




The Complexity of Deciding Characteristic Formulae in Van Glabbeek’s Branching-Time Spectrum

Luca Aceto   

Department of Computer Science, Reykjavik University, Iceland
Gran Sasso Science Institute, L’Aquila, Italy

Antonis Achilleos   

Department of Computer Science, Reykjavik University, Iceland

Aggeliki Chalki   

Department of Computer Science, Reykjavik University, Iceland

Anna Ingólfssdóttir   

Department of Computer Science, Reykjavik University, Iceland

Abstract

Characteristic formulae give a complete logical description of the behaviour of processes modulo some chosen notion of behavioural semantics. They allow one to reduce equivalence or preorder checking to model checking, and are exactly the formulae in the modal logics characterizing classic behavioural equivalences and preorders for which model checking can be reduced to equivalence or preorder checking.

This paper studies the complexity of determining whether a formula is characteristic for some process in each of the logics providing modal characterizations of the simulation-based semantics in van Glabbeek’s branching-time spectrum. Since characteristic formulae in each of those logics are exactly the satisfiable and prime ones, this article presents complexity results for the satisfiability and primality problems, and investigates the boundary between modal logics for which those problems can be solved in polynomial time and those for which they become computationally hard.

Amongst other contributions, this article also studies the complexity of constructing characteristic formulae in the modal logics characterizing simulation-based semantics, both when such formulae are presented in explicit form and via systems of equations.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Complexity theory and logic

Keywords and phrases Characteristic formulae, prime formulae, bisimulation, simulation relations, modal logics, complexity theory, satisfiability

Digital Object Identifier 10.4230/LIPIcs.CSL.2025.26

Related Version *Full Version:* <https://doi.org/10.48550/arXiv.2405.13697> [1]

Funding This work has been funded by the projects “Open Problems in the Equational Logic of Processes (OPEL)” (grant no. 196050), “Mode(1)s of Verification and Monitorability” (MoVeMnt) (grant no. 217987), and “Learning and Applying Probabilistic Systems” (grant no. 206574-051) of the Icelandic Research Fund.

Acknowledgements The authors thank the anonymous reviewers for comments that led to improvements in the paper. This paper is dedicated to the memory of Rance Cleaveland (1961–2024), who used characteristic formulae to compute behavioural relations, logically and efficiently.



© Luca Aceto, Antonis Achilleos, Aggeliki Chalki, and Anna Ingólfssdóttir;
licensed under Creative Commons License CC-BY 4.0

33rd EACSL Annual Conference on Computer Science Logic (CSL 2025).

Editors: Jörg Endrullis and Sylvain Schmitz; Article No. 26; pp. 26:1–26:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Several notions of behavioural relations have been proposed in concurrency theory to describe when one process is a suitable implementation of another. Many such relations have been catalogued by van Glabbeek in his seminal linear-time/branching-time spectrum [22], together with a variety of alternative ways of describing them including testing scenarios and axiom systems. To our mind, modal characterizations of behavioural equivalences and preorders are some of the most classic and pleasing results in concurrency theory – see, for instance, [25] for the seminal Hennessy-Milner theorem and [12, 16, 17, 22] for similar results for other relations in van Glabbeek’s spectrum and other settings. By way of example, in their archetypal modal characterization of bisimilarity, Hennessy and Milner have shown in [25] that, under a mild finiteness condition, two processes are bisimilar if, and only if, they satisfy the same formulae in a multi-modal logic that is now often called Hennessy-Milner logic. Apart from its intrinsic theoretical interest, this seminal logical characterization of bisimilarity means that, when two processes are *not* bisimilar, there is always a formula that distinguishes between them. Such a formula describes a reason why the two processes are not bisimilar, provides useful debugging information and can be algorithmically constructed over finite processes – see, for instance, [8, 14] and [35], where Martens and Groote show that, in general, computing minimal distinguishing Hennessy-Milner formulae is NP-hard.

On the other hand, the Hennessy-Milner theorem seems to be less useful to show that two processes *are* bisimilar, since that would involve verifying that they satisfy the same formulae, and there are infinitely many of those. However, as shown in works such as [3, 6, 12, 23, 39], the logics that underlie classic modal characterization theorems for equivalences and preorders over processes allow one to express *characteristic formulae*. Intuitively, a characteristic formula $\chi(p)$ for a process p gives a complete logical characterization of the behaviour of p modulo the behavioural semantics of interest \lesssim , in the sense that any process is related to p with respect to \lesssim if, and only if, it satisfies $\chi(p)$.¹ Since the formula $\chi(p)$ can be constructed from p , characteristic formulae reduce the problem of checking whether a process q is related to p by \lesssim to a model checking problem, viz. whether q satisfies $\chi(p)$. See, for instance, the classic reference [15] for applications of this approach.

Characteristic formulae, thus, allow one to reduce equivalence and preorder checking to model checking. But what model checking problems can be reduced to equivalence/preorder checking ones? To the best of our knowledge, that question was first studied by Boudol and Larsen in [11] in the setting of modal refinement over modal transition systems. See [3, 4] for other contributions in that line of research. The aforementioned articles showed that characteristic formulae coincide with those that are *satisfiable* and *prime*. (A formula is prime if whenever it entails a disjunction $\varphi_1 \vee \varphi_2$, then it must entail φ_1 or φ_2 .) Moreover, characteristic formulae with respect to bisimilarity coincide with the formulae that are satisfiable and *complete* [7]. (A modal formula is complete if, for each formula φ , it entails either φ or its negation.) The aforementioned results give semantic characterizations of the formulae that are characteristic within the logics that correspond to the behavioural semantics in van Glabbeek’s spectrum. Those characterizations tell us for what logical specifications model checking can be reduced to equivalence or preorder checking. However,

¹ Formulae akin to characteristic ones first occurred in the study of equivalence of structures using first-order formulae up to some quantifier rank. See, for example, the survey paper [40] and the textbook [20]. The existence of formulae in first-order logic with counting that characterize graphs up to isomorphism has significantly contributed to the study of the complexity of the Graph Isomorphism problem – see, for instance, [13, 30].

given a specification expressed as a modal formula, can one decide whether that formula is characteristic and therefore can be model checked using algorithms for behavioural equivalences or preorders? And, if so, what is the complexity of checking whether a formula is characteristic? Perhaps surprisingly, those questions were not addressed in the literature until the recent papers [2, 7], where it is shown that, in the setting of the modal logics that characterize bisimilarity over natural classes of Kripke structures and labelled transition systems, the problem of checking whether a formula is characteristic for some process modulo bisimilarity is computationally hard and, typically, has the same complexity as validity checking, which is PSPACE-complete for Hennessy-Milner logic and EXP-complete for its extension with fixed-point operators [26, 33] and the μ -calculus [31].

The aforementioned hardness results for the logics characterizing bisimilarity tell us that deciding whether a formula is characteristic in bisimulation semantics is computationally hard. But what about the less expressive logics that characterize the coarser semantics in van Glabbeek's spectrum? And for what logics characterizing relations in the spectrum does computational hardness manifest itself? Finally, what is the complexity of computing a characteristic formula for a process?

The aim of this paper is to answer the aforementioned questions for some of the simulation-based semantics in the spectrum. In particular, we study the complexity of determining whether a formula is characteristic modulo the simulation [36], complete simulation and ready simulation preorders [10, 34], as well as the trace simulation and the n -nested simulation preorders [24]. Since characteristic formulae are exactly the satisfiable and prime ones for each behavioural relation in van Glabbeek's spectrum [3], the above-mentioned tasks naturally break down into studying the complexity of satisfiability and primality checking for formulae in the fragments of Hennessy-Milner logic that characterize those preorders. By using a reduction to the, seemingly unrelated, reachability problem in *alternating graphs*, as defined by Immerman in [28, Definition 3.24], we discover that both those problems are decidable in polynomial time for the simulation and the complete simulation preorders, as well as for the ready simulation preorder when the set of actions has constant size. On the other hand, when the set of actions is unbounded (that is, it is an input of the algorithmic problem at hand), the problems of checking satisfiability and primality for formulae in the logic characterizing the ready simulation preorder are NP-complete and coNP-complete respectively. We also show that deciding whether a formula is characteristic in that setting is US-hard [9] (that is, it is at least as hard as the problem of deciding whether a given Boolean formula has exactly one satisfying truth assignment) and belongs to DP, which is the class of languages that are the intersection of one language in NP and of one in coNP [38].² These negative results are in stark contrast with the positive results for the simulation and the complete simulation preorder, and indicate that augmenting the logic characterizing the simulation preorder with formulae that state that a process cannot perform a given action suffices to make satisfiability and primality checking computationally hard. In passing, we also prove that, in the presence of at least two actions, (1) for the logics characterizing the trace simulation and 2-nested simulation preorders, satisfiability and primality checking are NP-complete and coNP-hard respectively, and deciding whether a formula is characteristic is US-hard, (2) for the logic that characterizes the trace simulation preorder, deciding whether a formula is characteristic is fixed-parameter tractable [18], with the modal depth of the input formula as the parameter, when the size of the action set is a constant, and (3) deciding whether

² The class DP contains both NP and coNP, and is contained in the class of problems that can be solved in polynomial time with an NP oracle.

a formula is characteristic in the modal logic for the 3-nested simulation preorder [24] is PSPACE-hard. (The proof of the last result relies on “simulating” Ladner’s reduction proving the PSPACE-hardness of satisfiability for modal logic [32] using the limited alternations of modal operators allowed by the logic for the 3-nested simulation preorder.)

We also study the complexity of computing characteristic formulae for finite, loop-free processes modulo the above-mentioned simulation semantics. To do so, we consider two different representations for formulae, namely an explicit form, where formulae are given by strings of symbols generated by their respective grammars, and a declarative form, where formulae are described by systems of equations. We prove that, even for the coarsest semantics we consider, such as the simulation and complete simulation preorders, computing the characteristic formula in explicit form for a finite, loop-free process cannot be done in polynomial time, unless $P = NP$. On the other hand, the characteristic formula for a process modulo the preorders we study, apart from the trace simulation preorder, can be computed in polynomial time if the output is given in declarative form. Intuitively, this is due to the fact that, unlike the explicit form, systems of equations allow for sharing of subformulae and there are formulae for which this sharing leads to an exponentially more concise representation. Finally, in sharp contrast to that result, we prove that, modulo the trace simulation preorder, even if characteristic formulae are always of polynomial declaration size and polynomial equational length, they cannot be efficiently computed unless $P = NP$. In passing, we remark that all the aforementioned lower and upper bounds hold also for finite processes with loops, provided that, as done in [6, 29, 39], we add greatest fixed points or systems of equations interpreted as greatest fixed points to the modal logics characterizing the semantics we study in this article.

We summarize our results in Table 2. We provide their proofs in the technical appendices of the full version of the paper [1].

2 Preliminaries

In this paper, we model processes as finite, loop-free *labelled transition systems* (LTS). A finite LTS is a triple $\mathcal{S} = (P, A, \longrightarrow)$, where P is a finite set of states (or processes), A is a finite, non-empty set of actions and $\longrightarrow \subseteq P \times A \times P$ is a transition relation. As usual, we use $p \xrightarrow{a} q$ instead of $(p, a, q) \in \longrightarrow$. For each $t \in A^*$, we write $p \xrightarrow{t} q$ to mean that there is a sequence of transitions labelled with t starting from p and ending at q . An LTS is *loop-free* iff $p \xrightarrow{t} p$ holds only when t is the empty trace ε . A process q is *reachable* from p if $p \xrightarrow{t} q$, for some $t \in A^*$. We define the *size* of an LTS $\mathcal{S} = (P, A, \longrightarrow)$, denoted by $|\mathcal{S}|$, to be $|P| + |\longrightarrow|$. The *size of a process* $p \in P$, denoted by $|p|$, is the cardinality of $\text{reach}(p) = \{q \mid q \text{ is reachable from } p\}$ plus the cardinality of the set \longrightarrow restricted to $\text{reach}(p)$. We define the set of *initials* of p , denoted $I(p)$, as the set $\{a \in A \mid p \xrightarrow{a} p' \text{ for some } p' \in P\}$. We write $p \xrightarrow{a}$ if $a \in I(p)$, $p \not\xrightarrow{a}$ if $a \notin I(p)$, and $p \not\rightarrow$ if $I(p) = \emptyset$. A sequence of actions $t \in A^*$ is a *trace* of p if there is a q such that $p \xrightarrow{t} q$. We denote the set of traces of p by $\text{traces}(p)$. The *depth* of a finite, loop-free process p , denoted by $\text{depth}(p)$, is the length of a longest trace t of p .

In what follows, we shall often describe finite, loop-free processes using the fragment of Milner’s CCS [37] given by the following grammar:

$$p ::= 0 \mid a.p \mid p + p,$$

where $a \in A$. For each action a and terms p, p' , we write $p \xrightarrow{a} p'$ iff

- (i) $p = a.p'$ or
- (ii) $p = p_1 + p_2$, for some p_1, p_2 , and $p_1 \xrightarrow{a} p'$ or $p_2 \xrightarrow{a} p'$ holds.

In this paper, we consider the following relations in van Glabbeek's spectrum: simulation, complete simulation, ready simulation, trace simulation, 2-nested simulation, 3-nested simulation, and bisimilarity. Their definitions are given below.

► **Definition 1** ([37, 22, 3]). *We define each of the following preorders as the largest binary relation over P that satisfies the corresponding condition.*

- (a) Simulation preorder (S): $p \lesssim_S q \Leftrightarrow$ for all $p \xrightarrow{a} p'$ there exists some $q \xrightarrow{a} q'$ such that $p' \lesssim_S q'$.
- (b) Complete simulation (CS): $p \lesssim_{CS} q \Leftrightarrow$
 - (i) for all $p \xrightarrow{a} p'$ there exists some $q \xrightarrow{a} q'$ such that $p' \lesssim_{CS} q'$, and
 - (ii) $I(p) = \emptyset$ iff $I(q) = \emptyset$.
- (c) Ready simulation (RS): $p \lesssim_{RS} q \Leftrightarrow$
 - (i) for all $p \xrightarrow{a} p'$ there exists some $q \xrightarrow{a} q'$ such that $p' \lesssim_{RS} q'$, and
 - (ii) $I(p) = I(q)$.
- (d) Trace simulation (TS): $p \lesssim_{TS} q \Leftrightarrow$
 - (i) for all $p \xrightarrow{a} p'$ there exists some $q \xrightarrow{a} q'$ such that $p' \lesssim_{TS} q'$, and
 - (ii) $\text{traces}(p) = \text{traces}(q)$.
- (e) n -Nested simulation (nS), where $n \geq 1$, is defined inductively as follows: The 1-nested simulation preorder \lesssim_{1S} is \lesssim_S , and the n -nested simulation preorder \lesssim_{nS} for $n > 1$ is the largest relation such that $p \lesssim_{nS} q \Leftrightarrow$
 - (i) for all $p \xrightarrow{a} p'$ there exists some $q \xrightarrow{a} q'$ such that $p' \lesssim_{nS} q'$, and
 - (ii) $q \lesssim_{(n-1)S} p$.
- (f) Bisimilarity (BS): \lesssim_{BS} is the largest symmetric relation satisfying the condition defining the simulation preorder.

It is well-known that bisimilarity is an equivalence relation and all the other relations are preorders [22, 37]. We sometimes write $p \sim q$ instead of $p \lesssim_{BS}$. Moreover, we have that $\sim \subsetneq \lesssim_{3S} \subsetneq \lesssim_{2S} \subsetneq \lesssim_{TS} \subsetneq \lesssim_{RS} \subsetneq \lesssim_{CS} \subsetneq \lesssim_S$ – see [22].

► **Definition 2** (Kernels of the preorders). *For each $X \in \{S, CS, RS, TS, 2S, 3S\}$, the kernel \equiv_X of \lesssim_X is the equivalence relation defined thus: for every $p, q \in P$, $p \equiv_X q$ iff $p \lesssim_X q$ and $q \lesssim_X p$.*

Each relation \lesssim_X , where $X \in \{S, CS, RS, TS, 2S, 3S, BS\}$, is characterized by a fragment \mathcal{L}_X of Hennessy-Milner logic, **HML**, defined as follows [22, 3].

► **Definition 3.** *For $X \in \{S, CS, RS, TS, 2S, 3S, BS\}$, \mathcal{L}_X is defined by the corresponding grammar given below ($a \in A$):*

- (a) \mathcal{L}_S : $\varphi_S ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi_S \wedge \varphi_S \mid \varphi_S \vee \varphi_S \mid \langle a \rangle \varphi_S$.
- (b) \mathcal{L}_{CS} : $\varphi_{CS} ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi_{CS} \wedge \varphi_{CS} \mid \varphi_{CS} \vee \varphi_{CS} \mid \langle a \rangle \varphi_{CS} \mid \mathbf{0}$, where $\mathbf{0} = \bigwedge_{a \in A} [a] \mathbf{ff}$.
- (c) \mathcal{L}_{RS} : $\varphi_{RS} ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi_{RS} \wedge \varphi_{RS} \mid \varphi_{RS} \vee \varphi_{RS} \mid \langle a \rangle \varphi_{RS} \mid [a] \mathbf{ff}$.
- (d) \mathcal{L}_{TS} : $\varphi_{TS} ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi_{TS} \wedge \varphi_{TS} \mid \varphi_{TS} \vee \varphi_{TS} \mid \langle a \rangle \varphi_{TS} \mid \psi_{TS}$, where $\psi_{TS} ::= \mathbf{ff} \mid [a] \psi_{TS}$.
- (e) \mathcal{L}_{2S} : $\varphi_{2S} ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi_{2S} \wedge \varphi_{2S} \mid \varphi_{2S} \vee \varphi_{2S} \mid \langle a \rangle \varphi_{2S} \mid \neg \varphi_S$.
- (f) \mathcal{L}_{3S} : $\varphi_{3S} ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi_{3S} \wedge \varphi_{3S} \mid \varphi_{3S} \vee \varphi_{3S} \mid \langle a \rangle \varphi_{3S} \mid \neg \varphi_{2S}$.
- (g) **HML** (\mathcal{L}_{BS}): $\varphi_{BS} ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi_{BS} \wedge \varphi_{BS} \mid \varphi_{BS} \vee \varphi_{BS} \mid \langle a \rangle \varphi_{BS} \mid [a] \varphi_{BS} \mid \neg \varphi_{BS}$.

Note that the explicit use of negation in the grammar for \mathcal{L}_{BS} is unnecessary. However, we included the negation operator explicitly so that \mathcal{L}_{BS} extends syntactically each of the other modal logics presented in Definition 3.

26:6 The Complexity of Deciding Characteristic Formulae

Given a formula $\varphi \in \mathcal{L}_{BS}$, the *modal depth* of φ , denoted by $\text{md}(\varphi)$, is the maximum nesting of modal operators in φ . (See [1, Appendix A] for the formal definition.)

Truth in an LTS $\mathcal{S} = (P, A, \longrightarrow)$ is defined via the satisfaction relation \models as follows:

- $p \models \mathbf{tt}$ and $p \not\models \mathbf{ff}$;
- $p \models \neg\varphi$ iff $p \not\models \varphi$;
- $p \models \varphi \wedge \psi$ iff both $p \models \varphi$ and $p \models \psi$;
- $p \models \varphi \vee \psi$ iff $p \models \varphi$ or $p \models \psi$;
- $p \models \langle a \rangle \varphi$ iff there is some $p \xrightarrow{a} q$ such that $q \models \varphi$;
- $p \models [a]\varphi$ iff for all $p \xrightarrow{a} q$ it holds that $q \models \varphi$.

If $p \models \varphi$, we say that φ is true, or satisfied, in p . If φ is satisfied in every process in every LTS, we say that φ is valid. Formula φ_1 entails φ_2 , denoted by $\varphi_1 \models \varphi_2$, if every process that satisfies φ_1 also satisfies φ_2 . Moreover, φ_1 and φ_2 are logically equivalent, denoted by $\varphi_1 \equiv \varphi_2$, if $\varphi_1 \models \varphi_2$ and $\varphi_2 \models \varphi_1$. A formula φ is *satisfiable* if there is a process that satisfies φ . Finally, $\text{Sub}(\varphi)$ denotes the set of subformulae of formula φ .

For $\mathcal{L} \subseteq \mathcal{L}_{BS}$, we define the dual fragment of \mathcal{L} to be $\bar{\mathcal{L}} = \{\varphi \mid \neg\varphi \in \mathcal{L}\}$, where $\neg\mathbf{tt} = \mathbf{ff}$, $\neg\mathbf{ff} = \mathbf{tt}$, $\neg(\varphi \wedge \psi) = \neg\varphi \vee \neg\psi$, $\neg(\varphi \vee \psi) = \neg\varphi \wedge \neg\psi$, $\neg[a]\varphi = \langle a \rangle \neg\varphi$, $\neg\langle a \rangle \varphi = [a]\neg\varphi$, and $\neg\neg\varphi = \varphi$. It is not hard to see that $p \models \neg\varphi$ iff $p \not\models \varphi$, for every process p . Given a process p , we define $\mathcal{L}(p) = \{\varphi \in \mathcal{L} \mid p \models \varphi\}$. A simplification of the Hennessy-Milner theorem gives a modal characterization of bisimilarity over finite processes. An analogous result is true for every preorder examined in this paper.

► **Theorem 4** (Hennessy-Milner theorem [25]). *For all processes p, q in a finite LTS, $p \sim q$ iff $\mathcal{L}_{BS}(p) = \mathcal{L}_{BS}(q)$.*

► **Proposition 5** ([22, 3]). *Let $X \in \{S, CS, RS, TS, 2S, 3S\}$. Then $p \lesssim_X q$ iff $\mathcal{L}_X(p) \subseteq \mathcal{L}_X(q)$, for all $p, q \in P$.*

► **Remark 6.** Neither \mathbf{ff} nor disjunction are needed in several of the modal characterizations presented in the above result. The reason for adding those constructs to all the logics is that doing so makes our subsequent results more general and uniform. For example, having \mathbf{ff} and disjunction in all logics allows us to provide algorithms that determine whether a formula in a logic \mathcal{L} is prime with respect to a sublogic.

► **Definition 7** ([11, 4]). *Let $\mathcal{L} \subseteq \mathcal{L}_{BS}$. A formula $\varphi \in \mathcal{L}_{BS}$ is prime in \mathcal{L} if for all $\varphi_1, \varphi_2 \in \mathcal{L}$, $\varphi \models \varphi_1 \vee \varphi_2$ implies $\varphi \models \varphi_1$ or $\varphi \models \varphi_2$.*

When the logic \mathcal{L} is clear from the context, we say that φ is prime. Note that every unsatisfiable formula is trivially prime in \mathcal{L} , for every \mathcal{L} .

► **Example 8.** The formula $\langle a \rangle \mathbf{tt}$ is prime in \mathcal{L}_S . Indeed, let $\varphi_1, \varphi_2 \in \mathcal{L}_S$ and assume that $\langle a \rangle \mathbf{tt} \models \varphi_1 \vee \varphi_2$. Since $a.0 \models \langle a \rangle \mathbf{tt}$, without loss of generality, we have that $a.0 \models \varphi_1$. We claim that $\langle a \rangle \mathbf{tt} \models \varphi_1$. To see this, let p be some process such that $p \models \langle a \rangle \mathbf{tt}$ – that is, a process such that $p \xrightarrow{a} p'$ for some p' . It is easy to see that $a.0 \lesssim_S p$. Since $a.0 \models \varphi_1$, Proposition 5 yields that $p \models \varphi_1$, proving our claim and the primality of $\langle a \rangle \mathbf{tt}$. On the other hand, the formula $\langle a \rangle \mathbf{tt} \vee \langle b \rangle \mathbf{tt}$ is not prime in \mathcal{L}_S . Indeed, $\langle a \rangle \mathbf{tt} \vee \langle b \rangle \mathbf{tt} \models \langle a \rangle \mathbf{tt} \vee \langle b \rangle \mathbf{tt}$, but neither $\langle a \rangle \mathbf{tt} \vee \langle b \rangle \mathbf{tt} \models \langle a \rangle \mathbf{tt}$ nor $\langle a \rangle \mathbf{tt} \vee \langle b \rangle \mathbf{tt} \models \langle b \rangle \mathbf{tt}$ hold.

The definition of a characteristic formula within logic \mathcal{L} is given next.

► **Definition 9** ([5, 23, 39]). Let $\mathcal{L} \subseteq \mathcal{L}_{BS}$. A formula $\varphi \in \mathcal{L}$ is characteristic for $p \in P$ within \mathcal{L} iff, for all $q \in P$, it holds that $q \models \varphi \Leftrightarrow \mathcal{L}(p) \subseteq \mathcal{L}(q)$. We denote by $\chi(p)$ the unique characteristic formula for p with respect to logical equivalence.

► **Remark 10.** Let $X \in \{S, CS, RS, TS, 2S, 3S, BS\}$. In light of Theorem 4 and Proposition 5, a formula $\varphi \in \mathcal{L}_X$ is characteristic for p within \mathcal{L}_X iff, for all $q \in P$, it holds that $q \models \varphi \Leftrightarrow p \lesssim_X q$. This property is often used as an alternative definition of characteristic formula for process p modulo \lesssim_X . In what follows, we shall employ the two definitions interchangeably.

In [3, Table 1 and Theorem 5], Aceto, Della Monica, Fabregas, and Ingólfssdóttir presented characteristic formulae for each of the semantics we consider in this paper, and showed that characteristic formulae are exactly the satisfiable and prime ones.

► **Proposition 11** ([3]). For every $X \in \{S, CS, RS, TS, 2S\}$, $\varphi \in \mathcal{L}_X$ is characteristic for some process within \mathcal{L}_X iff φ is satisfiable and prime in \mathcal{L}_X .

► **Remark 12.** Proposition 11 is the only result we use from [3] and we employ it as a “black box”. The (non-trivial) methods used in the proof of that result given in that reference do not play any role in our technical developments.

We note, in passing, that the article [3] does not deal explicitly with $3S$. However, its results apply to all the n -nested simulation preorders.

We can also consider characteristic formulae modulo equivalence relations as follows.

► **Definition 13.** Let $X \in \{S, CS, RS, TS, 2S, 3S, BS\}$. A formula $\varphi \in \mathcal{L}_X$ is characteristic for $p \in P$ modulo \equiv_X iff for all $q \in P$, it holds that $q \models \varphi \Leftrightarrow \mathcal{L}_X(p) = \mathcal{L}_X(q)$.³

When studying the complexity of finding a characteristic formula for some process p with respect to the behavioural relations we have introduced above, we will need some way of measuring the size of the resulting formula as a function of $|p|$. A formula in \mathcal{L}_X , where $X \in \{S, CS, RS, TS, 2S, 3S, BS\}$, can be given in *explicit form* as in Definition 3 or by means of a system of equations. In the latter case, we say that the formula is given in *declarative form*. For example, formula $\phi = \langle a \rangle (\langle a \rangle \mathbf{tt} \wedge \langle b \rangle \mathbf{tt}) \wedge \langle b \rangle (\langle a \rangle \mathbf{tt} \wedge \langle b \rangle \mathbf{tt})$ can be represented by the equations $\phi = \langle a \rangle \phi_1 \wedge \langle b \rangle \phi_1$ and $\phi_1 = \langle a \rangle \mathbf{tt} \wedge \langle b \rangle \mathbf{tt}$. We define:

- the *size* of formula φ , denoted by $|\varphi|$, to be the number of symbols that appear in the explicit form of φ ,
- the *declaration size* of formula φ , denoted by $\text{decl}(\varphi)$, to be the number of equations that are used in the declarative form of φ , and
- the *equational length* of formula φ , denoted by $\text{eqlen}(\varphi)$, to be the maximum number of symbols that appear in an equation in the declarative form of φ .

For example, for the aforementioned formula ϕ , we have that $|\phi| = 13$, $\text{decl}(\phi) = 2$, and $\text{eqlen}(\phi) = 5$. Note that $\text{decl}(\varphi) \leq |\text{Sub}(\varphi)| \leq |\varphi|$, for each φ .

3 The complexity of deciding characteristic formulae modulo preorders

In this section, we address the complexity of deciding whether formulae in \mathcal{L}_S , \mathcal{L}_{CS} , \mathcal{L}_{RS} , \mathcal{L}_{TS} , \mathcal{L}_{2S} , and \mathcal{L}_{3S} are characteristic. Since characteristic formulae in those logics are exactly the satisfiable and prime ones [3, Theorem 5], we study the complexity of checking satisfiability and primality separately in Subsections 3.1 and 3.2.

³ The above definition can also be phrased as follows: A formula $\varphi \in \mathcal{L}_X$ is characteristic for p modulo \equiv_X iff, for all $q \in P$, it holds that $q \models \varphi \Leftrightarrow p \equiv_X q$. This version of the definition is used, in the setting of bisimilarity, in references such as [2, 29].

3.1 The complexity of satisfiability

To address the complexity of the satisfiability problem in \mathcal{L}_S , \mathcal{L}_{CS} , or \mathcal{L}_{RS} , we associate a set $I(\varphi) \subseteq 2^A$ to every formula $\varphi \in \mathcal{L}_{RS}$. Intuitively, $I(\varphi)$ describes all possible sets of initial actions that a process p can have, when $p \models \varphi$.

► **Definition 14.** Let $\varphi \in \mathcal{L}_{RS}$. We define $I(\varphi)$ inductively as follows:

- (a) $I(\mathbf{tt}) = 2^A$,
 - (b) $I(\mathbf{ff}) = \emptyset$,
 - (c) $I([a]\mathbf{ff}) = \{X \mid X \subseteq A \text{ and } a \notin X\}$,
 - (d) $I(\langle a \rangle \varphi) = \begin{cases} \emptyset, & \text{if } I(\varphi) = \emptyset, \\ \{X \mid X \subseteq A \text{ and } a \in X\}, & \text{otherwise} \end{cases}$
 - (e) $I(\varphi_1 \vee \varphi_2) = I(\varphi_1) \cup I(\varphi_2)$,
 - (f) $I(\varphi_1 \wedge \varphi_2) = I(\varphi_1) \cap I(\varphi_2)$.
- Note that $I(\mathbf{0}) = \{\emptyset\}$.

► **Lemma 15.** For every $\varphi \in \mathcal{L}_{RS}$, the following statements hold:

- (a) for every $S \subseteq A$, $S \in I(\varphi)$ iff there is a process p such that $I(p) = S$ and $p \models \varphi$.
- (b) φ is unsatisfiable iff $I(\varphi) = \emptyset$.

When the number of actions is constant, $I(\varphi)$ can be computed in linear time for every $\varphi \in \mathcal{L}_{RS}$. For \mathcal{L}_{CS} , we need even less information; indeed, it is sufficient to define $I(\varphi)$ so that it encodes whether φ is unsatisfiable, or is satisfied only in deadlocked states (that is, states with an empty set of initial actions), or is satisfied only in processes that are not deadlocked, or is satisfied both in some deadlocked and non-deadlocked states. This information can be computed in linear time for every $\varphi \in \mathcal{L}_{CS}$, regardless of the size of the action set.

► **Corollary 16.**

- (a) Satisfiability of formulae in \mathcal{L}_{CS} and \mathcal{L}_S is decidable in linear time.
- (b) Let $|A| = k$, where $k \geq 1$ is a constant. Satisfiability of formulae in \mathcal{L}_{RS} is decidable in linear time.

On the other hand, if we can use an unbounded number of actions, the duality of $\langle a \rangle$ and $[a]$ can be employed to define a polynomial-time reduction from SAT, the satisfiability problem for propositional logic, to satisfiability in \mathcal{L}_{RS} . Moreover, if we are allowed to nest $[a]$ modalities ($a \in A$) and have at least two actions, we can encode n propositional literals using formulae of $\log n$ size and reduce SAT to satisfiability in \mathcal{L}_{TS} in polynomial time. Finally, satisfiability in \mathcal{L}_{2S} is in NP, which can be shown by an appropriate tableau construction.

► **Proposition 17.** Let either $X = RS$ and $|A|$ be unbounded or $X \in \{TS, 2S\}$ and $|A| > 1$. Satisfiability of formulae in \mathcal{L}_X is NP-complete.

Deciding satisfiability of formulae in $\overline{\mathcal{L}}_{2S}$ when $|A| > 1$, turns out to be PSPACE-complete. (A proof is provided in [1, Appendix B.6].) This means that satisfiability of \mathcal{L}_{3S} is also PSPACE-complete, since $\overline{\mathcal{L}}_{2S} \subseteq \mathcal{L}_{3S}$.

► **Proposition 18.** Let $|A| > 1$. Satisfiability of formulae in \mathcal{L}_{3S} is PSPACE-complete.

3.2 The complexity of primality

We now study the complexity of checking whether a formula is prime in the logics that characterize some of the relations in Definition 1.

■ **Table 1** Rules for the simulation preorder. If \forall is displayed in the conclusion of a rule, then the rule is called universal. Otherwise, it is called existential.

$\frac{\varphi_1 \vee \varphi_2, \varphi \Rightarrow \psi}{\varphi_1, \varphi \Rightarrow \psi \mid \forall \varphi_2, \varphi \Rightarrow \psi} \text{ (L}\vee_1\text{)}$	$\frac{\varphi, \varphi_1 \vee \varphi_2 \Rightarrow \psi}{\varphi_1, \varphi \Rightarrow \psi \mid \forall \varphi_2, \varphi \Rightarrow \psi} \text{ (L}\vee_2\text{)}$
$\frac{\varphi_1 \wedge \varphi_2, \varphi \Rightarrow \langle a \rangle \psi}{\varphi_1, \varphi \Rightarrow \langle a \rangle \psi \mid \exists \varphi_2, \varphi \Rightarrow \langle a \rangle \psi} \text{ (L}\wedge_1\text{)}$	$\frac{\varphi, \varphi_1 \wedge \varphi_2 \Rightarrow \langle a \rangle \psi}{\varphi_1, \varphi \Rightarrow \langle a \rangle \psi \mid \exists \varphi_2, \varphi \Rightarrow \langle a \rangle \psi} \text{ (L}\wedge_2\text{)}$
$\frac{\varphi_1, \varphi_2 \Rightarrow \psi_1 \wedge \psi_2}{\varphi_1, \varphi_2 \Rightarrow \psi_1 \mid \forall \varphi_1, \varphi_2 \Rightarrow \psi_2} \text{ (R}\wedge\text{)}$	$\frac{\varphi_1, \varphi_2 \Rightarrow \psi_1 \vee \psi_2}{\varphi_1, \varphi_2 \Rightarrow \psi_1 \mid \exists \varphi_1, \varphi_2 \Rightarrow \psi_2} \text{ (R}\vee\text{)}$
$\frac{\langle a \rangle \varphi_1, \langle a \rangle \varphi_2 \Rightarrow \langle a \rangle \psi}{\varphi_1, \varphi_2 \Rightarrow \psi} \text{ (}\diamond\text{)}$	$\frac{\varphi_1, \varphi_2 \Rightarrow \mathbf{tt}}{\mathbf{TRUE}} \text{ (tt)}$

Primality in \mathcal{L}_S . Unsatisfiable formulae are trivially prime. Note also that in the case that $|A| = 1$, all satisfiable formulae in \mathcal{L}_S are prime. To address the problem for any action set, for every satisfiable formula $\varphi \in \mathcal{L}_S$ we can efficiently compute a logically equivalent formula φ' given by the grammar $\varphi ::= \mathbf{tt} \mid \langle a \rangle \varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$. We examine the complexity of deciding primality of such formulae.

► **Proposition 19.** *Let $\varphi \in \mathcal{L}_S$ such that $\mathbf{ff} \notin \text{Sub}(\varphi)$. Deciding whether φ is prime is in P.*

Proof. We describe algorithm `PrimesS` that, on input φ , decides primality of φ . `PrimesS` constructs a rooted directed acyclic graph, denoted by G_φ , from the formula φ as follows. Every vertex of the graph is either of the form $\varphi_1, \varphi_2 \Rightarrow \psi$ – where φ_1, φ_2 and ψ are sub-formulae of φ –, or `TRUE`. The algorithm starts from vertex $x = (\varphi, \varphi \Rightarrow \varphi)$ and applies some rule in Table 1 to x in top-down fashion to generate one or two new vertices that are given at the bottom of the rule. These vertices are the children of x and the vertex x is labelled with either \exists or \forall , depending on which one is displayed at the bottom of the applied rule. If x has only one child, `PrimesS` labels it with \exists . The algorithm recursively continues this procedure on the children of x . If no rule can be applied on a vertex, then this vertex has no outgoing edges. For the sake of clarity and consistency, we assume that right rules, i.e. (R \vee) and (R \wedge), are applied before the left ones, i.e. (L \vee_i) and (L \wedge_i), $i = 1, 2$, by the algorithm. The graph generated in this way is an *alternating graph*, as defined by Immerman in [28, Definition 3.24] (see also [1, Appendix A]). In G_φ , the source vertex s is $\varphi, \varphi \Rightarrow \varphi$, and the target vertex t is `TRUE`. Algorithm `PrimesS` solves the problem `REACHa` on input G_φ , where `REACHa` is `REACHABILITY` on alternating graphs and is defined in [28, pp. 53–54]. It accepts φ iff `REACHa` accepts G_φ . Intuitively, the source vertex $(\varphi, \varphi \Rightarrow \varphi)$ can reach the target vertex `TRUE` in the alternating graph G_φ exactly when for each pair of disjuncts ψ_1 and ψ_2 in the disjunctive normal form of φ there is a disjunct ψ_3 in the disjunctive normal form of φ that is entailed by both ψ_1 and ψ_2 . It turns out that this is a necessary and sufficient condition for the primality of φ . For example, consider the formula $\langle a \rangle \mathbf{tt} \vee \langle b \rangle \mathbf{tt}$. There is no disjunct of $\langle a \rangle \mathbf{tt} \vee \langle b \rangle \mathbf{tt}$ that is entailed by both $\langle a \rangle \mathbf{tt}$ and $\langle b \rangle \mathbf{tt}$. This is because that formula is not prime, as we observed in Example 8. On the other hand, the formula $\langle a \rangle \mathbf{tt} \vee \langle a \rangle \langle b \rangle \mathbf{tt}$ is prime since each of its disjuncts entails $\langle a \rangle \mathbf{tt}$. The full technical details are included in [1, Appendix C.1]. Note that graph G_φ is of polynomial size and there is a linear-time algorithm solving `REACHa` [28]. ◀

26:10 The Complexity of Deciding Characteristic Formulae

Primality in \mathcal{L}_{CS} . Note that, in the case of \mathcal{L}_{CS} , the rules in Table 1 do not work any more because, unlike \mathcal{L}_S , the logic \mathcal{L}_{CS} can express some “negative information” about the behaviour of processes. For example, let $A = \{a\}$ and $\varphi = \langle a \rangle \mathbf{tt}$. Then, Primes_S accepts φ , even though φ is not prime in \mathcal{L}_{CS} . Indeed, $\varphi \models \langle a \rangle \langle a \rangle \mathbf{tt} \vee \langle a \rangle \mathbf{0}$, but $\varphi \not\models \langle a \rangle \langle a \rangle \mathbf{tt}$ and $\varphi \not\models \langle a \rangle \mathbf{0}$. However, we can overcome this problem as described in the proof sketch of Proposition 20 below.

► **Proposition 20.** *Let $\varphi \in \mathcal{L}_{CS}$ be a formula such that every $\psi \in \text{Sub}(\varphi)$ is satisfiable. Deciding whether φ is prime is in \mathbf{P} .*

Proof. Consider the algorithm that first computes the formula φ^\diamond by applying rule $\langle a \rangle \mathbf{tt} \rightarrow_\diamond \mathbf{tt}$, and rules $\mathbf{tt} \vee \psi \rightarrow_{\mathbf{tt}} \mathbf{tt}$ and $\mathbf{tt} \wedge \psi \rightarrow_{\mathbf{tt}} \psi$ modulo commutativity on φ . It holds that φ is prime iff φ^\diamond is prime and $\varphi^\diamond \models \varphi$. Next, the algorithm decides primality of φ^\diamond by solving reachability on a graph constructed as in the case of simulation using the rules in Table 1, where rule (tt) is replaced by rule (0), whose premise is $\mathbf{0}, \mathbf{0} \Rightarrow \mathbf{0}$ and whose conclusion is \mathbf{TRUE} . To verify $\varphi^\diamond \models \varphi$, the algorithm computes a process p for which φ^\diamond is characteristic within \mathcal{L}_{CS} and checks whether $p \models \varphi$. In fact, the algorithm has also a preprocessing phase during which it applies a set of rules on φ and obtains an equivalent formula with several desirable properties. See [1, Appendix C.2] for full details. ◀

Primality in \mathcal{L}_{RS} . The presence of formulae of the form $[a]\mathbf{ff}$ in \mathcal{L}_{RS} means that a prime formula $\varphi \in \mathcal{L}_{RS}$ has at least to describe which actions are necessary or forbidden for any process that satisfies φ . For example, let $A = \{a, b\}$. Then, $\langle a \rangle \mathbf{0}$ is not prime, since $\langle a \rangle \mathbf{0} \models (\langle a \rangle \mathbf{0} \wedge [b]\mathbf{ff}) \vee (\langle a \rangle \mathbf{0} \wedge \langle b \rangle \mathbf{tt})$, and $\langle a \rangle \mathbf{0}$ entails neither $\langle a \rangle \mathbf{0} \wedge [b]\mathbf{ff}$ nor $\langle a \rangle \mathbf{0} \wedge \langle b \rangle \mathbf{tt}$. Intuitively, we call a formula φ *saturated* if φ describes exactly which actions label the outgoing edges of any process p such that $p \models \varphi$. Formally, φ is saturated iff $I(\varphi)$ is a singleton.

If the action set is bounded by a constant, given φ , we can efficiently construct a formula φ^s such that (1) φ^s is saturated and for every $\langle a \rangle \varphi' \in \text{Sub}(\varphi^s)$, φ' is saturated, (2) φ is prime iff φ^s is prime and $\varphi^s \models \varphi$, and (3) primality of φ^s can be efficiently reduced to $\text{REACH}_a(G_{\varphi^s})$.

► **Proposition 21.** *Let $|A| = k$, where $k \geq 1$ is a constant, and $\varphi \in \mathcal{L}_{RS}$ be such that if $\psi \in \text{Sub}(\varphi)$ is unsatisfiable, then $\psi = \mathbf{ff}$ and ψ occurs in the scope of some $[a]$. Deciding whether φ is prime is in \mathbf{P} .*

As the following result indicates, primality checking for formulae in \mathcal{L}_{RS} becomes computationally hard when $|A|$ is not a constant.

► **Proposition 22.** *Let $|A|$ be unbounded. Deciding primality of formulae in \mathcal{L}_{RS} is coNP -complete.*

Proof. We give a polynomial-time reduction from SAT to deciding whether a formula in \mathcal{L}_{RS} is not prime. Let φ be a propositional formula over x_0, \dots, x_{n-1} . We set $\varphi' = (\varphi \wedge \neg x_n) \vee (x_n \wedge \bigwedge_{i=1}^{n-1} \neg x_i)$ and φ'' to be φ' where x_i is substituted with $\langle a_i \rangle \mathbf{0}$ and $\neg x_i$ with $[a_i]\mathbf{ff}$, where $A = \{a_0, \dots, a_n\}$. As φ'' is satisfied in $a_n.\mathbf{0}$, it is a satisfiable formula, and so φ'' is prime in \mathcal{L}_{RS} iff φ'' is characteristic within \mathcal{L}_{RS} . We show that φ is satisfiable iff φ'' is not characteristic within \mathcal{L}_{RS} . Let φ be satisfiable and let s denote a satisfying assignment of φ . Consider $p_1, p_2 \in P$ such that:

- $p_1 \xrightarrow{a_i} \mathbf{0}$ iff $s(x_i) = \text{true}$, for $0 \leq i \leq n-1$, and $p_1 \not\xrightarrow{a_n}$, and
- $p_2 \xrightarrow{a_n} \mathbf{0}$ and $p_2 \not\xrightarrow{a}$ for every $a \in A \setminus \{a_n\}$.

It holds that $p_i \models \varphi''$, $i = 1, 2$, $p_1 \not\lesssim_{RS} p_2$, and $p_2 \not\lesssim_{RS} p_1$. Suppose that there is a process q , such that φ'' is characteristic for q within \mathcal{L}_{RS} . If $q \xrightarrow{a_n}$, then $q \not\lesssim_{RS} p_1$. On the other hand, if $q \not\xrightarrow{a_n}$, then $q \not\lesssim_{RS} p_2$. So, both cases lead to a contradiction, which means that φ'' is not characteristic within \mathcal{L}_{RS} . For the converse implication, assume that φ is unsatisfiable. This implies that there is no process satisfying the first disjunct of φ'' . Thus, φ'' is characteristic for p_2 , described above, within \mathcal{L}_{RS} .

Proving the matching upper bound is non-trivial. There is a coNP algorithm that uses properties of prime formulae and rules of Table 1, carefully adjusted to the case of ready simulation. We describe the algorithm and prove its correctness in [1, Appendix C.3.2]. ◀

Primality in \mathcal{L}_{TS} . If we have more than one action, a propositional literal can be encoded by using the restricted nesting of modal operators that is allowed by the grammar for \mathcal{L}_{TS} . This observation is the crux of the proof of the following result.

▶ **Proposition 23.** *Let $|A| > 1$. Deciding primality of formulae in \mathcal{L}_{TS} is coNP-hard.*

Proof. Let $A = \{0, 1\}$. The proof follows the steps of the proof of Proposition 22. The initial and basic idea is that given an instance φ of SAT over x_1, \dots, x_n , every x_i is substituted with $[\overline{b_{i1}}]\mathbf{ff} \wedge \langle b_{i1} \rangle([\overline{b_{i2}}]\mathbf{ff} \wedge \langle b_{i2} \rangle(\dots([\overline{b_{ik}}]\mathbf{ff} \wedge \langle b_{ik} \rangle \mathbf{0}) \dots))$ and $\neg x_i$ with $[b_{i1}][b_{i2}] \dots [b_{ik}]\mathbf{ff}$, where $b_{i1} \dots b_{ik}$ is the binary representation of i and $\overline{b} = 0$, if $b = 1$, and $\overline{b} = 1$, if $b = 0$. For more technical details, see [1, Appendix C.4.1]. ◀

In contrast to the case for \mathcal{L}_{RS} , bounding the size of the action set is not sufficient for deciding primality of formulae in \mathcal{L}_{TS} in polynomial time. However, we show that both satisfiability and primality become efficiently solvable if we bound both $|A|$ and the modal depth of the input formula.

▶ **Proposition 24.** *Let $|A| = k$ and $\varphi \in \mathcal{L}_{TS}$ with $\text{md}(\varphi) = d$, where $k, d \geq 1$ are constants. Then, there is an algorithm that decides whether φ is satisfiable and prime in linear time.*

Proof. It is necessary and sufficient to check that there is a process p with $\text{depth}(p) \leq d$ such that (1) $p \models \varphi$ and (2) for every q with $\text{depth}(q) \leq d + 1$, if $q \models \varphi$ then $p \lesssim_{TS} q$. Since k and d are considered to be constants, there is an algorithm that does so and requires linear time in $|\varphi|$. In particular, the algorithm runs in $\mathcal{O}(2^{2k^{d+1}} \cdot k^{d+1} \cdot |\varphi|)$. ◀

To classify the problem of deciding whether formulae in \mathcal{L}_{TS} are characteristic when $|A|$ is bounded, let us briefly introduce fixed-parameter tractable problems – see, for instance, [19, 21] for textbook accounts of this topic. Let $L \subseteq \Sigma^* \times \Sigma^*$ be a parameterized problem. We denote by L_y the associated fixed-parameter problem $L_y = \{x \mid (x, y) \in L\}$, where y is the parameter. Then, $L \in \text{FPT}$ (or L is fixed-parameter tractable) if there are a constant α and an algorithm to determine if (x, y) is in L in time $f(|y|) \cdot |x|^\alpha$, where f is a computable function [18].

▶ **Corollary 25.** *Let $|A| = k$, where $k \geq 1$ is a constant. The problems of deciding whether formulae in \mathcal{L}_{TS} are satisfiable, prime, and characteristic are in FPT, with the modal depth of the input formula as the parameter.*

We note that the coNP-hardness argument from Proposition 23 applies also to logics that include \mathcal{L}_{TS} . Since $\mathcal{L}_{TS} \subseteq \mathcal{L}_{2S}$, the coNP-hardness of deciding primality of formulae in \mathcal{L}_{TS} with $|A| > 1$ implies the same lower bound for deciding primality of formulae in \mathcal{L}_{2S} when $|A| > 1$. Next, we show that in \mathcal{L}_{3S} with $|A| > 1$ the problem becomes PSPACE-hard.

26:12 The Complexity of Deciding Characteristic Formulae

Primality in \mathcal{L}_{3S} . Let $|A| > 1$. PSPACE-hardness of \mathcal{L}_{3S} -satisfiability implies PSPACE-hardness of $\overline{\mathcal{L}}_{3S}$ -validity. Along the lines of the proof of [2, Theorem 26], we prove the following result.

► **Proposition 26.** *Let $|A| > 1$. Deciding prime formulae within \mathcal{L}_{3S} is PSPACE-hard.*

► **Remark 27.** Note that primality within \mathcal{L}_{BS} coincides with primality modulo \sim . In [2], primality modulo \sim is called completeness and it is shown to be decidable in PSPACE. However, the algorithm used in [2] does not immediately imply that primality within \mathcal{L}_{3S} is in PSPACE.

Interestingly, PSPACE-hardness of \mathcal{L}_{2S} -validity implies the following theorem.

► **Theorem 28.** *Let $X \in \{CS, RS, TS, 2S, 3S\}$ and $|A| > 1$. The problem of deciding whether a formula in $\overline{\mathcal{L}}_{2S}$ is prime in \mathcal{L}_X is PSPACE-hard.*

Proof. We reduce \mathcal{L}_{2S} -validity to this problem. Let $\varphi \in \mathcal{L}_{2S}$. The reduction will return a formula φ' , such that φ is \mathcal{L}_{2S} -valid if and only if φ' is prime in \mathcal{L}_X . If $0 \not\models \varphi$, then let $\varphi' = \mathbf{tt}$; in this case, φ is not valid and \mathbf{tt} is not prime in \mathcal{L}_X . Otherwise, let $\varphi' = \mathbf{0} \vee \neg\varphi$. If φ is valid, then $\varphi' \equiv \mathbf{0}$ and therefore φ' is prime in \mathcal{L}_X . On the other hand, if φ is not valid, then there is some process $p \models \neg\varphi$. From $0 \models \varphi$, it holds that $p \xrightarrow{a}$. Then, $\varphi' \models \mathbf{0} \vee \bigvee_{a \in A} \langle a \rangle \mathbf{tt}$, but $\varphi' \not\models \mathbf{0}$ and $\varphi' \not\models \bigvee_{a \in A} \langle a \rangle \mathbf{tt}$. Since $\mathbf{0} \vee \bigvee_{a \in A} \langle a \rangle \mathbf{tt} \in \mathcal{L}_{CS}$, φ' is not prime in \mathcal{L}_X , where $X \in \{CS, RS, TS, 2S, 3S\}$. ◀

Theorem 28 shows that when deciding primality in \mathcal{L}_X , if we allow the input to be in a logic \mathcal{L} that is more expressive than \mathcal{L}_X , the computational complexity of the problem can increase. It is then reasonable to constrain the input of the problem to be in \mathcal{L}_X in order to obtain tractable problems as in the case of \mathcal{L}_S and \mathcal{L}_{CS} .

Before we give our main result summarizing the complexity of deciding characteristic formulae, we introduce two classes that play an important role in pinpointing the complexity of deciding characteristic formulae within \mathcal{L}_{RS} , \mathcal{L}_{TS} , and \mathcal{L}_{2S} . The first class is $\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP and } L_2 \in \text{coNP}\}$ [38] and the second one is US [9], which is defined thus: A language $L \in \text{US}$ iff there is a non-deterministic Turing machine T such that, for every instance x of L , $x \in L$ iff T has exactly one accepting path. The problem UNIQUE SAT , viz. the problem of deciding whether a given Boolean formula has exactly one satisfying truth assignment, is US -complete and $\text{US} \subseteq \text{DP}$ [9].

► **Theorem 29.**

- (a) *Deciding characteristic formulae within \mathcal{L}_S , \mathcal{L}_{CS} , or \mathcal{L}_{RS} with a bounded action set is in P.*
- (b) *Deciding characteristic formulae within \mathcal{L}_{RS} with an unbounded action set is US-hard and belongs to DP.*
- (c) *Deciding characteristic formulae within \mathcal{L}_{TS} or \mathcal{L}_{2S} is US-hard.*
- (d) *Deciding characteristic formulae within \mathcal{L}_{3S} is PSPACE-hard.*

4 Finding characteristic formulae: The gap between trace simulation and the other preorders

Let $X \in \{S, CS\}$ or $X = RS$ and $|A|$ is bounded by a constant. The complexity of finding characteristic formulae within \mathcal{L}_X depends on the representation of the output. If the characteristic formula has to be given in explicit form, then the following result holds.

■ **Table 2** The complexity of deciding satisfiability and primality, and of finding characteristic formulae for different logics. Finding_{decl} (resp. Finding_{expl}) denotes the problem of finding the characteristic formula for a given finite loop-free process, when the output is given in declarative (resp. explicit) form. Superscripts $= k$, $> k$, and > 1 mean that the action set is bounded by a constant, unbounded, and has more than one action, respectively. FP is the class of functions computable in polynomial time. All the results shown in white cells have been proven in this paper, whereas results in light gray are from [2].

	\mathcal{L}_S	\mathcal{L}_{CS}	$\mathcal{L}_{RS}^{=k}$	$\mathcal{L}_{RS}^{>k}$	$\mathcal{L}_{TS}^{>1}$	$\mathcal{L}_{2S}^{>1}$	$\mathcal{L}_{3S}^{>1}$	\mathcal{L}_{BS}
Satisfiability	P	P	P	NP-comp.	NP-comp.	NP-comp.	PSPACE-comp.	PSPACE-comp.
Primality	P	P	P	coNP-comp.	coNP-hard	coNP-hard	PSPACE-hard	PSPACE-comp.
Finding_{decl}	FP	FP	FP	FP	NP-hard	FP	FP	FP
Finding_{expl}	NP-hard							

► **Proposition 30.** *Let $X \in \{S, CS\}$ or $X = RS$ and $|A|$ is bounded by a constant. If finding the characteristic formula within \mathcal{L}_X for a given finite loop-free process can be done in polynomial time when the output is given in explicit form, then $P = NP$.*

Proof. If the assumption of the proposition is true, the results of this paper allow us to decide trace equivalence of two finite loop-free processes in polynomial time. (For details, the reader can see [1, Appendix E.1].) Since trace equivalence for such processes is coNP-complete [27, Theorem 2.7(1)], this implies that $P = NP$. ◀

However, if output formulae are given in declarative form, then finding characteristic formulae within \mathcal{L}_X , where $X \in \{nS, CS, RS, BS\}$, $n \geq 1$, can be done in polynomial time.

► **Proposition 31.** *For every $X \in \{nS, CS, RS, BS\}$, where $n \geq 1$, there is a polynomial-time algorithm that, given a finite loop-free process p , outputs a formula in declarative form that is characteristic for p within \mathcal{L}_X .*

Proof. The proof relies on inductive definitions of characteristic formulae within \mathcal{L}_X , where $X \in \{S, CS, RS, 2S, BS\}$, given in [29, 6], and within \mathcal{L}_{nS} , $n \geq 3$, given in [1, Appendix E.1]. These definitions guarantee that there are polynomial-time recursive procedures which construct characteristic formulae within \mathcal{L}_X . We prove the proposition for $X = 2S$ below.

Given a finite loop-free process p , the characteristic formula for p within \mathcal{L}_{2S} is defined as follows: $\chi_{2S}(p) = \bar{\chi}_S(p) \wedge \bigwedge_{a \in A} \bigwedge_{p \xrightarrow{a} p'} \langle a \rangle \chi_{2S}(p')$, where $\bar{\chi}_S(p) = \bigwedge_{a \in A} [a] \bigvee_{p \xrightarrow{a} p'} \bar{\chi}_S(p')$.

Consider the algorithm that recursively constructs $\chi_{2S}(p)$. The algorithm has to construct $\chi_{2S}(p')$ and $\bar{\chi}_S(p')$ for every $p' \in \text{reach}(p)$, yielding a linear number of equations. Moreover, for every $p' \in \text{reach}(p)$, $\bar{\chi}_S(p')$ is of linear size in $|p'|$. If $p' = 0$, then $\bar{\chi}_S(p') = \bigwedge_{a \in A} [a] \mathbf{ff}$. Otherwise, $|\bar{\chi}_S(p')| = \mathcal{O}(|\{p'' \mid p' \xrightarrow{a} p''\}| + |A|)$, where $|A|$ is added because for every $a \in A$ such that $p' \xrightarrow{a}$, $[a] \mathbf{ff}$ is a conjunct of $\bar{\chi}_S(p')$. Note that for every p'' , if $\bar{\chi}_S(p'')$ occurs in $\bar{\chi}_S(p')$, it is considered to add 1 to the size of $\bar{\chi}_S(p')$. Therefore, $|\bar{\chi}_S(p')|$ is of linear size in $|p'|$. Using a similar argument, we can show that $\chi_{2S}(p')$ is of linear size. Thus, the algorithm constructs a linear number of equations, each of which is of linear size in $|p|$. The proofs for $X \in \{nS, CS, RS, BS\}$, $n \neq 2$, are analogous. ◀

26:14 The Complexity of Deciding Characteristic Formulae

► **Remark 32.** Note that the recursive procedures given in [29, 6] and [1, Appendix E.1] provide characteristic formulae for finite processes with loops provided that we enrich the syntax of our logics by adding greatest fixed points. See, for example, [6]. Consequently, constructing characteristic formulae for finite processes within \mathcal{L}_X , $X \in \{nS, CS, RS, BS\}$, $n \geq 1$, can be done in polynomial time.

We now present the complexity gap between finding characteristic formulae for preorders CS, RS, BS , and nS , $n \geq 1$, and the same search problem for preorder TS . In the former case, there are characteristic formulae with both declaration size and equational length that are polynomial in the size of the processes they characterize, and they can be efficiently computed. On the contrary, for TS , even if characteristic formulae are always of polynomial declaration size and polynomial equational length, they *cannot* be efficiently computed unless $P = NP$.

► **Proposition 33.** *Assume that for every finite loop-free process p , there is a characteristic formula within \mathcal{L}_{TS} for p , denoted by $\chi(p)$, such that both $\text{decl}(\chi(p))$ and $\text{eqlen}(\chi(p))$ are in $\mathcal{O}(|p|^k)$ for some $k \in \mathbb{N}$. Given a finite loop-free process p , if $\chi(p)$ can be computed in polynomial time, then $P = NP$.*

Next, we prove that we do not expect that a finite loop-free process p has always a short characteristic formula within \mathcal{L}_{TS} when this is combined with a second condition. To show that statement, we need the following lemma.

► **Lemma 34.** *For every finite p and q , $\text{traces}(p) = \text{traces}(q)$ iff $p \lesssim_{TS} p+q$ and $q \lesssim_{TS} p+q$.*

Proof. If $\text{traces}(p) = \text{traces}(q)$, then $p \lesssim_{TS} p+q$. Indeed, for every $p \xrightarrow{a} p'$, it holds that $p+q \xrightarrow{a} p'+q$ and, trivially, $p' \lesssim_{TS} p'$. Moreover, $\text{traces}(p+q) = \text{traces}(p) \cup \text{traces}(q) = \text{traces}(p)$. Symmetrically, $q \lesssim_{TS} p+q$. Conversely, if $p \lesssim_{TS} p+q$ and $q \lesssim_{TS} p+q$, then $\text{traces}(p+q) = \text{traces}(p) = \text{traces}(q)$, and we are done. ◀

► **Proposition 35.** *Assume that the following two conditions hold:*

1. *For every finite loop-free process p , there is a characteristic formula within \mathcal{L}_{TS} for p , denoted by $\chi(p)$, such that both $\text{decl}(\chi(p))$ and $\text{eqlen}(\chi(p))$ are in $\mathcal{O}(|p|^k)$ for some $k \in \mathbb{N}$.*
2. *Given a finite loop-free process p and a formula φ in declarative form, deciding whether φ is characteristic for p within \mathcal{L}_{TS} is in NP.*

Then $NP = \text{coNP}$.

Proof. We describe an NP algorithm \mathcal{A} that decides non-membership in SAT and makes use of conditions 1 and 2 of the proposition. Let ϕ be an input CNF formula to SAT. Algorithm \mathcal{A} computes the DNF formula $\neg\phi$ for which it needs to decide DNF-TAUTOLOGY. Then, \mathcal{A} reduces DNF-TAUTOLOGY to deciding trace equivalence of processes p_0 and q constructed as described in the proof of [27, Theorem 2.7(1)]. \mathcal{A} can decide if $\text{traces}(p_0) = \text{traces}(q)$ by checking $p_0 \lesssim_{TS} p_0+q$ and $q \lesssim_{TS} p_0+q$ because of Lemma 34. Finally, \mathcal{A} reduces $p_0 \lesssim_{TS} p_0+q$ (resp. $q \lesssim_{TS} p_0+q$) to model checking: it needs to check whether $p_0+q \models \chi(p_0)$ (resp. $p_0+q \models \chi(q)$). To this end, \mathcal{A} guesses two formulae φ_{p_0} and φ_q in declarative form of polynomial declaration size and equational length, and two witnesses that verify that φ_{p_0} and φ_q are characteristic within \mathcal{L}_{TS} for p_0 and q , respectively. This can be done due to conditions 1 and 2. \mathcal{A} rejects the input iff both $p_0+q \models \chi(p_0)$ and $p_0+q \models \chi(q)$ are true. ◀

5 A note on deciding characteristic formulae modulo equivalence relations

So far, we have studied the complexity of algorithmic problems related to characteristic formulae in the modal logics that characterize the simulation-based preorders in van Glabbeek's spectrum. As shown in [3], those logics are powerful enough to describe characteristic formulae for each finite, loop-free process up to the preorder they characterize. It is therefore natural to wonder whether they can also express characteristic formulae modulo the kernels of those preorders. The following result indicates that the logics \mathcal{L}_X , where $X \in \{S, CS, RS\}$, have very weak expressive power when it comes to defining characteristic formulae modulo \equiv_X .

► **Proposition 36.** *No formula in \mathcal{L}_S is characteristic for some process p with respect to \equiv_S . For $X \in \{CS, RS\}$, a formula φ is characteristic for some process p with respect to \equiv_X iff it is logically equivalent to $\bigwedge_{a \in A} [a]\mathbf{ff}$.*

Proof. Assume, towards contradiction, that there is a formula φ_c^S in \mathcal{L}_S that is characteristic for some process p with respect to \equiv_S . Let ℓ be the depth of p and $a \in A$. Define process $q = p + a^{\ell+1}0$ – that is, q is a copy of p with an additional path that has exactly $\ell + 1$ a -transitions. It is easy to see that $p \lesssim_S q$, but $q \not\lesssim_S p$. Since $p \models \varphi_c^S$, it holds that $q \models \varphi_c^S$. However, $q \not\equiv_S p$, which contradicts our assumption that φ_c^S is characteristic for p with respect to \equiv_S . For $X \in \{CS, RS\}$, note that a formula φ is logically equivalent to $\bigwedge_{a \in A} [a]\mathbf{ff}$ iff it is satisfied only by processes without outgoing transitions, and so it is characteristic for any such process modulo \equiv_X . To prove that no formula is characteristic for some process p with positive depth modulo \equiv_{CS} or \equiv_{RS} , a similar argument to the one for \equiv_S can be used. For \equiv_{RS} , the action a should be chosen such that $p \xrightarrow{a} p'$ for some p' . ◀

For TS and $2S$, there are non-trivial characteristic formulae modulo \equiv_{TS} and \equiv_{2S} , respectively. For example, if $A = \{a, b\}$, the formula $\varphi_a = \langle a \rangle ([a]\mathbf{ff} \wedge [b]\mathbf{ff}) \wedge [b]\mathbf{ff} \wedge [a][a]\mathbf{ff} \wedge [a][b]\mathbf{ff}$ is satisfied only by processes that are equivalent, modulo those equivalences, to process $p_a = a.0$ that has a single transition labelled with a . Thus, φ_a is characteristic for p_a modulo both \equiv_{TS} and \equiv_{2S} . We can use the following theorem as a tool to prove hardness of deciding characteristic formulae modulo some equivalence relation. Theorem 37 below is an extension of [2, Theorem 26], so that it holds for every X such that a characteristic formula modulo \equiv_X exists, namely $X \in \{CS, RS, TS, 2S, 3S, BS\}$.

► **Theorem 37.** *Let $X \in \{CS, RS, TS, 2S, 3S, BS\}$. Validity in $\overline{\mathcal{L}}_X$ reduces in polynomial time to deciding characteristic formulae with respect to \equiv_X .*

Note that, from the results of Subsection 3.1, validity in $\overline{\mathcal{L}}_{RS}$ with an unbounded action set, $\overline{\mathcal{L}}_{TS}$ with $|A| > 1$, and $\overline{\mathcal{L}}_{2S}$ with $|A| > 1$ is coNP-complete, whereas validity in $\overline{\mathcal{L}}_{3S}$ with $|A| > 1$ is PSPACE-complete. Consequently, from Theorem 37, deciding whether a formula is characteristic modulo \equiv_{RS} with an unbounded action set, \equiv_{TS} with $|A| > 1$, and \equiv_{2S} with $|A| > 1$ is coNP-hard. That problem is PSPACE-hard modulo \equiv_{3S} with $|A| > 1$.

6 Conclusions

In this paper, we studied the complexity of determining whether a formula is characteristic for some finite, loop-free process in each of the logics providing modal characterizations of the simulation-based semantics in van Glabbeek's branching-time spectrum [22]. Since, as shown in [3], characteristic formulae in each of those logics are exactly the satisfiable and prime ones, we gave complexity results for the satisfiability and primality problems,

and investigated the boundary between logics for which those problems can be solved in polynomial time and those for which they become computationally hard. Our results show that computational hardness already manifests itself in ready simulation semantics [10, 34] when the size of the action set is not a constant. Indeed, in that setting, the mere addition of formulae of the form $[a]\mathbf{ff}$ to the logic that characterizes the simulation preorder yields a logic whose satisfiability and primality problems are NP-hard and coNP-hard respectively. Moreover, we show that deciding primality in the logic characterizing 3-nested simulation is PSPACE-hard in the presence of at least two actions.

Amongst others, we also studied the complexity of constructing characteristic formulae in each of the logics we consider, both when such formulae are presented in explicit form and in declarative form. In particular, one of our results identifies a sharp difference between trace simulation and the other semantics when it comes to constructing characteristic formulae. For all the semantics apart from trace simulation, there are characteristic formulae that have declaration size and equational length that are polynomial in the size of the processes they characterize and they can be efficiently computed. On the contrary, for trace simulation, even if characteristic formulae are always of polynomial declaration size and polynomial equational length, they *cannot* be efficiently computed, unless $P = NP$.

Our results are summarized in Table 2 and open several avenues for future research that we are currently pursuing. First of all, the precise complexity of primality checking is still open for the logics characterizing the n -nested simulation semantics. We conjecture that checking primality in \mathcal{L}_{2S} is coNP-complete and that PSPACE-completeness holds for n -nested simulation when $n \geq 3$. Next, we plan to study the complexity of deciding whether formulae are characteristic in the extensions of the modal logics we have considered in this article with greatest fixed points. Indeed, in those extended languages, one can define characteristic formulae for finite processes. It is known that deciding whether a formula is characteristic is PSPACE-complete for **HML**, but becomes EXP-complete for its extension with fixed-point operators – see reference [2]. It would be interesting to see whether similar results hold for the other logics. Finally, building on the work presented in [3], we plan to study the complexity of the algorithmic questions considered in this article for (some of) the linear-time semantics in van Glabbeek’s spectrum.

References

- 1 Luca Aceto, Antonis Achilleos, Aggeliki Chalki, and Anna Ingólfssdóttir. The complexity of deciding characteristic formulae in van glabbeek’s branching-time spectrum. *CoRR*, abs/2405.13697, 2024. doi:10.48550/arXiv.2405.13697.
- 2 Luca Aceto, Antonis Achilleos, Adrian Francalanza, and Anna Ingólfssdóttir. The complexity of identifying characteristic formulae. *J. Log. Algebraic Methods Program.*, 112:100529, 2020. doi:10.1016/j.jlamp.2020.100529.
- 3 Luca Aceto, Dario Della Monica, Ignacio Fábregas, and Anna Ingólfssdóttir. When are prime formulae characteristic? *Theor. Comput. Sci.*, 777:3–31, 2019. doi:10.1016/j.tcs.2018.12.004.
- 4 Luca Aceto, Ignacio Fábregas, David de Frutos-Escrig, Anna Ingólfssdóttir, and Miguel Palomino. Graphical representation of covariant-contravariant modal formulae. In Bas Luttik and Frank Valencia, editors, *Proceedings 18th International Workshop on Expressiveness in Concurrency, EXPRESS 2011, Aachen, Germany, 5th September 2011*, volume 64 of *EPTCS*, pages 1–15, 2011. doi:10.4204/EPTCS.64.1.
- 5 Luca Aceto, Anna Ingólfssdóttir, Kim Guldstrand Larsen, and Jiri Srba. *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, USA, 2007.

- 6 Luca Aceto, Anna Ingólfssdóttir, Paul Blain Levy, and Joshua Sack. Characteristic formulae for fixed-point semantics: a general framework. *Math. Struct. Comput. Sci.*, 22(2):125–173, 2012. doi:10.1017/S0960129511000375.
- 7 Antonis Achilleos. The completeness problem for modal logic. In *Proc. of Logical Foundations of Computer Science - International Symposium, LFCS 2018*, volume 10703 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2018. doi:10.1007/978-3-319-72056-2_1.
- 8 Benjamin Bisping, David N. Jansen, and Uwe Nestmann. Deciding all behavioral equivalences at once: A game for linear-time-branching-time spectroscopy. *Log. Methods Comput. Sci.*, 18(3), 2022. doi:10.46298/lmcs-18(3:19)2022.
- 9 Andreas Blass and Yuri Gurevich. On the unique satisfiability problem. *Inf. Control.*, 55(1-3):80–88, 1982. doi:10.1016/S0019-9958(82)90439-9.
- 10 Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995. doi:10.1145/200836.200876.
- 11 Gérard Boudol and Kim Guldstrand Larsen. Graphical versus logical specifications. *Theor. Comput. Sci.*, 106(1):3–20, 1992. doi:10.1016/0304-3975(92)90276-L.
- 12 Michael C. Browne, Edmund M. Clarke, and Orna Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theor. Comput. Sci.*, 59:115–131, 1988. doi:10.1016/0304-3975(88)90098-9.
- 13 Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Comb.*, 12(4):389–410, 1992. doi:10.1007/BF01305232.
- 14 Rance Cleaveland. On automatically explaining bisimulation inequivalence. In Edmund M. Clarke and Robert P. Kurshan, editors, *Computer Aided Verification, 2nd International Workshop, CAV '90*, volume 531 of *Lecture Notes in Computer Science*, pages 364–372. Springer, 1990. doi:10.1007/BFb0023750.
- 15 Rance Cleaveland and Bernhard Steffen. Computing behavioural relations, logically. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, volume 510 of *Lecture Notes in Computer Science*, pages 127–138. Springer, 1991. doi:10.1007/3-540-54233-7_129.
- 16 David de Frutos-Escrig, Carlos Gregorio-Rodríguez, Miguel Palomino, and David Romero-Hernández. Unifying the linear time-branching time spectrum of process semantics. *Log. Methods Comput. Sci.*, 9(2), 2013. doi:10.2168/LMCS-9(2:11)2013.
- 17 Rocco De Nicola and Frits W. Vaandrager. Three logics for branching bisimulation. *J. ACM*, 42(2):458–487, 1995. doi:10.1145/201019.201032.
- 18 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM J. Comput.*, 24(4):873–921, 1995. doi:10.1137/S0097539792228228.
- 19 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
- 20 Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical logic (2. ed.)*. Undergraduate texts in mathematics. Springer, 1994.
- 21 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.
- 22 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. doi:10.1016/b978-044482830-9/50019-9.
- 23 Susanne Graf and Joseph Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Inf. Control.*, 68(1-3):125–145, 1986. doi:10.1016/S0019-9958(86)80031-6.
- 24 Jan Friso Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Inf. Comput.*, 100(2):202–260, 1992. doi:10.1016/0890-5401(92)90013-6.
- 25 Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985. doi:10.1145/2455.2460.

- 26 Sören Holmström. A refinement calculus for specifications in Hennessy-Milner logic with recursion. *Formal Aspects Comput.*, 1(3):242–272, 1989. doi:10.1007/BF01887208.
- 27 Harry B. Hunt III, Daniel J. Rosenkrantz, and Thomas G. Szymanski. On the equivalence, containment, and covering problems for the regular and context-free languages. *J. Comput. Syst. Sci.*, 12(2):222–268, 1976. doi:10.1016/S0022-0000(76)80038-4.
- 28 Neil Immerman. *Descriptive Complexity*. Springer, 1999. doi:10.1007/978-1-4612-0539-5.
- 29 Anna Ingólfssdóttir, Jens Christian Godskesen, and Michael Zeeberg. Fra Hennessy-Milner logik til CCS-processor. Master’s thesis, Aalborg University, 1987. In Danish.
- 30 Sandra Kiefer, Pascal Schweitzer, and Erkal Selman. Graphs identified by logics with counting. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*, volume 9234 of *Lecture Notes in Computer Science*, pages 319–330. Springer, 2015. doi:10.1007/978-3-662-48057-1_25.
- 31 Dexter Kozen. Results on the propositional μ -calculus. *Theor. Comput. Sci.*, 27:333–354, 1983. doi:10.1016/0304-3975(82)90125-6.
- 32 Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977. doi:10.1137/0206033.
- 33 Kim Guldstrand Larsen. Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theor. Comput. Sci.*, 72(2&3):265–288, 1990. doi:10.1016/0304-3975(90)90038-J.
- 34 Kim Guldstrand Larsen and Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991. doi:10.1016/0890-5401(91)90030-6.
- 35 Jan Martens and Jan Friso Groote. Computing minimal distinguishing Hennessy-Milner formulas is NP-hard, but variants are tractable. In Guillermo A. Pérez and Jean-François Raskin, editors, *34th International Conference on Concurrency Theory, CONCUR 2023*, volume 279 of *LIPICs*, pages 32:1–32:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.CONCUR.2023.32.
- 36 Robin Milner. An algebraic definition of simulation between programs. In D. C. Cooper, editor, *Proceedings of the 2nd International Joint Conference on Artificial Intelligence, IJCAI 1971*, pages 481–489. William Kaufmann, 1971. URL: <http://ijcai.org/Proceedings/71/Papers/044.pdf>.
- 37 Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.
- 38 Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of facets (and some facets of complexity). *J. Comput. Syst. Sci.*, 28(2):244–259, 1984. doi:10.1016/0022-0000(84)90068-0.
- 39 Bernhard Steffen and Anna Ingólfssdóttir. Characteristic formulae for processes with divergence. *Inf. Comput.*, 110(1):149–163, 1994. doi:10.1006/inco.1994.1028.
- 40 Wolfgang Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science. In Marie-Claude Gaudel and Jean-Pierre Jouannaud, editors, *TAPSOFT’93: Theory and Practice of Software Development, International Joint Conference CAAP/FASE*, volume 668 of *Lecture Notes in Computer Science*, pages 559–568. Springer, 1993. doi:10.1007/3-540-56610-4_89.