# Comparing implicit representations of large CTMCs

Gianfranco Ciardo[1]*, Massimo Forno[2], Paul L. E. Grieco[3], Andrew S. Miner[4]

[1] *Department of Computer Science, College of William and Mary, Williamsburg, VA 23185, USA*
*ciardo@cs.wm.edu, Phone: (757) 221-3478, Fax: (757) 221-1717*
[2] *Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149 Torino, Italy*
*forno@di.unito.it, Phone: +39-348-74-25-716, Fax: +39-011-75-16-03*
[3] *Department of Computer Science, College of William and Mary, Williamsburg, VA 23185, USA*
*pleg@cs.wm.edu, Phone: (757) 221-3469, Fax: (757) 221-1717*
[4] *Department of Computer Science, Iowa State University, Ames, IA 50011-1040, USA*
*asminer@iastate.edu, Phone: (515) 294-2392, Fax: (515) 294-0258*

KEY WORDS: Kronecker Algebra, Decision Diagrams

High-level formalisms of stochastic structures can express very large continuous-time Markov chains (CTMC). State explosion poses interesting challenges in terms of data structures and numerical solution algorithms. *Implicit* representations exploit structural properties of the high-level model [6], to avoid storing each nonzero matrix entry *explicitly*. Implicit structures used to store the CTMC include Kronecker algebra [3], multi-terminal BDDs [10], and matrix diagrams [8], all of which require specialized multiplication algorithms. These methods can be very compact in practice, although in the worst case they require more memory than explicit storage. In this paper, we begin an in-depth comparison of implicit CTMC techniques, we present some initial observations and discuss the currently-known tradeoffs between the various techniques. We identify several questions that are worth further investigation.

## *Implicit approaches*

A high-level model with an underlying CTMC $\{X_t : t \geq 0\}$ encodes the *potential state space* $\widehat{\mathcal{S}}$, which we assume finite; the *potential transition rate matrix* $\widehat{\mathbf{R}} : \widehat{\mathcal{S}} \times \widehat{\mathcal{S}} \to \mathbb{R}$, where $\widehat{\mathbf{R}}[\mathbf{i},\mathbf{j}]$ is the rate at which the CTMC transitions to state $\mathbf{j}$ when it is in state $\mathbf{i}$; and the *initial probability vector* $\widehat{\boldsymbol{\pi}}_0 : \widehat{\mathcal{S}} \to \mathbb{R}$, where $\widehat{\boldsymbol{\pi}}_0[\mathbf{i}] = \Pr\{X_0 = \mathbf{i}\}$ is the probability that the CTMC is in state $\mathbf{i}$ at time 0. We focus on the computation of the steady-state probability vector $\boldsymbol{\pi}$ of an ergodic CTMC, but our discussion also applies to transient and cumulative analysis.

We use the term *potential* to stress that not all states in $\widehat{\mathcal{S}}$ must be reachable. The *(actual) state space* $\mathcal{S}$ depends on $\widehat{\mathbf{R}}$ and $\widehat{\boldsymbol{\pi}}_0$, and can be built using a logic (non-numeric) algorithm. Define the *initial state set* as $\mathcal{S}^{init} = \{\mathbf{i} : \widehat{\boldsymbol{\pi}}_0[\mathbf{i}] > 0\}$ and the *next-state function* as $\mathcal{N}(\mathcal{X}) = \{\mathbf{j} : \exists \mathbf{i} \in \mathcal{X}, \widehat{\mathbf{R}}[\mathbf{i},\mathbf{j}] > 0\}$, for $\mathcal{X} \subseteq \widehat{\mathcal{S}}$. $\mathcal{S}$ is the reflexive and transitive closure of $\mathcal{S}^{init}$ with respect to $\mathcal{N}$: $\mathcal{S} = \mathcal{S}^{init} \cup \mathcal{N}(\mathcal{S}^{init}) \cup \mathcal{N}^2(\mathcal{S}^{init}) \cup \cdots = \mathcal{N}^*(\mathcal{S}^{init})$. We define the *(actual) transition rate matrix* $\mathbf{R}$ as the submatrix $\widehat{\mathbf{R}}[\mathcal{S},\mathcal{S}]$.

**Structured high-level models** Implicit techniques require that the *(global) state* $\mathbf{i}$ be *structured* as a vector of *local states*. This can be achieved by decomposing the model into $K$ submodels, or *levels*, which we number from $K$ down to 1, so that $\mathbf{i} = (\mathbf{i}_K, \ldots, \mathbf{i}_1)$. For $K \geq k \geq 1$, let $\mathcal{S}_k$ be the $k^{\text{th}}$ *local state space*, so that $\widehat{\mathcal{S}} = \mathcal{S}_K \times \cdots \times \mathcal{S}_1$. For simplicity, we assume that each $\mathcal{S}_k$ is known a priori and we map its elements to $\{0, \ldots, n_k - 1\}$. If needed, we could determine them on-the-fly during implicit state space generation [7].

*Correspondence to: Gianfranco Ciardo, College of William and Mary, Department of Computer Science, PO Box 8795, Williamsburg, VA 23187-8795, USA; ciardo@cs.wm.edu, Phone: (757) 221-3478, Fax: (757) 221-1717.

Work supported in part by the National Science Foundation under grants CCR-0219745 and ACI-0203971 and by the National Aeronautics and Space Administration under grant NAG-1-02095.

Most implicit techniques also require a structuring of the transition rate matrix according to a set $\mathcal{E}$ of *events*: $\widehat{\mathbf{R}} = \sum_{\alpha \in \mathcal{E}} \widehat{\mathbf{R}}_\alpha$, where $\widehat{\mathbf{R}}_\alpha[\mathbf{i},\mathbf{j}]$ is the rate at which the CTMC moves from state $\mathbf{i}$ to state $\mathbf{j}$ due to the occurrence, or *firing*, of event $\alpha$. Fig. 1(a) shows our running example, a simple closed queuing system with two queues and three customers.

**Kronecker descriptors, Fig. 1(b)** One of the most widely adopted implicit approaches is based on using Kronecker operators [9] to express $\widehat{\mathbf{R}}$. The idea initially proposed by Plateau [12] partitions the events into $K$ *local* classes $\mathcal{E}_k$ (affecting only submodel $k$, for $K \geq k \geq 1$) plus one *synchronizing* class $\mathcal{E}_S$ (affecting two or more submodels). Then, $\widehat{\mathbf{R}} = \bigoplus_{K \geq k \geq 1} \mathbf{R}_k + \sum_{\alpha \in \mathcal{E}_S} \bigotimes_{K \geq k \geq 1} \mathbf{R}_{\alpha,k}$, where $\mathbf{R}_{\alpha,k} : \mathcal{S}_k \times \mathcal{S}_k \rightarrow \mathbb{R}$ is a (small) matrix describing the effect and rate of event $\alpha$ on submodel $k$, and $\mathbf{R}_k = \sum_{\alpha \in \mathcal{E}_k} \mathbf{R}_{\alpha,k}$. If $\widehat{\mathbf{R}}$ can be expressed in this manner we say that the partition is *Kronecker consistent.* While this expression has computational advantages, we can ignore the distinction between local and synchronizing events, and simply write $\widehat{\mathbf{R}} = \sum_{\alpha \in \mathcal{E}} \bigotimes_{K \geq k \geq 1} \mathbf{R}_{\alpha,k}$, since $\mathbf{R}_{\alpha,k}$ is the identity if the event $\alpha$ is independent of level $k$.

**MTMDDs, Fig. 1(c,e)** A *decision diagram* is an acyclic graph with nodes organized in levels, a single *root* at level $K$, and two *terminal* nodes, 0 and 1. Binary decision diagrams (BDDs) [2] in particular have been employed for *model-checking* discrete-state systems [4]. Multi-way decision diagrams (MDDs) extend BDDs by letting $\mathcal{S}_k$ contain $n_k \geq 2$ local states [11]. We employ them to generate the state space of enormous but finite systems [6], making this an inexpensive step compared to the CTMC solution that must, in our case, follow.

To encode a transition rate matrix, *(real-valued) multi-terminal* decision diagrams are required. In the literature, binary-choice real-valued diagrams, i.e., MTBDDs (also called *algebraic decision diagrams* [1]) are most common, but we allow multi-way choices, i.e., MTMDDs, for generality and a better comparison with the other methods considered. MTMDDs can store either $\widehat{\mathbf{R}}$ or $\mathbf{R}$. In the latter case, strictly speaking, the MTMDD still encodes a $|\widehat{\mathcal{S}}| \times |\widehat{\mathcal{S}}|$ real matrix just like $\widehat{\mathbf{R}}$, but the rows and columns corresponding to unreachable states are set to zero. While, in the case of $\widehat{\mathbf{R}}$, the submatrix $\widehat{\mathbf{R}}[\widehat{\mathcal{S}} \backslash \mathcal{S}, \widehat{\mathcal{S}}]$ can have nonzero entries. The use of $\mathbf{R}$ tends to increase the size of the MTMDD, but it may lead to faster numerical algorithms.

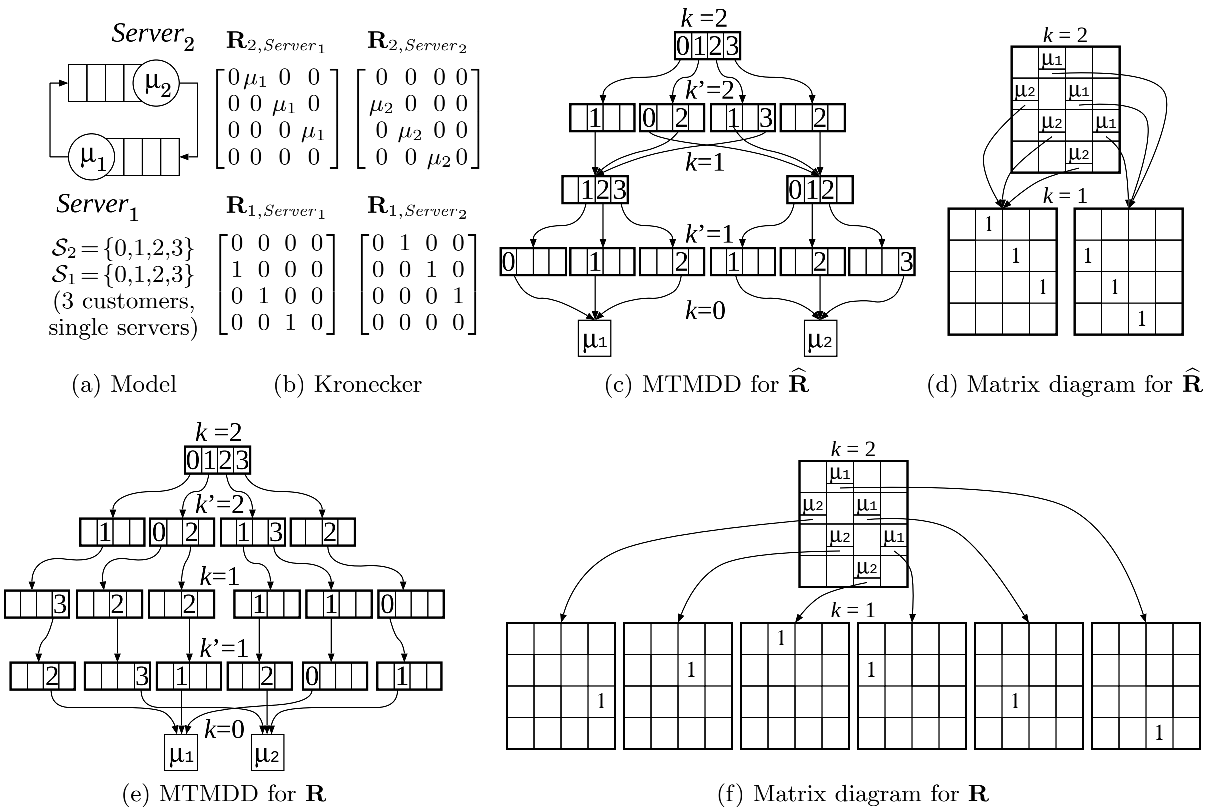
**Matrix diagrams, Fig. 1(d,f)** A *matrix diagram* is an *edge-valued* decision diagram where the arcs from each non-terminal node are organized into a matrix. Unlike MTMDDs, which store the entry values in terminal nodes, a matrix diagram "distributes" them within the graph structure. Each arc not pointing to terminal node 0 has an associated non-zero real value, and the value of an entry of the encoded matrix is determined by multiplying the real values on the corresponding path through the matrix diagram (if the path leads to terminal node 0, the value of the entry is 0).

As for MTMDDs, the matrix diagram for $\widehat{\mathbf{R}}$ can be easily built from the Kronecker descriptor [8] and, given $\mathcal{S}$ encoded as an MDD, it can be modified so that elements in unreachable rows and columns are zero. Again, the resulting representation of $\mathbf{R}$ usually contains more nodes than $\widehat{\mathbf{R}}$.

### *Comparison criteria*

**Potential vs. Actual** Early numerical algorithms using Kronecker descriptors ignored the distinction between $\widehat{\mathbf{R}}$ and $\mathbf{R}$ by using power method or Jacobi iterations on a vector $\widehat{\mathbf{x}}$ of size $|\widehat{\mathcal{S}}|$ initialized to $\widehat{\boldsymbol{\pi}}_0$. However, if $|\widehat{\mathcal{S}}| \gg |\mathcal{S}|$, as is often the case, the memory wasted by $\widehat{\mathbf{x}}$ can be significant, possibly outweighing the memory saved by using Kronecker descriptors. Methods using "actual-sized" vectors $\mathbf{x} \in \mathbb{R}^{|\mathcal{S}|}$ were developed later [3]. They employ knowledge of $\mathcal{S}$ and their time efficiency can range from somewhat worse to much better than the "potential-sized vector" methods, depending on the ratio $|\mathcal{S}|/|\widehat{\mathcal{S}}|$. The use of $\mathbf{x}$ rather than $\widehat{\mathbf{x}}$ enables the use of Gauss-Seidel and SOR iterations using implicit techniques. MTMDDs and matrix diagrams can actually remove or at least set to zero unreachable entries, but this usually increases the size of the representation. We are exploring the space and time implications of using them to store $\widehat{\mathbf{R}}$, "skipping" the unreachable entries during numerical solution, as we must do for Kronecker, or to store $\mathbf{R}$.

**Selecting and ordering submodels** It is well-known that the ordering of the levels can greatly influence the size of a decision diagram. For example, a good rule of thumb is that the "from" and "to" levels in an MTMDD should be interleaved. In addition, our "multi-way" choice introduces further degrees of freedom: we can range from having $K = \lceil \log |\widehat{\mathcal{S}}| \rceil$ binary local state spaces $\mathcal{S}_k$, which is potentially easier to implement but requires more nodes, to a single local state space $\mathcal{S}_1 \equiv \mathcal{S}$, which coincides with explicit approaches. An


Server$_2$
Server$_1$
$\mu_1$
$\mu_2$
$\mathcal{S}_2 = \{0,1,2,3\}$
$\mathcal{S}_1 = \{0,1,2,3\}$
(3 customers, single servers)
$\mathbf{R}_{2,Server_1}$
$\mathbf{R}_{2,Server_2}$
$\mathbf{R}_{1,Server_1}$
$\mathbf{R}_{1,Server_2}$
$k$=2
$k'$=2
$k$=1
$k'$=1
$k$=0
$k$ = 2
$k$ = 1

(a) Model (b) Kronecker (c) MTMDD for $\widehat{\mathbf{R}}$ (d) Matrix diagram for $\widehat{\mathbf{R}}$

(e) MTMDD for $\mathbf{R}$ (f) Matrix diagram for $\mathbf{R}$

**Figure 1:** A queuing network and various implicit representations for the underlying $\mathbf{R}$ or $\widehat{\mathbf{R}}$.

advantage of Kronecker descriptors is that the memory, although not the time, requirements are independent of the submodel ordering.

**Skipping levels** Two canonical versions of decision diagrams exist: *reduced*, where a node is eliminated if it is *redundant*, i.e., all its arcs point to the same node, and *quasi-reduced*, where all arcs span only one level; neither permits duplicate nodes. The definition of redundant is appropriate when encoding *sets*, but not *relations* since, in the latter, *identity transformations*, i.e., $\mathbf{i}'_k = \mathbf{i}_k$, are much more common than *don't care transformations*, i.e., the value of $\mathbf{i}'_k$ is a constant independent of $\mathbf{i}_k$, or it can be anything when $\mathbf{i}_k$ has a given value. Indeed, in our experiments, we observed that many MTMDD nodes, e.g., 85.5% in the actual MTMDD of our **FMS-5** model, have a single non-zero arc leaving them; many of these are needed to represent identity transformations. This drawback is not shared by Kronecker, which fully exploits the case $\mathbf{R}_{\alpha,k} = \mathbf{I}$, nor by matrix diagrams, for which a skipped level means "identity," not "don't care."

**Terminal vs. edge values** With Kronecker and matrix diagrams, an entry value is obtained by multiplying edge values seen along a path. MTMDDs avoid these multiplications by storing entry values as terminal nodes, but this explicit flavor comes at cost: their memory requirements are at least proportional to the number of *unique* entries in $\mathbf{R}$. Indeed, the Kronecker product of $K$ $n \times n$ full matrices, which occupy $O(Kn^2)$ memory, requires $O(n^{2K})$ memory if stored with MTMDDs and all resulting entries are unique. While such pathological cases may not arise in practice, complex state-dependent rate dependencies may severely degrade the effectiveness of MTMDDs.

*Preliminary results and ongoing work*

We have implemented each of the described data structures in our tool SMART [5]. Table I displays memory consumption requirements for each method. We present three models, and vary the number of submodels used in the Kronecker partition by row.

| **Model** | $\lvert\widehat{\mathcal{S}}\rvert$ | $\lvert\mathcal{S}\rvert$ | **MDD** | **Explicit** | **Kron** | **Pot** | **Act** | **Pot** | **Act** |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | **MXD** | **MXD** | **MTMDD** | **MTMDD** |
| **FMS-5** | $2.02 \times 10^9$ | 852,012 | 30,490 | 82,727,748 | 34,484 | 29,496 | 138,244 | 5,681,256 | 5,440,256 |
| **FMS-21** | $5.85 \times 10^{16}$ | 852,012 | 22,038 | 82,727,748 | 6,924 | 4,272 | 255,988 | 130,584 | 1,725,656 |
| **KAN-3** | 11,261,376 | 11,261,376 | 13,868 | 1,433,553,408 | 92,476 | 98,614 | 98,634 | 28,961,792 | 29,072,092 |
| **KAN-4** | 49,787,136 | 11,261,376 | 2,410 | 1,433,553,408 | 12,420 | 13,180 | 15,352 | 435,840 | 461,968 |
| **KAN-16** | $3.32 \times 10^{13}$ | 11,261,376 | 3,972 | 1,433,553,408 | 4,868 | 3,402 | 37,072 | 43,308 | 180,276 |
| **MSER-2** | 83,440 | 20,860 | 17,020 | 1,349,452 | 107,736 | 107,510 | 103,120 | 15,464,048 | 15,491,300 |
| **MSER-4** | 2,622,400 | 20,860 | 64,898 | 1,349,452 | 136,424 | 126,584 | 121,358 | 111,760,872 | 25,844,792 |
| **MSER-6** | 6,713,344 | 20,860 | 65,050 | 1,349,452 | 136,396 | 126,362 | 121,614 | 111,752,656 | 25,844,500 |

**Table I:** Memory requirement of representation for various models (in bytes).

Kronecker and Matrix Diagrams take advantage of edge values, stored in the descriptor matrices or internal nodes, to facilitate the reuse of submatrices; they are also able to avoid storing identity transformations alltogether. For these reasons, they are typically more compact than MTMDD representations. The use of large submodels implied by a partition into few levels exacerbates this tendency, because each node on level $k$ of the MTMDD requires $O(|\mathcal{S}_k|)$ memory. However, by storing terminal values, MTMDDs avoid floating point multiplications along entry paths.

In both decision diagram methods, there can be stark differences in the memory requirements of potential and actual representations. In most cases, the representation of $\mathbf{R}$ requires more memory, since nodes shared between reachable and unreachable paths must be split. This is common, and appears in our graphical example. On the other hand, **MSER** model shows that the representation of $\mathbf{R}$ can be smaller than that of $\widehat{\mathbf{R}}$. This occurs when the reduction in the number of non-zeroes allows many nodes to be eliminated from the graph altogether. All partitions of the **MSER** model contain at least one very large local state space, which may contribute to excess nodes in the potential representation. Due to extensive state-dependent behavior, this local state space cannot be further subdivided into Kronecker consistent submodels.

The table shows that partition choices can have a dramatic effect on memory consumption. There is a clear tradeoff between having few submodels, which is closer to explicit storage, and many, which will require more complex symbolic representations. Increasing the number of submodels may also have an adverse effect on the computation time for multiplication.

In future work, we will explore multiplication algorithms that can be used to find numerical solutions of CTMCs encoded implicitly, and compare the time complexities of these algorithms with explicit methods.

## REFERENCES

1. R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Maciii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10(2/3):171–206, Apr. 1997.
2. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comp.*, 35(8):677–691, Aug. 1986.
3. P. Buchholz, G. Ciardo, S. Donatelli, and P. Kemper. Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. *INFORMS J. Comp.*, 12(3):203–222, 2000.
4. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: $10^{20}$ states and beyond. In *Proc. 5th Annual IEEE Symp. on Logic in Computer Science*, pages 428–439, Philadelphia, PA, 4–7 June 1990. IEEE Comp. Soc. Press.
5. Gianfranco Ciardo, Robert L. Jones, III, Andrew S. Miner, and Radu Siminiceaunu. Logical and stochastic modeling with SMART. In Peter Kemper and William H. Sanders, editors, *Proc. Modelling Techniques and Tools for Computer Performance Evaluation*, Urbana-Champaign, IL, USA, September 2003. Springer-Verlag. To appear.
6. G. Ciardo, G. Luettgen, and R. Siminiceanu. Saturation: An efficient iteration strategy for symbolic state space generation. In T. Margaria and W. Yi, editors, *Proc. TACAS*, LNCS 2031, pages 328–342, Genova, Italy, Apr. 2001. Springer-Verlag.
7. G. Ciardo, R. Marmorstein, and R. Siminiceanu. Saturation unbound. In H. Garavel and J. Hatcliff, editors, *Proc. TACAS*, LNCS 2619, pages 379–393, Warsaw, Poland, Apr. 2003. Springer-Verlag.
8. G. Ciardo and A. S. Miner. A data structure for the efficient Kronecker solution of GSPNs. In P. Buchholz, editor, *Proc. PNPM'99*, pages 22–31, Zaragoza, Spain, Sept. 1999. IEEE Comp. Soc. Press.
9. M. Davio. Kronecker products and shuffle algebra. *IEEE Trans. Comp.*, C-30:116–125, Feb. 1981.
10. H. Hermanns, M. Kwiatkowska, G. Norman, D. Parker, and M. Siegle. On the use of MTBDDs for performability analysis and verification of stochastic systems. *Journal of Logic and Algebraic Programming: Special Issue on Probabilistic Techniques for the Design and Analysis of Systems*, 2003. To appear.
11. T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli. Multi-valued decision diagrams: theory and applications. *Multiple-Valued Logic*, 4(1–2):9–62, 1998.
12. B. Plateau. On the stochastic structure of parallelism and synchronisation models for distributed algorithms. In *Proc. ACM SIGMETRICS*, pages 147–153, Austin, TX, USA, May 1985.