This folder contains PRISM models and a semantic interpreter for a language $\overline{\mathcal{L}}_{QP}$ described in the paper

"Quantitative Programming and Continuous-Time Markov Chains"

Author: Eneia Nicolae Todoran (SYNASC 2023 submission 26)

Quantitative programming is a new computing paradigm which provides support for the formal verification of concurrent programs with large state spaces. The state space of programs is partitioned into bisimulation equivalence classes, and concurrent programs are translated into corresponding finite state models which are analyzed using model checking techniques. In this article we employ probabilistic model checking techniques.

For formal verification, in this paper we use continuous-time Markov chains (CTMCs). The paper offers formal verification experiments performed using PRISM probabilistic model checker (www.prismmodelchecker.org).

- The PRISM CTMC model described in Section IV is available in this folder in the file `epollingsystem-ctmc.prism`.
- The identifiers `nmax`, `vwms` and `vw1` are the PRISM counterparts of the constant $\nu$, the variables $v^{\sim}_{wms}$ and $v^{\sim}_{w1}$, respectively, declared in the program under verification presented in Section III. As explained in the paper, the variables $v^{\sim}_{wms}$ and $v^{\sim}_{w1}$ are control variables used to monitor the data variables $w_{ms}$ and $w_1$, respectively. The corresponding CTMC model is built taking into consideration the execution rates of the program statements.
- Since the cardinality of the multiset[1] stored in variable $w_{ms}$ can increase indefinitely, the state space of the program under verification is (conceptually) infinite. By partitioning the state space of programs into bisimulation equivalence classes, the approach presented in the paper enables the formal verification of concurrent programs with large (even infinite) state spaces.[2]
- Section IV offers PRISM experiments that can be performed using the model contained in file `epollingsystem-ctmc.prism` and the properties contained in file `epollingsystem-ctmc.props`. These experiments are briefly described below.

[1]The cardinality of a multiset is the sum of the multiplicities of all elements stored in the multiset.

[2]We say "(conceptually) infinite" because the state space of any real computer is finite. However, (although the lenght of list $w_{ms}$ is $\leq \nu$) the cardinality of the multiset that is stored in variable $w_{ms}$ can increase indefinitely. Hence, the number of different multisets that can be stored in $w_{ms}$ is not bounded; the only limitation is given by the memory size of the computer on which the program is running.

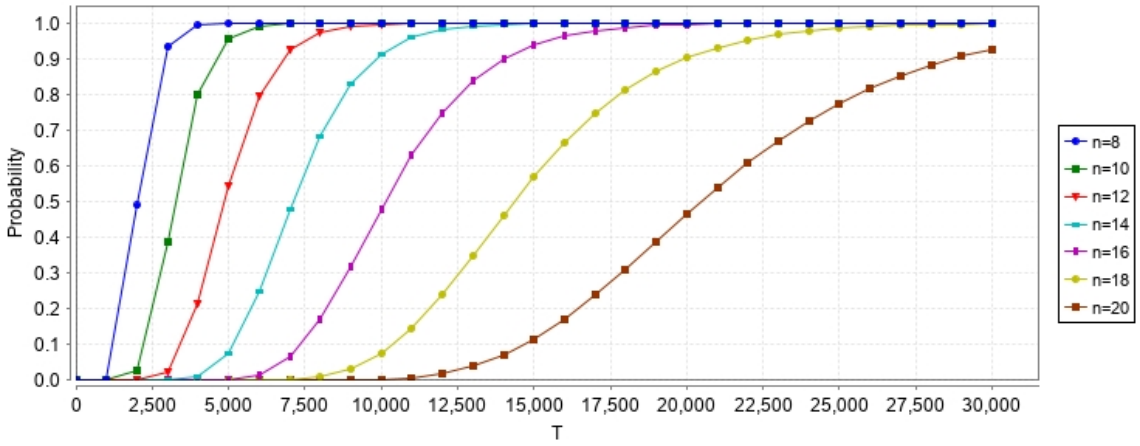


Figure 1: The probability of the multiset $w_{ms}$ eventually containing `n` distinct elements within `T` time units

- In the PRISM experiments presented below we put $\nu = 25$.
- We use the following constants:

```
const int T;
const int n;
const nmax = 25;
```

1. Using the PRISM property specification language, the probability of the multiset $w_{ms}$ eventually containing `n` distinct elements within `T` time units can be specified by:

   ```
   P=? [ F<=T  (vwms = n)]
   ```

   Figure 1 presents a PRISM experiment for this property. In this experiment `n` ranges from `8` to `20` with step `2`, and `T` ranges from `0` to `30000` with step `1000`.

2. By using the PRISM property specification language, the transient probability that at time instant `T` the element which is generated by station $S1$ (and stored in variable $w_1$) is already in the multiset $w_{ms}$ can be specified as follows:

   ```
   P=? [F[T,T] (vw1>0) ]
   ```

   Figure 2 shows a PRISM experiment based on this property, where `T` ranges from `0` to `40000` with step `2500` (`n` is fixed).

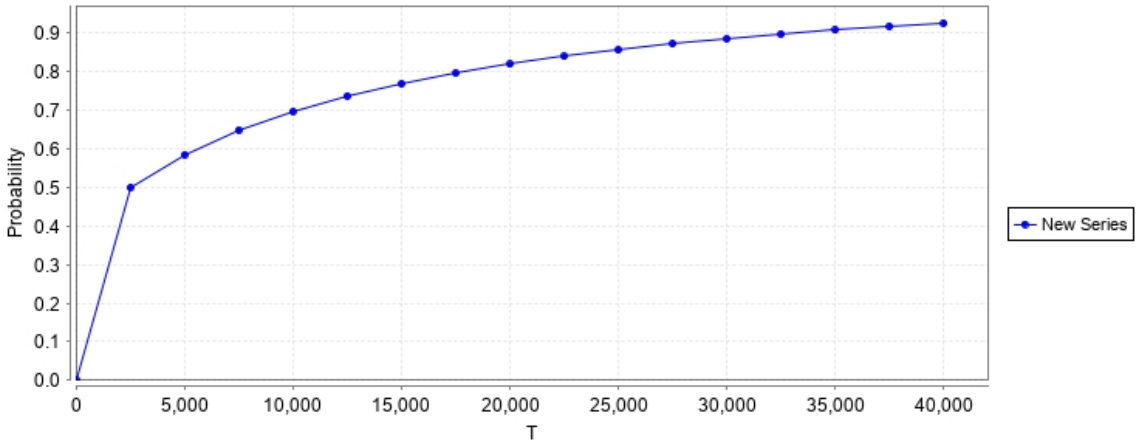


Figure 2: The transient probability that at time instant `T` the element which is generated by station $S1$ (and stored in variable $w_1$) is already in the multiset $w_{ms}$

- Further PRISM experiments are provided in the (sub) folder `furtherExperiments` (based on files `epollingsystem2-ctmc.prism` and `epollingsystem2-ctmc.props`). Due to space limitations, the PRISM experiments provided in the (sub) folder `furtherExperiments` are not included in the paper.
- The file `Lqp.hs` contains a semantic interpreter for the language $\overline{\mathcal{L}}_{QP}$. The semantic interpreter for $\overline{\mathcal{L}}_{QP}$ is designed with continuations and is implemented in Haskell (`www.haskell.org`). The language $\overline{\mathcal{L}}_{QP}$ is described in Section II of the paper.
  - The semantic interpreter can be used to run the $\overline{\mathcal{L}}_{QP}$ program presented in Section III (further explanations are provided as comments in file `Lqp.hs`).