Probabilistic Model Checking of Dekker's Two-Process Mutual Exclusion Protocol

Models, Property Questions, and Graphs

CS490 Capstone Research Notes

October 16, 2026

Abstract

This report explains a simplified three-state PRISM model of Dekker's mutual-exclusion protocol for two processes. Two cyclic versions are analysed: one in which each scheduled process always requests entry, and one in which process 0 requests with probability 0.7 and process 1 with probability 0.3 when scheduled outside the critical section. Both are Markov decision processes (MDPs), because the next enabled process is selected nondeterministically by a scheduler. We translate each property into a concrete research question and plot model-checking results for horizons $K = 0, \ldots, 30$. A separate one-run model stops after both processes have completed one visit; it is used to explain per-run resting reward. Throughout, "resting" means time spent after withdrawing a request, not all time between requesting and entering.

1 What the model represents

Dekker's protocol coordinates two processes that share a critical section. The desired safety condition is that they are never in the critical section simultaneously. The PRISM models use two asynchronous modules, `p0` and `p1`. An enabled command from either module can be selected at a global step; in an MDP, this selection is nondeterministic rather than a 50/50 CPU lottery [1].

Table 1: State variables and their interpretations.

Variable	Meaning
`s0`, `s1`	Local program counters: 0 is outside or finishing exit, 1 is trying or yielding, and 2 is inside the critical section.
`want0`, `want1`	Whether the process currently announces its wish to enter. At local state 1, `want=false` distinguishes a withdrawn/yielding process from a process that is still trying.
`turn`	Tie-breaker. When both want entry, the process whose turn it is not withdraws its flag.
`done0`, `done1`	Present only in the one-run variant. A process sets its flag after leaving the critical section and clearing its request; it cannot request again.

The probabilistic request commands have the form

```
const double request0 = 0.7;
const double request1 = 0.3;
[] s0=0 & !want0 ->
    request0 : (want0'=true) & (s0'=1) +
    (1-request0) : (s0'=0);
```

The 0.7 and 0.3 values apply *after the respective process is scheduled while outside*. They are not probabilities that the CPU schedules process 0 or process 1. The scheduler remains nondeterministic. In the deterministic-request version, a scheduled outside process requests immediately. The cyclic models can repeat visits indefinitely; the one-run model disables a process after its first complete exit.

1.1 What "resting time" counts

The current reward structures define

$$r_0(s) = \mathbf{1}[s_0 = 1 \wedge \neg\text{want0} \wedge \text{turn} = 1], \tag{1}$$

$$r_1(s) = \mathbf{1}[s_1 = 1 \wedge \neg\text{want1} \wedge \text{turn} = 0], \tag{2}$$

$$r_{\text{total}}(s) = r_0(s) + r_1(s). \tag{3}$$

For the cyclic models, K is an integer horizon in *global transitions*, not a number of independent runs. The PRISM expression `F<=K` ϕ asks whether a state satisfying ϕ is reached within K steps; `C<=K` accumulates reward over the first K steps. The suffixes `max` and `min` optimise over scheduler choices [2, 3].

2 Properties: what question does each ask?

Table 2: Reachability and safety questions for the cyclic models. Each row is a valid PRISM property.

Property	Question answered from the initial state
`Pmax=? [ F<=K (s0=1 & want0) ]`	How likely can P0 be to request entry by step K, under the most favourable scheduler?
`Pmax=? [ F<=K (s1=1 & want1) ]`	The same request question for P1.
`Pmax=? [ F<=K s0=2 ]`	What is the largest probability that P0 enters the critical section by K?
`Pmax=? [ F<=K s1=2 ]`	The same entry question for P1.
`Pmax=? [ F<=K (s0=2 | s1=2) ]`	What is the largest probability that at least one process enters by K?
`Pmin=? [ F<=K s0=2 ]`	What entry probability can P0 guarantee despite the least favourable scheduler?
`Pmin=? [ F<=K s1=2 ]`	The same guaranteed-entry question for P1.
`Pmax=? [ F<=K "p0_resting" ]`	How likely can P0 be to yield at least once by K?
`Pmax=? [ F<=K "p1_resting" ]`	The same first-yield question for P1.
`Pmax=? [ F<=K "both_inside" ]`	Can any scheduler produce a mutual-exclusion violation by K?
`Pmax=? [ F<=K "both_yielded" ]`	Can both processes simultaneously be in their withdrawn waiting states by K?

Important: `Rmax` of P0's rest and `Rmax` of P1's rest may use different scheduler strategies. Therefore, their reported values must *not* be added to obtain `Rmax` of total rest. The total-reward property optimises the sum under one scheduler.

Table 3: Reward questions. These are expected values, not sample-path counts.

Property	Question answered
`R{"p0_rest"}max=? [ C<=K ]`	Maximum expected accumulated P0 resting reward during the first K global steps.
`R{"p1_rest"}max=? [ C<=K ]`	The analogous maximum for P1.
`R{"total_rest"}max=? [ C<=K ]`	Maximum expected sum of the two resting rewards during the first K steps.
`R{"total_rest"}min=? [ F "done" ]`	In the one-run model, minimum expected total rest before both finish once.
`R{"total_rest"}max=? [ F "done" ] / 2`	In the one-run model, maximum expected total rest divided by two completed visits: an upper-bound average per process.
`Pmin=? [ F "done" ]`	In the one-run model, do both processes eventually complete with probability one even under the least favourable scheduler?

3 Graphs from PRISM model checking

All plotted points were exported from PRISM 4.10.1 with its explicit engine, using exact finite-horizon model checking rather than path sampling. The underlying numerical data are in `graphs.dat`; the graphs are rendered directly by `pgfplots` in this Overleaf project. Each point uses the initial state and a separate optimisation over schedulers. A curve is therefore *not* a trace of one execution.

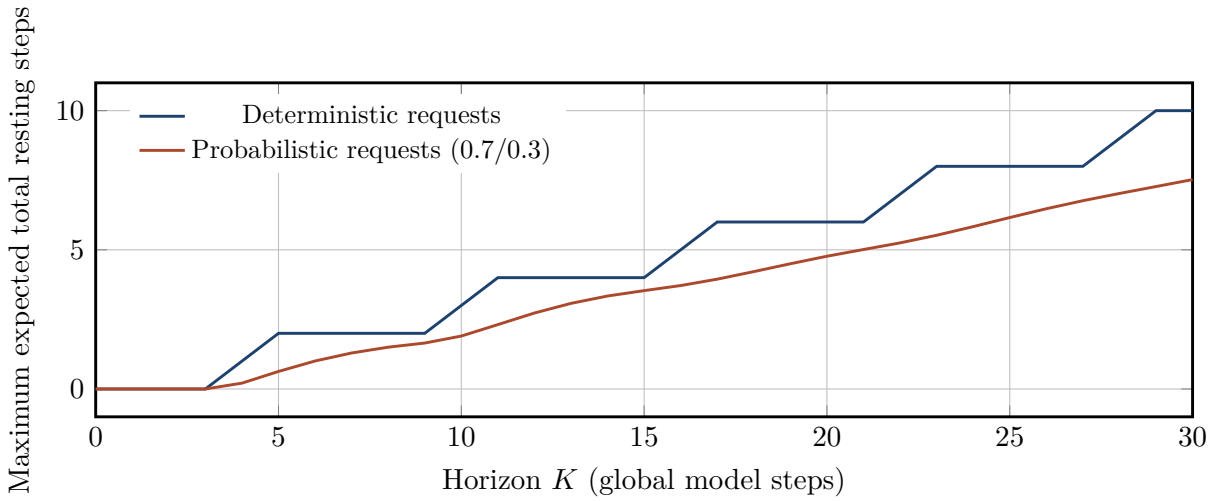


Figure 1: Worst-scheduler expected cumulative rest, $R^{\text{total}}_{\max}[C^{\leq K}]$, for the two cyclic models. At $K = 20$, the values are 6.000 and 4.770, respectively. These are not the average lengths of individual waiting episodes.

4 Results that a graph alone does not establish

The unbounded cyclic-model checks return $P_{\max}(F\,\texttt{both_inside}) = 0$ and $P_{\max}(F\,\texttt{both_yielded}) = 0$ for both request variants. Thus the safety statements hold over all reachable steps in these finite models,

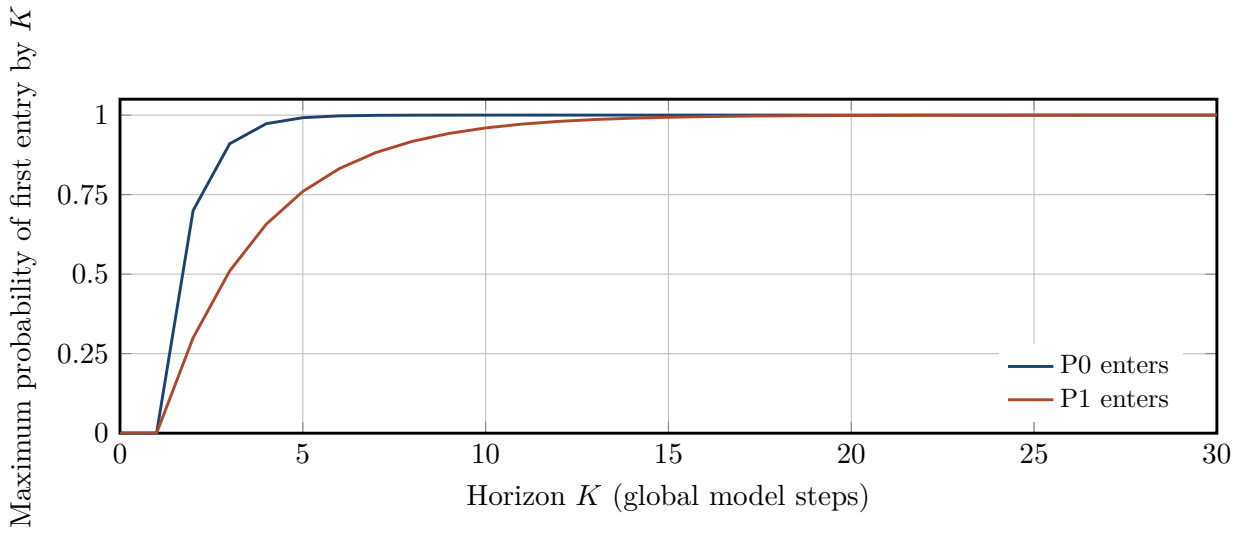


Figure 2: Best-scheduler bounded entry probabilities in the probabilistic-request model. At $K = 10$, they are approximately 0.99998 for P0 and 0.95965 for P1. The corresponding minimum probabilities are 0 at $K = 30$: without a scheduler fairness assumption, neither process is guaranteed a bounded entry time.

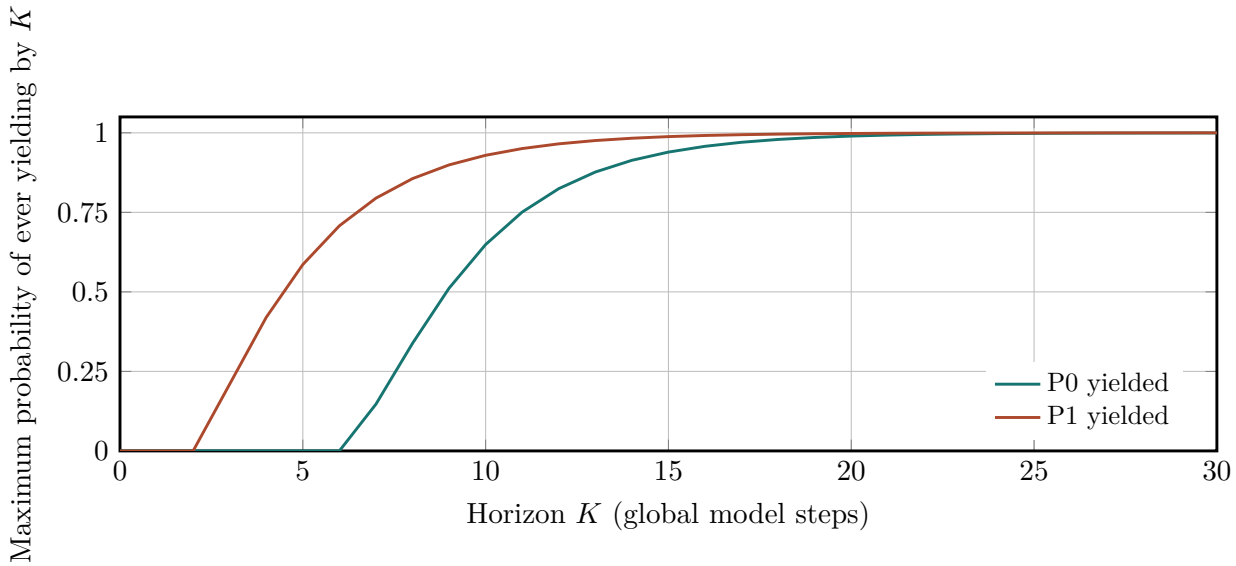


Figure 3: Best-scheduler probability of seeing at least one withdrawn/yielding state. This is a reachability probability, not a duration. The two curves can differ because of request probabilities, the initial `turn=0`, and scheduler optimisation.

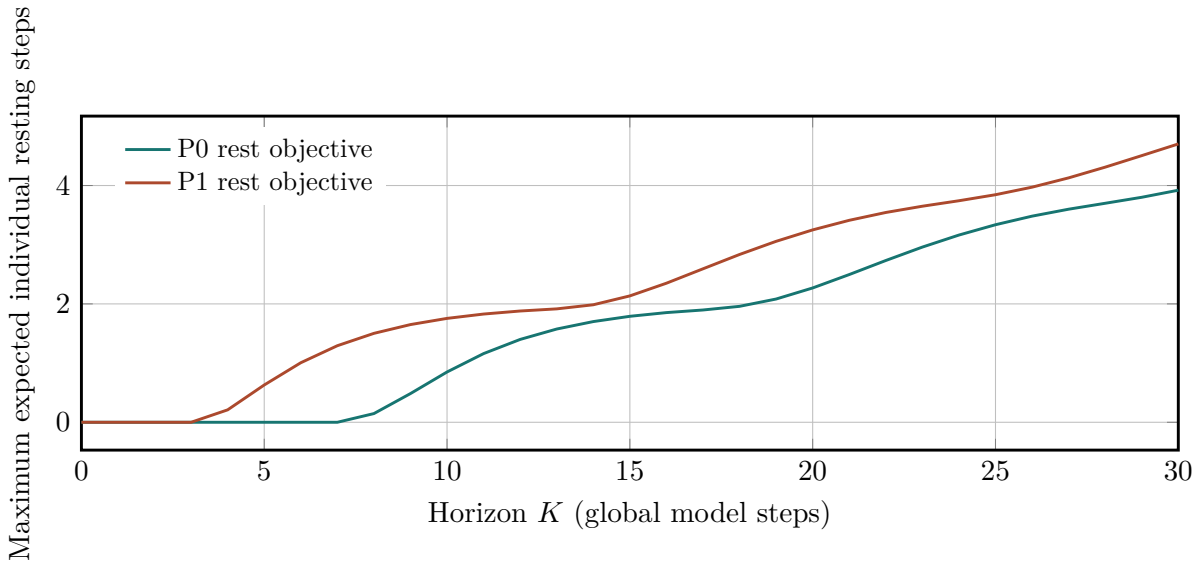


Figure 4: Individual maximum expected cumulative resting rewards in the probabilistic-request model. Each series is optimised separately; their vertical sum is not the total-rest curve in Fig. 1.

not merely through $K = 30$. Conversely, $P_{\min}(F\, s_i = 2) = 0$ for each process in the cyclic models: an adversarial scheduler can indefinitely favour the other repeatedly requesting process. This is a scheduler/liveness limitation, not a mutual-exclusion failure.

The one-run model asks a different question. Each process is disabled after its first complete exit, and `done` means that both have finished. PRISM reports The values near 2 and 1 are numerical

Table 4: One-run results from the probabilistic-request model.

One-run property	Value	Interpretation
$P_{\min}(F\,\texttt{done})$	1	Both eventually finish, even under the least favourable scheduler.
$P_{\max}(F\,\texttt{both_inside})$	0	Mutual exclusion is preserved.
$R^{\text{total}}_{\min}(F\,\texttt{done})$	0	Some scheduling choices avoid any yielding rest.
$R^{\text{total}}_{\max}(F\,\texttt{done})$	≈ 2	Worst-scheduler expected total resting steps.
$R^{\text{total}}_{\max}(F\,\texttt{done})/2$	≈ 1	Worst-scheduler expected rest per completed visit.

model-checking approximations (respectively 1.999996 and 0.999998 with the run's tolerance). They do not mean that every observed run rests for exactly two steps. A run in which P0 completes before P1 requests has total rest 0. The one-run completion result does not contradict the cyclic model's minimum eventual-entry probability of 0: the one-run model disables completed processes, whereas the cyclic model lets them request indefinitely.

5 Scope and next research steps

This is an abstract interleaving model, not a timing model of CPU instructions. A state reward of one is one *model transition* while withdrawn; it is not a second of wall-clock time. Busy-wait reads with no state change are abstracted away. To study the full delay from announcing a request to entering, a separate waiting-time reward should count *all* states in the trying phase, including `want=true`, with a

clearly defined start and stop event. To estimate an ordinary average under a realistic scheduler, specify that scheduling policy explicitly (for example, a fixed random or round-robin scheduler); do not equate PRISM's $R_{\max}$ with Monte Carlo averages. PRISM warns that simulation of an MDP randomly resolves nondeterministic choices, so a simulation mean is not an estimate of the worst-scheduler optimum [5].

Reproducibility

The Overleaf project contains `graphs.dat`, three PRISM model files under `models/`, and separate property files under `properties/`. The cyclic graph data were obtained with PRISM 4.10.1 using $K = 0{:}30$ and the explicit engine. For example, after placing the project files in one directory, the total-rest experiment can be rerun as:

```
prism models/dekker_three_state_prob_07_03.pm \
  properties/cyclic_properties.pctl \
  -prop 12 -const K=0:30 -exportresults stdout -explicit
```

The one-run results use `models/dekker_one_run_prob_07_03.pm` with `properties/one_run_properties.pctl`. Do not apply the `done` property to a cyclic model; that label exists only in the one-run model. PRISM experiments can export every point in a sweep [4].

References

[1] PRISM Manual, "Parallel Composition." `https://www.prismmodelchecker.org/manual/ThePRISMLanguage/ParallelComposition`

[2] PRISM Manual, "The P Operator." `https://www.prismmodelchecker.org/manual/PropertySpecification/ThePOperator`

[3] PRISM Manual, "Reward-based Properties." `https://www.prismmodelchecker.org/manual/PropertySpecification/Reward-basedProperties`

[4] PRISM Manual, "Experiments." `https://www.prismmodelchecker.org/manual/RunningPRISM/Experiments`

[5] PRISM Manual, "Statistical Model Checking." `https://www.prismmodelchecker.org/manual/RunningPRISM/StatisticalModelChecking`