Advances in model representations

Markus Siegle

Lehrstuhl für Informatik 7, University of Erlangen-Nürnberg, Germany
siegle@informatik.uni-erlangen.de

Abstract. We review high-level specification formalisms for Markovian performability models, thereby emphasising the role of structuring concepts as realised par excellence by stochastic process algebras. Symbolic representations based on decision diagrams are presented, and it is shown that they quite ideally support compositional model construction and analysis.

1 Introduction

Stochastic models have a long tradition in the areas of performance and dependability evaluation. Since their specification at the level of the Markov chain is tedious and error-prone, several high-level model specification formalisms have been developed, such as queueing networks, stochastic Petri nets and networks of stochastic automata, which allow humans to describe the intended behaviour at a convenient level of abstraction. Although under Markovian assumptions the analysis of the underlying stochastic process does not pose any conceptual problems, the size of the underlying state space often renders models intractable in practice. Structuring concepts have shown to be of great value in order to alleviate this well-known state space explosion problem.

A process algebra is a mathematically founded specification formalism which provides compositional features, such as parallel composition of components, abstraction from internal actions, and the replacing of components by behaviourally equivalent ones. Therefore, stochastic extensions of process algebras are among the methods of choice for constructing complex, hierarchically structured stochastic models.

Recently, decision diagrams, which were originally developed as memory-efficient representations of Boolean functions in the area of hardware verification, have been extended in order to capture the numerical information which is contained in stochastic models. They have already been successfully used as the underlying data structure in prototype tools for performance analysis and verification of probabilistic systems. In this paper, it is shown that symbolic representations based on decision diagrams are particularly attractive if applied in a compositional context, as provided, for example, by a process algebraic specification formalism. In many cases, decision diagrams allow extremely compact representations of huge state spaces, and it has been demonstrated that all steps of model construction, manipulation and analysis (be it model checking, numerical analysis, or a combination of the two) can be carried out on the decision diagram based representations. Thus, we argue that decision diagrams fit in well with structured modelling formalisms and open new ways towards increasing the range of manageable performance evaluation and verification problems.

This paper does not intend to present new research results, but to survey the history of structured model representations, with special emphasis on process algebras and symbolic encodings. We provide many pointers to further reading, without attempting to be exhaustive.

The paper is organised as follows: In Sec. 2, we survey the evolution from monolithic to modular model specification formalisms. Sec. 3 reviews the concept of stochastic process algebras. Sec. 4 introduces the symbolic representation of Markovian models with the help of multi-terminal binary decision diagrams and describes compositional model construction, manipulation and analysis on the basis of this data structure. The paper concludes with Sec. 5.

2 From unstructured to structured models

2.1 Monolithic model representations

Continuous Time Markov Chains (CTMC) are the basic formalism for specifying performance and dependability models[1]. A CTMC consists of a (finite, for our purpose) set of states and a finite set of transitions between states. The transitions are labelled by positive reals, called the transition rates, which determine transition probabilities and state sojourn times (the latter being exponentially distributed). Time-dependent state probabilities can be derived by solving a system of ordinary differential equations, and steady-state probabilities are calculated by solving a linear system of equations (see, for instance [99]). In order to save memory space, CTMCs are commonly represented as sparse matrices, where essentially only the non-zero entries are stored.

The direct specification of a CTMC at the level of individual states and state-to-state transitions is tedious and error-prone, and therefore only feasible for very small models. This motivated researchers to develop high-level specification formalisms for defining Markovian models at a level of abstraction which is more convenient for the human modeller. The most popular of these formalisms are queueing networks and stochastic Petri nets.

Queueing networks (QN), developed mainly in the 1960ies and 1970ies for modelling time-sharing and polling systems, describe customers moving between stations where they receive service after possibly waiting for a service unit to become available. The aim of analysis is typically the mean or distribution of the number of customers at a station, the customer throughput at a station, or the waiting time. The success of queueing networks stems mainly from the fact that for the class of product form networks [5] very efficient analysis algorithms, such as Buzen's algorithm [24] or mean-value analysis [86], are known, and that software tools for the specification and analysis of QN models were available at

[1] In this paper, we do not consider the line of research on non-Markovian models such as described, for example, in [47].

an early stage [89, 100]. Although QN have been extended in various directions, e.g. in order to model the forking and synchronisation of jobs (fork-join QNs, [3, 68, 80]), the formalism of QNs is not suitable for the modelling of arbitrary systems, but specialised to the application area of shared resource systems.

Stochastic Petri nets (SPN) were developed in the 1980ies for modelling complex synchronisation schemes which cannot easily be expressed by queueing models [79]. The modelling primitives of Petri nets (places, transitions, markings) are very basic and do not carry any application-specific semantics. For that reason, Petri nets are universally applicable and very flexible, which is reflected by the fact that they have been successfully applied to many different areas of application. In the class of generalised SPNs (GSPN) [1, 2], transitions are either timed or immediate. Timed transitions are associated with an exponentially distributed firing time, while immediate transitions fire as soon as they are enabled. During the analysis of a GSPN, the reachability graph is generated and the so-called vanishing markings, which are due to the firing of immediate transitions, are eliminated. The result is a CTMC whose analysis yields (steady-state or transient) state probabilities, i.e. the probabilities of the individual net markings, from which high-level measures can be computed.

Some software tools for performance modelling, e.g. USENUM [90], MARCA [98], MOSEL [8] and DNAmaca [70], implement their own specialised model description languages, which can also be considered as high-level specification formalisms for CTMCs.

With the help of the high-level model specification formalisms considered so far it is possible to specify larger CTMCs than at the state-to-state level, but these formalisms do not support the concepts of modularity, hierarchy or composition of submodels. As a result, the models are monolithic and may be difficult to understand and debug. Moreover, state space generation and numerical analysis of very large monolithic CTMCs is often not feasible in practice due to memory and CPU time limitations, which is referred to as the notorious state space explosion problem.

A large state space may become tractable if it is decomposed into smaller parts [95, 33]. Instead of analysing one large system, the decomposition approach relies on analysing several small subsystems, analysing an aggregated overall system, and afterwards combining the subsystems' solutions accordingly. In general, this approach works well for nearly completely decomposable (NCD) systems whose state space can be partitioned into disjoint subsets of states, such that there is a lot of interaction between states belonging to the same subset, but little interaction between states belonging to different subsets. For the class of reversible Markov chains, the decomposition/aggregation approach yields exact results [32]. We mention that the approach may also be applied iteratively [34, 25]. The major question is, of course, how to best partition a given state space, and in general this information should be derived from a modular high-level model specification. Approximate decomposition-based analysis methods for stochastic process algebra models (see Sec. 3) are discussed in [73], where time scale decomposition is based on the concept of NCD Markov chains [65, 72], and response time approximation relies on a structural decomposition for the special class of decision-free processes [74]. Another such approach, based on the exploitation of the structure of a special class of process algebraic models, is described in [7]. Approximate decomposition-based analysis for nearly-independent GSPN structures is considered in [27, 30].

2.2 Modular model representations

Queueing models, stochastic Petri nets and the tool-specific modelling languages mentioned above do not offer the possibility of composing an overall model from components which can be specified in isolation. Such a composition, however, is a highly desirable feature when modelling complex systems, since it enables human users to focus on manageable parts from which a whole system can be constructed. For instance, modern performance analysis advocates a separation of the load model and the machine model, an idea developed already in [69, 62, 63], and similar ideas are also applied in stochastic rendezvous networks [101] and layered queueing networks [87]. As another, specific example, suppose one wished to model a communication system where two partners communicate over some communication medium. The model should reflect this structure, i.e. it should consist of three interacting submodels, one for each partner, and one for the medium. The user should be able to specify these three submodels more or less independently of each other and then simply specify the way in which they interact.

In the basic GSPN formalism, a model consists of a single net which covers the whole system to be studied. Therefore, GSPN models of complex systems tend to become very large and confused and suffer from the state space explosion problem. Stochastic activity networks [88, 35] constitute an approach to the structuring of GSPNs through the sharing of places between different subnets. In the presence of symmetric submodels, they tackle the state space explosion problem by directly generating a reduced reachability graph in which all mutually symmetric markings are collapsed into one. Symmetries also play a predominant role for the analysis of stochastic well-formed coloured Petri nets [26, 43], where a reduced reachability graph is constructed directly from the net description, without the need to construct the full reachability graph first. Another line of research is concerned with building SPNs in a structured way, basically by synchronising subnets via common transitions, which is an instance of the Kronecker approach described below.

Stochastic automata networks (SAN)[2], developed in the 1980ies and 1990ies [82–85], consist of several stochastic automata, basically CTMCs whose transitions are labelled with event names, which run in parallel and may perform certain synchronising events together. Thus, the SAN formalism is truly structured, since it allows the user to specify an overall model as a collection of interacting submodels. The major attraction of SANs is their memory-efficient

[2] The acronym SAN is also used for stochastic activity networks (see above), but in this paper it stands for stochastic automata networks.

representation of the generator matrix of the Markov chain underlying the overall model. This so-called Kronecker (or tensor) approach has since been adapted to queueing networks [17], stochastic Petri nets [14–16, 21, 28, 39, 40], stochastic process algebras [18] and other structured modelling frameworks [19, 22, 91, 92].

The Kronecker approach realises an implicit, space-efficient representation of the transition rate matrix of a stuctured Markov model. Suppose we have two independent CTMCs $\mathcal{C}_1$ and $\mathcal{C}_2$ which are given by their transition rate matrices R_1 and R_2 (of size d_1 and d_2). Let us consider the combined stochastic process $\mathcal{C}$ whose state space is the Cartesian product of the state spaces of $\mathcal{C}_1$ and $\mathcal{C}_2$. Process $\mathcal{C}$ possesses the transition rate matrix R which is given by the Kronecker sum of R_1 and R_2:

$$R \;=\; R_1 \oplus R_2 \;=\; R_1 \otimes I_{d_2} + I_{d_1} \otimes R_2$$

where $\otimes$ denotes Kronecker product, $\oplus$ denotes Kronecker sum and I_d denotes an identity matrix of size d [37]. If, however, $\mathcal{C}_1$ and $\mathcal{C}_2$ are not independent, but perform certain transitions synchronously, the expression for the overall transition rate matrix changes to

$$R \;=\; R_{1,i} \oplus R_{2,i} + \sum_{a \in S} \lambda_a \cdot R_{1,a} \otimes R_{2,a}$$

where $R_{1,i}$ and $R_{2,i}$ contain those transitions which $\mathcal{C}_1$ and $\mathcal{C}_2$ perform independently of each other, and $R_{1,a}$ and $R_{2,a}$ contain those transitions which are caused by an event a from the set of synchronising events S. Here it is assumed that the resulting rate of the synchronising event a is given by λ_a, i.e. it is a predetermined rate, and matrices $R_{1,a}$ and $R_{2,a}$ are indicator matrices which contain only zeroes and ones. (It is also possible that $R_{1,a}$ and $R_{2,a}$ contain rates, in which case in the above subexpression $\lambda_a \cdot R_{1,a} \otimes R_{2,a}$ has to be replaced by $R_{1,a} \otimes R_{2,a}$. This would mean that the resulting rate of a synchronising event is equal to the product of the rates of the participating processes.) For the general case, where the overall model consists of K submodels, the expression for the overall transition rate matrix is given by

$$R \;=\; \bigoplus_{k=1}^{K} R_{k,i} + \sum_{a \in S} \lambda_a \cdot \bigotimes_{k=1}^{K} R_{k,a}$$

The strength of the Kronecker approach lies in its memory-efficiency (it suffices to store a set of matrices of the size of the submodels) and in the fact that for performing numerical analysis, the potentially very large overall transition rate matrix never needs to be constructed or stored explicitly. The compactness of the representation of the transition rate matrix carries over to the generator matrix and to the iteration matrices for some of the common stationary iterative methods. Thus, iterative numerical schemes which rely on matrix-vector multiplication as their basic operation, can be performed directly on the tensor descriptor of the iteration matrix (Plateau [82] used the power method, and Buchholz [13] describes Kronecker-based power, Jacobi, modified Gauss-Seidel, JOR and modified SOR methods). Efficient algorithms for the multiplication of a vector with a Kronecker descriptor are analysed in [42, 97] and in [20], where, however, the authors state that "... all Kronecker-based algorithms are less computationally efficient than a conventional multiplication where [the matrix] R ist stored in sparse format ... " and "This suggests that, in practice, the real advantage of Kronecker-based methods lies exclusively in their large memory savings".

When working with the Kronecker approach, the set of states reachable from the initial state may be only a small subset of the Cartesian product of the involved submodel state spaces. This is known as the "potential versus actual state space" problem. If the actual state space is not known before numerical analysis starts, a probability vector of the size of the potential state space must be allocated, which can waste a considerable amount of memory space and even make the whole analysis impracticable. For that reason, Kronecker-based reachability techniques have been developed, which allow one to work on the actual state space or a limited superset thereof [20, 29, 67, 78].

3 Stochastic process algebras

In this section, we briefly review the concept of stochastic process algebras (SPA). Since process algebras feature composition operators that allow one to construct complex specifications from smaller ones, we argue that they quite ideally support the specification and analysis of structured models. Next we define a simple SPA language which supports both Markovian and immediate transitions.

Definition 1. Stochastic process algebra language $\mathcal{L}$
Let Act be the set of valid action names and Pro the set of process names. Let action $\tau \in Act$ denote the internal, invisible action. For $P, P_i \in \mathcal{L}$, $a \in Act$, $S \subseteq Act \setminus \{\tau\}$, and $X \in Pro$, the set $\mathcal{L}$ of valid expressions is definded by the following language elements:

stop	inaction		
$a; P$	immediate prefix	$(a, \lambda); P$	Markovian prefix
$P_1 + P_2$	choice	$P_1 \vert[S]\vert P_2$	parallel composition
hide a **in** P	hiding	X	process instantiation

A set of definitions of the form $X := P$ constitutes a process environment. ■

With the help of a structured operational semantics, a transition system whose states correspond to process terms can be derived as the semantic model of a process algebraic specification. For a discussion of the full set of semantic rules for Markovian process algebras similar to our language $\mathcal{L}$ we refer the interested reader to the literature, see e.g. [6, 48, 53, 55, 64]. Here we only mention two selected rules. The first is the rule for synchronisation of two processes via Markovian transitions which can be written as follows:

$$\frac{P \xrightarrow{b,\lambda} P' \qquad Q \xrightarrow{b,\mu} Q'}{P|[S]|Q \xrightarrow{b,\phi(\lambda,\mu)} P'|[S]|Q'} \quad b \in S$$

Note that this rule is parametric in a function ϕ determining the rate of synchronisation, since different synchronisation policies (minimum, maximum, product, ...) are possible. In the process algebra TIPP [60], ϕ is instantiated by multiplication, since strong bisimilarity (see below) is a congruence with respect to parallel composition and abstraction, provided that ϕ is distributive over summation of real values, see [60, 52, 55]. Note that the apparent rate construction of PEPA [64] requires a function $\phi(P, Q, \lambda, \mu)$ instead of $\phi(\lambda, \mu)$.

The second rule is the one for hiding in the case of immediate transitions, which states that an immediate transition labelled by a is turned into an internal immediate transition labelled by τ:

$$\frac{P \overset{a}{\dashrightarrow} P'}{\textbf{hide } a \textbf{ in } P \overset{\tau}{\dashrightarrow} \textbf{hide } a \textbf{ in } P'}$$

As we shall see, internal immediate transitions, as generated by this rule, play a key role during the transformation from a transition system to a CTMC. For our stochastic process algebra language $\mathcal{L}$, the resulting semantic model is an extended stochastic labelled transition system (ESLTS):

Definition 2. Extended Stochastic Labelled Transition System (ESLTS)
Let S be a finite set of states. Let $s_0 \in S$ be the initial state. Let Act be a finite set of action labels. Let $\dashrightarrow$ be defined as follows:

$$\dashrightarrow \;\subseteq\; S \times Act \times S$$

Let $\longrightarrow$ be defined as follows:

$$\longrightarrow \;\subseteq\; S \times Act \times \mathbb{R}^{>0} \times S$$

We call $\mathcal{T} = (S, Act, \dashrightarrow, \longrightarrow, s_0)$ an Extended Stochastic Labelled Transition System. *If $(x, a, y) \in \dashrightarrow$, we say that there is an immediate a-transition from state x to state y and write $x \overset{a}{\dashrightarrow} y$. If $(x, b, \lambda, y) \in \longrightarrow$, we say that there is a Markovian b-transition from state x to state y with rate λ and write $x \xrightarrow{b,\lambda} y$.* ■

Note that in view of the symbolic representation described below, we restricted ourselves to finite-state transition systems. An ESLTS whose set of immediate transitions is empty is called SLTS. An example ESLTS is depicted in Fig. 3 (left). Basically, immediate transitions lead to the existence of vanishing (instable) states. These are states which are left as soon as they are entered, i.e. their sojourn time is zero. Conversely, tangible (stable) states are states whose sojourn time has an exponential distribution, i.e. is strictly positive. For the performability analysis of an SPA model, a CTMC is constructed from the ESLTS and analysed with conventional numerical methods. The CTMC is obtained by hiding of all action labels, elimination of the vanishing states and proper cumulation of all Markovian transitions between a given ordered pair of states.

For a compositional framework, as in the context of stochastic process algebras, we propose to refine the well-known notion of vanishing states in the following way:

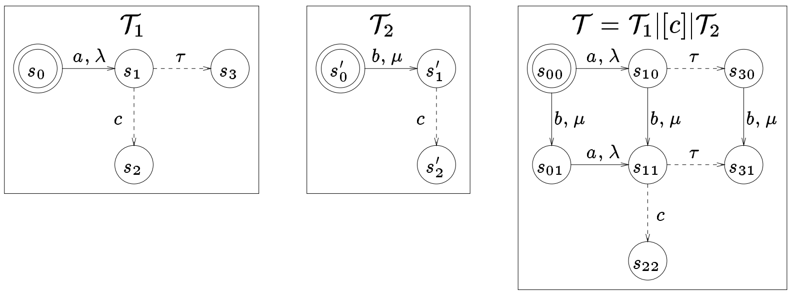

Fig. 1. Role of visible immediate transitions during parallel composition

Definition 3. Compositionally vanishing states
A state s of an ESLTS is called vanishing *if there is at least one internal immediate transition emanating from s (written $s \overset{\tau}{\dashrightarrow} s'$). A state s of an ESLTS is called* compositionally vanishing *if it is vanishing and if there is no visible immediate transition emanating from s (written $s \overset{a}{\not\dashrightarrow} s''$, where $a \neq \tau$).* ■

The idea is that even an immediate transition may be delayed if it is visible, since it may be kept waiting by a synchronisation partner which is not yet ready to participate in the synchronisation. Since synchronisation on internal τ-transitions is not allowed, one can be sure that internal immediate transitions will not be delayed. Compositionally vanishing states can be eliminated either before or after composition of subprocesses, but a vanishing state that may also be left by at least one visible immediate transition must not be eliminated before composition[3]. An example for such a situation is shown in Fig. 1 which shows two ESLTSs, $\mathcal{T}_1$ and $\mathcal{T}_2$, which are composed in parallel, synchronising on action c. The resulting ESLTS, $\mathcal{T}$, is shown on the right hand side of the figure. State s_1 in ESLTS $\mathcal{T}_1$, which is vanishing but not compositionally vanishing, must not be eliminated before parallel composition takes place, since its elimination would disable any c-transition in the combined transition system $\mathcal{T}$. In the resulting ESLTS, state s_{10} is a compositionally vanishing state which can be eliminated, whereas state s_{11} is not. However, if action c is hidden in ESLTS $\mathcal{T}$ (since further synchronisation on c is not required), state s_{11} becomes compositionally vanishing and can be also eliminated (its elimination, however, requires a proper treatment of non-determinism as explained below). Note also that there may be one or several Markovian transitions emanating from a vanishing state, but they are never taken. As an example, in Fig. 1 the transition $s_{10} \xrightarrow{b,\mu} s_{11}$ will never be taken, since the competing internal immediate transition $s_{10} \overset{\tau}{\dashrightarrow} s_{30}$ will always take place first. Therefore transition $s_{10} \xrightarrow{b,\mu} s_{11}$ can safely be deleted without changing the behaviour of the ESLTS.

[3] To complete the picture: A state s is called tangible if there is no immediate transition (i.e. neither visible nor internal) emanating from s. In the remaining case (where there is at least one visible immediate transition, but no internal immediate transition emanating from s) the state is called inconclusive.

The basic strategy of elimination of compositionally vanishing states is to redirect transitions leading to such a state to its successor states. In the case where a compositionally vanishing state has more than one outgoing internal immediate transitions, it is not specified which of them will be taken. This is an instance of non-determinism. In order to resolve such non-determinism, one may assign probabilities or weights to internal immediate transitions. Transitions leading to the compositionally vanishing state can then be redirected to its successor states, taking into account these probabilities.

The concept of bisimilarity is of great importance for SPAs, since it establishes the equivalence between processes, and since it is the basis for state space reduction. Unfortunately, it is beyond the scope of the present paper to discuss bisimulation relations in detail, so we refer to the literature, e.g. [52, 57, 60, 64].

4 Symbolic representations

In this section we present space-efficient symbolic representations of transition systems with the help of binary decision diagrams (BDD). We review multi-terminal BDDs (MTBDD), also called algebraic decision diagrams, since they are capable of representing real-valued functions [4, 31, 46].

4.1 Multi-terminal BDDs

Let $\mathbb{B} = \{0, 1\}$ denote the set of Booleans[4]. An MTBDD is a graph-based representation of a function $f : \mathbb{B}^n \mapsto \mathbb{R}$.

Definition 4. Multi-Terminal Binary Decision Diagram (MTBDD)
Let $Vars = \{\mathsf{v}_1, \ldots, \mathsf{v}_n\}$ be a set of Boolean variables with a fixed total ordering $\prec \subset Vars \times Vars$. An (ordered) Multi-Terminal Binary Decision Diagram *over $\langle Vars, \prec\rangle$ is a rooted directed acyclic graph $\mathsf{M} = (Vert, \mathsf{var}, \mathsf{else}, \mathsf{then}, \mathsf{value})$ defined by*
- *a finite nonempty set of vertices $Vert = T \cup NT$, where T (NT) is the set of terminal (non-terminal) vertices,*
- *a function $\mathsf{var} : NT \mapsto Vars$,*
- *two edge-defining functions $\mathsf{else} : NT \mapsto Vert$ and $\mathsf{then} : NT \mapsto Vert$,*
- *a function $\mathsf{value} : T \mapsto \mathbb{R}$,*

with the following constraints:
$\forall x \in NT : \mathsf{else}(x) \in T \vee \mathsf{var}(\mathsf{else}(x)) \succ \mathsf{var}(x)$
$\forall x \in NT : \mathsf{then}(x) \in T \vee \mathsf{var}(\mathsf{then}(x)) \succ \mathsf{var}(x)$ ■

Note that, according to Def. 4, a binary decision tree is an MTBDD. However, we are mainly interested in reduced MTBDDs, defined as follows:

Definition 5. Reducedness of an MTBDD
An MTBDD M is called reduced *if and only if the following conditions hold:*

[4] We use the real numbers 0 and 1 to represent Boolean values, since in the context of MTBDDs Boolean variables will be involved in arithmetic calculations.

1. $\forall x \in NT\colon \ \mathsf{else}(x) \neq \mathsf{then}(x)$
2. $\forall x, y \in NT\colon \ x \neq y \Rightarrow (\mathsf{var}(x) \neq \mathsf{var}(y) \vee \mathsf{else}(x) \neq \mathsf{else}(y) \vee \mathsf{then}(x) \neq \mathsf{then}(y))$
3. $\forall x, y \in T\colon \ x \neq y \Rightarrow \mathsf{value}(x) \neq \mathsf{value}(y)$ ■

The first condition states that there are no "don't care" vertices, i.e. vertices with identical then- and else-successors. The second condition states that there are no two isomorphic non-terminal vertices, and the third condition states that there are no two isomorphic terminal vertices. Bryant [12] proposed a recursive procedure to reduce BDDs[5] that can be applied to MTBDDs as well, and from now on, unless otherwise stated, we assume that MTBDDs are reduced. Fig. 2 (right) shows a reduced MTBDD. In the graphical representation, the edge from a vertex x to $\mathsf{then}(x)$ is drawn solid, and the edge from x to $\mathsf{else}(x)$ is drawn dashed. All vertices that are drawn on one level are labelled with the same Boolean variable, as indicated at the left of the decision diagram. In order to keep the figure clear, all edges leading to the zero-valued terminal vertex are not drawn, i.e. every non-terminal vertex with only one outgoing edge drawn has its other outgoing edge leading to the zero-valued terminal vertex.

Each MTBDD vertex unambiguously defines a real-valued function, based on the so-called Shannon expansion which states that

$$f(\mathsf{v}_1, \ldots, \mathsf{v}_n) = (1 - \mathsf{v}_1) \cdot f(0, \mathsf{v}_2, \ldots, \mathsf{v}_n) + \mathsf{v}_1 \cdot f(1, \mathsf{v}_2, \ldots, \mathsf{v}_n)$$

The terms $f(0, \mathsf{v}_2, \ldots, \mathsf{v}_n)$ and $f(1, \mathsf{v}_2, \ldots, \mathsf{v}_n)$ are called the cofactors of the function f with respect to the Boolean variable v_1.

Definition 6. Function f_x represented by an MTBDD vertex
The real-valued function f_x represented by an MTBDD vertex $x \in Vert$ is recursively defined as follows:

- *if $x \in T$ then $f_x = \mathsf{value}(x)$,*
- *else (if $x \in NT$) $f_x = (1 - \mathsf{var}(x)) \cdot f_{\mathsf{else}(x)} + \mathsf{var}(x) \cdot f_{\mathsf{then}(x)}$* ■

Most times one is interested in the case where x corresponds to the MTBDD root. In that case we write f_{M} instead of f_x, where x is the root vertex of MTBDD M. The two subgraphs of MTBDD M corresponding to the cofactors of f_{M} are denoted $\mathsf{M}^{\mathsf{then}}$ and $\mathsf{M}^{\mathsf{else}}$, where $\mathsf{M}^{\mathsf{then}}$ represents $f_{\mathsf{M}}(1, \mathsf{v}_2, \ldots, \mathsf{v}_n)$ and $\mathsf{M}^{\mathsf{else}}$ represents $f_{\mathsf{M}}(0, \mathsf{v}_2, \ldots, \mathsf{v}_n)$. For a fixed ordering of Boolean variables, reduced MTBDDs form a *canonical* representation of real-valued functions, i.e. if M, M' are two reduced MTBDDs over the same ordered set of Boolean variables $Vars$ such that $f_{\mathsf{M}} = f_{\mathsf{M}'}$, then M and M' are isomorphic.

It should be noted that, given a Boolean function, the size of the resulting MTBDD is highly dependent on the chosen variable ordering. As a prominent example, consider the function $f_{\mathsf{Id}} = \prod_{k=1}^{n} (\mathsf{s}_k \equiv \mathsf{t}_k)$, which can be interpreted as an identity matrix of size 2^n. Under the interleaved variable ordering $\mathsf{s}_1 \prec \mathsf{t}_1 \prec \ldots \prec \mathsf{s}_n \prec \mathsf{t}_n$ the number of vertices needed to represent this function is $3n + 2$, i.e. logarithmic in the size of the matrix. In contrast, using the straight-forward

[5] A BDD is an MTBDD where $\forall x \in T\colon \mathsf{value}(x) \in \{0, 1\}$.

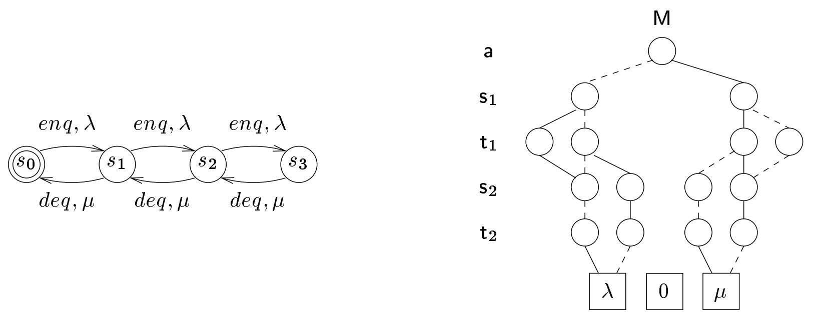

Fig. 2. SLTS and corresponding MTBDD

ordering $\mathsf{s}_1 \prec \ldots \prec \mathsf{s}_n \prec \mathsf{t}_1 \prec \ldots \prec \mathsf{t}_n$, the number of vertices is $3 \cdot 2^n - 1$. Since identity matrices play an important role during the parallel composition of transition systems (see below), their compact representation is an essential feature of MTBDDs.

A comprehensive set of logical and arithmetic operations can be realised efficiently on MTBDDs, such that it is possible to perform calculations on the functions which are represented by the decision diagrams. The operation APPLY, for instance, combines two MTBDDs by a binary arithmetic operator, RESTRICT fixes the value of one or more variables of the MTBDD, and ABSTRACT combines restricted copies of an MTBDD by an associative binary operator. In general, algorithms for MTBDD construction and manipulation are variants of their corresponding BDD algorithms [12]. They all follow a recursive descent scheme according to the above Shannon expansion, and their efficiency is achieved through the clever use of a hash-based vertex table and a cache where intermediate results are stored for later re-use [11]. MTBDDs are very well suited for the compact representation of block-structured matrices, and symbolic algorithms for matrix multiplication and other linear algebra operations exist [4, 46, 51]. However, existing implementations of MTBDD-based matrix multiplication and vector matrix multiplication are considerably slower than their sparse counterparts.

4.2 Symbolic representation of transition systems

Fig. 2 shows an SLTS and its symbolic representation by an MTBDD. Since the set *Act* of this SLTS only contains two elements, a single Boolean variable suffices to encode the action label (the case $\mathsf{a} = 0$ encodes action *enq*, and $\mathsf{a} = 1$ encodes action *deq*). Since the SLTS has four states, two bits are required to encode the state identity. We use Boolean variables $\mathsf{s}_1, \mathsf{s}_2$ to encoded a transition's source state, and $\mathsf{t}_1, \mathsf{t}_2$ to encode its target state. The transition $s_2 \xrightarrow{enq,\lambda} s_3$, for example, is encoded by the combination $(\mathsf{a}, \mathsf{s}_1, \mathsf{t}_1, \mathsf{s}_2, \mathsf{t}_2) = (0, 1, 1, 0, 1)$. Note the interleaving of the variables for source and target state.

If MTBDD M represents SLTS $\mathcal{T}$ we write $\mathsf{M} \rhd \mathcal{T}$. For the symbolic representation of an ESTLS $\mathcal{T}$, one employs two separate decision diagrams, i.e. an MTBDD M^I which encodes all immediate transitions, and an MTBDD M^M which encodes all Markovian transitions, as shown in the example of Fig. 3. We

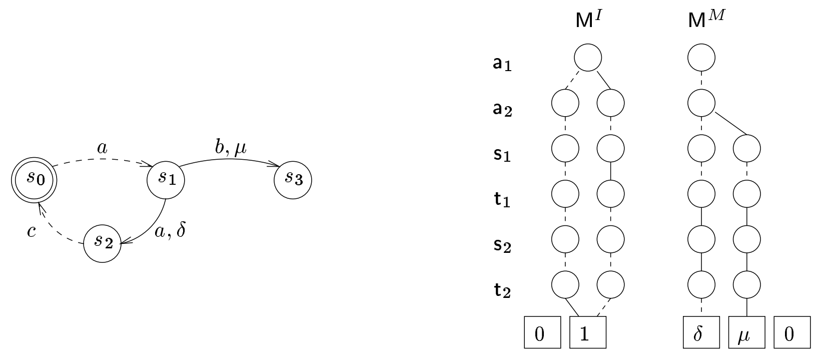

Fig. 3. Encoding of an example ESLTS

then write $(\mathsf{M}^I, \mathsf{M}^M) \triangleright \mathcal{T}$. Basically, M^I is a BDD, since it does not encode any rate values. However, in certain situations one may wish to associate immediate transitions with numerical values, for instance to associate them with weights or probabilities, a feature which may be needed when resolving non-determinism between several concurrently enabled internal immediate transitions. In this case, M^I is a proper MTBDD with possibly more than two terminal vertices. Our tool IM-CAT[6] realises this scheme. A second alternative for the symbolic representation of ESLTSs, where both Markovian and immediate transitions are represented by a single MTBDD, is described in [58].

4.3 Symbolic manipulation and analysis of transition systems

In this section, we discuss the construction, manipulation and analysis of transition systems represented by MTBDDs. Given a transition system, its symbolic representation can be easily constructed as the sum of the MTBDDs encoding the individual transitions. However, we recommend this procedure only for small transition systems, since the encoding of monolithical transition systems does usually not yield compact representations. Large transition systems should be generated in a compositional fashion from components, following the parallel composition operator of process algebras.

Parallel composition: We now describe MTBDD-based parallel composition, but for simplicity we restrict ourselves to the case of SLTS. The general ESLTS case works in a similar way. Consider the parallel composition of two SLTSs $\mathcal{T}_1$ and $\mathcal{T}_2$ where actions from the set $S \subseteq \mathit{Act} \setminus \{\tau\}$ shall take place in a synchronised way. Using process algebraic notation, we can express this as $\mathcal{T} = \mathcal{T}_1|[S]|\mathcal{T}_2$, where $\mathcal{T}$ is the resulting SLTS. Assume that the MTBDDs which correspond to SLTSs $\mathcal{T}_1$ and $\mathcal{T}_2$ have already been generated and are denoted M_1 and M_2, i.e. $\mathsf{M}_i \triangleright \mathcal{T}_i$ for $i \in \{1, 2\}$. The set of synchronising actions S can also be encoded in the standard way as a BDD, say S (action labels are encoded by the same Boolean variables in $\mathsf{M}_1, \mathsf{M}_2$ and S). The MTBDD M representing

[6] IM-CAT is a tool for the compositional construction and analysis of ESLTSs [44, 45] which uses the CUDD library [96].

$\mathcal{T}$ (i.e. $\mathsf{M} \rhd \mathcal{T}$) is constructed as follows:

$$\begin{aligned} \mathsf{M} = \quad & (\mathsf{M}_1 \cdot \mathsf{S}) \cdot (\mathsf{M}_2 \cdot \mathsf{S}) \\ & + \ \mathsf{M}_1 \cdot (1 - \mathsf{S}) \cdot \mathsf{Id}_2 \ + \ \mathsf{M}_2 \cdot (1 - \mathsf{S}) \cdot \mathsf{Id}_1 \end{aligned}$$

The term on the first line is for the synchronising actions in which both $\mathcal{T}_1$ and $\mathcal{T}_2$ participate. The multiplication $\mathsf{M}_1{\cdot}\mathsf{S}$ selects that part of SLTS $\mathcal{T}_1$ which corresponds to actions from the set S, and similarly for $\mathsf{M}_2{\cdot}\mathsf{S}$. By then taking the product of these two terms one obtains the encoding of those transitions where both partners simultaneously make a move. The two symmetric terms on the second line are for those actions which $\mathcal{T}_1$ ($\mathcal{T}_2$) performs independently of $\mathcal{T}_2$ ($\mathcal{T}_1$) — these actions are all from the complement of S, encoded by $(1-\mathsf{S})$ — and the multiplication with Id_2 (Id_1) ensures that $\mathcal{T}_2$ ($\mathcal{T}_1$) remains stable, i.e. does not change its state. Note that for the synchronising transitions, calculated by the first line in the above expression, the resulting rate is given by the product $\lambda \cdot \mu$. Should one wish to employ a different function $\phi(\lambda, \mu)$, for instance the maximum function, one would simply have to replace the first line of the above expression by $\textsc{Max}(\mathsf{M}_1{\cdot}\mathsf{S}, \mathsf{M}_2{\cdot}\mathsf{S})$, where $\textsc{Max}$ is the maximum function on MTBDDs which can be realised with the help of a particular instance of the standard $\textsc{Apply}$ algorithm.

Enders et al. [41], who considered the parallel composition of BDDs generated from CCS terms, showed that the size of the symbolic representation is proportional to the sum of the sizes of its components, provided that the components are loosely coupled and provided that the interleaved variable ordering is used. We now state a similar result for the parallel composition of ESLTSs. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two ESLTSs represented by MTBDDs, i.e. $(\mathsf{M}_i^I, \mathsf{M}_i^M) \rhd \mathcal{T}_i$ ($i = 1, 2$), using the interleaved variable ordering. Let $(\mathsf{M}^I, \mathsf{M}^M) \rhd \mathcal{T}_1|[S]|\mathcal{T}_2$ where M^M is constructed from M_1^M, S and M_2^M as above, and M^I is constructed from M_1^I, S and M_2^I in a similar way. Then the number of vertices of M^M is *linear* in the number of vertices of M_1^M and M_2^M, and the number of vertices of M^I is *linear* in the number of vertices of M_1^I and M_2^I. For a proof of this important property see [94].

The fact that parallel composition of components can be realised symbolically in such a way that the size of the data structure grows only linearly compares favourably to the exponential growth resulting from the usual interleaving of causally independent transitions (as generated, for instance, by the operational semantics of process algebras). This feature may be exploited in order to obtain extremely compact representations of huge transition systems. In fact, one can safely state that symbolic representations are only beneficial if they are used in a compositional context.

Reachability analysis: The MTBDD M resulting from the parallel composition of two partners M_1 and M_2 encodes all transitions which are possible in the product space of the two partner processes (called the potential state space). Given a pair of initial states for SLTSs $\mathcal{T}_1$ and $\mathcal{T}_2$, however, only part of this product space (the actual state space) may be reachable due to synchronisation constraints. Therefore, M potentially includes transitions emanating from unreachable states. In this situation, reachability analysis is an important tool for reducing the size of the underlying SLTS. Reachability analysis can be performed efficiently on the symbolic representation of the resulting transition system $\mathcal{T}$ (as described for the purely functional LTS case in [9]), both for SLTSs and ESLTSs represented by MTBDDs.

Hiding: Hiding of action labels, i.e. replacing a visible action a by the internal action τ, can be performed on the MTBDD-based representation of an ESLTS with the help of the operations Restrict and Apply, by selecting and modifying that part of the MTBDD which encodes a-transitions, and by afterwards recombining it with the remaining part of the MTBDD. While the hiding of Markovian transitions does not enable reductions of the transition system, the hiding of immediate transitions may lead to compositionally vanishing states which can be eliminated. In [94], we describe a symbolic algorithm for the elimination of such states that offers a flexible mechanism for resolving nondeterminism between several internal immediate transitions, as realised in our tool Im-Cat.

Bisimulation: Symbolic characterisation of strong and weak bisimulation and symbolic algorithms for computing a factorisation of the state space have been described in the literature [9,23,41]. In [61], symbolic algorithms for computing strong and weak Markovian bisimulation on ESLTSs are described in detail, using decision node BDDs (DNBDD) [93], another extension of BDDs for the representation of real-valued functions, as the underlying data structure. These algorithms follow the well-known strategy of iterative refinement and can readily be implemented with the help of MTBDDs.

Numerical analysis: Numerical analysis can be carried out directly on the symbolic representation of the Markov chain [49–51]. Direct methods for calculating steady-state probabilities are generally unsuitable for symbolic implementation, since each step modifies the structure of the coefficient matrix and thus the MTBDD structure, which causes considerable overhead to keep the representation canonical and destroys its compactness [4]. For the analysis of large Markov chains based on their symbolic representation, iterative methods are more suitable. Apart from a general matrix powering algorithm [59] that can be instantiated as the power method or the method of Jacobi[7], the projection methods BiCGStab [45,94] and CGS [36] have been realised on the basis of MTBDDs. Unfortunately, the symbolic implementations of these algorithms are all substantially slower than their sparse-matrix counterparts, a fact which is due to the relatively poor performance of symbolic vector-matrix multiplication, as has been observed also in [4,36,38,44,71].

4.4 Compactness of the symbolic representation

As an example (taken from [66] and also used in [56]), we consider a cyclic server polling system consisting of d stations and a server. The MTBDD rep-

[7] In principle, the method of Gauss-Seidel can also be realised by vector-matrix multiplication, but this requires the inversion of a triangular matrix which usually leads to inefficient encodings.

d	reach. states	transitions	MTBDD size compositional before reachability	MTBDD size compositional after reachability	MTBDD size monolithic
3	36	84	169	203	351
5	240	800	387	563	1,888
7	1,344	5,824	624	1,087	9,056
10	15,360	89,600	1,163	2,459	69,580
15	737,280	6.144e+6	2,191	6,317	–
20	3.14573e+07	3.40787e+08	3,704	13,135	–

Table 1. Statistics for the polling system

resentation of the overall polling model ($\mathcal{T}_{poll}$) is constructed with the help of MTBDD-based parallel composition from $d+1$ elementary transition systems[8], one for the server ($\mathcal{T}_{serv}$) and one for each station ($\mathcal{T}_{stat_i}$), according to $\mathcal{T}_{poll} := \mathcal{T}_{serv}|[S]|(\mathcal{T}_{stat_1}|[\emptyset]| \dots |[\emptyset]|\mathcal{T}_{stat_d})$. The order in which the component MTBDDs are generated turned out to be of great importance for the resulting MTBDD size, since it determines the ordering of the MTBDD variables. Unfortunately, it is not obvious a priori which component ordering yields small MTBDDs.

In Tab. 1, the sizes of the resulting MTBDDs are given for different values of d. The first column of the table contains the number of stations d, the 2nd (3rd) column contains the number of reachable states (the number of transitions), and the remaining columns give the number of vertices of the corresponding MTBDDs[9]. Tab. 1 shows that even for an extremely large state space, the MTBDD representation can be very compact, if it is constructed in a compositional fashion. The last column of Tab. 1 shows the number of MTBDD vertices which one would obtain if one took the monolithic transition system of the overall model as generated by TIPPTOOL (which does not contain unreachable states), and directly encoded it as an MTBDD. Clearly, this method cannot be recommended: Apart from the fact that the transition system of the overall model may not be available due to its excessive size and generation time (as indicated by the "–" entries), the growth of the MTBDD sizes is prohibitive. As expected, the figures in column 4 grow linearly, whereas the ones in column 6 grow exponentially.

The MTBDDs generated compositionally represent all transitions which are possible within the potential state space. As can be observed from the 5th column of Tab. 1, determining the set of reachable states and "deleting" the transitions which originate in unreachable states considerably increases the size of the MTBDDs. Therefore, in general, although MTBDD-based reachability analysis is very fast, it is recommended to work with MTBDDs which represent the potential rather than the actual state space, and store the reachability predicate in a separate BDD. It may seem quite surprising that restriction to the reachable part of the transition system increases the size of the MTBDD. However, similar phenomena can be observed when performing symbolic elimination of vanishing states or symbolic bisimulation: The size of the symbolic representation grows

[8] The elementary transition systems were generated by the stochastic process algebra tool TIPPTOOL [54] and then encoded as individual MTBDDs.

[9] Since the considered version of the polling model does not contain immediate transitions, a single MTBDD (representing Markovian transitions) is sufficient.

although the underlying transition system is reduced, i.e. fewer states and transitions are represented. Our explanation for such counter-intuitive behaviour is that the regularity of the model gets lost through the reduction, which destroys the regularity of the MTBBD and thus its compactness (see [59]).

We now mention some figures concerning the MTBDD-based numerical analysis of the polling system: For the case $d = 7$, and working on the potential state space, the MTBDD representing the power iteration matrix, as generated by IM-CAT, has 806 vertices and takes 0.8 seconds to construct[10]. One iteration of the power scheme takes at the average 0.122 seconds, but it takes a ridiculous 8070 power iterations to converge. The MTBDD representing the Jacobi iteration matrix for the same system is larger, it has 1639 vertices and takes 18.94 second to construct, but one Jacobi iteration takes only 0.101 seconds and convergence is reached after 240 iterations. Unfortunately, these speeds are unacceptably slow, if compared to state-of-the-art sparse matrix implementations such as TIPP-TOOL's solver (based on SparseLib1.3 by K. Kundert, UC Berkeley), where one Gauss-Seidel iteration takes only 0.0013 seconds.

5 Discussion and conclusion

In this paper, we have reviewed the evolution from monolithic to modular model representations. In particular, we have described space-efficient symbolic representations of compositional Markov models stemming from process algebraic specifications, thereby emphasising the role of symbolic parallel composition.

We briefly mention two other data structures related to decision diagrams and developed for the analysis of Markovian systems: Matrix diagrams [28, 76–78], an extension of multi-valued decision diagrams, enable the compact representation of structured GSPN models, and probabilistic decision graphs [10] enable a consise representation of probability vectors and probabilistic transition system.

As we have seen, the main bottleneck of the symbolic modelling procedure is numerical analysis. Therefore, speeding up MTBDD-based vector-matrix multiplication remains a major area of research. A promising approach to this problem that combines the advantages of sparse and MTBDD-based representations will be described in [81]. A totally different direction is taken in [75], where special-purpose hardware for the support of basic MTBDD operations has been developed. Parallelisation and distribution of MTBDD manipulation algorithms are also candidates for improving the speed of MTBDD-based numerical analysis.

In summary, we argue that modular model specifications and symbolic representations are a perfect match, and that this combination should play a leading role in future performability analysis and verification projects.

[10] The experimental results were obtained on a SUN 5/10 workstation, equipped with 1GB of main memory and running at 300 MHz.

Acknowledgements

The author wishes to thank Holger Hermanns, Marta Kwiatkowska, Gethin Norman and David Parker for many fruitful discussions. Edgar Frank deserves credit for his implementation work on our tool IM-CAT. Thanks also to Lennard Kerber and Matthias Kuntz for critical comments on an earlier version of the paper.

References

1. M. Ajmone Marsan, G. Balbo, and G. Conte. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.
2. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with generalized stochastic Petri nets.* Wiley, 1995.
3. F. Baccelli, W.A. Massey, and D. Towsley. Acyclic Fork–Join Queuing Networks. *Journal of the ACM*, 36(3):615–642, 1989.
4. R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic Decision Diagrams and their Applications. *Formal Methods in System Design*, 10(2/3):171–206, April/May 1997.
5. F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open, Closed and Mixed Networks of Queues with Different Classes of Customers. *Journal of the ACM*, 22(2):248–260, 1975.
6. M. Bernardo and R. Gorrieri. A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time. *Theoretical Computer Science*, 202:1–54, 1998.
7. H. Bohnenkamp and B. Haverkort. Semi-Numerical Solution of Stochastic Process Algebra Models. In J.-P. Katoen, editor, *ARTS'99*, pages 228–243. Springer, LNCS 1601, 1999.
8. G. Bolch and S. Greiner. Modeling and Performance Evaluation of Production Lines Using the Modeling Language MOSEL. In *Proc. of the 2nd IEEE/ECLA/IFIP Int. Conf. on Architectures and Design Methods for Balanced Automation Systems*, pages 163–174, June 1996.
9. A. Bouali and R. de Simone. Symbolic Bisimulation Minimisation. In *Computer Aided Verification*, pages 96–108, 1992. LNCS 663.
10. M. Bozga and O. Maler. On the Representation of Probabilities over Structured Domains. In N. Halbwachs and D. Peled, editors, *Int. Conf. on Computer-Aided Verification (CAV'99)*, pages 261–273. Springer, LNCS 1633, July 1999.
11. K.S. Brace, R.L. Rudell, and R.E. Bryant. Efficient Implementation of a BDD Package. In *27th ACM/IEEE Design Automation Conf.*, pages 40–45, 1990.
12. R.E. Bryant. Graph-based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.
13. P. Buchholz. *Die strukturierte Analyse Markovscher Modelle.* PhD thesis, Universität Dortmund, 1991 (in German).
14. P. Buchholz. A Hierarchical View of GCSPNs and Its Impact on Qualitative and Quantitative Analysis. *Journal of Parallel and Distributed Computing*, 15(3):207–224, July 1992.
15. P. Buchholz. Aggregation and Reduction Techniques for Hierarchical GCSPNs. In *Proc. of PNPM '93*, pages 216–225, Tolouse, October 1993.
16. P. Buchholz. Hierarchies in Colored GSPNs. In M. Ajmone Marsan, editor, *14th Int. Conf. on Application and Theory of Petri Nets*, pages 106–125. Springer, LNCS 691, 1993.
17. P. Buchholz. A class of hierarchical queueing networks and their analysis. *Queueing Systems*, 15:59–80, 1994.
18. P. Buchholz. Markovian Process Algebra: Composition and Equivalence. In U. Herzog and M. Rettelbach, editors, *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, pages 11–30. Arbeitsberichte des IMMD No. 27/4, Universität Erlangen-Nürnberg, July 1994.

19. P. Buchholz. A Framework for the Hierarchical Analysis of Discrete Event Dynamic Systems. Habilitation thesis, Universität Dortmund, 1996.
20. P. Buchholz, G. Ciardo, S. Donatelli, and P. Kemper. Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. *INFORMS Journal of Computing*, 12(3):203–222, Summer 2000.
21. P. Buchholz and P. Kemper. On Generating a Hierarchy for GSPN Analysis. *Performance Evaluation Review (ACM Sigmetrics)*, 26(2):5–14, August 1998.
22. P. Buchholz and P. Kemper. A Toolbox for the Analysis of Discrete Event Dynamic Systems. In N. Halbwachs and D. Peled, editors, *Computer Aided Verification*, pages 483–486. Springer, LNCS 1633, 1999.
23. J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computation*, (98):142–170, 1992.
24. J.P. Buzen. Computational Algorithms for Closed Queueing Networks with Exponential Servers. *Communications of the ACM*, 16:527–531, 1973.
25. W.L. Cao and W.J. Stewart. Iterative Aggregation/Disaggregation Techniques for Nearly Uncoupled Markov Chains. *Journal of the ACM*, 32(3):702–719, July 1985.
26. G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. On Well-Formed Coloured Nets and their Symbolic Reachability Graph. In *Proc. of the 11th Int. Conf. on Application and Theory of Petri Nets*, pages 387–410, Paris, June 1990. Reprinted in High-level Petri Nets, K. Jensen, G. Rozenberg, eds., Springer 1991.
27. G. Ciardo. *Analysis of Large Stochastic Petri Net Models.* PhD thesis, Duke University, Durham, NC, USA, 1989.
28. G. Ciardo and A.S. Miner. A data structure for the efficient Kronecker solution of GSPNs. In P. Buchholz and M. Silva, editors, *PNPM'99*, pages 22–31. IEEE computer society, 1999.
29. G. Ciardo and M. Tilgner. Parametric State Space Structuring. Technical Report ICASE Report No. 97-67, ICASE, 1997.
30. G. Ciardo and K.S. Trivedi. A decomposition approach for stochastic reward net models. *Performance Evaluation*, 18(1):37–59, July 1993.
31. E.M. Clarke, K.L. McMillan, X. Zhao, M. Fujita, and J. Yang. Spectral Transforms for Large Boolean Functions with Applications to Technology Mapping. In *30th Design Automation Conf.*, pages 54–60. ACM/IEEE, 1993.
32. A.E. Conway and N.D. Georganas. *Queueing Networks – Exact Computational Algorithms.* MIT Press, 1989.
33. P.J. Courtois. *Decomposability, queueing and computer system applications.* ACM monograph series, 1977.
34. P.J. Courtois. On Time and Space Decomposition of Complex Structures. *Communications of the ACM*, 28(6):590–603, June 1985.
35. J. Couvillion, R. Freire, R. Johnson, W.D. Obal II, M.A. Qureshi, M. Rai, W.H. Sanders, and J. Tvedt. Performability modeling with UltraSAN. *IEEE Software*, 8(5):69–80, September 1991.
36. I. Davies. Symbolic techniques for the performance analysis of generalised stochastic Petri nets. Master's thesis, University of Cape Town, Department of Computer Science, January 2001.
37. M. Davio. Kronecker Products and Shuffle Algebra. *IEEE Transactions on Computers*, C-30(2):116–125, February 1981.
38. L. de Alfaro, M. Kwiatkowska, G. Norman, D. Parker, and R. Segala. Symbolic Model Checking for Probabilistic Processes using MTBDDs and the Kronecker Representation. In S. Graf and M. Schwartzbach, editors, *TACAS'2000*, pages 395–410, Berlin, 2000. Springer, LNCS 1785.
39. S. Donatelli. Superposed stochastic automata: a class of stochastic Petri nets with parallel solution and distributed state space. *Performance Evaluation*, 18(1):21–36, July 1993.
40. S. Donatelli. Superposed Generalized Stochastic Petri Nets: Definition and Efficient Solution. In M. Silva, editor, *15th Int. Conf. on Application and Theory of Petri Nets*, Zaragoza, June 1994.
41. R. Enders, T. Filkorn, and D. Taubner. Generating BDDs for symbolic model checking in CCS. *Distributed Computing*, (6):155–164, 1993.

42. P. Fernandes, B. Plateau, and W.J. Stewart. Efficient Descriptor-Vector Multiplications in Stochastic Automata Networks. *Journal of the ACM*, 45(3):381–414, May 1998.
43. G. Franceschinis and M. Ribaudo. Efficient Performance Analysis Techniques for Stochastic Well-Formed Nets and Stochastic Process Algebras. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets II: Applications*, pages 386–437. Springer, LNCS 1492, 1998.
44. E. Frank. Codierung und numerische Analyse von Transitionssystemen unter Verwendung von MTBDDs. Student's thesis, Universität Erlangen–Nürnberg, October 1999 (in German).
45. E. Frank. Erweiterung eines MTBDD-basierten Werkzeugs für die Analyse stochastischer Transitionssysteme. Technical Report Inf 7, 01/00, Universität Erlangen–Nürnberg, January 2000 (in German).
46. M. Fujita, P. McGeer, and J.C.-Y. Yang. Multi-terminal Binary Decision Diagrams: An efficient data structure for matrix representation. *Formal Methods in System Design*, 10(2/3):149–169, April/May 1997.
47. R. German. *Performance Analysis of Communication Systems — Modelling with Non-Markovian Stochastic Petri Nets.* Wiley, 2000.
48. N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and Distributed System Design: The Integration of Functional Specification and Performance Analysis Using Stochastic Process Algebras. In *Proc. of PERFORMANCE 1993, Tutorial*, pages 121–146. Springer LNCS 729, 1993.
49. G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Probabilistic Analysis of Large Finite State Machines. In *31st Design Automation Conf.*, pages 270–275, San Diego, CA, June 1994. ACM/IEEE.
50. G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Symbolic Algorithms to Calculate Steady-State Probabilities of a Finite State Machine. In *European Design Automation Conf.*, pages 214–218, Paris, February 1994. IEEE.
51. G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Markovian Analysis of Large Finite State Machines. *IEEE Transactions on CAD*, 15(12):1479–1493, Dec. 1996.
52. H. Hermanns. *Interactive Markov Chains.* PhD thesis, Universität Erlangen-Nürnberg, September 1998. Arbeitsberichte des IMMD No. 32/7.
53. H. Hermanns, U. Herzog, and J.-P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 2001. to appear.
54. H. Hermanns, U. Herzog, U. Klehmet, V. Mertsiotakis, and M. Siegle. Compositional performance modelling with the TIPPtool. *Performance Evaluation*, 39(1-4):5–35, January 2000.
55. H. Hermanns, U. Herzog, and V. Mertsiotakis. Stochastic Process Algebras - Between LOTOS and Markov Chains. *Computer Networks and ISDN systems (CNIS)*, 30(9-10):901–924, 1998.
56. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov Chain Model Checker. In S. Graf and M. Schwartzbach, editors, *TACAS'2000*, pages 347–362, Berlin, 2000. Springer, LNCS 1785.
57. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. Towards model checking stochastic process algebra. In W. Grieskamp, T. Santen, and B. Stoddart, editors, *2nd Int. Conf. on Integrated Formal Methods*, pages 420–439, Dagstuhl, November 2000. Springer, LNCS 1945.
58. H. Hermanns, M. Kwiatkowska, G. Norman, D. Parker, and M. Siegle. On the use of MTBDDs for performability analysis and verification of stochastic systems. (in preparation).
59. H. Hermanns, J. Meyer-Kayser, and M. Siegle. Multi Terminal Binary Decision Diagrams to Represent and Analyse Continuous Time Markov Chains. In B. Plateau, W.J. Stewart, and M. Silva, editors, *3rd Int. Workshop on the Numerical Solution of Markov Chains*, pages 188–207. Prensas Universitarias de Zaragoza, 1999.
60. H. Hermanns and M. Rettelbach. Syntax, Semantics, Equivalences, and Axioms for MTIPP. In U. Herzog and M. Rettelbach, editors, *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, pages 71–88. Arbeitsberichte des IMMD No. 27/4, Universität Erlangen-Nürnberg, July 1994.

61. H. Hermanns and M. Siegle. Bisimulation Algorithms for Stochastic Process Algebras and their BDD-based Implementation. In J.-P. Katoen, editor, *ARTS'99, 5th Int. AMAST Workshop on Real-Time and Probabilistic Systems*, pages 144–264. Springer, LNCS 1601, 1999.
62. U. Herzog. Leistungsbewertung und Modellbildung für Parallelrechner. *Informationstechnik (it)*, 31(1):31–38, 1989. (in German).
63. U. Herzog. Performance Evaluation and Formal Description. In V.A. Monaco and R. Negrini, editors, *Advanced Computer Technology, Reliable Systems and Applications, Proceedings*, pages 750–756. IEEE Comp. Soc. Press, 1991.
64. J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.
65. J. Hillston and V. Mertsiotakis. A Simple Time Scale Decomposition Technique for Stochastic Process Algebras. *The Computer Journal*, 38(7):566–577, December 1995. Special issue: Proc. of the 3rd Workshop on Process Algebras and Performance Modelling.
66. O.C. Ibe and K.S. Trivedi. Stochastic Petri Net Models of Polling Systems. *IEEE Journal on Selected Areas in Communications*, 8(9):1649–1657, December 1990.
67. P. Kemper. Reachability analysis based on structured representation. In J. Billington and W. Reisig, editors, *Application and Theory of Petri Nets*, pages 269–288. Springer, LNCS 1091, 1999.
68. C. Kim and A.K. Agrawala. Analysis of the Fork–Join Queue. *IEEE Transactions on Computers*, 38(2):250–255, 1989.
69. W. Kleinöder. *Stochastische Bewertung von Aufgabenstrukturen für hierarchische Mehrrechnersysteme.* PhD thesis, Universität Erlangen–Nürnberg, Arbeitsberichte des IMMD No. 15/10, August 1982 (in German).
70. W. Knottenbelt. Generalized Markovian Analysis of Timed Transition Systems. Master's thesis, University of Cape Town, June 1996.
71. M. Kwiatkowska, G. Norman, and R. Segala. Automated Verification of a Randomised Distributed Consensus Protocol Using Cadence SMV and PRISM. Technical Report CSR-01-1, School of Computer Science, University of Birmingham, January 2001.
72. V. Mertsiotakis. Time Scale Decomposition of Stochastic Process Algebra Models. In E. Brinksma and A. Nymeyer, editors, *Proc. of 5th Workshop on Process Algebras and Performance Modelling.* CTIT Technical Report series, No. 97-14, University of Twente, June 1997.
73. V. Mertsiotakis. *Approximate Analysis Methods for Stochastic Process Algebras.* PhD thesis, Universität Erlangen-Nürnberg, 1998.
74. V. Mertsiotakis and M. Silva. Throughput Approximation of Decision Free Processes Using Decomposition. In *Proc. of the 7th Int. Workshop on Petri Nets and Performance Models*, pages 174–182, St. Malo, June 1997. IEEE CS-Press.
75. M. Meyer. Entwurf eines spezialisierten Coprozessors für die Manipulation von Binären Entscheidungsdiagrammen. Student's thesis, Universität Erlangen–Nürnberg, January 2001 (in German).
76. A. Miner. Efficient solution of GSPNs using matrix diagrams. In *Petri Nets and Performance models (PNPM).* IEEE Computer Society Press, 2001. (to appear).
77. A. Miner and G. Ciardo. Efficient reachability set generation and storage using decision diagrams. In H. Kleijn and S. Donatelli, editors, *Application and Theory of Petri Nets 1999*, pages 6–25. Springer, LNCS 1639, 1999.
78. A.S. Miner, G. Ciardo, and S. Donatelli. Using the exact state space of a Markov model to compute approximate stationary measures. *Performance Evaluation Review*, 28(1):207–216, June 2000. Proc. of ACM SIGMETRICS 2000.
79. M.K. Molloy. Performance Analysis Using Stochastic Petri Nets. *IEEE Transactions on Computers*, C-31:913–917, September 1982.
80. R. Nelson and A.N. Tantawi. Approximate Analysis of Fork/Join Synchronization in Parallel Queues. *IEEE Transactions on Computers*, 37(6):739–743, 1988.
81. D. Parker. *Implementation of symbolic model checking for probabilistic systems.* PhD thesis, School of Computer Science, University of Birmingham, 2001. (to appear).
82. B. Plateau. On the Synchronization Structure of Parallelism and Synchronization Models for Distributed Algorithms. In *Proc. of ACM SIGMETRICS*, pages 147–154, Austin, TX, August 1985.

83. B. Plateau and K. Atif. Stochastic Automata Network for Modeling Parallel Systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, 1991.
84. B. Plateau and J.-M. Fourneau. A Methodology for Solving Markov Models of Parallel Systems. *Journal of Parallel and Distributed Computing*, 12:370–387, 1991.
85. B. Plateau, J.-M. Fourneau, and K.-H. Lee. PEPS: A Package for Solving Complex Markov Models of Parallel Systems. In *Proc. of the 4th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 341–360, Palma (Mallorca), September 1988.
86. M. Reiser and S. Lavenberg. Mean Value Analysis of Closed Multichain Queueing Networks. *Journal of the ACM*, 27(2):313–322, 1980.
87. J.A. Rolia and K.C. Sevcik. The Method of Layers. *IEEE Transactions on Software Engineering*, 21(8):689–700, August 1995.
88. W.H. Sanders and J.F. Meyer. Reduced Base Model Construction Methods for Stochastic Activity Networks. *IEEE Journal on Selected Areas in Communications*, 9(1):25–36, January 1991.
89. C. Sauer and E. McNair. The Evolution of the Research Queueing Package RESQ. In *Proc. of the First Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, Paris, May 1984.
90. M. Sczittnick. Techniken zur funktionalen und quantitativen Analyse von Markoffschen Rechensystemmodellen. Master's thesis, Universität Dortmund, August 1987 (in German).
91. M. Siegle. Using Structured Modelling for Efficient Performance Prediction of Parallel Systems. In G.R. Joubert, D. Trystram, F.J. Peters, and D.J. Evans, editors, *Parallel Computing: Trends and Applications, Proc. of the Int. Conf. ParCo93*, pages 453–460. North-Holland, 1994.
92. M. Siegle. *Beschreibung und Analyse von Markovmodellen mit großem Zustandsraum.* PhD thesis, Universität Erlangen–Nürnberg, 1995 (in German).
93. M. Siegle. Compositional Representation and Reduction of Stochasitic Labelled Transition Systems based on Decision Node BDDs. In D. Baum, N. Müller, and R. Rödler, editors, *MMB'99*, pages 173–185, Trier, September 1999. VDE Verlag.
94. M. Siegle. Behaviour analysis of communication systems: Stochastic modelling and analysis. Habilitation thesis, University of Erlangen-Nürnberg, 2001 (to appear).
95. H.A. Simon and A. Ando. Aggregation of Variables in Dynamic Systems. *Econometrica*, 29:111–138, 1961.
96. F. Somenzi. CUDD: Colorado University Decision Diagram Package, Release 2.3.0. User's Manual and Programmer's Manual, September 1998.
97. W. Stewart, K. Atif, and B. Plateau. The Numerical Solution of Stochastic Automata Networks. Rapport Apache 6, Institut IMAG, LGI, LMC, Grenoble, November 1993.
98. W.J. Stewart. MARCA: Markov Chain Analyzer, A Software Package for Markov Modeling. In W.J. Stewart, editor, *Numerical Solution of Markov Chains.* Marcel Dekker, 1991.
99. W.J. Stewart. *Introduction to the numerical solution of Markov chains.* Princeton University Press, 1994.
100. M. Veran and D. Potier. QNAP2: A Portable Environment for Queueing Systems Modelling. In *Proc. of the First Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, Paris, May 1984.
101. M. Woodside, J.E. Neilson, D.C. Petriu, and S. Majumdar. The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server-like Distributed Software. *IEEE Transactions on Computers*, 44(1):20–34, January 1995.