of p is it the case that $s_0 \models \mathbb{P}_{\geq \frac{1}{2}}(F^{\leq 2}\ a)$?

   (b) What is the expected number of time steps until a holds, in terms of p? *Hint: think about what type of distribution this is.*

   (c) Consider the DTMC in Figure 2(b). How can we modify it, without increasing the number of states, so that it takes on average 5 time steps to reach s_1 from s_0, and 10 time steps to reach s_2 from s_1?

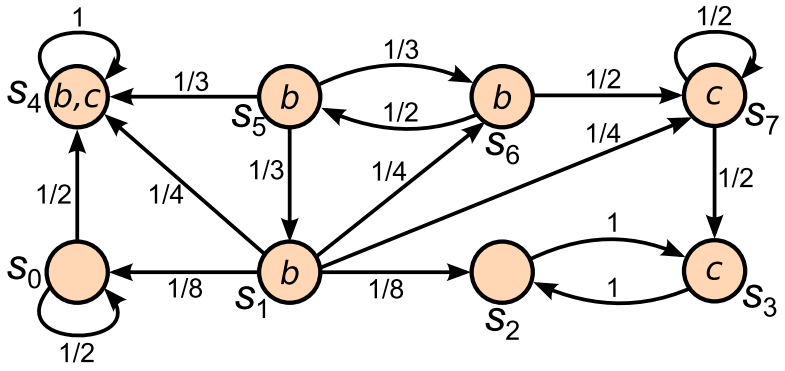

Figure 3: DTMC for question A2.4

4. Consider the DTMC in Figure 3, which we will call $\mathcal{M} = (S, \boldsymbol{P}, s_1, AP, L)$, where $AP = \{\, b, c \,\}$. The initial state is s_1 and the labels are shown on the states. Determine which states of the DTMC satisfy the following PCTL formulae, by following the PCTL model checking algorithm:

   (a) $\mathbb{P}_{\geq \frac{17}{19}}(b\ U\ c)$

   (b) $\mathbb{P}_{\geq \frac{1}{2}}(X\ \mathbb{P}_{> \frac{1}{3}}((b \vee c)\ U^{\leq 2}\ (b \wedge c)))$

Part B: Intermediate Problems

You are required to answer all *of the problems in Parts B1 and B2. These are more involved than the problems in Part A, and require a more detailed understanding of the course material. Your answers should be submitted as a self-contained report — rather than providing a list of answers, you should present a write-up that is readable, explanatory, and addresses all of the questions posed.*

Part B1: Practical Problems

1. *Lottery scheduling* is a scheduling discipline where each task is assigned a number of tickets. To decide which task to execute, the scheduler randomly selects a ticket (under a uniform distribution), and the task owning the ticket is allowed to execute. A lottery scheduler can be either *pre-emptive*, in which a task can only execute for a certain amount of time before being interrupted, or *non pre-emptive* if a task always executes to completion once selected.

   (a) Construct a PRISM model of a pre-emptive lottery scheduler with three clients, where each task is given a single ticket, and the quantum (the length of time before a task is interrupted) is one time unit. You will have to decide on an appropriate way to model the tickets, which means deciding on an appropriate level of abstraction.

   (b) We're now interested in the time taken for the first task from $Client_1$ to complete. To do so, let's introduce a new module called *Monitor*, which has a single Boolean variable called *finished*. The module should only perform $finish_1$ actions — the first time it does a $finish_1$ it changes its state so that *finished* = `true`, and for subsequent actions it remains in this state.

   (c) Write down a PCTL formula (using the '`P=?`' syntax), expressing the probability that the first task from $Client_1$ completes within t time units — you should define t as an integer constant, without a value. Use the experimentation feature of PRISM (as described in the tutorial), to plot a graph of this probability, for $0 \leq t \leq 20$.

   (d) Is it possible for your lottery scheduler to suffer from starvation?

   (e) Modify your model so that the quantum is two time units. How does this affect the time taken to complete a task from $Client_1$?

2. Rather than assigning the same number of tickets to each task, we can give varying priority to different tasks by assigning them *different* numbers of tickets.

   (a) By assigning different numbers of tickets to different tasks, approximate an SRT scheduler using lottery scheduling.

   (b) How does this affect the time taken to complete a task from $Client_1$? Is starvation possible?

   (c) Increase the number of clients in your model until you reach the maximum for which PRISM can still build and solve the underlying DTMC.

   (d) Investigate what happens if one client tries to perform long jobs, but the others all generate very short jobs. Is the scheduler vulnerable to a denial of service attack?

   (e) Can you express denial of service as a PCTL property? Explain your answer.

Part B2: Theoretical Problems

1. A step-bounded until formula $\Phi_1\ U^{\leq n}\ \Phi_2$ in PCTL is satisfied of a path if Φ_2 holds after at most n time steps, and until then Φ_1 always holds. We would like to generalise this to the form $\Phi_1\ U^I\ \Phi_2$, where I is an interval over the natural numbers.

   (a) Define the semantics $\mathcal{M}, \sigma \models \Phi_1\ U^I\ \Phi_2$ of this new path formula, where σ is a path in the DTMC $\mathcal{M}$.

   (b) Consider four cases on the form of the interval I, for $0 < n \leq n'$: $[0, \infty)$, $[0, n]$, $[n, \infty)$ and $[n, n']$. For each case, can you encode the formula $\mathbb{P}_J(\Phi_1\ U^I\ \Phi_2)$ in terms of the existing PCTL operators? If so, show how, and if not, explain why.

   (c) For the cases where you cannot encode the formula, suggest a modification to PCTL (other than adding the new path operator) that would allow you to encode it.

   (d) Describe a model checking algorithm for $\mathbb{P}_J(\Phi_1\ U^{\geq n}\ \Phi_2)$, i.e. $\mathbb{P}_J(\Phi_1\ U^{[n,\infty)}\ \Phi_2)$.

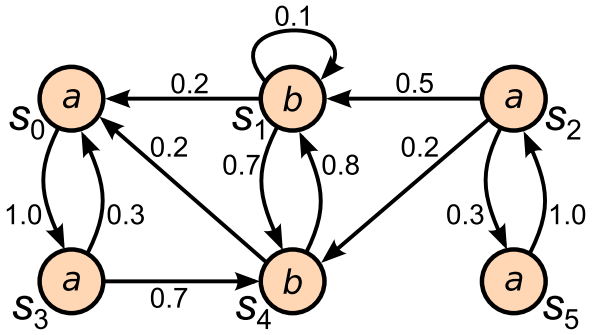

Figure 4: DTMC for question B2.2

2. Consider the DTMC in Figure 4, where s_0 is the initial state, and the labels are shown on the states. We will call this $\mathcal{M} = (S, \boldsymbol{P}, s_0, AP, L)$, where $AP = \{\, a, b \,\}$.

   (a) Write down the coarsest probabilistic bisimulation relation you can find between the states of $\mathcal{M}$: $R \subseteq S \times S$, and show that it is a probabilistic bisimulation.

   (b) Construct the bisimulation quotient $\mathcal{M}/\!\sim$ of $\mathcal{M}$.

   (c) What is the relationship between the steady state distribution of $\mathcal{M}$, and that of its bisimulation quotient, $\mathcal{M}/\!\sim$?

   (d) If you could choose a different labelling function L for the DTMC $\mathcal{M}$, what choice would you make to give the coarsest possible probabilistic bisimulation? From a modelling point of view, what does this choice of labelling function correspond to?

   (e) An equivalent formulation of probabilistic bisimulation is *lumpability*, which arose from the mathematical development of Markov chains. A partitioning of the state space of a DTMC (i.e. a labelling function L such that $|L(s)| = 1$ for all s) is said to be lumpable if for all pairs of states $s_1, s_2 \in S$ such that $L(s_1) = L(s_2)$, and all atomic propositions $a \in AP$:

   $$\sum_{s' \in L^{-1}(a)} \boldsymbol{P}(s_1, s') = \sum_{s' \in L^{-1}(a)} \boldsymbol{P}(s_2, s')$$

   Prove that probabilistic bisimulation implies lumpability.

(f) Is it possible to write a PCTL* formula that can distinguish between two states in a lumpable partition (assuming that you use the same labelling function in both instances)? Explain your answer.

Part C: Advanced Problems

You are required to answer the problems in just one *of the following parts* — either *Part C1* or *Part C2. These are more open ended problems, allowing you to explore some of the concepts in the course in more depth. Your answers should be submitted as a report, as for Part B.*

Part C1: Practical Problems

1. Consider an extension of the lottery scheduler from part B1, where there are multiple servers — the scheduler has to decide not only the order in which tasks execute, but which server executes them. Modify your lottery scheduler to model a system with two servers. You will need to make your own decisions about how to do this, in terms of separating the functionality of the scheduler from the execution of the tasks (up until now, these have been combined in a single module). Investigate the relative responsiveness of the system in a number of scenarios — you should include cases where the choice of server is made probabilistically, and cases where one server is reserved for high priority tasks.

Part C2: Theoretical Problems

1. Theorem 10.40 in Baier & Katoen (page 786) states the following complexity result for PCTL model checking of a formula Φ on a DTMC $\mathcal{M} = (S, \boldsymbol{P}, \iota_{init}, AP, L)$:

$$O(\text{poly}(|S|) \cdot n_{\max} \cdot |\Phi|)$$

where $n_{\max}$ is the maximal bound on an until formula occurring in Φ. Explain this complexity result, in relation to the model checking algorithm for PCTL — you do not need to give a formal proof, but you should clearly explain why each term in the expression arises.

Now consider a PCTL formula $\mathbb{P}_{\leq p}(\Phi_1\ U^{\leq n}\ \Phi_2)$, where Φ_1 and Φ_2 are atomic propositions. Explain in detail a model checking algorithm for this formula, and determine the complexity of this algorithm. In particular, you should determine what the polynomial on the size of the state space is. Is this the best complexity that you can achieve, or is there a more efficient algorithm?