PRISM for Discrete Time Markov Chains

Andreas Plank

December 7, 2017

Contents

Introduction to Markov Chains

Definition

A stochastic process $X(n)_{n\in\mathbb{N}_0}$ is called a Markov Chain with discrete time if for all i, j, i_0, ..., i_{n-1} $\in$ E

$$\begin{aligned}\mathbb{P}[X(n+1) = j | X(n) = i, X(n-1) = i_{n-1}, ..., X(0) = i_0] \\ = \mathbb{P}[X(n+1) = j | X(n) = i]\end{aligned}$$

where E is the state space of the process $X(n)_{n\in\mathbb{N}_0}$

Introduction to Markov Chains

Definition

The expression

$$\mathbb{P}[X(n+1) = j | X(n) = i] = p_{ij}(n)$$

with $0 \leq p_{ij}(n) \leq 1$ is called the transition probability from i to j at time n

Definition

The matrix containing the transition probabilities

$$\mathbf{P} = (p_{ij})_{i,j \in E} = \begin{pmatrix} p_{11} & p_{12} & \dots \\ p_{21} & p_{22} & \dots \\ \dots & \dots & \dots \end{pmatrix} \tag{1}$$

is called the transition matrix.

Introduction to Markov Chains

Definition

The expression

$$\mathbb{P}[X(n) = i] = \pi_i(n), i \in E, n \in \mathbb{N}_0$$

is called state probability of i∈E at time n.

Definition

A Markov Chain with discrete time has a stationary distribution $\pi_i, i \in E$ if and only if

$$lim_{n \to \infty} \pi_i(n) = \pi_i$$

Example 1

Given a discrete Markov Chain with

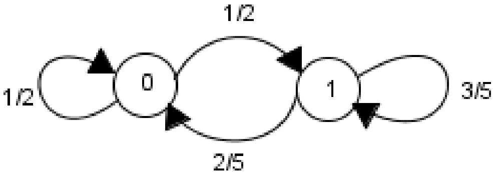
$$
\begin{aligned}
E &= 0, 1 \\
p_{11} &= \mathbb{P}[X(n+1) = 0 | X(n) = 0] = \frac{1}{2} \\
p_{12} &= \mathbb{P}[X(n+1) = 0 | X(n) = 1] = \frac{1}{2} \\
p_{21} &= \mathbb{P}[X(n+1) = 1 | X(n) = 0] = \frac{2}{5} \\
p_{22} &= \mathbb{P}[X(n+1) = 1 | X(n) = 1] = \frac{3}{5}
\end{aligned}
$$

or written as a matrix

$$
\mathbf{P} = (p_{ij})_{i,j \in E} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{2}{5} & \frac{3}{5} \end{pmatrix} \tag{2}
$$

Example 1

The model as state transition diagram



Example1

Computation of the stationary distribution according to theorems for discrete Markov Chains. Solving

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$$
$$\sum_{i \in E} \pi_i = 1$$

yields $\pi_0 = \frac{4}{9} = 0.444\dot{4}$ and $\pi_1 = \frac{5}{9} = 0.555\dot{5}$

Example 1

According model in PRISM

```
dtmc

module example1

s . [0..1] init 0;

[ ] s=0 → 0.5 : (s'=0) + 0.5 : (s'=1);
[ ] s=1 → 0.6 : (s'=1) + 0.4 : (s'=0);

endmodule
```

Explanation of the code

dtmc

describes the type of model we are using

s . [0..1] init 0

initialization of the states

$$[\,]\ s=0 \rightarrow 0.5 : (s'=0) + 0.5 : (s'=1);$$
$$[\,]\ s=1 \rightarrow 0.6 : (s'=1) + 0.4 : (s'=0);$$

defining the transition probabilities

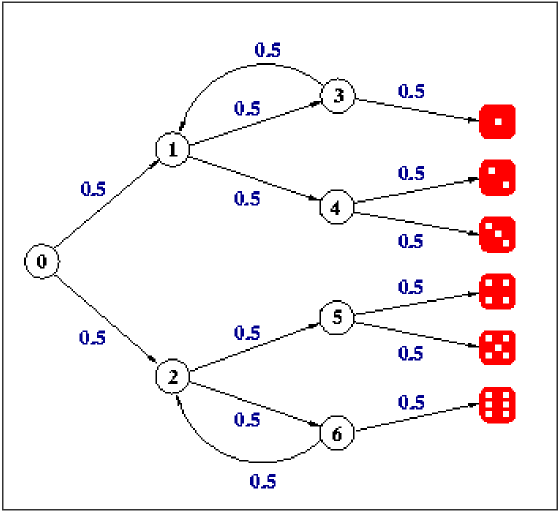
Example 1

Steady state probabilities according to PRISM

```
Starting iterations...

Steady state detected at iteration 8

Iterative method: 8 iterations in 0.00 seconds (average 0.000000, setup 0.00)

Printing transient probabilities in plain text format below:
0:(0)=0.4444444500000001
1:(1)=0.5555555500000001

Time for transient probability computation: 0.0 seconds.
```

Example 2

Given a discrete Markov Chain with according state transition diagram



Example 2

According matrix with transition probabilities

$$\mathbf{P} = (p_{ij})_{i,j \in E} = \begin{pmatrix}
0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \tag{3}$$

Example 2

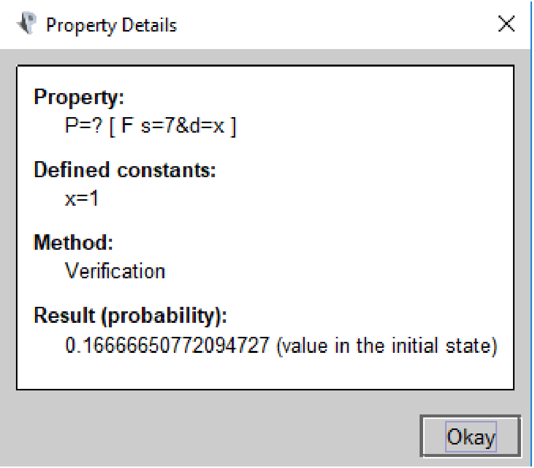
According model in PRISM

```
dtmc

module die

s . [0..7] init 0;
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)

endmodule
```

Example 2

We can check properties of the model using the following code:

$$P = ?[F\ s = 7 \& d = x]$$



Properties

Our model can be analysed by using properties.

The expression

$$P =?[F\ s = 7 \& d = x]$$

gives back the probability that at some point we get into state 7 with a dice roll given by the user.

If we define an explicit value for ?

$$P > 0.5[F\ s = 7 \& d = x]$$

we get a boolean value indicating if the given condition is true.

Properties

The F in

$$P =?[F\ s = 7\&d = x]$$

determines the path property as eventually.

It is also possible to define other path properties

- X: next
- U: until
- F: eventually
- G: always
- W: weak until
- R: release

Properties

It is also possible to compute long time probabilities via steady state operator S

The expression

$$S =?[s = 7 \& d = x]$$

gives back the steady state probability of state 7 with a dice roll given by the user.

Herman's self stabilising algorithm

The last example is about Herman*s self stabilising algorithm. This algorithm contains a network of processes which will return to a "legal" state if they start in an "illegal" state in finite time without outside intervention. The legal states are also called stable states.

In the following example a state is stable if there is only one process that has the same value than the process on its left.

Example:

0 1 0 0 1 0 1 is stable

0 1 1 1 0 1 0 is not stable

Herman's self stabilising algorithm

According model in PRISM

```
dtmc

module process1

x . [0..1] init 1;

[step] (x1=x7) → 0.5 : (x1'=0) + 0.5 : (x1'=1);
[step] !(x1=x7) → (x1'=x7)

endmodule


module process2 = process1[x1=x2, x7=x1] endmodule
module process3 = process1[x1=x3, x7=x2] endmodule
module process4 = process1[x1=x4, x7=x3] endmodule
module process5 = process1[x1=x5, x7=x4] endmodule
module process6 = process1[x1=x6, x7=x5] endmodule
module process7 = process1[x1=x7, x7=x6] endmodule


//formula, for use in properties: number of tokens
//(i.e. number of processes that have the same value as the process to their left)
formula num_tokens = (x1=x2?1:0)+(x2=x3?1:0)+(x3=x4?1:0)+(x4=x5?1:0)+(x5=x6?1:0)+(x6=x7?1:0)+(x7=x1?1:0);


//rewards (tocalculate expecte number of steps)
rewards "steps"
        true : 1
endrewards
```

Herman's self stabilising algorithm

The expression [step] in

```
[step] (x1=x7) → 0.5 : (x1'=0) + 0.5 : (x1'=1);
[step] (x1=x7) → (x1'=x7)
```

is responsible for a simultaneous execution of the two statements.

The expression formula in

```
formula num_tokens = (x1=x2?1:0)+(x2=x3?1:0)+(x3=x4?1:0)+(x4=x5?1:0)+(x5=x6?1:0)+(x6=x7?1:0)+(x7=x1?1:0);
```

is used to increase the reusability of certain expressions.

The expression

```
(x1=x2?1:0)
```

yields 1 if x1=x2 else 0

Herman's self stabilising algorithm

For Properties with a undefined variable x we can define experiments witch create multiple instances of a model. The property

$$P >= 1[F\ num\_tokens = 1]$$

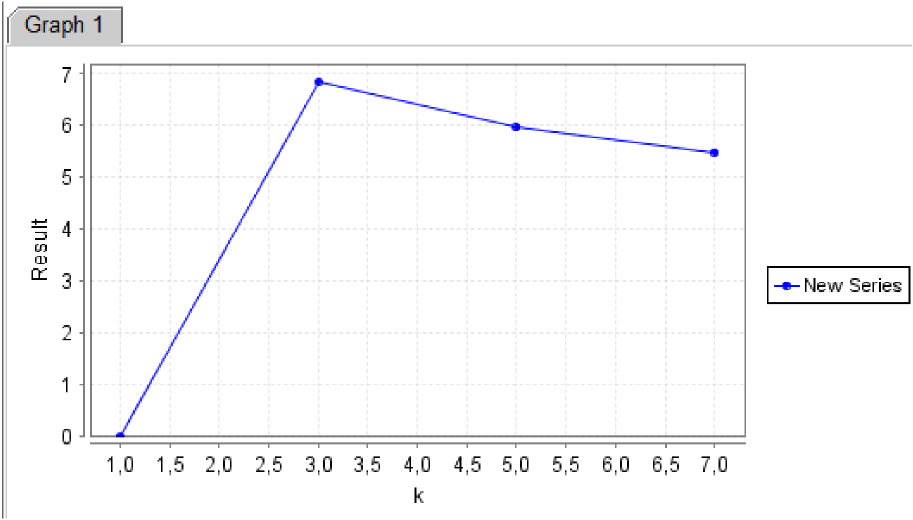
indicates the model reaches a stable state starting with 7 tokens (all processes are 1).

We now adjust our problem to allow stable states with user given amounts of start tokens.

$$filter(max, R =?[F\ num\_tokens = 1], num\_tokens = k)$$

Herman's self stabilising algorithm

As a result for our experiment we get the following graph



Definition

We call c_{ij} the reward(or cost) of a transition from i $\in$ E to j $\in$ E.

The expression

rewards "steps"
    true : 1
endrewards

describes the reward for our model.

PRISM can handle these rewards via the reward-operator R.
The expression

$$R =?[F \; num\_tokens = 1]$$

gives back the steps needed until a stable state is reached

References

- http://www.prismmodelchecker.org/tutorial/
- http://www.prismmodelchecker.org/manual
- [KNP10c] Marta Kwiatkowska, Gethin Norman and David Parker. Advances and Challenges of Probabilistic Model Checking. In Proc. 48th Annual Allerton Conference on Communication, Control and Computing, pages 1691-1698, IEEE Press. Invited paper. October 2010. http://www.prismmodelchecker.org/papers/allerton10.pdf
- [KNP11] Marta Kwiatkowska, Gethin Norman and David Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In Proc. 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of LNCS, pages 585-591, Springer. July 2011. http://www.prismmodelchecker.org/papers/cav11.pdf
- Lecture Markov Chains by Assoz. Univ.-Prof Dr. Dmitry Efrosinin http://www.jku.at/stochastik/content/e140956/e199111

Thank you for your attention