Equivalence Checking 40 Years After: A Review of Bisimulation Tools

Hubert Garavel Frédéric Lang

Univ. Grenoble Alpes, INRIA, CNRS, Grenoble INP, LIG, France
E-mail: {hubert.garavel,frederic.lang}@inria.fr

Abstract

Equivalence checking is a formal verification approach that consists in proving that two programs or models are related modulo some equivalence relation, or that one is included in the other modulo some preorder relation. In the case of concurrent systems, which are often represented using labelled transition systems, the relations used for equivalence checking are bisimulations and their simulation preorders. In the case of probabilistic or stochastic systems, which are usually represented using Markov chains, the relations used for equivalence checking are lumpability, probabilistic and stochastic equivalences, and their associated preorders. The present article provides a synthetic overview of 40 years of research in the design of algorithms and software tools for equivalence checking.

1 Introduction

The present article was written in honor of Frits Vaandrager and included in a collective *Festschrift* book offered to him at the occasion of his 60th birthday.

Frits Vaandrager has published an impressive list of papers addressing very diverse topics in formal methods and concurrency theory, among which: operational semantics and SOS rules, process algebra, Petri nets, input-output automata, timed automata and real-time models, probabilistic automata, hybrid input-output automata and hybrid systems, action-based and state-based temporal logics, testing theory, automata learning, as well as formal modelling and verification of many industrial protocols. Among such a wealth of contributions, we have chosen to focus on bisimulations and equivalence checking, a topic that Frits Vaandrager contributed to advance significantly.

In formal methods, one never proves that a system (or a program) is correct *in itself*, but only that it is correct *with respect to* its specifications. Thus, formal verification does not consist in checking one artefact, but in comparing two artefacts one against the other, i.e., a system against its specifications that express desirable, expected properties. Depending on the formalism used for specifications, two cases need to be distinguished:

- If the system and its specifications are expressed in two different formalisms, one needs to prove that the system *satisfies* its specifications. A major approach is *model checking* [97], in which specifications are expressed in some temporal logic.
- If the system and its specifications are expressed in the same formalism, one needs to prove that the system is *equivalent* to its specifications or, in a weaker form, that the system *contains* or is *included* in its specifications. Such verification approaches are usually referred to as *equivalence checking* and, more often than not, the system is a much more complex artefact than its specifications, in which many low-level implementation details are abstracted away.

The present article presents a brief history of equivalence checking in the context of concurrent systems — leaving aside the widespread application of equivalence checking in the hardware-design industry to make sure that logic-synthesis tools preserve the intended behaviour of circuits.

The foundations of equivalence checking for concurrent systems have been laid in the 1970s. On the practical side, the protocol-engineering community developed verification techniques based on the systematic exploration of all reachable states of a concurrent system [371] [389]. On the theoretical side, it became clear that the semantics of concurrent systems could be adequately represented using state-transition models, such as LTSs (Labelled Transition Systems) [268] or *Kripke structures* [281] — although alternative models, such as Petri nets, would also coexist.

Finding the right equivalence relation to compare two state-transition models describing two concurrent systems was a non-trivial problem, as the two main equivalence relations known at that time were not appropriate: on the one hand, *graph isomorphism* was too strong, requiring both models to be strictly identical modulo permutations, whereas both models often have different levels of abstraction; on the other hand, *language equivalence* was too weak, only checking that the sets of traces of both models are the same, thus failing to make distinctions such as $a.(b + c)$ vs $(a.b) + (a.c)$, which are essential as far as the semantics of concurrent systems is considered.

To address this problem, *behavioural* equivalences have been introduced. These are equivalence relations situated between graph isomorphism and language equivalence: they are coarser than the former and finer than the latter. Important examples of behavioural equivalence are *strong equivalence* and *observational equivalence* [339] (the latter being also called *weak equivalence*). A major breakthrough was made with the concept of *bisimulation* [356], which provides a conceptual framework for these equivalences — see [20] for an insightful account of these foundational steps of concurrency theory. In a nutshell, bisimulation identifies all states having the same future, i.e., all states from which one can execute the same (possibly infinite) trees of actions.

There exist many behavioural relations; an overview of them can be found in [406]. Each equivalence relation comes with its associated *preorder*. Therefore, equivalence checking consists in verifying that two systems are related modulo some equivalence relation, or that one system is included in the other modulo some preorder relation. Fortunately, not all of these relations are needed in practice: our experience shows that most real-life case studies can be addressed using only a handful of well-chosen relations.

In the case of probabilistic and stochastic systems, which are usually represented using Markov chains or models derived from Markov chains, bisimulation coincides with the concept of *lumpability* [269] and, thus, serves as a basis for the definition of probabilistic and stochastic equivalences on Markov chains.

The present article provides a retrospective account of 40 years of research in equivalence-checking techniques for concurrent systems. Compared to prior surveys on bisimulations [406] [20], we focus here on the design of algorithms and software tools for the implementation of bisimulations on finite-state systems, leaving aside all aspects related to theorem proving — see [364] for a survey on bisimulation as a proof method. The article is organized chronologically: Sections 2–5 present the main achievements done during the 1980s, 1990s, 2000s, and 2010s decades, respectively, and Section 6 gives a few concluding remarks.

2 Retrospective of the 1980s

2.1 Bisimulation Tools Based on Term Rewriting

The book of Robin Milner on CCS [339] and the seminal article of David Park providing a co-inductive definition of bisimulation [356], together with subsequent publications [340] [341], laid the theoretical foundations for a deep research area, in which bisimulations are closely associated to process calculi. The definition of process calculi using either structural operational semantics [361] [362] [363] or algebraic semantics [27] initially led to consider term rewrite engines and theorem provers as natural approaches for implementing process calculi and bisimulations.

Among such early attempts, CIRCAL [336] was a tool that used algebraic laws to perform proofs of equivalence between two process-calculus programs.

ECRINS [314] was a tool for manipulating process calculi, the operators of which were defined using conditional rewrite rules. A semi-decision algorithm for strong bisimulation was implemented in ECRINS, and the tool was able to prove strong bisimulation automatically between CCS-like programs, without process recursion but with free process variables. Another use of ECRINS was to prove the correctness of semantic translation from one process calculus to another [133] [134].

BIP (Bisimulation Prover) [84] was a verification tool for protocol specifications written in the ECCS language. This tool used an enhanced version of Sanderson's algorithm for bisimulation [376]; this enhanced version removes one limitation of Sanderson's by accepting the presence of nondeterminism and τ-transitions, i.e., internal (or silent, or non-observable) actions of an LTS.

CRLAB [124] [125] was a tool that used term rewriting to compare processes written in CCS or basic LOTOS (i.e., the subset of LOTOS [254] without value passing) modulo observational equivalence.

2.2 Algorithms for Bisimulations on Finite-State Systems

However, alternative approaches to term rewriting quickly emerged. In these approaches, each program is first translated into a finite-state LTS, which is only possible if the program does not have an infinite number of states (e.g., does not handle unbounded data types nor infinitely many concurrent processes) and if the number of states is small enough to fit into the memory of available computers (i.e., does not face the well-known *state explosion* problem, which is a limiting factor for the verification of complex programs).

Assuming that an LTS has been successfully generated, one then executes a different, bisimulation-specific algorithm to *minimize* this LTS according to a chosen bisimulation (e.g., strong, observational, etc.); given that bisimulations are equivalence relations, such a minimal LTS always exists and is unique modulo a renaming of states. The same minimization algorithm can also be reused for *equivalence checking* purpose, i.e., to compare whether two LTSs are bisimilar or not: this is done by applying a minimization algorithm on the disjoint union of both LTSs and checking whether initial states are in the same equivalence class.

In a nutshell, these alternative approaches give up the generality of term rewriting (which can handle infinite-state programs but, in practice, is quite limited in the size of programs that can be handled) to adopt a less general approach (which only handles finite-state programs, but of a greater complexity that exceeds the capabilities of human reasoning). Finite-state approaches also have the advantage of being fully automated, meaning that bisimulation can be decided without human intervention.

In the 1980s, two key algorithms for computing bisimulation on finite LTSs have been proposed. In two articles [262] [261] that recently received the 2021 Dijkstra Prize, Paris Kanellakis and Scott Smolka proposed an algorithm for checking the equivalence of CCS processes. Their algorithm performs *relational coarsest partitioning* (also known as *coarsest partition refinement*): initially, all states are in the same set, which is progressively partitioned to separate states having different futures. The time and space complexities of their algorithm are $O(mn)$ and $O(m + n)$, respectively, where m is the number of transitions and n the number of states[1].

[1] We use the same definitions of m and n throughout this article.

In a subsequent article [354], Robert Paige and Robert Tarjan proposed a more efficient algorithm for the same problem. Their algorithm only addresses strong bisimulation. Its time and space complexities are $O(m \log n)$ and $O(m + n)$, respectively.

2.3 Early Bisimulation Tools

The Kanellakis-Smolka and Paige-Tarjan algorithms aroused great interest and triggered the development of numerous software tools.

SCAN [347] was probably the first implementation of bisimulations. This tool could reduce and compare, with respect to strong or observational equivalence, networks of finite automata composed together using parallel operators and *filtering* (a combination of hiding and relabeling to abstract away the internal behaviour of composed automata).

AUTO [410] [411] [412] [66] [315] [67] [129] was a tool for the analysis and manipulation of finite LTSs, which could be either specified using the MEIJE process calculus [17] or described graphically using the AUTOGRAPH editor and composed in parallel and connected together using synchronisation ports and communication wires. AUTO offered primitives to minimize an LTS modulo strong bisimulation, observational bisimulation, or trace language equivalence, to determinize an LTS, to compute the transitive closure of τ-transitions, and to eliminate τ-loops and single τ-transitions. It could also check the equivalence of two LTSs for strong or observational bisimulation.

VENUS [344] [385] was a tool for minimizing and comparing CCS processes modulo strong or observational bisimulations. It also supported operations dedicated to τ-transitions, such as the elimination of τ-chains and τ-circuits.

TAV [190] [53] [54] was a tool that could check strong or observational equivalence between two CCS processes, and also check whether a CCS process satisfied a temporal-logic formula expressed in the Hennessy-Milner logic extended with recursion. A distinctive feature of TAV was the possibility to provide an *explanation* given as a Hennessy-Milner logic formula computed using a backtracking algorithm [240]. Another algorithm for providing a temporal-logical formula that differentiates two LTSs that are not strongly bisimilar was proposed in [98].

SQUIGGLES [49] [50] was a tool that extended the Paige-Tarjan algorithm to compare LTSs or programs written in basic LOTOS with respect to strong equivalence, observational equivalence, or testing equivalence.

WINSTON [320] was a tool that could compare networks of finite-state processes using strong and observational equivalences implemented with the Kanellakis-Smolka algorithm.

ALDEBARAN [146] [147] [148] [149] was a tool for minimizing and comparing LTSs according to strong bisimulation, observational equivalence, acceptance model equivalence, or safety equivalence; it could also display the equivalence class of each state of the LTS. Contrary to most other tools written in functional/declarative languages such as Lisp, Prolog, etc., ALDEBARAN was written in C and implemented an adaptation of the Paige-Tarjan algorithm. The simple file format used by ALDEBARAN for storing LTSs became popular and has been known since as the AUT format[2].

The ACP Bisimulation Tool [436] was a tool to compare programs written in the ACP_τ process calculus [28] modulo a weak relation called *bisimulation with τ-abstraction*. It computed a transitive closure of τ-transitions, followed by a relational coarsest partition algorithm.

PIPN [19] was a tool that could minimize, according to observational equivalence or trace equivalence, the labelled reachability graphs generated from Petri nets. PIPN was part of ESTIM [378], a simulation and verification tool for communicating state machines specified in the ESTELLE language [253].

CWB (Concurrency Workbench) [105] [104] was an integrated tool for verifying networks of finite-state processes described in CCS, then converted to a state-transition model called *transition graphs*. Three kinds of analyses were supported by CWB: model checking (evaluation of temporal-logic formulas written in the propositional *mu*-calculus), equivalence checking (comparison of two transition graphs modulo $\mathcal{C}$-bisimulation, a generic relation from which strong equivalence, observational equivalence, must equivalence, and testing failures equivalence can be derived), and preorder checking (comparison of two transition graphs modulo $\mathcal{C}$-preorder, from which bisimulation divergence preorder, may preorder, must preorder, and testing preorders can be obtained). Equivalence checking was based on the Kanellakis-Smolka algorithm, while preorder checking was done using an ad hoc, less efficient algorithm. An early application of CWB to the analysis of mutual exclusion algorithms can be found in [418].

Most of the aforementioned tools performed at least two different tasks: (i) generate the LTSs corresponding to concurrent systems described either as networks of automata composed in parallel, or as process-calculi specifications; and (ii) minimize and/or compare these LTSs modulo various equivalence or preorder relations. Both tasks are subject to antagonistic implementation constraints: task (i) must store in memory the concrete contents of each state of the LTS being generated, while the transitions of the LTS can just be written to disk; conversely, task (ii) must store in memory all the transitions, and can ignore the concrete contents of states, which can just be treated like abstract numbers. It is thus difficult for the same tool to be optimally efficient in both tasks. One solution is to have separate tools, dedicated either to task (i) or to task (ii).

The first instance of a specialized tool for task (i) was CÆSAR [168] [182], a tool for translating value-passing LOTOS specifications into LTSs encoded in the various file formats accepted by ALDEBARAN, AUTO, CWB, PIPN, SCAN, SQUIGGLES, etc. A distinctive feature of ALDEBARAN, CÆSAR, and their companion tools later integrated in the CADP toolbox [151] was to be plain, ordinary Unix commands that could be directly invoked from the shell

[2] `https://cadp.inria.fr/man/aut.html`

with appropriate command-line options, and that communicated via files containing LTSs. Such an architectural design was a major departure from other bisimulation tools, most of which were built around custom command-line interpreters (with ad hoc primitives for, e.g., loading, generating, minimizing, and saving LTSs). It has been progressively adopted by other tools during the next decades.

3 Retrospective of the 1990s

3.1 New Algorithms for Bisimulations

The 1990s have been a very active decade, in which new equivalences, preorders, and algorithms have been invented and implemented in tools.

Testing equivalences [123], introduced in the 1980s, define that two models (typically a high-level specification and a lower-level implementation) are equivalent iff an observer interacting with them cannot distinguish one from the other by means of testing. Although these equivalences are weaker (i.e., less discriminative) than bisimulations, Cleaveland and Hennessy gave an algorithm [101] based on a characterization of these relations in terms of bisimulation. This algorithm was implemented in CWB.

Safety equivalence [56], named so because it preserves safety properties, is the equivalence relation obtained by the logical conjunction of two *$\tau^*.a$ preorders* [155] that abstract away τ-transitions. Algorithms for minimizing and comparing LTSs modulo safety equivalence were implemented in ALDEBARAN.

Branching bisimulation [407] [408] was introduced by van Glabbeek and Weijland after observing that Milner's observational equivalence does not fully respect the branching structure of LTSs. To compute branching bisimulation, which is slightly stronger than observational equivalence, Groote and Vaandrager proposed an algorithm [200] [201] for *relational coarsest partitioning with stuttering*. A key idea of the algorithm is the possibility to compress each strongly connected component of states connected by τ-transitions into a single state beforehand, using existing linear-time algorithms. This algorithm has a worst-case time complexity $O(mn)$ and a space complexity $O(m + n)$. Thus, branching bisimulation can be implemented more efficiently than observational equivalence, which has progressively been superseded by branching equivalence, except in the CCS community where observational equivalence remained popular, probably by fidelity to the foundations set by Milner. The Groote-Vaandrager algorithm was implemented in an efficient prototype named Branching Tool, as well as in other tools, among which ALDEBARAN and AUTO.

Groote and Vaandrager also proposed a format of Plotkin-style structural operational semantics rules [202] that guarantees, among other properties, that strong bisimulation is a congruence on the states of any transition system described using this format.

It was shown [365] that, for all equivalences between strong bisimulation and trace equivalences (i.e., almost all equivalences of practical interest), checking equivalence of two networks of LTSs is PSPACE-hard because, in the general case, one cannot avoid computing the product state space of each network.

De Nicola and Vaandrager investigated the logical characterization of branching bisimulation, and exhibited three logics such that two LTSs are branching bisimilar iff they exactly satisfy the same formulas of these logics [127] [128].

There have been other attempts at computing bisimulations using model checkers, such as MEC [12] (later integrated in ALTARICA [13]) and SPIN [140] [141]. For instance, MEC did not implement dedicated algorithms for equivalence checking, but could define bisimulation as a concise formula in fix-point logic [11] [194], thus enabling bisimulation to be verified as a particular case of model checking.

3.2 Algorithms for On-the-Fly Verification

While the mainstream approach, so far, consisted in first generating LTSs before minimizing them or checking their equivalence, novel *on-the-fly* approaches emerged, where an LTS is *reduced* (i.e., partially minimized) while being generated, or where equivalence between two LTSs is checked while these LTSs are generated. Such approaches, which had already been experimented successfully for model checking, may avoid storing the entire set of states and/or transitions, especially when one does not need to fully explore both LTSs to decide that they are not equivalent.

An on-the-fly algorithm for equivalence checking was first proposed by Fernandez and Mounier [153] [345]. Their algorithm was not based on partition refinement, which requires to compute the sets of states beforehand, but instead explored the states of the synchronous product of the LTSs under comparison, until a verdict can be given. Various adaptations of their algorithm to compute strong, branching, observational, and $\tau^*.a$ bisimulations, as well as safety equivalence and the corresponding preorders, have been implemented in ALDEBARAN [154]. A variant of their algorithm was later proposed, which does not store all states of the synchronous product, but only enough states so that the verification terminates [258].

A similar approach for on-the-fly equivalence checking modulo strong and observational bisimulations was implemented in the LOLA tool [348].

Lee and Yannakakis proposed another on-the-fly algorithm [300] for minimizing an LTS modulo strong bisimulation while this LTS is being generated.

A generic architecture, named OPEN/CÆSAR [170], was proposed for developing on-the-fly verification tools rationally. This architecture achieves a clear separation between, on the one hand, an LTS that is generated on-the-fly (e.g., from a process-calculus specification or a network of communicating automata) and, on the other hand, a verification algorithm that explores this LTS guided by a specific goal or property to be proven. One of the first applications of OPEN/CÆSAR was REDUCTOR [170], a tool that reduced LTSs on the fly modulo $\tau^*.a$ equivalence.

Partial-order reductions are techniques for reducing the size of the state space, by exploiting the independence of transitions to avoid unnecessary interleavings. Such techniques, provided they preserve a behavioural relation of interest, may be particularly useful when doing minimization or equivalence checking on the fly. Partial-order reductions were first studied in the context of linear-time semantics, then in the context of branching-time semantics to check branching bisimulation [400] [184], strong bisimulation (between two LTSs having the same independent transitions) [244], and *failures refinement* [422].

3.3 Algorithms for Symbolic Verification

New algorithms were proposed, based on a symbolic representation of the system under verification in the form of a BDD (Binary Decision Diagram) [75].

An approach was proposed, in which CCS processes are translated to BDDs, thus allowing bisimulations (encoded as temporal logic formulas) to be computed on such a symbolic representation [139].

Bouali and De Simone proposed a symbolic algorithm [60] dedicated to the minimization of networks of LTSs according to strong bisimulation, as well as variants for observational and branching bisimulations.

Another symbolic algorithm [305] was proposed for comparing CCS processes modulo observational bisimulation, and compared, on a few benchmarks, to the equivalence-checking algorithm implemented in CWB.

Fisler and Vardi [158] [159] [160] implemented, using BDDs, three minimization algorithms in the setting of finite transition systems, the states of which are labelled with atomic propositions. Their experimental results indicate that BDDs are not a silver bullet for bisimulation minimization: the number of BDD nodes needed to compute the bisimulation relation grows quickly, outweighing the potential benefits of minimizing the global state space before performing symbolic model checking.

3.4 Algorithms for Compositional Verification

Given a concurrent system, e.g., a network of LTSs, compositional verification consists in generating a reduced (or even minimized) LTS for this system, modulo some behavioural relation of interest [146] [320] [372] [430] [391] [390] [399]. If this relation is a congruence for the operators of the network (typically, parallel composition and label hiding), which is the case of most bisimulations, then the generation can be done incrementally, by alternating operator applications and reductions of the intermediate LTSs.

However, while doing so, state explosion may happen in intermediate LTSs, due to the existence of transitions that are fireable locally, but not globally, as they could not meet the synchronization constraints in the entire network. Graf, Steffen, and Lüttgen proposed an approach to solve this problem, based on *interface specifications* (represented as LTSs) that cut off globally unfireable transitions [192] [193]. This approach was extended by Krimm and Mounier, and implemented in PROJECTOR [280], an on-the-fly tool developed using the OPEN/CÆSAR architecture.

A related approach, called *compositional reachability* [92], performs compositional reduction modulo observational equivalence while verifying a property represented as an LTS. This approach was implemented in the TRACTA tool [92], which also supported a variant [91] of Graf-Steffen-Lüttgen interface specifications.

A comprehensive survey on compositional verification, from the 1990s to the present, can be found in [179].

3.5 Enhanced Bisimulation Tools

Most bisimulation tools developed in the 1980s quickly became unavailable, due to lack of software maintenance at a time where processors, operating systems, and programming languages were rapidly evolving. There was, however, the notable exception of three tools (namely, ALDEBARAN, AUTO, and CWB), the development of which steadily progressed during the 1990s.

ALDEBARAN [155], so far mainly based on the Paige-Tarjan algorithm, was enhanced in four directions: counterexample generation algorithms, implementation of the Groote-Vaandrager algorithm for branching bisimulation, novel on-the-fly equivalence-checking algorithms [155] [345], and symbolic verification algorithms based on BDDs [152] [271] [270]. ALDEBARAN was a key component of the CADP verification toolbox [151] [150] [169] [71] [172], which gathered a growing number of closely interconnected tools. Its synergy with CADP brought ALDEBARAN at least three benefits: the existence of an efficient LTS generator, the aforementioned compiler CÆSAR for LOTOS; the availability of BCG (*Binary Coded Graphs*), a compact file format[3] for storing large LTSs; and the integration within EUCALYPTUS, a graphical user interface that simplified the invocation of ALDEBARAN with command-line options.

ALDEBARAN, together with companion tools of CADP, has been used in numerous case studies[4] by scientists of many universities worldwide. Among the case studies involving equivalence checking, one can mention, in chronological order: a car overtaking protocol [142], dynamically changing communication structures [166], an ATM switch [157], a plain ordinary telephone service (starting from an existing specification [144]), a framework for groupware development [272], a trusted third-party protocol between video-on-demand service

[3] https://cadp.inria.fr/man/bcg.html
[4] http://cadp.inria.fr/case-studies

providers and customers [299] [183], a railway signalling system [164], a bounded retransmission protocol [323] [324] (starting from an existing specification [203]), the TCP Internet transport protocol [379], feature interactions in telephony systems [277], several variants of distributed leader election algorithms for unidirectional ring networks [181], a bus arbiter of a multiprocessor architecture [86], the link layer protocol of the IEEE-1394 serial bus [381], a departure clearance protocol for air traffic [121], testing of a distributed leader election algorithm [397] [396], a flow-control protocol for a high-speed network [222], patterns for software architecture styles [216], an invoicing system [380] [382], a protocol for road traffic control [424] [423], asynchronous circuits [431] [432] [433], a reliable data-transfer service [312], an abstraction-display-controller model for user interfaces [321], a distributed cluster file system [358] [359], an ISO high-speed transport protocol [25], highly reliable and reusable CORBA applications [279], a leader election protocol for home audio/video networks [367], synchronous hardware [215] [214] [213], a protocol for deploying intelligent telecommunication services [14] [15] [16], a handshake authentication protocol [298], a datalink system for air traffic control [373] [374] [375], a radio protocol for mobile telecommunications [309], a protection system against cloning of cellular phones [350] [349], and hardware/software codesign [429] [22]. Such an impressive list clearly indicates that formal verification should not be restricted to model checking only, and that equivalence checking and bisimulations also have a major role to play.

The evolution of AUTO and its associated graphical editor AUTOGRAPH also continued in the 1990s [369] [317] [318] [370]. Companion tools were developed, among which: MAUTO, which extended AUTO to a large family of process calculi; FCTOOL [58], which implemented efficient partitioning algorithms (derived from the Groote-Vaandrager algorithm) for strong, branching, and observational bisimulations; and the FC2TOOLS set [65] [64] [62] [63], which gathered a collection of tools offering both *explicit* and *implicit* (i.e., BDD-based) bisimulation algorithms, and designed around FC2, a dedicated file format for LTSs and networks of LTSs.

A few case studies were done using these tools, e.g., a bus instrumentation protocol specified in LOTOS [18], a sliding window protocol specified in LOTOS [316], a lift controller specified in ESTEREL [207], and a secure datagram protocol specified in LOTOS [188].

CWB also pursued its evolution during the 1990s. The performance of its LTS minimization algorithm was assessed in [143], the introduction of priorities was presented in [259], and an overall presentation of CWB can be found in [106]. Contrary to ALDEBARAN and AUTO, the development of which remained centralized in Grenoble and Sophia-Antipolis, respectively, CWB adopted a more decentralized approach: the original software was maintained in Edinburgh [342] [386][5] in collaboration with other universities (e.g., Sussex), while a new branch emerged in the United States under the successive names of Concurrency Factory [100] [102], NCSU Concurrency Workbench [108], Concurrency Workbench of

[5] See also [387] and [388] for a reflection on the development of verification tools.

North Carolina [107] [109], and Concurrency Workbench of the New Century [103] [110].

Besides the use of CWB by Swedish telecom companies to analyze parts of the GSM and ISDN protocols, other case studies used the equivalence-checking features of CWB, e.g., alternating bit protocol with lossy buffers [99], instruction pipelining and parallel memory models for the SPARC architecture [282], formalization in LOTOS of the GKS computer-graphics standard [366], and access monitoring for information flow security [161].

3.6 New Bisimulation Tools

Besides enhancements brought to already existing bisimulation tools, many new tools were developed during the 1990s.

PisaTool [249] [252] used term rewriting to compare CCS processes for strong, observational, or branching equivalences. Instead of relying on interleaving semantics, this tool took into account some aspects of true concurrency by introducing a parametric representation of finite-state systems and bisimulations.

SEVERO [83] was both an equivalence checker and a model checker for the ACTL temporal logic [126], applicable to finite-state processes that were converted to BDDs and reduced modulo observational equivalence using a symbolic minimization algorithm.

ARA (Advanced Reachability Analysis) [401] was a tool for minimization and equivalence checking of basic LOTOS processes modulo *CFFD equivalence* [398], a relation weaker than observational equivalence. ARA supported compositional LTS construction and partial-order reductions. It was used, e.g., to verify a small protocol for client-server communication [274] and, using compositional verification and in combination with CADP, a reliable data-transfer service [312].

CCSTOOL2 [409] was a modular tool performing various computations on finite-state CCS descriptions, including minimization for strong and observational bisimulations.

JACK (Just Another Concurrency Kit) [61] was an integrated environment, with a graphical user interface, gathering verification tools designed around the FC2 file format. JACK supported model checking of ACTL logic formulas, as well as equivalence checking using CRLAB (term rewriting) and AUTO (algorithms for finite-state models). The equivalence-checking features of JACK have been used to verify hardware components [122], a railway interlocking system [30] [31], and, in combination with CADP, an abstraction-display-controller model for user interfaces [321].

YAPV [36] was a research tool based on true concurrency, instead of interleaving semantics. It used a variant of the Kanellakis-Smolka algorithm to check bisimulation between processes.

BIDMIN [218] was a minimization tool written in Ada. It supported strong bisimulation, as well as variants of observational and branching bisimulations (e.g., rooted, divergence-preserving, etc.), these variants differing only in the definition of the initial partition of the LTS state set.

FDR2 [368] was a verification tool for the CSP process calculus [74]. It implemented strong bisimulation, as well as *failures* and *failures/divergences* equivalences, which were preferred to observational or branching bisimulation for checking equivalence and refinement (i.e., preorder inclusion) of CSP processes. FDR2 was used to formally verify STATEMATE statecharts [167].

ARC [355] was another formal verification tool for CSP, which had similar equivalence checking features as FDR2, but in which LTSs were represented symbolically using ordered BDDs.

XEVE [59] was a verification environment for ESTEREL specifications, which used the Bouali-De Simone algorithm to perform symbolic minimization modulo strong bisimulation.

ObjectGeode, an industrial environment for simulating and verifying SDL programs, was equipped with on-the-fly equivalence checking capabilities by a connection to CADP [273].

Getting an accurate panorama of bisimulation tools by reading publications is uneasy, as related work was not always cited properly. Fortunately, a few benchmarks and surveys exist, e.g., a gentle introduction [145] to the principles of bisimulation tools, followed by an overview of ALDEBARAN, AUTO, CWB, PVE, TAV, etc.; a comparative evaluation [278] of bisimulation techniques in ALDEBARAN, AUTO, CWB, TAV, WINSTON, etc.; an overview [313] of tools available in the early 1990s, among which ALDEBARAN, AUTO, CWB, ECRINS, FCTOOL, MAUTO, MEC, TAV, etc.; a study [217] of the respective performances of ALDEBARAN, BIDMIN, CWB, and FCTOOL/HOGGAR; and a survey [250] [251] of tools for the analysis of distributed systems specified using process calculi.

3.7 Bisimulation Tools for Timed and Hybrid Systems

There has been a variety of tools for modelling and analyzing real-time systems. First attempts used a discrete-time model, in which a special "tick" action represented the elapsing of one time unit. Since this approach did not scale to realistic systems, requiring too many transitions to model long delays, dense-time models supported by methods from continuous mathematics were progressively adopted, either as timed extensions of process calculi [118] [195] [301] [302] or as *timed automata* [6], i.e., automata extended with real-valued clock variables. Tools for the analysis of timed systems often involved the generation of an untimed finite-state abstraction of the state space (e.g., a *region graph* [5]), that takes time constraints into account.

The first minimization algorithms for timed automata [3] [4] generated a minimized region graph on the fly, by extending a prior algorithm [57] that directly generates a minimal LTS for strong bisimulation (rather than first generating an LTS that is minimized later).

VERSA [96] [95] [93] was a tool for the analysis of discrete-time systems with prioritized resources and events, described in a timed process calculus called ACSR (Algebra of Communicating Shared Resources). Besides checking equivalences by means of term rewriting, VERSA automatically translated ACSR processes with bounded delays to LTSs, and compared these (finite-state) LTSs modulo strong bisimulation and $\tau^*.a$ equivalence using the Kanellakis-Smolka algorithm. XVERSA [94] was an extension of VERSA with a graphical user interface.

TPWB (Timing and Probability Workbench) [165] was a tool for analyzing finite-state, discrete-time systems described in TPCCS [208] [209], an extension of CCS with time and probabilities. It used an adaptation of the Kanellakis-Smolka algorithm with lumping of probabilities to perform minimization and equivalence checking on finite-state systems modulo strong bisimulation.

EPSILON [189] [191] [85] was an extension of TAV for analyzing dense-time systems described in TMS (Timed Modal Specifications), a formalism for timed networks inspired by TCCS [420] (a variant of CCS with a delay operator) extended with “may” and “must” modalities. EPSILON implemented equivalence and preorder checking for strong and observational bisimulations, as well as time-abstracted versions of these relations that gave finite representations of these networks. When the check was negative, EPSILON could generate, like TAV, a distinguishing formula in timed Hennessy-Milner logic. EPSILON was used to analyze a steam generator [284] [283].

KRONOS [119] [394] [434] [69] [70] was a tool for minimization, equivalence checking, and preorder checking of timed automata. It used a time-abstracting bisimulation and linear constraints on the clocks of the timed automaton to generate an untimed abstraction of the state space. The resulting LTS could be minimized and checked using ALDEBARAN. KRONOS was used in many case studies, and also as a back-end [238] for the verification of systems described in ET-LOTOS [301] [302], a timed extension of LOTOS. Later, Extended KRONOS (or OPEN/KRONOS) [393] enhanced KRONOS with a richer input language and on-the-fly verification capabilities based upon the OPEN/CÆSAR architecture.

TREAT (Timed Reachability Analysis Tool) [263] was a tool for timed automata. To fight the state explosion problem, TREAT generated untimed abstractions that preserved two behavioural relations named *history equivalence* and *transition bisimulation*.

RT-MEC [81], a component of PEP (Programming Environment based on Petri nets), was a tool for the analysis of systems modelled as Petri nets with dense time. It implemented equivalence checking modulo strong bisimulation and timed bisimulation, using partial-order reductions and on-the-fly techniques to generate a reduced region graph.

Hybrid automata are infinite-state models to describe digital programs interacting with an analog environment. Although hybrid automata encompass timed automata, there have been few tools implementing bisimulations on hybrid automata. A notable exception was HYTECH [221] [220], which used bisimulations to reduce hybrid systems to finite LTSs. When such a reduction was possible, the hybrid automaton could be model checked.

A performance comparison between EPSILON, KRONOS, HYTECH, and the UPPAAL tool [24] (which does not use the concept of bisimulations) can be found in [295].

3.8 Bisimulation Tools for Probabilistic and Stochastic Systems

The advent of process calculi in the 1980s gave a new impulse to the study of Markovian models. In order to finely describe both the functional and performance aspects of concurrent systems, process calculi have been extended in various ways with non-functional concepts, such as probabilities and random durations, e.g., [233] [242]. At a lower abstraction level, extended models have been proposed, combining LTSs, to model functional aspects, and DTMCs (Discrete-Time Markov Chains) or CTMCs (Continuous-Time Markov Chains), to model performance aspects.

To analyze such models, in addition to traditional techniques (steady-state and transient analyses, simulation, etc.) and novel quantitative model-checking approaches, various bisimulations have also been defined, such as probabilistic bisimulations [296] [297] [241] [245] [48] [310] [383] and Markovian/stochastic bisimulations [233] [226] [234] [76]. These equivalences combine the concepts of bisimulation for the LTSs and of lumpability for the Markov chain aspects.

At first, such bisimulations were used for algebraic proofs of equivalence and performance calculations on simple models. But dedicated algorithms were progressively designed, e.g., [21], and implemented in already existing or novel software tools.

TIPPtool [275] [232] was a tool for creating and analyzing concurrent systems described in the TIPP language, a stochastic process calculus based on LOTOS. Initiated in 1992, TIPPtool has been progressively extended with many features for functional and performance analyses. Bisimulations on LTSs and Markovian models played a major role in TIPPtool, especially for applying compositional minimization techniques [239] [227] [224] [225] in order to contain state explosion. Strong and observational bisimulations were implemented using the Kanellakis-Smolka algorithm, but an export to CADP's AUT format was also available; the Baier algorithm was used for Markovian bisimulations; symbolic algorithms based on BDDs were also developed for this purpose [235] [236]. Various systems have been analyzed using TIPPtool, e.g., an alternating bit protocol [231], a communication protocol [162], a plain-old telephone system (tackled using TIPPtool in combination with CADP) [229], and an hospital communication system [225].

TwoTowers [35] [34] was a tool for the functional and performance analysis of systems modelled in EMPA, a stochastic timed process calculus. TwoTowers combined two existing tools: the aforementioned Concurrency Workbench of North Carolina (for model checking, equivalence checking, and preorder checking) and MarCA (for steady-state and transient performance analysis); it also supported the strong extended Markovian reward bisimulation [33]. The equivalence-checking capabilities of TwoTowers have been used to analyze a randomized distributed algorithm for the dining philosophers problem [35] and a token ring protocol [32].

The APNN (Abstract Petri Net Notation) toolbox [23] [77] [78] was a set of tools for the functional and quantitative analysis of discrete-event dynamic systems. It offered many features, such as model checking, numerical analysis of Markov chains, and simulation, but also supported various kinds of bisimulations used to reduce, by means of compositional minimization techniques, the size of the generated state spaces.

3.9 Bisimulation Tools for Mobile Systems

Mobile process calculi, such as the π-calculus [337] [338], enable the description of concurrent systems with potentially infinite state spaces, due to the dynamic creation of agents and communication channels. For such systems, verification approaches based on finite-state systems are hardly applicable, especially if they require an exhaustive exploration of the state space before verification can take place. For such problems, formal proofs (which are outside the scope of this survey) are the approach of choice [364]. Nevertheless, there have been attempts at developing automated equivalence-checking tools based on bisimulation theory for analyzing such systems.

MWB (Mobility Workbench) [413] [416] [414] was a tool for checking whether two π-calculus programs are equivalent with respect to open bisimulation [377]. MWB was based upon an on-the-fly algorithm (inspired from the Fernandez-Mounier approach) in which LTSs where generated on demand. MWB was later extended with an algorithm for checking symbolic hyperequivalence [357] for the fusion calculus [415, Chapter 7].

The π-environment [156] was a tool for checking (strong or observational) early and late bisimulations on finite-state processes specified in the π-calculus. This tool did not work on the fly, but relied on the aforementioned AUTO and JACK tools.

One can also mention algorithms for checking symbolic (strong or observational) bisimulations between value-passing LTSs extended with variables, inputs, and assignments [306] [219] [307] [303] [248]. These algorithms have been transposed to compute bisimulations for the π-calculus [304] [308], but do not seem to have been implemented.

4 Retrospective of the 2000s

4.1 New Algorithm for Strong Bisimulation

A new *fast bisimulation* algorithm [136] [137] for strong bisimulation was proposed, which extends the Paige-Tarjan algorithm with a notion of *state rank* defined as the maximum distance to a sink state (if any), not counting transitions which are internal to strongly connected components. Fast bisimulation is more efficient than the Paige-Tarjan algorithm on LTSs containing sink states, especially acyclic LTSs. It was implemented in the COPS checker [360] for security properties. A symbolic version of this algorithm was implemented using BDDs [135].

A survey on the complexity of some of the behavioural equivalences presented in [406], considering their application to finite- and infinite-state models, can be found in [343].

4.2 New Bisimulation Tools

In the 2000s, a new generation of bisimulation tools appeared, which progressively superseded the tools developed during the previous decades.

BCG_MIN [174] used the Kanellakis-Smolka and the Groote-Vaandrager algorithms to minimize LTSs modulo strong and branching bisimulations. BCG_MIN was released as a component of the CADP toolbox and used BCG as a native file format to represent LTSs compactly. Since BCG_MIN was globally more efficient than ALDEBARAN and could handle larger graphs, the original ALDEBARAN tool was replaced in 2005 by a backward-compatible shell script invoking BCG_MIN and other tools of CADP [176].

The μCRL toolset [38] [39] used partition-refinement algorithms to perform equivalence checking and minimization of LTSs modulo strong and branching bisimulations. It generated LTSs in the AUT and BCG formats of CADP, and implemented the OPEN/CÆSAR interface, so that most tools of CADP could be used on μCRL specifications. It was succeeded by the mCRL2 toolset [198], which also supports equivalence checking based on bisimulations.

TVT (Tampere Verification Tool) [417], which succeeded the ARA tool, implemented strong bisimulation and CFFD equivalence.

CHISIGMA [55] was a tool environment that could minimize, modulo strong and branching bisimulations, LTSs generated from specifications written in the process language χ_σ.

ABC [72] checked the equivalence of π-calculus processes modulo open bisimulation.

TAPAS [82], which was developed for teaching purpose, implemented comparison and minimization of LTSs modulo strong, observational, and branching bisimulations.

LTSA (Labelled Transition System Analyser) [319] performed minimization modulo observational equivalence of LTSs generated from FSP (Finite-State Processes) specifications.

4.3 Bisimulation Tools Using On-the-Fly Verification

Significant advances in on-the-fly reduction and on-the-fly equivalence checking of LTSs have been made during the 2000s.

A key tool of CADP for on-the-fly verification is EXP.OPEN, which explores the state space of networks of LTSs composed together using synchronization vectors or parallel composition operators borrowed to various process calculi (CCS, CSP, LOTOS, LNT, or μCRL), as well as hiding, renaming, cutting, or priority operators. EXP.OPEN was enhanced with on-the-fly partial-order reductions that preserve strong, branching, or stochastic branching bisimulations [289].

The ARCATS tool [90] [89] implemented a different approach to on-the-fly reduction: it incrementally generated an LTS reduced for branching bisimulation, by alternating steps of partial LTS generation and steps of branching minimization of the partial LTS already generated.

A generic library, named CÆSAR_SOLVE [328], was developed, as part of CADP, for solving BESs (Boolean Equation Systems) [7] [326]. BESs are effective models in which both model-checking and equivalence-checking problems can be conveniently encoded. Given a BES, the CÆSAR_SOLVE library explores an LTS on the fly, using the features provided by OPEN/CÆSAR, in order to compute the truth value of certain BES variables.

To reduce an LTS partially, during its generation, one can apply the notion of *τ-confluence* [199], a form of partial-order reduction that analyzes τ-transitions and preserves branching bisimulation. Based on this idea, an algorithm was proposed [45] and implemented in the μCRL toolset, with the help of an automated theorem prover to identify τ-confluent transitions.

This idea was also implemented in CADP by enhancing the aforementioned REDUCTOR tool that performed $\tau^*.a$ reduction. The new version of REDUCTOR [327] supports many other reductions, among which τ-confluence, *τ-closure* (transitive reflexive closure over τ-transitions), *τ-compression* (collapsing of strongly connected components made of τ-transitions), safety reduction, etc. It uses OPEN/CÆSAR and CÆSAR_SOLVE to detect τ-confluent transitions on the fly.

The definition of τ-confluence was later generalized to visible actions [291], so as to enable better reductions in a compositional-verification setting.

CADP was also enriched with another tool, named BISIMULATOR [26] [330] [331], which checks whether two LTSs are equivalent modulo strong, branching, observational, or $\tau^*.a$ bisimulations, as well as safety equivalence. The comparison is expressed in terms of a BES, which is resolved on the fly using CÆSAR_SOLVE, one of the two LTSs being explored on demand using OPEN/CÆSAR. This general approach based on BESs subsumes the dedicated Fernandez-Mounier algorithms for checking bisimulations on the fly. BISIMULATOR, REDUCTOR, and BCG_MIN have been used in many case studies, e.g., the verification of a plant unit for drilling of metal products [329].

Conversely, BESs (which can be seen as a particular form of LTSs) can be minimized using strong bisimulation and an extension of strong bisimulation called *idempotence-identifying bisimulation* [267]. Such minimization preserves the truth values and, if applied before solving the original BESs, may speed up the resolution.

Equivalence checking of infinite-state models can be expressed in terms of PBESs (Parameterized Boolean Equation Systems) [326], an extension of BESs designed for model checking value-passing temporal-logic formulas [325]. This problem was addressed in [88] for branching and observational bisimulations. In this approach, the comparison of two models (represented using linear process equations) is encoded as a PBES, which is generated automatically, but whose resolution cannot be fully automated and may thus require human intervention.

4.4 Bisimulation Tools Based on Compositional Verification

Compositional verification makes intensive use of bisimulations and requires different tools for, e.g., generating, minimizing, and composing LTSs in parallel.

To ease compositional verification, CADP was enriched with SVL [173] [288], which is both a scripting language and a compiler. SVL enables verification scenarios to be specified simply and executed efficiently, and thus offers an alternative to graphical user interfaces when dealing with complex, repetitive tasks. The SVL language has operators to generate, compose in parallel, minimize, and compare LTSs modulo various equivalence relations; LTSs can be either given in low-level formats (AUT, BCG, etc.) or specified using high-level languages (LOTOS, LNT [180], etc.); other operators support compositional-verification strategies and abstractions based on (handwritten or automatically generated [290]) interface specifications. The SVL compiler translates SVL scripts into shell scripts that invoke the appropriate CADP tools (BCG_MIN, EXP.OPEN, PROJECTOR, etc.), relieving users from taking care of command-line options and auxiliary files. SVL was used in several case studies [111] [395] [332].

4.5 Bisimulation Tools Based on Parallel/Distributed Computing

Blom and Orzan proposed both sequential and parallel/distributed algorithms for minimizing LTSs modulo strong bisimulation [40] [42] [43] [44] and branching bisimulation [41] (later improved in [46]). These algorithms, which are gathered in [352], are based on partition refinement and the novel concept of *signatures* (two states having different signatures cannot be bisimilar).

Although their sequential algorithms have worst-case time complexity $O(mn^2)$, which is higher than the best algorithms for strong and branching bisimulations, they exhibit more opportunities for parallelization.

Indeed, their parallel algorithms, which distribute an LTS across several machines, exhibit a linear speed-up, meaning that the time taken by these algorithms linearly decreases when the number of machines increases. A policy based on abstract interpretation for distributing the LTS across machines was proposed in [353].

4.6 Bisimulation Tools Based on Symbolic Verification

The SIGREF tool [428] implemented several equivalence relations (including strong, branching, observational, and *orthogonal* [29] bisimulations, as well as safety equivalence) using sequential algorithms combining Blom-Orzan signatures with a BDD representation of the LTS. Several papers were published, detailing an algorithm for branching bisimulation [426], optimization techniques for BDD-based bisimulation computation [427], and an efficient algorithm for Markov chains [425].

4.7 Bisimulation Tools for Timed Systems

Research on timed bisimulations, which was very active in the 1990s, has seemingly slowed down during the 2000s. This is most likely related to the decline of research on timed process calculi, progressively replaced by simpler models based on timed automata. As a consequence, timed bisimulations and timed modal μ-calculus have been replaced by more elementary analyses, such as reachability and safety properties computed on networks of timed automata. In this respect, the good performance of UPPAAL may have favored this evolution (see the related discussion in [421]). However, one can mention two advances in timed bisimulations during the 2000s.

A new bisimulation-based *faster-than* preorder [311] was proposed to compare asynchronous processes with respect to their worst-case timing behaviour.

A new equivalence-checking algorithm for timed branching bisimulation of communicating timed automata was designed and implemented in the RED tool [419], improving over prior algorithms for timed branching bisimulation.

4.8 Bisimulation Tools for Probabilistic and Stochastic Systems

The aforementioned BCG_MIN tool was designed to handle, not only LTSs, but also extended models combining normal transitions (labelled with an action), probabilistic transitions (whose label is a probability), and/or stochastic transitions (whose label is the rate parameter of an exponential distribution governing a random delay). Such models encompass DTMCs, which contain only probabilistic transitions, CTMCs, which contain only stochastic transitions, IMCs (Interactive Markov Chains) [73] [223] [230], which contain both normal and stochastic transitions, and IPCs (Interactive Probabilistic Chains) [114] [112], which contain both normal and probabilistic transitions. BCG_MIN can minimize such models for various equivalences that combine strong or branching bisimulation with lumpability and, in the stochastic case, maximal progress. BCG_MIN, together with other tools for steady-state and transient analysis [228], has made CADP the actual successor of the TIPPtool [171] and has been used in several performance studies.

Version 4.0 of TwoTowers [2] implemented equivalence checking, for strong and observational Markovian equivalences, of architectural specifications written in a language named Æmilia. This tool was used to assess the securing strategy implemented in a trusted device for security architectures [1].

The PEPA workbench [185] was extended to support PEPA nets [186], a formalism that describes mobile agent systems using Petri nets in which mobile program code (expressed using the stochastic process calculus PEPA) moves across the places of the net. This workbench implemented *net bisimulation* [187], an equivalence relation for minimizing marking graphs of PEPA nets, and was used in many case studies.

Bisimulation algorithms for minimizing DTMCs and CTMCs have been proposed [131] [132]. These algorithms, which are based on symbolic representations and Blom-Orzan signatures, have been implemented in the SIGREF tool [425]. Independently, it has been evidenced that minimizing a DTMC or CTMC before analysis improves performance [265].

Symmetry reductions that preserve strong probabilistic bisimulation have been proposed [286], which may generate DTMCs, CTMCs, and MDPs (Markov Decision Processes) by several orders of magnitude smaller. These reductions have been implemented, using BDDs, in the PRISM model checker [287].

Finally, a minimization algorithm for observational bisimulation of acyclic IMCs with inputs and outputs was proposed, with an application to the analysis of DFTs (Dynamic Fault Trees) [116].

5 Retrospective of the 2010s

5.1 New Bisimulation Tools

The BCG_MIN tool of CADP was entirely rewritten in 2010. The new version 2.0 [177] [178] relies on the sequential Blom-Orzan algorithms for strong and branching bisimulations. Despite the worst-case time complexity $O(mn^2)$ of the Blom-Orzan algorithms is higher than the worst-case time complexity $O(mn)$ of the Groote-Vaandrager and Kanellakis-Smolka algorithms implemented in BCG_MIN 1.0, the new version of BCG_MIN is statistically faster, which seems to indicate that the worst cases for signature-based algorithms rarely occur in practice (crafted worst-case examples are given in [196, Section 8]).

BCG_CMP[6] uses the same algorithms as BCG_MIN 2.0 to check the equivalence of two LTSs modulo strong, branching, divergence-preserving branching, and observational bisimulations. When both LTSs are not equivalent, it generates an LTS explaining where and why bisimulation does not hold.

LTSMIN [47] [264] is a comprehensive model-checking tool set, which also implements strong and branching bisimulations using the distributed Blom-Orzan algorithms. LTSMIN is used as a backend by, e.g., the mCRL2 toolset [115].

The aforementioned FDR2 toolset for analyzing CSP processes was extended with new features [10], among which minimization algorithms [68] for strong, observational, and delay bisimulations to reduce state spaces before verification.

RELTS [335] [334] and T-BEG [276] are tools that implement strong bisimulation in terms of game theory. Another tool [79] defines various simulation relations in terms of an antagonistic two-player game. Also, game-theoretic definitions of branching and divergence-preserving branching bisimulations have been given in [120].

Educational motivation was behind the development of tools such as CAAL (Concurrency Workbench, Aalborg Edition) [9], which implements various strong or observational, timed or untimed, equivalences and preorders, and PSEUCO [37], which supports strong bisimulation. These tools exhibit fancy Web interfaces that help teaching concurrency theory in university courses.

One can also mention SMART [346], which implements strong and observational bisimulations using multiway decision diagrams, GREASE [163], which checks strong and observational bisimulations on-the-fly using syntactic criteria to try finding a counter-example as soon as possible, and an implementation of branching bisimulation dedicated to the reduction of BIP (Behaviour-Interaction-Priority) models [351].

5.2 Bisimulation Tools for Probabilistic and Stochastic Systems

Foundations were laid for strong and observational bisimulations and preorders on Markov automata, a combination of probabilistic automata and IMCs [138].

The aforementioned minimization tool BCG_MIN 2.0 was equipped with probabilistic and stochastic bisimulations [113]. For these relations (as well as for strong and branching bisimulations on LTSs), BCG_MIN 2.0 was found to use less memory and to be faster than BCG_MIN 1.0 [178]. Support for the same probabilistic and stochastic bisimulations was added in BCG_CMP too.

MRMC (Markov Reward Model Checker) [435] [266] is a tool for verifying properties (expressed as CSL or PCTL temporal-logic formulas with their reward extensions) on probabilistic models. To alleviate state explosion, MRMC may minimize these models modulo strong bisimulation.

[6] `http://cadp.inria.fr/man/bcg_cmp.html`

Polynomial algorithms for probabilistic observational bisimulation on probabilistic automata [237] [211] and alternating probabilistic bisimulation on interval MDPs [212] were proposed and implemented. The latter relation was shown to be compositional [210].

An algorithm [130] for directly generating a DTMC minimized modulo probabilistic bisimulation from a probabilistic program described by guarded commands was proposed and implemented in the PRISM model checker using the SMT solver Z3. Another approach for generating, using PRISM, a DTMC from an RTL (Register Transfer Level) description and minimizing it, using SIGREF, modulo probabilistic bisimulation can be found in [87].

An approach was proposed [384] to accelerate the model checking of PCTL formulas on probabilistic automata, by iteratively refining an abstraction of a probabilistic automaton, using incrementally computed bisimulations and without resorting to any kind of counterexample analysis.

A fast algorithm for strong probabilistic bisimulation [204] was proposed and implemented in the mCRL2 toolset [80].

Another fast algorithm [257] was given to minimize, modulo branching bisimulation, DTMCs with labelled states.

Probabilistic bisimulation was also applied to infinite-state parameterized systems, i.e., systems with an arbitrary number of processes [243]. The approach was experimented in a prototype tool that was not made public.

5.3 Bisimulation Tools for Mobile Systems

Two new tools for analyzing extensions of the π-calculus were released during the 2010s.

PWB (Psi-Calculus Workbench) [51] [52] was a generic tool for analyzing mobile processes by means of symbolic simulation and equivalence checking modulo symbolic (strong or observational) bisimulations [260].

SPEC [392] was an equivalence-checking tool for open bisimulation on security protocols specified in the spi-calculus.

5.4 Bisimulation Tools Based on Parallel/Distributed Computing

There have been commendable efforts to parallelize mainstream partition-refinement algorithms for minimizing LTSs modulo strong bisimulation. A parallel version of the Paige-Tarjan algorithm, in combination with Blom-Orzan signatures, was proposed in [285], and a parallel version of the Kanellakis-Smolka algorithm is given in [322].

Combinations of symbolic techniques and parallel algorithms have also been explored in the second half of the 2010s. SIGREFMC [402] [405] was a bisimulation tool providing the same functionalities as SIGREF, but based on SYLVAN [403]

[404], a parallel implementation of BDDs on multi-core architectures.

While SIGREFMC encoded bisimulations as partitions of the state space, in the lineage of partition-refinement algorithms, a different approach was investigated in [246] [247], where strong bisimulation was encoded directly as a relation, like in the Bouali-De Simone algorithm.

5.5 Bisimulation Tools Based on Compositional Verification

A new approach, called *smart reduction* [117], was proposed for the compositional minimization of networks of LTSs. Smart reduction analyzes the synchronizations between concurrent processes to infer a suitable order in which processes are composed and minimized. Such a heuristic, which tries to avoid state explosion by keeping the size of intermediate LTSs as small as possible, was implemented in the SVL scripting language of CADP [175].

A new family of equivalence relations, named *sharp bisimulations* [294], which combine strong bisimulation and divergence-preserving branching bisimulation [333], was defined. Sharp bisimulations provide effective reductions while preserving given temporal-logic formulas [293]. They have been implemented in the BCG_MIN and BCG_CMP tools of CADP, with user-friendly support in SVL.

Smart reduction and sharp bisimulations play a major role in modern approaches to compositional verification. Together with recent developments [292] around the idea of *partial model checking* [8], they enabled scientists from Grenoble and Pisa to solve nearly all the parallel problems of the RERS[7] verification challenge in 2019[8] and 2020[9].

5.6 Recent Results for Strong and Branching Bisimulations

An asymptotic lower bound $\Omega((m+n)\log n)$ on the time complexity of partition refinement algorithms was established [197].

A new (successively revised, improved, and simplified) minimization algorithm for branching bisimulation was proposed [205] [206] [196] [255] [256]. Its worst-case time complexity $O(m \log n)$ is lower than that of the Groote-Vaandrager algorithm, which has been the best-known algorithm since the early 1990s, and equal to that of the Paige-Tarjan algorithm, which is still the reference algorithm for strong bisimulation. This new algorithm has been implemented in the mCRL2 toolset.

[7] http://rers-challenge.org
[8] http://cadp.inria.fr/news12.html
[9] http://cadp.inria.fr/news13.html#section-3

6 Conclusion

Although model checking and equivalence checking have been discovered nearly at the same time in the early 1980s, model checking is now widespread in academia and industry, whereas equivalence checking plays a more discrete role. It nevertheless found numerous applications in the verification of communication protocols, hardware circuits, distributed systems, security systems, web services, etc. Actually, compared to model checking, equivalence checking presents several advantages:

- It is conceptually simpler, as it does not require learning another language (i.e., temporal logics) to express the properties under verification.
- It enables *visual checking*, an easy form of verification done by abstracting away certain observable actions of the system (i.e., by renaming them to τ-transitions), minimizing the resulting state space modulo some weak bisimulation, and visually inspecting the minimized state space if it is small enough.
- It may increase the effectiveness of model checking, as compositional state-space verification techniques based upon, e.g., congruence properties, smart reductions, and sharp bisimulations, are often capable of generating large state spaces that could not be explored otherwise.

For these reasons, we believe that equivalence checking should play a growing role in the future, in close combination with model checking. This could resolve the longstanding dilemma between *state-based models* (in which information is attached to states, as in Kripke structures) and *action-based models* (in which information is attached to transitions, as in LTSs and Markov chains) by giving an advantage to the latter models. Model checking is equally applicable to both action- and state-based models (although with slightly different temporal logics [126]), but most bisimulation tools have been designed to operate on action-based models, which suggests that the latter models are more suitable where model checking and equivalence checking are to be used together.

During the last forty years, the development of algorithms and tools for checking bisimulations on finite- or infinite-state systems has steadily progressed. These essential achievements have been spanning over several decades, which is no surprise, keeping in mind how theoretically involved are these algorithms and how technically involved are these tools subject to severe performance requirements. A remarkable example of such long-lasting research and commitment is the Groote-Vaandrager algorithm for branching bisimulation [201], which has been gradually refined to lower its complexity [256].

It is worth noticing that most of the bisimulation tools developed for equivalence checking are no longer available today. Quite often, publications are the only remaining indication that such tools have existed; in some cases (e.g., for the promising BIDMIN tool), formal publications are even lacking. A counterexample is the CADP toolbox, the bisimulation tools of which have been, over several decades, constantly enhanced or replaced by better, backward-compatible tools.

Software tools get obsolete due to incompatible evolutions of programming languages and operating systems, but they also get abandoned when their authors leave academia or move from one university to another; this suggests that overemphasis on professional mobility may hamper long-term development of perennial software tools.

Finally, the development of bisimulation tools has probably suffered from additional factors, among which: (i) the lack of standard file formats agreed upon by the community, beyond the rather inefficient AUT format; (ii) the lack of benchmark examples, with the notable exception of VLTS[10], which plays the role of a de-facto test suite; and (iii) the lack of yearly software competitions dedicated to equivalence checking. We hope that the present survey will draw the attention to the past achievements and future promises of this research field.

Acknowledgements

We are grateful to Rance Cleaveland, Rocco De Nicola, Jan Friso Groote, Laurent Mounier, Elie Najm, and the anonymous referees for their valuable comments about this article. We also would like to thank the DBLP and Google Scholar teams, whose long-term undertaking made it possible to tackle such a comprehensive study.

[10] `https://cadp.inria.fr/resources/vlts`

References

[1] Alessandro Aldini and Marco Bernardo. An Integrated View of Security Analysis and Performance Evaluation: Trading QoS with Covert Channel Bandwidth. In Maritta Heisel, Peter Liggesmeyer, and Stefan Wittmann, editors, *Proceedings of the 23rd International Conference on Computer Safety, Reliability, and Security (SAFECOMP'04), Potsdam, Germany*, volume 3219 of *Lecture Notes in Computer Science*, pages 283–296. Springer, September 2004.

[2] Alessandro Aldini and Marco Bernardo. TwoTowers 4.0: Towards the Integration of Security Analysis and Performance Evaluation. In *1st International Conference on Quantitative Evaluation of Systems (QEST 2004), Enschede, The Netherlands*, pages 336–337. IEEE Computer Society, September 2004.

[3] Rajeev Alur, Costas Courcoubetis, David L. Dill, Nicolas Halbwachs, and Howard Wong-Toi. An Implementation of Three Algorithms for Timing Verification Based on automata emptiness. In *Proceedings of the Real-Time Systems Symposium, Phoenix, Arizona, USA, December 1992*, pages 157–166. IEEE Computer Society, 1992.

[4] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, David L. Dill, and Howard Wong-Toi. Minimization of Timed Transition Systems. In Rance Cleaveland, editor, *Proceedings of the 3rd International Conference on Concurrency Theory (CONCUR'92), Stony Brook, NY, USA*, volume 630 of *Lecture Notes in Computer Science*, pages 340–354. Springer, August 1992.

[5] Rajeev Alur and David L. Dill. Automata For Modeling Real-Time Systems. In Mike S. Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90), Warwick University, England, UK*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, July 1990.

[6] Rajeev Alur and David L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.

[7] Henrik Reif Andersen. Model Checking and Boolean Graphs. *Theoretical Computer Science*, 126(1):3–30, 1994.

[8] Henrik Reif Andersen. Partial Model Checking. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science LICS (San Diego, California, USA)*, pages 398–407. IEEE Computer Society Press, June 1995.

[9] Jesper Rank Andersen, Nicklas Andersen, Søren Enevoldsen, Mathias M. Hansen, Kim G. Larsen, Simon R. Olesen, Jirí Srba, and Jacob K. Wortmann. CAAL: Concurrency Workbench, Aalborg Edition. In Martin Leucker, Camilo Rueda, and Frank D. Valencia, editors, *Proceedings of the 12th International Colloquium on Theoretical Aspects of Computing (ICTAC'15), Cali, Colombia*, volume 9399 of *Lecture Notes in Computer Science*, pages 573–582. Springer, October 2015.

[10] Philip J. Armstrong, Michael Goldsmith, Gavin Lowe, Joël Ouaknine, Hristina Palikareva, A. W. Roscoe, and James Worrell. Recent Developments in FDR. In P. Madhusudan and Sanjit A. Seshia, editors, *Proceedings of the 24th International Conference on Computer Aided Verification (CAV'12), Berkeley, CA, USA*, volume 7358 of *Lecture Notes in Computer Science*, pages 699–704. Springer, July 2012.

[11] André Arnold. Verification and Comparison of Transition Systems. In Marie-Claude Gaudel and Jean-Pierre Jouannaud, editors, *Proceedings of the International Conference on Theory and Practice of Software Development (TAP-SOFT'93), Orsay, France*, volume 668 of *Lecture Notes in Computer Science*, pages 121–135. Springer, April 1993.

[12] André Arnold, Didier Bégay, and Paul Crubillé. *Construction and Analysis of Transition Systems with MEC*, volume 3 of *AMAST Series in Computing*. World Scientific, 1994.

[13] André Arnold, Gérald Point, Alain Griffault, and Antoine Rauzy. The AltaRica Formalism for Describing Concurrent Systems. *Fundamenta Informaticae*, 40(2-3):109–124, 1999.

[14] Thomas Arts and Izak van Langevelde. How μCRL Supported a Smart Redesign of a Real-life Protocol. In Stefania Gnesi and Diego Latella, editors, *Proceedings of the 4th International ERCIM Workshop on Formal Methods for Industrial Critical Systems (Trento, Italy)*, pages 31–53. ERCIM, CNR, July 1999.

[15] Thomas Arts and Izak van Langevelde. Verifying a Smart Design of TCAP: A Synergetic Experience. Research Report SEN-R9910, CWI, 1999.

[16] Thomas Arts and Izak van Langevelde. Correct Performance of Transaction Capabilities. In Antti Valmari and Alex Yakovlev, editors, *Proceedings of the 2nd International Conference on Application of Concurrency to System Design (ICACSD'01), Newcastle upon Tyne, UK*, pages 35–42. IEEE Computer Society, June 2001.

[17] Didier Austry and Gérard Boudol. Algèbre de Processus et Synchronisation. *Theoretical Computer Science*, 30:91–131, 1984.

[18] Pierre Azema, Khalil Drira, and François Vernadat. A Bus Instrumentation Protocol Specified in LOTOS. In Juan Quemada, José Manas, and Enrique Vázquez, editors, *Proceedings of the 3rd International Conference on Formal Description Techniques FORTE'90 (Madrid, Spain)*. North-Holland, November 1990.

[19] Pierre Azéma, François Vernadat, and Jean-Christophe Lloret. Requirement Analysis for Communication Protocols. In Joseph Sifakis, editor, *Proceedings of the 1st Workshop on Automatic Verification Methods for Finite State Systems (CAV'89), Grenoble, France*, volume 407 of *Lecture Notes in Computer Science*, pages 286–293. Springer, June 1989.

[20] Jos C. M. Baeten and Davide Sangiorgi. Concurrency Theory: A Historical Perspective on Coinduction and Process Calculi. In Jörg H. Siekmann, editor, *Computational Logic*, volume 9 of *Handbook of the History of Logic*, pages 399–442. Elsevier, 2014.

[21] Christel Baier. Polynomial Time Algorithms for Testing Probabilistic Bisimulation and Simulation. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification (CAV'96), New Brunswick, NJ, USA*, volume 1102 of *Lecture Notes in Computer Science*, pages 50–61. Springer, July 1996.

[22] Fabrice Baray and Pierre Wodey. Verification in the Codesign Process by Means of LOTOS Based Model-Checking. In Stefania Gnesi, Ina Schieferdecker, and Axel Rennoch, editors, *Proceedings of the 5th International Workshop on Formal*

Methods for Industrial Critical Systems (FMICS'00), Berlin, Germany, GMD Report 91, pages 87–108, Berlin, April 2000.

[23] Falko Bause, Peter Buchholz, and Peter Kemper. A Toolbox for Functional and Quantitative Analysis of DEDS. In Ramón Puigjaner, Nunzio N. Savino, and Bartomeu Serra, editors, *Proceedings of the 10th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools (TOOLS'98), Palma de Mallorca, Spain*, volume 1469 of *Lecture Notes in Computer Science*, pages 356–359. Springer, September 1998.

[24] Johan Bengtsson, Kim Guldstrand Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. UPPAAL – A Tool Suite for Automatic Verification of Real-Time Systems. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Proceedings of the DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, Ruttgers University, New Brunswick, NJ, USA*, volume 1066 of *Lecture Notes in Computer Science*, pages 232–243. Springer, October 1995.

[25] Abderrahim Benslimane and Abdelhafid Abouaissa. XTP Specification and Validation with LOTOS. In *Proceedings of the Western MultiConference WMC'98, Communication Networks and Distributed Systems Modeling and Simulation CNDS'98 (San Diego, California, USA)*. Society for Computer Simulation International, January 1998.

[26] Damien Bergamini, Nicolas Descoubes, Christophe Joubert, and Radu Mateescu. BISIMULATOR: A Modular Tool for On-the-Fly Equivalence Checking. In Nicolas Halbwachs and Lenore Zuck, editors, *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05), Edinburgh, Scotland, UK*, volume 3440 of *Lecture Notes in Computer Science*, pages 581–585. Springer, April 2005.

[27] J. A. Bergstra and J. W. Klop. Process Algebra for Synchronous Communication. *Information and Computation*, 60(1–3):109–137, 1984.

[28] Jan A. Bergstra and Jan Willem Klop. Algebra of Communicating Processes with Abstraction. *Theoretical Computer Science*, 37:77–121, 1985.

[29] Jan A. Bergstra, Alban Ponse, and Mark van der Zwaag. Branching Time and Orthogonal Bisimulation Equivalence. *Theoretical Computer Science*, 309(1-3):313–355, 2003.

[30] Cinzia Bernardeschi, Alessandro Fantechi, and Stefania Gnesi. An Industrial Application for the JACK Environment. *Journal of Systems and Software*, 39(3):249–264, 1997.

[31] Cinzia Bernardeschi, Alessandro Fantechi, Stefania Gnesi, Salvatore Larosa, Giorgio Mongardi, and Dario Romano. A Formal Verification Environment for Railway Signaling System Design. *Formal Methods in System Design*, 12(2):139–161, 1998.

[32] M. Bernardo and M. Bravetti. Functional and Performance Modeling and Analysis of Token Ring Using EMPA. In Pierpaolo Degano, Ugo Vaccaro, and Giuseppe Pirillo, editors, *Proceedings of the 6th Italian Conference on Theoretical Computer Science*, 1998.

[33] Marco Bernardo. An Algebra-Based Method to Associate Rewards with EMPA Terms. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Proceedings of the 24th International Colloquium on Automata, Languages and Programming (ICALP'97), Bologna, Italy*, volume 1256 of *Lecture Notes in Computer Science*, pages 358–368. Springer, July 1997.

[34] Marco Bernardo. Implementing Symbolic Models for Value Passing in TwoTowers. In Boudewijn R. Haverkort, Henrik C. Bohnenkamp, and Connie U. Smith, editors, *Proceedings of the 11th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools (TOOLS'00), Schaumburg, IL, USA*, volume 1786 of *Lecture Notes in Computer Science*, pages 370–373. Springer, March 2000.

[35] Marco Bernardo, Rance Cleaveland, Steve Sims, and W. Stewart. TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems. In Stanislaw Budkowski, Ana R. Cavalli, and Elie Najm, editors, *Proceedings of the IFIP TC6/WG6.1 Joint 11th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols and 18th International Workshop on Protocol Specification, Testing and Verification (FORTE/PSTV'98), Paris, France*, volume 135 of *IFIP Conference Proceedings*, pages 457–467. Kluwer, November 1998.

[36] Alessandro Bianchi, Stefano Coluccini, Pierpaolo Degano, and Corrado Priami. An Efficient Verifier of Truly Concurrent Properties. In Victor E. Malyshkin, editor, *Proceedings of the 3rd International Conference on Parallel Computing Technologies (PaCT-95), Saint Petersburg, Russia*, volume 964 of *Lecture Notes in Computer Science*, pages 36–50. Springer, September 1995.

[37] Sebastian Biewer, Felix Freiberger, Pascal Leo Held, and Holger Hermanns. Teaching Academic Concurrency to Amazing Students. In Luca Aceto, Giorgio Bacci, Giovanni Bacci, Anna Ingólfsdóttir, Axel Legay, and Radu Mardare, editors, *Models, Algorithms, Logics and Tools – Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday*, volume 10460 of *Lecture Notes in Computer Science*, pages 170–195. Springer, 2017.

[38] Stefan Blom, Wan Fokkink, Jan Friso Groote, Izak van Langevelde, Bert Lisser, and Jaco van de Pol. μCRL: A Toolset for Analysing Algebraic Specifications. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01), Paris, France*, volume 2102 of *Lecture Notes in Computer Science*, pages 250–254. Springer, 2001.

[39] Stefan Blom, Jan Friso Groote, Izak van Langevelde, Bert Lisser, and Jaco van de Pol. New Developments Around the mCRL Tool Set. *Electronic Notes in Theoretical Computer Science*, 80:284–288, 2003.

[40] Stefan Blom and Simona Orzan. A Distributed Algorithm for Strong Bisimulation Reduction of State Spaces. *Electronic Notes in Theoretical Computer Science*, 68(4):523–538, 2002.

[41] Stefan Blom and Simona Orzan. Distributed Branching Bisimulation Reduction of State Spaces. *Electronic Notes in Theoretical Computer Science*, 89(1):99–113, 2003.

[42] Stefan Blom and Simona Orzan. Distributed State Space Minimization. *Electronic Notes in Theoretical Computer Science*, 80:109–123, 2003.

[43] Stefan Blom and Simona Orzan. A Distributed Algorithm for Strong Bisimulation Reduction of State Spaces. *Software Tools for Technology Transfer*, 7(1):74–86, 2005.

[44] Stefan Blom and Simona Orzan. Distributed State Space Minimization. *Software Tools for Technology Transfer*, 7(3):280–291, 2005.

[45] Stefan Blom and Jaco van de Pol. State Space Reduction by Proving Confluence. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification (CAV'02), Copenhagen, Denmark*, volume 2404 of *Lecture Notes in Computer Science*, pages 596–609. Springer, July 2002.

[46] Stefan Blom and Jaco van de Pol. Distributed Branching Bisimulation Minimization by Inductive Signatures. In Lubos Brim and Jaco van de Pol, editors, *Proceedings of the 8th International Workshop on Parallel and Distributed Methods in verifiCation (PDMC'09), Eindhoven, The Netherlands*, volume 14 of *Electronic Proceedings in Theoretical Computer Science*, pages 32–46, November 2009.

[47] Stefan Blom, Jaco van de Pol, and Michael Weber. LTSmin: Distributed and Symbolic Reachability. In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *Proceedings of the 22nd International Conference on Computer Aided Verification CAV 2010 (Edinburgh, UK)*, volume 6174 of *Lecture Notes in Computer Science*, pages 354–359. Springer, July 2010.

[48] Richard Blute, Josée Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for Labelled Markov Processes. In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science (LICS'97), Warsaw, Poland*, pages 149–158. IEEE Computer Society, June 1997.

[49] Tommaso Bolognesi and Maurizio Caneve. SQUIGGLES: A Tool for the Analysis of LOTOS Specifications. In Kenneth J. Turner, editor, *Proceedings of the 1st International Conference on Formal Description Techniques (FORTE'88), Stirling, Scotland*, pages 201–216. North-Holland, September 1988.

[50] Tommaso Bolognesi and Maurizio Caneve. Equivalence Verification: Theory, Algorithms, and a Tool. In Peter van Eijk, Chris A. Vissers, and Michel Diaz, editors, *The Formal Description Technique LOTOS*, pages 303–326. North-Holland, 1989.

[51] Johannes Borgström, Ramunas Gutkovas, Ioana Rodhe, and Björn Victor. A Parametric Tool for Applied Process Calculi. In Josep Carmona, Mihai T. Lazarescu, and Marta Pietkiewicz-Koutny, editors, *Proceedings of the 13th International Conference on Application of Concurrency to System Design (ACSD'13), Barcelona, Spain*, pages 180–185. IEEE Computer Society, July 2013.

[52] Johannes Borgström, Ramunas Gutkovas, Ioana Rodhe, and Björn Victor. The Psi-Calculi Workbench: A Generic Tool for Applied Process Calculi. *ACM Transactions on Embedded Computing Systems*, 14(1):9:1–9:25, 2015.

[53] Anders Børjesson, Kim Guldstrand Larsen, and Arne Skou. Generality in Design and Compositional Verification using TAV. In Michel Diaz and Roland Groz, editors, *Proceedings of the IFIP TC6/WG6.1 5th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'92), Perros-Guirec, France*, volume C-10 of *IFIP Transactions*, pages 449–464. North-Holland, October 1992.

[54] Anders Børjesson, Kim Guldstrand Larsen, and Arne Skou. Generality in Design and Compositional Verification Using TAV. *Formal Methods in System Design*, 6(3):239–258, 1995.

[55] Victor Bos. ChiSigma Manual. available from ResearchGate, 2002.

[56] Ahmed Bouajjani, Jean-Claude Fernandez, Susanne Graf, Carlos Rodríguez, and Joseph Sifakis. Safety for Branching Time Semantics. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Proceedings of the 18th International Colloquium on Automata, Languages and Programming (ICALP'91), Madrid, Spain*, volume 510 of *Lecture Notes in Computer Science*, pages 76–92. Springer, July 1991.

[57] Ahmed Bouajjani, Jean-Claude Fernandez, and Nicolas Halbwachs. Minimal Model Generation. In Edmund M. Clarke and Robert P. Kurshan, editors, *Proceedings of the 2nd International Workshop on Computer-Aided Verification (CAV'90), Rutgers, NJ, USA*, volume 531 of *Lecture Notes in Computer Science*, pages 197–203. Springer, June 1990.

[58] Amar Bouali. Weak and Branching Bisimulation in FcTool. Research Report 1575, INRIA, 1992.

[59] Amar Bouali. XEVE: An ESTEREL Verification Environment (Version v1_3). Technical Report 214, INRIA, 1997.

[60] Amar Bouali and Robert de Simone. Symbolic Bisimulation Minimisation. In Gregor von Bochmann and David K. Probst, editors, *Proceedings of the 4th International Workshop on Computer Aided Verification (CAV'92), Montreal, Canada*, volume 663 of *Lecture Notes in Computer Science*, pages 96–108. Springer, June 1992.

[61] Amar Bouali, Stefania Gnesi, and Salvatore Larosa. The Integration Project for the JACK Environment. Research Report CS-R9443, CWI, 1994.

[62] Amar Bouali, Annie Ressouche, Valérie Roy, and Robert de Simone. The FC2TOOLS Set. In Martin Wirsing and Maurice Nivat, editors, *Proceedings of the 5th International Conference on Algebraic Methodology and Software Technology (AMAST'96), Munich, Germany*, volume 1101 of *Lecture Notes in Computer Science*, pages 595–598. Springer, July 1996.

[63] Amar Bouali, Annie Ressouche, Valérie Roy, and Robert de Simone. The Fc2Tools Set. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification (CAV'96), New Brunswick, New Jersey, USA*, volume 1102 of *Lecture Notes in Computer Science*, pages 441–445. Springer, July–August 1996.

[64] Amar Bouali, Annie Ressouche, Valérie Roy, and Robert de Simone. The Fc2Tools Set (Tool Demonstration). In Tiziana Margaria and Bernhard Steffen, editors, *Proceedings of the 2nd International Workshop Tools and Algorithms for Construction and Analysis of Systems (TACAS'96), Passau, Germany*, volume 1055 of *Lecture Notes in Computer Science*, page 396. Springer, March 1996.

[65] Amar Bouali, Annie Ressouche, Valérie Roy, and Robert de Simone. The FC-TOOLS User Manual (Version 1.0). Technical Report RT-0191, INRIA, June 1996.

[66] Gérard Boudol, Robert de Simone, and Didier Vergamini. Experiment with AUTO and AUTOGRAPH on a Simple Case of Sliding Window Protocol. Research Report 870, INRIA, 1988.

[67] Gérard Boudol, Valérie Roy, Robert de Simone, and Didier Vergamini. Process Calculi, from Theory to Practice: Verification Tools. In Joseph Sifakis, editor, *Proceedings of the 1st International Workshop on Automatic Verification Methods*

for Finite State Systems (CAV'89), Grenoble, France, volume 407 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1989.

[68] Alexandre Boulgakov, Thomas Gibson-Robinson, and A. W. Roscoe. Computing Maximal Weak and Other Bisimulations. *Formal Aspects of Computing*, 28(3):381–407, 2016.

[69] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A Model-Checking Tool for Real-Time Systems. In Alan J. Hu and Moshe Y. Vardi, editors, *Proceedings of the 10th International Conference on Computer Aided Verification (CAV'98), Vancouver, BC, Canada*, volume 1427 of *Lecture Notes in Computer Science*, pages 546–550. Springer, June 1998.

[70] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. KRONOS: A Model-Checking Tool for Real-Time Systems (Tool Presentation for FTRTFT'98). In Anders P. Ravn and Hans Rischel, editors, *Proceedings of the 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98), Lyngby, Denmark*, volume 1486 of *Lecture Notes in Computer Science*, pages 298–302. Springer, September 1998.

[71] Marius Bozga, Jean-Claude Fernandez, Alain Kerbrat, and Laurent Mounier. Protocol Verification with the ALDEBARAN Toolset. *International Journal on Software Tools for Technology Transfer*, 1(1-2):166–184, 1997.

[72] Sébastien Briais. ABC User's Guide. 2005.

[73] Ed Brinksma and Holger Hermanns. Process Algebra and Markov Chains. In Ed Brinksma, Holger Hermanns, and Joost-Pieter Katoen, editors, *Lectures on Formal Methods and Performance Analysis, Revised Lectures of the 1st EEF/Euro Summer School on Trends in Computer Science, Berg en Dal, The Netherlands*, volume 2090 of *Lecture Notes in Computer Science*, pages 183–231. Springer, July 2000.

[74] Stephen D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A Theory of Communicating Sequential Processes. *Journal of the ACM*, 31(3):560–599, July 1984.

[75] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8), 1986.

[76] Peter Buchholz. Equivalence Relations for Stochastic Automata Networks. In W. J. Stewart, editor, *Computation with Markov Chains: Proceedings of the 2nd International Workshop on the Numerical Solution of Markov Chains*, pages 197–216. Kluwer, 1995.

[77] Peter Buchholz and Peter Kemper. A Toolbox for the Analysis of Discrete Event Dynamic Systems. In Nicolas Halbwachs and Doron A. Peled, editors, *Proceedings of the 11th International Conference on Computer Aided Verification (CAV'99), Trento, Italy*, volume 1633 of *Lecture Notes in Computer Science*, pages 483–486. Springer, July 1999.

[78] Peter Buchholz and Peter Kemper. Modular State Level Analysis of Distributed Systems Techniques and Tool Support. In Rance Cleaveland, editor, *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99), Amsterdam, The Netherlands*, volume 1579 of *Lecture Notes in Computer Science*, pages 420–434. Springer, March 1999.

[79] Peter E. Bulychev. Game-Theoretic Simulation Checking Tool. *Programming and Computer Software*, 37(4):200–209, 2011.

[80] Olav Bunte, Jan Friso Groote, Jeroen J. A. Keiren, Maurice Laveaux, Thomas Neele, Erik P. de Vink, Wieger Wesselink, Anton Wijs, and Tim A. C. Willemse. The mCRL2 Toolset for Analysing Concurrent Systems - Improvements in Expressivity and Usability. In Tomás Vojnar and Lijun Zhang, editors, *Proceedings (Part II) of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'19), Prague, Czech Republic*, volume 11428 of *Lecture Notes in Computer Science*, pages 21–39. Springer, April 2019.

[81] Alexandre V. Bystrov and I. B. Virbitskaite. Implementing Model Checking and Equivalence Checking for Time Petri Nets by the RT-MEC Tool. In Victor E. Malyshkin, editor, *Proceedings of the 5th International Conference on Parallel Computing Technologies (PaCT'99), Saint Petersburg, Russia*, volume 1662 of *Lecture Notes in Computer Science*, pages 194–199. Springer, September 1999.

[82] Francesco Calzolai, Rocco De Nicola, Michele Loreti, and Francesco Tiezzi. TAPAs: A Tool for the Analysis of Process Algebras. *Transactions on Petri Nets and Other Models of Concurrency*, 1:54–70, 2008.

[83] Paolo Camurati, Fulvio Corno, and Paolo Prinetto. An Efficient Tool for System-Level Verification of Behaviors and Temporal Properties. In *Proceedings of the European Design Automation Conference (EURO-DAC'93), Hamburg, Germany*, pages 124–129. IEEE Computer Society, September 1993.

[84] Vincenza Carchiolo and Alberto Faro. A Tool for the Automated Verification of ECCS Specifications of OSI Protocols. In P. Varaiya and A. B. Kurzhanski, editors, *Proceedings of the IIASA Conference on Discrete Event Systems: Models and Applications, Sopron, Hungary*, number 103 in Lecture Notes in Control and Information Sciences, pages 57–68. Springer, 1987.

[85] Karlis Cerans, Jens Chr. Godskesen, and Kim G. Larsen. Timed Modal Specification – Theory and Tools. Research Report RS-97-11, BRICS, 1997.

[86] Ghassan Chehaibar, Hubert Garavel, Laurent Mounier, Nadia Tawbi, and Ferruccio Zulian. Specification and Verification of the PowerScale Bus Arbitration Protocol: An Industrial Experiment with LOTOS. In Reinhard Gotzhein and Jan Bredereke, editors, *Proceedings of the IFIP Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification (FORTE/PSTV'96), Kaiserslautern, Germany*, pages 435–450. Chapman & Hall, October 1996. Full version available as INRIA Research Report RR-2958.

[87] Liang Chen, Mojtaba Ebrahimi, and Mehdi Baradaran Tahoori. Quantitative Evaluation of Register Vulnerabilities in RTL Control Paths. In Giorgio Di Natale, editor, *Proceedings of the 19th IEEE European Test Symposium (ETS'14), Paderborn, Germany*, pages 1–2. IEEE, May 2014.

[88] Taolue Chen, Bas Ploeger, Jaco van de Pol, and Tim A. C. Willemse. Equivalence Checking for Infinite Systems Using Parameterized Boolean Equation Systems. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07), Lisbon, Portugal*, volume 4703 of *Lecture Notes in Computer Science*, pages 120–135. Springer, September 2007.

[89] Yung-Pin Cheng, Yu-Ru Cheng, and Hong-Yi Wang. ARCATS: A Scalable Compositional Analysis Tool Suite. In Hisham Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC'06), Dijon, France*, pages 1852–1853. ACM, April 2006.

[90] Yung-Pin Cheng, Hong-Yi Wang, and Yu-Ru Cheng. On-the-Fly Branching Bisimulation Minimization for Compositional Analysis. In Oscar H. Ibarra and Hsu-Chun Yen, editors, *Proceedings of the 11th International Conference on Implementation and Application of Automata (CIAA'06), Taipei, Taiwan*, volume 4094 of *Lecture Notes in Computer Science*, pages 219–229. Springer, August 2006.

[91] S. C. Cheung and J. Kramer. Enhancing Compositional Reachability Analysis with Context Constraints. In *Proceedings of the 1st ACM SIGSOFT International Symposium on the Foundations of Software Engineering (Los Angeles, CA, USA)*, pages 115–125. ACM Press, December 1993.

[92] Shing-Chi Cheung, Dimitra Giannakopoulou, and Jeff Kramer. Verification of Liveness Properties Using Compositional Reachability Analysis. In Mehdi Jazayeri and Helmut Schauer, editors, *Proceedings of the 6th European Conference on Software Engineering (ESEC/FSE'97), Zurich, Switzerland*, volume 1301 of *Lecture Notes in Computer Science*, pages 227–243. Springer, September 1997.

[93] Duncan Clarke. VERSA: Verification, Execution and Rewrite System for ASCR. Technical Report MS-CIS-95-34, University of Pennsylvania, 1995.

[94] Duncan Clarke, Hanêne Ben-Abdallah, Insup Lee, Hong-Liang Xie, and Oleg Sokolsky. XVERSA: An Integrated Graphical and Textual Toolset for the Specification and Analysis of Resource-Bound Real-Time Systems. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification (CAV'96), New Brunswick, NJ, USA*, volume 1102 of *Lecture Notes in Computer Science*, pages 402–405. Springer, July 1996.

[95] Duncan Clarke and Insup Lee. VERSA: A Tool for Analyzing Resource-Bound Real-Time Systems. *Journal of Computer Software Engineering*, 3(2), 1995.

[96] Duncan Clarke, Insup Lee, and Hong-Liang Xie. VERSA: A Tool for the Specification and Analysis of Resource-Bound Real-Time Systems. Technical Report MS-CIS-93-77, University of Pennsylvania, 1993.

[97] Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis. Model Checking: Algorithmic Verification and Debugging. *Communications of the ACM*, 52(11):74–84, 2009.

[98] Rance Cleaveland. On Automatically Explaining Bisimulation Inequivalence. In Edmund M. Clarke and Robert P. Kurshan, editors, *Proceedings of the 2nd International Workshop on Computer Aided Verification (CAV'90), New Brunswick, NJ, USA*, volume 531 of *Lecture Notes in Computer Science*, pages 364–372. Springer, June 1990.

[99] Rance Cleaveland. Analysing Concurrent Systems Using the Concurrency Workbench. In Peter E. Lauer, editor, *Functional Programming, Concurrency, Simulation and Automated Reasoning: International Lecture Series 1991-1992, McMaster University, Hamilton, Ontario, Canada*, volume 693 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 1993.

[100] Rance Cleaveland, Jayesh N. Gada, Philip M. Lewis, Scott A. Smolka, Oleg Sokolsky, and Shipei Zhang. The Concurrency Factory - Practical Tools for Specification, Simulation, Verification, and Implementation of Concurrent Systems. In Guy E. Blelloch, K. Mani Chandy, and Suresh Jagannathan, editors, *Proceedings of the DIMACS Workshop on Specification of Parallel Algorithms, Princeton, New Jersey, USA*, volume 18 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 75–89. DIMACS/AMS, May 1994.

[101] Rance Cleaveland and Matthew Hennessy. Testing Equivalence as a Bisimulation Equivalence. In Joseph Sifakis, editor, *Proceedings of the 1st International Workshop on Automatic Verification Methods for Finite State Systems (CAV'89), Grenoble, France*, volume 407 of *Lecture Notes in Computer Science*, pages 11–23. Springer, June 1989.

[102] Rance Cleaveland, Philip M. Lewis, Scott A. Smolka, and Oleg Sokolsky. The Concurrency Factory: A Development Environment for Concurrent Systems. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification (CAV'96), New Brunswick, New Jersey, USA*, volume 1102 of *Lecture Notes in Computer Science*, pages 398–401. Springer, August 1996.

[103] Rance Cleaveland, Tan Li, and Steve Sims. The Concurrency Workbench of the New Century (Version 1.2) – User's Manual. State University of New York at Stony Brook, July 2000.

[104] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. A Semantics Based Verification Tool for Finite State Systems. In Ed Brinksma, Giuseppe Scollo, and Chris A. Vissers, editors, *Proceedings of the 9th International Symposium on Protocol Specification, Testing and Verification (PSTV'89), Enschede, The Netherlands*, pages 287–302. North-Holland, 1989.

[105] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. The Concurrency Workbench. In Joseph Sifakis, editor, *Proceedings of the 1st Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, France*, volume 407 of *Lecture Notes in Computer Science*, pages 24–37. Springer, June 1989.

[106] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. The Concurrency Workbench: A Semantics-Based Tool for the Verification of Concurrent Systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, 1993.

[107] Rance Cleaveland and Steve Sims. The Concurrency Workbench of North Carolina User's manual (version 1.0), 1996.

[108] Rance Cleaveland and Steve Sims. The NCSU Concurrency Workbench. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification (CAV'96), New Brunswick, NJ, USA*, volume 1102 of *Lecture Notes in Computer Science*, pages 394–397. Springer, July 1996.

[109] Rance Cleaveland and Steve Sims. Generic Tools for Verifying Concurrent Systems. In *Proceedings of the 1998 ARO/ONR/NSF/DARPA Monterey Workshop on Engineering Automation for Computer Based Systems*, pages 38–46, 1999.

[110] Rance Cleaveland and Steve Sims. Generic Tools for Verifying Concurrent Systems. *Science of Computer Programming*, 42(1):39–47, 2002.

[111] Manuel Aguilar Cornejo, Hubert Garavel, Radu Mateescu, and Noël de Palma. Specification and Verification of a Dynamic Reconfiguration Protocol for Agent-Based Applications. In Aleksander Laurentowski, Jacek Kosinski, Zofia Mossurska, and Radoslaw Ruchala, editors, *Proceedings of the 3rd IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems (DAIS'01), Krakow, Poland*, pages 229–242. Kluwer Academic Publishers, September 2001. Full version available as INRIA Research Report RR-4222.

[112] Nicolas Coste. *Vers la prédiction de performance de modèles compositionnels dans les architectures GALS*. PhD thesis, Université de Grenoble, June 2010.

[113] Nicolas Coste, Hubert Garavel, Holger Hermanns, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. Ten Years of Performance Evaluation for Concurrent Systems Using CADP. In Tiziana Margaria and Bernhard Steffen, editors, *Proceedings of the 4th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation ISoLA 2010 (Amirandes, Heraclion, Crete), Part II*, volume 6416 of *Lecture Notes in Computer Science*, pages 128–142. Springer, October 2010.

[114] Nicolas Coste, Holger Hermanns, Etienne Lantreibecq, and Wendelin Serwe. Towards Performance Prediction of Compositional Models in Industrial GALS Designs. In Ahmed Bouajjani and Oded Maler, editors, *Proceedings of the 21th International Conference on Computer Aided Verification (CAV'09), Grenoble, France*, volume 5643 of *Lecture Notes in Computer Science*, pages 204–218. Springer, July 2009.

[115] Sjoerd Cranen, Jan Friso Groote, Jeroen J. A. Keiren, Frank P. M. Stappers, Erik P. de Vink, Wieger Wesselink, and Tim A. C. Willemse. An Overview of the mCRL2 Toolset and Its Recent Advances. In Nir Piterman and Scott A. Smolka, editors, *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'13), Rome, Italy*, volume 7795 of *Lecture Notes in Computer Science*, pages 199–213. Springer, 2013.

[116] Pepijn Crouzen, Holger Hermanns, and Lijun Zhang. On the Minimisation of Acyclic Models. In Franck van Breugel and Marsha Chechik, editors, *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08), Toronto, Canada*, volume 5201 of *Lecture Notes in Computer Science*, pages 295–309. Springer, August 2008.

[117] Pepijn Crouzen and Frédéric Lang. Smart Reduction. In Dimitra Giannakopoulou and Fernando Orejas, editors, *Proceedings of Fundamental Approaches to Software Engineering (FASE'11), Saarbrücken, Germany*, volume 6603 of *Lecture Notes in Computer Science*, pages 111–126. Springer, March 2011.

[118] Jim W. Davies and Steve A. Schneider. A Brief History of Timed CSP. *Theoretical Computer Science*, 138(2):243–271, February 1995.

[119] Conrado Daws, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. The Tool KRONOS. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Proceedings of the DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, Ruttgers University, New Brunswick, NJ, USA*, volume 1066 of *Lecture Notes in Computer Science*, pages 208–219. Springer, October 1995.

[120] David de Frutos-Escrig, Jeroen J. A. Keiren, and Tim A. C. Willemse. Branching Bisimulation Games. In Elvira Albert and Ivan Lanese, editors, *Proceedings*

of the 36th IFIP WG 6.1 International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE'16), Heraklion, Crete, Greece, volume 9688 of *Lecture Notes in Computer Science*, pages 142–157. Springer, June 2016.

[121] Anthony de Jacquier, Thierry Massart, and Christian Hernalsteen. Vérification et correction d'un protocole de contrôle aérien. Technical report 363, Université Libre de Bruxelles, May 1997.

[122] Rocco De Nicola, Alessandro Fantechi, Stefania Gnesi, Salvatore Larosa, and Gioia Ristori. Verifying Hardware Components with JACK. In Paolo Camurati and Hans Eveking, editors, *Proceedings of the IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'95), Frankfurt/Main, Germany*, volume 987 of *Lecture Notes in Computer Science*, pages 246–260. Springer, October 1995.

[123] Rocco De Nicola and Matthew C. B. Hennessy. Testing Equivalences for Processes. *Theoretical Computer Science*, 34:83–133, 1984.

[124] Rocco De Nicola, Paola Inverardi, and Monica Nesi. Using the Axiomatic Presentation of Behavioural Equivalences for Manipulating CCS Specifications. In Joseph Sifakis, editor, *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, France*, volume 407 of *Lecture Notes in Computer Science*, pages 54–67. Springer, 1989.

[125] Rocco De Nicola, Paola Inverardi, and Monica Nesi. Equational Reasoning About LOTOS Specifications: A Rewriting Approach. In *Proceedings of the 6th International Workshop on Software Specification and Design*, pages 148–155, 1991.

[126] Rocco De Nicola and Frits W. Vaandrager. Action versus State based Logics for Transition Systems. In Irène Guessarian, editor, *Semantics of Systems of Concurrent Processes – Proceedings of the LITP Spring School on Theoretical Computer Science, La Roche Posay, France*, volume 469 of *Lecture Notes in Computer Science*, pages 407–419. Springer, 1990.

[127] Rocco De Nicola and Frits W. Vaandrager. Three Logics for Branching Bisimulation (Extended Abstract). In *Proceedings of the 4th Annual Symposium on Logic in Computer Science (LICS'89), Pacific Grove, California, USA*, pages 118–129. IEEE Computer Society, 1990.

[128] Rocco De Nicola and Frits W. Vaandrager. Three Logics for Branching Bisimulation. *Journal of the ACM*, 42(2):458–487, 1995.

[129] Robert de Simone and Didier Vergamini. Aboard AUTO. Technical Report 111, INRIA, 1989.

[130] Christian Dehnert, Joost-Pieter Katoen, and David Parker. SMT-Based Bisimulation Minimisation of Markov Models. In Roberto Giacobazzi, Josh Berdine, and Isabella Mastroeni, editors, *Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'13), Rome, Italy*, volume 7737 of *Lecture Notes in Computer Science*, pages 28–47. Springer, January 2013.

[131] Salem Derisavi. A Symbolic Algorithm for Optimal Markov Chain Lumping. In Orna Grumberg and Michael Huth, editors, *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07), Braga, Portugal*, volume 4424 of *Lecture Notes in Computer Science*, pages 139–154. Springer, March 2007.

[132] Salem Derisavi. Signature-based Symbolic Algorithm for Optimal Markov Chain Lumping. In *Proceedings of the 4th International Conference on the Quantitative Evaluation of Systems (QEST'07), Edinburgh, Scotland, UK*, pages 141–150. IEEE Computer Society, September 2007.

[133] Guillaume Doumenc and Eric Madelaine. Une traduction de PLOTOS en MEIJE. Research Report RR-0938, INRIA, 1988.

[134] Guillaume Doumenc, Eric Madelaine, and Robert De Simone. Proving Process Calculi Translations in ECRINS: The pureLOTOS -¿ MEIJE Example. Research Report RR-1192, INRIA, 1990.

[135] Agostino Dovier, Raffaella Gentilini, Carla Piazza, and Alberto Policriti. Rank-Based Symbolic Bisimulation (and Model Checking). *Electronic Notes in Theoretical Computer Science*, 67:166–183, 2002.

[136] Agostino Dovier, Carla Piazza, and Alberto Policriti. A Fast Bisimulation Algorithm. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01), Paris, France*, volume 2102 of *Lecture Notes in Computer Science*, pages 79–90. Springer, July 2001.

[137] Agostino Dovier, Carla Piazza, and Alberto Policriti. An Efficient Algorithm for Computing Bisimulation Equivalence. *Theoretical Computer Science*, 311(1-3):221–256, 2004.

[138] Christian Eisentraut, Holger Hermanns, and Lijun Zhang. Concurrency and Composition in a Stochastic World. In Paul Gastin and François Laroussinie, editors, *Proceedings of the 21th International Conference on Concurrency Theory CONCUR 2010 (Paris, France)*, volume 6269 of *Lecture Notes in Computer Science*, pages 21–39. Springer, August 2010.

[139] Reinhard Enders, Thomas Filkorn, and Dirk Taubner. Generating BDDs for Symbolic Model Checking in CCS. In K. G. Larsen and A. Skou, editors, *Proceedings of the 3rd Workshop on Computer-Aided Verification (CAV'91), Aalborg, Denmark*, volume 575 of *Lecture Notes in Computer Science*, pages 203–213. Springer, July 1991.

[140] Hakan Erdogmus. Verifying Semantic Relations in SPIN. In *Proceedings of the 1st SPIN Workshop (Montréal, Québec)*, 1995.

[141] Hakan Erdogmus, Robert de B. Johnston, and Charles Cleary. Formal Verification Based on Relation Checking in SPIN: A Case Study. In *Proceedings of the 1st Workshop on Formal Methods in Software Practice (San Diego, California, USA)*, 1995.

[142] Patrik Ernberg, Lars-åke Fredlund, and Bengt Jonsson. Specification and Validation of a Simple Overtaking Protocol using LOTOS. T 90006, Swedish Institute of Computer Science, Kista, Sweden, October 1990.

[143] Patrik Ernberg and Lars-Åke Fredlund. Identifying Some Bottlenecks of the Concurrency Workbench. Research Report T90/9002, SICS, 1990.

[144] Patrik Ernberg, Thomas Hovander, and Francisco Monfort. Specification and Implementation of an ISDN Telephone System Using LOTOS. In Michel Diaz and Roland Groz, editors, *Proceedings of the IFIP TC6/WG6.1 5th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'92), Perros-Guirec, France*, volume C-10 of *IFIP Transactions*, pages 171–186. North-Holland, October 1992.

[145] Klaus Estenfeld, Hans-Albert Schneider, Dirk Taubner, and Erik Tidén. Computer Aided Verification of Parallel Processes. In Andreas Pfitzmann and Eckart Raubold, editors, *Proceedings Verläßliche Informationssysteme (VIS'91), GI-Fachtagung, Darmstadt, Germany*, volume 271 of *Informatik-Fachberichte*, pages 208–226. Springer, March 1991.

[146] Jean-Claude Fernandez. *ALDEBARAN : Un système de vérification par réduction de processus communicants*. PhD thesis, Université Joseph Fourier (Grenoble), May 1988.

[147] Jean-Claude Fernandez. ALDEBARAN: A Tool for Verification of Communicating Processes. Rapport SPECTRE C14, Laboratoire de Génie Informatique – Institut IMAG, Grenoble, September 1989.

[148] Jean-Claude Fernandez. *ALDEBARAN: User's Manual*. Laboratoire de Génie Informatique – Institut IMAG, Grenoble, January 1989.

[149] Jean-Claude Fernandez. An Implementation of an Efficient Algorithm for Bisimulation Equivalence. *Science of Computer Programming*, 13(2–3):219–236, May 1990.

[150] Jean-Claude Fernandez, Hubert Garavel, Alain Kerbrat, Radu Mateescu, Laurent Mounier, and Mihaela Sighireanu. CADP (CÆSAR/ALDEBARAN Development Package): A Protocol Validation and Verification Toolbox. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification (CAV'96), New Brunswick, New Jersey, USA*, volume 1102 of *Lecture Notes in Computer Science*, pages 437–440. Springer, August 1996.

[151] Jean-Claude Fernandez, Hubert Garavel, Laurent Mounier, Anne Rasse, Carlos Rodríguez, and Joseph Sifakis. A Toolbox for the Verification of LOTOS Programs. In Lori A. Clarke, editor, *Proceedings of the 14th International Conference on Software Engineering (ICSE'14), Melbourne, Australia*, pages 246–259. ACM, May 1992.

[152] Jean-Claude Fernandez, Alain Kerbrat, and Laurent Mounier. Symbolic Equivalence Checking. In Costas Courcoubetis, editor, *Proceedings of the 5th Workshop on Computer-Aided Verification (CAV'93), Heraklion, Greece*, volume 697 of *Lecture Notes in Computer Science*, pages 85–96. Springer, June 1993.

[153] Jean-Claude Fernandez and Laurent Mounier. Verifying Bisimulations "On the Fly". In Juan Quemada, José Manas, and Enrique Vázquez, editors, *Proceedings of the 3rd International Conference on Formal Description Techniques (FORTE'90), Madrid, Spain*. North-Holland, November 1990.

[154] Jean-Claude Fernandez and Laurent Mounier. A Tool Set for Deciding Behavioral Equivalences. In *Proceedings of CONCUR'91, Amsterdam, The Netherlands*, August 1991.

[155] Jean-Claude Fernandez and Laurent Mounier. "On the Fly" Verification of Behavioural Equivalences and Preorders. In Kim G. Larsen and A. Skou, editors, *Proceedings of the 3rd Workshop on Computer-Aided Verification (CAV'91), Aalborg, Denmark*, volume 575 of *Lecture Notes in Computer Science*, pages 181–191. Springer, July 1991.

[156] G. Ferrari, G. Modoni, and P. Quaglia. Towards a Semantic-Based Verification Environment for the π-Calculus. In Alfredo De Santis, editor, *Proceedings of the 5th Italian Conference on Theoretical Computer Science*, 1995.

[157] Arnaud Février, Elie Najm, N. Prost, and F. Robles. Verifying an ATM Switch with Formal Methods, 1994.

[158] Kathi Fisler and Moshe Y. Vardi. Bisimulation Minimization in an Automata-Theoretic Verification Framework. In Ganesh Gopalakrishnan and Phillip J. Windley, editors, *Proceedings of the 2nd International Conference on Formal Methods in Computer-Aided Design (FMCAD'98), Palo Alto, California, USA*, volume 1522 of *Lecture Notes in Computer Science*, pages 115–132. Springer, November 1998.

[159] Kathi Fisler and Moshe Y. Vardi. Bisimulation and Model Checking. In Laurence Pierre and Thomas Kropf, editors, *Proceedings of the 10th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'99), Bad Herrenalb, Germany*, volume 1703 of *Lecture Notes in Computer Science*, pages 338–342. Springer, September 1999.

[160] Kathi Fisler and Moshe Y. Vardi. Bisimulation Minimization and Symbolic Model Checking. *Formal Methods in System Design*, 21(1):39–78, 2002.

[161] Riccardo Focardi and Roberto Gorrieri. The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.

[162] G Franceschinis and M Ribaudo. Symmetric and Behavioural Aggregation in a Simple Protocol Example. In *Proceedings of the 6th International Workshop on Process Algebra and Performance Modelling (PAPM'98), Nice, France*, 1998.

[163] Nicoletta De Francesco, Giuseppe Lettieri, Antonella Santone, and Gigliola Vaglini. GreASE: A Tool for Efficient "Nonequivalence" Checking. *ACM Transactions on Software Engineering Methodology*, 23(3):24:1–24:26, 2014.

[164] Lars-Åke Fredlund and Fredrik Orava. An Experiment in Formalizing and Analysing Railyard Configurations. In Z. Brezočnik and T. Kapus, editors, *Proceedings of COST 247 International Workshop on Applied Formal Methods in System Design (Maribor, Slovenia)*, pages 51–60. University of Maribor, Slovenia, June 1996.

[165] Lars-Åke Fredlund. The Timing and Probability Workbench: A Tool for Analysing Timed Processes, 1994. available from CiteSeer.

[166] Lars-Åke Fredlund and Fredrik Orava. Modelling Dynamic Communication Structures in LOTOS. In Ken R. Parker and Gordon A. Rose, editors, *Proceedings of the IFIP TC6/WG6.1 4th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'91), Sydney, Australia*, volume C-2 of *IFIP Transactions*, pages 185–200. North-Holland, November 1991.

[167] Kay Fuhrmann and Jan Hiemer. Formal Verification of Statemate-Statecharts. In Rudolf Berghammer and Yassine Lakhnech, editors, *Proceedings of the International Workshop Tool Support for System Specification, Development and Verification, Malente, Germany*, Advances in computing science, pages 92–107. Springer, June 1998.

[168] Hubert Garavel. *Compilation et vérification de programmes LOTOS*. PhD thesis, Université Joseph Fourier (Grenoble), November 1989.

[169] Hubert Garavel. An Overview of the Eucalyptus Toolbox. In Z. Brezočnik and T. Kapus, editors, *Proceedings of the COST 247 International Workshop*

on Applied Formal Methods in System Design (Maribor, Slovenia), pages 76–88. University of Maribor, Slovenia, June 1996.

[170] Hubert Garavel. OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing. In Bernhard Steffen, editor, *Proceedings of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98), Lisbon, Portugal*, volume 1384 of *Lecture Notes in Computer Science*, pages 68–84. Springer, March 1998. Full version available as INRIA Research Report RR-3352.

[171] Hubert Garavel and Holger Hermanns. On Combining Functional Verification and Performance Evaluation using CADP. In Lars-Henrik Eriksson and Peter A. Lindsay, editors, *Proceedings of the 11th International Symposium of Formal Methods Europe (FME'02), Copenhagen, Denmark*, volume 2391 of *Lecture Notes in Computer Science*, pages 410–429. Springer, July 2002. Full version available as INRIA Research Report 4492.

[172] Hubert Garavel, Mark Jorgensen, Radu Mateescu, Charles Pecheur, Mihaela Sighireanu, and Bruno Vivien. CADP'97 – Status, Applications and Perspectives. In Ignac Lovrek, editor, *Proceedings of the 2nd COST 247 International Workshop on Applied Formal Methods in System Design (Zagreb, Croatia)*, June 1997.

[173] Hubert Garavel and Frédéric Lang. SVL: a Scripting Language for Compositional Verification. In Myungchul Kim, Byoungmoon Chin, Sungwon Kang, and Danhyung Lee, editors, *Proceedings of the 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'01), Cheju Island, Korea*, pages 377–392. Kluwer Academic Publishers, August 2001. Full version available as INRIA Research Report RR-4223.

[174] Hubert Garavel, Frédéric Lang, and Radu Mateescu. An Overview of CADP 2001. *European Association for Software Science and Technology (EASST) Newsletter*, 4:13–24, August 2002. Also available as INRIA Technical Report RT-0254 (December 2001).

[175] Hubert Garavel, Frédéric Lang, and Radu Mateescu. Compositional Verification of Asynchronous Concurrent Systems Using CADP. *Acta Informatica*, 52(4):337–392, April 2015.

[176] Hubert Garavel, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes. In Werner Damm and Holger Hermanns, editors, *Proceedings of the 19th International Conference on Computer Aided Verification (CAV'07), Berlin, Germany*, volume 4590 of *Lecture Notes in Computer Science*, pages 158–163. Springer, July 2007.

[177] Hubert Garavel, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. CADP 2010: A Toolbox for the Construction and Analysis of Distributed Processes. In Parosh A. Abdulla and K. Rustan M. Leino, editors, *Proceedings of the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'11), Saarbrücken, Germany*, volume 6605 of *Lecture Notes in Computer Science*, pages 372–387. Springer, March 2011.

[178] Hubert Garavel, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. CADP 2011: A Toolbox for the Construction and Analysis of Distributed Processes. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 15(2):89–107, April 2013.

[179] Hubert Garavel, Frédéric Lang, and Laurent Mounier. Compositional Verification in Action. In Falk Howar and Jiri Barnat, editors, *Proceedings of the 23rd International Conference on Formal Methods for Industrial Critical Systems (FMICS'18), Maynooth, Ireland – Essays Dedicated to Susanne Graf at the Occasion of Her 60th Birthday*, volume 11119 of *Lecture Notes in Computer Science*, pages 189–210. Springer, September 2018.

[180] Hubert Garavel, Frédéric Lang, and Wendelin Serwe. From LOTOS to LNT. In Joost-Pieter Katoen, Rom Langerak, and Arend Rensink, editors, *ModelEd, TestEd, TrustEd – Essays Dedicated to Ed Brinksma on the Occasion of His 60th Birthday*, volume 10500 of *Lecture Notes in Computer Science*, pages 3–26. Springer, October 2017.

[181] Hubert Garavel and Laurent Mounier. Specification and Verification of Various Distributed Leader Election Algorithms for Unidirectional Ring Networks. *Science of Computer Programming*, 29(1–2):171–197, July 1997. Special issue on Industrially Relevant Applications of Formal Analysis Techniques. Full version available as INRIA Research Report RR-2986.

[182] Hubert Garavel and Joseph Sifakis. Compilation and Verification of LOTOS Specifications. In L. Logrippo, R. L. Probert, and H. Ural, editors, *Proceedings of the 10th IFIP International Symposium on Protocol Specification, Testing and Verification (PSTV'90), Ottawa, Canada*, pages 379–394. North-Holland, June 1990.

[183] François Germeau and Guy Leduc. Model-based Design and Verification of Security Protocols using LOTOS. In Hilarie Orman and Catherine Meadows, editors, *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols (Rutgers University, New Jersey, USA)*, September 1997.

[184] Rob Gerth, Ruurd Kuiper, Doron A. Peled, and Wojciech Penczek. A Partial Order Approach to Branching Time Logic Model Checking. *Information and Computation*, 150(2):132–152, 1999.

[185] Stephen Gilmore and Jane Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In Günter Haring and Gabriele Kotsis, editors, *Proceedings of the 7th International Conference on Computer Performance Evaluation, Modeling Techniques and Tools (TOOLS'94), Vienna, Austria*, volume 794 of *Lecture Notes in Computer Science*, pages 353–368. Springer, May 1994.

[186] Stephen Gilmore, Jane Hillston, and Leïla Kloul. PEPA Nets. In Mariacarla Calzarossa and Erol Gelenbe, editors, *Performance Tools and Applications to Networked Systems – Revised Tutorial Lectures from MASCOTS'03*, volume 2965 of *Lecture Notes in Computer Science*, pages 311–335. Springer, 2003.

[187] Stephen Gilmore, Jane Hillston, Leïla Kloul, and Marina Ribaudo. PEPA Nets: A Structured Performance Modelling Formalism. *Performance Evaluation*, 54(2):79–104, 2003.

[188] Stefania Gnesi, Eric Madelaine, and Gioia Ristori. *An Exercise in Protocol Verification*, pages 255–279. Kluwer, 1995.

[189] Jens Chr. Godskesen and Kim G. Larsen. User's Manual for Epsilon (Draft). available from CiteSeer.

[190] Jens Chr. Godskesen, Kim G. Larsen, and Michael Zeeberg. TAV (Tools for Automatic Verification). In Joseph Sifakis, editor, *Proceedings of the 1st International Workshop on Automatic Verification Methods for Finite State Systems (CAV'89), Grenoble, France*, June 1989. article present only in the participants proceedings, not in the LNCS 407 post-proceedings volume.

[191] Jens Chr. Godskesen, Kim Guldstrand Larsen, and Arne Skou. Automatic Verification of Real-Time Systems Using Epsilon. In Son T. Vuong and Samuel T. Chanson, editors, *Proceedings of the 14th IFIP TC6/WG6.1 International Symposium on Protocol Specification, Testing and Verification (PSTV'94), Vancouver, British Columbia, Canada*, volume 1 of *IFIP Conference Proceedings*, pages 323–330. Chapman & Hall, 1994.

[192] Susanne Graf and Bernhard Steffen. Compositional Minimization of Finite State Systems. In Edmund M. Clarke and Robert P. Kurshan, editors, *Proceedings of the 2nd Workshop on Computer-Aided Verification (CAV'90), Rutgers, New Jersey, USA*, volume 531 of *Lecture Notes in Computer Science*, pages 186–196. Springer, June 1990.

[193] Susanne Graf, Bernhard Steffen, and Gerald Lüttgen. Compositional Minimization of Finite State Systems Using Interface Specifications. *Formal Aspects of Computing*, 8(5):607–616, September 1996.

[194] Alain Griffault and Aymeric Vincent. The Mec 5 Model-Checker. In Rajeev Alur and Doron A. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification (CAV'04), Boston, MA, USA*, volume 3114 of *Lecture Notes in Computer Science*, pages 488–491. Springer, July 2004.

[195] Jan Friso Groote. The Syntax and Semantics of Timed μCRL. Technical Report SEN-R9709, CWI, Amsterdam, The Netherlands, June 1997.

[196] Jan Friso Groote, David N. Jansen, Jeroen J. A. Keiren, and Anton Wijs. An $O(m \log n)$ Algorithm for Computing Stuttering Equivalence and Branching Bisimulation. *ACM Transactions on Computational Logic*, 18(2):13:1–13:34, 2017.

[197] Jan Friso Groote, Jan Martens, and Erik P. de Vink. Bisimulation by Partitioning Is $\Omega((m+n) \log n)$. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory (CONCUR'21), Virtual Conference*, volume 203 of *LIPIcs*, pages 31:1–31:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, August 2021.

[198] Jan Friso Groote, Aad Mathijssen, Muck van Weerdenburg, and Yaroslav S. Usenko. From μCRL to mCRL2: Motivation and Outline. *Electronic Notes in Theoretical Computer Science*, 162:191–196, 2006.

[199] Jan Friso Groote and M. P. A. Sellink. Confluence for Process Verification. *Theoretical Computer Science*, 170(1–2):47–81, 1996.

[200] Jan Friso Groote and Frits Vaandrager. An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence. CS-R 9001, Centrum voor Wiskunde en Informatica, Amsterdam, January 1990.

[201] Jan Friso Groote and Frits Vaandrager. An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence. In Mike S. Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90), Warwick, England*, volume 443 of *Lecture Notes in Computer Science*, pages 626–638. Springer, July 1990.

[202] Jan Friso Groote and Frits W. Vaandrager. Structured Operational Semantics and Bisimulation as a Congruence. *Information and Computation*, 100(2):202–260, 1992.

[203] Jan Friso Groote and Jaco van de Pol. A Bounded Retransmission Protocol for Large Data Packets. In Martin Wirsing and Maurice Nivat, editors, *Proceedings of the 5th International Conference on Algebraic Methodology and Software Technology (AMAST'96), Munich, Germany*, volume 1101 of *Lecture Notes in Computer Science*, pages 536–550. Springer, July 1996.

[204] Jan Friso Groote, Jao Rivera Verduzco, and Erik P. de Vink. An Efficient Algorithm to Determine Probabilistic Bisimulation. *Algorithms*, 11(9):131, 2018.

[205] Jan Friso Groote and Anton Wijs. An $O(m \log n)$ Algorithm for Stuttering Equivalence and Branching Bisimulation. In Marsha Chechik and Jean-François Raskin, editors, *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16), Eindhoven, The Netherlands*, volume 9636 of *Lecture Notes in Computer Science*, pages 607–624. Springer, April 2016.

[206] Jan Friso Groote and Anton Wijs. An $O(m \log n)$ Algorithm for Stuttering Equivalence and Branching Bisimulation. Technical Report abs/1601.01478, arXiv Computing Research Repository, 2016.

[207] Nicolas Halbwachs. Using Auto for Esterel Program Verification. In *Synchronous Programming of Reactive Systems*, Springer International Series in Engineering and Computer Science, chapter 10, pages 149–155. Springer, 1993.

[208] Hans A. Hansson. *Time and Probability in Formal Design of Distributed Systems.* PhD thesis, University Uppsala, Sweden, 1991.

[209] Hans A. Hansson. *Time and Probability in Formal Design of Distributed Systems.* Elsevier, 1994.

[210] Vahid Hashemi, Holger Hermanns, Lei Song, K. Subramani, Andrea Turrini, and Piotr J. Wojciechowski. Compositional Bisimulation Minimization for Interval Markov Decision Processes. In Adrian-Horia Dediu, Jan Janoušek, Carlos Martín-Vide, and Bianca Truthe, editors, *Proceedings of the 10th International Conference on Language and Automata Theory and Applications (LATA'16), Prague, Czech Republic*, volume 9618 of *Lecture Notes in Computer Science*, pages 114–126. Springer, March 2016.

[211] Vahid Hashemi, Holger Hermanns, and Andrea Turrini. On the Efficiency of Deciding Probabilistic Automata Weak Bisimulation. *Electronic Communication of the European Association of Software Science and Technology*, 66, 2013.

[212] Vahid Hashemi, Andrea Turrini, Ernst Moritz Hahn, Holger Hermanns, and Khaled M. Elbassioni. Polynomial-Time Alternating Probabilistic Bisimulation for Interval MDPs. In Kim Guldstrand Larsen, Oleg Sokolsky, and Ji Wang, editors, *Proceedings of the 3rd International Symposium on Dependable Software Engineering: Theories, Tools, and Applications (SETTA'17), Changsha, China*, volume 10606 of *Lecture Notes in Computer Science*, pages 25–41. Springer, October 2017.

[213] Ji He and Kenneth J. Turner. Modelling and Verifying Synchronous Circuits in DILL. Technical Report CSM-152, Department of Computing Science and Mathematics, University of Stirling, 1999.

[214] Ji He and Kenneth J. Turner. Protocol-Inspired Hardware Testing. In Gyula Csopaki, Sarolta Dibuz, and Katalin Tarnay, editors, *Proceedings of the IFIP 12th International Workshop on Testing of Communicating Systems (IWTCS'99), Budapest, Hungary*, pages 131–147. Kluwer Academic, September 1999.

[215] Ji He and Kenneth J. Turner. Specification and Verification of Synchronous Hardware using LOTOS. In Jianping Wu, Samuel T. Chanson, and Qiang Gao, editors, *Proceedings of the IFIP Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing, and Verification (FORTE/PSTV'99), Beijing, China*, pages 295–312. Kluwer Academic Publishers, October 1999.

[216] Maritta Heisel and Nicole Lévy. Using LOTOS Patterns to Characterize Architectural Styles. In Michel Bidoit and Max Dauchet, editors, *Proceedings of the 7th International Conference on Theory and Practice of Software Development (TAPSOFT'97), Lille, France*, volume 1214 of *Lecture Notes in Computer Science*, pages 818–832. Springer, April 1997.

[217] Vesa Hellgren. Performance Evaluation of Four Verification Tools: ALDEBARAN, BIDMIN, Concurrency Workbench and HOGGAR. Technical report, University of Helsinki, Department of Computer Science, August 1995. 4 pages.

[218] Vesa Hellgren. User's Manual: BIDMIN Version 1.2. Technical report, University of Helsinki, Department of Computer Science, August 1995. 10 pages.

[219] Matthew Hennessy and Huimin Lin. Symbolic Bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

[220] Monika Rauch Henzinger, Thomas A. Henzinger, and Peter W. Kopke. Computing Simulations on Finite and Infinite Graphs. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 453–462. IEEE Computer Society, 1995.

[221] Thomas A. Henzinger. Hybrid Automata with Finite Bisimulations. In Zoltán Fülöp and Ferenc Gécseg, editors, *Proceedings of the 22nd International Colloquium on Automata, Languages and Programming (ICALP'95), Szeged, Hungary*, volume 944 of *Lecture Notes in Computer Science*, pages 324–335. Springer, July 1995.

[222] Marc Herbert. Evaluation de performances et spécification formelle sur un réseau de stations haut débit. Master's thesis, Institut National des Télécommunications, Laboratoire pour les hautes performances en calcul (Lyon, France), December 1997.

[223] Holger Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *Lecture Notes in Computer Science*. Springer, 2002.

[224] Holger Hermanns, Ulrich Herzog, Ulrich Klehmet, Vassilis Mertsiotakis, and Markus Siegle. Compositional Performance Modelling with the TIPPtool. In Ramón Puigjaner, Nunzio N. Savino, and Bartomeu Serra, editors, *Proceedings of the 10th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools (TOOLS'98), Palma de Mallorca, Spain*, volume 1469 of *Lecture Notes in Computer Science*, pages 51–62. Springer, September 1998.

[225] Holger Hermanns, Ulrich Herzog, Ulrich Klehmet, Vassilis Mertsiotakis, and Markus Siegle. Compositional Performance Modelling with the TIPPtool. *Performance Evaluation*, 39(1-4):5–35, 2000.

[226] Holger Hermanns, Ulrich Herzog, and Vassilis Mertsiotakis. Stochastic Process Algebras as a Tool for Performance and Dependability Modelling. In *Proceedings of the 1995 International Computer Performance and Dependability Symposium*, pages 102–111. IEEE, 1995.

[227] Holger Hermanns, Ulrich Herzog, and Vassilis Mertsiotakis. Stochastic Process Algebras – Between LOTOS and Markov Chains. *Computer Networks*, 30(9-10):901–924, 1998.

[228] Holger Hermanns and Christophe Joubert. A Set of Performance and Dependability Analysis Components for CADP. In Hubert Garavel and John Hatcliff, editors, *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03), Warsaw, Poland*, volume 2619 of *Lecture Notes in Computer Science*, pages 425–430. Springer, April 2003.

[229] Holger Hermanns and Joost-Pieter Katoen. Automated Compositional Markov Chain Generation for a Plain-Old Telephone System. *Science of Computer Programming*, 36:97–127, 2000.

[230] Holger Hermanns and Joost-Pieter Katoen. The How and Why of Interactive Markov Chains. In Frank S. de Boer, Marcello M. Bonsangue, Stefan Hallerstede, and Michael Leuschel, editors, *Revised Selected Papers of the 8th International Symposium on Formal Methods for Components and Objects (FMCO'09), Eindhoven, The Netherlands*, volume 6286 of *Lecture Notes in Computer Science*, pages 311–337. Springer, November 2009.

[231] Holger Hermanns, Vassilis Mertsiotakis, and Michael Rettelbach. Performance Analysis of Distributed Systems Using TIPP — A Case Study. In *Proceedings of the 10th U.K. Performance Engineering Workshop for Computer and Telecommunication Systems, Edinburgh, Scotland, United Kingdom*. Edinburgh University Press, September 1994.

[232] Holger Hermanns, Vassilis Mertsiotakis, and Markus Siegle. TIPPtool: Compositional Specification and Analysis of Markovian Performance Models. In Nicolas Halbwachs and Doron A. Peled, editors, *Proceedings of the 11th International Conference on Computer Aided Verification (CAV'99), Trento, Italy*, volume 1633 of *Lecture Notes in Computer Science*, pages 487–490. Springer, July 1999.

[233] Holger Hermanns and Michael Rettelbach. Syntax, Semantics, Equivalences, and Axioms for MTIPP. In Ulrich Herzog and Michael Rettelbach, editors, *Proceedings of the 2nd Workshop on Process Algebras and Performance Modelling (PAPM'94), Erlangen, Germany*, volume 1601 of *Lecture Notes in Computer Science*, pages 71–88. University of Erlangen-Nürnberg, Germany, July 1994.

[234] Holger Hermanns, Michael Rettelbach, and Thorsten Weiss. Formal Characterisation of Immediate Actions in SPA with Nondeterministic Branching. *Computer Journal*, 38(7):530–541, January 1995.

[235] Holger Hermanns and Markus Siegle. Bisimulation Algorithms for Stochastic Process Algebras and their BDD-based Implementation. In Joost-Pieter Katoen, editor, *Proceedings of the 5th International AMAST Workshop (ARTS'99), Bamberg, Germany*, volume 1601 of *Lecture Notes in Computer Science*, pages 244–265. Springer, May 1999.

[236] Holger Hermanns and Markus Siegle. Symbolic Minimisation of Stochastic Process Algebra Models. In Katharina Spies and Bernhard Schätz, editors, *Formale Beschreibungstechniken für verteilte Systeme, 9. GI/ITG-Fachgespräch, München, Juni 1999*, pages 73–82. Herbert Utz Verlag, 1999.

[237] Holger Hermanns and Andrea Turrini. Deciding Probabilistic Automata Weak Bisimulation in Polynomial Time. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'12), Hyderabad, India*, volume 18 of *LIPIcs*, pages 435–447. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, December 2012.

[238] Christian Hernalsteen. A Timed Automaton Model for ET-LOTOS Verification. In Atsushi Togashi, Tadanori Mizuno, Norio Shiratori, and Teruo Higashino, editors, *Proceedings of the IFIP TC6/WG6.1 Joint 10th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols and 17th International Workshop on Protocol Specification, Testing and Verification (FORTE/PSTV'97), Osaka, Japan*, volume 107 of *IFIP Conference Proceedings*, pages 193–204. Chapman & Hall, November 1997.

[239] Ulrich Herzog and Vassilis Mertsiotakis. Stochastic Process Algebras Applied to Failure Modelling. In Ulrich Herzog and Michael Rettelbach, editors, *Proceedings of the 2nd Workshop on Process Algebras and Performance Modelling, Regensberg, Germany*. Arbeitsberichte des IMMD, Universität Erlangen-Nürnberg, Germany, July 1994.

[240] Michael Hillerström. Verification of CCS-processes. Master's thesis, Computer Science Department, Aalborg University, 1987.

[241] Jane Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, University of Edinburgh, December 1994.

[242] Jane Hillston, Hermanns, Ulrich Herzog, Vassilis Mertsiotakis, and Michael Rettelbach. Stochastic Process Algebras: Integrating Qualitative and Quantitative Modelling. Technical report, IMMD VII, University of Erlangen-Nürnberg, Germany, 1994.

[243] Chih-Duo Hong, Anthony W. Lin, Rupak Majumdar, and Philipp Rümmer. Probabilistic Bisimulation for Parameterized Systems (with Applications to Verifying Anonymous Protocols). In Isil Dillig and Serdar Tasiran, editors, *Proceedings (Part I) of the 31st International Conference on Computer Aided Verification (CAV'19), New York City, NY, USA*, volume 11561 of *Lecture Notes in Computer Science*, pages 455–474. Springer, July 2019.

[244] Michaela Huhn, Peter Niebert, and Heike Wehrheim. Partial Order Reductions for Bisimulation Checking. In Vikraman Arvind and Ramaswamy Ramanujam, editors, *Proceedings of the 18th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'98), Chennai, India*, volume 1530 of *Lecture Notes in Computer Science*, pages 271–282. Springer, December 1998.

[245] Michael Huth and Marta Kwiatkowska. On Probabilistic Model Checking, 1996. available from CiteSeer.

[246] Richard Huybers. *A Parallel Relation-Based Algorithm for Symbolic Bisimulation Minimization*. PhD thesis, Leiden University, 2018.

[247] Richard Huybers and Alfons Laarman. A Parallel Relation-Based Algorithm for Symbolic Bisimulation Minimization. In Constantin Enea and Ruzica Piskac, editors, *Proceedings of the 20th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'19), Cascais, Portugal*, volume 11388 of *Lecture Notes in Computer Science*, pages 535–554. Springer, January 2019.

[248] Anna Ingólfsdóttir and Huimin Lin. A Symbolic Approach to Value-Passing Processes. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra*, pages 427–478. North-Holland / Elsevier, 2001.

[249] Paola Inverardi, C Priami, and Daniel Yankelevich. Verification of Concurrent Systems in SML. In *Proceedings of ACM SIGPLAN Workshop on ML and its Applications*, pages 169–174, 1992.

[250] Paola Inverardi and Corrado Priami. Evaluation of Tools for the Analysis of Communicating Systems. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 45:158–185, 1991.

[251] Paola Inverardi and Corrado Priami. Automatic Verification of Distributed Systems: The Process Algebra Approach. *Formal Methods in System Design*, 8(1):7–38, 1996.

[252] Paola Inverardi, Corrado Priami, and Daniel Yankelevich. Automatizing Parametric Reasoning on Distributed Concurrent Systems. *Formal Aspects of Computing*, 6(6):676–695, 1994.

[253] ISO/IEC. ESTELLE – A Formal Description Technique Based on an Extended State Transition Model. International Standard 9074, International Organization for Standardization – Information Processing Systems – Open Systems Interconnection, Geneva, September 1988.

[254] ISO/IEC. LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. International Standard 8807, International Organization for Standardization – Information Processing Systems – Open Systems Interconnection, Geneva, September 1989.

[255] David N. Jansen, Jan Friso Groote, Jeroen J. A. Keiren, and Anton Wijs. A Simpler $O(m \log n)$ Algorithm for Branching Bisimilarity on Labelled Transition Systems. Technical Report abs/1909.10824, arXiv Computing Research Repository, 2019.

[256] David N. Jansen, Jan Friso Groote, Jeroen J. A. Keiren, and Anton Wijs. An $O(m \log n)$ Algorithm for Branching Bisimilarity on Labelled Transition Systems. In Armin Biere and David Parker, editors, *Proceedings of the 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'20), Dublin, Ireland*, volume 12079 of *Lecture Notes in Computer Science*, pages 3–20. Springer, April 2020.

[257] David N. Jansen, Jan Friso Groote, Ferry Timmers, and Pengfei Yang. A Near-Linear-Time Algorithm for Weak Bisimilarity on Markov Chains. In Igor Konnov and Laura Kovács, editors, *Proceedings of the 31st International Conference on Concurrency Theory (CONCUR'20), Vienna, Austria*, volume 171 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, September 2020.

[258] Claude Jard, Thierry Jéron, Jean-Claude Fernandez, and Laurent Mounier. On-the-Fly Verification of Finite Transition Systems. Research Report 1861, INRIA, 1993.

[259] Claus Torp Jensen. The Concurrency Workbench with Priorities. In Kim Guldstrand Larsen and Arne Skou, editors, *Proceedings of the 3rd International Workshop on Computer Aided Verification (CAV'91), Aalborg, Denmark*, volume 575 of *Lecture Notes in Computer Science*, pages 147–157. Springer, July 1991.

[260] Magnus Johansson, Björn Victor, and Joachim Parrow. Computing Strong and Weak Bisimulations for Psi-Calculi. *Journal of Logical and Algebraic Methods in Programming*, 81(3):162–180, 2012.

[261] P. C. Kanellakis and S. A. Smolka. CCS Expressions, Finite State Processes, and Three Problems of Equivalence. *Information and Computation*, 86(1):43–68, May 1990.

[262] Paris C. Kanellakis and Scott A. Smolka. CCS Expressions, Finite State Processes, and Three Problems of Equivalence. In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *Proceedings of the 2nd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada*, pages 228–240. ACM, August 1983.

[263] Inhye Kang, Insup Lee, and Young Si Kim. A State Minimization Technique for Timed Automata. In *Proceedings of the International Workshop on Verification of Infinite State Systems (INFINITY'96), Pisa, Italy*, pages 115–126, August 1996.

[264] Gijs Kant, Alfons Laarman, Jeroen Meijer, Jaco van de Pol, Stefan Blom, and Tom van Dijk. LTSmin: High-Performance Language-Independent Model Checking. In Christel Baier and Cesare Tinelli, editors, *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'15), London, UK*, volume 9035 of *Lecture Notes in Computer Science*, pages 692–707. Springer, April 2015.

[265] Joost-Pieter Katoen, Tim Kemna, Ivan S. Zapreev, and David N. Jansen. Bisimulation Minimisation Mostly Speeds Up Probabilistic Model Checking. In Orna Grumberg and Michael Huth, editors, *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07), Braga, Portugal*, volume 4424 of *Lecture Notes in Computer Science*, pages 87–101. Springer, March 2007.

[266] Joost-Pieter Katoen, Ivan S. Zapreev, Ernst Moritz Hahn, Holger Hermanns, and David N. Jansen. The Ins and Outs of the Probabilistic Model Checker MRMC. *Performance Evaluation*, 68(2):90–104, 2011.

[267] Jeroen Keiren and Tim A. C. Willemse. Bisimulation Minimisations for Boolean Equation Systems. In Kedar S. Namjoshi, Andreas Zeller, and Avi Ziv, editors, *Hardware and Software: Verification and Testing — Revised Selected Papers of the 5th International Haifa Verification Conference (HVC'09), Haifa, Israel*, volume 6405 of *Lecture Notes in Computer Science*, pages 102–116. Springer, October 2009.

[268] Robert M. Keller. Formal Verification of Parallel Programs. *Communications of the ACM*, 19(7):371–384, 1976.

[269] John G. Kemeny and J. Laurie Snell. *Finite Markov Chains*. Undergraduate Texts in Mathematic. Springer, 1976.

[270] Alain Kerbrat. *Méthodes symboliques pour la vérification de processus communicants: Etude et mise en œuvre.* PhD thesis, Université Joseph Fourier (Grenoble), November 1994.

[271] Alain Kerbrat. Reachable State Space Analysis of LOTOS Programs. In Dieter Hogrefe and Stefan Leue, editors, *Proceedings of the 7th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols FORTE'94 (Bern, Switzerland)*, October 1994.

[272] Alain Kerbrat and Slim Ben Atallah. Formal Specification of a Framework for Groupware Development. In Gregor v. Bochmann, Rachida Dssouli, and Omar Rafiq, editors, *Proceedings of the 8th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'95), Montreal, Quebec, Canada*, IFIP Advances in Information and Communication Technology, pages 303–310. Springer, October 1995.

[273] Alain Kerbrat, Carlos Rodriguez, and Yves Lejeune. Interconnecting the ObjectGEODE and CÆSAR/ALDEBARAN Toolsets. In Ana Cavalli and Amardeo Sarma, editors, *Proceedings of the 8th SDL Forum (Evry, France)*, September 1997.

[274] Antti Kervinen, Antti Valmari, and Risto Järnström. Debugging a Real-Life Protocol with CFFD-Based Verification Tools. In Stefania Gnesi and Ulrich Ultes-Nitsche, editors, *Proceedings of the 6th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'01), Paris, France*, pages 13–27. Université Paris 7 – LIAFA and INRIA Rhône-Alpes, July 2001.

[275] Ulrich Klehmet and Vassilis Mertsiotakis. TIPPtool: Timed Processes and Performability Evaluation (User's Guide–Version 2.3), 1998. available from CiteSeer.

[276] Barbara König, Christina Mika-Michalski, and Lutz Schröder. User Manual T-Beg: A Tool for Behavioural Equivalence Games. 2002.

[277] Henri Korver. Detecting Feature Interactions with Cæsar/Aldebaran. *Science of Computer Programming*, 29(1–2):259–278, July 1997. Special issue on Industrially Relevant Applications of Formal Analysis Techniques.

[278] Henri P. Korver. The Current State of Bisimulation Tools. P 9101, Centrum voor Wiskunde en Informatica, Amsterdam, January 1991.

[279] Bernd J. Krämer, Norbert Völker, Reiner Lichtenecker, and Hans-Friedrich Kötter. Deriving CORBA Applications from Formal Specifications. *Journal of Systems Integration*, 8(2):143–158, 1998.

[280] Jean-Pierre Krimm and Laurent Mounier. Compositional State Space Generation from LOTOS Programs. In Ed Brinksma, editor, *Proceedings of the 3rd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'97), University of Twente, Enschede, The Netherlands*, volume 1217 of *Lecture Notes in Computer Science*, pages 239–258. Springer, April 1997. Extended version with proofs available as Research Report VERIMAG RR97-01.

[281] Saul Kripke. Semantical Considerations on Modal Logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[282] Padmanabhan Krishnan. A Case Study in Specifying and Testing Architectural Features. *Microprocessors and Microsystems*, 18(3):123–130, 1994.

[283] Carsten H. Kristensen, Jøgen H. Andersen, and Arne Skou. Specification and Automated Verification of Real-Time Behaviour: A Case Study. *Annual Reviews in Control*, 20:55–70, 1996.

[284] Carsten H. Kristensen, Jørgen H. Andersen, and Arne Skou. Specification and Automated Verification of Real-Time Behaviour: A Case Study. In *Proceedings of the 3rd IFAC/IFIP Workshop on Algorithms and Architectures for Real-Time Control (AARTC'95), Ostend, Belgium*, 1995.

[285] Konrad Kulakowski. Concurrent Bisimulation Algorithm. Technical Report abs/1311.7635, arXiv Computing Research Repository, 2013.

[286] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Symmetry Reduction for Probabilistic Model Checking. In Thomas Ball and Robert B. Jones, editors, *Proceedings of the 18th International Conference on Computer Aided Verification (CAV'06), Seattle, WA, USA*, volume 4144 of *Lecture Notes in Computer Science*, pages 234–248. Springer, August 2006.

[287] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11), Snowbird, UT, USA*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, July 2011.

[288] Frédéric Lang. Compositional Verification using SVL Scripts. In Joost-Pieter Katoen and Perdita Stevens, editors, *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02), Grenoble, France*, volume 2280 of *Lecture Notes in Computer Science*, pages 465–469. Springer, April 2002.

[289] Frédéric Lang. EXP.OPEN 2.0: A Flexible Tool Integrating Partial Order, Compositional, and On-the-fly Verification Methods. In Judi Romijn, Graeme Smith, and Jaco van de Pol, editors, *Proceedings of the 5th International Conference on Integrated Formal Methods (IFM'05), Eindhoven, The Netherlands*, volume 3771 of *Lecture Notes in Computer Science*, pages 70–88. Springer, November 2005. Full version available as INRIA Research Report RR-5673.

[290] Frédéric Lang. Refined Interfaces for Compositional Verification. In Elie Najm, Jean-François Pradat-Peyre, and Véronique Viguié Donzeau-Gouge, editors, *Proceedings of the 26th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'06), Paris, France*, volume 4229 of *Lecture Notes in Computer Science*, pages 159–174. Springer, September 2006. Full version available as INRIA Research Report RR-5996.

[291] Frédéric Lang and Radu Mateescu. Partial Order Reductions Using Compositional Confluence Detection. In Ana Cavalcanti and Dennis Dams, editors, *Proceedings of the 16th International Symposium on Formal Methods (FM'09), Eindhoven, The Netherlands*, volume 5850 of *Lecture Notes in Computer Science*, pages 157–172. Springer, November 2009.

[292] Frédéric Lang and Radu Mateescu. Partial Model Checking Using Networks of Labelled Transition Systems and Boolean Equation Systems. *Logical Methods in Computer Science*, 9(4):1–32, October 2013.

[293] Frédéric Lang, Radu Mateescu, and Franco Mazzanti. Compositional Verification of Concurrent Systems by Combining Bisimulations. In Maurice ter Beek, Annabelle McIver, and José N. Oliveira, editors, *Proceedings of the 23rd International Symposium on Formal Methods — 3rd World Congress on Formal Methods (FM'19), Porto, Portugal*, volume 11800 of *Lecture Notes in Computer Science*, pages 196–213. Springer, 2019.

[294] Frédéric Lang, Radu Mateescu, and Franco Mazzanti. Sharp Congruences Adequate with Temporal Logics Combining Weak and Strong Modalities. In Armin Biere and David Parker, editors, *Proceedings of the 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'20), Dublin, Ireland*, volume 12079 of *Lecture Notes in Computer Science*, pages 57–76. Springer, April 2020.

[295] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. Model-Checking for Real-Time Systems. In Horst Reichel, editor, *Proceedings of the 10th International Symposium on Fundamentals of Computation Theory (FCT'95), Dresden, Germany*, volume 965 of *Lecture Notes in Computer Science*, pages 62–88. Springer, August 1995.

[296] Kim Guldstrand Larsen and Arne Skou. Bisimulation Through Probabilistic Testing. In *Conference Record of the 16th Annual ACM Symposium on Principles of Programming Languages (POPL'89), Austin, Texas, USA*, pages 344–352. ACM Press, January 1989.

[297] Kim Guldstrand Larsen and Arne Skou. Bisimulation Through Probabilistic Testing. *Information and Computation*, 94(1):1–28, 1991.

[298] Guy Leduc. Verification of Two Versions of the Challenge Handshake Authentication Protocol (CHAP). *Annals of Telecommunications*, 55(1–2):18–30, January–February 2000.

[299] Guy Leduc, Olivier Bonaventure, Eckhart Koerner, Luc Léonard, Charles Pecheur, and D. Zanetti. Specification and Verification of a TTP Protocol for the Conditional Access to Services. In *Proceedings of the 12th Jacques Cartier Workshop on "Formal Methods and their Applications: Telecommunications, VLSI and Real-Time Computerized Control System", Montréal, Canada*, October 1996.

[300] David Lee and Mihalis Yannakakis. Online Minimization of Transition Systems (Extended Abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada*, pages 264–274. ACM, May 1992.

[301] Luc Léonard and Guy Leduc. An Introduction to ET-LOTOS for the Description of Time-Sensitive Systems. *Computer Networks and ISDN Systems*, 29(3):271–292, 1997.

[302] Luc Léonard and Guy Leduc. A Formal Definition of Time in LOTOS. *Formal Aspects of Computing*, 10(3):248–266, 1998.

[303] Zhoujun Li and Huowang Chen. Computing Strong/Weak Bisimulation Equivalences and Observation Congruence for Value-Passing Processes. In Rance Cleaveland, editor, *Proceedings of the 5th International Conference Tools and Algorithms for Construction and Analysis of Systems, (TACAS'99), Amsterdam, The Netherlands*, volume 1579 of *Lecture Notes in Computer Science*, pages 300–314. Springer, March 1999.

[304] Zhoujun Li, Huowang Chen, and Bingshan Wang. Symbolic Transition Graph and its Early Bisimulation Checking Algorithms for the π-Calculus. *Science in China Series E: Technological Sciences*, 42(4):342–353, 1999.

[305] Reiner Lichtenecker, Klaus Gotthardt, and Janusz Zalewski. Automated Verification of Communication Protocols Using CCS and BDDs. In José D. P. Rolim,

editor, *Proceedings of the 12th International Parallel Processing Symposium and the 9th Symposium on Parallel and Distributed Processing (IPPS/SPDP'98), Orlando, Florida, USA*, volume 1388 of *Lecture Notes in Computer Science*, pages 1057–1066. Springer, March–April 1998.

[306] Huimin Lin. A Verification Tool for Value-Passing Processes. In André A. S. Danthine, Guy Leduc, and Pierre Wolper, editors, *Proceedings of the IFIP TC6/WG6.1 13th International Symposium on Protocol Specification, Testing and Verification (PSTV'93), Liège, Belgium*, volume C-16 of *IFIP Transactions*, pages 79–92. North-Holland, May 1993.

[307] Huimin Lin. Symbolic Transition Graph with Assignment. In Ugo Montanari and Vladimiro Sassone, editors, *Proceedings of the 7th International Conference on Concurrency Theory (CONCUR'96), Pisa, Italy*, volume 1119 of *Lecture Notes in Computer Science*, pages 50–65. Springer, August 1996.

[308] Huimin Lin. Computing Bisimulations for Finite-Control pi-Calculus. *Journal of Computer Science and Technology*, 15(1):1–9, 2000.

[309] Luigi Logrippo, Laurent Andriantsiferana, and Brahim Ghribi. Prototyping and Formal Requirement Validation of GPRS: A Mobile Data Packet Radio Service for GSM. In Charles B. Weinstock and John Rushby, editors, *Proceedings of the 7th IFIP International Working Conference on Dependable Computing for Critical Applications (DCCA-7), San Jose, CA, USA*, January 1999.

[310] Natalia López and Manuel Núñez. An Overview of Probabilistic Process Algebras and their Equivalences. In Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, Joost-Pieter Katoen, and Markus Siegle, editors, *Validation of Stochastic Systems – A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 89–123. Springer, 2004.

[311] Gerald Lüttgen and Walter Vogler. Bisimulation on Speed: Worst-Case Efficiency. *Information and Computation*, 191(2):105–144, 2004.

[312] Matti Luukkainen and Ari Ahtiainen. Compositional Verification of Large SDL Systems. In *Proceedings of the 1st Workshop of the SDL Forum Society on SDL and MSC (SAM'98), Berlin, Germany*, June 1998.

[313] Eric Madelaine. Verification Tools from the CONCUR Project. *EATCS Bulletin*, 47:110–120, 1992.

[314] Eric Madelaine and Robert de Simone. ECRINS: un laboratoire de preuve pour les calculs de processus. Rapport de recherche 672, INRIA, May 1987.

[315] Eric Madelaine and Didier Vergamini. AUTO: A Verification Tool for Distributed Systems Using Reduction of Finite Automata Networks. In Son T. Vuong, editor, *Proceedings of the 2nd IFIP International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'89), Vancouver, BC, Canada*, pages 61–66. North-Holland, December 1989.

[316] Eric Madelaine and Didier Vergamini. Specification and Verification of a Sliding Window Protocol in LOTOS. In Ken R. Parker and Gordon A. Rose, editors, *Proceedings of the 4th IFIP TC6/WG6.1 International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'91), Sydney, Australia*, volume C-2 of *IFIP Transactions*, pages 495–510. North-Holland, November 1991.

[317] Eric Madelaine and Didier Vergamini. Tool Demonstration: Tools for Process Algebras. In Ken R. Parker and Gordon A. Rose, editors, *Proceedings of the 4th IFIP TC6/WG6.1 International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'91), Sydney, Australia*, volume C-2 of *IFIP Transactions*, pages 463–466. North-Holland, November 1991.

[318] Eric Madelaine and Didier Vergamini. Verification of Communicating Processes by Means of Automata Reduction and Abstraction. In Alain Finkel and Matthias Jantzen, editors, *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science (STACS'92), Cachan, France*, volume 577 of *Lecture Notes in Computer Science*, pages 613–614. Springer, February 1992.

[319] Jeff Magee and Jeff Kramer. *Concurrency: State Models and Java Programs.* Wiley, 1999.

[320] Jawahar Malhotra, Scott A. Smolka, Alessandro Giacalone, and Robert M. Shapiro. A Tool for Hierarchical Design and Simulation of Concurrent Systems. In *Proceedings of the BCS-FACS Workshop on Specification and Verification of Concurrent Systems, Stirling, Scotland, UK*, pages 140–152. British Computer Society, July 1988.

[321] Panos Markopoulos, Jon Rowson, and Peter Johnson. Dialogue Modelling in the Framework of an Interactor Model. In F. Bodart and J. Vanderdonckt, editors, *Proceedings of the 3rd International Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS '96 (Namur, Belgium)*. University of Namur, June 1996.

[322] Jan Martens, Jan Friso Groote, Lars B. van den Haak, Pieter Hijma, and Anton Wijs. A Linear Parallel Algorithm to Compute Bisimulation and Relational Coarsest Partitions. In Gwen Salaün and Anton Wijs, editors, *Proceedings of the 17th International Conference on Formal Aspects of Component Software (FACS'21), Virtual Event*, volume 13077 of *Lecture Notes in Computer Science*, pages 115–133. Springer, October 2021.

[323] Radu Mateescu. Formal Description and Analysis of a Bounded Retransmission Protocol. In Z. Brezočnik and T. Kapus, editors, *Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design (Maribor, Slovenia)*, pages 98–113. University of Maribor, Slovenia, June 1996. Also available as INRIA Research Report RR-2965.

[324] Radu Mateescu. Vérification de systèmes répartis : l'exemple du protocole BRP. *Technique et Science Informatiques*, 16(6):725–751, June 1997.

[325] Radu Mateescu. Local Model-Checking of an Alternation-Free Value-Based Modal Mu-Calculus. In Annalisa Bossi, Agostino Cortesi, and Francesca Levi, editors, *Proceedings of the 2nd International Workshop on Verification, Model Checking and Abstract Interpretation (VMCAI'98), Pisa, Italy.* University Ca' Foscari of Venice, September 1998.

[326] Radu Mateescu. *Vérification des propriétés temporelles des programmes parallèles.* PhD thesis, Institut National Polytechnique de Grenoble, April 1998.

[327] Radu Mateescu. On-the-fly State Space Reductions for Weak Equivalences. In Tiziana Margaria and Mieke Massink, editors, *Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'05),*

Lisbon, Portugal, pages 80–89. ERCIM, ACM Computer Society Press, September 2005.

[328] Radu Mateescu. CAESAR_SOLVE: A Generic Library for On-the-Fly Resolution of Alternation-Free Boolean Equation Systems. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 8(1):37–56, February 2006. Full version available as INRIA Research Report RR-5948, July 2006.

[329] Radu Mateescu. Specification and Analysis of Asynchronous Systems using CADP. In Stephan Merz and Nicolas Navet, editors, *Modeling and Verification of Real-Time Systems – Formalisms and Software Tools*, chapter 5, pages 141–170. ISTE publishing / John Wiley, 2008.

[330] Radu Mateescu and Emilie Oudot. Bisimulator 2.0: An On-the-Fly Equivalence Checker based on Boolean Equation Systems. In *Proceedings of the 6th ACM-IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'08), Anaheim, CA, USA*, pages 73–74. IEEE Computer Society Press, June 2008.

[331] Radu Mateescu and Emilie Oudot. Improved On-the-Fly Equivalence Checking using Boolean Equation Systems. In Klaus Havelund, Rupak Majumdar, and Jens Palsberg, editors, *Proceedings of the 15th International SPIN Workshop on Model Checking of Software (SPIN'08), Los Angeles, USA*, volume 5156 of *Lecture Notes in Computer Science*, pages 196–213. Springer, August 2008. Full version available as INRIA Research Report RR-6777.

[332] Radu Mateescu, Pascal Poizat, and Gwen Salaün. Adaptation of Service Protocols Using Process Algebra and On-the-Fly Reduction Techniques. In Athman Bouguettaya, Ingolf Krueger, and Tiziana Margaria, editors, *Proceedings of the 6th International Conference on Service Oriented Computing (ICSOC'08), Sydney, Australia*, volume 5364 of *Lecture Notes in Computer Science*, pages 84–99. Springer, December 2008.

[333] Radu Mateescu and Anton Wijs. Property-Dependent Reductions Adequate with Divergence-Sensitive Branching Bisimilarity. *Science of Computer Programming*, 96(3):354–376, 2014.

[334] Mihir Mehta and Shibashis Guha. ReLTS 1.0 User Manual. 2014.

[335] Mihir Mehta, Shibashis Guha, and S. Arun-Kumar. ReLTS: A Tool for Checking Generalized Behavioural Relations over LTSs. 2014.

[336] George J. Milne. CIRCAL and the Representation of Communication, Concurrency, and Time. *ACM Transactions on Programming Languages and Systems*, 7(2):270–298, April 1985.

[337] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes I. *Information and Computation*, 100(1):1–40, September 1992.

[338] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes II. *Information and Computation*, 100(1):41–77, September 1992.

[339] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[340] Robin Milner. Calculi for Synchrony and Asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[341] Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[342] Faron Moller. The Edinburgh Concurrency Workbench (Version 6.1). User manual, Laboratory for the Foundations of Computer Science, University of Edinburgh, 1992.

[343] Faron Moller, Scott A. Smolka, and Jirí Srba. On the Computational Complexity of Bisimulation, Redux. *Information and Computation*, 194(2):129–143, 2004.

[344] Amelia Soriano Montes. *VENUS: un outil d'aide à la vérification des systèmes communicants*. PhD thesis, Institut National Polytechnique de Grenoble, January 1987.

[345] Laurent Mounier. *Méthodes de vérification de spécifications comportementales : étude et mise en œuvre*. PhD thesis, Université Joseph Fourier (Grenoble), January 1992.

[346] Malcolm Mumme and Gianfranco Ciardo. An Efficient Fully Symbolic Bisimulation Algorithm for Non-Deterministic Systems. *International Journal of Foundations of Computer Science*, 24(2):263–282, 2013.

[347] Elie Najm, Stanislaw Budkowski, Thierry Gilot, and Leon Lumbroso. General Presentation of SCAN – A Distributed Systems Modelling and Validation Tool. In Michel Diaz, editor, *Proceedings of the 5th IFIP International Workshop on Protocol Specification, Testing, and Verification (PSTV'85), Moissac, France*, pages 103–118. North-Holland, June 1985.

[348] Martín Llamas Nistal, Juan Quemada, and Manuel J. Fernández Iglesias. Direct Verification of Bisimulations. In Reinhard Gotzhein and Jan Bredereke, editors, *Proceedings of the IFIP TC6/WG6.1 Joint 9th International Conference on Formal Description Techniques and 16th International Workshop on Protocol Specification, Testing and Verification (FORTE/PSTV'96), Kaiserslautern, Germany*, volume 69 of *IFIP Conference Proceedings*, pages 349–363. Chapman & Hall, October 1996.

[349] Mirela Sechi Moretti Annoni Notare, A. Boukerche, Fernando Augusto da Silva Cruz, Bernardo Goncalves Riso, and Carlos Becker Westphall. Security Management Against Cloning Mobile Phones. In *Seamless Interconnection for Universal Services, Global Telecommunications Conference (GLOBECOM'99), Cat. No.99CH37042*, volume 3, pages 1969–1973, 1999.

[350] Mirela Sechi Moretti Annoni Notare, Fernando Augusto da Silva Cruz, Bernardo Goncalves Riso, and Carlos Becker Westphall. Wireless Communications: Security Management Against Cloned Cellular Phones. In *Proceedings of the IEEE Wireless Communications and Networking Conference WCNC'99 (New Orleans, LA, USA)*, pages 1412–1416. IEEE, September 1999.

[351] Mohamad Noureddine, Mohamad Jaber, Simon Bliudze, and Fadi A. Zaraket. Reduction and Abstraction Techniques for BIP. In Ivan Lanese and Eric Madelaine, editors, *Revised Selected Papers of the 11th International Symposium on Formal Aspects of Component Software (FACS'14), Bertinoro, Italy*, volume 8997 of *Lecture Notes in Computer Science*, pages 288–305. Springer, September 2014.

[352] Simona Orzan. *On Distributed Verification and Verified Distribution*. PhD thesis, Vrije Universiteit Amsterdam, 2004.

[353] Simona Orzan, Jaco van de Pol, and Miguel Valero Espada. A State Space Distribution Policy Based on Abstract Interpretation. *Electronic Notes in Theoretical Computer Science*, 128(3):35–45, 2005.

[354] Robert Paige and Robert E. Tarjan. Three Partition Refinement Algorithms. *SIAM Journal of Computing*, 16(6):973–989, December 1987.

[355] Atanas N. Parashkevov and Jay Yantchev. ARC - A Verification Tool for Concurrent Systems. In *Proceedings of the 3rd Australasian Parallel and Real-Time Conference*, 1996.

[356] David Park. Concurrency and Automata on Infinite Sequences. In Peter Deussen, editor, *Theoretical Computer Science – Proceedings of the 5th GI-Conference, Karlsruhe, Germany*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, March 1981.

[357] Joachim Parrow and Björn Victor. The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS'98), Indianapolis, Indiana, USA*, pages 176–185. IEEE Computer Society, June 1998.

[358] Charles Pecheur. Advanced Modelling and Verification Techniques Applied to a Cluster File System. Research Report RR-3416, INRIA, Grenoble, May 1998.

[359] Charles Pecheur. Advanced Modelling and Verification Techniques Applied to a Cluster File System. In Robert J. Hall and Ernst Tyugu, editors, *Proceedings of the 14th IEEE International Conference on Automated Software Engineering (ASE'99) Cocoa Beach, Florida, USA*. IEEE Computer Society, October 1999. Extended version available as INRIA Research Report RR-3416.

[360] Carla Piazza, Enrico Pivato, and Sabina Rossi. CoPS – Checker of Persistent Security. In Kurt Jensen and Andreas Podelski, editors, *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04), Barcelona, Spain*, volume 2988 of *Lecture Notes in Computer Science*, pages 144–152. Springer, March 2004.

[361] Gordon Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Denmark, 1981.

[362] Gordon D. Plotkin. A Structural Approach to Operational Semantics. *Journal of Logic and Algebraic Programming*, 60–61:17–139, 2004.

[363] Gordon D. Plotkin. The Origins of Structural Operational Semantics. *Journal of Logic and Algebraic Programming*, 60–61:3–15, 2004.

[364] Damien Pous and Davide Sangiorgi. Bisimulation and Coinduction Enhancements: A Historical Perspective. *Formal Aspects of Computing*, 31(6):733–749, 2019.

[365] Alexander Moshe Rabinovich. Checking Equivalences Between Concurrent Systems of Finite Agents (Extended Abstract). In Werner Kuich, editor, *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92), Vienna, Austria*, volume 623 of *Lecture Notes in Computer Science*, pages 696–707. Springer, July 1992.

[366] Chris Reade. Process Algebra in the Specification of Graphics Standards. *Computer Standards & Interfaces*, 17:277–290, 1995.

[367] Judi Romijn. *Analysing Industrial Protocols with Formal Methods.* PhD thesis, University of Twente, The Netherlands, September 1999.

[368] A. William Roscoe. *The Theory and Practice of Concurrency.* Prentice Hall, 1998.

[369] Valérie Roy and Robert de Simone. Auto/Autograph. In R. P. Kurshan and E. M. Clarke, editors, *Proceedings of the 2nd Workshop on Computer-Aided Verification (Rutgers, New Jersey, USA)*, volume 3 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 477–491. AMS-ACM, June 1990.

[370] Valérie Roy and Robert de Simone. Auto/Autograph. *Formal Methods in System Design*, 1(2/3):239–249, 1992.

[371] Harry Rudin, Colin H. West, and Pitro Zafiropulo. Automated Protocol Validation: One Chain of Development. *Computer Networks*, 2:373–380, 1978.

[372] Krishan K. Sabnani, Aleta M. Lapone, and M. Ümit Uyar. An Algorithmic Procedure for Checking Safety Properties of Protocols. *IEEE Transactions on Communications*, 37(9):940–948, September 1989.

[373] Meurig Sage and Chris W. Johnson. A Declarative Prototyping Environment for the Development of Multi-User Safety-Critical Systems. In *Proceedings of the 17th International System Safety Conference (ISSC'99), Orlando, Florida, USA*. System Safety Society, August 1999.

[374] Meurig Sage and Chris W. Johnson. Formally Verified Rapid Prototyping for Air Traffic Control. In *Proceedings of the 3rd Workshop on Human Error, Safety and Systems Development, Liege, Belgium*, 1999.

[375] Meurig Sage and Chris W. Johnson. Formally Verified Rapid Prototyping for Air Traffic Control. *Reliability Engineering and System Safety*, 75(2):121–132, February 2002.

[376] M. T. Sanderson. Proof Techniques for CCS. Internal Report CST-19-82, University of Edinburgh, 1982.

[377] Davide Sangiorgi. A Theory of Bisimulation for the π-Calculus. In Eike Best, editor, *Proceedings of the 4th International Conference on Concurrency Theory (CONCUR'93), Hildesheim, Germany*, volume 715 of *Lecture Notes in Computer Science*, pages 127–142. Springer, August 1993.

[378] Pierre de Saqui-Sannes and Jean-Pierre Courtiat. From the Simulation to the Verification of ESTELLE* Specifications. In Son T. Vuong, editor, *Proceedings of the 2nd International Conference on Formal Description Techniques FORTE'89 (Vancouver B.C., Canada)*. North-Holland, December 1989.

[379] Ina Schieferdecker. Abruptly-Terminated Connections in TCP – A Verification Example. In Z. Brezočnik and T. Kapus, editors, *Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design, Maribor, Slovenia*, pages 136–145. University of Maribor, Slovenia, June 1996.

[380] Mihaela Sighireanu. Model-Checking Validation of the LOTOS Descriptions of the Invoicing Case Study. In Henri Habrias, editor, *Proceedings of the International Workshop on Comparing System Specification Techniques (Nantes, France)*, March 1998.

[381] Mihaela Sighireanu and Radu Mateescu. Validation of the Link Layer Protocol of the IEEE-1394 Serial Bus ("FireWire"): an Experiment with E-LOTOS. In Ignac Lovrek, editor, *Proceedings of the 2nd COST 247 International Workshop on Applied Formal Methods in System Design (Zagreb, Croatia)*, June 1997. Full version available as INRIA Research Report RR-3172.

[382] Mihaela Sighireanu and Kenneth J. Turner. Requirement Capture, Formal Description and Verification of an Invoicing System. Research Report RR-3575, INRIA, Grenoble, December 1998.

[383] Ana Sokolova and Erik P. de Vink. Probabilistic Automata: System Types, Parallel Composition and Comparison. In Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, Joost-Pieter Katoen, and Markus Siegle, editors, *Validation of Stochastic Systems – A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 1–43. Springer, 2004.

[384] Lei Song, Lijun Zhang, Holger Hermanns, and Jens Chr. Godskesen. Incremental Bisimulation Abstraction Refinement. *ACM Transactions on Embedded Computing Systems*, 13(4s):142:1–142:23, 2014.

[385] Amelia Soriano. Prototype de VENUS: Un Outil d'Aide à la Vérification de Systèmes Communicants. In Robert Cori and Martin Wirsing, editors, *Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science (STACS'88), Bordeaux, France*, volume 294 of *Lecture Notes in Computer Science*, pages 401–402. Springer, February 1988.

[386] Perdita Stevens. The Edinburgh Concurrency Workbench (Version 7.1). User manual, Laboratory for the Foundations of Computer Science, University of Edinburgh, 1997.

[387] Perdita Stevens. A Verification Tool Seveloper's Vade Mecum. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 2(2):89–94, 1998.

[388] Perdita Stevens. Some Issues in the Software Engineering of Verification Tools. In Rance Cleaveland, editor, *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99), Amsterdam, The Netherlands*, volume 1579 of *Lecture Notes in Computer Science*, pages 435–438. Springer, March 1999.

[389] Carl A. Sunshine. Survey of Protocol Definition and Verification Techniques. *Computer Networks*, 2(4–5):346–350, 1978.

[390] Kuo-Chung Tai and Pramod V. Koppol. An Incremental Approach to Reachability Analysis of Distributed Programs. In *Proceedings of the 7th International Workshop on Software Specification and Design, Los Angeles, CA, USA*, pages 141–150, Piscataway, NJ, December 1993. IEEE Press.

[391] Kuo-Chung Tai and Pramod V. Koppol. Hierarchy-Based Incremental Reachability Analysis of Communication Protocols. In *Proceedings of the IEEE International Conference on Network Protocols, San Francisco, CA, USA*, pages 318–325, Piscataway, NJ, October 1993. IEEE Press.

[392] Alwen Tiu, Nam Nguyen, and Ross Horne. SPEC: An Equivalence Checker for Security Protocols. In Atsushi Igarashi, editor, *Proceedings of the 14th Asian Symposium on Programming Languages and Systems (APLAS'16), Hanoi, Vietnam*, volume 10017 of *Lecture Notes in Computer Science*, pages 87–95. Springer, November 2016.

[393] Stavros Tripakis. Extended Kronos/CADP Tool: Minimization, On-the-Fly Verification and Compositionality. Technical Report T226, VERIMAG, Grenoble, France, April 1999.

[394] Stavros Tripakis and Sergio Yovine. Analysis of Timed Systems Based on Time-Abstracting Bisimulation. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th International Conferenceo on Computer Aided Verification (CAV'96), New Brunswick, NJ, USA*, volume 1102 of *Lecture Notes in Computer Science*, pages 232–243. Springer, July 1996.

[395] Frédéric Tronel, Frédéric Lang, and Hubert Garavel. Compositional Verification Using CADP of the ScalAgent Deployment Protocol for Software Components. In Elie Najm, Uwe Nestmann, and Perdita Stevens, editors, *Proceedings of the 6th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS'03), Paris, France*, volume 2884 of *Lecture Notes in Computer Science*, pages 244–260. Springer, November 2003. Full version available as INRIA Research Report RR-5012.

[396] Andreas Ulrich. A Description Model to Support Test Suite Derivation for Concurrent Systems. In M. Zitterbart, editor, *Kommunikation in Verteilten Systemen, GI/ITG-Fachtagung (KiVS'97), Braunschweig, Germany*, pages 151–166. Springer, 1997.

[397] Andreas Ulrich and Hartmut König. Specification-based Testing of Concurrent Systems. In Teruo Higashino and Atsushi Togashi, editors, *Proceedings of the IFIP Joint International Conference on Formal Description Techniques and Protocol Specification, Testing, and Verification (FORTE/PSTV'97), Ozaka, Japan*. Chapman & Hall, November 1997.

[398] A. Valmari and M. Tienari. An Improved Failure Equivalence for Finite-State Systems with a Reduction Algorithm. In Bengt Jonsson, Joachim Parrow, and Björn Pehrson, editors, *Proceedings of the 11th IFIP International Workshop on Protocol Specification, Testing and Verification (Stockholm, Sweden)*. North-Holland, June 1991.

[399] Antti Valmari. Compositional State Space Generation. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1993 – Papers from the 12th International Conference on Applications and Theory of Petri Nets (ICATPN'91), Gjern, Denmark*, volume 674 of *Lecture Notes in Computer Science*, pages 427–457. Springer, 1993.

[400] Antti Valmari. Stubborn Set Methods for Process Algebras. In Doron A. Peled, Vaughan R. Pratt, and Gerard J. Holzmann, editors, *Proceedings of the DIMACS Workshop on Partial Order Methods in Verification, Princeton, New Jersey, USA*, volume 29 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 213–231. DIMACS/AMS, July 1996.

[401] Antti Valmari, Jukka Kemppainen, Matthew Clegg, and Mikko Levanto. Putting Advanced Reachability Analysis Techniques Together: The ARA Tool. In Jim Woodcock and Peter Gorm Larsen, editors, *Proceedings of the 1st International Symposium of Formal Methods Europe (FME'93), Odense, Denmark*, volume 670 of *Lecture Notes in Computer Science*, pages 597–616. Springer, April 1993.

[402] Tom van Dijk and Jaco van de Pol. Multi-Core Symbolic Bisimulation Minimisation. In Marsha Chechik and Jean-François Raskin, editors, *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16), Eindhoven, The Netherlands*, volume 9636 of *Lecture Notes in Computer Science*, pages 332–348. Springer, April 2016.

[403] Tom van Dijk and Jaco van de Pol. SYLVAN: Multi-Core Framework for Decision Diagrams. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 19(6):675–696, 2017.

[404] Tom van Dijk and Jaco van de Pol. Multi-Core Decision Diagrams. In Youssef Hamadi and Lakhdar Sais, editors, *Handbook of Parallel Constraint Reasoning*, pages 509–545. Springer, 2018.

[405] Tom van Dijk and Jaco van de Pol. Multi-Core Symbolic Bisimulation Minimisation. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 20(2):157–177, 2018.

[406] Rob J. van Glabbeek. The Linear Time - Branching Time Spectrum I. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. North-Holland / Elsevier, 2001.

[407] Rob J. van Glabbeek and W. Peter Weijland. Branching-Time and Abstraction in Bisimulation Semantics (Extended Abstract). In Gerhard X. Ritter, editor, *Proceedings of the IFIP 11th World Computer Congress, San Francisco, USA*, pages 613–618. North-Holland/IFIP, August–September 1989. Also available as CWI Technical Report CS-R8911, Amsterdam, The Netherlands.

[408] Rob J. van Glabbeek and W. Peter Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[409] A. van Rangelrooij and J. P. M. Voeten. CCSTOOL2: An Expansion, Minimization and Verification Tool for Finite State CCS Descriptions. Research Report 94-E-284, Eindhoven University of Technology, 1994.

[410] Didier Vergamini. Verification by Means of Observational Equivalence on Automata. Research Report 0501, INRIA, 1986.

[411] Didier Vergamini. *Vérification de réseaux d'automates finis par équivalences observationnelles: le système AUTO*. PhD thesis, Université de Nice, 1987.

[412] Didier Vergamini. Verification of Distributed Systems: An Experiment. In Jean-Eric Pin, editor, *Formal Properties of Finite Automata and Applications, Proceedings of the LITP Spring School on Theoretical Computer Science, Ramatuelle, France*, volume 386 of *Lecture Notes in Computer Science*, pages 249–259. Springer, 1988.

[413] Björn Victor. A Verification Tool for the Polyadic π-Calculus. Licentiate thesis, Department of Computer Systems, Uppsala University, Sweden, May 1994. Available as report DoCS 94/50.

[414] Björn Victor. The Mobility Workbench User's Guide, Polyadic version 3.122, 1995.

[415] Björn Victor. *The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes*. PhD thesis, Department of Computer Systems, Uppsala University, Sweden, June 1998. Available as report DoCS 98/98.

[416] Björn Victor and Faron Moller. The Mobility Workbench – A Tool for the π-Calculus. In David L. Dill, editor, *Proceedings of the 6th International Conference on Computer Aided Verification (CAV'94), Stanford, California, USA*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer, June 1994.

[417] Heikki Virtanen, Henri Hansen, Antti Valmari, Juha Nieminen, and Timo Erkkilä. Tampere Verification Tool. In Kurt Jensen and Andreas Podelski, editors, *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04), Barcelona, Spain*, volume 2988 of *Lecture Notes in Computer Science*, pages 153–157. Springer, March 2004.

[418] David Walker. Automated Analysis of Mutual Exclusion Algorithms using CCS. Research Report ECS-LFCS-89-91, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburg, 1989.

[419] Farn Wang. Symbolic Branching Bisimulation-Checking of Dense-Time Systems in an Environment. In Rupak Majumdar and Paulo Tabuada, editors, *Proceedings of the 12th International Conference on Hybrid Systems: Computation and Control (HSCC'09), San Francisco, CA, USA*, volume 5469 of *Lecture Notes in Computer Science*, pages 485–489. Springer, April 2009.

[420] Wang Yi. Real-Time Behaviour of Asynchronous Agents. In Jos C. M. Baeten and Jan Willem Klop, editors, *Proceedings of the Conference on Theories of Concurrency: Unification and Extension (CONCUR'90), Amsterdam, The Netherlands*, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer, August 1990.

[421] Wang Yi. A Tool Environment for The Development of Embedded Systems, 1999.

[422] Heike Wehrheim. Partial Order Reductions for Failures Refinement. In Ilaria Castellani and Björn Victor, editors, *6th International Workshop on Expressiveness in Concurrency (EXPRESS'99), Eindhoven, The Netherlands*, volume 27 of *Electronic Notes in Theoretical Computer Science*, pages 71–84. Elsevier, August 1999.

[423] Tim Willemse, Jan Tretmans, and Arjen Klomp. A Case Study in Formal Methods: Specification and Validation of the OM/RR Protocol. In Stefania Gnesi, Ina Schieferdecker, and Axel Rennoch, editors, *Proceedings of the 5th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'00), Berlin, Germany*, GMD Report 91, pages 331–344, Berlin, April 2000.

[424] Tim A.C. Willemse. The Specification and Validation of the OM/RR-Protocol. Master's thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands, June 1998.

[425] Ralf Wimmer, Salem Derisavi, and Holger Hermanns. Symbolic Partition Refinement with Dynamic Balancing of Time and Space. In *Proceedings of the 5th International Conference on the Quantitative Evaluation of Systems (QEST'08), Saint-Malo, France*, pages 65–74. IEEE Computer Society, September 2008.

[426] Ralf Wimmer, Marc Herbstritt, and Bernd Becker. Minimization of Large State Spaces using Symbolic Branching Bisimulation. In Matteo Sonza Reorda, Ondrej Novák, Bernd Straube, Hana Kubátová, Zdenek Kotásek, Pavel Kubalík, Raimund Ubar, and Jirí Bucek, editors, *Proceedings of the 9th IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems (DDECS'06), Prague, Czech Republic*, pages 9–14. IEEE Computer Society, April 2006.

[427] Ralf Wimmer, Marc Herbstritt, and Bernd Becker. Optimization Techniques for BDD-based Bisimulation Computation. In Hai Zhou, Enrico Macii, Zhiyuan Yan, and Yehia Massoud, editors, *Proceedings of the 17th ACM Great Lakes Symposium on VLSI, Stresa, Lago Maggiore*, pages 405–410. ACM, March 2007.

[428] Ralf Wimmer, Marc Herbstritt, Holger Hermanns, Kelley Strampp, and Bernd Becker. SIGREF – A Symbolic Bisimulation Tool Box. In Susanne Graf and Wenhui Zhang, editors, *Proceedings of the 4th International Symposium on Automated Technology for Verification and Analysis (ATVA'06), Beijing, China*, volume 4218 of *Lecture Notes in Computer Science*, pages 477–492. Springer, October 2006.

[429] Pierre Wodey and Fabrice Baray. Linking Codesign and Verification by Means of E-LOTOS FDT. In Lech Józwiak, editor, *Proceedings of the Euromicro Workshop*

on Digital System Design: Architectures, Methods and Tools (Milano, Italy). IEEE, September 1999.

[430] Wei Jen Yeh and Michal Young. Compositional Reachability Analysis Using Process Algebra. In *Proceedings of the ACM SIGSOFT Symposium on Testing, Analysis, and Verification (SIGSOFT'91), Victoria, British Columbia, Canada*, pages 49–59. ACM Press, October 1991.

[431] Michael Yoeli. Modulo-3 Transition Counter: A Case Study in LOTOS-Based Verification. Technical Report TR CS0950, Technion, Computer Science Department, Haifa, Israel, February 1998.

[432] Michael Yoeli. Examples of LOTOS-Based Verification of Asynchronous Circuits. Technical Report TR CS-2001-08, Technion, Computer Science Department, Haifa, Israel, February 2001.

[433] Michael Yoeli and Abraham Ginzburg. LOTOS/CADP-Based Verification of Asynchronous Circuits. Technical Report TR CS-2001-09, Technion, Computer Science Department, Haifa, Israel, March 2001.

[434] Sergio Yovine. Kronos: A Verification Tool for Real-Time Systems. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 1(1/2):123–133, October 1997.

[435] Ivan Zapreev and Christina Jansen. MRMC Test Suite - Version 1.4.1. 2009.

[436] Han Zuidweg. Verification by Abstraction and Bisimulation. In Joseph Sifakis, editor, *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems (CAV'89), Grenoble, France*, volume 407 of *Lecture Notes in Computer Science*, pages 105–116. Springer, June 1989.