Model Checking and Validity in Propositional and Modal Inclusion Logics[*]

Lauri Hella[1], Antti Kuusisto[2], Arne Meier[3], and Jonni Virtema[4]

1 **University of Tampere, Finland**
`lauri.hella@uta.fi`
2 **University of Bremen, Germany**
`antti.j.kuusisto@gmail.com`
3 **Leibniz Universität Hannover, Germany**
`meier@thi.uni-hannover.de`
4 **University of Helsinki, Helsinki, Finland**
`jonni.virtema@helsinki.fi`

Abstract

Propositional and modal inclusion logic are formalisms that belong to the family of logics based on team semantics. This article investigates the model checking and validity problems of these logics. We identify complexity bounds for both problems, covering both lax and strict team semantics. By doing so, we come close to finalising the programme that ultimately aims to classify the complexities of the basic reasoning problems for modal and propositional dependence, independence, and inclusion logics.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Inclusion Logic, Model Checking, Complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.32

1 Introduction

Team semantics is the mathematical framework of modern logics of dependence and independence, which, unlike Tarski semantics, is not based on singletons as satisfying elements (e.g., first-order assignments or points of a Kripke structure) but on sets of such elements. More precisely, a first-order team is a set of first-order assignments that have the same domain of variables. As a result, a team can be interpreted as a database table, where variables correspond to attributes and assignments to records. Team semantics originates from the work of Hodges [17], where it was shown that Hintikka's IF-logic can be based on a compositional (as opposed to game-theoretic) semantics. In 2007, Väänänen [24] proposed a fresh approach to logics of dependence and independence. Väänänen adopted team semantics as a core notion for his *dependence logic*. Dependence logic extends first-order logic by atomic statements such as *the value of variable x is determined by the value of y*. Such a statement is not meaningful under a single assignment, however, when evaluated over a team, such a statement corresponds precisely to functional dependence of database theory when the team is interpreted as a database table.

[*] The second and the last author acknowledges support from Jenny and Antti Wihuri Foundation. The last author is also supported by the grant 292767 of the Academy of Finland. The third author is supported by the DFG grant ME 4279/1-1. We thank the anonymous referees for their comments.

Besides functional dependence, there are many other important dependency notions used in fields like statistics and database theory, which give rise to interesting logics based on team semantics. The two most widely studied of these new logics are *independence logic* of Grädel and Väänänen [10], and *inclusion logic* of Galliani [5]. Inclusion logic extends first-order logic by atomic statements of the form $x \subseteq y$, which is satisfied in a team X if any value that appears as a value for x in X also appears as a value of y in X. Dependence and independence logics are equi-expressive with existential second-order logic and thus capture the complexity class NP [24, 10]. Surprisingly, inclusion logic has the same expressive power as *positive greatest fixed point logic* GFP^+ [7]. Since on finite structures, GFP^+ coincides with *least fixed point logic* LFP, it follows from the Immermann-Vardi-Theorem that inclusion logic captures the complexity class P on finite ordered structures. Interestingly under a semantical variant of inclusion logic called *strict semantics* the expressive power of inclusion logic rises to existential second-order logic [6]. Moreover, the fragment of inclusion logic (under strict semantics) in which only k universally quantified variables may occur captures the complexity class $\mathsf{NTIME}_{\mathsf{RAM}}(n^k)$ (i.e., structures that can be recognised by a nondeterministic random access machine in time $\mathcal{O}(n^k)$) [11]. The above characterisations exemplify that, indeed, inclusion logic and its fragments have very compelling descriptive complexity-theoretic properties.

In this paper, we study propositional and modal inclusion logic under both the standard semantics (i.e., *lax semantics*) and strict semantics. The research around propositional and modal logics with team semantics has concentrated on classifying the complexity and definability of the related logics. Due to very active research efforts, the complexity and definability landscape of these logics is understood rather well; see the survey of Durand et al. [4] and the references therein for an overview of the current state of the research. In the context of propositional logic (modal logic, resp.) a team is a set of propositional assignments with a common domain of variables (a subset of the domain a Kripke structure, resp.). *Extended propositional inclusion logic* (*extended modal inclusion logic*, resp.) extends propositional logic (modal logic, resp.) with *propositional inclusion atoms* $\varphi \subseteq \psi$, where φ and ψ are formulae of propositional logic (modal logic, resp.). Inclusion logics have fascinating properties also in the propositional setting. The following definability results hold for the standard lax semantics. A class of team pointed Kripke models is definable in extended modal inclusion logic iff $(\mathfrak{M}, \emptyset)$ is in the class for every model $\mathfrak{M}$, the class is closed under taking unions, and the class is closed under the so-called *team k-bisimulation*, for some finite k [16]. From this, a corresponding characterisation for extended propositional inclusion logic directly follows: a class of propositional teams is definable in extended propositional inclusion logic iff the empty team is in the class, and the class is closed under taking unions. In [21, 22] (global) model definability and frame definability of team based modal logics are studied. It is shown that surprisingly, in both cases, (extended) modal inclusion logic collapses to modal logic.

This paper investigates the complexity of the model checking and the validity problem for propositional and modal inclusion logic. The complexity of the satisfiability problem of modal inclusion logic was studied by Hella et al. [15]. The study on the validity problem of propositional inclusion logic was initiated by Hannula et al. [12], where the focus was on more expressive logics in the propositional setting. Consequently, the current paper directly extends the research effort initiated in these papers. It is important to note that since the logics studied in this paper are not closed under taking negations, the connection between the satisfiability problem and the validity problem fails. In [12] it was shown that, under lax semantics, the validity problem for propositional inclusion logic is coNP-complete. Here we obtain an identical result for the strict semantics. However, surprisingly, for model checking the picture looks quite different. We establish that whereas the model checking problem for propositional inclusion logic is P-complete under lax semantics, the problem becomes NP-complete for the strict variant. Also surprisingly, for model checking in the modal setting, we obtain the identical results (as in the propositional setting): modal inclusion logic is P-complete under lax semantics and NP-complete under strict semantics. Nevertheless, for the validity problem, the modal variants are much more complex than the propositional ones; we establish coNEXP-hardness for both strict and lax semantics.

2 Propositional logics with team semantics

Let D be a finite, possibly empty set of proposition symbols. A function $s\colon D \to \{0,1\}$ is called an *assignment*. A set X of assignments $s\colon D \to \{0,1\}$ is called a *team*. The set D is the *domain* of X. We denote by 2^D the set of *all assignments* $s\colon D \to \{0,1\}$. If $\vec{p} = (p_1, \ldots, p_n)$ is a tuple of propositions and s is an assignment, we write $s(\vec{p})$ for $(s(p_1), \ldots, s(p_n))$.

Let Φ be a set of proposition symbols. The syntax of propositional logic $\mathsf{PL}(\Phi)$ is given by the following grammar: $\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi)$, where $p \in \Phi$.

We denote by $\models_{\mathsf{PL}}$ the ordinary satisfaction relation of propositional logic defined via assignments in the standard way. Next we give team semantics for propositional logic.

▶ **Definition 1** (Lax team semantics)**.** Let Φ be a set of atomic propositions and let X be a team. The satisfaction relation $X \models \varphi$ is defined as follows.

$$\begin{aligned}
X &\models p &&\Leftrightarrow \quad \forall s \in X : s(p) = 1,\\
X &\models \neg p &&\Leftrightarrow \quad \forall s \in X : s(p) = 0.\\
X &\models (\varphi \wedge \psi) &&\Leftrightarrow \quad X \models \varphi \text{ and } X \models \psi.\\
X &\models (\varphi \vee \psi) &&\Leftrightarrow \quad Y \models \varphi \text{ and } Z \models \psi, \text{ for some } Y, Z \text{ such that } Y \cup Z = X.
\end{aligned}$$

The lax team semantics is considered to be the standard semantics for team-based logics. In this paper, we also consider a variant of team semantics called the *strict team semantics*. In strict team semantics, the above clause for disjunction is redefined as follows:

$$X \models_{\mathsf{str}} (\varphi \vee \psi) \Leftrightarrow Y \models \varphi \text{ and } Z \models \psi, \text{ for some } Y, Z \text{ such that } Y \cap Z = \emptyset \text{ and } Y \cup Z = X.$$

When L denotes a team-based propositional logic, we let $\mathsf{L}_{\mathsf{str}}$ denote the variant of the logic with strict semantics. Moreover, in order to improve readability, for strict semantics we use $\models_{\mathsf{str}}$ instead of $\models$. As a result lax semantics is used unless otherwise specified. The next proposition shows that the team semantics and the ordinary semantics for propositional logic defined via assignments (denoted by $\models_{\mathsf{PL}}$) coincide.

▶ **Proposition 2** ([24])**.** *Let φ be a formula of propositional logic and let X be a propositional team. Then $X \models \varphi$ iff $\forall s \in X : s \models_{\mathsf{PL}} \varphi$.*

The syntax of *propositional inclusion logic* $\mathsf{PInc}(\Phi)$ is obtained by extending the syntax of $\mathsf{PL}(\Phi)$ by the grammar rule $\varphi ::= \vec{p} \subseteq \vec{q}$, where $\vec{p}$ and $\vec{q}$ are finite tuples of proposition variables with the same length. The semantics for propositional inclusion atoms is defined as follows:

$$X \models \vec{p} \subseteq \vec{q} \text{ iff } \forall s \in X\, \exists t \in X : s(\vec{p}) = t(\vec{q}).$$

▶ Remark. *Extended propositional inclusion logic* is the variant of PInc in which inclusion atoms of the form $\vec{\varphi} \subseteq \vec{\psi}$, where $\vec{\varphi}$ and $\vec{\psi}$ are tuples of PL-formulae, are allowed. Observe that this extension does not increase the complexity of the logic and on that account, in this paper, we only consider the non-extended variant.

	p	q	r
s_1	1	0	0
s_2	1	1	1
s_3	0	1	0

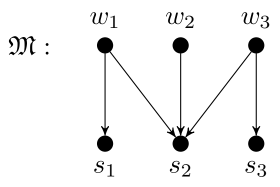

Figure 1 Assignments for teams in Example 4 and the Kripke model for Example 19.

Table 1 Complexity of the satisfiability, validity and model checking problems for propositional logics under both systems of semantics. The shown complexity classes refer to completeness results. † In [15] NEXP-completeness is claimed. However there is a mistake in the proof and the authors of [15] now have a proof for EXP-completeness.

	Satisfiability		Validity		Model checking	
	strict	lax	strict	lax	strict	lax
PL	— NP [3, 19] —		—— coNP [3, 19] ——		——— NC^1 [1] ———	
PInc	EXP†	EXP [15]	coNP [Th. 6]	coNP [12]	NP [Th. 14]	P [Th. 10]

Note that PInc is not a downward closed logic[1]. However, analogously to FO-inclusion-logic [5], satisfaction of PInc-formulas is closed under taking unions.

▶ **Proposition 3** (Closure under unions). *Let $\varphi \in$ PInc and let X_i, for $i \in I$, be teams. Suppose that $X_i \models \varphi$ for each $i \in I$. Then $\bigcup_{i\in I} X_i \models \varphi$.*

Similarly as in first-order team semantics [5], also for propositional logic the strict and the lax semantics coincide; meaning that $X \models \varphi$ iff $X \models_{\mathsf{str}} \varphi$ for all X and φ. However this does not hold for propositional inclusion logic, for the following example shows that $\mathsf{PInc}_{\mathsf{str}}$ is not union closed. Moreover, we will show that the two different semantics lead to different complexities for the related model checking problems.

▶ **Example 4.** Let s_1, s_2, and s_3 be as in Figure 1 and define $\varphi := \big(p \wedge (p \subseteq r)\big) \vee \big(q \wedge (q \subseteq r)\big)$. Note that $\{s_1, s_2\} \models_{\mathsf{str}} \varphi$ and $\{s_2, s_3\} \models_{\mathsf{str}} \varphi$, but $\{s_1, s_2, s_3\} \not\models_{\mathsf{str}} \varphi$.

However, $\mathsf{PInc}_{\mathsf{str}}$ satisfies a useful weaker form of union closure: it is straightforward to prove by an induction on the formula structure that it is closed under unions of *singleton teams*.

▶ **Lemma 5.** *Let X be a team and $\varphi \in \mathsf{PInc}_{\mathsf{str}}$. If $\{s\} \models_{\mathsf{str}} \varphi$ for every $s \in X$, then $X \models_{\mathsf{str}} \varphi$.*

3 Complexity of propositional inclusion logic

We now define the model checking, satisfiability, and validity problems in the context of team semantics. Let L be a propositional logic with team semantics. A formula $\varphi \in$ L is *satisfiable*, if there exists a non-empty team X such that $X \models \varphi$. A formula $\varphi \in$ L is *valid* if $X \models \varphi$ holds for all teams X such that the propositions in φ are in the domain of X. The

[1] A logic L is downward closed if "$X \models \varphi$ and $Y \subseteq X$ implies $Y \models \varphi$" holds for every formula $\varphi \in$ L and teams X and Y.

satisfiability problem SAT(L) and the validity problem VAL(L) are defined in an obvious way: Given a formula $\varphi \in \mathsf{L}$, decide whether the formula is satisfiable (valid, respectively). For the model checking problem MC(L) we consider combined complexity: Given a formula $\varphi \in \mathsf{L}$ and a team X, decide whether $X \models \varphi$. See Table 1 for known complexity results for PL and PInc, together with partial results of this paper.

It was shown by Hannula et al. [12] that the validity problem of PInc is coNP-complete. Here we establish that the corresponding problem for $\mathsf{PInc}_{\mathsf{str}}$ is also coNP-complete. Our proof is similar to theirs [12], except that instead of union closure we use Lemma 5.

▶ **Theorem 6.** *The validity problem for* $\mathsf{PInc}_{\mathsf{str}}$ *is* coNP*-complete w.r.t.* $\leq_m^{\log}$.

Proof Sketch. The coNP-hardness follows from the fact that PL is a sublogic of $\mathsf{PInc}_{\mathsf{str}}$ and since the validity problem of PL is coNP-hard. On the other hand, by Lemma 5, a formula $\varphi \in \mathsf{PInc}_{\mathsf{str}}$ is valid iff it is satisfied by all singleton teams $\{s\}$. It is easy to see that, over a singleton team $\{s\}$, any inclusion atom is equivalent to a short PL-formula. Consequently, there is a short PL-formula φ^* which is valid iff φ is valid. Since VAL(PL) is in coNP, the same holds for VAL($\mathsf{PInc}_{\mathsf{str}}$). ◀

3.1 Model checking in lax semantics is P-complete

In this section we construct a reduction from the monotone circuit value problem to the model checking problem of PInc. For a deep introduction to circuits see Vollmer [25].

▶ **Definition 7.** A *monotone Boolean circuit* with n input gates and one output gate is a 3-tuple $C = (V, E, \alpha)$, where (V, E) is a finite, simple, directed, acyclic graph, and $\alpha\colon V \to \{\vee, \wedge, x_1, \dots, x_n\}$ is a function such that the following conditions hold:

1. Every $v \in V$ has in-degree 0 or 2.
2. There exists exactly one $w \in V$ with out-degree 0. We call this node w the *output gate* of C and denote it by g_{out}.
3. If $v \in V$ is a node with in-degree 0, then $\alpha(v) \in \{x_1, \dots, x_n\}$.
4. If $v \in V$ has in-degree 2, then $\alpha(v) \in \{\vee, \wedge\}$.
5. For each $1 \leq i \leq n$, there exists exactly one $v \in V$ with $\alpha(v) = x_i$.

Let $C = (V, E, \alpha)$ be a monotone Boolean circuit with n input gates and one output gate. Any sequence $b_1, \dots, b_n \in \{0, 1\}$ of bits of length n is called an *input* to the circuit C. A function $\beta\colon V \to \{0, 1\}$ defined such that

$$\beta(v) := \begin{cases} b_i & \text{if } \alpha(v) = x_i \\ \min\big(\beta(v_1), \beta(v_2)\big) & \text{if } \alpha(v) = \wedge, \text{ where } v_1 \neq v_2 \text{ and } (v_1, v), (v_2, v) \in E, \\ \max\big(\beta(v_1), \beta(v_2)\big) & \text{if } \alpha(v) = \vee, \text{ where } v_1 \neq v_2 \text{ and } (v_1, v), (v_2, v) \in E. \end{cases}$$

is called the *valuation of the circuit C under the input $b_1, \dots, b_n$*. The *output of the circuit C* is then defined to be $\beta(g_{\text{out}})$.

The *monotone circuit value problem* (MCVP) is the following decision problem: Given a monotone circuit C and an input $b_1, \dots, b_n \in \{0, 1\}$, is the output of the circuit 1?

▶ **Proposition 8** ([9]). MCVP *is* P*-complete w.r.t.* $\leq_m^{\log}$ *reductions.*

▶ **Lemma 9.** MC(PInc) *under lax semantics is* P*-hard w.r.t.* $\leq_m^{\log}$.

Proof. We will establish a $\leq_m^{\log}$-reduction from MCVP to the model checking problem of PInc under lax semantics. Since MCVP is P-complete, the claim follows. More precisely, we will show how to construct, for each monotone Boolean circuit C with n input gates and for each input $\vec{b}$ for C, a team $X_{C,\vec{b}}$ and a PInc-formula φ_C such that $X_{C,\vec{b}} \models \varphi_C$ iff the output of the circuit C with the input $\vec{b}$ is 1.

We use teams to encode valuations of the circuit. For each gate v_i of a given circuit, we identify an assignment s_i. The crude idea is that if s_i is in the team under consideration, the value of the gate v_i with respect to the given input is 1. The formula φ_C is used to quantify a truth value for each Boolean gate of the circuit, and then for checking that the truth values of the gates propagate correctly. We next define the construction formally.

Let $C = (V, E, \alpha)$ be a monotone Boolean circuit with n input gates and one output gate and let $\vec{b} = (b_1 \dots b_n) \in \{0,1\}^n$ be an input to the circuit C. We define that $V = \{v_0, \dots, v_m\}$ and that v_0 is the output gate of C. Define

$$\tau_C := \{p_0, \dots, p_m, p_\top, p_\bot\} \cup \{p_{k=i\vee j} \mid i < j, \alpha(v_k) = \vee, \text{ and } (v_i, v_k), (v_j, v_k) \in E\}.$$

For each $i \leq m$, we define the assignment $s_i \colon \tau_C \to \{0,1\}$ as follows:

$$s_i(p) := \begin{cases} 1 & \text{if } p = p_i \text{ or } p = p_\top, \\ 1 & \text{if } p = p_{k=i\vee j} \text{ or } p = p_{k=j\vee i} \text{ for some } j, k \leq m, \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, we define $s_\bot(p) = 1$ iff $p = p_\bot$ or $p = p_\top$. We note that the assignment $s_\bot$ will be the only assignment that maps $p_\bot$ to 1. We make use of the fact that for each gate v_i of C, it holds that $s_\bot(p_i) = 0$. We define

$$X_{C,\vec{b}} := \big\{s_i \mid \alpha(v_i) \in \{\wedge, \vee\}\big\} \cup \big\{s_i \mid \alpha(v_i) \in \{x_i \mid b_i = 1\}\big\} \cup \{s_\bot\},$$

that is, $X_{C,\vec{b}}$ consists of assignments for each of the Boolean gates, assignments for those input gates that are given 1 as an input, and of the auxiliary assignment $s_\bot$.

Let X be any nonempty subteam of $X_{C,\vec{b}}$ such that $s_\bot \in X$. We have

$$\begin{aligned} &X \models p_\top \subseteq p_0 && \text{iff } s_0 \in X \\ &X \models p_i \subseteq p_j && \text{iff } (s_i \in X \text{ implies } s_j \in X) \\ &X \models p_k \subseteq p_{k=i\vee j} && \text{iff } (i < j, (v_i, v_k), (v_j, v_k) \in E, \alpha(v_k) = \vee \\ &&& \quad \text{and } s_k \in X \text{ implies that } s_i \in X \text{ or } s_j \in X) \end{aligned} \tag{1}$$

Recall the intuition that $s_i \in X$ should hold iff the value of the gate v_i is 1. Define

$$\begin{aligned} \psi_{\text{out}=1} &:= p_\top \subseteq p_0, \\ \psi_\wedge &:= \bigwedge\{p_i \subseteq p_j \mid (v_j, v_i) \in E \text{ and } \alpha(p_i) = \wedge\}, \\ \psi_\vee &:= \bigwedge\{p_k \subseteq p_{k=i\vee j} \mid i < j, (v_i, v_k) \in E, (v_j, v_k) \in E, \text{ and } \alpha(v_k) = \vee\}, \\ \varphi_C &:= \neg p_\bot \vee (\psi_{\text{out}=1} \wedge \psi_\wedge \wedge \psi_\vee). \end{aligned}$$

Now observe that $X_{C,\vec{b}} \models \varphi_C$ iff the output of C with the input $\vec{b}$ is 1.

The idea of the reduction is the following: The disjunction in ϕ_C is used to guess a team Y for the right disjunct that encodes the valuation β of the circuit C. The right disjunct is then evaluated with respect to the team Y with the intended meaning that $\beta(v_i) = 1$

whenever $s_i \in Y$. Note that Y is always as required in (1). The formula $\psi_{\mathrm{out}=1}$ is used to state that $\beta(v_0) = 1$, whereas the formulae $\psi_\wedge$ and $\psi_\vee$ are used to propagate the truth value 1 down the circuit. The assignment $s_\perp$ and the proposition $p_\perp$ are used as an auxiliary to make sure that Y is nonempty and to deal with the propagation of the value 0 by the subformulae of the form $p_i \subseteq p_j$. Finally, it is easy to check that the reduction can be computed in logspace. ◀

For the proof of the above lemma it is not important that lax semantics is considered; the same proof works also for the strict semantics. However, as we will show in the next section, we can show a stronger result for the model checking problem of $\mathsf{PInc}_{\mathsf{str}}$; namely that it is NP-hard. In Section 5.1 we will show that the model checking problem for modal inclusion logic with lax semantics is in P (Lemma 21). Since PInc is essentially a fragment of this logic, by combining Lemmas 9 and 21, we obtain the following theorem.

▶ **Theorem 10.** MC(PInc) *under lax semantics is* P*-complete w.r.t.* $\leq_m^{\log}$.

3.2 Model checking in strict semantics is NP-complete

In this section we reduce the set splitting problem, a well-known NP-complete problem, to the model checking problem of $\mathsf{PInc}_{\mathsf{str}}$.

▶ **Definition 11.** The *set splitting* problem is the following decision problem:
Input: A family $\mathcal{F}$ of subsets of a finite set S.
Problem: Do there exist subsets S_1 and S_2 of S such that

1. S_1 and S_2 are a partition of S (i.e., $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$),
2. for each $A \in \mathcal{F}$, there exist $a_1, a_2 \in A$ such that $a_1 \in S_1$ and $a_2 \in S_2$?

▶ **Proposition 12** ([8]). *The set splitting problem is* NP*-complete w.r.t.* $\leq_m^{\log}$.

The following proof relies on the fact that strict semantics is considered. It cannot hold for lax semantics unless P = NP.

▶ **Lemma 13.** MC($\mathsf{PInc}_{\mathsf{str}}$) *is* NP*-hard with respect to* $\leq_m^{\log}$.

Proof. We give a reduction from the set splitting problem to the model checking problem of PInc under strict semantics.

Let $\mathcal{F}$ be an instance of the set splitting problem. We stipulate that $\mathcal{F} = \{B_1, \dots, B_n\}$ and that $\bigcup \mathcal{F} = \{a_1, \dots, a_k\}$, where $n, k \in \mathbb{N}$. We will introduce fresh propositions p_i and q_j for each point $a_i \in \bigcup \mathcal{F}$ and set $B_j \in \mathcal{F}$. We will then encode the family of sets $\mathcal{F}$ by assignments over these propositions; each assignment s_i will correspond to a unique point a_i. Formally, let $\tau_{\mathcal{F}}$ denote the set $\{p_1, \dots, p_k, q_1, \dots, q_n, p_\top, p_c, p_d\}$ of propositions. For each $i \in \{1, \dots, k, c, d\}$, we define the assignment $s_i \colon \tau_{\mathcal{F}} \to \{0, 1\}$ as follows:

$$s_i(p) := \begin{cases} 1 & \text{if } p = p_i \text{ or } p = p_\top, \\ 1 & \text{if, for some } j,\ p = q_j \text{ and } a_i \in B_j, \\ 0 & \text{otherwise.} \end{cases}$$

Define $X_{\mathcal{F}} := \{s_1, \dots, s_k, s_c, s_d\}$, that is, $X_{\mathcal{F}}$ consists of assignments s_i corresponding to each of the points $a_i \in \bigcup \mathcal{F}$ and of two auxiliary assignments s_c and s_d. Note that the only assignment in $X_{\mathcal{F}}$ that maps p_c (p_d, resp.) to 1 is s_c (s_d, resp.) and that every assignment maps $p_\top$ to 1. Moreover, note that for $1 \leq i \leq k$ and $1 \leq j \leq n$, $s_i(q_j) = 1$ iff $a_i \in B_j$. Now define

$$\varphi_{\mathcal{F}} := (\neg p_c \wedge \bigwedge_{i \leq n} p_\top \subseteq q_i) \vee (\neg p_d \wedge \bigwedge_{i \leq n} p_\top \subseteq q_i).$$

We claim that $X_{\mathcal{F}} \models_{\mathsf{str}} \varphi_{\mathcal{F}}$ iff the output of the set splitting problem with input $\mathcal{F}$ is "yes". ◀

In Section 5.1 we establish that the model checking problem of modal inclusion logic with strict semantics is in NP (Theorem 24). Since $\mathsf{PInc}_{\mathsf{str}}$ is essentially a fragment of this logic, together with Lemma 13, we obtain the following theorem.

▶ **Theorem 14.** *$\mathrm{MC}(\mathsf{PInc}_{\mathsf{str}})$ is NP-complete with respect to $\leq_m^{\log}$.*

4 Modal logics with team semantics

Let Φ be a set of proposition symbols. The syntax of modal logic $\mathsf{ML}(\Phi)$ is generated by the following grammar: $\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \Diamond\varphi \mid \Box\varphi$, where $p \in \Phi$. By $\varphi^\perp$ we denote the formula that is obtained from $\neg\varphi$ by pushing all negation symbols to the atomic level using the standard duality between $\wedge$ ($\Box$) and $\vee$ ($\Diamond$). A (Kripke) Φ-*model* is a tuple $\mathfrak{M} = (W, R, V)$, where W, called the *domain* of $\mathfrak{M}$, is a non-empty set, $R \subseteq W \times W$ is a binary relation, and $V : \Phi \to \mathcal{P}(W)$ is a valuation of the proposition symbols. By $\models_{\mathsf{ML}}$ we denote the *satisfaction relation* of modal logic that is defined via pointed Φ-*models* in the standard way. Any subset T of the domain of a Kripke model $\mathfrak{M}$ is called *a team of* $\mathfrak{M}$. Before we define *team semantics* for ML, we introduce some auxiliary notation.

▶ **Definition 15.** Let $\mathfrak{M} = (W, R, V)$ be a model and T and S teams of $\mathfrak{M}$. Define that

$$R[T] := \{w \in W \mid \exists v \in T \text{ s.t. } vRw\} \text{ and } R^{-1}[T] := \{w \in W \mid \exists v \in T \text{ s.t. } wRv\}.$$

For teams T and S of $\mathfrak{M}$, we write $T[R]S$ if $S \subseteq R[T]$ and $T \subseteq R^{-1}[S]$.

Accordingly, $T[R]S$ holds if and only if for every $w \in T$, there exists some $v \in S$ such that wRv, and for every $v \in S$, there exists some $w \in T$ such that wRv. We are now ready to define team semantics for ML.

▶ **Definition 16** (Lax team semantics). Let $\mathfrak{M}$ be a Kripke model and T a team of $\mathfrak{M}$. The satisfaction relation $\mathfrak{M}, T \models \varphi$ for $\mathsf{ML}(\Phi)$ is defined as follows.

$$\begin{aligned}
\mathfrak{M}, T \models p &\quad\Leftrightarrow\quad w \in V(p) \text{ for every } w \in T.\\
\mathfrak{M}, T \models \neg p &\quad\Leftrightarrow\quad w \notin V(p) \text{ for every } w \in T.\\
\mathfrak{M}, T \models (\varphi \wedge \psi) &\quad\Leftrightarrow\quad \mathfrak{M}, T \models \varphi \text{ and } \mathfrak{M}, T \models \psi.\\
\mathfrak{M}, T \models (\varphi \vee \psi) &\quad\Leftrightarrow\quad \mathfrak{M}, T_1 \models \varphi \text{ and } \mathfrak{M}, T_2 \models \psi \text{ for some } T_1 \text{ and } T_2 \text{ s.t. } T_1 \cup T_2 = T.\\
\mathfrak{M}, T \models \Diamond\varphi &\quad\Leftrightarrow\quad \mathfrak{M}, T' \models \varphi \text{ for some } T' \text{ s.t. } T[R]T'.\\
\mathfrak{M}, T \models \Box\varphi &\quad\Leftrightarrow\quad \mathfrak{M}, T' \models \varphi, \text{ where } T' = R[T].
\end{aligned}$$

Analogously to the propositional case, we also consider the *strict* variant of team semantics for modal logic. In the *strict* team semantics, we have the following alternative semantic definitions for the disjunction and diamond (where W denotes the domain of $\mathfrak{M}$).

$$\begin{aligned}
\mathfrak{M}, T \models_{\mathsf{str}} (\varphi \vee \psi) &\quad\Leftrightarrow\quad \mathfrak{M}, T_1 \models \varphi \text{ and } \mathfrak{M}, T_2 \models \psi\\
&\qquad\quad \text{for some } T_1 \text{ and } T_2 \text{ such that } T_1 \cup T_2 = T \text{ and } T_1 \cap T_2 = \emptyset.\\
\mathfrak{M}, T \models_{\mathsf{str}} \Diamond\varphi &\quad\Leftrightarrow\quad \mathfrak{M}, f(T) \models \varphi \text{ for some } f : T \to W \text{ s.t. } \forall w \in T : wRf(w).
\end{aligned}$$

Table 2 Complexity of satisfiability, validity and model checking for modal logics under both strict and lax semantics. The given complexity classes refer to completeness results and "-h." denotes hardness. The complexities for Minc and EMinc coincide, see Theorems 23, 24, and 26.
† In [15] NEXP-completeness is claimed. However there is a mistake in the proof and the authors of [15] now have a proof for EXP-completeness.

	Satisfiability		Validity		Model checking	
	strict	lax	strict	lax	strict	lax
ML	- PSPACE [18] -		——— PSPACE [18] ———		——— P [2, 23] ———	
Minc	EXP†	EXP [15]	coNEXP-h. [Th. 25]	coNEXP-h. [Th. 25]	NP [Th. 24]	P [Th. 23]

When L is a team-based modal logic, we let $\mathsf{L}_{\mathsf{str}}$ to denote its variant with strict semantics. As in the propositional case, for strict semantics we use $\models_{\mathsf{str}}$ instead of $\models$. The formulae of ML have the following flatness property.

▶ **Proposition 17** (Flatness, see, e.g., [4]). *Let $\mathfrak{M}$ be a Kripke model and T be a team of $\mathfrak{M}$. Then, for every formula φ of* $\mathsf{ML}(\Phi)$*:* $\mathfrak{M}, T \models \varphi \Leftrightarrow \forall w \in T : \mathfrak{M}, w \models_{\mathsf{ML}} \varphi$.

The syntax of *modal inclusion logic* $\mathsf{Minc}(\Phi)$ and *extended modal inclusion logic* $\mathsf{EMinc}(\Phi)$ is obtained by extending the syntax of $\mathsf{ML}(\Phi)$ by the following grammar rule for each $n \in \mathbb{N}$:

$$\varphi ::= \varphi_1, \ldots, \varphi_n \subseteq \psi_1, \ldots, \psi_n,$$

where $\varphi_1, \psi_1, \ldots, \varphi_n, \psi_n \in \mathsf{ML}(\Phi)$. Additionally, for $\mathsf{Minc}(\Phi)$, we require that φ_1, ψ_1, $\ldots$, φ_n, ψ_n are proposition symbols. The semantics for these inclusion atoms is defined as follows:

$$\mathfrak{M}, T \models \varphi_1, \ldots, \varphi_n \subseteq \psi_1, \ldots, \psi_n \Leftrightarrow \forall w \in T \exists v \in T : \bigwedge_{1 \le i \le n} (\mathfrak{M}, \{w\} \models \varphi_i \Leftrightarrow \mathfrak{M}, \{v\} \models \psi_i).$$

The following proposition is proven in the same way as the analogous results for first-order inclusion logic [5]. A modal logic L is union closed if $\mathfrak{M}, T \models \varphi$ and $\mathfrak{M}, S \models \varphi$ implies that $\mathfrak{M}, T \cup S \models \varphi$, for every $\varphi \in \mathsf{L}$.

▶ **Proposition 18** (Union Closure). *The logics* ML, Minc, EMinc *are union closed.*

Analogously to the propositional case, it is easy to establish that for ML the strict and the lax semantics coincide (for a proof in the first-order setting see [5]). Again, as in the propositional case, this does not hold for Minc or EMinc. Note that since $\mathsf{PInc}_{\mathsf{str}}$ is not union closed, neither is $\mathsf{Minc}_{\mathsf{str}}$, nor $\mathsf{EMinc}_{\mathsf{str}}$.

In contrary to the propositional case, Lemma 5 fails in the modal case as the following example illustrates.

▶ **Example 19.** Let $\mathfrak{M}$ be as depicted in the table of Figure 1 and let φ denote the $\mathsf{PInc}_{\mathsf{str}}$-formula of Example 4. Now $\mathfrak{M}, \{w_i\} \models_{\mathsf{str}} \Box\varphi$, for $i \in \{1, 2, 3\}$, but $\mathfrak{M}, \{w_1, w_2, w_3\} \not\models_{\mathsf{str}} \Box\varphi$.

5 Model checking and validity in modal team semantics

The model checking, satisfiability, and validity problems in the context of team semantics of modal logic are defined analogously to the propositional case. Let $\mathsf{L}(\Phi)$ be a modal logic with team semantics. A formula $\varphi \in \mathsf{L}(\Phi)$ is *satisfiable*, if there exists a Kripke Φ-model

$\mathfrak{M}$ and a non-empty team T of $\mathfrak{M}$ such that $\mathfrak{M}, T \models \varphi$. A formula $\varphi \in \mathsf{L}(\Phi)$ is *valid*, if $\mathfrak{M}, T \models \varphi$ holds for every Φ-model $\mathfrak{M}$ and every team T of $\mathfrak{M}$. The satisfiability problem SAT(L) and the validity problem VAL(L) are defined in the obvious way: Given a formula $\varphi \in \mathsf{L}$, decide whether the formula is satisfiable (valid, respectively). For model checking MC(L) we consider combined complexity: Given a formula $\varphi \in \mathsf{L}$, a Kripke model $\mathfrak{M}$, and a team T of $\mathfrak{M}$, decide whether $\mathfrak{M}, T \models \varphi$. See Table 2 for known complexity results on ML and Minc, together with partial results of this paper.

5.1 Complexity of model checking

Let $\mathfrak{M}$ be a Kripke model, T be a team of $\mathfrak{M}$, and φ be a formula of Minc. By maxsub(T, φ), we denote the maximum subteam T' of T such that $\mathfrak{M}, T' \models \varphi$. Since Minc is union closed (cf. Proposition 18), such a maximum subteam always exists.

For a proof of the following lemma, see the full version [14] of this article.

▶ **Lemma 20.** *If φ is a proposition symbol, its negation, or an inclusion atom, then maxsub(T, φ) can be computed in polynomial time with respect to $|T| + |\varphi|$.*

For the following lemma it is crucial that lax semantics is considered. The lemma cannot hold for strict semantics unless $\mathsf{P} = \mathsf{NP}$.

▶ **Lemma 21.** MC(Minc) *under lax semantics is in* P.

Proof. We will present a labelling algorithm for model checking $\mathfrak{M}, T \models \varphi$. Let subOcc$(\varphi)$ denote the set of all *occurrences* of subformulae of φ. Below we denote occurrences as if they were formulae, but we actually refer to some particular occurrence of the formula.

A function $f\colon \text{subOcc}(\varphi) \to \mathcal{P}(W)$ is called a labelling function of φ in $\mathfrak{M}$. We will next give an algorithm for computing a sequence $f_0, f_1, f_2, \ldots$, of such labelling functions.

- Define $f_0(\psi) = W$ for each $\psi \in \text{subOcc}(\varphi)$.
- For odd $i \in \mathbb{N}$, define $f_i(\psi)$ bottom up as follows:
  1. For literal ψ, define $f_i(\psi) := \text{maxsub}(f_{i-1}(\psi), \psi)$.
  2. $f_i(\psi \wedge \theta) := f_i(\psi) \cap f_i(\theta)$.
  3. $f_i(\psi \vee \theta) := f_i(\psi) \cup f_i(\theta)$.
  4. $f_i(\Diamond\psi) := \{w \in f_{i-1}(\Diamond\psi) \mid R[w] \cap f_i(\psi) \neq \emptyset\}$.
  5. $f_i(\Box\psi) := \{w \in f_{i-1}(\Box\psi) \mid R[w] \subseteq f_i(\psi)\}$.
- For even $i \in \mathbb{N}$ larger than 0, define $f_i(\psi)$ top to bottom as follows:
  1. Define $f_i(\varphi) := f_{i-1}(\varphi) \cap T$.
  2. If $\psi = \theta \wedge \gamma$, define $f_i(\theta) := f_i(\gamma) := f_i(\theta \wedge \gamma)$.
  3. If $\psi = \theta \vee \gamma$, define $f_i(\theta) := f_{i-1}(\theta) \cap f_i(\theta \vee \gamma)$ and $f_i(\gamma) := f_{i-1}(\gamma) \cap f_i(\theta \vee \gamma)$.
  4. If $\psi = \Diamond\theta$, define $f_i(\theta) := f_{i-1}(\theta) \cap R[f_i(\Diamond\theta)]$.
  5. If $\psi = \Box\theta$, define $f_i(\theta) := f_{i-1}(\theta) \cap R[f_i(\Box\theta)]$.

By a straightforward induction on i, we can prove that $f_{i+1}(\psi) \subseteq f_i(\psi)$ holds for every $\psi \in \text{subOcc}(\varphi)$. The only nontrivial induction step is that for $f_{i+1}(\theta)$ and $f_{i+1}(\gamma)$, when $i+1$ is even and $\psi = \theta \wedge \gamma$. To deal with this step, observe that, by the definition of f_{i+1} and f_i, we have $f_{i+1}(\theta) = f_{i+1}(\gamma) = f_{i+1}(\psi)$ and $f_i(\psi) \subseteq f_i(\theta), f_i(\gamma)$, and by the induction hypothesis on ψ, we have $f_{i+1}(\psi) \subseteq f_i(\psi)$.

It follows that there is an integer $j \leq 2 \cdot |W| \cdot |\varphi|$ such that $f_{j+2} = f_{j+1} = f_j$. We denote this fixed point f_j of the sequence $f_0, f_1, f_2, \ldots$ by f_∞. By Lemma 20 the outcome of maxsub$(\cdot, \cdot)$ is computable in polynomial time with respect to its input. That being, clearly f_{i+1} can be computed from f_i in polynomial time with respect to $|W| + |\varphi|$. On that account f_∞ is also computable in polynomial time with respect to $|W| + |\varphi|$.

We will next prove by induction on $\psi \in \mathrm{subOcc}(\varphi)$ that $\mathfrak{M}, f_\infty(\psi) \models \psi$. Note first that there is an odd integer i and an even integer j such that $f_\infty = f_i = f_j$.

1. If ψ is a literal, the claim is true since $f_\infty = f_i$ and $f_i(\psi) = \mathrm{maxsub}(f_{i-1}(\psi), \psi)$.
2. Assume next that $\psi = \theta \wedge \gamma$, and the claim holds for θ and γ. Since $f_\infty = f_j$, we have $f_\infty(\psi) = f_\infty(\theta) = f_\infty(\gamma)$, as a result, by induction hypothesis, $\mathfrak{M}, f_\infty(\psi) \models \theta \wedge \gamma$.
3. In the case $\psi = \theta \vee \gamma$, we obtain the claim $\mathfrak{M}, f_\infty(\psi) \models \psi$ by using the induction hypothesis, and the observation that $f_\infty(\psi) = f_i(\psi) = f_i(\theta) \cup f_i(\gamma) = f_\infty(\theta) \cup f_\infty(\gamma)$.
4. Assume then that $\psi = \Diamond\theta$. Since $f_\infty = f_i$, we have $f_\infty(\psi) = \{w \in f_{i-1}(\psi) \mid R[w] \cap f_\infty(\theta) \neq \emptyset\}$, as a consequence $f_\infty(\psi) \subseteq R^{-1}[f_\infty(\theta)]$. On the other hand, since $f_\infty = f_j$, we have $f_\infty(\theta) = f_{j-1}(\theta) \cap R[f_\infty(\psi)]$, for this reason $f_\infty(\theta) \subseteq R[f_\infty(\psi)]$. Thus $f_\infty(\psi)[R]f_\infty(\theta)$, and using the induction hypothesis, we see that $\mathfrak{M}, f_\infty(\psi) \models \psi$.
5. Assume finally that $\psi = \Box\theta$. Since $f_\infty = f_i$, we have $R[f_\infty(\psi)] \subseteq f_\infty(\theta)$. On the other hand, since $f_\infty = f_j$, we have $f_\infty(\theta) \subseteq R[f_\infty(\psi)]$. This shows that $f_\infty(\theta) = R[f_\infty(\psi)]$, that being the case by the induction hypothesis, $\mathfrak{M}, f_\infty(\psi) \models \psi$.

In particular, if $f_\infty(\varphi) = T$, then $\mathfrak{M}, T \models \varphi$. Consequently, to complete the proof of the lemma, it suffices to prove that the converse implication is true, as well. To prove this, assume that $\mathfrak{M}, T \models \varphi$. Then for each $\psi \in \mathrm{subOcc}(\varphi)$, there is a team T_ψ such that

1. $T_\varphi = T$.
2. If $\psi = \theta \wedge \gamma$, then $T_\psi = T_\theta = T_\gamma$.
3. If $\psi = \theta \vee \gamma$, then $T_\psi = T_\theta \cup T_\gamma$.
4. If $\psi = \Diamond\theta$, then $T_\psi[R]T_\theta$.
5. If $\psi = \Box\theta$, then $T_\theta = R[T_\psi]$.
6. If ψ is a literal, then $\mathfrak{M}, T_\psi \models \psi$.

We prove by induction on i that $T_\psi \subseteq f_i(\psi)$ for all $\psi \in \mathrm{subOcc}(\varphi)$. For $i = 0$, this is obvious, since $f_0(\psi) = W$ for all ψ. Assume next that $i+1$ is odd and the claim is true for i. We prove the claim $T_\psi \subseteq f_i(\psi)$ by induction on ψ.

1. If ψ is a literal, then $f_{i+1}(\psi) = \mathrm{maxsub}(f_i(\psi), \psi)$. Since $\mathfrak{M}, T_\psi \models \psi$, and by induction hypothesis, $T_\psi \subseteq f_i(\psi)$, the claim $T_\psi \subseteq f_{i+1}(\psi)$ is true.
2. Assume that $\psi = \theta \wedge \gamma$. By induction hypothesis on θ and γ, we have $T_\psi = T_\theta \subseteq f_{i+1}(\theta)$ and $T_\psi = T_\gamma \subseteq f_{i+1}(\gamma)$. For this reason, we get $T_\psi \subseteq f_{i+1}(\theta) \cap f_{i+1}(\gamma) = f_{i+1}(\psi)$.
3. The case $\psi = \theta \vee \gamma$ is similar to the previous one; we omit the details.
4. If $\psi = \Diamond\theta$, then $f_{i+1}(\psi) = \{w \in f_i(\psi) \mid R[w] \cap f_{i+1}(\theta) \neq \emptyset\}$. By the two induction hypotheses on i and θ, we have $\{w \in T_\psi \mid R[w] \cap T_\theta \neq \emptyset\} \subseteq f_{i+1}(\psi)$. The claim follows from this, since the condition $R[w] \cap T_\theta \neq \emptyset$ holds for all $w \in T_\psi$.
5. The case $\psi = \Box\theta$ is again similar to the previous one, so we omit the details.

Assume then that $i+1$ is even and the claim is true for i. This time we prove the claim $T_\psi \subseteq f_i(\psi)$ by top to bottom induction on ψ.

1. By assumption, $T_\varphi = T$, whence by induction hypothesis, $T_\varphi \subseteq f_i(\varphi) \cap T = f_{i+1}(\varphi)$.
2. Assume that $\psi = \theta \wedge \gamma$. By induction hypothesis on ψ, we have $T_\psi \subseteq f_{i+1}(\psi)$. Since $T_\psi = T_\theta = T_\gamma$ and $f_{i+1}(\psi) = f_{i+1}(\theta) = f_{i+1}(\gamma)$, this implies that $T_\theta \subseteq f_{i+1}(\theta)$ and $T_\gamma \subseteq f_{i+1}(\gamma)$.
3. Assume that $\psi = \theta \vee \gamma$. Using the fact that $T_\theta \subseteq T_\psi$, and the two induction hypotheses on i and ψ, we see that $T_\theta \subseteq f_i(\theta) \cap T_\psi \subseteq f_i(\theta) \cap f_{i+1}(\psi) = f_{i+1}(\theta)$. Similarly, we see that $T_\gamma \subseteq f_{i+1}(\gamma)$.

4. Assume that $\psi = \Diamond\theta$. By the induction hypothesis on i, we have $T_\theta \subseteq f_i(\theta)$, and by the induction hypothesis on ψ, we have $T_\theta \subseteq R[T_\psi] \subseteq R[f_{i+1}(\psi)]$. Accordingly, we see that $T_\theta \subseteq f_i(\theta) \cap R[f_{i+1}(\psi)] = f_{i+1}(\theta)$.
5. The case $\psi = \Box\theta$ is similar to the previous one; we omit the details.

It follows now that $T = T_\varphi \subseteq f_\infty(\varphi)$. Since $f_\infty(\varphi) \subseteq f_2(\varphi) \subseteq T$, we conclude that $f_\infty(\varphi) = T$. This completes the proof of the implication $\mathfrak{M}, T \models \varphi \;\Rightarrow\; f_\infty(\varphi) = T$. ◀

The following lemma then follows, since in the context of model checking, we may replace modal formulae that appear as parameters in inclusion atoms by fresh proposition symbols with the same extension.

▶ **Lemma 22.** MC(EMinc) *under lax semantics is in* P.

By combining Lemmas 9, 21, and 22, we obtain the following theorem.

▶ **Theorem 23.** MC(Minc) *and* MC(EMinc) *are* P*-complete w.r.t.* $\leq_m^{\log}$.

▶ **Theorem 24.** MC(Minc$_{\mathsf{str}}$) *and* MC(EMinc$_{\mathsf{str}}$) *are* NP*-complete w.r.t.* $\leq_m^{\log}$.

Proof. The NP-hardness follows from the propositional case, i.e., from Lemma 13.

In order to establish inclusion, we note that the obvious brute force algorithm for model checking for EMinc works in NP: For disjunctions and diamonds, we use nondeterminism to guess the correct partitions or successor teams, respectively. Conjunctions are dealt with sequentially and for boxes the unique successor team can be computed by brute force in quadratic time. Checking whether a team satisfies an inclusion atom or a (negated) proposition symbol can be computed by brute force in polynomial time (this also follows directly from Lemma 20). ◀

5.2 Complexity of validity

The following result involves a reduction from a complement problem of the validity problem of dependency quantified Boolean formulas [20]. The details can be found in the full version [14] of this article.

▶ **Theorem 25.** VAL(Minc) *and* VAL(Minc$_{\mathsf{str}}$) *are* coNEXP*-hard w.r.t.* $\leq_m^{\log}$.

While the exact complexities of the problems VAL(Minc) and VAL(EMinc) remain open, it is straightforward to establish that the complexities coincide. In the proof of the theorem below, we introduce fresh proposition symbols for each modal formula that appears as a parameter for an inclusion atom. We then replace these formulas by the fresh proposition symbols and separately force, by using ML, that the extensions of the proposition symbols and modal formulae are the same, respectively. See [14] for a detailed proof.

▶ **Theorem 26.** *Let* C *be a complexity class that is closed under polynomial time reductions. Then* VAL(Minc) *under lax (strict) semantics in complete for* C *if and only if* VAL(EMinc) *under lax (strict) semantics in complete for* C.

6 Conclusion

In this paper, we investigated the computational complexity of model checking and validity for propositional and modal inclusion logic to complete the complexity landscape of these problems in the mentioned logics. In particular, we gave emphasis to the subtle influence of which semantics is considered: strict or lax. The model checking problem for these logics under strict semantics was shown to be NP-complete and under lax semantics P-complete. The validity problem was shown to be coNP-complete for the propositional strict semantics case, for the lax semantics coNP-completeness was established earlier by Hannula et al. [12]. For the modal case, we obtained a coNEXP lower bound under lax as well as strict semantics. The upper bound is left open for further research. We however established that, if closed under polynomial time reductions, the complexities of VAL(Minc) and VAL(EMinc), and VAL($\mathsf{Minc}_{\mathsf{str}}$) and VAL($\mathsf{EMinc}_{\mathsf{str}}$) coincide, respectively.

We conclude with an open problem. Let ML($\sim$) denote the extension of modal logic by the contradictory negation $\sim$ with the following semantics: $\mathfrak{M}, T \models {\sim}\varphi$ iff $\mathfrak{M}, T \not\models \varphi$. What is the complexity of VAL(ML($\sim$))? It is known that VAL(PL($\sim$)) is complete for *alternating exponential time with polynomially many alternations* [13]; this is the best known lower bound for VAL(ML($\sim$)). Decidability with non-elementary upper bound can be obtained, e.g., by using team-bisimulation and Hintikka-types; no better upper bound is known.

References

1 S. R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proc. 19th STOC*, pages 123–131, 1987.

2 E. Clarke, E. A. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM ToPLS*, 8(2):244–263, 1986.

3 S. A. Cook. The complexity of theorem proving procedures. In *Proc. 3rd STOC*, pages 151–158, 1971.

4 A. Durand, J. Kontinen, and H. Vollmer. Expressivity and complexity of dependence logic. In S. Abramsky, J. Kontinen, J. Väänänen, and H. Vollmer, editors, *Dependence Logic: Theory and Applications*, pages 5–32. Springer, 2016.

5 P. Galliani. Inclusion and exclusion dependencies in team semantics - on some logics of imperfect information. *Ann. Pure Appl. Logic*, 163(1):68–84, 2012. `doi:10.1016/j.apal.2011.08.005`.

6 P. Galliani, M. Hannula, and J. Kontinen. Hierarchies in independence logic. In *Proc. 22nd CSL*, volume 23 of *LIPIcs*, pages 263–280, 2013.

7 P. Galliani and L. Hella. Inclusion logic and fixed point logic. In *Proc. 22nd CSL*, LIPIcs, pages 281–295, 2013. `doi:10.4230/LIPIcs.CSL.2013.281`.

8 M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

9 L. M. Goldschlager. The monotone and planar circuit value problems are log-space complete for P. *SIGACT News*, 9:25–29, 1977.

10 E. Grädel and J. Väänänen. Dependence and independence. *Studia Logica*, 101(2):399–410, 2013.

11 M. Hannula and J. Kontinen. Hierarchies in independence and inclusion logic with strict semantics. *J. Log. Comput.*, 25(3):879–897, 2015. `doi:10.1093/logcom/exu057`.

12 M. Hannula, J. Kontinen, J. Virtema, and H. Vollmer. Complexity of propositional independence and inclusion logic. In *Proc. 40th MFCS*, pages 269–280, 2015. `doi:10.1007/978-3-662-48057-1_21`.

13 M. Hannula, J. Kontinen, J. Virtema, and H. Vollmer. Complexity of propositional logics in team semantics. *CoRR, extended version of [12]*, abs/1504.06135, 2015.

14 L. Hella, A. Kuusisto, A. Meier, and J. Virtema. Model checking and validity in propositional and modal inclusion logics. *CoRR*, abs/1609.06951, 2016.

15 L. Hella, A. Kuusisto, A. Meier, and H. Vollmer. Modal inclusion logic: Being lax is simpler than being strict. In *Proc. 40th MFCS*, pages 281–292, 2015. `doi:10.1007/978-3-662-48057-1_22`.

16 L. Hella and J. Stumpf. The expressive power of modal logic with inclusion atoms. In *Proc. 6th GandALF*, pages 129–143, 2015. `doi:10.4204/EPTCS.193.10`.

17 W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL*, 5(4):539–563, 1997.

18 R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

19 L. A. Levin. Universal sorting problems. *Problems of Inform. Transm.*, 9:265–266, 1973.

20 G. Peterson, J. Reif, and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Math. with Applications*, 41(7-8):957–992, 2001. `doi:10.1016/S0898-1221(00)00333-3`.

21 K. Sano and J. Virtema. Characterizing frame definability in team semantics via the universal modality. In *Proc. of WoLLIC 2015*, pages 140–155, 2015.

22 K. Sano and J. Virtema. Characterizing relative frame definability in team semantics via the universal modality. In *Proc. of WoLLIC 2016*, pages 392–409, 2016.

23 P. Schnoebelen. The complexity of temporal logic model checking. In *Proc. 4th AiML*, pages 393–436, 2002.

24 J. Väänänen. *Dependence Logic.* Cambridge University Press, 2007.

25 H. Vollmer. *Introduction to Circuit Complexity – A Uniform Approach.* Texts in Theoretical Computer Science. Springer Verlag, Berlin Heidelberg, 1999.