An Introduction to Probabilistic Modelling

Often we are concerned with questions of *reliability*, of *probable* occurrence of a fault or error;

- with questions of *probability of occurrence* of some sequence of actions.

We might also be interested in *performance*, measured by *average* or *expected* values of a variable (throughput? error count?) that may vary at random.

So far, we have used models which can be visualised as "state machines" or *automata*. The transitions between states may have been

- *deterministic*, with at most one transition possible from each state; or
- *in*deterministic, with possible a choice of transitions from a state.

We are now interested in models in which the transitions between states are *probabilistic*.

An Introduction to Probabilistic Modelling

Probalistic modelling employs *probabilistic automata*: "state machines", labelled transition systems similar to those used in SPIN or UPPAAL.

- A state of the model corresponds to a location in the labelled transition system
- An action of the model corresponds to a passage (arrow) from one location to another
- An action may be guarded by a condition: the action is enabled - may be taken - only if the condition is "true"
- An automaton is *deterministic* if at most one action from a state may be enabled at a time; a *in*deterministic automaton may offer a choice of actions from a state.
- A probabilistic automaton offers probabilistic choice from each state: during a run, a choice is made at random from a number of (enabled) actions.
  - deterministic & probabilistic -> from every state, a random choice is made from a single set of actions (whose probabiltiies add up to 1);
  - indeterministic & probabilistic -> there may be, from a state, more than one set of actions from which a randome choice is made.

An Introduction to Probabilistic Modelling

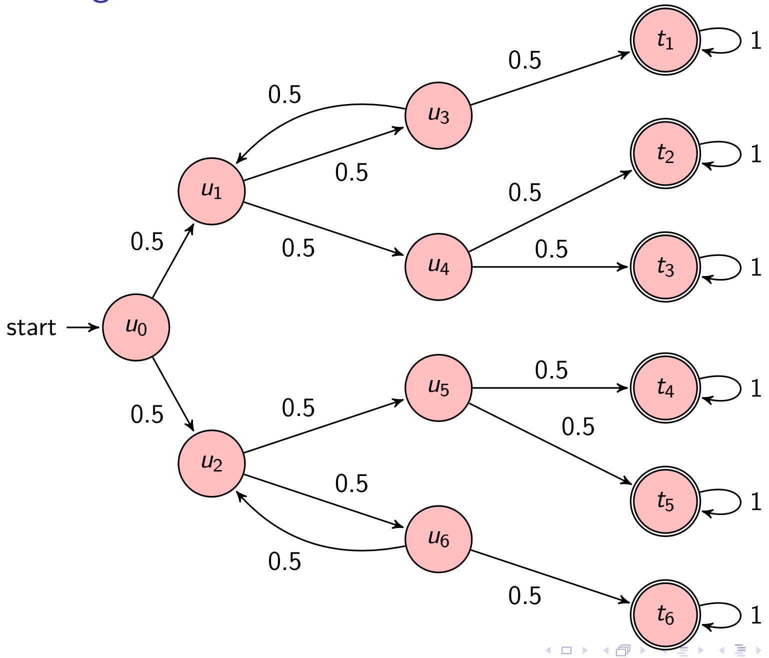
We shall be looking mainly at a simple type of “state machine” with probabilistic transitions, called a *discrete-time Markov chain*, and its use in model-checking systems where probability or randomness plays a significant role. It is deterministic and probabilistic in the sense above.

- We will do this using model-checking software tool, *PRISM*, with an associated formal language in which we can construct Markov-chain-based models and test their properties.
- You have met a couple of *temporal logics, CTL* and *LTL*, in which we can express properties of models built in UPPAAL or SPIN. We shall meet *probabilistic computation tree logic, PCTL* for expressing properties of probabilistic models.

Acknowledgement

These lectures are partly based on those of Dave Parker, published on the PRISM web site: see “further reading” at the end of these slides.

Simulating a die roll with coin tosses



Simulating a die roll with coin tosses

A probabilistic model due to to Knuth and Yao give an amusing introductory example. The diagram shows a *probabilistic automaton* which shows how the roll of a die can be simulated by tosses of a fair coin.

- Start in location u_0;
- repeatedly toss a coin: on HEAD take the upper edge transition; on TAIL, the lower edge;
- At locations $u_0, \ldots, u_6$ the outcome is as yet undecided.
- The process is repeated until one of the accepting states $t_1, \ldots t_6$ is reached.
- State $t_k \Rightarrow$ the outcome is k.

Notice the edges are labelled with *probabilities* and the edges out of a particular node have probabilities adding up to 1.

How does it work?

t_2, for example, can be reached from u_0 in various ways:

- $u_0 u_1 u_4 t_2$
- $u_0 u_1 u_3 u_1 u_4 t_2$
- $u_0 u_1 u_3 u_1 u_3 u_1 u_4 t_2$
- etc: in general $u_0 u_1 (u_3 u_1)^n u_4 t_2$, for $n = 0, 1, 2, 3, \dots$ (to ∞)

To calculate the probability of each of these paths, multiply probabilities along the edges: $0.5 \times (0.5 \times 0.5)^n \times 0.5 \times 0.5 = \frac{1}{8} \times (\frac{1}{4})^n$.
Each of these (for $n = 0, 1, 2, \dots$) is an *alternative* path from u_0 to t_2: so *add up* these probabilities to get the probability of outcome t_2:

$$Pr(reach\ t_2) = \sum_{n=0}^{\infty} \frac{1}{8} \times \left(\frac{1}{4}\right)^n = \frac{1}{6}$$

(an example of a *geometric series*).
Can you see how you get this same calculation with each of $t_1, \dots, t_6$?

- For $k = 1, \dots, 6$, $Pr(reach\ t_k) = \frac{1}{6}$.

How many coin tosses does it take?

It follows that *with probability 1*, one of the states $t_1, \ldots, t_6$ is reached. Notice that to get one of these outcomes always takes an odd number of tosses, at least 3.

- For $n = 0, 1, \ldots$ the probability of getting to a t state in $2n + 3$ tosses is $\frac{1}{8} \times (\frac{1}{4})^n \times 6$
- It could take many tosses but the probability becomes vanishingly small as $n \to \infty$.

We might like to know the *average* number of tosses it will take to reach a t state. If we multiply each possible number by its probability and then add up we get

$$\text{Avg no of tosses} = \sum_{n=0}^{\infty} (2n+3) \frac{1}{8} \left(\frac{1}{4}\right)^n \times 6$$

With some maths crunching we find this comes to $3\frac{2}{3}$. On average we need less than 4 tosses to get a result, a terminal t-state.

What is all this good for?

We have used automata or “state machines” to model systems, and model checking tools such as SPIN or UPPAAL to examine possible behaviours of the system as sequences of transitions between states. For example, to test whether ...

- (safety) no behaviour of the system will reach undesirable state s;
- (liveness) one a a set of desirable states will eventually be reached by every behaviour;
- no behaviour of the system leads to deadlock

These models may (and usually do) have some *non-determinism* and we have been interested in the *possibility* or the *necessity* of paths (sequences of transitions) from an initial state.
With *probabilistic automata* we can attach *probabilities* to transitions and ask about the *probability* of a path, or the *probability* that some state (or deadlock) will be reached.

Discrete-time Markov Chains

One type of probabilistic automaton is the *discrete-time Markov chain*. This is essentially a state-transition system in which transitions between states are decorated with probabilties, and the probabilities of transitions out of each state *add up to 1*. Formally, a DTMC is a structure (S, s_{ini}, P, L) where

- S is a set of *states* - usually finite but can be countably infinite;
- $s_{ini} \in S$ - an *initial state*. Some folks have instead a probability distribution over a set of possible initial states;
- $P : S \times S \rightarrow [0, 1]$ - the matrix of transition probabilities: $P(s, s') =$ the probability of a transition from s to s'. P is a *stochastic matrix*:
  - $\forall s, s' \in S : 0 \leq P(s, s') \leq 1$;
  - $\forall s \in S : \sum_{s' \in S} P(s, s') = 1$.
- L is a function which associates with each state s a set $L(s)$ of *propositions* true there.

Note that a Markov chain is *memoryless*: the behaviour from a state s is *independent* of how it got to state s.

Die-roll with coin-tosses as a discrete-time Markov Chain

$$
\begin{aligned}
S &= \{u_0, u_1, u_2, u_3, u_4, u_5, u_6, t_1, t_2, t_3, t_4, t_5, t_6\} \\
s_{ini} &= u_0 \\
P &= \begin{bmatrix}
0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\
0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \\
L(u_k) &= \{\texttt{k},\ \texttt{undecided}\} \qquad (k = 0, \ldots, 6) \\
L(t_k) &= \{\texttt{result-k}\} \qquad (k = 1, \ldots, 6)
\end{aligned}
$$

PRISM

A model-checking (software) tool based on probabilistic automata.

- One expresses a system which has probabilistic behavour in the PRISM formal modelling language.
- Underneath, the modelling is by probabilistic automata, for instance, discrete-time Markov chains.
- The tool will run simulations of the system based on the model.
- The tool will check or verify properties of the model expressed in a suitable temporal logic.

We shall look at a PRISM model the die-roll-by-coin-tosses process as an example of this.

A PRISM model of die-roll-by-coin-tosses

```
dtmc

rewards  "steps"
true: 1;
endrewards

module die

// local state
s : [0..7] init 0;
// value of the die
d : [0..6] init 0;

[] s=0 -> 0.5 : (s'=1) + 0.5 : (s'=2);
[] s=1 -> 0.5 : (s'=3) + 0.5 : (s'=4);
[] s=2 -> 0.5 : (s'=5) + 0.5 : (s'=6);
[] s=3 -> 0.5 : (s'=1) + 0.5 : (s'=7) & (d'=1);
[] s=4 -> 0.5 : (s'=7) & (d'=2) + 0.5 : (s'=7) & (d'=3);
[] s=5 -> 0.5 : (s'=7) & (d'=4) + 0.5 : (s'=7) & (d'=5);
[] s=6 -> 0.5 : (s'=2) + 0.5 : (s'=7) & (d'=6);
[] s=7 -> (s'=7);
```

A PRISM model of die-roll-by-coin-tosses

- This model is headed `dtmc` - it is a discrete-time Mrkov chain
- The state is determined by two variables, `(s,d)`.
  - `s` takes integer values in the range 0..7 and is initially 0
  - `d` takes integer values in the range 0..6 and is initially 0
- The "operation" of the model is specified by several commands beginning `[]`. For instance
  - `[] s=3 -> 0.5 : (s'=1) + 0.5 : (s'=7) & (d'=1);`
  - means if s=3 then choose at random one of the following: with 0.5 probability update s to 1; and with 0.5 probability update s to 7 and d to 1.
  - State (s=3,d=0) corresponds to location u_3 on the graph (diagram), (s=1,d=0) to u_1 and (s=7,d=1) to t_1.
  - Notice that *primed* variable are updated values and non=rimed variables "original" values. Updates are expressions with a primed variable on the left of an '='. Unprimed variable can appear in guard expressions and on the right hand side of updates.
  - The '+' signs indicate probabilistic choice between updates, with indicated probabilities. The probabilities must total to 1.0.
- Try to match the locations on the graph with (s,d) values and all out of each location with a command.

Using the PRISM model

States

- $\{(s,d)|s=0...6, d=0\}$ correspond to the u_n locations, $n=0...6$; while
- $\{(s,d)|s=7, d=1...6\}$ correspond to the t_n locations, $n=1...6$.
- Some (s,d) value pairs never occur
- What is the significance of the command `[] s=7 -> (s'=7);`?

"Rewards"

- The model has a "rewards" section which declares a variable `steps`
  - At each location, the reward variable is updated provided the condition is met.
  - In this case, the condition `true` is always met and the update is an increment of `1`.
- Thus the `steps` variable counts visits to locations.

Using the PRISM model

PRISM will run simulations: you can see, step by step,

- locations (states) begin visited,
- commands "firing" and updates being chosen at random.

PRISM will verify properties.

- "What is the probability the model reaches a t state?"
- "What is the probability the model reaches a state where the result is 6?"

Such properties are expressed in the temporal logic PCTL, for instance

- $\mathbb{P}_{<1}\Diamond(\texttt{result-1} \vee \texttt{result-2} \vee \texttt{result-3} \vee \texttt{result-4} \vee \texttt{result-5} \vee \texttt{result-6})$
- $\mathbb{P}_{<1}\mathbf{F}(\texttt{result-1} \vee \texttt{result-2} \vee \texttt{result-3} \vee \texttt{result-4} \vee \texttt{result-5} \vee \texttt{result-6})$
    - "Is the probability less than 1 that a t-state is reached?"

Using the PRISM model

PRISM actually offers

- `P=? [ F s=7 & d=6 ]`
  - What is the probability that the state (7,6) is eventually reached?
- `R=? [ F s=7 & d=6 ]`
  - What is the *expected* (average) number of steps to reach the state (7,6)?

Using the PRISM model

`http://www.prismmodelchecker.org/tutorial/`

- The first of the PRISM tutorials here (Part 1) explores the model we have been discussing in this lecture and introduces a number of practical features of the PRISM tool.

`http://www.prismmodelchecker.org/manual/`

- Further details of the PRISM modelling language are set out in the PRISM manual which can be read online at this URL. A PDF version is distributed witht the PRISM software.
- PRISM itself is a free download from `prismmodelchecker.org`

Further Reading

A comprehensive set of tutorial material can be found at the PRISM web site, `http://www.prismmodelchecker.org/`. In particular, the 'Documentation' tab gives links -

1. Lectures → probabilisitc model checking, a set of 20 lectures by Dave Parker (`http://www.prismmodelchecker.org/lectures/pmc/`). These cover several different types of probabilistic automaton in some depth. We are mainly concerned with matter in lectures 2-7.
2. This Lectures link also points to three other links to other series of lectures.
3. Tutorial→ a practical tutorial on the use of the PRISM model checking tool which we will be using in lab work.
4. Manual→ a manual of PRISM and the PRISM modelling language.

There is also, of course, a download link for the PRISM model checking tool, which is free and reasonably light-weight.
The textbook Principles of Model Checking by C Baier and J-P Katoen (MIT Press, 2008) is highly recommended. Chapter 10 gives a pretty rigorous introduction to Markov Chains and discrete-time probabilistic model checking.