The Complexity of Identifying Characteristic Formulae☆,☆☆

Luca Aceto[a,b], Antonis Achilleos[a], Adrian Francalanza[c], Anna Ingólfsdóttir[a]

[a] *School of Computer Science, Reykjavik University, Reykjavik, Iceland*
[b] *Gran Sasso Science Institute, L'Aquila, Italy*
[c] *Dept. of Computer Science, ICT, University of Malta, Msida, Malta*

Abstract

We introduce the completeness problem for Modal Logic (possibly with fixpoint operators) and examine its complexity. A formula is called complete, if any two satisfying processes are bisimilar. The completeness problem is closely connected to the characterization problem, which asks whether a given formula characterizes a given process up to bisimulation equivalence. We discover that completeness, characterization, and validity have the same complexity — with some exceptions for which there are, in general, no complete formulae. To prove our upper bounds, we present a non-deterministic procedure with an oracle for validity that combines tableaux and a test for bisimilarity, and determines whether a formula is complete.

Keywords: Modal Logic, μ-calculus, Hennessy-Milner Logic with Recursion, Completeness, Characteristic Formulae, Computational Complexity, Bisimulation

1. Introduction

We propose and study two related problems, parameterized with respect to a logic: the completeness and the characterization problem. We prove matching upper and lower complexity bounds for these problems for a variety of modal logics [5] interpreted over a Labelled Transition System (LTS), including the μ-calculus [19], and a collection of well-known epistemic logics [9].

Characteristic formulae are formulae that characterize a process up to some notion of behavioural equivalence or preorder, which in our case is bisimilarity [23]: a formula φ is characteristic for a process p when every process q is bisimilar

☆Part of this work has already appeared as [2].

☆☆This research was supported by the project "TheoFoMon: Theoretical Foundations for Monitorability" (grant number: 163406-051) and the project "Epistemic Logic for Distributed Runtime Monitoring" (grant number: 184940-051) of the Icelandic Research Fund.

Email addresses: `luca@ru.is` (Luca Aceto), `antonios@ru.is` (Antonis Achilleos), `adrian.francalanza@um.edu.mt` (Adrian Francalanza), `annai@ru.is` (Anna Ingólfsdóttir)

to p exactly when it satisfies φ. A construction of characteristic formulae for variants of CCS processes [23] was introduced in [14]. This construction allows one to verify that two CCS processes are equivalent by reducing this problem to model checking. Similar constructions were studied later in, for instance, [29, 24, 1]. We are interested in detecting when a formula is characteristic for a certain process. We call this the characterization problem and we determine its complexity for a representative collection of logics, including a selection of modal logics without recursion and the μ-calculus [19].

The completeness problem is similar to the characterization problem. For a logic $\mathbf{L}$ that is interpreted over an LTS, we call a formula φ *complete* when for every formula ψ on the same propositional variables as φ, we can derive from φ in $\mathbf{L}$ either the formula ψ or its negation. Asking whether a formula is complete for a modal logic is the same as asking if any two processes that satisfy it are bisimilar to each other. Therefore, a complete formula is characteristic if and only if it is satisfiable. As we see in the following sections, the completeness and characterization problems, both tend to have the same complexity as validity.

Given Modal Logic's wide applicability and the importance of logical completeness in general, we find it surprising that, to the best of our knowledge, the completeness problem for Modal Logic has not been studied as a computational problem so far. On the other hand, the complexity of satisfiability (and thus validity) for Modal Logic has been studied extensively — for example, see [20, 16, 15]. We examine the completeness problem for several well-known modal logics, namely the extensions of $\mathbf{K}$ by the axioms Factivity, Consistency, Positive Introspection, and Negative Introspection (also known as T, D, 4, and 5, respectively) — i.e. the ones between $\mathbf{K}$ and $\mathbf{S5}$— and their multi-modal variations. We discover that the complexity of validity and completeness tend to be the same: the completeness problem is coNP-complete if the logic has Negative Introspection and and only one box and one diamond modality (which is the case in classic Modal Logic), and it is PSPACE-complete otherwise. There are exceptions: for certain logics (namely $\mathbf{D}$, $\mathbf{T}$, and the multi-modal versions of their extensions), the completeness problem as we define it is trivial, as these logics have no finite complete theories. On the other hand, when we add the μ-calculus recursive operators to the modal language, we can write complete formulae for all of these logics, an observation that is consistent with [18], where Ingólfsdóttir et al. show that the fragment of the μ-calculus that only uses greatest fixpoints suffices to construct characteristic formulae for any state of any finite model.

Part of our motivation for studying the completeness problem also comes from [3], where Artemov raises the following issue. It is standard practice in Game Theory (and Epistemic Game Theory) to reason about a game based on a model of the game description. However, in an epistemic setting it is often the case that the game specification is not complete; thus any conclusions reached by examining any single model are precarious. Therefore, Artemov argues for the need to verify the completeness of game descriptions, and proposes a syntactic, proof-centered approach, which is more robust and general than a model-theoretic one, and which is based on a syntactic formal description of the game. Artemov's approach is more sound, in that it allows one to draw only conclusions that can be safely derived from the game specification; on the other hand, the model-based approach has been largely successful in Game Theory for a long time. He explains that if we can determine that the syntactic specification of a game is complete, then the syntactic and semantic approaches are equivalent and we can describe the game efficiently, using one model. Using the previously-introduced terminology from concurrency theory, the game specification is, in that case, characteristic for that model modulo bisimilarity.

For a formula/specification φ (for example, a syntactic description of a game), if we are interested in the formulae we can derive from φ (the conclusions we can draw from the game description), knowing that φ is complete can give a significant computational advantage. If φ is complete and consistent, for a model $\mathcal{M}$ for φ, formula ψ can be derived from φ exactly when ψ is satisfied in $\mathcal{M}$ at the same state as φ. Thus, knowing that φ is complete allows us to reduce a derivability problem to a model checking problem, which is easier to solve (see, for example, [15]). This approach may be useful when we need to examine multiple conclusions, especially if the model for φ happens to be small. On the other hand, if we discover that φ is incomplete, then it may need to be refined, as a specification.

Normal forms for Modal Logic were introduced by Fine [10] and they can be used to prove soundness, completeness, and the finite frame property for several modal logics with respect to their classes of frames. Normal forms are modal formulae that completely describe the behaviour of a Kripke model up to a given distance d from a state, with respect to a number of propositional variables. Therefore, every complete formula without fixpoint operators is equivalent to a normal form, but not all normal forms are complete, as they may be agnostic with respect to states located further than d steps away.

We focus on a definition of completeness that emphasizes the formula's ability to either affirm or reject every possible conclusion. We can define that a formula is complete up to depth d for logic $\mathbf{L}$ when it is equivalent to a normal form of modal depth (the nesting depth of a formula's modalities) at most d. We can then consider a version of the completeness problem that asks one to determine if a formula is complete up to a certain depth. If we are interested in completely describing a setting, the definition we use for completeness is more appropriate. However, it is not hard to imagine situations where this variation of completeness is the notion that fits better, either as an approximation of the epistemic depth agents reason with, or, perhaps, as a description of process behaviour for a limited amount of time. We examine this variation in Section 7.

Overview. In Section 2, we give the necessary background and definitions. Before delving into the complexity of the completeness problem for a logic, we must first answer a more fundamental question: does this logic have *any* complete formulae, or are we wasting effort? Section 3 answers this question for each of the logics that we define in the paper. Section 4 focuses on logics over one action, with Negative Introspection, but without recursive operators. As that section demonstrates, Negative Introspection imposes a special structure on the the models for the logic, which affects our algorithms and sometimes reduces the complexity of the problem. Section 5 gives a general lower bound for the complexity of the completeness problem, and upper bounds for logics without Negative Introspection — but possibly with recursive operators — by a non-deterministic procedure that uses an oracle from the logic's validity problem. Section 6 gives an alternative procedure to decide the completeness problem for multi-modal logics with Negative Introspection. Section 7 examines certain variations of the completeness problem, including a discussion of Fine's normal forms [10], and concludes the paper. The results of Sections 3, 4, 5, and 6 are summarized in Table 2.

Differences from [2]. This paper's results about one-action modal logics without recursive operators have already appeared in [2]. In this paper, we extend the results of [2] to multi-modal logics that may have recursive operators; we give a simpler procedure to decide the completeness (and characterization) problem; moreover, we make the connection between the completeness and characterization problems more prominent.

2. Background

This section introduces the logics that we focus on and the problems that we examine in this paper, as well as necessary background on bisimulation and on the complexity of the validity problem for Modal Logic and the μ-calculus. We start by defining the formulae of our logics.

Definition 1. *We consider formulae constructed from the following grammar:*

$$\begin{aligned}\varphi,\psi \in L(P) ::= p \quad &|\ \neg p \quad |\ \mathtt{tt} \quad |\ \mathtt{ff} \quad |\ X \quad |\ \varphi\wedge\psi \quad |\ \varphi\vee\psi\\ &|\ \langle\alpha\rangle\varphi \quad |\ [\alpha]\varphi \quad |\ \mu X.\varphi \quad |\ \nu X.\varphi,\end{aligned}$$

where X comes from a countably infinite set of logical variables, $\textsc{LVr}$*, α from a finite set of actions,* $\textsc{Act}$*, and p from a finite set of propositional variables, P. Let* $\textsc{PVr}$ *be the (countably infinite) set of* all *propositional variables; therefore, $P \subseteq \textsc{PVr}$. When $\textsc{Act} = \{\alpha\}$, we may use $\Box\varphi$ instead of $[\alpha]\varphi$, and $\Diamond\varphi$ instead of $\langle\alpha\rangle\varphi$. We may write $[\textsc{Act}]\varphi$ to mean $\bigwedge_{\alpha\in\textsc{Act}}[\alpha]\varphi$.*

$L(P)$ is a multi-modal version of the language of the μ-calculus [19] — or an extension of μHML [21] with propositional variables. It is the most general language that we consider in this paper.

We interpret formulae on the states of a labelled transition system (LTS). An LTS, or model, is a quadruple $\langle \textsc{Prc}, \textsc{Act}, \rightarrow, V\rangle$ where $\textsc{Prc}$ is a set of states or processes, $\textsc{Act}$ is the set of actions, $\rightarrow \subseteq \textsc{Prc}\times\textsc{Act}\times\textsc{Prc}$ is a transition relation, and $V : \textsc{Prc} \rightarrow 2^{\textsc{PVr}}$ determines on which states a propositional variable is true. For $P \subseteq \textsc{PVr}$ a finite set of variables, $V_P : \textsc{Prc} \rightarrow 2^P$ is the restriction of V on P: $V_P(s) = V(s)\cap P$. For simplicity, we assume that $\textsc{Prc}$ is always finite.

$$\llbracket \texttt{tt}, \rho \rrbracket_{\mathcal{M}} = \text{PRC}, \qquad \llbracket \texttt{ff}, \rho \rrbracket_{\mathcal{M}} = \emptyset, \qquad \llbracket X, \rho \rrbracket_{\mathcal{M}} = \rho(X),$$
$$\llbracket \varphi_1 \wedge \varphi_2, \rho \rrbracket_{\mathcal{M}} = \llbracket \varphi_1, \rho \rrbracket_{\mathcal{M}} \cap \llbracket \varphi_2, \rho \rrbracket_{\mathcal{M}}, \qquad \llbracket \varphi_1 \vee \varphi_2, \rho \rrbracket_{\mathcal{M}} = \llbracket \varphi_1, \rho \rrbracket_{\mathcal{M}} \cup \llbracket \varphi_2, \rho \rrbracket_{\mathcal{M}},$$
$$\llbracket [\alpha]\varphi, \rho \rrbracket_{\mathcal{M}} = \left\{ s \mid \forall t.\ s \xrightarrow{\alpha} t \text{ implies } t \in \llbracket \varphi, \rho \rrbracket_{\mathcal{M}} \right\},$$
$$\llbracket \langle\alpha\rangle\varphi, \rho \rrbracket_{\mathcal{M}} = \left\{ s \mid \exists t.\ s \xrightarrow{\alpha} t \text{ and } t \in \llbracket \varphi, \rho \rrbracket_{\mathcal{M}} \right\},$$
$$\llbracket \mu X.\varphi, \rho \rrbracket_{\mathcal{M}} = \bigcap \left\{ S \mid S \supseteq \llbracket \varphi, \rho[X \mapsto S] \rrbracket_{\mathcal{M}} \right\}, \qquad \llbracket p, \rho \rrbracket_{\mathcal{M}} = \{ s \mid p \in V(s) \},$$
$$\llbracket \nu X.\varphi, \rho \rrbracket_{\mathcal{M}} = \bigcup \left\{ S \mid S \subseteq \llbracket \varphi, \rho[X \mapsto S] \rrbracket_{\mathcal{M}} \right\}, \qquad \llbracket \neg p, \rho \rrbracket_{\mathcal{M}} = \{ s \mid p \notin V(s) \}.$$

Table 1: LTS semantics

State **nil** represents any state that cannot transition anywhere: $\forall\alpha\forall s.\ \texttt{nil} \stackrel{\alpha}{\not\rightarrow} s$. The set of reachable sets from state s by any sequence of 0 or more transitions is called $Reach(s)$, which we assume to be finite for each s. The size of s is $|s| = |Reach(s)|$, and $|\varphi|$ is the length of φ as a string of symbols. All our complexity results are with respect to these measures.

Formulae are evaluated in the context of an LTS $\mathcal{M}$ and an environment, $\rho : \text{LVAR} \to 2^{\text{PRC}}$, which gives values to the logical variables. For an environment ρ, variable X, and set $S \subseteq \text{PRC}$, we write $\rho[X \mapsto S]$ for the environment that maps X to S and all $Y \neq X$ to $\rho(Y)$. The semantics for our formulae is given through a function $\llbracket - \rrbracket_{\mathcal{M}}$, defined in Table 1. The negation $\neg\varphi$ of a formula and implication $\varphi \supset \psi$ (to be read as "φ implies ψ") are constructed as usual, where $\llbracket \neg X, \rho \rrbracket_{\mathcal{M}} = \text{PRC} \backslash \rho(X)$. A formula is closed when every occurrence of a variable X is in the scope of recursive operator νX or μX. Henceforth we consider only closed formulae. As the environment has no effect on the semantics of a closed formula φ, we write $\mathcal{M}, s \models_{\mathcal{M}} \varphi$ for $s \in \llbracket \varphi, \rho \rrbracket_{\mathcal{M}}$. If $s \models_{\mathcal{M}} \varphi$, we say that φ is true, or satisfied, in s. For each of the logics that we consider in this paper, we assume a fixed LTS that contains all the possible finite behaviours and only those. That is, we can think of the fixed LTS for the logic as the collection of (isomorphic copies of) all other LTSs that are suitable for the logic. When the particular LTSs do not matter, we often omit them from the notation, and we can assume that we work in that fixed LTS.

For a formula φ, $P(\varphi)$ is the set of propositional variables that appear in φ; $sub(\varphi)$ is the set of subformulae of φ and $\overline{sub}(\varphi) = sub(\varphi) \cup \{\neg\psi \mid \psi \in sub(\varphi)\}$. For Φ a nonempty finite set of formulae, $\bigwedge \Phi$ is a conjunction of all elements of Φ and $\bigwedge \emptyset = \texttt{tt}$; we define $\bigvee \Phi$ similarly. The modal depth $md(\varphi)$ of a formula φ without recursion is the largest nesting depth of its modal operators:

$$md(p) = md(\neg p) = md(\texttt{ff}) = 0;$$
$$md(\varphi \wedge \psi) = md(\varphi \vee \psi) = \max\{md(\varphi), md(\psi)\}; \quad \text{and}$$
$$md([\alpha]\varphi) = md(\langle\alpha\rangle\varphi) = md(\varphi) + 1.$$

For every $d \geq 0$, $\overline{sub}_d(\varphi) = \{\psi \in \overline{sub}(\varphi) \mid md(\psi) \leq d\}$.

Depending on how we further restrict our syntax, and the LTS, we can describe several logics. Without further restrictions, the resulting logic is the μ-calculus [19]. The max-fragment (resp. min-fragment) of the μ-calculus is the fragment that only allows the νX (resp. the μX) recursive operator. If $|\textsc{Act}| = k$ and we allow no recursive operators (or recursion variables), then we have the basic modal logic $\mathbf{K}_k$, and further restrictions on the LTS can result in a wide variety of modal logics (see [4]).

We give names to the following LTS constraints.[1] For every $\alpha \in \textsc{Act}$:

D**:** there is no `nil` state in the LTS — in other words, $\xrightarrow{\alpha}$ is serial;

T**:** $\xrightarrow{\alpha}$ is reflexive — in other words, $\forall s.\ s \xrightarrow{\alpha} s$;

4**:** $\xrightarrow{\alpha}$ is transitive — in other words, $\forall s, t, r.\ (s \xrightarrow{\alpha} t \xrightarrow{\alpha} r \ \Rightarrow\ s \xrightarrow{\alpha} r)$;

5**:** $\xrightarrow{\alpha}$ is euclidean — in other words, $\forall s, t, r.$ if $s \xrightarrow{\alpha} t$ and $s \xrightarrow{\alpha} r$, then $t \xrightarrow{\alpha} r$.

We note that in any LTS that satisfies constraints T, 4, and 5, each $\xrightarrow{\alpha}$ is an equivalence relation. Though, as we will see in Section 4, even with only constraints T and 5, we can show that each $\xrightarrow{\alpha}$ is an equivalence relation.

We consider modal logics that are interpreted over LTSs that satisfy a combination of these constraints. Of course, not all combinations make sense: D, which we call Consistency, is a special case of T, called Factivity. Constraint 4 is Positive Introspection and 5 is called Negative Introspection. Given a logic $\mathbf{L}$ and constraint c, $\mathbf{L}+c$ is the logic that is interpreted over all LTSs that satisfy all the constraints of $\mathbf{L}$ and c. Logic $\mathbf{D}_k$ is $\mathbf{K}_k + D$, $\mathbf{T}_k$ is $\mathbf{K}_k + T$, $\mathbf{K4}_k = \mathbf{K}_k + 4$, $\mathbf{D4}_k = \mathbf{K}_k + D + 4 = \mathbf{D}_k + 4$, $\mathbf{S4}_k = \mathbf{K}_k + T + 4 = \mathbf{T}_k + 4 = \mathbf{K4}_k + T$, $\mathbf{KD45}_k = \mathbf{D4}_k + 5$, and $\mathbf{S5}_k = \mathbf{S4}_k + 5$. When $k = 1$, we usually omit it from the subscript of a logic's name. For $\mathbf{L}$ being one of the logics above, $\mathbf{L}^\mu$ is the logic that results from $\mathbf{L}$ after we allow recursive operators in the syntax. Therefore, the μ-calculus is $\mathbf{K}_k^\mu$.

From now on, unless we explicitly say otherwise, by a logic or modal logic, we mean one of the logics we have defined above.

For a logic $\mathbf{L}$ and formulae φ, ψ, we say that ψ is a consequence of φ in $\mathbf{L}$ and write $\varphi \models_{\mathbf{L}} \psi$ when for every state s of every LTS $\mathcal{M}$ for $\mathbf{L}$, $\mathcal{M}, s \models \varphi$ implies $\mathcal{M}, s \models \psi$. We call a formula satisfiable (*reps.* valid) for a logic $\mathbf{L}$, if it is satisfied in some (*reps.* every) state of an LTS (*resp.* of every LTS) for $\mathbf{L}$. We opt for a semantic presentation and therefore do not provide any proof systems for the logics we examine. The following Theorem 1 justifies our approach. For the case of the μ-calculus, the theorem comes from [19, 30, 32]; for the other logics, it has a long history and the reader can consult [7, 5].

Theorem 1. *Let* $\mathbf{L}$ *be either the* μ*-calculus or one of the logics without recursive operators. A formula* φ *is valid for* $\mathbf{L}$ *if and only if it is provable in* $\mathbf{L}$*;* φ *is satisfiable for* $\mathbf{L}$ *if and only if it is satisfied in a state of a finite LTS for* $\mathbf{L}$*.*

---

[1] These conditions correspond to the usual axioms for normal modal logics — see [5, 4, 9].

2.1. Bisimulation

A classic and important notion of equivalence over states in a variety of state-transition models is bisimilarity. A binary relation $\mathcal{R}$ is a *bisimulation* (respectively, bisimulation modulo P) when the following conditions are satisfied for all $(s, s') \in \mathcal{R}$:

- $V(s) = V'(s')$ (resp. $V_P(s) = V'_P(s')$).
- For all α and t such that $s \xrightarrow{\alpha} t$, there exists some t' s.t. $(t, t') \in \mathcal{R}$ and $s' \xrightarrow{\alpha} t'$.
- For all α and t' such that $s' \xrightarrow{\alpha} t'$, there exists some t s.t. $(t, t') \in \mathcal{R}$ and $s \xrightarrow{\alpha} t$.

We call states s and s' *bisimilar* (resp. bisimilar modulo P) and write $s \sim s'$ (resp. $s \sim_P s'$) if there is a bisimulation (resp. bisimulation modulo P) $\mathcal{R}$ such that $s\mathcal{R}s'$. When s and s' are states of different LTSs $\mathcal{M}$ and $\mathcal{M}'$, respectively, we write $(\mathcal{M}, s) \sim (\mathcal{M}', s')$ (resp. $(\mathcal{M}, s) \sim_P (\mathcal{M}', s')$) instead of $s \sim s'$ (resp. $s \sim_P s'$), to specify that each state behaves according to the corresponding LTS. However, the LTS is omitted when it is the same, or it does not matter.

The following simplification of the Hennessy-Milner Theorem [17] gives a very useful characterization of state equivalence; Proposition 3 is its direct consequence.

Theorem 2 (Hennessy-Milner Theorem)**.** *$s \sim_P s'$ if and only if s and s' satisfy the same formulae (without recursion operators) in $L(P)$.*[2]

Definition 2. *We call a formula φ* characteristic *for state s when for every state t in an LTS for $\mathbf{L}$, $s \sim_{P(\varphi)} t$ if and only if $t \models \varphi$. A formula φ is called* complete *when for every $\psi \in L(P(\varphi))$, ψ or $\neg\psi$ is a consequence of φ in $\mathbf{L}$.*

Proposition 3. *A formula φ is complete for a logic $\mathbf{L}$ if and only if for every two LTSs $\mathcal{M}$ and $\mathcal{M}'$ for $\mathbf{L}$, and states $s \in [\![\varphi]\!]_{\mathcal{M}}$ and $s' \in [\![\varphi]\!]_{\mathcal{M}'}$, $(\mathcal{M}, s) \sim_{P(\varphi)} (\mathcal{M}', s')$.*

Paige and Tarjan in [27] give an efficient algorithm for checking whether two states are bisimilar. Proposition 4 is a variation on their result to account for receiving the set P of propositional variables as part of the algorithm's input.

Proposition 4. *There is an algorithm which, given two states s and s' and a finite set of propositional variables P, determines whether $s \sim_P s'$ in time $O(|P| \cdot (|s|^2 + |s'|^2) \cdot \log(|s| + |s'|))$.*

Definition 3 (Characterization, Completeness)**.** *The characterization problem for $\mathbf{L}$ is the following:* Given a formula φ and a state s, is φ characteristic for s? *The completeness problem for $\mathbf{L}$ is:* Given a formula φ, is φ complete?

[2]We remark that we assume only LTSs that are finite — and therefore also image-finite.

2.2. The Complexity of Satisfiability

For logic **L**, the satisfiability problem for **L**, or **L**-satisfiability is the problem that asks, given a formula φ, if φ is satisfiable. Similarly, the validity problem for **L** asks if φ is valid.

The classical complexity results for Modal Logic are due to Ladner [20], who established PSPACE-completeness for the satisfiability of **K**, **T**, **D**, **K4**, **D4**, and **S4** and NP-completeness for the satisfiability of **S5**. Halpern and Rêgo later characterized the NP–PSPACE gap for one-action logics by the presence or absence of Negative Introspection [16], resulting in Theorem 5.

Theorem 5 ([20, 16])**.** *If* $\mathbf{L} \in \{\mathbf{K}, \mathbf{T}, \mathbf{D}, \mathbf{K4}, \mathbf{D4}, \mathbf{S4}\}$*, then* $\mathbf{L}$*-validity is* PSPACE*-complete and* $\mathbf{L} + 5$*-validity is* coNP*-complete.*

Theorem 6 ([15])**.** *If* $k > 1$ *and* **L** *has a combination of constraints from* $D, T, 4, 5$ *and no recursive operators, then* $\mathbf{L}_k$*-validity is* PSPACE*-complete.*

Remark 1. We note that Halpern and Moses in [15] only prove these bounds for the cases of $\mathbf{K}_k, \mathbf{T}_k, \mathbf{S4}_k, \mathbf{KD45}_k$, and $\mathbf{S5}_k$. However, it is not hard to see that their methods also work for the rest of the logics of Theorem 6. ■

The following theorems give the complexity of validity for the μ-calculus and its fragments.

Theorem 7 ([19])**.** *The validity problem for the* μ*-calculus is* EXP*-complete.*

Proposition 8. *The validity problem for the min- and max-fragments of the* μ*-calculus is* EXP*-complete, even when* $|\text{Act}| = 1$ *and* $P = \emptyset$*.*

Proof sketch. It is known that satisfiability for the min- and max-fragments of the μ-calculus (on one or more action) is EXP-complete. It is in EXP due to Theorem 7, and these fragments suffice [28] to describe the PDL formula that is constructed by the reduction used in [11] to prove EXP-hardness for PDL. Therefore, that reduction can be adjusted to prove that satisfiability for the min- and max-fragments of the μ-calculus is EXP-complete. It follows that validity for the min- and max-fragments of the μ-calculus (on one or more action) is also EXP-complete. To see that this lower bound transfers to the fragment with one action and without variables, we can encode a one-action formula with k variables into one without any, by replacing each $\Box$ by $\Box^{k+1}$ and each $\Diamond$ by $\Diamond^{k+1}$, and for each $1 \leq i \leq k$, p_i by $\Diamond^i\Box\mathtt{ff}$. It is then not hard to see that this encoding is satisfiability-preserving. □

To the best of our knowledge, there are no complexity results for the validity of logics with both LTS constraints and recursion operators. However, it is not hard to express in such logics that formula φ is common knowledge, with formula $\nu X.\varphi \wedge [\text{Act}]X$. Since validity for $\mathbf{L}_k$ with common knowledge (and without recursive operators) and $k > 1$ is EXP-complete [15][3], $\mathbf{L}_k^\mu$ must be EXP-hard.

[3]Similarly to Remark 1, [15] does not explicitly cover all these cases, but the techniques can be adjusted.

Lemma 9. *Validity for $\mathbf{L}_k^\mu$, where $k > 1$, is* EXP*-hard.*

We conjecture that validity is EXP-complete for these logics, possibly even when $k = 1$, but proving such bounds for validity falls out of the scope of this paper.

In the following, when P is evident from the context, we will often omit any reference to it and instead of bisimulation modulo P, we will call the relation simply bisimulation.

3. The Completeness Problem and Triviality

A first and fundamental question that we need to answer concerning the completeness problem for $\mathbf{L}$ is whether there are any satisfiable and complete formulae for it. If the answer is negative, then the problem is trivial. We examine this question with parameters the logic $\mathbf{L}$ and whether P, the set of propositional variables we use, is empty or not. If for some logic $\mathbf{L}$ the problem is nontrivial, then we give a complete formula $\varphi_P^{\mathbf{L}}$ that uses exactly the propositional variables in P. We see that when $P = \emptyset$, completeness can become trivial for another reason: for some logics, when $P = \emptyset$, all formulae are complete. On the other hand, when $P \neq \emptyset$, the formula $\bigwedge P$ is incomplete for every logic.

3.1. Completeness and $\mathbf{K}$

Whether $P = \emptyset$ or not, completeness is nontrivial for $\mathbf{K}_k$ and $\mathbf{K4}_k$: let $\varphi_P^{\mathbf{K}_k} = \varphi_P^{\mathbf{K4}_k} = \bigwedge P \wedge [\textsc{Act}]\mathtt{ff}$ for every finite set P of propositional variables. Formula $\mathtt{tt}$ is incomplete for $\mathbf{K}_k$ and $\mathbf{K4}_k$.

Lemma 10. *Formula $\varphi_P^{\mathbf{K}_k}$ is complete and satisfiable for $\mathbf{K}_k$ and for $\mathbf{K4}_k$.*

Proof. A state that satisfies $\varphi_P^{\mathbf{K}_k}$ is s, where $V(s) = P$ and $\forall t.\ s \not\to t$. If there is another state $s' \models \varphi_P^{\mathbf{K}_k}$, then $s' \models [\textsc{Act}]\mathtt{ff}$, so there are no transitions from s'; therefore, $\mathcal{R} = \{(s, s')\}$ is a bisimulation. □

Notice that if φ is complete for $\mathbf{L}$, then it is complete for every bisimulation-invariant extension of $\mathbf{L}$. Therefore, $\varphi_P^{\mathbf{K}_k}$ is complete for all the other logics on k actions. However, we are looking for *satisfiable and complete* formulae for the different logics, so finding one complete formula for $\mathbf{K}_k$ is not enough.

Lemma 11. *Formula $\bigwedge P \wedge [\textsc{Act}]\mathtt{ff}$ is complete and satisfiable for the μ-calculus.*

Proof. Similar to the proof of Lemma 10. □

On the other hand, if $\mathbf{L}'$ is an extension of $\mathbf{L}$ (by a set of LTS restrictions) and a formula φ is complete for $\mathbf{L}$ and satisfiable for $\mathbf{L}$, then we know that φ is satisfiable and complete for all logics between (and including) $\mathbf{L}$ and $\mathbf{L}'$. Unfortunately, the following lemma demonstrates that we cannot use this convenient observation to reuse $\varphi_P^{\mathbf{K}_k}$ — except for $\mathbf{K5}_k$ and $\mathbf{K45}_k$ (see Lemma 19).

3.2. Completeness and Consistency with no Introspection

When **L** has restriction T or D, but neither 4 nor 5, P determines if a satisfiable formula is complete.

Lemma 12. *Let* **L** *be either* $\mathbf{D}_k$ *or* $\mathbf{T}_k$*. A satisfiable formula* φ *is complete with respect to* **L** *if and only if* $P(\varphi) = \emptyset$*.*

Proof. When $P = \emptyset$, all states in an LTS for $\mathbf{D}_k$ or $\mathbf{T}_k$ are bisimilar through the total bisimulation; therefore, all formulae φ with $P(\varphi) = \emptyset$ are trivially complete as every two satisfiable formulae in $L(\emptyset)$ are logically equivalent. We now consider the case for $P \neq \emptyset$; first assume that $\mathbf{L} = \mathbf{D}_k$. Let $\mathcal{M} = \langle W, \textsc{Act}, \rightarrow, V\rangle$ be an LTS for $\mathbf{D}_k$. Let $d = md(\varphi)$ and let $s \models \varphi$; let $x \notin \mathit{Reach}(s)^*$, $s_0 = s$, and

$$\Pi_d = \{s_0 \cdots s_k \in W^* \mid k \leq d \text{ and for all } 0 \leq i < k,\ \exists \alpha \in \textsc{Act}.\ s_i \xrightarrow{\alpha} s_{i+1}\}.$$

Then, we define $\mathcal{M}_1 = \langle W', \textsc{Act}, \rightarrow', V_1'\rangle$ and $\mathcal{M}_2 = \langle W', \textsc{Act}, \rightarrow', V_2'\rangle$, where

$$\begin{aligned}
W' &= \Pi_d \cup \{x\};\\
\xrightarrow{\alpha}{}' &= \{(zt, ztt') \in W'^2 \mid t \xrightarrow{\alpha} t'\} \ \cup\ \{(s_0 s_1 \cdots s_d, x) \in W'^2\} \ \cup\ \{(x,x)\}\\
V_i'(zt) &= V(t),\ \text{for } i = 1,2,\ 0 \leq |z| < d;\\
V_1'(x) &= \emptyset;\ \text{ and }\ V_2'(x) \ =\ P.
\end{aligned}$$

If $\mathbf{L} = \mathbf{T}_k$, the difference is that the transition relation for α must be defined as the reflexive closure of $\xrightarrow{\alpha}{}'$, as defined above. The remaining arguments are the same. To prove that $\mathcal{M}_1, s \models \varphi$ and $\mathcal{M}_2, s \models \varphi$, we prove that for $\psi \in \mathit{sub}(\varphi)$, for every $i = 1,2$ and $z = s_0 \cdots s_k \in \Pi_d$, where $k \leq d - md(\psi)$,

$$\mathcal{M}_i, z \models \psi \quad \text{if and only if} \quad \mathcal{M}, s_k \models \psi.$$

We use induction on ψ. If ψ is a literal or a constant, the claim is immediate and so are the cases of the $\wedge, \vee$ connectives. If $\psi = [\alpha]\psi'$, then $md(\psi') = md(\psi) - 1$; $\mathcal{M}_i, z \models \psi$ iff for every $z \xrightarrow{\alpha}{}' z'$, $\mathcal{M}_i, z' \models \psi'$ iff for every $s_k \rightarrow' t$, $\mathcal{M}, t \models \psi'$ (by the Inductive Hypothesis) iff $\mathcal{M}, s_k \models \psi$; the case of $\psi = \langle\alpha\rangle\psi'$ is similar.

If $(\mathcal{M}_1, s) \sim (\mathcal{M}_2, s)$ through bisimulation $\mathcal{R}$, then notice that in both $\mathcal{M}_1$ and $\mathcal{M}_2$, any path of length at least $d+1$ from s will end up at x; therefore, by the conditions of bisimulation, $(\mathcal{M}_1, x)\mathcal{R}(\mathcal{M}_2, x)$, which is a contradiction, since $V_1'(x) \neq V_2'(x)$. So, φ is satisfied in two non-bisimilar states for **L**, and is thus incomplete. □

The proof of Lemma 12 relies on the fact that $\mathbf{D}_k$ and $\mathbf{T}_k$ do not allow any recursive operators, and therefore each formula can describe an LTS up to a distance equal to its modal depth. On the other hand, if we allow recursion, we can use it to characterize a single-state LTS that satisfies all propositions. Let $\varphi_P^{\mathbf{D}_k^\mu} = \varphi_P^{\mathbf{T}_k^\mu} = \nu X. \bigwedge P \wedge [\textsc{Act}]X$. This formula is satisfiable and complete for every bisimulation-invariant logic that has at least restriction D.

Lemma 13. *Formula* $\varphi_P^{\mathbf{D}_k^\mu}$ *is complete and satisfiable for all* $\mathbf{L}_k^\mu{+}D$ *and* $\mathbf{L}_k^\mu{+}T$*.*

3.3. Completeness, Consistency, and Positive Introspection

For every finite P, let $\varphi_P^{\mathbf{D4}} = \varphi_P^{\mathbf{S4}} = \bigwedge P \wedge \Box \bigwedge P$. As the following lemma demonstrates, $\varphi_P^{\mathbf{D4}}$ is a complete formula for **D4** and **S4**.

Lemma 14. *For every finite P, $\varphi_P^{\mathbf{D4}}$ is complete for* **D4** *and* **S4***; all formulae in $L(\emptyset)$ are complete for* **D4** *and* **S4**.

Proof. Let $s \models \varphi_P^{\mathbf{D4}}$ and $s' \models \varphi_P^{\mathbf{D4}}$; let $\mathcal{R} = Reach(s) \times Reach(s')$; it is not hard to verify that $\mathcal{R}$ is a bisimulation. Notice that if $P = \emptyset$, then $\varphi_P^{\mathbf{D4}}$ is a tautology, thus all formulae are complete. Proposition 3 yields that, in that case, all consistent formulae are tautologies. In fact, $\forall s, s'.\ s \sim_\emptyset s'$, so if $[\![\varphi]\!] \neq \emptyset$, then $[\![\varphi]\!] = \textsc{Prc}$. □

It is straightforward to see that for one action, $\varphi_P^{\mathbf{D4}}$ is satisfiable for every modal logic **L**: consider a state a, where $\bigwedge P$ holds at every state in $Reach(a)$. Therefore:

Corollary 15. *$\varphi^{\mathbf{D4}}$ is satisfiable and complete for every consistent*[4] *logic on one action that extends* **D4** *by a set of conditions for the LTS transitions.*[5]

When $k > 1$, the situation is similar as for **D** and **T**.

Lemma 16. *Let* **L** *be either* $\mathbf{D4}_k$ *or* $\mathbf{S4}_k$*, where $k > 1$. A satisfiable formula φ is complete with respect to* **L** *if and only if $P(\varphi) = \emptyset$.*

Proof. The proof is similar to the one for Lemma 12, only we change the definitions of Π_d, W', and $\xrightarrow{\alpha}'$.

$$\Pi_d = \{(s_0, \alpha_0) \cdots (s_k, \alpha_k) \in (W \times \textsc{Act})^* \mid \text{for all } 0 \leq i < k,\ s_i \xrightarrow{\alpha_i} s_{i+1}\}.$$

For $(s_0, \alpha_0) \cdots (s_k, \alpha_k) \in \Pi_d$, $l((s_0, \alpha_0) \cdots (s_k, \alpha_k))$ is defined recursively:

$$\begin{aligned} l((s_0, \alpha_0)) &= 0; \\ l(z(s_i, \alpha_i)(s_{i+1}, \alpha_i)) &= l(z(s_i, \alpha_i)); \text{ and} \\ l(z(s_i, \alpha_i)(s_{i+1}, \alpha_{i+1})) &= l(z(s_i, \alpha_i)) + 1, \text{ if } \alpha_i \neq \alpha_{i+1}. \end{aligned}$$

Then, $W' = \{z \in \Pi_d \mid l(z) \leq d\} \cup \{x\}$; and $\xrightarrow{\alpha}'$ is the appropriate closure of $\{(z, z'(t, \alpha)) \in W'\} \cup \{(z, x) \mid z = x \text{ or } l(z) = d\}$. The remaining arguments are similar to the ones from the proof of Lemma 12. □

[4] If the logic is not consistent, then it has no models, and therefore $\varphi^{\mathbf{D4}}$ cannot be satisfiable.

[5] Although for the purposes of this paper we only consider a specific set of modal logics, it is interesting to note that the corollary can be extended to a much larger class of logics.

3.4. Consistency and Negative Introspection

For a logic $\mathbf{L} = \mathbf{L}' + 5$ on one action, let $\varphi_P^{\mathbf{L}} = \bigwedge P \wedge \Diamond\Box \bigwedge P$.

Lemma 17. *For any logic $\mathbf{L} = \mathbf{L}' + 5$ on one action, $\varphi_P^{\mathbf{L}}$ is a satisfiable complete formula for $\mathbf{L}$.*

Proof. By Lemma 21, all states are flat, and therefore $\varphi_P^{\mathbf{L}}$ is complete. It is satisfied in state a, where $V(a) = P$ and a transitions exactly to a. □

When $P = \emptyset$, we can distinguish two cases. If $\mathbf{L}' \in \{\mathbf{D}, \mathbf{D4}, \mathbf{T}, \mathbf{S4}\}$, then $\varphi_\emptyset^{\mathbf{L}}$ is a tautology, therefore all formulae in $L(P)$ are complete for $\mathbf{L}$.[6] If $\mathbf{L}' \in \{\mathbf{K}, \mathbf{K4}\}$, by Lemma 21, a state would either satisfy $\varphi_P^{\mathbf{L}}$ or $\Box\mathtt{ff}$, depending on whether it has a transition or not, because it is flat. Therefore, if $P = \emptyset$ the completeness problem for **K5** and **K45** is not trivial, but it is easy to solve: a formula with no propositional variables is complete for $\mathbf{L} \in \{\mathbf{K5}, \mathbf{K45}\}$ if it is satisfied in at most one of the two states for $\mathbf{L}$, which are non-bisimilar modulo $\emptyset$.

Corollary 18. *If $P = \emptyset$, the completeness problem for* **K5** *and* **K45** *is in* P.

Lemma 19. *Formula $\bigwedge P \wedge [\textsc{Act}]$ is satisfiable and complete for $\mathbf{K5}_k$ and $\mathbf{K45}_k$.*

Proof. A corollary of Lemma 12. □

Similarly to the cases of $\mathbf{D}_k$, $\mathbf{T}_k$, $\mathbf{D4}_k$, and $\mathbf{S4}_k$, for $k > 1$, a formula is complete for one of the corresponding logics with the addition of Negative Introspection, if and only if it has no propositional variables.

Lemma 20. *Let $\mathbf{L}$ be one of $\mathbf{D}_k$, $\mathbf{T}_k$, $\mathbf{D4}_k$, and $\mathbf{S4}_k$, where $k > 1$. Formula φ is complete for $\mathbf{L} + 5$ if and only if $P(\varphi) = \emptyset$.*

Proof. Similar to the proofs of Lemmata 12 and 16, taking the accessibility relation conditions into account. □

3.5. Completeness and Modal Logics: Summary

A logic $\mathbf{L}$ has a nontrivial completeness problem if for $P \neq \emptyset$, there are complete formulae for $\mathbf{L}$. From the logics we examined, logics $\mathbf{D}_k$ and $\mathbf{T}_k$ for $k > 0$, and multi-action versions of logics with D and T have trivial completeness problems, as long as recursion is not allowed in formulae. Table 2 summarizes the results of this section and of the following sections regarding the completeness problem. As the table demonstrates, we can distinguish the following cases. For $\mathbf{K}_k$, the completeness problem is non-trivial and PSPACE-complete; this does not change when we add axiom 4. Once we add axiom D to $\mathbf{K}_k$, but neither 4 nor 5, the completeness problem becomes trivial; adding the stronger axiom T does not change the situation. Curiously, adding both 4 and D or T to $\mathbf{K}_k$

[6] This is also a corollary of Lemma 12, as these are extensions of **D** and **T**.

Logic	$P = \emptyset$	$P \neq \emptyset$
$\mathbf{K}_k$, $\mathbf{K4}_k$	PSPACE-complete	PSPACE-complete
$\mathbf{D}_k$, $\mathbf{T}_k$	trivial (all)	trivial (none)
D4, **S4**	trivial (all)	PSPACE-complete
$\mathbf{D4}_k$, $\mathbf{S4}_k$, $k > 1$	trivial (all)	trivial (none)
K5, **K45**	in P	coNP-complete
$\mathbf{K5}_k$, $\mathbf{K45}_k$, $k > 0$	PSPACE-complete	PSPACE-complete
$\mathbf{L}_1 + 5$, $\mathbf{L}_1 \neq \mathbf{K}, \mathbf{K4}$	trivial (all)	coNP-complete
$\mathbf{L}_k + 5$, $\mathbf{L} \neq \mathbf{K}, \mathbf{K4}$, $k > 1$	trivial (all)	trivial (none)
(min-, max-) μ-calculus	EXP-complete	EXP-complete
$\mathbf{L}_k^{\mu}$, $k > 1$	varies	EXP-hard, $\mathsf{NPSPACE}^C$

Table 2: The complexity of the completeness problem for different logics. Trivial (all) indicates that all formulae in this case are complete for the logic; trivial (none) indicates that there is no satisfiable and complete formula for the logic. C is the complexity class for which the validity problem for $\mathbf{L}_k^{\mu}$ is C-complete. These results are given in Theorem 26, Proposition 27, Lemma 9, and Corollaries 30, 31, and 42.

makes the completeness problem PSPACE-complete again, except when $P = \emptyset$ or $k > 1$. Regardless of other axioms, if the logic has Negative Introspection, completeness is coNP-complete — unless $P = \emptyset$ or $k > 1$, when the situation depends on whether the logic has D (or the stronger T) or not. As Lemma 11 demonstrates, the μ-calculus and its min- and max-fragments have non-trivial completeness problems. In general, adding recursive operators to the language always allows us to write complete formulae, an observation which is consistent with the results in [18], where it is shown that the max-fragment of the μ-calculus suffices to give characteristic formulae for any state.

4. The Completeness Problem and Negative Introspection: The One-action Case

In this section, we explain how to adapt Halpern and Rêgo's techniques from [16] to prove similar complexity bounds for the completeness problem for logics on one action, no recursive operators, and with Negative Introspection. Thus, we assume in this section a logic $\mathbf{L}$ without recursive operators, and that $|\textsc{Act}| = 1$. In the course of proving the coNP upper bound for logics with Negative Introspection, Halpern and Rêgo describe in [16] a construction that they attribute to Nagle and Thomason [26], which provides a model of a particular form for a satisfiable formula. From this construction, we can extract Lemma 21 to follow.

For a logic $\mathbf{L} + 5$, we call a state s in an LTS for $\mathbf{L} + 5$ *flat* when there is a set of states W, such that:

- $Reach(s) = \{s\} \cup W$;
- the restriction of $\rightarrow$ on $Reach(s)$ is $R \cup E$, where $R \subseteq \{s\} \times W$ and E is an equivalence relation on W; and

- if $\mathbf{L} \in \{\mathbf{T}, \mathbf{S4}\}$, then $s \in W$.

Lemma 21 informs us that all states are flat for logics with restriction 5.

Lemma 21. *In an LTS with restriction* 5*, every state is a flat state.*

Proof. Let $W = \{t \in \mathit{Reach}(s) \mid \exists t' {\in} \mathit{Reach}(s).\ t' \to t\}$. Therefore $\mathit{Reach}(s) = W \cup \{s\}$ and if $\mathbf{L} \in \{\mathbf{T}, \mathbf{S4}\}$, then $s \in W$. Since restriction 5 is in effect, $\to$ is euclidean. Therefore, the restriction of $\to$ on W is reflexive. This in turn means that $\to$ is symmetric in W: if $t_1, t_2 \in W$ and $t_1 \to t_2$, since $t_1 \to t$, we also have $t_2 \to t_1$. Finally, $\to$ is transitive in W: if $t_1 \to t_2 \to t_3$ and $t_1, t_2, t_3 \in W$, then $t_2 \to t_1$, so $t_1 \to t_3$ by 5. Therefore $\to$ is an equivalence relation when restricted on W and we are done. $\square$

The construction from [20] and [16] continues to filter the states of W, resulting in a small state for a formula φ. Using this construction, Halpern and Rêgo prove Corollary 22 [16, Theorem 3.1]; the coNP upper bound for $\mathbf{L}+5$-validity of Theorem 5 is a direct consequence. We present the proof of this result for completeness, and because it is of importance for what follows.

Corollary 22. *Formula φ is $\mathbf{L}+5$-satisfiable if and only if it is satisfied in a flat state of size at most $O(|\varphi|)$ in an LTS for $\mathbf{L}+5$.*

Proof. The "if" direction is immediate. To prove the other direction, we continue from the proof of Lemma 21. If $W = \emptyset$, then we are done. Otherwise, let $S_\Diamond$ be the set which contains every $\Diamond\psi$ subformula of φ that is true in s and every $\Box\psi$ subformula of φ that is not true in s; let $T_\Diamond$ be the set which contains every $\Diamond\psi$ subformula of φ that is true in some state of $\mathit{Reach}(s)$ and every $\Box\psi$ subformula of φ that is not true in some state of $\mathit{Reach}(s)$. For every $\Diamond\psi \in S_\Diamond$ we fix a state $s_\psi \in W$, such that $s \to s_\psi$ and where ψ is true; for every $\Box\psi \in S_\Diamond$ we fix a state $s_\psi \in W$, such that $s \to s_\psi$ and where ψ is not true; for every $\Diamond\psi \in T_\Diamond \setminus S_\Diamond$ we fix a state $s_\psi \in W$, where ψ is true; finally, for every $\Box\psi \in T_\Diamond \setminus S_\Diamond$ we fix a state $s_\psi \in W$, where ψ is not true. Notice that each $s_\psi \in W$ and thus it is accessible from every state in W. We construct the LTS $\mathcal{M}_\varphi = \langle W_\varphi, \textsc{Act}, \to_\varphi, V_\varphi \rangle$ for $\mathbf{L}+5$, where

$$W_\varphi = \{s\} \cup \{s_\psi \in W \mid \Diamond\psi \in T_\Diamond \text{ or } \Box\psi \in T_\Diamond\},$$

$\to_\varphi$ the restriction of R on W_φ, and $V_\varphi(a) = V(a)$ for all $a \in W_\varphi$.

It is not hard to confirm that $|W_\varphi| \leq |\varphi|$, since $|T_\Diamond| \leq |\varphi| - 1$ (at least one of the subformulae of φ is a propositional variable or $\mathtt{ff}$). Furthermore, $\mathcal{M}_\varphi, s \models \varphi$. Specifically, for all $t \in W_\varphi$ and $\psi \in \mathit{sub}(\varphi)$, we prove by induction on ψ that $\mathcal{M}, t \models \psi$ if and only if $\mathcal{M}_\varphi, t \models \psi$, where $\mathcal{M}$ is the fixed LTS for $\mathbf{L}+5$. Propositional cases are easy. If $\psi = \Diamond\chi$ and $\mathcal{M}, t \models \psi$, then there is some t_χ, such that $\mathcal{M}, t_\chi \models \chi$ and by the definition of t_χ, $t \to t_\chi$, therefore by the inductive hypothesis, $\mathcal{M}_\varphi, t_\chi \models \chi$ and thus $\mathcal{M}_\varphi, t \models \psi$. If $\psi = \Diamond\chi$ and $\mathcal{M}_\varphi, t \models \psi$, then there is some $t \to_\varphi c \in W_\varphi$, such that $\mathcal{M}_\varphi, c \models \chi$; by the inductive hypothesis, $\mathcal{M}, c \models \chi$ and since $\to_\varphi$ is the restriction of $\to$ on W_φ, $t \to c$, so $\mathcal{M}, t \models \psi$. The cases where $\psi = \Box\chi$ are similar.

What remains is to demonstrate that $\mathcal{M}_\varphi$ remains an LTS for $\mathbf{L}+5$. It is not hard to confirm that through this filtering, transitivity, euclidicity, and reflexivity are preserved for $\to_\varphi$ (since they are preserved by restrictions on subsets of binary relations). As for seriality, it is enough to run this construction on $\varphi \wedge \Diamond \texttt{tt}$ if necessary, thus increasing the upper bound on the number of states from $|\varphi|$ to $|\varphi|+1$. $\square$

Since we ask whether a formula is complete, instead of whether it is satisfiable, we want to be able to find two small non-bisimilar states for φ when φ is incomplete. In order to do so, it is useful to have a characterization of bisimilarity between flat models.

Lemma 23. *Flat states s and s' are bisimilar modulo P if and only if $V_P(s) = V_P(s')$ and:*

- $\forall t \in \mathit{Reach}(s)\ \exists t' \in \mathit{Reach}(s').\ V_P(t) = V'_P(t')$;
- $\forall t' \in \mathit{Reach}(s')\ \exists t \in \mathit{Reach}(s).\ V_P(t) = V'_P(t')$;
- $\forall t {\in} \mathit{Reach}(s)$, *if* $s \to t$, *then* $\exists t' {\in} \mathit{Reach}(s').\ s' \to t'$ *and* $V_P(t) = V_P(t')$; *and*
- $\forall t' {\in} \mathit{Reach}(s')$, *if* $s' \to t'$, *then* $\exists t {\in} \mathit{Reach}(s).\ s \to b$ *and* $V_P(t) = V'_P(t')$.

Proof. If these conditions are met, we can define bisimulation $\mathcal{R}$ such that $s\mathcal{R}s'$ and for $t \in \mathit{Reach}(a)$ and $t' \in \mathit{Reach}(s')$, $t\mathcal{R}t'$ iff $V_P(t) = V_P(t')$; on the other hand, if there is a bisimulation relating s and s', then it is not hard to see that these conditions hold by the definition of bisimulation — for both claims, notice that the conditions above, given that the states are flat, correspond exactly to the conditions from the definition of bisimulation. $\square$

This gives us Corollary 24 below, which is a useful characterization of incomplete formulae.

Corollary 24. *Formula φ is incomplete for a logic* $\mathbf{L}$ *on one action, Negative Introspection, and with no recursive operators, if and only if there are two flat states s and s' in an LTS for* $\mathbf{L}$, *whose size is $O(|\varphi|)$ such that*

1. *s and s' satisfy φ, and*
2. *s and s' are not bisimilar modulo $P(\varphi)$.*

Proof. If φ has two non-bisimilar states, then by Proposition 3, it is incomplete. On the other hand, if φ is incomplete, again by Proposition 3 and Lemma 21, φ has two non-bisimilar flat states, s and s'. We will now construct a distinguishing formula ψ for s and s'. By Lemma 23 and without loss of generality, we can distinguish three cases:

- there is some $p \in V_P(s) \setminus V_P(s')$: in this case, let $\psi = p$;

- there is some t, such that $s \to t$ and for all t', $s' \to t'$ implies $V_P(t) \neq V_P(t')$: in this case, let

$$\psi = \Diamond(\bigwedge V_P(t) \wedge \neg \bigvee (P \setminus V_P(t)));$$

- there is some $t \in \mathit{Reach}(s)$, such that for all $t' \in \mathit{Reach}(s')$, $V_P(t) \neq V_P(t')$: in this case, let

$$\psi = \Diamond\Diamond(\bigwedge V_P(t) \wedge \neg \bigvee (P \setminus V_P(t))).$$

In all these cases, both $\varphi \wedge \psi$ and $\varphi \wedge \neg\psi$ are satisfiable and of size $O(|\varphi|)$, so by Corollary 22, each is satisfied in a non-bisimilar flat state of size $O(|\varphi|)$. $\square$

Our first complexity result is a consequence of Corollary 24 and Proposition 4:

Corollary 25. *The completeness problem for logic* **L** *on one action, with Negative Introspection, and with no recursive operators, is in* coNP*.*

5. The Complexity of Completeness

Our main result is that for a modal logic **L**, the completeness problem has the same complexity as validity for **L**, as long as we allow for propositional variables in a formula and the completeness problem for **L** is nontrivial (see also Table 2). For the lower bounds, we consider hardness under polynomial-time reductions. As the hardness results are relative to complexity classes that include coNP, these reductions suffice.

5.1. A Lower Bound

We present a lower bound for the complexity of the completeness problem: we show that the completeness problem is at least as hard as validity for a logic, as long as it is nontrivial.

Theorem 26. *Let* **L** *be a logic that has a nontrivial completeness problem and let C be a complexity class. If* **L***-validity is C-hard, then the completeness problem for* **L** *is C-hard.*

Proof. To prove the theorem we present a reduction from **L**-validity to the completeness problem for **L**. The exceptions are the min- and max-fragments of the μ-calculus, for which the reduction is from the max- and min-fragment, respectively. From a formula φ, the reduction constructs in polynomial time a formula φ_c, such that φ is provable if and only is φ_c is complete. For each logic **L** with nontrivial completeness and finite set of propositional variables P, in Section 3 we provided a complete formula $\varphi_P^{\mathbf{L}}$. This formula is satisfied in a state of size at most 2, which can be generated in time $O(|P|)$. Let $s_{\mathbf{L}}$ be such a state for $\varphi_P^{\mathbf{L}}$. We assume that $P \neq \emptyset$.

All states that satisfy $\varphi_P^{\mathbf{L}}$ are bisimilar to $s_{\mathbf{L}}$ (Proposition 3). Given a formula $\varphi \in L(P)$, we determine in linear time if $s_{\mathbf{L}} \models \varphi$. Then, we have two cases:

$s_{\mathbf{L}} \not\models \varphi$, in which case φ is not valid and we set $\varphi_c = \bigwedge P$.

$s_{\mathbf{L}} \models \varphi$, so $\neg\varphi \wedge \varphi_P^{\mathbf{L}}$ is not satisfiable, in which case we set $\varphi_c = \varphi \supset \varphi_P^{\mathbf{L}}$. We demonstrate that φ is valid if and only if $\varphi \supset \varphi_P^{\mathbf{L}}$ is complete.

If φ is valid, then $\varphi \supset \varphi_P^{\mathbf{L}}$ is equivalent to $\varphi_P^{\mathbf{L}}$, which is complete.

On the other hand, if $\varphi \supset \varphi_P^{\mathbf{L}}$ is complete and a is any state, we show that $a \models \varphi$, implying that if $\varphi \supset \varphi_P^{\mathbf{L}}$ is complete, then φ is valid. If $s \sim_P s_{\mathbf{L}}$, then from our assumptions $s \not\models \neg\varphi$, thus $s \models \varphi$. On the other hand, if $s \not\sim_P s_{\mathbf{L}}$, since $s_{\mathbf{L}} \models \varphi \supset \varphi_P^{\mathbf{L}}$ and $\varphi \supset \varphi_P^{\mathbf{L}}$ is complete, $s \not\models \varphi \supset \varphi_P^{\mathbf{L}}$, therefore $s \models \varphi$. □

Theorem 26 applies to more than the logics that we have defined in Section 2. For Propositional Logic, completeness amounts to the problem of determining whether a formula does not have *two* distinct satisfying assignments, therefore it is coNP-complete. By similar reasoning, completeness for First-order Logic is undecidable, as satisfiability is undecidable [13].

5.2. Upper Bounds

The purpose of this subsection is to define matching upper bounds for the logics defined in Section 2. The easiest cases are the logics on one action with restriction 5. Immediately from Theorem 26 and Corollary 25:

Proposition 27. *The completeness problem for a logic* $\mathbf{L}+5$ *on one action is* coNP*-complete.*

For the logics without recursion operators and without restriction 5 or on more that one action, by Theorem 5, satisfiability and validity are both PSPACE-complete. So, completeness is PSPACE-hard, if it is nontrivial. It remains to show that it is also in PSPACE. Similarly, for the μ-calculus, its min- and max-fragments, and for the variants of the other logics with recursive operators, completeness is EXP-hard; it remains to show that completeness is in EXP for the μ-calculus, and in $\mathsf{NPSPACE}^C$ for each $\mathbf{L}_k^\mu$, where validity for $\mathbf{L}_k^\mu$ is in C. To this end we present two similar procedures that decide completeness for a formula, depending on whether the logic has Negative Introspection or not. We call them the CC and CC5 Procedures. Parts of each procedure are similar to the tableaux by Fitting [12] and Massacci [22] for Modal Logic, in that the procedure explores local views of a tableau branch. For more on tableaux the reader can see [8]. The CC and CC5 Procedures are non-deterministic polynomial space algorithms that use an oracle for the logic's satisfiability and validity problems. It accepts exactly the incomplete formulae, thus establishing the required matching upper bounds for the completeness problem.

The CC Procedure for Modal Logic $\mathbf{L}$ *on* φ

In this subsection, we present the CC Procedure for a fixed logic $\mathbf{L}$ that does not have Negative Introspection, but may have recursive operators and any number of actions. Intuitively, the procedure tries to construct two satisfying

Initialize:	Non-deterministically generate formula states a and b that include φ; if there are none, then return "`reject`". If $a \neq b$, then return "`accept`."
Condition A:	If $\models th(a) \supset [\textsc{Act}]\texttt{ff}$, then return "`reject`".
Construction:	Non-deterministically generate an α-child c of a.
Condition B:	If $\not\models th(a) \supset \langle\alpha\rangle th(c)$, then return "`accept`."
Next Step:	Otherwise, set $a := c$ and continue from Condition A.

Table 3: The CC Procedure on φ for a logic **L** without constraint 5.

states for φ and at the same time demonstrate that these states are not bisimilar. We first give a few definitions that we need to describe the procedure.

We assume that each fixpoint operator in φ applies on a unique recursion variable, and therefore for each recursion variable X that appears in φ, there is a unique fixpoint operator that binds it. Thus, we can define the closure of a formula $\psi \in \overline{sub}(\varphi)$, which we identify with $\underline{\psi}$. For our procedure, *formula states* are maximally **L**-satisfiable subsets of $\overline{sub}(\varphi)$. We say that a formula state c is an α-child of formula state a when there are LTS states s and t, such that $s \models a$, $t \models c$, and $s \xrightarrow{\alpha} t$. For a formula state a, let $th(a) = \bigwedge a$.

Lemma 28. *For formula state a:*

- *if $\varphi_1 \wedge \varphi_2 \in a$, then $\varphi_1, \varphi_2 \in a$;*
- *if $\varphi_1 \vee \varphi_2 \in a$, then $\varphi_1 \in a$ or $\varphi_2 \in a$;*
- *if $[\alpha]\psi \in a$ and **L** has constraint T, then $\psi \in a$;*
- *for every $p \in P$, either $p \in a$ or $\neg p \in a$; and*
- *$|a| \leq 2 \cdot |\varphi|$.*

Formula state c is an α-child of formula state a if and only if $th(a) \wedge \langle\alpha\rangle th(c)$ is satisfiable.

Proof. By straightforward arguments. □

The procedure generates sets of formulae and ensures that they are formula states — so that all relevant information is present, due to maximality, and so that they indeed represent LTS state, due to satisfiability. If the current state can be satisfied in two non-bisimilar LTS states (say, s and t), then the procedure should be able to provide a child, representing a state accessible from s or t that is not bisimilar to any state accessible from t or s, respectively. Since the formula states are maximally consistent, two states that are not identical can only be satisfied in non-bisimilar LTS states. The procedure is given in Table 3.

Remark 2. The CC procedure is not appropriate for logics with Negative Introspection, as it may accept a formula that is not complete. For example, consider logic **K5** (or, similarly, $\mathbf{K5}_k$), and let

$$\varphi = p \wedge \neg q \;\wedge\; \Box(p \vee q) \;\wedge\; \Box\Box(\neg p \vee \neg q) \;\wedge\; \Diamond\neg p \;\wedge\; \Diamond\neg q \;\wedge\; \Diamond\Diamond(\neg p \wedge \neg q).$$

Then, there is only one (up to bisimilarity) way to satisfy φ: we use states s, t_1, t_2, t_3, such that $s \to t_1, t_2$ and $t_i \to t_j$ for all $1 \leq i, j \leq 3$. Propositional variable p is true exactly at s and t_2, and q is true exactly at t_1.

It is not hard to see, then, that any other satisfying state for φ must be bisimilar to s. However, assuming the procedure first generates the formula state that contains all subformulae of φ that are satisfied in s, it can then pick the child state c that is satisfied at t_1. That child state contains $\neg p$, q, $p \vee q$, $\Diamond(\neg p \wedge \neg q)$, $\neg p \vee \neg q$, $\Box(\neg p \vee \neg q)$, $\Box\Box(\neg p \vee \neg q)$, $\Diamond\neg p$, $\Diamond\neg q$, $\Diamond\Diamond(\neg p \wedge \neg q)$, and various conjunctions of these formulae. We let the reader verify that these formulae can derive neither $\Diamond p$ nor $\Box\neg p$. Therefore, the procedure can generate a subsequent child c' that contains p, then see that $\not\models th(c) \supset \Diamond th(c')$, and therefore accept the input. ■

This subsection's main theorem is Theorem 29 and informs us that our procedure can determine the completeness of formula φ in a finite number of steps. That the completeness problem for logics without axiom 5 is in PSPACE is a direct corollary, as at every step of the procedure, we only need to store a polynomial amount of information, namely up to two formula states and the size of each state is linear in that of φ by Lemma 28. The proof of Theorem 29 is given in Subsection 5.3.

Theorem 29. *For a logic without Negative Introspection, the CC Procedure accepts φ if and only if φ is incomplete.*

Corollary 30. *The completeness problem for logic* **L** *without Negative Introspection, is in $\mathsf{NPSPACE}^C$, where* **L***-validity is in complexity class C.*

Proof. The CC Procedure is a non-deterministic polynomial-space algorithm with an oracle from C. Each condition that it needs to check is either a closure condition or a condition for the consistency or validity of formulae of polynomial size with respect to $|\varphi|$; therefore, those conditions can be verified either directly or with an oracle from C. □

Corollary 31. *The completeness problem for $\mathbf{K}_k$, $\mathbf{K4}_k$,* **D4***, and* **S4** *is PSPACE-complete; the completeness problem for the μ-calculus and its min- and max-fragments is EXP-complete.*

Proof. PSPACE-hardness and EXP-hardness are consequences of Theorem 26. From Corollary 30, the completeness problem for **K**, **K4**, **D4**, and **S4** is in $\mathsf{NPSPACE}^{\mathsf{PSPACE}} = \mathsf{PSPACE}$, and for the μ-calculus and its min- and max-fragments it is in $\mathsf{NPSPACE}^{\mathsf{EXP}} = \mathsf{EXP}$. □

For the characterization problem, we are given one of φ's satisfying states, so it is a reasonable expectation that the problem became easier. Unfortunately, the characterization problem has exactly the same complexity as the completeness problem. We can easily reduce characterization to completeness by dropping the state from the input. On the other hand, the reduction from validity to completeness of Theorem 26 still works in this case, as it can easily be adjusted to additionally provide the satisfying model of the complete formula $\varphi_P^{\mathbf{L}}$.

Theorem 32. *The characterization problem for logic* $\mathbf{L}$ *without Negative Introspection, is in* $\mathsf{NPSPACE}^C$*, where* $\mathbf{L}$*-validity is in complexity class* C*. Specifically, the characterization problem for* $\mathbf{K}_k$*,* $\mathbf{K4}_k$*,* $\mathbf{D4}$*, and* $\mathbf{S4}$ *is* PSPACE*-complete; the characterization problem for the* μ*-calculus and its min- and max-fragments is* EXP*-complete.*

5.3. The Proof of Theorem 29

We prove that the CC Procedure has a way to accept φ if and only if φ is satisfied in two non-bisimilar states. By Theorem 2, the theorem follows.

We first assume that there are two non-bisimilar pointed states w *and* w'*, such that* $w \models \varphi$ *and* $w' \models \varphi$*. We prove that the CC Procedure accepts* φ *in a finite number of steps.* We call the states w and w' the underlying states, or model states, to distinguish them from the formula states that the process uses. Let $f : \textsc{Prc} \times \textsc{Prc} \times \textsc{Act} \to \textsc{Prc}$ be a partial function that maps every pair (s, t) of non-bisimilar pairs and action α to a model state c accessible from s or t by action α that is non-bisimilar to every state that t or s, respectively, can transition to with α. We call f a choice-function. We can see that the procedure can maintain that the maximal state it generates each time is satisfied in two non-bisimilar states s, t: at the beginning these are w and w'. At every step, the procedure can pick an action $\alpha \in \textsc{Act}$ and an α-child c that is contained in $f(s, t, \alpha)$. If $\not\models th(a) \supset \langle\alpha\rangle th(c)$, then the procedure terminates and accepts the input. Otherwise, c is satisfied in $f(s, t, \alpha)$ and in another state that is non-bisimilar to $f(s, t, \alpha)$. Let that other state be called a counterpart of $f(s, t, \alpha)$.

We demonstrate that if φ is incomplete, then the CC Procedure will accept φ after a finite number of steps. As we have seen above, the procedure, given non-bisimilar states s and t of φ, always has a child to play according to f. For convenience, we can assume that the LTS has no cycles, so the choice-function never repeats a choice during a process run. If for every choice of f, the process does not terminate, then we show that $w \sim w'$, reaching a contradiction. Let $\mathcal{R} = \sim \cup Z$, where $\sim$ is the bisimilarity relation between states, and xZy when for some choice-function, there is an infinite execution of the procedure, in which y is a counterpart of x, or x a counterpart of y. If $x\mathcal{R}y$, either $x \sim y$, so $V_P(x) = V_P(y)$, or xZy, so, again, $V_P(x) = V_P(y)$, since x and y satisfy the same maximal state. If $x\mathcal{R}y$ and $x \xrightarrow{\alpha} x'$, then if $x \sim y$, immediately there is some $y \xrightarrow{\alpha} y'$ so that $x' \sim y'$; if x is a counterpart of y or y is a counterpart of x during a non-terminating run, then for every x' accessible from x (the case is symmetric for a y' accessible from y), either x' is bisimilar to some y' accessible from y, or we can alter the choice-function f that the procedure uses so that $x' = f(x, y, \alpha)$. Since for that altered f, the procedure does not terminate, x' has a counterpart as well. Therefore, the bisimulation conditions are satisfied and $\mathcal{R}$ is a bisimulation. If for every choice-function, the procedure never terminates, then $w \sim w'$, and we have reached a contradiction. Therefore, there is a choice-function f that ensures the procedure terminates after a finite number of steps. Thus, the procedure can non-deterministically follow an appropriate choice-function f and accept after a finite number of steps.

On the other hand, we prove that if φ is complete, then the CC Procedure can never accept φ. For this, we use the following technical lemmata:

Lemma 33. *If s is a formula state and $th(s)$ is incomplete, then there are some $\alpha \in \text{Act}$ and $[\alpha]\psi \in L(P(\varphi))$, such that $th(s) \wedge [\alpha]\psi$ and $th(s) \wedge \langle\alpha\rangle\neg\psi$ are both satisfiable.*

Proof. If $th(s)$ is incomplete, then there is some $\chi \in L(P(\varphi))$ without recursion operators, such that $th(s) \wedge \chi$ and $th(s) \wedge \neg\chi$ are both satisfiable. We can then proceed by induction on χ to prove the claim. Note that χ cannot be a propositional constant or variable. □

Lemma 34. *If a formula state a has a child c, then formula $th(a) \wedge \langle\alpha\rangle th(c)$ is satisfiable; if for formula state a, there is some $\psi \in \overline{sub}(\varphi)$, such that $th(a) \wedge \langle\alpha\rangle\psi$ is satisfiable, then a has a child c, such that $\psi \in c$.*

Proof. Immediate from our definitions and Lemma 28. □

Lemma 35. *Let $s \neq d$ be formula states and ψ a formula. If for some $\alpha \in \text{Act}$, $th(d) \wedge [\alpha]\psi$ is satisfiable (resp. $th(d) \wedge \psi$ is satisfiable, if* **L** *does not have the constraint* 4*), then for every $\beta \in \text{Act}$, $th(s) \wedge [\beta](\neg th(d) \vee [\alpha]\psi)$ is satisfiable (resp. $th(s) \wedge [\beta](\neg th(d) \vee \psi)$ is satisfiable).*

Proof. We can assume that $\models_{\mathbf{L}} th(s) \supset \langle\beta\rangle th(d)$, because otherwise, the lemma follows immediately. Let $\mathcal{M}, w \models th(s)$, where $\mathcal{M} = \langle W, \text{Act}, \rightarrow, V\rangle$ is an LTS for **L**. Let $D, x \models th(d) \wedge [\alpha]\psi$, where $D = \langle W_d, \text{Act}, \rightarrow_d, V_d\rangle$ is an LTS for **L**; let $\mathcal{M}' = \langle W \cup W_d, \text{Act}, \rightarrow_2, V'\rangle$, where $V'(a) = V(a)$ if $a \in W$ and $V'(a) = V_d(a)$ otherwise, and $\xrightarrow{\gamma}_2$ is (resp. when **L** has constraint 4, $\xrightarrow{\gamma}_2$ is the transitive closure of) the collection of

- all pairs $(a, b) \in \xrightarrow{\gamma}$ for which $b \not\models th(d)$ or $\gamma \neq \beta$ or $w \overset{\beta}{\not\rightarrow} b$,
- all pairs (a, x) for which there is some $b \in W$, such that $\mathcal{M}, b \models th(d)$, $w \xrightarrow{\beta} b$, and $a \xrightarrow{\gamma} b$, and
- all pairs in $\xrightarrow{\gamma}_d$.

It is not hard to observe that for every $a \in W_d$, $(\mathcal{M}', a) \sim_{P(\varphi)} (D, a)$, and therefore, for every $\chi \in \overline{sub}(\varphi)$ and $a \in W_d$, $\mathcal{M}', a \models \chi$ iff $D, a \models \chi$. For convenience, we now abuse notation and consider environments for $D, \mathcal{M}$, and $\mathcal{M}'$ to be defined on the whole of W'. Then, by induction on the formulae, we can prove that for every $\chi \in \overline{sub}(\varphi)$, environment ρ, and $a \in W$, $a \in [\![\chi, \rho]\!]_{\mathcal{M}}$ iff $a \in [\![\chi, \rho]\!]_{\mathcal{M}'}$. In fact, because $\overline{sub}(\varphi)$ is closed under negation, for each case we can simply prove that if $a \in [\![\chi, \rho]\!]_{\mathcal{M}}$, then $a \in [\![\chi, \rho]\!]_{\mathcal{M}'}$:

The cases for constants, literals, variables and their negations, and boolean connectives are immediate.

If $\chi = [\gamma]\chi'$**,** then if $\gamma \neq \beta$, nothing changed. For $\gamma = \beta$, let b be such that $a \xrightarrow{\beta}_2 b$. If $b \in W$, then by the inductive hypothesis, $\mathcal{M}', b \models \chi'$. Otherwise, $b = x$ or $x \xrightarrow{\beta} b$, and therefore, there is some $c \in W$, such that $\mathcal{M}, c \models th(d)$ and $a \xrightarrow{\beta} c$. We also have that $\mathcal{M}, c \models \chi'$, and by the maximality of d, this implies that $\chi' \in d$ (resp. $\chi, \chi' \in d$ when the logic has constraint 4). Therefore, $D, x \models \chi'$ (resp. $D, x \models \chi, \chi'$), meaning that $\mathcal{M}', b \models \chi'$.

The case for $\chi = \langle\gamma\rangle\chi'$ is more straightforward.

If $\chi = \nu X.\chi'$**,** then let $S = [\![\chi, \rho]\!]_{\mathcal{M}} \cup [\![\chi, \rho]\!]_{\mathsf{D}}$. From the semantics of Table 1, it suffices now to see that from the inductive hypothesis, $S = [\![\chi', \rho[X \mapsto S]]\!]_{\mathcal{M}} \cup [\![\chi', \rho[X \mapsto S]]\!]_{\mathsf{D}} \subseteq [\![\chi', \rho[X \mapsto S]]\!]_{\mathcal{M}'}$.

If $\chi = \mu X.\chi'$**,** then $\neg\chi = \nu X.\chi''$ for some $\chi'' \in \overline{sub}(\varphi)$. Then, it suffices to prove that if $\mathcal{M}', a \models \neg\chi$, then $\mathcal{M}, a \models \neg\chi$, which can be shown similarly to the previous case.

Therefore, $\mathcal{M}', w \models th(s) \wedge [\beta](\neg th(d) \vee [\alpha]\psi)$. ☐

Using similar constructions, one can prove the following lemma:

Lemma 36. *Let $s \neq d$ be formula states and ψ a formula. If for some $\alpha \in$ Act, $th(s) \wedge \langle\alpha\rangle th(d)$ and $th(d) \wedge \psi$ are satisfiable, then $th(s) \wedge \langle\alpha\rangle(th(d) \wedge \psi)$ is satisfiable.*

Lemma 37. *For formula states s and c, for which c is an α-child of s, if $th(s)$ is complete, then so is $th(c)$.*

Proof. To reach a contradiction, we assume that $th(s)$ is complete and $th(c)$ is not. Then, $s \neq c$ and by Lemma 33, there is some $[\beta]\psi \in L(P(\varphi))$, such that $th(c) \wedge [\beta]\psi$ and $th(c) \wedge \langle\beta\rangle\neg\psi$ are both satisfiable. Furthermore, by Lemma 34, $th(s) \wedge \langle\alpha\rangle th(c)$ is satisfiable. Therefore, by Lemmata 35 and 36, $th(s) \wedge [\alpha](\neg th(d) \vee [\beta]\psi)$ and $th(s) \wedge \langle\alpha\rangle(th(d) \wedge \neg[\beta]\psi)$ are both, respectively satisfiable for some $[\beta]\psi$, and therefore $th(s)$ is not complete. ☐

We can now finish the proof of Theorem 29. By Lemma 37, all states that appear during a run are complete. If at some point, the process picks a child c of a, then by Lemma 34, $th(a) \wedge \langle\alpha\rangle th(c)$ is satisfiable; since a is complete, $\models th(a) \supset \langle\alpha\rangle th(c)$. Therefore, there is no way for the procedure to accept if the input formula is complete. ☐

Initialize:	Non-deterministically generate formula states a and b that include φ; if there are none, then return "`reject`". If $a \neq b$, then return "`accept`." Nondeterministically choose some $\alpha \in \textsc{Act}$.
Condition A:	If $\models th(a) \supset [\alpha]\mathtt{ff}$, then return "`reject`".
Construction:	Non-deterministically pick some $\alpha \in \textsc{Act}$ and generate a full α-child set C of a, such that $\lvert C\rvert \leq \lvert\varphi\rvert$.
Condition B:	If $\not\models th(a) \supset \langle\alpha\rangle th(c)$ for some child $c \in C$ of a, then return "`accept`."
Condition C:	If $\not\models th(a) \supset \langle\alpha\rangle\langle\alpha\rangle th(c)$ for some $c \in C$, then return "`accept`."
Next Step:	Otherwise, non-deterministically pick a $c \in C$ and a $\beta \neq \alpha$, and set $a := c$ and $\alpha := \beta$, and continue from Condition A.

Table 4: The CC5 Procedure on φ for logic **L** with constraint 5.

6. Multi-action Logics with Negative Introspection

We now consider the case of be a logic **L** with restriction 5. We still consider a formula φ that is tested for incompleteness.

As Remark 2 demonstrates, in the presence of Negative Introspection, the child of a complete state might not be complete. Therefore, the CC5 procedure maintains a set of sufficiently many children states for a given action α.

For a formula state a, we call a set C of formula states an α-child set of a when for some $C' \subseteq C$, there are states s and s_c for every $c \in C$, such that $s \models th(a)$ and for every $c \in C$, $s_c \models th(c)$, for every $c \in C'$, $s \xrightarrow{\alpha} s_c$, and for all $c, c' \in C$, $s_c \xrightarrow{\alpha} s_{c'}$. C is a full α-child set of a when $\forall\langle\alpha\rangle\psi \in a.\ \exists c \in C'.\ \psi \in c$ and $\forall c \in C.\ \forall\langle\alpha\rangle\psi \in c.\ \exists c' \in C.\ \psi \in c'$.

For example, for the states s, t_1, t_2, t_3 from Remark 2, if a is the formula state that is saisfied at s, then a full α-child set of a is $C = \{c_1, c_2, c_3\}$, where c_1 is satisfied at t_1, c_2 at t_2, and c_3 at t_3. The subset $C' \subseteq C$, as described above would then be $\{c_1, c_2\}$, as these formula states correspond to LTS states that are directly accessible from s.

We can now describe the CC5 Procedure for logics with constraint 5. See Table 4. To prove the correctness of the procedure, we require the following definition and lemmata. First, to prove that if the input formula is incomplete, then the CC5 procedure can accept, we must know that there is alway a full α-child set of an appropriate size to generate.

Lemma 38. *Let* **L** *have constraint* 5*,* $\alpha \in \textsc{Act}$*, and* a *be a formula state. Then, there is a full* α*-child set* C *of* a*, such that* $\lvert C\rvert \leq \lvert\varphi\rvert$*.*

Proof. Similar to the proofs of Lemma 21 and Corollary 22. □

To show that if the formula is complete, then the procedure can never accept, we must use a similar argument as for the proof of Theorem 29. There, we had to show that the formula state is always complete. However, due to Remark 2, this in not the case for logics with Negative Introspection. Therefore, completeness is now preserved through the set of states that we maintain.

Definition 4. *Let* $\mathbf{L}$ *have constraint* 5 *and* $\alpha \in \textsc{Act}$. *We call a finite set* $\{\varphi_1, \ldots, \varphi_k\} \subseteq L(P)$ *of* k *formulae* α*-complete, when for all states* $s_1, \ldots, s_k$ *and* $t_1, \ldots, t_k$ *in LTSs for* $\mathbf{L}$*, if for every* $1 \leq i, j \leq k$ $s_i \xrightarrow{\alpha} s_j$, $t_i \xrightarrow{\alpha} t_j$*, and* $s_i \models \varphi_i$ *and* $t_i \models \varphi_i$*, then for every* $1 \leq i \leq k$, $s_i \sim_P t_i$.

Lemma 39. *Let* $\mathbf{L}$ *have constraint* 5, $\alpha \in \textsc{Act}$*, let* a *be a formula state, such that* $th(a)$ *is complete, and let* C *be a full* α*-child set of* a*. Then,* $\{th(c) \mid c \in C\}$ *is* α*-complete.*

Proof. We prove the contrapositive of the lemma. Let $\mathcal{M}, s \models th(a)$, where $\mathcal{M} = \langle W, \textsc{Act}, \rightarrow, V\rangle$ and $s \in W$. Let for every $c \in C$, $s^1_c \in W_1$ and $s^2_c \in W_2$, where for $i = 1, 2$, $\mathcal{M}_i = \langle W_i, \textsc{Act}, \rightarrow_i, V_i\rangle$, such that for every $c, c' \in C$, $s^i_c \xrightarrow{\alpha}_i s^i_{c'}$, $\mathcal{M}_i, s^i_c \models th(c)$, and $W \cap W_i = \emptyset$. Let for $i = 1, 2$, $\mathcal{M}'_i = (W \cup W_i \cup \{s'\}, \textsc{Act}, \rightarrow'_i, V'_i)$, where $s' \notin W \cup W_1 \cup W_2$;

$$\begin{aligned}
\xrightarrow{\beta}{}'_i \;&=\; \xrightarrow{\beta} \;\cup\; \xrightarrow{\beta}_i \;\cup\; \{(s', t) \mid s \xrightarrow{\beta} t\} \;\cup\; \{(t, s') \mid t \xrightarrow{\beta} s\}\\
&\quad\;\cup\; \{(s', s') \mid \mathbf{L} \text{ has constraint } T\}, \quad \text{for } \beta \neq \alpha, \text{ and}\\
\xrightarrow{\alpha}{}'_i \;&=\; \xrightarrow{\alpha}_i \;\cup\; \{(s', s^i_c) \mid c \in C \text{ and } c \text{ is a child of } a\} \;\cup\; \{(t, t') \in \xrightarrow{\alpha} \mid t \neq s'\}\\
&\quad\;\cup\; \{(s', s'), (s^i_c, s') \mid c \in C \text{ and } \mathbf{L} \text{ has constraint } T\};
\end{aligned}$$

and $V'_i(t) = V(t)$ if $t \in W$, $V'_i(t) = V_i(t)$ if $t \in W_i$, and $V'_i(s') = a \cap P(\varphi)$. It is not hard to see that if $\exists c \in C.\ (\mathcal{M}_1, s^1_c) \not\sim (\mathcal{M}_2, s^2_c)$, then $(\mathcal{M}'_1, s') \not\sim (\mathcal{M}'_2, s')$. To complete the proof of the lemma, it remains to demonstrate that for $i = 1, 2$, $\mathcal{M}_i, s' \models th(a)$. This can be with a similar induction as in the proof of Lemma 35. ☐

Lemma 40. *Let* $\mathbf{L}$ *have constraint* 5, $\alpha, \beta \in \textsc{Act}$*, where* $\alpha \neq \beta$*, let* C *be a set of formula-states, such that* $\{th(c) \mid c \in C\}$ *is* α*-complete, and let* C' *be a full* β*-child set of some* $a' \in C$*. Then,* $\{th(c) \mid c \in C'\}$ *is* β*-complete.*

Proof. Similar to the proof of Lemma 39. ☐

Using the above lemmata, we can prove the correctness of the CC5 procedure in a way similar to what we did for the CC procedure:

Theorem 41. *Let* $\mathbf{L}$ *have constraint* 5*. The CC5 procedure accepts* φ *if and only if* φ *is not* $\mathbf{L}$*-complete.*

Corollary 42. *The Completeness problem for a logic* $\mathbf{L}$ *with more than one action and constraint* 5 *is* PSPACE*-complete when recursion is not allowed, and otherwise, in* $\mathsf{NPSPACE}^C$*, where* $\mathbf{L}^\mu$*-validity is in* C.

Theorem 43. *The Characterization problem for a logic* $\mathbf{L}$ *with more than one action and constraint* 5 *is* PSPACE*-complete when recursion is not allowed, and otherwise, in* $\mathsf{NPSPACE}^C$*, where* $\mathbf{L}^\mu$*-validity is in* C.

7. Variations and Other Considerations

There are several variations one may consider for the completeness problem. One may define the completeness of a formula in a different way, or consider a different logic, depending on the intended application. One may also wonder whether we could attempt a solution to the completeness problem by using Fine's normal forms [10]. In this section, we will examine some of these variations on the completeness theme.

7.1. Characteristic Formulae

It may be more appropriate, depending on the case, to check whether a formula is *satisfiable and complete*, that is, whether there is a state for which the formula is characteristic. In this case, we can simply alter the CC Procedure so that it accepts right away if the formula is not satisfiable. Furthermore, notice that the reduction used in the proof for Theorem 26 constructs satisfiable formulae. Therefore, this problem has the same complexity as the completeness problem, for most cases, as long as satisfiability has the same complexity as validity.

For logics on one action and with Negative Introspection (and plain Propositional Logic), this is not necessarily the case. For these logics, the language of satisfiable and complete formulae is US-complete, where a language U is in US when there is a nondeterministic Turing machine T, so that for every instance x of U, $x \in U$ if and only if T has exactly one accepting computation path for x[7] [6]: UniqueSAT is a complete problem for US and a special case of this variation of the completeness problem. The class US is not known to be the same as coNP.

From now on, we only consider logics on one action.

7.2. Completeness and Normal Forms for Modal Logic

In [10], Fine introduced normal forms for Modal Logic. The sets F_P^d are defined recursively on the depth d, which is a nonnegative integer, and depend on the set of propositional variables P (we use a variation on the presentation from [25]):

$$F_P^0 = \left\{ \bigwedge_{p \in S} p \wedge \bigwedge_{p \notin S} \neg p \mid S \subseteq P \right\}; \quad \text{and}$$

$$F_P^{d+1} = \left\{ \varphi_0 \wedge \bigwedge_{\varphi \in S} \Diamond\varphi \wedge \Box \bigvee_{\varphi \in S} \varphi \mid S \subseteq F_P^d,\ \varphi_0 \in F_P^0 \right\}.$$

For example, formula $\varphi_P^{\mathbf{K}} = \bigwedge P \wedge \Box\mathtt{ff}$ from Section 3 is a normal form in F_P^1.

[7]We note that the definition of US is different from UP [31]; for UP, if T has an accepting path for x, then it is *guaranteed* that it has a unique accepting path for x.

Theorem 44 (from [10]). *For every modal formula φ of modal depth at most d, if φ is* **K***-satisfiable, then there is some finite $S \subseteq F_P^d$, so that $\models_{\mathbf{K}} \varphi \leftrightarrow \bigvee S$.*

Furthermore, as Fine [10] demonstrated, normal forms are mutually exclusive: no two distinct normal forms from F_P^d can be true at the same state of a model. Normal forms are not necessarily complete by our definition (for example, consider $p \wedge \Diamond p \wedge \Box p$ for $P = \{p\}$), but, at least for **K**, it is not hard to distinguish the complete ones; by induction on d, one can show that:

Lemma 45. *Formula $\varphi \in F_P^d$ is complete for* **K** *if and only if $md(\varphi) < d$.*

Therefore, for **K**, the satisfiable and complete formulae are exactly the ones that are equivalent to such a complete normal form. However, using this observation to test formulae for completeness by guessing a complete normal form and verifying that it is equivalent to our input formula can be very costly, as normal forms can be of very large size: $|F_P^0| = 2^{|P|}$; $|F_P^{d+1}| = |P| \cdot 2^{|F_P^d|}$; and if $\psi \in F_P^d$, $|\psi|$ can be up to $|P| + 2|F_P^{d-1}|$. We would be guaranteed a normal form of reasonable (that is, polynomial with respect to $|\varphi|$) size to compare with φ only if φ uses a small (logarithmic with respect to $|\varphi|$) number of variables and its modal depth is very small compared to $|\varphi|$ (that is, $md(\varphi) = O(\log^*(|\varphi|))$).

7.3. Completeness up to Depth

Fine's normal forms [10] can inspire us to consider a relaxation of the definition of completeness. We call a formula φ *complete up to its depth* for a logic **L** exactly when for every formula $\psi \in L(P(\varphi))$ of modal depth at most $md(\varphi)$, either $\varphi \models \psi$ or $\varphi \models \neg\psi$. Immediately from Theorem 44, we have that:

Lemma 46. *All normal forms are complete up to their depths.*

Lemma 47. *Formula φ is satisfiable and complete up to its depth for logic* **L** *if and only if it is equivalent in* **L** *to a normal form from $F_P^{md(\varphi)}$.*

Proof. From Theorem 44, if φ is satisfiable, then it is equivalent to some $\bigvee S$, where $S \subseteq F_P^{md(\varphi)}$, but if it is also complete up to its depth, then it can derive a the normal form $\psi \in S$; so, $\models \varphi \supset \psi$, but also $\models \psi \supset \bigvee S$ and $\bigvee S$ is equivalent to φ. For the other direction, notice that every normal form in $F_P^{md(\varphi)}$ is either complete or has the same modal depth as φ, so by Lemma 46, φ is equivalent to a normal form. In the first case, it is complete and, in the second case, it is complete up to its depth. □

Therefore, all modal logics have formulae that are complete up to their depth. In fact, for any finite set of propositional variables P and $d \geq 0$, we can define $\varphi_P^d = \bigwedge_{i=0}^{d} \Box^i \bigwedge P$, which is equivalent in **T** and **D** to a normal form (by induction on d). Then, we can use a reduction similar to the one from the proof of Theorem 26 to prove that, for every modal logic, completeness up to depth is as hard as validity.

Proposition 48. *For any complexity class C and logic* **L***, if* **L***-validity is C-hard, then completeness up to depth is C-hard.*

Proof. The proof is similar to the one for Theorem 26 and is by reduction from **L**-validity. We are given a formula $\varphi \in L(P)$ — and we assume that $P \neq \emptyset$. For $\mathbf{L} \neq \mathbf{T}, \mathbf{D}$, $\varphi^{\mathbf{L}}_P(d) = \varphi^{\mathbf{L}}_P$ as defined in Section 3; for $\mathbf{L} = \mathbf{T}$ or $\mathbf{D}$, let $\varphi^{\mathbf{L}}_P(d) = \varphi^d_P$ as defined above. We also assume an appropriate $M_{\mathbf{L}}, a_{\mathbf{L}} \models \varphi^{\mathbf{L}}_P(d)$. If $M_{\mathbf{L}}, a_{\mathbf{L}} \not\models \varphi$, let $\varphi_c = \bigwedge P \wedge \Box\mathtt{tt}$; otherwise, let $\varphi_c = \varphi \supset \varphi^{\mathbf{L}}_P(d)$. For the second case, if φ is provable, then φ_c is equivalent to $\varphi^{\mathbf{L}}_P(d)$, which is complete up to its depth. If φ_c is complete up to its depth, then by Lemma 47, it is equivalent to a normal form $\psi \in F^d_P$. So, ψ is equivalent to $\varphi_c = \varphi \supset \varphi^{\mathbf{L}}_P(d)$, which is equivalent to $\neg \bigvee S \vee \varphi^{\mathbf{L}}_P(d)$ for some $S \subseteq F^d_P$, by Theorem 44. Since normal forms are mutually exclusive, $\bigvee S$ is equivalent to $\neg \bigvee (F^d_P \setminus S)$, so ψ is equivalent to $\bigvee (F^d_P \setminus S) \vee \varphi^{\mathbf{L}}_P(d)$. Therefore, either $S = F^d_P$ and ψ is equivalent to $\varphi^{\mathbf{L}}_P(d)$, or $F^d_P \setminus S$ is a singleton of a normal form equivalent to $\varphi^{\mathbf{L}}_P(d)$. In the first case, φ is provable, because for any model $\mathcal{M}, a$, by Theorem 44, $\mathcal{M}, a \models \bigvee F^d_P$, so $\mathcal{M}, a \models \varphi$. The second case cannot hold, because it would mean that φ is equivalent to $\neg\varphi^{\mathbf{L}}_P(d)$, but $M_{\mathbf{L}}, a_{\mathbf{L}} \models \varphi$. □

We demonstrate that this variation of the completeness problem is in PSPACE when the logic is **K**; it seems plausible that one can follow similar approaches that use normal forms for the remaining modal logics. We leave this topic for future work.

Proposition 49. *A formula φ is complete up to its depth for **K** if and only if $\varphi \wedge \Box^{md(\varphi)+1}\mathtt{ff}$ is complete for **K**.*

Proof. Let $\psi \in F^d_P$ be a normal form. Then, $\psi \wedge \Box^{d+1}\mathtt{ff}$ is equivalent in **K** to $\psi^{+1} \in F^{d+1}_P$, which is ψ after we replace all $\Diamond\psi'$ in ψ by $\Diamond(\psi' \wedge \Box\mathtt{ff})$, where $\psi' \in F^0_P$. Notice that $\psi_1, \psi_2 \in F^d_P$ are distinct normal forms if and only if ψ_1^{+1}, ψ_2^{+1} are distinct normal forms in F^r_P for every $r > d$. So, φ is complete up to its depth for **K** if and only if $\varphi \wedge \Box^{md(\varphi)+1}\mathtt{ff}$ is complete for **K**. □

7.4. More Logics

There is more to Modal Logic than what we have covered in this paper, so perhaps there is also more to discover about the completeness problem. We based the decision procedure for the completeness problem for each logic on a decision procedure for satisfiability. We distinguished two cases:

- If the logic has axiom 5, then to test satisfiability we guess a small model and we use model checking to verify that the model satisfies the formula. This procedure uses the small model property of these logics (Corollary 22). To test for completeness, we guess *two* small models; we verify that they satisfy the formula and that they are non-bisimilar. We could try to use a similar approach for another logic based on a decision procedure for satisfiability based on a small model property (for, perhaps, another meaning for "small'). To do so successfully, a small model property may not suffice. We need to first demonstrate that, for this logic, a formula that is satisfiable and incomplete has *two* small non-bisimilar models.

- For the other logics, we can use a tableau to test for satisfiability. We were able to combine the tableaux for these logics with bisimilarity-testing to provide an optimal — when the completeness problem is not trivial — procedure for testing for completeness. For logics where a tableau gives an optimal procedure for testing for satisfiability, this is, perhaps, a promising approach to test also for completeness.

Another direction of interest would be to consider axiom schemes as part of the input — as we have seen, axiom 5 together with $\varphi^{\mathbf{S5}}$ is complete for $\mathbf{T}$, when no modal formula is.

References

[1] Luca Aceto, Anna Ingolfsdottir, Paul Blain Levy, and Joshua Sack. Characteristic formulae for fixed-point semantics: a general framework. *Mathematical Structures in Computer Science*, 22(02):125–173, 2012.

[2] Antonis Achilleos. The completeness problem for modal logic. In *Logical Foundations of Computer Science*, Lecture Notes in Computer Science, pages 1–21. Springer, 2018.

[3] Sergei Artemov. Syntactic epistemic logic. In *Book of Abstracts,15th Congress of Logic, Methodology and Philosophy of Science CLMPS 2015*, pages 109–110.

[4] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[5] Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier Science, 2006.

[6] Andreas Blass and Yuri Gurevich. On the unique satisfiability problem. *Information and Control*, 55(1-3):80 – 88, 1982.

[7] Alexander Chagrov and Michael Zakharyaschev. Modal logic, 1997.

[8] Marcello D'Agostino, Dov M Gabbay, Reiner Hähnle, and Joachim Posegga. *Handbook of tableau methods*. Springer, 1999.

[9] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.

[10] Kit Fine. Normal forms in modal logic. *Notre Dame journal of formal logic*, 16(2):229–237, 1975.

[11] Michael J Fischer and Richard E Ladner. Propositional dynamic logic of regular programs. *Journal of computer and system sciences*, 18(2):194–211, 1979.

[12] Melvin Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, 13(2):237–247, 1972.

[13] Kurt Gödel. Über formal unentscheidbare sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für mathematik und physik*, 38(1):173–198, 1931.

[14] Susanne Graf and Joseph Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Information and Control*, 68(1-3):125–145, January 1986.

[15] Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[16] Joseph Y Halpern and Leandro Chaves Rêgo. Characterizing the NP-PSPACE gap in the satisfiability problem for modal logic. *Journal of Logic and Computation*, 17(4):795–806, 2007.

[17] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM (JACM)*, 32(1):137–161, 1985.

[18] Anna Ingólfsdóttir, Jens Christian Godskesen, and Michael Zeeberg. Fra Hennessy-Milner logik til CCS-processer. *Master's thesis, Department of Computer Science, Aalborg University*, 1987.

[19] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

[20] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

[21] Kim Guldstrand Larsen. Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theoretical Computer Science*, 72(2&3):265–288, 1990.

[22] Fabio Massacci. Single step tableaux for modal logics. *Journal of Automated Reasoning*, 24(3):319–364, 2000.

[23] Robin Milner. *Communication and concurrency.* Prentice-Hall, Inc., 1989.

[24] Markus Mller-Olm. Derivation of characteristic formulae. *Electronic Notes in Theoretical Computer Science*, 18:159–170, 1998.

[25] Lawrence S Moss. Finite models constructed from canonical formulas. *Journal of Philosophical Logic*, 36(6):605–640, 2007.

[26] Michael C. Nagle and S. K. Thomason. The extensions of the modal logic K5. *Journal of Symbolic Logic*, 50(1):102–109, 1985.

[27] Robert Paige and Robert E Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[28] Vaughan R. Pratt. A decidable mu-calculus: Preliminary report. In *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*. IEEE, oct 1981.

[29] Bernhard Steffen and Anna Ingólfsdóttir. Characteristic formulas for processes with divergence. *Information and Computation*, 110(1):149–163, 1994.

[30] Robert S Streett and E Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and computation*, 81(3):249–264, 1989.

[31] Leslie G Valiant. Relative complexity of checking and evaluating. *Information processing letters*, 5(1):20–23, 1976.

[32] Igor Walukiewicz. Completeness of kozen's axiomatisation of the propositional -calculus. *Inform. and Comput*, pages 14–24, 1995.