Model checking and validity in propositional and modal inclusion logics*

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

[1] University of Tampere, Finland, lauri.hella@uta.fi
[2] University of Bremen, Germany, antti.j.kuusisto@gmail.com
[3] Leibniz Universitt Hannover, Germany, meier@thi.uni-hannover.de
[4] University of 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.

Keywords: Inclusion Logic – Model Checking – Complexity

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 [19], where it was shown that Hintikka's IF-logic can be based on a compositional (as opposed to game-theoretic) semantics. In 2007, Väänänen [31] 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*. Clearly 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.

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 [12], and *inclusion logic* of Galliani [7]. 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 equal-expressive with existential second-order logic and accordingly capture the complexity class NP [31, 12]. Surprisingly, inclusion logic has the same expressive power as *positive greatest fixed point logic* GFP^+ [9]. Since on finite structures, GFP^+ coincides with

*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.

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 [8]. 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 $\mathcal{O}(n^k)$) [15]. That being so, indeed, inclusion logic and its fragments have very interesting 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. [5] 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.). 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 [18]. From this, a corresponding characterization for extended propositional inclusion logic follows directly. In [26, 27] (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. [16]. The study on the validity problem of propositional inclusion logic was initiated by Hannula et al. [13], 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 closed under negation, the connection between the satisfiability problem and the validity problem fails. In [13] 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, we obtain remarkable in the modal 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; we establish coNEXP-hardness for both strict and lax semantics. For an overview of the results of this paper together with the known complexity results from the literature, see tables 3–5 on page 17.

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:

$$\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi), \text{ where } p \in \Phi.$$

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.*

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

The lax team semantics is considered 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:

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

When L denotes a team-based propositional logic, we let L_s denote the variant of the logic with strict semantics. Moreover, in order to improve readability, for strict semantics we use $\models_s$ 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 ([31]) *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 3 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. It is easy to see that this extension does not increase complexity of the logic and on that account, in this paper, we only consider the non-extended variant.*

It is easy to check that PInc is not a downward closed logic[1]. However, analogously to FO-inclusion-logic [7], the same holds for PInc w.r.t. unions:

Proposition 4 (Closure under unions) *Let $\varphi \in \mathsf{PInc}$ and let X_i, for $i \in I$, be teams. Suppose that $X_i \models \varphi$ for each $i \in I$. Then $\bigcup_{i \in I} X_i \models \varphi$.*

It is easy to see that, by Proposition 2, for propositional logic the strict and the lax semantics coincide; meaning that $X \models \varphi$ iff $X \models_s \varphi$ for all X and φ. However this does not hold for propositional inclusion logic, for the following example shows that PInc_s is not union closed. Moreover, we will show that the two different semantics lead to different complexities for the related model checking problems.

Example 5 *Let s_1, s_2, and s_3 be as in Table 1 and define $\varphi := \big(p \wedge (p \subseteq r)\big) \vee \big(q \wedge (q \subseteq r)\big)$. Note that $\{s_1, s_2\} \models_s \varphi$ and $\{s_2, s_3\} \models_s \varphi$, but $\{s_1, s_2, s_3\} \not\models_s \varphi$.*

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

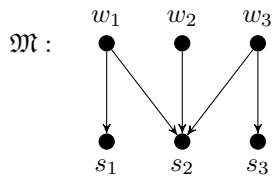

Figure 1: Assignments for teams in Example 5 and the Kripke model for Example 20.

	Satisfiability		Validity		Model checking	
	strict	lax	strict	lax	strict	lax
PL	NP [4, 22]		coNP [4, 22]		NC[1] [2]	
PInc	EXP [17]	EXP [16]	coNP [Th. 7]	coNP [13]	NP [Th. 15]	P [Th. 11]

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.

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 $\varphi \in \mathsf{L}$ is *satisfiable*, if there exists a non-empty team X such that $X \models \varphi$. A formula $\varphi \in \mathsf{L}$ is *valid*, if $X \models \varphi$ holds for all teams X such that the proposition symbols in φ are in the domain of X. 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 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 in [13] that the validity problem of PInc is coNP-complete. Here we establish that the corresponding problem for PInc_s is also coNP-complete. Our proof is similar to the one in [13]. However the proof of [13] uses the fact that PInc is union closed, while the same is not true for PInc_s (cf. Example 5).

Lemma 6 *Let X be a propositional team and $\varphi \in \mathsf{PInc}_s$. If $\{s\} \models_s \varphi$ for every $s \in X$ then $X \models_s \varphi$.*

Proof The proof is by a simple induction on the structure of the formula. The cases for atomic formulae and conjunction are trivial. The case for disjunction is easy: Assume that $\{s\} \models_s \varphi \vee \psi$ for every $s \in X$. Consequently for every $s \in X$ either $\{s\} \models_s \varphi$ or $\{s\} \models_s \psi$. As a result there exists Y and Z such that $Y \cup Z = X$, $Y \cap Z = \emptyset$, $\forall s \in Y : \{s\} \models_s \varphi$, and $\forall s \in Z : \{s\} \models_s \psi$. By the induction hypothesis $Y \models_s \varphi$ and $Z \models_s \psi$. Consequently, $X \models_s \varphi \vee \psi$. ∎

Theorem 7 *The validity problem for PInc_s is coNP-complete w.r.t. $\leq_m^{\log}$.*

Proof The coNP-hardness follows via Proposition 2 from the fact that the validity problem of PL is coNP-hard. Accordingly, it suffices to show VAL(PInc_s) $\in$ coNP. It is easy to check that, by Lemma 6, a formula $\varphi \in \mathsf{PInc}_s$ is valid iff it is satisfied by all singleton teams $\{s\}$. Note also that, over a singleton team $\{s\}$, an inclusion atom $(p_1, \ldots, p_n) \subseteq (q_1, \ldots, q_n)$ is equivalent to the PL-formula

$$\bigwedge_{1 \leq i \leq n} (p_i \wedge q_i) \vee (\neg p_i \wedge \neg q_i).$$

[1] A logic L is downward closed if the implication $X \models \varphi$ and $Y \subseteq X \Rightarrow Y \models \varphi$ holds for every formula $\varphi \in \mathsf{L}$ and teams X and Y.

Denote by φ^* the PL-formula obtained by replacing all inclusion atoms in φ by their PL-translations. By the above, φ is valid iff φ^* is valid. Since VAL(PL) is in coNP the claim follows. ∎

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 [33] by Vollmer.

Definition 8 *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 9 ([11]) MCVP *is* P*-complete w.r.t. $\leq_m^{\log}$ reductions.*

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

Proof We will establish a LOGSPACE-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 and then discuss the background intuition in more detail.

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

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

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

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

It is quite straightforward to check (see details below) 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 $\psi_{\text{out}=1}$ 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 $s_\bot$ and the proposition $p_\bot$ 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 $p_i \subseteq p_j$.

Now observe that the team $X_{C,\vec{b}}$ can be easily computed by a logspace Turing machine which scans the input for $\wedge$-gates, $\vee$-gates, and true input gates, and then outputs the corresponding team members s_i in a bitwise fashion. The formula φ_C can be computed in logspace as well:

1. the left disjunct does not depend on the input,
2. for $\psi_\wedge$ we only need to scan for the $\wedge$-gates and output the inclusion-formulae for the corresponding edges,
3. for $\psi_\vee$ we need to maintain two binary counters for i and j, and use them for searching for those disjunction gates that satisfy $i < j$.

Consequently, 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 next, we can show a stronger result for the model checking problem of PInc_s; 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 22). Since PInc is essentially a fragment of this logic, by combining Lemmas 10 and 22, we obtain the following theorem.

Theorem 11 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 PInc_s.

Definition 12 *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 13 ([10]) *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 $\mathsf{P} = \mathsf{NP}$.

Lemma 14 MC(PInc) *under strict semantics is* NP*-hard w.r.t. $\leq_m^{\log}$.*

Proof We give a reduction from the set splitting problem [10, SP4] to the model checking problem of PInc under strict semantics.

Let $\mathcal{F}$ be an instance of the set splitting problem. We stipulate that $\mathcal{F} = \{B_1, \ldots, B_n\}$ and that $\bigcup \mathcal{F} = \{a_1, \ldots, a_k\}$, where $n, k \in \mathbb{N}$. We will introduce fresh proposition symbols 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 proposition symbols; each assignment s_i will correspond to a unique point a_i. Formally, let $\tau_{\mathcal{F}}$ denote the set $\{p_1, \ldots, p_k, q_1, \ldots, q_n, p_\top, p_c, p_d\}$ of proposition symbols. For each $i \in \{1, \ldots, k, c, d\}$, 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 $X_{\mathcal{F}} := \{s_1, \ldots, s_k, s_c, s_d\}$, 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

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

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

The proof is straightforward. Note that $X_{\mathcal{F}} \models_s \varphi_{\mathcal{F}}$ holds iff $X_{\mathcal{F}}$ can be partitioned into two subteams Y_1 and Y_2 such that

$$Y_1 \models_s \neg p_c \wedge \bigwedge_{i \leq n} p_\top \subseteq q_i \text{ and } Y_2 \models_s \neg p_d \wedge \bigwedge_{i \leq n} p_\top \subseteq q_i.$$

■

Teams Y_1 and Y_2 are both nonempty, since $s_d \in Y_1$ and $s_c \in Y_2$. Also, for a nonempty subteam Y of $X_{\mathcal{F}}$, it holds that $Y \models_s p_\top \subseteq q_j$ iff there exists $s_i \in Y$ such that $s_i(q_j) = 1$, or equivalently, $a_i \in B_j$.

It is now evident that if $X_{\mathcal{F}} \models_s \varphi_{\mathcal{F}}$ holds then the related subteams Y_1 and Y_2 directly construct a positive answer to the set splitting problem. Likewise, any positive answer to the set splitting problem can be used to directly construct the related subteams Y_1 and Y_2.

In order to compute the assignments s_i and by this the team $X_{\mathcal{F}}$ on a logspace machine we need to implement two binary counters to count through $1 \leq i \leq k$ for the propositions p_i and $1 \leq j \leq n$ for the propositions q_i. The formula $\varphi_{\mathcal{F}}$ is constructed in logspace by simply outputting it step by step with the help of a binary counter for the interval $1 \leq i \leq n$. As a result the whole reduction can be implemented on a logspace Turing machine.

In Section 5.1 we establish that the model checking problem of modal inclusion logic with strict semantics is in NP (Theorem 25). Since PInc is essentially a fragment of this logic, together with Lemma 14, we obtain the following theorem.

Theorem 15 MC(PInc) *under strict semantics is* NP*-complete w.r.t.* $\leq_m^{\log}$.

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:

$$\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \Diamond\varphi \mid \Box\varphi, \text{where } p \in \Phi.$$

By $\varphi^{\perp}$ we denote the formula that is obtained from $\neg\varphi$ by pushing all negation symbols to the atomic level. 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 $V\colon \Phi \to \mathcal{P}(W)$ 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 16 *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 17 (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.*

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

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 $\mathfrak{M}$).

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

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

Proposition 18 (Flatness, see, e.g., [5]) *Let $\mathfrak{M}$ be a Kripke model and T be a team of $\mathfrak{M}$. Then, for every formula φ of $\mathsf{ML}(\Phi)$: $\mathfrak{M}, T \models \varphi \Leftrightarrow \forall w \in T : \mathfrak{M}, w \models_{\mathsf{ML}} \varphi$.*

	Satisfiability		Validity		Model checking	
	strict	lax	strict	lax	strict	lax
ML	— PSPACE [21] —		——— PSPACE [21] ———		——— P [3, 28] ———	
Minc	EXP [17]	EXP [16]	coNEXP-h. [C. 32]	coNEXP-h. [L. 31]	NP [Th. 25]	P [Th. 24]

Table 2: Complexity of satisfiability, validity and model checking for modal logics under both systems of semantics. The given complexity classes refer to completeness results and "-h." denotes hardness. The complexities for Minc and EMinc coincide, see Theorems 24, 25, and 33.

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, …, φ_n, ψ_n are proposition symbols in Φ. The semantics for these inclusion atoms is defined as follows:

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

The following proposition is proven in the same way as the analogous results for first-order inclusion logic [7]. 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 19 (Union Closure) *The logics* ML, Minc, EMinc *are union closed.*

Analogously to the propositional case, it is easy to see that, by Proposition 18, for ML the strict and the lax semantics coincide. Again, as in the propositional case, this does not hold for Minc or EMinc. (Note that since PInc_s is not union closed (cf. Example 5) neither Minc_s nor EMinc_s is as well.)

In contrary to the propositional case, Lemma 6 fails in the modal case as the following example illustrates.

Example 20 *Let $\mathfrak{M}$ be as depicted in the table of Figure 1 and let φ denote the PInc_s-formula of Example 5. Now $\mathfrak{M}, \{w_i\} \models_s \Box\varphi$, for $i \in \{1, 2, 3\}$, but $\mathfrak{M}, \{w_1, w_2, w_3\} \not\models_s \Box\varphi$.*

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 $\mathrm{maxsub}(T, \varphi)$, we denote the maximum subteam T' of T such that $\mathfrak{M}, T' \models \varphi$. Since Minc is union closed (cf. Proposition 19), such a maximum subteam always exists.

Lemma 21 *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 $|T| + |\varphi|$.*

PROOF If φ is a proposition symbol or its negation, the claim follows from flatness in a straightforward way. Assume then that $T = \{w_1, \dots w_n\}$ and $\varphi = p_1, \dots, p_k \subseteq q_1, \dots, q_k$. Let $G = (V, E)$ be a directed graph such that $V = T$ and $(u, v) \in E$ iff the value of p_i in u is the same as the value of q_i in v, for each $1 \leq i \leq k$.

The graph G describes the inclusion dependencies between the points in the following sense: if $w \in \text{maxsub}(T, \varphi)$, then there exists some $v \in \text{maxsub}(T, \varphi)$ such that $(w, v) \in E$. Clearly G can be computed in time $\mathcal{O}(n^2 k)$. In order to construct maxsub(T, φ), we, round by round, delete all vertices from G with out-degree 0. Formally, we define a sequence $G_0, \dots, G_n$ of graphs recursively. We define that $G_0 := G$ and that G_{j+1} is the graph obtained from G_j by deleting all of those vertices from G_j that have out-degree 0 in G_j. Let i be the smallest integer such that $G_i = (V_i, E_i)$ has no vertices of out-degree 0. Clearly $i \leq n$, and moreover, G_i is computable from G in time $\mathcal{O}(n^3)$. It is easy to check that $V_i = \text{maxsub}(T, \varphi)$. ∎

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 22 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 $f_0, f_1, f_2, \dots$, 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 $f_0, f_1, f_2, \dots$ by f_∞. By Lemma 21 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 $\psi \in \text{subOcc}(\varphi)$ 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, $\mathfrak{M}, f_\infty(\psi) \models \theta \wedge \gamma$, as desired.

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 $f_\infty(\theta) \subseteq R[f_\infty(\psi)]$. Accordingly, $f_\infty(\psi)[R]f_\infty(\theta)$, 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, $T_\varphi = T$, on that account 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 $\mathfrak{M}, T \models \varphi \;\Rightarrow\; f_\infty(\varphi) = T$. ∎

Lemma 23 MC(EMinc) *under lax semantics is in* P.

Proof The result follows by a polynomial time reduction to the model checking problem of Minc: Let $(W, R, V), T$ be a team pointed Kripke model and φ be a formula of EMinc. Let $\varphi_1, \ldots, \varphi_n$ be exactly those subformulae of φ that occur as a parameter of some inclusion atom in φ and let $p_1, \ldots, p_n$ be distinct fresh proposition symbols. Let V' be a valuation defined as follows

$$V'(p) := \begin{cases} \{w \in W \mid (W, R, V), w \models_{\mathsf{ML}} \varphi_i\} & \text{if } p = p_i, \\ V(p) & \text{otherwise.} \end{cases}$$

Let φ^* denote the formula obtained from φ by simultaneously substituting each φ_i by p_i. It is easy to check that $(W, R, V), T \models \varphi$ if and only if $(W, R, V'), T \models \varphi^*$. Moreover, φ^* can be clearly computed form φ in polynomial time. Likewise, V' can be computed in polynomial time; since each φ_i is a modal formula the truth set of that formula in (W, R, V) can be computed in polynomial time by the standard labelling algorithm used in modal logic (see e.g., [1]), and the numbers of such computations is bounded above by the size of φ. As a consequence the result follows form Lemma 22. ∎

By combining Lemmas 10, 22, and 23, we obtain the following theorem.

Theorem 24 MC(Minc) *and* MC(EMinc) *under lax semantics are* P*-complete w.r.t.* $\leq_m^{\log}$.

Theorem 25 MC(Minc) *and* MC(EMinc) *under strict semantics are* NP*-complete w.r.t.* $\leq_m^{\log}$.

Proof The NP-hardness follows from the propositional case, i.e., by Lemma 14.

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 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 21). ∎

5.2 Dependency quantifier Boolean formulae

Deciding whether a given quantified Boolean formula (qBf) is valid is a canonical PSPACE-complete problem. *Dependency quantifier Boolean formulae* introduced by Peterson et al. [25] are variants of qBfs for which the corresponding decision problem is NEXP-complete. In this section, we define the related coNEXP-complete complementary problem. For the definitions related to dependency quantifier Boolean formulae, we follow Virtema [32].

QBfs extend propositional logic by allowing a prenex quantification of proposition symbols. Formally, the set of *qBfs* is built from formulae of propositional logic by the following grammar:

$$\varphi ::= \exists p\, \varphi \mid \forall p\, \varphi \mid \theta,$$

where p is a propositional variable (i.e., a proposition symbol) and θ is formula of propositional logic. The semantics for qBfs is defined via assignments $s\colon \mathrm{PROP} \to \{0,1\}$ in the obvious way. When C is a set of propositional variables, we denote by $\vec{c}$ the canonically ordered tuple of the variables in the set C. When p is a propositional variable and $b \in \{0,1\}$ is a truth value, we denote by $s(p \mapsto b)$ the modified assignment defined as follows:

$$s(p \mapsto b)(q) := \begin{cases} b & \text{if } q = p, \\ s(q) & \text{otherwise.} \end{cases}$$

A formula that does not have any free variables is called *closed*. We denote by QBF the set of exactly all closed quantified Boolean formulae.

Proposition 26 ([30]) *The validity problem of* QBF *is* PSPACE*-complete w.r.t.* $\leq_m^{\log}$.

A *simple qBf* is a closed qBf of the type $\varphi := \forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$, where θ is a propositional formula and the propositional variables p_i, q_j are all distinct. Any tuple $(C_1, \ldots, C_k)$ such that $C_1, \ldots, C_k \subseteq \{p_1, \ldots, p_n\}$ is called a *constraint* for φ. Intuitively, a constraint $C_j = \{p_1, p_3\}$ can be seen as a dependence atom $\mathrm{dep}(p_1, p_3, q_j)$[2]. A constraint $C_j = \{p_1, p_3\}$ can be also interpreted to indicate that the semantics of $\exists g_j$ is defined, if skolemised, via a Skolem function $f_j(p_1, p_3)$.

Definition 27 *A simple qBf* $\forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$ *is* valid under a constraint $(C_1, \ldots, C_k)$, *if there exist functions* $f_1, \ldots, f_k$ *with* $f_i\colon \{0,1\}^{|C_i|} \to \{0,1\}$ *such that for each assignment* $s\colon \{p_1, \ldots, p_n\} \to \{0,1\}$, $s(q_1 \mapsto f_1(s(\vec{c}_1)), \ldots, q_k \mapsto f_k(s(\vec{c}_k))) \models \theta$.

A *dependency quantifier Boolean formula* is a pair $(\varphi, \vec{C})$, where φ is a simple quantified Boolean formula and $\vec{C}$ is a constraint for φ. We say that $(\varphi, \vec{C})$ is *valid*, if φ is valid under the constraint $\vec{C}$. Let DQBF denote the set of all dependency quantifier Boolean formulae.

Proposition 28 ([25, 5.2.2]) *The validity problem of* DQBF *is* NEXP*-complete w.r.t.* $\leq_m^{\log}$.

Definition 29 *Given a simple qBf* $\forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$, *we say it is* non-valid under a constraint $(C_1, \ldots, C_k)$, *if for all functions* $f_1, \ldots, f_k$ *with* $f_i\colon \{0,1\}^{|C_i|} \to \{0,1\}$, *there exists an assignment* $s\colon \{p_1, \ldots, p_n\} \to \{0,1\}$ *such that* $s(q_1 \mapsto f_1(s(\vec{c}_1)), \ldots, q_k \mapsto f_k(s(\vec{c}_k))) \not\models \theta$.

It is straightforward to see that non-validity problem of DQBF is the complement problem of the validity problem of DQBF. Accordingly, the following corollary follows.

Corollary 30 *The non-validity problem of* DQBF *is* coNEXP*-complete w.r.t.* $\leq_m^{\log}$.

5.3 Complexity of the validity problem is coNEXP-hard

In this section we give a reduction from the non-validity problem of DQBF to the validity problem of Minc.

Lemma 31 VAL(Minc) *under lax semantics is* coNEXP*-hard w.r.t.* $\leq_m^{\log}$.

Proof We provide a $\leq_m^{\log}$-reduction from the non-validity problem of DQBF to the validity problem of Minc.

Recall Definition 29. In our reduction we will encode all the possible modified assignments of Definition 29 by points in Kripke models. First we enforce binary (assignment) trees of depth n in our structures. Leafs of the binary tree will correspond to the set of assignments $s\colon \{p_1, \ldots, p_n\} \to \{0,1\}$. The binary trees are forced in the standard way by modal formulae: The formula $\mathsf{branch}_{p_i} := \Diamond p_i \wedge \Diamond \neg p_i$ forces that there are ≥ 2 successor states which disagree on a proposition p_i. The formula $\mathsf{store}_{p_i} := (p_i \to \Box p_i) \wedge (\neg p_i \to \Box \neg p_i)$ is used to propagate chosen values for p_i to successors in the tree. Now define

$$\mathsf{tree}_{p,n} := \mathsf{branch}_{p_1} \wedge \bigwedge_{i=1}^{n-1} \Box^i \Big(\mathsf{branch}_{p_{i+1}} \wedge \bigwedge_{j=1}^{i} \mathsf{store}_{p_j} \Big),$$

[2] See Section 7 for a definition.

where $\Box^i\varphi := \overbrace{\Box\cdots\Box}^{i \text{ many}}\varphi$ is the i-times concatenation of $\Box$. The formula $\mathsf{tree}_{p,n}$ forces a complete binary assignment tree of depth n for proposition symbols $p_1,\ldots,p_n$. Notice that $\mathsf{tree}_{p,n}$ is an ML-formula and consequently flat (see Proposition 18). Let $\ell := \max\{|C_1|,\ldots,|C_k|\}$. Then define

$$\varphi_{\text{struc}} := \mathsf{tree}_{p,n} \wedge \Box^n\big(\mathsf{tree}_{t,\ell}\big) \wedge \Box^{n+\ell}\big((p_\theta \leftrightarrow \theta) \wedge p_\top \wedge \neg p_\bot\big) \wedge \Box^n\Big(\bigwedge_{1\le i\le \ell}\Box^i\big(\bigwedge_{1\le j\le n}\mathsf{store}_{p_j} \wedge \bigwedge_{1\le r\le k}\mathsf{store}_{q_r}\big)\Big).$$

The formula φ_{struc} enforces the full binary assignment tree w.r.t. the p_is, enforces in its leaves trees of depth ℓ for variables t_i, identifies the truth of θ by a proposition p_θ at the depth $n+\ell$ as well as 1 by $t_\top$ and 0 by $t_\bot$, and then stores the values of the p_js and q_rs consistently in their subtrees of relevant depth. The points at depth n are used to encode the modified assignments of Definition 29.

Recall again Definition 29 and consider the simple qBf $\forall p_1\cdots\forall p_n\exists q_1\cdots\exists q_k\theta$ with constraint $(C_1,\ldots,C_k)$. Then consider some particular Kripke model with the structural properties described above. Shift your attention to those points in the enforced tree that are in depth n. Note first that if we restrict our attention to proposition symbols $p_1,\ldots,p_n$ all assignments are present. In fact the number points corresponding to some particular assignment can by anything ≥ 1. Values for the proposition symbols q_j and consequently for the functions f_j arise from the particular model; essentially, since we are considering validity, all possible values will be considered. In fact, in some particular models, the values of q_js are not functionally determined according to the related constraint C_j. We will next define a formula that will deal with those models in which, for some j, the values for q_j do not give rise to a function f_j in Definition 29. These unwanted models have to be "filtered" out by the formula through satisfaction. This violation is expressed via φ_{cons} defined as follows. Below we let $n_j = |C_j|$.

$$\varphi_{\text{cons}} := \bigvee_{\substack{1\le j\le k,\\ C_j=\{p_{i_1},\ldots,p_{i_{n_j}}\}}} (t_1\cdots t_{n_j}t_\bot \subseteq p_{i_1}\cdots p_{i_{n_j}}q_j) \wedge (t_1\cdots t_{n_j}t_\top \subseteq p_{i_1}\cdots p_{i_{n_j}}q_j). \tag{2}$$

Assume that $t_\top$ and $t_\bot$ correspond to the constant values 1 and 0, respectively. Moreover, for the time being, suppose that the values for the proposition symbols t_i have been existentially quantified (we will later show how this is technically done). Now the formula φ_{cons} essentially states that there exists a q_j that does not respect the constraint C_j.

Finally define

$$\varphi_{\text{non-val}} := \varphi^\bot_{\text{struc}} \vee \Big(\varphi_{\text{struc}} \wedge \Box^n\big(\Diamond^\ell(\varphi_{\text{cons}} \vee p_\bot \subseteq p_\theta)\big)\Big). \tag{3}$$

By $\varphi^\bot_{\text{struc}}$, we denote the negation normal form of the ML-formula $\neg\varphi_{\text{struc}}$. An important observation is that since φ_{struc} is an ML-formula, it is flat. Now the formula $\varphi_{\text{non-val}}$ is valid if and only if

$$\mathfrak{M}, T \models \Box^n\big(\Diamond^\ell(\varphi_{\text{cons}} \vee p_\bot \subseteq p_\theta)\big) \tag{4}$$

holds for every team pointed Kripke model $\mathfrak{M}, T$ that satisfies the structural properties forced by φ_{struc}. Let us now return to the formula (2). There, we assumed that the proposition symbols t_i had been quantified and that the symbols $p_\top$ and $p_\bot$ correspond to the logical constants. The latter part we already dealt with in the formula φ_{struc}. Recall that φ_{struc} forces full binary assignment trees for the t_is that start from depth n. The quantification of the t_is is done by selecting the corresponding successors by the diamonds $\Diamond^\ell$ in the formula (3). If $\mathfrak{M}, T$ is such that, for some j, q_j does not respect the constraint C_j, we use $\Diamond^\ell$ to guess a witness for the violation. It is then easy to check that the whole team obtained by evaluating the diamond prefix satisfies the formula φ_{cons}. On the other hand, if $\mathfrak{M}, T$ is such that for each j the value of q_j respects the constraint C_j, then the subformula $p_\bot \subseteq p_\theta$ forces

that there exists a point w in the team obtained from T by evaluating the modalities in (4) such that $\mathfrak{M}, \{w\} \not\models p_\theta$. In our reduction this means that w gives rise to a propositional assignment that falsifies θ as required in Definition 29.

It is now quite straightforward to show that a simple qBf $\forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$ is *non-valid under a constraint* $(C_1, \ldots, C_k)$ iff the Minc-formula $\varphi_{\text{non-val}}$ obtained as described above is valid.

In the following we show the correctness of the constructed reduction. By the observation made in (4) it suffices to show the following claim:

Claim *$\forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$ is non-valid under $(C_1, \ldots, C_k)$ iff $\mathfrak{M}, T \models \Box^n\big(\Diamond^\ell(\varphi_{cons} \vee p_\perp \subseteq p_\theta)\big)$ holds for every team pointed Kripke model $\mathfrak{M}, T$ that satisfies the structural properties forced by φ_{struc}.*

Proof (Proof of Claim) "⇒": Assume that the formula $\varphi := \forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$ is non-valid under the constraint $(C_1, \ldots, C_k)$. As a consequence, for every sequence of functions $f_1, \ldots, f_k$ of appropriate arities there exists an assignment $s\colon \{p_1, \ldots, p_n\} \to \{0, 1\}$ such that

$$s(q_1 \mapsto f_1\big(s(\vec{c}_1)\big), \ldots, q_k \mapsto f_k(s(\vec{c}_k))) \not\models \theta. \tag{5}$$

We will show that

$$\mathfrak{M}, T \models \Box^n\big(\Diamond^\ell(\varphi_{\text{cons}} \vee p_\perp \subseteq p_\theta)\big), \tag{6}$$

for each team pointed Kripke model $\mathfrak{M}, T$ that satisfies the structural properties forced by φ_{struc}.

Let $\mathfrak{M}, T$ be an arbitrary team pointed Kripke model that satisfies the required structural properties. Denote by S the team obtained from T after evaluating the first n $\Box$-symbols in (6). Note that each tuple of values assigned to $\vec{p} := (p_1, \ldots, p_n)$ is realised in S as the tree structure enforces all possible assignments over $\vec{p}$. Due to the forced structural properties, S and of any team obtainable from S by evaluating the k $\Diamond$-symbols in (6) realise exactly the same assignments for $\{p_1, \ldots, p_n, q_1, \ldots q_k\}$. Let S_k denote the set of exactly all points reachable from S by paths of length exactly k. For each point w denote by $w(\vec{q})$ the value of $\vec{q}$ in the world w. Note that for every ℓ-tuple of bits $\vec{b}$ and every point $w \in S$ there exists a point $v \in S_k$ such that $v(p_1, \ldots, p_n, q_1, \ldots q_k) = w(p_1, \ldots, p_n, q_1, \ldots q_k)$ and $v(t_1, \ldots, t_\ell) = \vec{b}$. Moreover, for any fixed $\vec{b}$, the team

$$\{w \in S_k \mid w(t_1, \ldots, t_\ell) = \vec{b}\}$$

is obtainable from S by evaluating the k $\Diamond$-symbols in (6). We have two cases:

1. There exists a constraint C_i, $1 \leq i \leq k$, and points $w, w' \in S$ with $w(\vec{c_i}) = w'(\vec{c_i})$ but $w(q_i) \neq w'(q_i)$. Now let S' be a team obtained from S by evaluating the k $\Diamond$-symbols in (6) such that, for every $w' \in S'$, $w(t_1, \ldots, t_\ell)$ is an expansion of $w(\vec{c_i})$. Now clearly $\mathfrak{M}, S' \models \varphi_{\text{cons}}$ and as a consequence $\mathfrak{M}, S \models \Diamond^\ell(\varphi_{\text{cons}} \vee p_\perp \subseteq p_\theta)$. From this (6) follows.

2. For each C_i, $1 \leq i \leq k$, and every $w, w' \in S$ it holds that if $w(\vec{c_i}) = w'(\vec{c_i})$ then $w(q_i) = w'(q_i)$. Let $f_1, \ldots, f_k$ be some functions that arise from the fact that the constraints $(C_1, \ldots, C_k)$ are satisfied in S. Since, by assumption, φ is non-valid under the constraint $(C_1, \ldots, C_k)$, it follows that there exists an assignment $s\colon \{p_1, \ldots, p_n\} \to \{0, 1\}$ such that (5) holds. Now recall that each tuple of values assigned to $\vec{p} := (p_1, \ldots, p_n)$ is realised in S. Accordingly, in particular, s and $s(q_1 \mapsto f_1\big(s(\vec{c}_1)\big), \ldots, q_k \mapsto f_k(s(\vec{c}_k)))$ are realised in S. For this reason $\mathfrak{M}, S \models \Diamond^\ell(p_\perp \subseteq p_\theta)$, from which (6) follows in a straightforward manner.

"⇐": Assume that $\mathfrak{M}, T \models \Box^n\big(\Diamond^\ell(\varphi_{\text{cons}} \vee p_\perp \subseteq p_\theta)\big)$ holds for every team pointed Kripke model $\mathfrak{M}, T$ that satisfies the structural properties forced by φ_{struc}. We need to show that φ is non-valid under the constraint $(C_1, \ldots, C_k)$. In order to show this, let $f_1, \ldots, f_k$ be arbitrary functions with arities that correspond to the constraint $(C_1, \ldots, C_k)$. Let $\mathfrak{M}, T$ be a team pointed Kripke model and S a team of $\mathfrak{M}$ such that

a) $\mathfrak{M}, T$ satisfies the structural properties forced by φ_{struc},

b) S is obtained from T by evaluating the n $\Box$-symbols,

c) $f_i\big(w(\vec{c}_i)\big) = w(q_i)$, for each $w \in S$ and $1 \leq i \leq k$.

It is easy to check that such a model and teams always exist. From the assumption we then obtain that

$$\mathfrak{M}, S \models \Diamond^{\ell}(\varphi_{\text{cons}} \vee p_{\perp} \subseteq p_{\theta}). \tag{7}$$

But since the values of q_is, by construction, do not violate the constraint $(C_1, \ldots, C_k)$, we obtain, with the help of the structural properties, that for (7) to hold is must be the case that $\mathfrak{M}, S_k \models p_{\perp} \subseteq p_{\theta}$, where S_k is some team obtained from S by evaluating the k $\Diamond$-symbols in (7). But this means that there exists an assignment $s\colon \{p_1, \ldots, p_n\} \to \{0, 1\}$ such that

$$s(q_1 \mapsto f_1\big(s(\vec{c}_1)\big), \ldots, q_k \mapsto f_k(s(\vec{c}_k))) \not\models \theta. \tag{8}$$

Consequently, the claim holds. ∎

In order to compute $\varphi_{\text{non-val}}$ two binary counters bounded above by $n + k + \ell$ need to be maintained. Note that $\log(n + k + \ell)$ is logarithmic with respect to the input length. That being the case, the reduction is computable in logspace and the lemma applies. ∎

The construction in the previous proof works also for strict semantics. In the proof of the claim a small adjustment is needed to facilitate the strict semantics of diamond. As a result we obtain the following.

Corollary 32 VAL(Minc) *under strict semantics is* coNEXP*-hard w.r.t.* $\leq_m^{\log}$.

While the exact complexities of the problems VAL(Minc) and VAL(EMinc) remain open, it is easy to establish that the complexities coincide.

Theorem 33 *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*.*

Proof Let φ be a formula of EMinc and k the modal depth of φ. Let $\varphi_1, \ldots, \varphi_n$ be exactly those subformulae of φ that occur as a parameter of some inclusion atom in φ and let $p_1, \ldots, p_n$ be distinct fresh proposition symbols. Define

$$\varphi_{\text{subst}} := \big(\bigwedge_{0 \leq i \leq k} \Box^i \bigwedge_{1 \leq j \leq n} (p_j \leftrightarrow \varphi_j) \big),$$
$$\varphi^* := \varphi^{\perp}_{\text{subst}} \vee (\varphi_{\text{subst}} \wedge \varphi^+),$$

where $\varphi^{\perp}_{\text{subst}}$ denotes the negation normal form of $\neg\varphi_{\text{subst}}$ and φ^+ is the formula obtained from φ by simultaneously substituting each φ_i by p_i. It is easy to check that φ is valid if and only if the Minc formula φ^* is. Clearly φ^* is computable from φ in polynomial time. ∎

6 Conclusion

In this paper we investigated the computational complexity of model checking and validity for propositional and modal inclusion logic in order to complete the complexity landscape of these problems in the mentioned logics. In particular we emphasise on the subtle influence of which semantics is considered: strict or lax. The model checking problem for these logics under strict semantics is NP-complete and under lax semantics P-complete. The validity problem is shown to be coNP-complete for the propositional strict semantics case. For the modal case we achieve a coNEXP lower bound under lax as well as strict semantics. The upper bound is left open for further research. It is however easy to establish that, if closed under polynomial reductions, the complexities of VAL(Minc) and VAL(EMinc), and VAL(Minc_s) and VAL(EMinc_s) coincide, respectively, see Proposition 33.

	PL Satisfiability Problem		ML Satisfiability Problem	
Operator	strict	lax	strict	lax
$\emptyset$	NP [4, 22]		PSPACE [21]	
dep(·)	NP [23]		NEXP [29]	
$\subseteq$	EXP [17]	EXP [16]	EXP [17]	EXP [16]
$\perp$	NP*	NP [13]	NEXP*	NEXP [20]
$\sim$	AEXP[*poly*]*	AEXP[*poly*] [13, 14]	?	?
all	AEXP[*poly*]*	AEXP[*poly*] [13, 14]	?	?

Table 3: Complexity of Satisfiability, where all $= \{\mathrm{dep}(\cdot), \subseteq, \perp, \sim\}$.
$\star$: Proof for lax semantics works also for strict semantics.
?: No nontrivial result is known.

	PL Model Checking		ML Model Checking	
Operator	strict	lax	strict	lax
$\emptyset$	NC^1 [2]		P [3, 28]	
dep(·)	NP [6]		NP [6]	
$\subseteq$	NP [Thm. 15]	P [Thm. 11]	NP [Thm. 25]	P [Thm. 24]
$\perp$	NP*	NP [13]	NP*	NP [20]
$\sim$	PSPACE*	PSPACE [13, 24]	PSPACE*	PSPACE [24]
all	PSPACE*	PSPACE [13, 24]	PSPACE*	PSPACE [13]

Table 4: Complexity of Model Checking, where all $= \{\mathrm{dep}(\cdot), \subseteq, \perp, \sim\}$.
$\star$: Proof for lax semantics works also for strict semantics.

7 Related work and further research

Tables 3–5 give an overview of the current state of research for satisfiability, model checking and validity in the propositional and modal team semantics setting for both strict and lax variants. In the tables AEXP[*poly*] refers to alternating exponential time with polynomially many alternations. We also identify the unclassified cases open for further research. As these tables also mention atoms which have not been considered elsewhere in this paper, we will introduce them shortly:

Let $\vec{p}$, $\vec{q}$, and $\vec{r}$ be tuples of proposition symbols and q a proposition symbol. Then $\mathrm{dep}(\vec{p}, r)$ is a *propositional dependence atom* and $\vec{q} \perp_{\vec{p}} \vec{r}$ is a *conditional independence atom* with the following semantics:

$$X \models \mathrm{dep}(\vec{p}, q) \;\Leftrightarrow\; \forall s,t \in X : s(\vec{p}) = t(\vec{p}) \text{ implies } s(q) = t(q).$$
$$X \models \vec{q} \perp_{\vec{p}} \vec{r} \;\Leftrightarrow\; \forall s,t \in X : \text{ if } s(\vec{p}) = t(\vec{p}), \text{then } \exists u \in X : u(\vec{p}\vec{q}) = s(\vec{p}\vec{q}) \text{ and } u(\vec{r}) = t(\vec{r}).$$

Intuitively, $\vec{q} \perp_{\vec{p}} \vec{r}$ states that for any fixed value for $\vec{p}$, $\vec{q}$ and $\vec{r}$ are informationally independent. We also consider the contradictory negation $\sim$ in our setting:

$$X \models \sim\varphi \text{ iff } X \not\models \varphi.$$

Semantics for these atoms in the modal setting is defined analogously. When $\mathcal{C}$ is a set of atoms, we denote by PL($\mathcal{C}$) and ML($\mathcal{C}$) the extensions of PL and ML, in the team semantics setting, by the atoms in $\mathcal{C}$, respectively.

A fruitful direction for future research is to study automatic reasoning in the team semantics setting.

References

[1] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Cambridge Univ. Press, 2001.

	PL Validity Problem		ML Validity Problem	
Operator	strict	lax	strict	lax
$\emptyset$	coNP [4, 22]		PSPACE [21]	
dep($\cdot$)	NEXP [32]		$\in$ NEXP$^{\mathsf{NP}}$ [32]	
$\subseteq$	coNP [Thm. 7]	coNP [13]	coNEXP-h [Cor. 32]	coNEXP-h [Lem. 31]
$\perp$	$\in$ coNEXP$^{\mathsf{NP}\star}$	$\in$ coNEXP$^{\mathsf{NP}}$ [13]	?	?
$\sim$	AEXP[*poly*]*	AEXP[*poly*] [13, 14]	?	?
all	AEXP[*poly*]*	AEXP[*poly*] [13, 14]	?	?

Table 5: Complexity of Validity, where all $= \{\text{dep}(\cdot), \subseteq, \perp, \sim\}$. Complexity classes refer to completeness results, "-h." denotes hardness and "$\in$" denotes containment.
$\star$: Proof for lax semantics works also for strict semantics.
?: No nontrivial result is known.

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

[3] 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.

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

[5] 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. 2016.

[6] J. Ebbing and P. Lohmann. Complexity of model checking for modal dependence logic. In *38th Proc. SOFSEM*, pages 226–237, 2012.

[7] P. Galliani. Inclusion and exclusion dependencies in team semantics - on some logics of imperfect information. *Ann. Pure Appl. Logic*, 163(1):68–84, 2012.

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

[9] P. Galliani and L. Hella. Inclusion logic and fixed point logic. In *Proc. 22nd CSL*, LIPIcs, pages 281–295, 2013.

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

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

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

[13] M. Hannula, J. Kontinen, J. Virtema, and H. Vollmer. Complexity of propositional independence and inclusion logic. In *Proc. 40th MFCS*, pages 269–280, 2015.

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

[15] Miika Hannula and Juha Kontinen. Hierarchies in independence and inclusion logic with strict semantics. *J. Log. Comput.*, 25(3):879–897, 2015.

[16] 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.

[17] L. Hella, A. Kuusisto, A. Meier, and H. Vollmer. Satisfiability of modal inclusion logic: Lax and strict semantics. 2017. Corrected version of [16], to appear soon on arXiv:1504.06409.

[18] L. Hella and J. Stumpf. The expressive power of modal logic with inclusion atoms. In *Proc. 6th GandALF*, pages 129–143, 2015.

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

[20] J. Kontinen, J.-S. Müller, H. Schnoor, and H. Vollmer. Modal independence logic. *Journal of Logic and Computation*, 2016.

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

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

[23] P. Lohmann and H. Vollmer. Complexity results for modal dependence logic. *Studia Logica*, 101(2):343–366, 2013.

[24] J.-S. Müller. *Satisfiability and Model Checking in Team Based Logics*. PhD thesis, Leibniz University of Hannover, 2014.

[25] 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.

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

[27] 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.

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

[29] M. Sevenster. Model-theoretic and computational properties of modal dependence logic. *Journal of Logic and Computation*, 19(6):1157–1173, 2009.

[30] L. J. Stockmeyer. and A. R. Meyer. Word problems requiring exponential time(preliminary report). In *Proc. 5th STOC*, pages 1–9, New York, NY, USA, 1973. ACM.

[31] J. Väänänen. *Dependence Logic*. Cambridge University Press, 2007.

[32] J. Virtema. Complexity of validity for propositional dependence logics. *Information and Computation*, 2016. Online first.

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