Explicit Model Checking of Very Large MDP using Partitioning and Secondary Storage*

Arnd Hartmanns Holger Hermanns
Saarland University – Computer Science, Saarbrücken, Germany

1 Introduction

The applicability of model checking to realistic safety- or performance-critical systems is limited by the state space explosion problem: The number of states of a model grows exponentially in the number of variables and parallel components, yet they have to be represented in limited computer memory in some form. Probabilistic model checking is particularly affected due to its additional numerical complexity. Several techniques are available to stretch its limits, including symbolic probabilistic model checking [1], partial order [2] and confluence reduction [23], and abstraction and refinement [15]. The common theme of these approaches is that they aim at reducing the state space or its representation such that it fits, in its entirety, into the main memory of the machine used for model checking. An alternative is to store this data on secondary storage such as hard disks or solid state drives and only load small parts of it into main memory when and as needed. This is attractive due to the vast difference in size between main memory and secondary storage: Typical workstations today posses on the order of 4-8 GB of main memory, but easily 1 TB or more of hard disk space. Moreover, with the advent of dynamically scalable cloud storage, virtually unlimited off-site secondary storage has become easily accessible. In this presentation, we outline a new approach to secondary storage-based model checking of the probabilistic-nondeterministic model of Markov decision processes (MDP). For conciseness, we refer to main memory as *memory* and to any kind of secondary storage as *disk* from now on.

Any disk-based approach to model-check models whose state spaces do not entirely fit into memory must solve two tasks: State space *exploration*, the generation and storage on disk of a representation of the reachable part of the state space, and the disk-based *analysis* to verify or compute the given properties of interest based on this representation. The core challenge is that the most common type of secondary storage, magnetic hard disks, exhibits extremely low random-access performance, yet standard memory-based methods for exploration and analysis perform large numbers of random accesses to the state space.

Previous Work A number solutions that reduce the amount of random accesses during exploration have been proposed in the literature. These fall into three broad categories: 1) Exploiting the layered structure of BFS by keeping only the current BFS layer in memory while delaying duplicate detection w.r.t. previous layers until the current one has been fully explored [7, 21]; 2) partitioning the state space according to some given or automatically computed partitioning function over the states and then loading only one partition into memory at a time in an iterative process [3, 11]; and 3) treating memory purely as a cache for a disk-based search, but using clever hashing and hash partitioning techniques to reduce and sequentialise disk accesses [13]. Exploration can naturally be combined on-the-fly with checking for the

*This work is supported by the EU 7th Framework Programme under grant agreements 295261 (MEALS) and 318490 (SENSATION), the DFG Transregional Collaborative Research Centre SFB/TR 14 AVACS, and by the CAS/SAFEA International Partnership Program for Creative Research Teams.

Submitted to:
QAPL 2015

reachability of error states, and methods to perform on-the-fly verification of liveness and LTL properties exist [4, 8, 10]. However, the analysis of other logics, such as CTL model checking with satisfaction sets, and other models, such as probabilistic model checking of MDP with value iteration, inherently require the entire explored state space for a dedicated analysis step following exploration.

Previous work on disk-based probabilistic model checking mainly focuses on the analysis phase. For purely stochastic Markovian models, i.e. in the absence of nondeterminism, classical block-iterative solution methods [22] can be used with disk-based (sparse) matrix representations of the model by loading into memory and analysing one block at a time (plus those that it depends on) iteratively until the method has converged for all blocks. Implementations can be divided into *matrix-out-of-core* and *complete out-of-core* approaches [20]. In the former [6], the vector of values being iteratively computed for the states of the model is still kept in memory in its entirety. This similar to how PRISM [16] uses binary decision diagrams in its "HYBRID" engine for the model only, while both model and values are represented symbolically in its "MTBDD" engine. For the nondeterministic-probabilistic model of MDP that we are concerned with, the default scalable analysis algorithm used in model checking is value iteration, an iterative fixpoint method that updates the values of each state based on a function over the values of its immediate successors until all changes remain below a given (relative or absolute) error. We are aware of only one explicitly disk-based approach to value iteration, which associates the values to the transitions instead of the states and is based on sequentially traversing two files containing the transitions that have been externally sorted by source and target states in each iteration [9].

2 Our Approach

The correctness of value iteration depends neither on the order in which the updates are performed nor on how many updates a state receives in one iteration. This fact can be exploited to improve its performance by taking the graph structure of the underlying model into account to perform more updates for "relevant" states in a "good" order [5]. However, more generally, this means that value iteration can also be performed in a block-iterative manner. In this presentation, we show that the state space partitioning approach to exploration combines very well with a block-iterative variant of value iteration to deliver a practical implementation of disk-based MDP model checking.

Based on a given partitioning function, our approach is to first explore the partitions of the state space using an explicit state representation while directly streaming a sparse matrix-like representation in a very compact "inverse sequential" format to disk, in a separate file for each partition. Only the initial states of a partition, i.e. the targets of cross edges from other partitions, are stored in an explicit representation on disk, too. When exploration is completed, the stored partitions are analysed using a block-based variant of value iteration, i.e. an outer loop iterates over the partitions on disk, for each of which value iterations are performed until convergence in an inner loop. The outer loop terminates when no more changes have been observed in a whole iteration. For each partition, this only requires its sparse matrix-like representation, the vector of (previously computed) values, and the values for the initial states of its successor partitions to be loaded into memory. All read and write operations on the files we generate on disk are sequential. We can thus easily add compression, which in our experiments reduces the amount of disk space needed by a factor of 10 without affecting overall runtime. The entire approach has been implemented as an extension of the mcsta tool [12] of the MODEST TOOLSET [14]. The implementation currently supports the computation of reachability probabilities and expected accumulated reward values. To the best of our knowledge, mcsta is at this point the only publicly available tool for disk-based model checking of MDP.

model		states	in-memory (mcsta)		disk-based (mcsta)			symbolic (PRISM MTBDD)	
N	K		time	memory	partitions	time	memory	time	memory
3	4	1 460 287	25 s	374 MB	12	32 s	243 MB	1 311 s	595 MB
	5	12 070 354	251 s	2 318 MB	15	296 s	667 MB	10 174 s	701 MB
	6	84 856 004	$>$ 4 GB		18	2 594 s	3 019 MB	$>$ 12 h	
4	3	8 218 017	179 s	1 890 MB	12	211 s	795 MB	5 467 s	737 MB
	4	133 301 572	$>$ 4 GB		16	3 753 s	3 159 MB	$>$ 12 h	

Table 1: Analysis performance for the CSMA/CD benchmark

3 Results and Conclusions

We have evaluated our approach and implementation on a number of existing case studies from the PRISM benchmark suite [17]. The largest model we consider is a version of the WLAN case study using the original, unscaled timing parameters of the physical layer (including the maximum send time of 15 717 time units). Its reachable state space contains almost 1.2 billion states and can be explored and analysed by mcsta in less than half a day using no more than 4 GB of memory and 10 GB of disk space on standard laptop hardware. In Table 1, we show the performance on the CSMA/CD benchmark, which can be scaled using model parameters N and K from small instances that can be checked in memory to very large state spaces. The reported time and memory usage is for exploration plus the computation of one reachability probability and two expected accumulated reward values. We compare with PRISM's fully symbolic MTBDD engine, which is the only publicly available alternative implementation we know of that can handle these very large state spaces (and expected reward properties).

Our technique is particularly efficient for the analysis of time-bounded properties on real-time extensions of MDP such as probabilistic timed automata [19]. These models can be converted to equivalent MDP using the digital clocks approach [18]. Since a time bound (or deadline) is not equivalent to counting steps in the MDP, it must be encoded with a global clock variable in the model that is never reset. This leads to severe state space explosion even for moderate deadlines. However, partitioning according to the values of the new global clock results in a linear and terminating partition graph, which is the best case for our technique. Consequently, the overhead of using the disk is very small and the enormous state space explosion caused by the time bounds is in effect neutralised completely.

In summary, the technique that we present combines the state space partitioning approach from disk-based search with a block-iterative variant of value iteration based on a very compact sparse matrix-like representation of the partitions on disk. It is a complete out-of-core method, and shows good performance on large case studies from the literature. The same technique can be readily used for other graph fixpoint-based model checking algorithms, for example CTL model checking with satisfaction sets. We plan to add methods to automatically select and refine the partitioning function [11] to increase usability.

References

[1] Luca de Alfaro, Marta Z. Kwiatkowska, Gethin Norman, David Parker & Roberto Segala (2000): *Symbolic Model Checking of Probabilistic Processes Using MTBDDs and the Kronecker Representation*. In: *TACAS, LNCS* 1785, Springer, pp. 395–410.

[2] Christel Baier, Pedro R. D'Argenio & Marcus Größer (2006): *Partial Order Reduction for Probabilistic Branching Time*. *Electr. Notes Theor. Comput. Sci.* 153(2), pp. 97–116.

[3] Tonglaga Bao & Michael D. Jones (2005): *Time-Efficient Model Checking with Magnetic Disk*. In: *TACAS*, pp. 526–540.

[4] Jiri Barnat, Lubos Brim & Pavel Simecek (2007): *I/O Efficient Accepting Cycle Detection*. In: *CAV*, *Lecture Notes in Computer Science* 4590, Springer, pp. 281–293.

[5] Peng Dai & Judy Goldsmith (2007): *Topological Value Iteration Algorithm for Markov Decision Processes*. In: *IJCAI*, pp. 1860–1865.

[6] Daniel D. Deavours & William H. Sanders (1997): *An Efficient Disk-Based Tool for Solving Very Large Markov Models*. In: *Computer Performance Evaluation, LNCS* 1245, Springer, pp. 58–71.

[7] Giuseppe Della Penna, Benedetto Intrigila, Enrico Tronci & Marisa Venturini Zilli (2002): *Exploiting Transition Locality in the Disk Based Mur phi Verifier*. In: *FMCAD, Lecture Notes in Computer Science* 2517, Springer, pp. 202–219.

[8] Stefan Edelkamp & Shahid Jabbar (2006): *Large-Scale Directed Model Checking LTL*. In: *SPIN, LNCS* 3925, Springer, pp. 1–18.

[9] Stefan Edelkamp, Shahid Jabbar & Blai Bonet (2007): *External Memory Value Iteration*. In: *ICAPS*, AAAI, pp. 128–135.

[10] Stefan Edelkamp, Peter Sanders & Pavel Simecek (2008): *Semi-external LTL Model Checking*. In: *CAV, LNCS* 5123, Springer, pp. 530–542.

[11] Sami Evangelista & Lars Michael Kristensen (2013): *Dynamic state space partitioning for external memory state space exploration*. *Sci. Comput. Program.* 78(7), pp. 778–795.

[12] Ernst Moritz Hahn, Arnd Hartmanns & Holger Hermanns (2014): *Reachability and Reward Checking for Stochastic Timed Automata*. *ECEASST* 70.

[13] Moritz Hammer & Michael Weber (2006): *"To Store or Not To Store" Reloaded: Reclaiming Memory on Demand*. In: *FMICS/PDMC, LNCS* 4346, Springer, pp. 51–66.

[14] Arnd Hartmanns & Holger Hermanns (2014): *The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification*. In: *TACAS, LNCS* 8413, Springer, pp. 593–598.

[15] Holger Hermanns, Björn Wachter & Lijun Zhang (2008): *Probabilistic CEGAR*. In: *CAV, LNCS* 5123, Springer, pp. 162–175.

[16] Marta Z. Kwiatkowska, Gethin Norman & David Parker (2011): *PRISM 4.0: Verification of Probabilistic Real-Time Systems*. In: *CAV, LNCS* 6806, Springer, pp. 585–591.

[17] Marta Z. Kwiatkowska, Gethin Norman & David Parker (2012): *The PRISM Benchmark Suite*. In: *QEST*, IEEE Computer Society, pp. 203–204.

[18] Marta Z. Kwiatkowska, Gethin Norman, David Parker & Jeremy Sproston (2006): *Performance analysis of probabilistic timed automata using digital clocks*. *Formal Methods in System Design* 29(1), pp. 33–78.

[19] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala & Jeremy Sproston (2002): *Automatic verification of real-time systems with discrete probability distributions*. *Theor. Comput. Sci.* 282(1), pp. 101–150.

[20] Rashid Mehmood (2004): *Serial Disk-based Analysis of Large Stochastic Models*. In: *Validation of Stochastic Systems, LNCS* 2925, Springer, pp. 230–255.

[21] Ulrich Stern & David L. Dill (1998): *Using Magnetic Disk Instead of Main Memory in the Mur*phi *Verifier*. In: *CAV, LNCS* 1427, Springer, pp. 172–183.

[22] William J. Stewart (1994): *Introduction to the num. solution of Markov Chains*. Princeton University Press.

[23] Mark Timmer, Mariëlle Stoelinga & Jaco van de Pol (2011): *Confluence Reduction for Probabilistic Systems*. In: *TACAS, LNCS* 6605, Springer, pp. 311–325.