Faculty of Informatics
Masaryk University
Czech Republic

From Infinite-State Systems to Translation of LTL to Automata

Habilitation Thesis
(Collection of Articles)

Jan Strejček

April 2012

Abstract

As computer systems are becoming ubiquitous and their importance and complexity are constantly increasing, the importance of verification tools applicable on these systems grows as well. Development of verification tools is usually preceded by studying decidability and complexity of various verification problems for suitable models of computer systems. While some computer systems or their parts can be appropriately modeled by finite-state models, the other systems need to be described by infinite-state models. Formalisms for description of infinite-state models can be divided into two basic categories: formalisms with Turing power and weaker formalisms. The formalisms of the first category have enough expressive power, but nearly all important verification problems are undecidable for them. Weaker formalisms can provide a good compromise between expressiveness and decidability of interesting problems. Many classical formalisms from the second category, e.g. *Petri nets* or *pushdown systems*, are covered by the formalism of *Process Rewrite Systems (PRS)*.

This thesis maps a part of author's research in the area of infinite-state systems and verification in general. We deal mainly with classes of infinite-state systems subsumed in the PRS formalism and its extensions. We study expressive power of these classes and decidability of reachability problem, model checking problem, and equivalence checking problem for these classes. The thesis contain also several results from related areas. In particular, we present decidability of a bounded reachability problem for a Turing-powerful formalism called *Asynchronous Dynamic Pushdown Networks*. Our results on model checking of PRS systems against formulae of *Linear Temporal Logic (LTL)* lead to interesting observations about a certain LTL fragment. These observations allow us to significantly improve translation of LTL to *Büchi automata*, which is a standard part of many LTL model checking algorithms for both finite and infinite-state systems.

The thesis has the form of a collection of articles accompanied by a commentary. The collection contains four journal papers and seven papers published in the proceedings of international conferences or workshops. One paper was written solely by the author of the thesis. In case of all other papers, the contribution is at least proportional to the number of co-authors. The author contributed to these papers in various ways: often by suggesting the topic, partly by bringing the ideas of solution, almost always by discussing and improving solutions and writing down significant parts of texts.

Abstrakt

Jak se počítačové systémy stávají všudypřítomnými a jak roste jejich význam i složitost, roste i význam nástrojů schopných tyto systémy verifikovat. Vývoji verifikačního nástroje zpravidla předchází výzkum rozhodnutelnosti a složitosti různých verifikačních problémů na vhodných modelech počítačových systémů. Zatímco některé systémy lze adekvátně modelovat pomocí konečně stavových modelů, jiné systémy je třeba modelovat jako nekonečně stavové. Formalismy pro popis nekonečně stavových modelů lze rozdělit do dvou základních kategorií: formalismy s Turingovskou silou a slabší formalismy. Formalismy první kategorie mají velkou popisnou sílu, ale téměř všechny důležité verifikační problémy jsou pro ně nerozhodnutelné. Slabší formalismy mohou nabídnout dobrý kompromis mezi vyjadřovací sílou a rozhodnutelností zajímavých problémů. Mnoho běžných formalismů z druhé kategorie (jako *Petriho sítě* nebo *zásobníkové systémy*) je zastřešeno formalismem *procesových přepisovacích systémů (PRS)*.

Tato habilitační práce mapuje část autorova výzkumu v oblasti nekonečně stavových systémů a verifikace obecně. Zabýváme se především třídami nekonečně stavových systémů pokrytých formalismem PRS a jeho rozšířeními. Zkoumáme vyjadřovací sílu těchto tříd a rozhodnutelnost problému dosažitelnosti, problému ověřování modelu a problému ověření ekvivalence pro tyto třídy. Habilitační práce dále obsahuje několik výsledků ze souvisejících oblastí. Zejména prezentujeme rozhodnutelnost omezené dosažitelnosti pro formalismus *asynchronních sítí dynamických zásobníkových systémů*, který má Turingovskou sílu. Naše výsledky o ověřování modelu pro PRS systémy a formule *lineární temporální logiky (LTL)* vedly k zajímavým poznatkům o jistém fragmentu LTL. Tyto poznatky nám dovolují významně zlepšit překlad LTL na *Büchiho automaty*, který je standardní součástí mnoha algoritmů pro ověřování modelu LTL vlastností pro konečně i nekonečně stavové systémy.

Tato habilitační práce je koncipována jako soubor uveřejněných vědeckých prací doplněný komentářem (§72 odst. 3 písmena b zákona č. 111/1998 Sb., o vysokých školách). Soubor obsahuje čtyři časopisecké články a sedm článků ze sborníků mezinárodních konferencí a seminářů. U jednoho článku je uchazeč jediným autorem. U ostatních článků je jeho autorský podíl přinejmenším úměrný počtu autorů. Uchazeč se podílel na vzniku těchto článků různými způsoby: často navrhl téma, někdy přišel s myšlenkami vedoucími k řešení, téměř vždy diskutoval a vylepšoval řešení a napsal podstatné části textů.

Acknowledgments

First of all, I would like to thank Mojmír Křetínský and Vojtěch Řehák for fruitful and long-lasting collaboration and wide-ranging discussions. I thank Mojmír Křetínský also for reading a draft of this thesis.

Many thanks go to other co-authors of papers included in this collection, namely to Tomáš Babiak, Ahmed Bouajjani, Laura Bozzelli, Javier Esparza, Stefan Schwoon, and Tayssir Touili.

Last but not least, I thank my wife Adriana and our children Zorka and Andy for their constant support, infinite patience, and love.

Contents

Part I

Commentary

Chapter 1

Introduction

1.1 Motivation

The significance of computer systems for our society is constantly increasing. A failure of these systems can have an enormous negative impact on an individual, a company, or even a country. Hence, there is a great demand for methods and technologies to increase reliability of computer systems. The traditional techniques like simulation, testing, and error-preventing methodologies for system design are clearly not sufficient. In particular, these methods aim to eliminate errors, but they do not demonstrate an absence of errors, i.e. they do not prove correctness of the systems.

Proving correctness of computer systems is the original motivation for development of *formal methods*, which started in the late sixties. The area of formal methods now subsume formalisms for specification and modeling of systems as well as techniques and algorithms for analysis and verification of systems or their models.

The modeling formalisms can be roughly divided into the following three categories according to their expressive power.

Formalisms describing finite-state systems
This category contains formalism which can describe only systems with a finite number of states. The category contains some low-level formalisms as *finite labeled transition systems* or *finite Kripke structures* as well as many high-level formalisms. High-level formalisms are typically designed for compact description of large finite-state systems. Finite-state systems are also called *finite systems*.

Formalisms with Turing power
This category contains formalisms that can accurately model all Turing machines. As a Turing machine can in one computation successively enter infinitely-many different states, the formalism can model systems with an infinite number of states.

Formalisms describing infinite-state systems without Turing power
This category contains formalisms that can describe systems with an infinite number of states, but they cannot accurately model all Turing machines.

Formalisms of each category have some benefits and drawbacks. Finite systems can faithfully model all hardware systems and thus also all self-contained software/hardware systems. Moreover, all verification problems are decidable for finite systems. Unfortunately, finite-state models of real systems are often extremely large even if their description in a suitable formalism can be relatively small. Thus, they cannot be effectively analyzed by standard algorithms designed for finite-state systems. It may therefore be more convenient to model them in formalisms for description of infinite-state systems and analyze them by the corresponding algorithms.

Modeling extremely large finite systems is not the only motivation for formalisms describing infinite-state systems. For example, if we want to analyze algorithms rather then programs, we also need to model them as infinite-state systems. The reason is that algorithms typically work with unbounded data domains, while all data domains in real programs are finite.

Formalisms with Turing power have an obvious drawback: even the very basic verification problems like reachability of a given state are not decidable. In practice, models in these formalisms are analyzed using algorithms with some imprecision. For example, there are bounded analysis approaches examining only a part of a state space.

Formalisms describing infinite-state systems without Turing power represent a compromise between expressive power and decidability of verification problems.

Formal methods cover several verification problems. The following three are the most traditional and the most prominent.

Reachability problem is the problem to decide whether a given state of a given system is reachable from the initial state or not. A typical application of reachability problem is verification of a basic *safety property*: a system is considered to be safe if a given error state is not reachable.

Model checking problem is the problem to decide whether each run of a given system satisfies a given specification. The specification is usually described by a formula of some temporal logic.

Equivalence checking problem is the problem to decide whether two given systems, typically a specification and its implementation, are equivalent with respect to a given behavioral equivalence.

All the mentioned problems are decidable for finite systems regardless the temporal logic used for specification in the case of model checking or behavioral equivalence in case of equivalence checking. On the contrary, none of the problems is decidable for formalisms with Turing power (unless we consider some trivial temporal logic or behavioral equivalence). The decidability of these problems for formalisms describing infinite-state systems without Turing power depends on concrete formalism or, in case of model checking or equivalence checking, on a combination of a concrete formalism and temporal logic or behavioral equivalence.

1.2 Focus

In this thesis, we focus mainly on the expressiveness and decidability issues of formalisms covered by *Process Rewrite Systems (PRS)* [69, 71]. PRS is a term rewriting formalism. Its popularity stems from two facts. First, many classical and well-studied formalisms for description of infinite-state systems can be uniformly obtained from PRS by imposing simple restrictions on syntax of rewrite rules. For example, *Petri nets* or *pushdown systems* can be seen as syntactical subclasses of PRS. Second, PRS has not Turing power and some interesting instances of verification problems are decidable for PRS. In the following chapter, we recapitulate basic decidability results for the mentioned verification problems and formalisms subsumed in PRS. In the model checking problem, we focus our attention to action-based semantics only. For the equivalence checking problem, out of dozens studied equivalences surveyed in [96], we consider only two most prominent equivalences, namely *strong* and *weak bisimulation equivalences* [73, 79, 74].

The second topic of this thesis is a translation of *Linear Temporal Logic (LTL)* [81] to *Büchi automata (BA)*. LTL is probably the most frequently used formalism for specification of system properties for model checking purposes. A translation of LTL to BA is the first step of many LTL model checking algorithms for both finite and infinite-state systems.

The relation between the two topics is seemingly feeble. However, our results in LTL to automata translation are directly motivated by our previous research of LTL model checking problem for PRS classes. Moreover, the results on LTL to automata translation have again direct consequences to decidability of LTL model checking for various classes of infinite-state systems (not only subclasses of PRS). This relationship is explained in Section 3.2.

Chapter 2

State of the Art

2.1 Process Rewrite Systems

The term rewriting formalism called *Process Rewrite Systems (PRS)* is introduced in [69]. PRS combines both sequential and parallel operators. Hence, it can be seen as an extension of a similar, but purely sequential formalism studied in [20] and a purely sequential and purely parallel formalisms studied in [76].

By imposing various restrictions on syntax of PRS rewrite rules we get several standard classes of infinite-state systems, namely *Basic Process Algebras (BPA)* [4], *Basic Parallel Processes (BPP)* [21], *Process Algebras (PA)* [4], *Pushdown Processes (PDA)*, and *Petri Nets (PN)*. Another syntax restrictions of PRS define the class of all *Finite Systems (FS)* and classes called *PAD* [69] (common generalization of PA and PDA) and *PAN* [68] (common generalization of PA and PN).

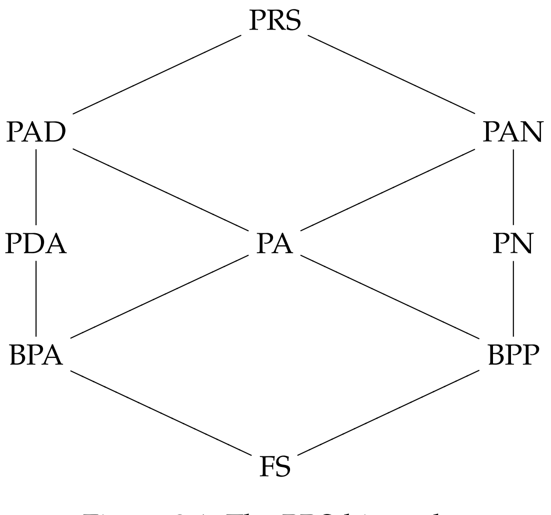


Figure 2.1: The PRS-hierarchy

The mentioned classes form so-called *PRS-hierarchy* depicted in Figure 2.1. The hierarchy is in fact a Hasse diagram where the classes are partially ordered according to their expressive power as follows: each class represents the set of corresponding labeled transition systems and the partial order compares these sets according to the set

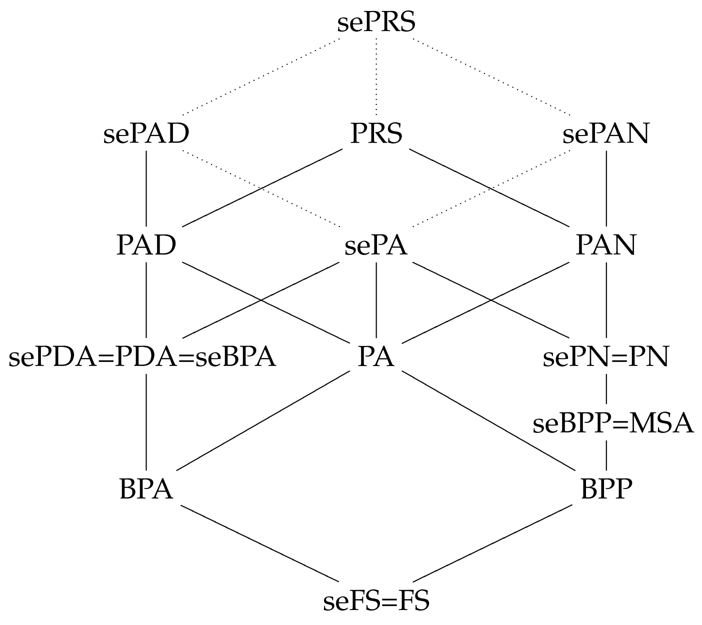


Figure 2.2: The state-extended PRS-hierarchy. The dotted lines represent the fact, that strict increase of expressive power is only conjectured.

inclusion modulo strong bisimulation equivalence [79, 74]. The shape of the hierarchy copies relations between syntactical restrictions defining the classes. Strictness of the hierarchy follows from [18, 76, 69].

A disadvantage of PRS is its limited expressive power. As PRS subsumes pushdown processes, it can accurately model sequential programs with unlimited recursion and both local and global variables over finite domains. As PRS contains Petri nets, it can model also synchronously communicating parallel systems with dynamic creation of new processes, but the processes cannot have unbounded recursion. PRS cannot model even two asynchronous sequential programs with unbounded recursion where one program sends one message to the other program. The connection between PRS classes programming features is nicely explained in [33].

The modeling power can be improved by enriching PRS with an additional finite-state control unit. The resulting formalism is called *state-extended PRS (sePRS)* [51, 44]. Using the syntactic restriction applied to PRS, we get analogous subclasses of sePRS, denoted by *se-* prefix. Addition of the finite control unit does not change expressive power of classes FS, PDA, and PN. The expressive power of other PRS subclasses strictly increases: seBPA class has the same expressive power as PDA [20], while seBPP coincides with a previously studied class called *Parallel Pushdown Processes (PPDA)* [76] or *Multiset Automata (MSA)* [77]. The strict increase of power is proven also for sePA, sePAD, and sePAN. In fact, we conjecture that the classes sePA, sePAD, sePAN, and sePRS form fresh classes of infinite-state systems. Figure 2.2 depicts the original hierarchy extended with the state-extended classes. The figure also marks relations that are not rigorously proven, but only conjectured. The shape of the hierarchy follows from the definition of sePRS, shape of the PRS hierarchy, results of [20, 71, 44], and our result of [55, 59]. The mentioned conjectures are discussed in [97].

2.1.1 Reachability Problem

The reachability problem, i.e. the problem to decide whether a given state is reachable from a given initial state, is decidable for PRS [71]. The proof employs the previously known results on decidability of reachability for Petri nets [67] and for pushdown processes [6].

Four out of the five new classes obtained by state-extension have Turing power. Indeed, sePA and its superclasses sePAD, sePAN, and sePRS can encode any *Minsky 2-counter machine* [75] and thus also any Turing machine. As a result, the reachability problem is undecidable for these classes [5]. The remaining class seBPP is a subclass of Petri nets and hence the reachability problem is also decidable for seBPP.

Other versions of reachability are studied as well. For example, *reachable property problem*, i.e. the problem to decide whether there exists a reachable state where all actions of one set are enabled and all actions of another set are disabled. The decidability of reachable property problem is the same as for the original reachability problem: it is decidable for PRS [71] and undecidable for sePA [5].

A lot of attention has been also devoted to *symbolic reachability* on various PRS subclasses, where the goal is to compute a finite representations of the set (or its under- or over-approximation) of all states reachable (or backward reachable) from a given set of states. The main results in this area can be found in [64, 36, 34, 11, 8, 12].

2.1.2 Model Checking Problem

For the model checking problem, we consider branching-time logics of Figure 2.3 and linear-time logic LTL.

The decidability boundary of the model checking problem heavily depends on the considered logic.

The logic EG and all stronger logics For the logic EG, the problem is decidable for PDA and its subclasses [78, 20]. The same result holds for all stronger logics, as the problem is decidable for PDA even for specification formulated in μ-calculus [19, 98].

On the contrary, the problem is undecidable for the logic EG (and thus also for all stronger logics) and all classes subsuming BPP [35]. In fact, an inspection of this undecidability proof exposes that the undecidability result for BPP holds already for a fragment of EG containing only formulae of the form $\mathsf{EG}\varphi$, where φ is a Hennessy-Milner formula.

The logic EF For the logic EF, the problem is decidable for PAD due to [70], while it is undecidable for PN [31]. The undecidability proof for PN actually works also for seBPP, as we mention in [57].

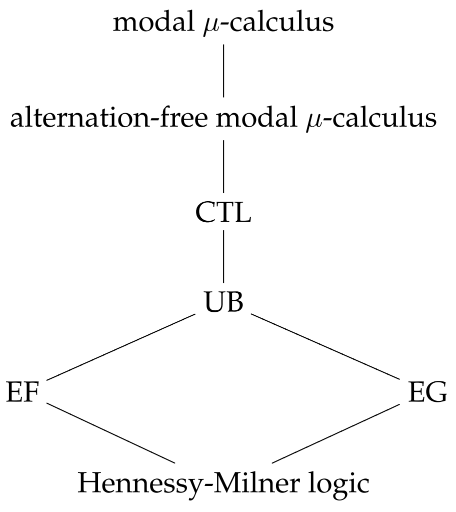


Figure 2.3: Hierarchy of branching-time logics

Formulae $\mathsf{EF}\varphi$**, where** φ **is a Hennessy-Milner formula** These formulae form a fragment of the logic EF. The model checking problem for such formulae is also called *reachability Hennessy-Milner property* and its relevance is advocated in [51, 54]. In contrast to model checking problem for the full logic EF, this problem is decidable for both PAD [70, 51] and PN [52], and undecidable for sePA [5]. We show that the problem is decidable also for PAN and PRS [57].

The Hennessy-Milner logic A Hennessy-Milner formula describes only the first n steps of system behaviors, where n depends on the formula. Hence, the logic in clearly decidable for all classes of PRS and their state-extended versions.

The logic LTL The LTL model checking problem is decidable for PDA [6] and PN [31]. The problem is undecidable for all classes subsuming PA [9, 70]. If we ignore finite runs of PRS systems, then the model checking problem is decidable for PA and the logic called *simple PLTL*$_\Box$ [9], and also for all PRS classes and fairness properties [13].

For detailed information on decidability and complexity of the model checking problem for PRS classes we refer to exhaustive surveys [70, 17].

2.1.3 Equivalence Checking Problem

Strong bisimulation equivalence is decidable for PDA [86] and for BPP [22]. The problem is undecidable for seBPP [76, 49]. Decidability for PA and PAD is an open question.

Strong bisimulation equivalence is usually considered to be too fine for practical purposes. Weak bisimulation equivalence seem to be more convenient. Unfortunately, there is no positive decidability result for weak bisimulation equivalence on any class of infinite-state systems mentioned so far. However, there are some positive results for strict subclasses of BPP [43, 32, 91]. The equivalence is clearly decidable for finite systems. It is undecidable for PDA [88], PA [89], and seBPP [76]. The decidability is conjectured for BPA [72] and for BPP [50].

For detailed information on decidability and complexity of the equivalence checking problem for PRS classes we refer to surveys [17, 53, 90].

2.2 Translation of LTL to Büchi Automata

For a long time, researchers aimed to find fast translations producing Büchi automata with a small number of states. This goal has led to development of several translation algorithms and many heuristics and optimizations including input formula reductions and optimizations of produced Büchi automata, see e.g. [41, 24, 25, 37, 87, 40, 42, 39, 28, 30].

As the time goes, the translation objectives and their importance are changing. For example, [85] demonstrates that for higher performance of the subsequent steps of the model checking process, it is more important to minimize the number of states with a nondeterministic choice than the number of all states in resulting automata. Note that there are LTL formulae for which no equivalent deterministic Büchi automaton exists. This new objective has led to development of algorithms focusing on determinism of produced automata. For example, [26] presents an effective algorithm translating LTL formulae of the fragment called *obligation* (see [66]) into *weak deterministic Büchi automata (WDBA)*. Moreover, WDBA can be minimized by the algorithm of [62].

There are also many tools translating LTL to Büchi automata. Intensive experiments [83] show that two of them, LTL2BA [40] and SPOT [28], notably outperform the others. While SPOT is under the gradual development following the current trends (see [27] for improvements made in the last four years), LTL2BA underwent only one minor update in 2007 since its start in 2001. As a result, SPOT usually produces more deterministic and smaller automata than LTL2BA, while LTL2BA is often a bit faster.

Chapter 3

Thesis Contribution

3.1 Process Rewrite Systems

We present two new extensions of PRS: *PRS with finite constraint system (fcPRS)* [92] and *weakly extended PRS (wPRS)* [56]. The latter extension is very similar to the extension with finite control unit called sePRS. The only difference is that a state of the added control unit can be changed only in accordance with a given partial order. We sometimes talk about *weak finite control unit*. Hence, wPRS can be seen as sePRS with a simple restriction. The extension with finite constrained system is inspired by *concurrent constraint programming* [84] and can be seen as a restricted version of wPRS. Both extensions increase expressive power of all the classes of the PRS-hierarchy except FS, PDA, and PN, but the extended classes still do not possess Turing power and some interesting verification problems remain decidable for them. Let us note that the increase of expressive power is again only conjectured for PRS.

All considered subclasses of PRS, fcPRS, wPRS, and sePRS can be ordered according to their expressive power (up to strong bisimulation) to the hierarchy depicted on Figure 3.1. Shape of the hierarchy directly follows from the definitions of considered formalisms except two cases: the equality between PDA and seBPA follows from [20] and the inclusion of PN in sePA follows from [55, 59], where we prove that each PN can be translated into a strongly bisimilar sePA. Strictness of the hierarchy is proven in [92, 56]. Conjectures are discussed in [97].

3.1.1 Reachability Problem

In [55, 61] we prove that reachability problem remains decidable for all classes of wPRS. The papers also present several applications of the result. In particular, the weak finite state unit of wPRS allow us to reduce two interesting problems to reachability. First, we show that model checking of certain simple safety properties is decidable for wPRS. The second studied problem is decidability of *weak trace equivalence* mentioned, for instance, in [45]. We prove semi-decidability of the weak trace non-equivalence for wPRS. Let us note that semi-decidability of this problem has been previously known only for PN [48], PDA [16], and PA [64]. Decidability of reachability for wPRS has also other applications, for example, in analysis of some cryptographic

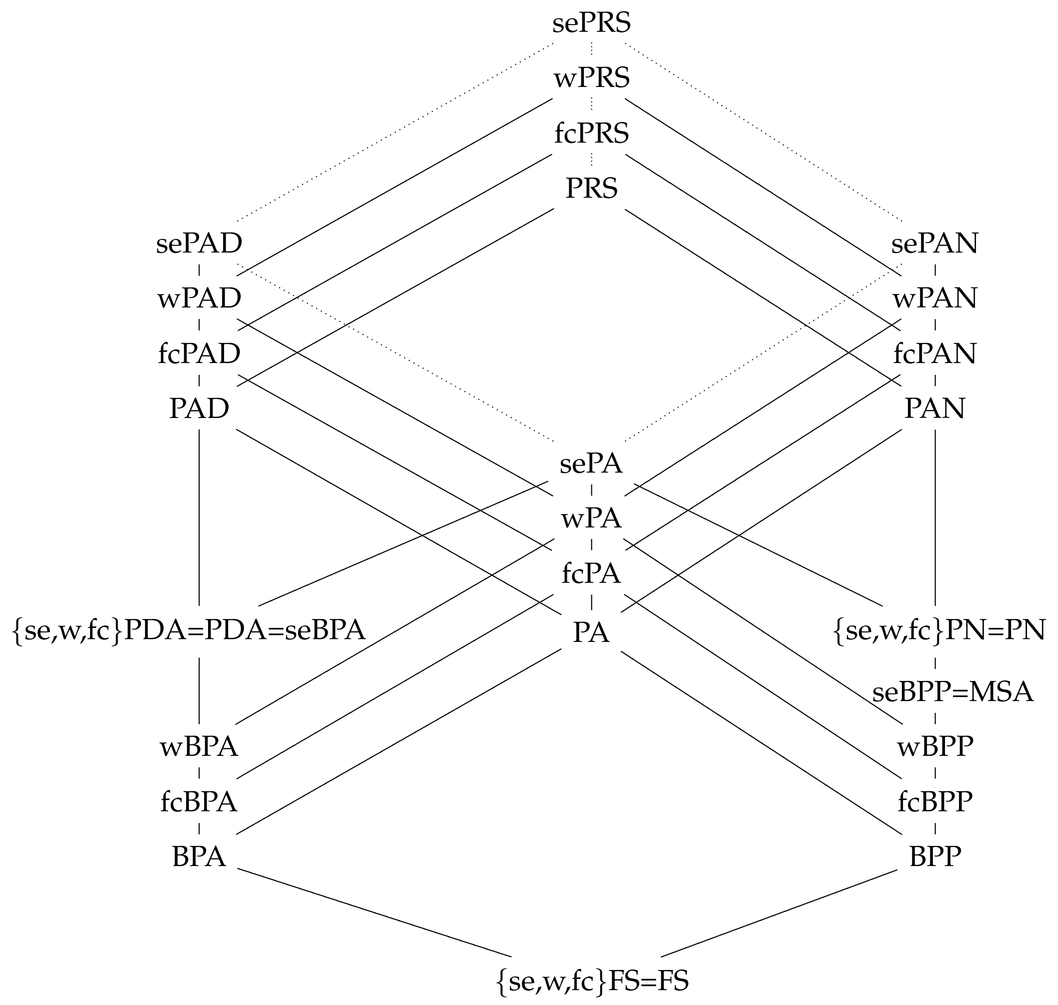


Figure 3.1: The hierarchy of PRS classes and their extended versions. The dotted lines represent the fact, that strict increase of expressive power is only conjectured.

protocols [46, 47].

In [10] we extend the result on symbolic reachability analysis for PAD presented in [12]. More precisely, [12] shows that, given a PAD in a certain normal form and a set of PAD states represented by a *commutative-hedge automaton (CH-automaton)*, one can compute another CH-automaton representing the set of all states that are (forward or backward) reachable from states of the original set. In [10] we prove the same for arbitrary wPAD system. The result has some consequences for decidability of the model checking problem for the logic EF and wPAD (see the following subsection).

In [7] we study a formalism called *Asynchronous Dynamic Pushdown Network (ADPN)*, which is similar to wPAD. In contrast to wPAD, ADPN can dynamically create new independent processes and global control unit of ADPN is not weak. Hence, ADPN are well suited to model multithreaded software with asynchronous communication. Unfortunately, the formalism has Turing power and the reachability problem is thus undecidable. Therefore, we study *k-bounded reachability problem* [82], i.e. we consider only reachability in *k contexts*, where a context is a transition sequence during which the global control unit of ADPN interacts with one process only. In [7] we generalize and improve the algorithm for k-bounded reachability analysis introduced in [82].

3.1.2 Model Checking Problem

We have studied decidability of model checking for the new subclasses of fcPRS and wPRS. In several cases we also refine decidability borders by considering finer fragments of temporal logics.

The logic EG and all stronger logics The decidability borderline of the model checking problem for the logic EG and all stronger logics is fully determined by the results mentioned in Subsection 2.1.2. However, we have strengthened the undecidability result for BPP: while [35] shows undecidability of the model checking problem for BPP and the logic EG, we show that the problem is undecidable even for the fragment of EG containing formulae of the form $\mathsf{EG}\varphi$, where φ is a Hennessy-Milner formula without nesting of temporal operators $\langle\cdot\rangle$ [57].

The logic EF In [10] we prove decidability of the model checking problem for wPAD and the EF logic. In fact, the result works with a more general logic than EF and a more general problem called *global model checking*. While the ordinary model checking decides validity of a given formula in a given state of the system, the global model checking computes the set of all states of the system satisfying the formula. In our case, the set is represented by a CH-automaton.

Formulae $\mathsf{EF}\varphi$, where φ is a Hennessy-Milner formula We study the reachability Hennessy-Milner property problem in [57]. We prove that the problem is decidable for the whole class of wPRS. This result answers open questions on decidability of the problem for classes PAN and PRS.

It is known that decidability of the reachability Hennessy-Milner property problem for a class C implies decidability of strong bisimilarity between systems of C and finite-state systems [51]. As a corollary we get that strong bisimulation equivalence between wPRS and FS is decidable, which answers the corresponding open problems formulated for PAN and PRS.

The Hennessy-Milner logic As discussed in Subsection 2.1.2, this model checking problem is decidable for sePRS all its subclasses.

The logic LTL Again, the situation for the LTL model checking problem is fully determined by the results mentioned in Subsection 2.1.2. In particular, LTL model checking is undecidable for all classes subsuming PA. However, in [14, 60, 15] we present some positive results for the whole wPRS class and some fragments of LTL. We have studied the problem for fragments of LTL formulae defined by a restricted set of temporal operators. For example, the fragment LTL(F, G) contains all LTL formulae build with temporal operators *eventually* (F) and *always*

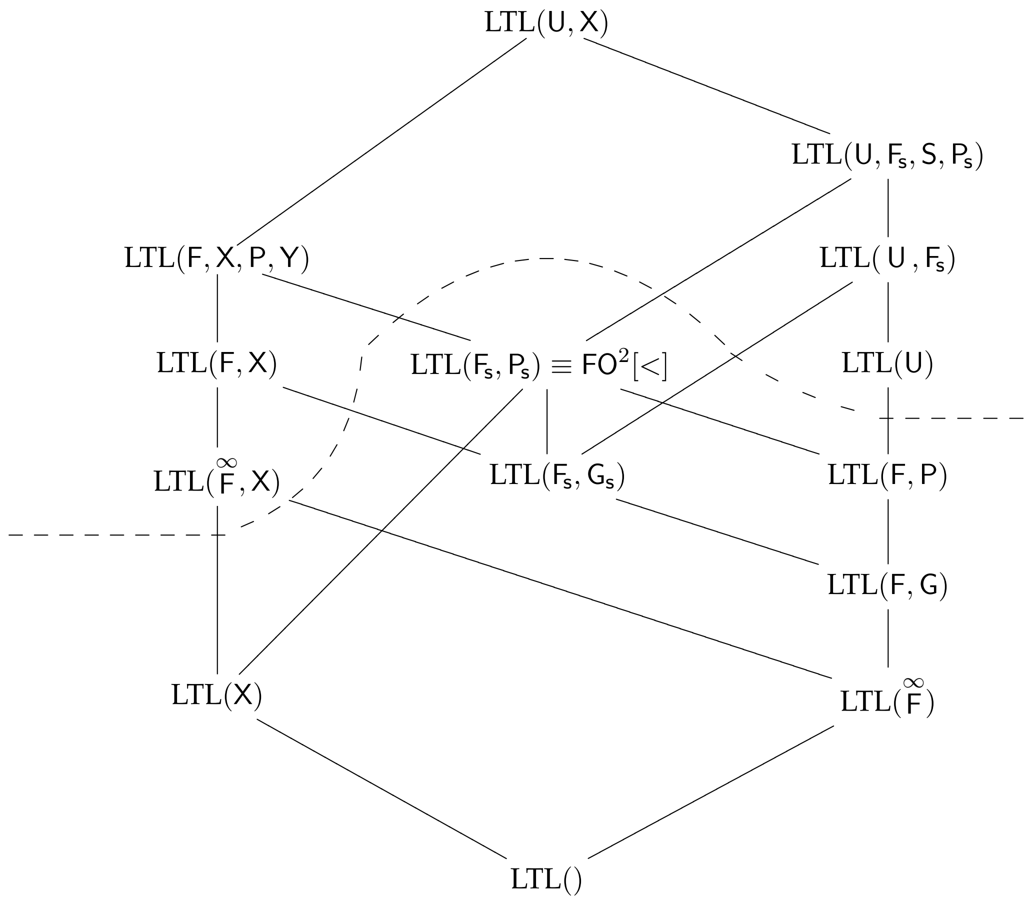


Figure 3.2: The hierarchy of LTL fragments with respect to their expressive power. The dashed line shows the decidability boundary of the model checking problem for wPRS: the problem is decidable for all the fragments below the line, while it is undecidable for all the fragments above the line (even if we consider PA systems only).

(G) only. The hierarchy of considered LTL fragments reflecting their expressive power is given on Figure 3.2. Note that LTL(U, X) corresponds to the full LTL while LTL() denotes the fragment without any temporal operators. All the mentioned fragments put no restrictions on negation and other boolean connectives. For definitions of temporal operators and for justification of the hierarchy we refer to an exhaustive survey [93].

In [14] we show that the model checking problem is decidable for wPRS and the fragment LTL(F_s, G_s) containing formulae with temporal operators *strict eventually* (F_s) and *strict always* (G_s). This fragment can express many properties of verification practice. The proof employs our previous results on reachability for wPRS and decidability of the model checking problem for PRS and fairness properties [13]. Further, in [14] we also prove that the model checking problem is undecidable for PA and fragments LTL($\overset{\infty}{F}$, X) or LTL(U).

In [60] we extend the positive decidability result for wPRS and LTL($\mathsf{F_s}, \mathsf{G_s}$) to a strictly stronger fragment LTL($\mathsf{F_s}, \mathsf{P_s}$) build with temporal operators *strict eventually* ($\mathsf{F_s}$) and its past counterpart *eventually in the strict past* ($\mathsf{P_s}$). The fragment is expressively equivalent to the fragment $\mathsf{FO}^2[<]$ of *first-order monadic logic of order* containing formulae with at most 2 variables and no successor predicate. For effective translation between LTL($\mathsf{F_s}, \mathsf{P_s}$) and $\mathsf{FO}^2[<]$ we refer to [38].

Finally, in [60] we show that the model checking problem is decidable for wPRS and the LTL fragment LTL$^{\text{det}}$ also known as *the common fragment of CTL and LTL* [65], where CTL refers to *Computation Tree Logic* [23].

3.1.3 Equivalence Checking Problem

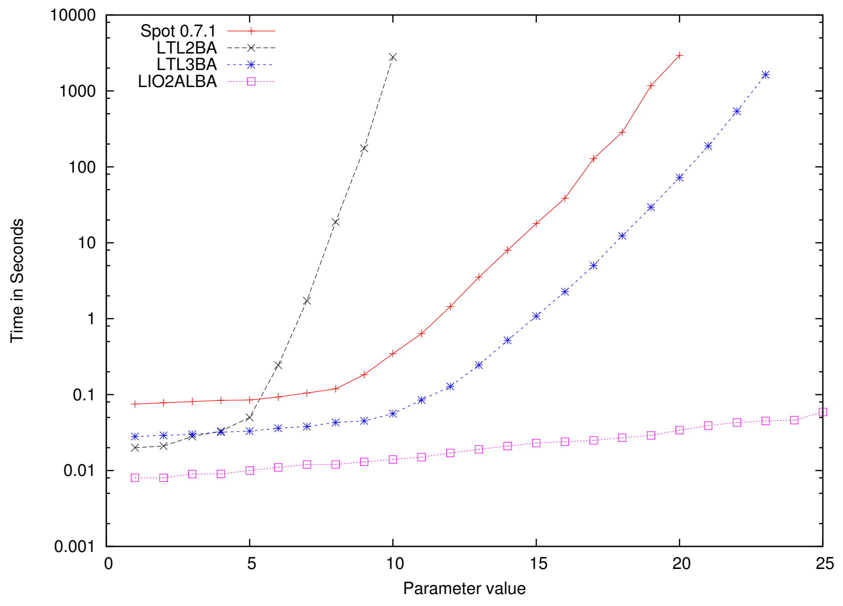
Besides the results on (semi)decidability of weak trace non-equivalence and strong equivalence with finite systems mentioned above, we add also one undecidability results regarding weak bisimilarity. In [58] we prove that the equivalence checking problem is undecidable for weak bisimulation equivalence even for classes fcBPA and fcBPP. Actually, we demonstrate that the undecidability holds for normed fcBPA and normed fcBPP, which are strict subclasses of fcBPA and fcBPP respectively. The decidability of equivalence checking for weak bisimilarity and BPA or BPP remains open.

3.2 Translation of LTL to Büchi Automata

Our decidability results of the model checking problem for wPRS and fragments LTL($\mathsf{F_s}, \mathsf{G_s}$), LTL($\mathsf{F_s}, \mathsf{P_s}$), and LTL$^{\text{det}}$ are all based on one auxiliary result: we introduce an LTL fragment $\mathcal{A}$ and we show that the *existential model checking problem*, i.e. the problem whether a given system has a run satisfying a given formula, is decidable for wPRS and formulae of $\mathcal{A}$. To prove decidability of the model checking problem for LTL($\mathsf{F_s}, \mathsf{G_s}$), LTL($\mathsf{F_s}, \mathsf{P_s}$), and LTL$^{\text{det}}$, we show that each formula φ of these fragments can be translated to a formula of $\mathcal{A}$ that is equivalent to $\neg\varphi$.

The fragment $\mathcal{A}$ is strictly more expressible than negations of formulae in LTL($\mathsf{F_s}, \mathsf{G_s}$), LTL($\mathsf{F_s}, \mathsf{P_s}$), and LTL$^{\text{det}}$. Further, the fragment $\mathcal{A}$ is not suitable for practical purposes as it has very restricted syntax. Thanks to their specific structure, formulae of $\mathcal{A}$ straightforwardly correspond to Büchi automata that are *linear* (i.e. the only cycles are self-loops) with a possible exception of terminal strongly connected components. Each terminal strongly connected component accepts infinite words over a set of letters, where some selected letters appear infinitely often. In [2], we define the class of these Büchi automata called *Almost Linear Büchi Automata (ALBA)*. Further, we introduce an LTL fragment called *LIO*, which is expressively equivalent to ALBA and thus also to $\mathcal{A}$. In contrast to $\mathcal{A}$, LIO is a syntactically rich fragment subsuming LTL(F, G). To show that LIO and ALBA can describe the same class of languages, we present translations between the two formalisms.

An improved version of the LIO to ALBA translation is described in [3], where we also provide complexity analysis of the translation. While general LTL to BA translations produce automata that are at most exponential in the size of input formulae,



$$\theta_n = \neg((\mathsf{GF}p_1 \wedge \ldots \wedge \mathsf{GF}p_n) \rightarrow \mathsf{G}(q \rightarrow \mathsf{F}r))$$

Figure 3.3: Time consumption of translation of parametric formula θ_n. The vertical axis is logarithmic and represents time in seconds. The horizontal axis represents the parameter n. All translators produce the same automata.

our LIO to ALBA translation produces automata of at most doubly exponential size. To analyze significance of LIO and applicability of the LIO to ALBA translation to specification formulae used in practice, we have implemented the translation and we study its usability on 75 LTL specification formulae from two public sources: *Spec Patterns* [29] and *BEEM* [80]. In fact, we consider negations of these formulae as model checking algorithms typically need Büchi automata for negated specification formulae. After some simple syntactic transformations, 50 out of the 75 negated formulae are syntactically in LIO. We have translated them by our LIO2ALBA translator and the general LTL2BA translator [40]. We have obtained automata of fully comparable sizes. Hence, many real-world specification formulae can be translated to automata of the specific shape without any substantial increase of size.

The specific shape of ALBA has already found an application in [95, 94]. The papers formulate a sufficient condition on a formalism for description of infinite-state systems that implies decidability of the existential model checking problem for the formalism and specifications in the form of ALBA automata. In connection with a translation of an LTL fragment to ALBA, the condition implies decidability of the model checking problem for negations of the LTL fragment. Using this technique, [95, 94] derive decidability of the model checking problem for LTL($\mathsf{F}_\mathsf{s}, \mathsf{G}_\mathsf{s}$) and all formalisms satisfying the condition. Let us note that the condition is quite general: it is satisfied by all subclasses of wPA and by many other formalisms, e.g. *concurrent pushdown systems* [82] and *ground-tree rewrite systems* [63].

Experiments with the LIO2ALBA translator expose the fact that the translation is extremely efficient on specific kinds of formulae comparing the two leading LTL to BA translators LTL2BA [40] and SPOT [28]. One such a case is illustrated on Figure 3.2.

Hence, we adopt some ideas of LIO to ALBA translation to improve the LTL to BA translation algorithm of [40]. Further, we suggest some techniques reducing nondeterminism of the produced automata. The improved translation is described in [1] and implemented in the tool called LTL3BA. Experimental results show that LTL3BA usually outperforms LTL2BA in all aspects and it competes with SPOT. However, Figure 3.2 demonstrates that LIO2ALBA can be massively faster than LTL3BA, at least in some cases. Hence, there is still some space for improvements of LTL to BA translations.

Chapter 4

Papers in Collection

4.1 Journal Papers

[59] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. Petri nets are less expressive than state-extended PA. *Theoretical Computer Science*, 394(1–2):134–140, 2008.

- The result has been originally published on CONCUR 2004 [55].
- Author's contribution: 33%, participating on all phases

[15] Laura Bozzelli, Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. On decidability of LTL model checking for process rewrite systems. *Acta Informatica*, 46(1):1–28, 2009.

- The result has been originally published on FSTTCS 2006 [14].
- Author's contribution: 25%, bringing the topic, participating on all phases, significant part of writing

[61] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. Reachability is decidable for weakly extended process rewrite systems. *Information and Computation*, 207(6):671–680, 2009.

- The result has been originally published on CONCUR 2004 [55].
- Author's contribution: 33%, participating on all phases

[3] Tomáš Babiak, Vojtěch Řehák, and Jan Strejček. Almost linear Büchi automata. *Mathematical Structures in Computer Science*, 22(02):203–235, 2012.

- The result has been originally published on EXPRESS 2009 [2].
- Author's contribution: 33%, bringing the topic, participating on all phases except implementation and measurements, significant part of writing

4.2 Proceedings Papers

[92] Jan Strejček. Rewrite systems with constraints. In *Proceedings of EXPRESS 2001*, volume 52 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2002.

- Author's contribution: 100%

[56] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. On extensions of process rewrite systems: Rewrite systems with weak finite-state unit. In *Proceedings of INFINITY 2003*, volume 98 of *Electronic Notes in Theoretical Computer Science*, pages 75–88. Elsevier Science Publishers, 2004.

- Author's contribution: 33%, participating on all phases

[57] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. Reachability of Hennessy-Milner properties for weakly extended PRS. In *Proceedings of FSTTCS 2005*, volume 3821 of *Lecture Notes in Computer Science*, pages 213–224. Springer, 2005.

- Author's contribution: 33%, participating on all phases

[7] Ahmed Bouajjani, Javier Esparza, Stefan Schwoon, and Jan Strejček. Reachability analysis of multithreaded software with asynchronous communication. In *Proceedings of FSTTCS 2005*, volume 3821 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2005.

- Author's contribution: 25%, substantial part of ideas, substantial part of technical writing

[58] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. Refining the undecidability border of weak bisimilarity. In *Proceedings of INFINITY 2005*, volume 149 of *Electronic Notes in Theoretical Computer Science*, pages 17–36. Elsevier Science Publishers, 2006.

- Author's contribution: 33%, participating on all phases

[10] Ahmed Bouajjani, Jan Strejček, and Tayssir Touili. On symbolic verification of weakly extended PAD. In *Proceedings of EXPRESS 2006*, volume 175(3) of *Electronic Notes in Theoretical Computer Science*, pages 47–64. Elsevier Science Publishers, 2007.

- Author's contribution: 33%, bringing the topic, participating on all phases

[1] Tomáš Babiak, Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. LTL to Büchi automata translation: Fast and more deterministic. In *Proceedings of TACAS 2012*, volume 7214 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2012.

- Author's contribution: 25%, bringing the topic and substantial part of ideas, significant part of writing

Bibliography

[1] Tomáš Babiak, Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. LTL to Büchi automata translation: Fast and more deterministic. In *Proceedings of TACAS 2012*, volume 7214 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2012.

[2] Tomáš Babiak, Vojtěch Řehák, and Jan Strejček. Almost linear Büchi automata. In *Proceedings of EXPRESS 2009*, volume 8 of *Electronic Proceedings in Theoretical Computer Science*, pages 16–25, 2009.

[3] Tomáš Babiak, Vojtěch Řehák, and Jan Strejček. Almost linear Büchi automata. *Mathematical Structures in Computer Science*, 22(02):203–235, 2012.

[4] Jan A. Bergstra and Jan Willem Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.

[5] Ahmed Bouajjani, Rachid Echahed, and Peter Habermehl. On the verification problem of nonregular properties for nonregular processes. In *Proceedings of LICS'95*. IEEE Computer Society Press, 1995.

[6] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability Analysis of Pushdown Automata: Application to Model-Checking. In *Proceedings of CONCUR'97*, volume 1243 of *Lectute Notes in Computer Science*, pages 135–150, 1997.

[7] Ahmed Bouajjani, Javier Esparza, Stefan Schwoon, and Jan Strejček. Reachability analysis of multithreaded software with asynchronous communication. In *Proceedings of FSTTCS 2005*, volume 3821 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2005.

[8] Ahmed Bouajjani, Javier Esparza, and Tayssir Touili. Reachability analysis of synchronized PA systems. In *Proceedings of INFINITY 2004*, volume 138 (3) of *Electronic Notes in Theoretical Computer Science*, pages 153–178, 2005.

[9] Ahmed Bouajjani and Peter Habermehl. Constrained Properties, Semilinear Systems, and Petri Nets. In *Proceedings of CONCUR'96*, volume 1119 of *Lectute Notes in Computer Science*, pages 481–497. Springer, 1996.

[10] Ahmed Bouajjani, Jan Strejček, and Tayssir Touili. On symbolic verification of weakly extended PAD. In *Proceedings of EXPRESS 2006*, volume 175(3) of *Electronic Notes in Theoretical Computer Science*, pages 47–64. Elsevier Science Publishers, 2007.

[11] Ahmed Bouajjani and Tayssir Touili. Reachability Analysis of Process Rewrite Systems. In *Proceedings of FSTTCS 2003*, volume 2914 of *Lectute Notes in Computer Science*, pages 74–87. Springer, 2003.

[12] Ahmed Bouajjani and Tayssir Touili. On computing reachability sets of process rewrite systems. In *Proceedings of RTA 2005*, volume 3467 of *Lectute Notes in Computer Science*, pages 484–499. Springer, 2005.

[13] Laura Bozzelli. Model checking for process rewrite systems and a class of action-based regular properties. In *Proceedings of VMCAI'05*, volume 3385 of *Lectute Notes in Computer Science*, pages 282–297. Springer, 2005.

[14] Laura Bozzelli, Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. On decidability of LTL model checking for process rewrite systems. In *Proceedings of FSTTCS 2006*, volume 4337 of *Lecture Notes in Computer Science*, pages 248–259. Springer, 2006.

[15] Laura Bozzelli, Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. On decidability of LTL model checking for process rewrite systems. *Acta Informatica*, 46(1):1–28, 2009.

[16] Julius Richard Büchi. Regular canonical systems. *Archiv für Mathematische Logik und Grundlagenforschung*, 6:91–111, 1964.

[17] Olaf Burkart, Didier Caucal, Faron Moller, and Bernhard Steffen. Verification on infinite structures. In *Handbook of Process Algebra*, pages 545–623. Elsevier, 2001.

[18] Olaf Burkart, Didier Caucal, and Bernhard Steffen. Bisimulation collapse and the process taxonomy. In *Proceedings of CONCUR'96*, volume 1119 of *Lectute Notes in Computer Science*, pages 247–262. Springer, 1996.

[19] Olaf Burkart and Bernhard Steffen. Model checking the full modal mu-calculus for infinite sequential processes. *Theoretical Computer Science*, 221(1-2):251–270, 1999.

[20] Didier Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.

[21] Søren Christensen. *Decidability and Decomposition in Process Algebras*. PhD thesis, Department of Computer Science, University of Edinburgh, 1993.

[22] Søren Christensen, Yoram Hirshfeld, and Faron Moller. Bisimulation is decidable for all basic parallel processes. In *Proceedings of CONCUR'93*, volume 715 of *Lectute Notes in Computer Science*, pages 143–157. Springer, 1993.

[23] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of IBM Workshop on Logic of Programs*, volume 131 of *Lectute Notes in Computer Science*, pages 52–71. Springer, 1981.

[24] Jean-Michel Couvreur. On-the-fly verification of temporal logic. In *Proceedings of FM'99*, volume 1708 of *Lectute Notes in Computer Science*, pages 253–271. Springer, 1999.

[25] Marco Daniele, Fausto Giunchiglia, and Moshe Y. Vardi. Improved automata generation for linear temporal logic. In *Proceedings of CAV'99*, volume 1633 of *Lectute Notes in Computer Science*, pages 249–260. Springer, 1999.

[26] Christian Dax, Jochen Eisinger, and Felix Klaedtke. Mechanizing the powerset construction for restricted classes of ω-automata. In *Proceedings of ATVA 2007*, volume 4762 of *Lectute Notes in Computer Science*, pages 223–236. Springer, 2007.

[27] Alexandre Duret-Lutz. LTL translation improvements in Spot. In *Proceedings of VECoS 2011*, Electronic Workshops in Computing. British Computer Society, 2011.

[28] Alexandre Duret-Lutz and Denis Poitrenaud. SPOT: An extensible model checking library using transition-based generalized Büchi automata. In *Proceedings of MASCOTS 2004*, pages 76–83. IEEE Computer Society Press, 2004.

[29] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Property specification patterns for finite-state verification. In *Proceedings of FMSP-98*, pages 7–15, New York, 1998. ACM Press.

[30] Rüdiger Ehlers and Bernd Finkbeiner. On the virtue of patience: Minimizing Büchi automata. In *Proceedings of SPIN 2010*, volume 6349 of *Lectute Notes in Computer Science*, pages 129–145. Springer, 2010.

[31] Javier Esparza. On the Decidability of Model Checking for Several mu-calculi and Petri Nets. In *Proceedings of CAAP'94*, volume 787 of *Lectute Notes in Computer Science*, pages 115–129. Springer, 1994.

[32] Javier Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundamenta Informaticae*, 31(1):13–25, 1997.

[33] Javier Esparza. Grammars as Processes. In *Formal and Natural Computing*, volume 2300 of *Lectute Notes in Computer Science*, pages 277–297. Springer, 2002.

[34] Javier Esparza, David Hansel, Peter Rossmanith, and Stefan Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of CAV 2000*, volume 1855 of *Lectute Notes in Computer Science*, pages 232–247, 2000.

[35] Javier Esparza and Astrid Kiehn. On the model checking problem for branching time logics and basic parallel processes. In *Proceedings of CAV'95*, volume 939 of *Lectute Notes in Computer Science*, pages 353–366. Springer, 1995.

[36] Javier Esparza and Andreas Podelski. Efficient algorithms for pre* and post* on interprocedural parallel flow graphs. In *Proceedings of POLP 2000*, pages 1–11. ACM Press, 2000.

[37] Kousha Etessami and Gerard J. Holzmann. Optimizing Büchi Automata. In Catuscia Palamidessi, editor, *Proceedings of CONCUR 2000*, volume 1877 of *Lectute Notes in Computer Science*, pages 153–167. Springer, 2000.

[38] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Information and Computation*, 179(2):279–295, 2002.

[39] Carsten Fritz. Constructing Büchi automata from linear temporal logic using simulation relations for alternating Büchi automata. In *Proceedings of CIAA 2003*, volume 2759 of *Lectute Notes in Computer Science*, pages 35–48. Springer, 2003.

[40] Paul Gastin and Denis Oddoux. Fast LTL to Büchi Automata Translation. In *Proceedings of CAV 2001*, volume 2102 of *Lectute Notes in Computer Science*, pages 53–65. Springer, 2001.

[41] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification Testing and Verification*, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1995.

[42] Dimitra Giannakopoulou and Flavio Lerda. From states to transitions: Improving translation of ltl formulae to Büchi automata. In Doron Peled and Moshe Vardi, editors, *Proceedings of FORTE 2002*, volume 2529 of *Lectute Notes in Computer Science*, pages 308–326. Springer, 2002.

[43] Yoram Hirshfeld. Bisimulation trees and the decidability of weak bisimulations. *Electronic Notes in Theoretical Computer Science*, 5:2–13, 1996.

[44] Yoram Hirshfeld and Faron Moller. Pushdown automata, multiset automata, and Petri nets. *Theoretical Computer Science*, 256(1-2):3–21, 2001.

[45] Charles Antony Richard Hoare. Communicating Sequential Processes. In *On the construction of programs – an advanced course*, pages 229–254. Cambrigde University Press, 1980.

[46] Hans Hüttel and Jiří Srba. Recursion vs. replication in simple cryptographic protocols. In *Proceedings of SOFSEM 2005: Theory and Practice of Computer Science*, volume 3381 of *Lectute Notes in Computer Science*, pages 178–187. Springer, 2005.

[47] Hans Hüttel and Jiří Srba. Decidability issues for extended ping-pong protocols. *Journal of Automated Reasoning*, 36(1–2):125–147, 2006.

[48] Petr Jančar. High Undecidability of Weak Bisimilarity for Petri Nets. In *Proceedings of TAPSOFT'95*, volume 915 of *Lectute Notes in Computer Science*, pages 349–363. Springer, 1995.

[49] Petr Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.

[50] Petr Jančar. Strong bisimilarity on basic parallel processes is PSPACE-complete. In *Proceedings of LICS 2003*, pages 218–227. IEEE Computer Society, 2003.

[51] Petr Jančar, Antonín Kučera, and Richard Mayr. Deciding bisimulation-like equivalences with finite-state processes. *Theoretical Computer Science*, 258:409–433, 2001.

[52] Petr Jančar and Faron Moller. Checking regular properties of Petri nets. In *Proceedings of CONCUR'95*, volume 962 of *Lectute Notes in Computer Science*, pages 348–362. Springer, 1995.

[53] Antonín Kučera and Petr Jančar. Equivalence-checking on infinite-state systems: Techniques and results. *Theory and Practice of Logic Programming*, 6(3):227–264, 2006.

[54] Antonín Kučera and Philippe Schnoebelen. A general approach to comparing infinite-state systems with their finite-state specifications. *Theoretical Computer Science*, 358(2-3):315–333, 2006.

[55] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. Extended process rewrite systems: Expressiveness and reachability. In *Proceedings of CONCUR 2004*, volume 3170 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2004.

[56] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. On extensions of process rewrite systems: Rewrite systems with weak finite-state unit. In *Proceedings of INFINITY 2003*, volume 98 of *Electronic Notes in Theoretical Computer Science*, pages 75–88. Elsevier Science Publishers, 2004.

[57] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. Reachability of Hennessy-Milner properties for weakly extended PRS. In *Proceedings of FSTTCS 2005*, volume 3821 of *Lecture Notes in Computer Science*, pages 213–224. Springer, 2005.

[58] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. Refining the undecidability border of weak bisimilarity. In *Proceedings of INFINITY 2005*, volume 149 of *Electronic Notes in Theoretical Computer Science*, pages 17–36. Elsevier Science Publishers, 2006.

[59] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. Petri nets are less expressive than state-extended PA. *Theoretical Computer Science*, 394(1–2):134–140, 2008.

[60] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. On decidability of LTL+past model checking for process rewrite systems. In *Joint Proceedings of INFINITY 2006, 2007, 2008*, volume 239 of *Electronic Notes in Theoretical Computer Science*, pages 105–117. Elsevier Science Publishers, 2009.

[61] Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. Reachability is decidable for weakly extended process rewrite systems. *Information and Computation*, 207(6):671–680, 2009.

[62] Christof Löding. Efficient minimization of deterministic weak omega-automata. *Information Processing Letters*, 79(3):105–109, 2001.

[63] Christof Löding. *Infinite Graphs Generated by Tree Rewriting*. PhD thesis, RWTH Aachen, 2003.

[64] Denis Lugiez and Philippe Schnoebelen. The regular viewpoint on PA-processes. In *Proceedings of CONCUR'98*, volume 1466 of *Lectute Notes in Computer Science*, pages 50–66. Springer, 1998.

[65] Monika Maidl. The common fragment of CTL and LTL. In *Proceedings of FOCS 2000*, pages 643–652. IEEE Computer Society, 2000.

[66] Zohar Manna and Amir Pnueli. A hierarchy of temporal properties. In *Proceedings of PODC'90*, pages 377–410. ACM press, 1990.

[67] Ernst W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal on Computing*, 13(3):441–460, 1984.

[68] Richard Mayr. Combining Petri nets and PA-processes. In *Proceedings of TACS '97*, volume 1281 of *Lectute Notes in Computer Science*, pages 547–561. Springer, 1997.

[69] Richard Mayr. Process rewrite systems. In *Proceedings of EXPRESS'97*, volume 7 of *Electronic Notes in Theoretical Computer Science*, pages 185–205, 1997.

[70] Richard Mayr. *Decidability and Complexity of Model Checking Problems for Infinite-State Systems*. PhD thesis, Technische Universität München, 1998.

[71] Richard Mayr. Process rewrite systems. *Information and Computation*, 156(1):264–286, 2000.

[72] Richard Mayr. Weak bisimilarity and regularity of context-free processes is EXPTIME-hard. *Theoretical Computer Science*, 330(3):553–575, 2005.

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

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

[75] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[76] Faron Moller. Infinite results. In *Proceedings of CONCUR'96*, volume 1119 of *Lectute Notes in Computer Science*, pages 195–216. Springer, 1996.

[77] Faron Moller. Pushdown Automata, Multiset Automata and Petri Nets. In *Proceedings of MFCS Workshop on concurrency*, volume 18, 1998.

[78] David E. Muller and Paul E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.

[79] David Michael Ritchie Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Theoretical Computer Science: 5th GI-Conference*, volume 104 of *Lectute Notes in Computer Science*, pages 167–183. Springer, 1981.

[80] Radek Pelánek. BEEM: Benchmarks for explicit model checkers. In *Proceedings of SPIN 2007*, volume 4595 of *Lectute Notes in Computer Science*, pages 263–267. Springer, 2007.

[81] Amir Pnueli. The Temporal Logic of Programs. In *Proceedings of FOCS'77*, pages 46–57. IEEE Computer Society, 1977.

[82] Shaz Qadeer and Jakob Rehof. Context-bounded model checking of concurrent software. In *Proceedings of TACAS 2005*, LNCS 3440, pages 93–107, 2005.

[83] Kristin Y. Rozier and Moshe Y. Vardi. LTL Satisfiability Checking. In *Proceedings of SPIN 2007*, volume 4595 of *Lectute Notes in Computer Science*, pages 149–167. Springer, 2007.

[84] Vijay A. Saraswat. *Concurrent constraint programming*. MIT Press, 1993.

[85] Roberto Sebastiani and Stefano Tonetta. "More Deterministic" vs. "Smaller" Büchi Automata for Efficient LTL Model Checking. In *Proceedings of CHARME 2003*, volume 2860 of *Lectute Notes in Computer Science*, pages 126–140. Springer, 2003.

[86] Géraud Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proceedings of FOCS'98*, pages 120–129. IEEE Computer Society, 1998.

[87] Fabio Somenzi and Roderick Bloem. Efficient Büchi Automata from LTL Formulae. In *Proceedings of CAV 2000*, volume 1855 of *Lectute Notes in Computer Science*, pages 248–263. Springer, 2000.

[88] Jiří Srba. Undecidability of weak bisimilarity for pushdown processes. In *Proceedings of CONCUR 2002*, volume 2421 of *Lectute Notes in Computer Science*, pages 579–593. Springer, 2002.

[89] Jiří Srba. Undecidability of weak bisimilarity for PA-processes. In *Proceedings of DLT 2002*, volume 2450 of *Lectute Notes in Computer Science*, pages 197–208. Springer, 2003.

[90] Jiří Srba. Roadmap of infinite results. Online version avaiable at `http://cs.au.dk/~srba/roadmap/`, 2006.

[91] Colin Stirling. Decidability of weak bisimilarity for a subset of basic parallel processes. In *Proceedings of FoSSaCS 2001*, volume 2030 of *Lectute Notes in Computer Science*, pages 379–393. Springer, 2001.

[92] Jan Strejček. Rewrite systems with constraints. In *Proceedings of EXPRESS 2001*, volume 52 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2002.

[93] Jan Strejček. *Linear Temporal Logic: Expressiveness and Model Checking*. PhD thesis, Faculty of Informatics, Masaryk University, 2004.

[94] Anthony Widjaja To. *Model Checking Infinite-State Systems: Generic and Specific Approaches*. PhD thesis, School of Informatics, University of Edinburgh, 2010.

[95] Anthony Widjaja To and Leonid Libkin. Algorithmic metatheorems for decidable ltl model checking over infinite systems. In *Proceedings of FOSSACS 2010*, volume 6014 of *Lectute Notes in Computer Science*, pages 221–236. Springer, 2010.

[96] Rob J. van Glabbeek. The Linear time – Branching Time Spectrum II. In *Proceedings of CONCUR'93*, volume 715 of *Lectute Notes in Computer Science*, pages 66–81. Springer, 1993.

[97] Vojtěch Řehák. *On Extensions of Process Rewrite Systems*. PhD thesis, Faculty of Informatics, Masaryk University Brno, 2007.

[98] Igor Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.