INFAMY: An Infinite-State Markov Model Checker*

E. Moritz Hahn, Holger Hermanns, Björn Wachter, Lijun Zhang

Universität des Saarlandes, Saarbrücken, Germany
{emh,hermanns,bwachter,zhang}@cs.uni-sb.de

Abstract. The design of complex concurrent systems often involves intricate performance and dependability considerations. Continuous-time Markov chains (CTMCs) are a widely used modeling formalism, where performance and dependability properties are analyzable by model checking. We present `INFAMY`, a model checker for arbitrarily structured infinite-state CTMCs. It checks probabilistic timing properties expressible in continuous stochastic logic (CSL). Conventional model checkers explore the given model exhaustively, which is often costly, due to state explosion, and impossible if the model is infinite. `INFAMY` only explores the model up to a finite depth, with the depth bound being computed *on-the-fly*. The computation of depth bounds is configurable to adapt to the characteristics of different classes of models.

1 Introducing INFAMY

Continuous-time Markov chains (CTMCs) are widely used in performance and dependability analysis and biological modeling. Properties are typically specified in continuous stochastic logic (CSL) [1], a logic inspired by CTL. In CSL, the until operator is equipped with a time interval to express properties such as: *"The probability to reach a goal within 2 hours while maintaining a probability of at least 0.5 of communicating periodically (every five minutes) with a base station, is at least 0.9"* via $\mathcal{P}_{\geq 0.9}\left(\left(\mathcal{P}_{\geq 0.5}\Diamond^{\leq 5}\text{communicate}\right)\mathcal{U}^{\leq 120}\text{ goal}\right)$. CSL model checking amounts to analysis of the transient (time-dependent) probability vectors [1], typically carried out by *uniformization*, where the transient probability is expressed by a weighted infinite sum (weights are given by a Poisson process). The standard methodology in CSL model checking is to truncate the infinite sum up to some pre-specified accuracy [2]. Outside the model checking arena, ideas have been developed [3,4,5] which not only truncate the infinite sum, but also the matrix representing the system, which admits transient analysis of CTMCs with large or even infinite state spaces, provided they are given implicitly in a

* This work is supported by the NWO-DFG bilateral project VOSS, by the DFG as part of the Transregional Collaborative Research Center SFB/TR 14 AVACS and the Graduiertenkolleg "Leistungsgarantien für Rechnersysteme", and has received funding from the European Community's Seventh Framework Programme under grant agreement n^o 214755.

modeling language. Harvesting and improving on these ideas, INFAMY is the first CSL model checker based on *truncation.* The underlying truncation technique was developed in [6]. Besides truncation, INFAMY features SPIN-like [7] state space exploration, and supports models given in a high-level description language.

Several other CSL model checkers exist, see [8] for an overview. Among them, PRISM [9] is a probabilistic model checker which uses advanced techniques for model representation and model checking for several stochastic model types. The model description language of INFAMY is based on the one of PRISM, but allows for infinite variable domains, while PRISM is restricted to finite models. Thus the tools are incomparable for infinite models. For several very large finite models, INFAMY is competitive with PRISM, as evident from Section 3. Model checkers based on discrete-event simulation [10,11,12,9] could, in principle, also analyze models with implicitly infinite state space, however they have not yet been applied to such models, and thus we cannot compare with them.

INFAMY is available at `http://depend.cs.uni-sb.de/~emh/infamy`.

2 Truncation-based Model Checking

INFAMY reads models in a guarded-command language extending the one of PRISM. The semantics of a model is a CTMC in which each state is a valuation of model variables. Initial states are specified by an expressions over model variables. The rest of the description consists of commands. Each command comprises a guard and a set of rates. Each rate is associated with an update formula. If a state fulfills the guard of a command, this state has a rate to each state obtained by the respective update formula. Contrary to PRISM, we allow variables with infinite range. Properties are specified in the time-dependent fragment of CSL which involves the Boolean operators, timed until and next operators.

Using truncation, we compute a finite submodel which is sufficient for checking the validity of a property given in time-dependent CSL. This is done by descending into the CSL formula and at the same time exploring the model subject to the different time bounds given in the subformulas as well as the rates occurring in the model (details can be found in [13]). Beginning with a set of start states – either the initial states or, when descending into nested formulas, states determined by a subformula, the model is explored breadth-first up to a certain depth. The lower the depth the higher the error in the probabilities computed during model checking. This error needs to be kept below a user-defined absolute error ε for all methods described in the following. As illustrated in Figure 1, our technique proceeds layer by layer exploring states with increasing *depth*, i.e. minimal distance from the start states, and estimates the error on the fly to determine if the depth is already sufficient. In the next paragraph, we discuss three error-estimation methods implemented in INFAMY, ranging from very precise and expensive to faster techniques with larger overestimation. An

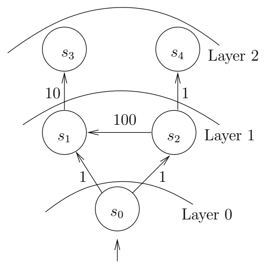

Fig. 1. Layers.

interesting trade-off arises between finding the smallest depth possible and the cost of error estimation.

Finite state projection (FSP). Munsky and Khammash [5] consider transient properties in the context of infinite-state biological models, including the one resulting from *Chemical Master Equation* problems. They build the CTMC incrementally in layers. Whenever adding a layer, they estimate the error by computing the probability of reaching the outermost layer from the initial states. FSP can find a minimal truncation point. However, as the exploration of new layers involves a stochastic analysis, relative computational cost of error estimation can be high.

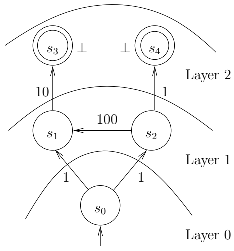

Fig. 2. FSP.

Uniform. While FSP performs error estimation on the full model, we recently developed the *Uniform* method [13], which abstracts the model to a chain, as illustrated in Figure 3. The probability of reaching the last state of the chain is an upper bound for the error. Compared to FSP, this admits a much faster error estimation, where the chain needs not even be constructed explicitly. We can just consider the maximal sum of rates $\lambda_{\max}$ leaving a state into the next layer. Using the Fox-Glynn algorithm [2], a right truncation point is computed by starting the algorithm with a value of $\lambda_{\max} \cdot t$ where t is the time-bound of the current subformula. Whenever a state with a higher sum is seen, we adjust the right truncation point.

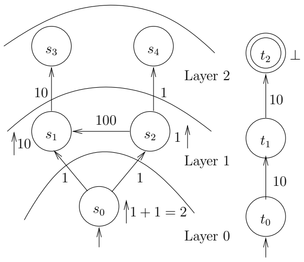

Fig. 3. Uniform method.

Layered. We consider more fine-grained chains than in the Uniform method that take into account maximal rates for individual layers, leading to a chain in which each edge is labeled with the maximal rate of the corresponding layer. This is depicted in Figure 4. Using the vector, we can construct a birth process in which the probability of reaching the last state within time bound t is larger than leaving the finite truncation. By building the birth process until we see a probability which is low enough, we obtain a method which may explore less layers than the Uniform one if the rates of the layers vary. However, we have to compute the probabilities within the birth process each time we explore a new layer. Thus model exploration may be more time-consuming. We call this the Layered method.

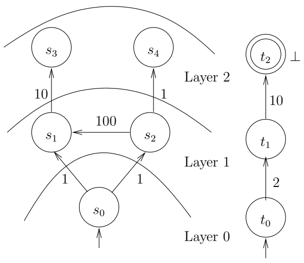

Fig. 4. Layered Method.

In all cases, after the exploration, usual CSL model checking is applied on the resulting submodel.

3 Results

We compare different configurations of `INFAMY`, i.e., FSP, Uniform and Layered method. Further, for finite models, we assess the effectiveness of truncation by comparing with `PRISM`. We denote the number of states on which model checking is performed by n. For `INFAMY`, this refers to the finite truncation, and, for `PRISM`, this refers to the full model. Where appropriate, we denote multiples of 1000 by "K". For all configurations, the probabilities listed in the experiments are exact up to $\varepsilon = 10^{-6}$ (or 10^{-12}, if the probability is close to 10^{-6}). The time column is of the form xx/yy, where xx is construction and yy is model check time. A full description of the models is given in [13].

Quasi-Birth-Death Processes [14]. We consider a system consisting of processors and an infinite queue for storing jobs. We compute the probability that, given all processors are busy and there are no jobs in the queue, within $t = 10$ time units a state is reached in which all processors are idle and the queue is empty. Results for different rates of incoming jobs λ are given below. The depth needed grows linearly with λ. Thus, the performance of the FSP method suffers from the high cost of repeated transient analysis.

λ	Uniform			Layered			FSP			prob.
	depth	time (s)	n	depth	time (s)	n	depth	time (s)	n	
40	609	1.1/0.2	2,434	533	1.1/0.2	2,130	473	28.1/0.2	1,890	4.21E-04
60	846	1.1/0.3	3,382	754	1.1/0.2	3,014	694	66.0/0.2	2,774	1.25E-04
80	1,077	1.1/0.3	4,306	971	1.1/0.3	3,882	911	125.5/0.3	3,642	5.26E-05
100	1,305	1.1/0.4	5,218	1,187	1.1/0.4	4,746	1,127	209.4/0.4	4,506	2.69E-05

Protein Synthesis [15]. We analyze a model of protein synthesis considering the property that, within time t but later than 10 time units, a state is reached, in which 20 or more proteins exist and the synthesizing gene is inactive. Overall, the FSP method is the most efficient here, because a rather low number of layers is relevant. In terms of model construction time, the Uniform method is best here.

t	Uniform			Layered			FSP			prob.
	depth	time (s)	n	depth	time (s)	n	depth	time (s)	n	
100	973	1.1/0.1	1,945	756	1.1/0.1	1,511	30	1.1/0.0	59	1.03E-03
500	3,636	1.1/6.0	7,271	3,308	1.2/5.0	6,615	35	1.2/0.0	69	4.39E-02
1000	6,830	1.1/39.3	13,659	6,420	1.4/34.7	12,839	36	1.4/0.0	71	9.99E-02
2000	13,103	1.1/276.9	26,205	12,577	2.3/255.3	25,153	37	1.7/0.0	73	2.02E-01

Grid-World Robot [11]. We consider a grid world in which a robot moves from the lower left to the upper right corner sending signals to a station. A janitor moves around and may block the robot for some time. In contrast to [11], the janitor can leave the $N \times N$ field of the robot, which leads to an infinite state space. We check a nested formula of the form mentioned in the introduction, namely whether the robot moves with probability 0.9 to the upper right corner within 100 time units while periodically communicating with the base. We give results for different N. Using the FSP method is advantageous, as each layer of the model contains quite a large number of states, and FSP only needs to explore a fraction of the layers compared to the other methods. As seen from the "prob." column, for $N = 2, 3$ the property holds, while, for $N = 4$, it does not. The reason for the decreasing probability is that the distance the robot has to travel increases with n. Thus the probability decreases that the robot will complete its travel in time.

An implicit representation of the edges of the model used to handle the large number of transitions. However, this increases the runtime in this example.

N	Uniform			Layered			FSP			prob.
	depth	time (s)	n	depth	time (s)	n	depth	time (s)	n	
2	905	15/4,479	9,807K	570	6/1,667	3,885K	107	1,417/37	135K	1.000
3	905	24/8,135	16,308K	571	11/2,979	6,475K	96	1,681/56	177K	0.902
4	905	33/13,323	22,781K	571	14/5,099	9,034K	98	2,754/99	253K	0.898

Workstation cluster [16]. We consider the *dependability of a fault-tolerant workstation cluster*, a model with finite state space. For the model with 512 workstations, we compute the probability that the QoS drops below minimum quality within one time unit, and compare with `PRISM`. Results are given for the fastest configuration of `PRISM` (sparse engine), and `INFAMY` (FSP) respectively.

The state space explored by `PRISM` has depth 1029 and includes 9.4 million states. Up to $t = 20$, `INFAMY` with FSP is faster than `PRISM`. However, for larger time bounds, the model construction time dominates, since for each layer the error estimate is recomputed. As observed by Munsky [5], this can be alleviated by adding more than one layer at each step. We consider a variant in which we double the number of layers we add per step, thus computing an error estimate every $1, 2, 4, 8, \ldots$ layers. We call this configuration *FSP exponential*. It is consistently the fastest method for $t \leq 50$, as shown in the last column of the table.

t	PRISM	FSP			FSP exponential			prob.
	time (s)	depth	time (s)	n	depth	time (s)	n	
5.0	4.8/147.3	37	7.1/0.4	23K	64	3.1/1.5	701K	1.21E-06
10.0	5.8/190.5	52	25.0/1.4	46K	64	4.4/2.4	701K	3.79E-06
20.0	6.0/315.3	80	209.1/7.8	111K	128	26.9/20.2	289K	1.01E-05
30.0	5.6/365.6	104	547.5/15.1	190K	128	38.7/28.3	289K	1.68E-05
50.0	5.4/502.5	147	2,610.5/46.2	382K	256	255.8/182.0	1,167K	3.04E-05

Tandem Queueing Network [17]. This model consists of two interconnected queues with capacity c. We consider the probability that the first queue becomes full before time 0.23. Larger time bounds are left out, since then this probability is one. The performance results for `INFAMY` using configuration Layered are better, while for large time bound ($t \geq 1$) the whole model will be explored thus `PRISM` will perform better.

c	PRISM			Layered			prob.
	depth	time	n	depth	time	n	
511	1,535	3.7/49.7	523,776	632	6.8/7.5	235,339	3.13E-02
1023	3,071	13.7/380.3	2,096,128	1,167	3.6/49.6	714,306	4.24E-03
2047	6,143	69.3/3,068.3	8,386,560	2,198	10.0/297.8	2,449,798	9.87E-05
4095	12,287	560.9/31,386.5	33,550,336	4,209	27.4/2,889.8	8,899,113	7.06E-08

4 Conclusion and Future Work

INFAMY enables model checking of infinite models and, for certain finite models, is competitive with the leading model checker PRISM in time and memory usage. As observed in our experiments, the appropriate truncation method strongly depends on the model and property under consideration. Therefore, INFAMY allows the user to select the method that fits best. We plan to implement heuristics that choose the estimation method automatically. Further, the approach implemented in INFAMY is applicable to richer languages than PRISM, e.g., languages that support dynamic process creation, as in the stochastic π-calculus.

References

1. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.K.: Model-Checking Continuous-Time Markov Chains. ACM Trans. Comput. Log **1** (2000) 162–170
2. Fox, B.L., Glynn, P.W.: Computing Poisson Probabilities. Commun. ACM **31** (1988) 440–445
3. Grassmann, W.K.: Finding Transient Solutions in Markovian Event Systems Through Randomization. In: NSMC'91. (1991) 357–371
4. van Moorsel, A.P.A., Sanders, W.H.: Adaptive Uniformization. Communications in Statistics - Stochastic Models **10** (1994) 619–647
5. Munsky, B., Khammash, M.: The Finite State Projection Algorithm for the Solution of the Chemical Master Equation. Journal of Chemical Physics **124** (2006)
6. Zhang, L., Hermanns, H., Hahn, E.M., Wachter, B.: Time-Bounded Model Checking of Infinite-State Continuous-Time Markov Chains. In: ACSD. (2008) 98–107
7. Holzmann, G.J.: The Model Checker SPIN. IEEE Trans. Software Eng. **23** (1997) 279–295
8. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model checking meets performance evaluation. SIGMETRICS Performance Evaluation Review **32** (2005) 10–15
9. Hinton, A., Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: A Tool for Automatic Verification of Probabilistic Systems. In: TACAS. (2006) 441–444
10. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate Probabilistic Model Checking. In: VMCAI. (2004) 73–84
11. Younes, H.L.S.: Ymer: A statistical model checker. In: CAV. (2005) 429–433
12. Zapreev, I.S.: Model Checking Markov Chains: Techniques and Tools. PhD thesis, University of Twente, Enschede, The Netherlands (2008)
13. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: Time-Bounded Model Checking of Infinite-State Continuous-Time Markov Chains. Technical Report No. 47, SFB/TR 14 AVACS (2009) To appear in Fundamenta Informaticae.
14. Katoen, J.P., Klink, D., Leucker, M., Wolf, V.: Three-Valued Abstraction for Continuous-Time Markov Chains. In: CAV. (2007) 311–324
15. Gross, P.J.E., Peccoud, J.: Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. Proc. Natl. Acad. Sci. USA **95** (1998) 6750–6755
16. Haverkort, B.R., Hermanns, H., Katoen, J.P.: On the Use of Model Checking Techniques for Dependability Evaluation. In: SRDS. (2000) 228–237
17. Hermanns, H., Meyer-Kayser, J., Siegle, M.: Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In Plateau, B., Stewart, W., Silva, M., eds.: NSMC. (1999) 188–207