Counting and Enumerating in First-Order Team Logics

Von der Fakultät für Elektrotechnik und Informatik

der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des akademischen Grades

Doktor rerum naturalium

(abgekürzt: Dr. rer. nat.)

genehmigte Dissertation

von Herrn

M. Sc. Fabian Müller

geboren am 09.10.1988

in Neunkirchen (Saar)

2024

Referent:	Prof. Heribert Vollmer, Leibniz Universität Hannover
Korreferentin:	Prof. Nadia Creignou, University of Aix-Marseille
Tag der Promotion:	05.03.2024

Abstract

Descriptive complexity theory is the study of the expressibility of computational problems in certain logics. Most of the results in this field use (fragments or extensions of) first-order logic or second-order logic to describe decision complexity classes. For example the complexity class NP can be characterized as the set of problems that are describable by a dependence logic formula, in short $\mathsf{NP} = \mathsf{FO}(=(\dots))$. Dependence logic is a certain team logic, where a team logic is an extension of first-order logic by some new atoms, with new semantics, called team semantics. Compared to decision complexity where one is interested in the existence of a solution to an input instance, in counting complexity one is interested in the number of solutions and in enumeration complexity one wants to compute all solutions. Counting complexity has been less studied in terms of descriptive complexity than decision complexity, whereas there are no results for enumeration complexity in this field. The latter is because the concept of hardness in the enumeration setting was first introduced rather recently.

In this thesis, we characterize counting and enumeration complexity classes with team logics and compare the results to the corresponding results for decision complexity classes. To study the framework of hard enumeration a bit more, we investigate further team logic based enumeration problems.

In the counting setting we characterize the classes $\#\mathsf{P}$ and $\#\cdot\mathsf{NP}$ as $\#\mathsf{P} = \#\mathsf{FOT}$ and $\#\cdot\mathsf{NP} = \#\mathsf{FO}(\perp)$. Furthermore, we establish two team logic based classes $\#\mathsf{FO}(\subseteq)$ and $\#\mathsf{FO}(=(\dots))$ which seem to have no direct counterpart in classical counting complexity, but contain problems that are complete under Turing reductions for $\#\mathsf{P}$ and $\#\cdot\mathsf{NP}$, respectively. To show the latter we identify a new $\#\cdot\mathsf{NP}$-complete problem with respect to Turing reductions.

We show that in the enumeration setting the classes behave very similarly to the corresponding classes in the decision setting. We translate the results $\mathsf{P} = \mathsf{FO}(\subseteq)$ and $\mathsf{NP} = \mathsf{FO}(=(\dots))$ to the enumeration setting which results in $\mathsf{DelP} = \mathsf{DelFO}(\subseteq)$ and $\mathsf{DelNP} = \mathsf{DelFO}(=(\dots))$. Furthermore, we identify several DelP and DelNP-complete problems which yield additional characterisations of DelP and DelNP. For one of the investigated problems we were only able to show $\mathsf{Del}^{+}\mathsf{NP}$ membership (and DelNP-hardness), a precise classification remains open.

Keywords: counting complexity, enumeration complexity, descriptive complexity, team logic

Zusammenfassung

In der deskriptiven Komplexitätstheorie wird die Ausdrückbarkeit von Berechnungsproblemen durch bestimmte Logiken erforscht. Meist wird hierbei die Prädikatenlogik der ersten (FO) oder zweiten Stufe (SO) oder Fragmente oder Erweiterungen davon verwendet. Zum Beispiel lässt sich die Klasse NP charakterisieren als die Menge der Probleme die durch jeweils eine Formel der Dependence-Logik beschreibbar sind, oder kurzgesagt $\mathsf{NP} = \mathsf{FO}(=(\dots))$. Dependence-Logik ist eine bestimmte Team-Logik, wobei eine Team-Logik eine Erweiterung von FO um neue atome mit einer neuen Semantik, der Team-Semantik. Im Vergleich zur Entscheidungskomplexität, bei der man an der Existenz einer Lösung zu einer gegebenen Eingabe interessiert ist, sucht man in der Zählkomplexität nach der Anzahl der Lösungen und in der Aufzählkomplexität die Lösungen selbst. Das Fachgebiet der Zählkomplexität wurde weniger untersucht in Hinblick auf die deskriptive Komplexität als die Entscheidungskomplexität, für die Aufzählkomplexität gibt es bislang keine Ergebnisse dahingehend. Letzteres resultiert daraus, dass das Konzept von Schwere in der Aufzählkomplexität noch recht jung ist.

In dieser Arbeit charakterisieren wir Zähl- und Auszählkomplexitätsklassen mit Team-Logiken und vergleichen die Ergebnisse mit ihren Pendants in der Entscheidungswelt. Um das Modell der schweren Aufzählkomplexität weiter zu untersuchen, betrachten wir weitere Aufzählprobleme aus dem Bereich der Team-Logik.

Im Fachgebiet der Zählkomplexität charakterisieren wir die Klassen $\#\mathsf{P}$ und $\#{\cdot}\mathsf{NP}$ als $\#\mathsf{P} = \#\mathsf{FOT}$ und $\#{\cdot}\mathsf{NP} = \#\mathsf{FO}(\perp)$. Weiterhin identifizieren wir zwei neue Team-Logik basierte Klassen, $\#\mathsf{FO}(\subseteq)$ und $\#\mathsf{FO}(=(\dots))$, für die es kein Pendant in der klassischen Zählkomplexität zu geben scheint. Jedoch enthalten diese Klassen Probleme, die unter Turing-Reduktionen für die Klassen $\#\mathsf{P}$ beziehungsweise $\#{\cdot}\mathsf{NP}$ vollständig sind. Um letzteres zu zeigen, identifizieren wir ein Problem, welches $\#{\cdot}\mathsf{NP}$-vollständig unter Turing-Reduktionen ist.

Wir zeigen, dass sich die Klassen der Aufzähl- und Entscheidungswelt ähnlich verhalten. Wir übertragen die Ergebnisse $\mathsf{P} = \mathsf{FO}(\subseteq)$ und $\mathsf{NP} = \mathsf{FO}(=(\dots))$ in die Aufzählwelt, was zu $\mathsf{DelP} = \mathsf{DelFO}(\subseteq)$, beziehungsweise $\mathsf{DelNP} = \mathsf{DelFO}(=(\dots))$ führt. Weiterhin identifizieren wir einige DelP- und DelNP-vollständige Probleme, welche zu weiteren Charakterisierungen dieser Klassen führen. Für eines der untersuchten Probleme konnten wir nur $\mathsf{Del}^{+}\mathsf{NP}$-Mitgliedschaft (und DelNP-Schwere) zeigen, eine genaue Einordnung ist noch zu finden.

Schlagworte: Zählkomplexität, Deskriptive Komplexität, Aufzählkomplexität, Team Logik

Contents

1 Introduction

The question "$\mathsf{P} = \mathsf{NP}$?" was first formulated by Cook and Levin in 1971 (see for example [Coo71]). Today this is the most important question in complexity theory and maybe even in whole computer science. There is a broad consensus that the answer must be "no", that is, $\mathsf{P} \neq \mathsf{NP}$, but so far nobody was able to prove it. In a sense, any research in the field of complexity theory does in part try to help answering this question, even if it has different focus.

In complexity theory one studies computational problems, most notably decision problems. In a decision problem we are given an input and like to answer the question if there is a solution to it. Such problems are classified in terms of their complexity by showing that they are contained in certain (decision) complexity classes. There are several ways to define complexity classes, the most prominent being by a computational model. Examples for computational models used in this area are Turing machines, circuits, random access machines or neuronal networks. Another way to define complexity classes is to use logical formulas to describe their problems. The beauty of this is that no resource bounds are needed, there is no computation, no algorithm, the complexity of a problem is determined solely by writing it down. Classically one uses fragments or extensions of first- or second-order logic to define such complexity classes. The logics in the focus of this thesis are team logics, which are an extension of classical first-order logic.

As mentioned besides decision problems there are also other kinds of problems that are studied in complexity theory. For example, there are search problems, where one wants to find a solution instead of just proving its existence. In counting complexity one is interested in the number of solutions, whereas in enumeration complexity one wants to enumerate all solutions to a given input. These settings are all tied to each other in one way or another. This offers the possibility to translate results between settings and compare different settings to get a better overall picture.

In this thesis we study the complexity of counting and enumeration problems defined by certain team logics.

1.1 Team Semantics

In classical first-order logic it is in some way possible to express certain dependencies. For example in the formula $f(x) = y$ the value of y depends on the value of x and this dependency is described by the function f. Having multiple quantifiers also creates dependency, in the formula $\forall x \exists y\ \psi(x, y)$, the value of y depends on the value of x. In 1959 Henkin took this kind of dependency one step further by extending first-order logic by his new *branching quantifiers*, which are also often called *Henkin quantifiers* [Hen61]. Henkin quantifiers are expressions of the following form:

$$\begin{pmatrix} \forall x_1 & \exists y_1 \\ \forall x_2 & \exists y_2 \end{pmatrix} \psi(x_1, x_2, y_1, y_2).$$

This specific formula has the meaning, "for all x_1 there is a y_1 depending only on x_1 and for all x_2 there is a y_2 depending only on x_2 such that $\psi(x_1, x_2, y_1, y_2)$ is true". Any such formula is equivalent to a second-order formula with free functional variables. For our example this formula would be $\forall x_1 \forall x_2\ \psi(x_1, x_2, f_1(x_1), f_2(x_2))$, where f_1, f_2 are free functional variables. In 1989 Hintakka and Sandu introduced two new quantifiers, the *backslash quantifier* and the *slash quantifier* [HS89]. Consider the following formula

$$\exists x \forall y \exists z \backslash \{x\}\ \psi_1'(x, y, z).$$

The backslash quantifier $\exists z \backslash \{x\}$ in this formula expresses that there is a value for z that only depends on the value of x such that ψ_1' holds. If we would replace the backslash by a slash quantifier the meaning of the formula would change to "there is a value for z that is independent of x such that ψ_1' holds". When extending first-order logic by the backslash or slash quantifier we get a new logic called *dependence friendly logic* and *independence friendly logic*, respectively.

In 2007 Väänänen extended first-order logic by a new atom, the *dependence atom* $=\!(\dots)$ and redefined the semantics to obtain a new logic $\mathsf{FO}(=\!(\dots))$, which is fittingly called *dependence logic* [Vää07]. Intuitively, the atom $=\!(x_1, \dots, x_n, y)$ expresses that the value of y depends on the values of $x_1, \dots, x_n$ and nothing else, without actually stating this dependency like in the previous example $f(x) = y$. The new semantics, called *team semantics*, defines satisfiability not for single assignments but for sets of assignments, called *teams*. The dependence atom thereby expresses dependencies that hold for all assignments in such a team. For example $=\!(x, y)$ is satisfied by a team, if any pair of assignments in that team have the following property: If the assignments agree on x they must agree on y as well.

Several new atoms have been defined for first-order logic with team semantics since then. Most notably the *independence atom* $x \perp_z y$ introduced by Grädel and Väänänen [GV13] and the *inclusion atom* $x \subseteq y$ introduced by Galliani [Gal12].

The atom $x \perp_z y$ expresses: the value of x is independent of the value of y if the values of z are constant. The inclusion atom $x \subseteq y$ expresses that the values that appear for x must only be values that also appear for y. These atoms define new logics, *independence logic* $\mathsf{FO}(\perp)$ and *inclusion logic* $\mathsf{FO}(\subseteq)$, respectively.

1.2 Descriptive Complexity

The field of descriptive complexity was founded by Ronald Fagin in 1974 with his result $\mathsf{NP} = \Sigma_1^1$, that is, NP is exactly the class of problems that can described by an existential second-order logic formula [Fag74]. In 1982, both Immerman and Vardi found (independently of each other) a characterisation of P in terms of first-order logic. They showed that $\mathsf{P} = \mathsf{LFP}$, that is, the problems in P are exactly those that can be described by a first-order formula with least fixed point quantifiers [Imm82, Imm86, Var82]. Several results followed in the field, the classes L, NL, PSPACE and EXP where characterized with similar logics by Immerman [Imm87]. The book "Descriptive Complexity" [Imm99] by Immerman gives a good overview over the field, it contains results that range from characterisations for $\mathsf{LOGTIME}$ to the arithmetic hierarchy.

Dependence logic, independence logic and inclusion logic were all studied in terms of descriptive complexity and expressive power. It was shown that over sentences dependence logic as well as independence logic are expressive equivalent to existential second-order logic [KV09, Gal12]. As a consequence, by Fagin's Theorem, both logics capture NP. Inclusion logic on the other hand has the same expressive power as LFP [GH13], hence it captures P. We will revisit these results in more detail in Section 2.6.

1.3 Counting Complexity

In 1979, Leslie Valiant published a paper in which he classified the complexity of the problem of computing the permanent of a binary matrix [Val79a]. He showed that this problem was complete for the new class $\#\mathsf{P}$ he defined in the same paper. He defined $\#\mathsf{P}$ as the set of counting problems that can be computed by counting the number of accepting paths of a nondeterministic polynomial-time Turing machine. A second paper of his contains a list of several additional $\#\mathsf{P}$-complete problems from the fields of graph theory and propositional logic. For these results he used Turing reductions, but it was later shown that several of these reductions can be made parsimonious, which is a stronger reducibility notion. In a rather confusing definition, he introduced further counting classes, probably the most interesting one of them is $\#\mathsf{NP} := \#\mathsf{P}^{\mathsf{NP}}$. The complexity class $\#\mathsf{L}$

was introduced by Alvarez and Jenner [ÀJ93], which is defined analogously to #P but on logarithmic space Turing machines. It was shown that this class is closely related to the problem of computing the determinant of a binary matrix [AO96, Vin91, Val92].

In 1991, Toda proposed a new operator "#·" (or "NUM") that defines a counting class #·C for arbitrary complexity classes C [Tod91]. By this definition #·C contains the problems that count the number of solutions y to a given input x such that $(x, y) \in \mathrm{D}$, where D is a decision problem in C. This way they also defined #·NP which is a more fitting version of #NP. Around the turn of the millennium two additional counting classes where introduced that are both subsets of #P, namely #PE [Pag01] and TotP [KPSZ01]. The "E" in #PE stands for "easy" which is fitting, since #PE was defined as the class of problems in #P that have an easy decision version (in this context "easy" means computable in polynomial time). The class TotP was defined similarly to #P but instead of counting only accepting paths, problems in TotP count any paths of nondeterministic polynomial-time Turing machines. It was shown that TotP is the closure under parsimonious reductions of self-reducible problems in #PE [PZ06] and thereby $\mathsf{TotP} \subseteq \#\mathsf{PE}$.

The first descriptive complexity result in the counting complexity setting was provided by Saluja et al. by showing that problems in #P can be described by first-order formulas with free relational variables and free individual variables, in short, $\#\mathsf{P} = \#\mathsf{FO}^{\mathrm{rel}}$ [SST95]. This result can be seen as a translation of Fagin's Theorem, since #P is often seen the equivalent of NP in the counting setting and $\mathsf{FO}^{\mathrm{rel}} = \Sigma_1^1$. More recent studies of Haak et al. showed that this result can also be achieved by counting functions instead of relations, which results in $\#\mathsf{P} = \#\mathsf{FO}^{\mathrm{func}}$ [DHKV21]. Arenas et al. introduced a new logic called quantitative second order logics and showed that several counting classes can be described by fragments of this logic. This way they characterised the classes $\#\mathsf{L}, \mathsf{FP}, \#\mathsf{P}$ and FPSPACE [AMR20]. In this thesis we characterize the classes #P and #·NP and introduce two new counting classes using team logics. These results are based on the paper "Counting of teams in first-order team logics" of Haak et al.[HKM+19].

1.4 Enumeration Complexity

For a long time this field has been on the more practical side. One was more interested in algorithms that are considered tractable than in hardness and completeness results. In this field tractable means that the algorithm has polynomial delay, that is, the time between any two outputs of the algorithm is bounded by a polynomial. However such algorithms can be forced to have an exponential overall running time, since the number of solutions might be exponential. Up until recently lower bounds in this field were translated to the decision setting. Results were of the form “When enumeration problem E is in class C_1 then decision problem D is in class C_2”, which, together with a result that implies $D \notin \mathsf{C}_2$, can then be seen as a lower bound.

In 2019, Creignou et al. introduced the classes DelC for arbitrary (decision) complexity classes C and proposed reducibility notions to be able to conduct hardness results [CKP+19]. By their definition DelC is the class of enumeration problems for which there exists an algorithm that has access to an oracle from C and outputs all solutions with polynomial delay. This definition yields a counterpart of the polynomial hierarchy in the enumeration setting. Hardness results in this framework are still closely tied to the decision setting, as hardness results from the decision setting translate to the enumeration setting. However, the converse does not hold. Creignou et al. were able to identify enumeration problems that are hard for DelNP, whereas the corresponding decision problem is still in P.

So far the only descriptive complexity results in the enumeration setting published are the ones by Haak et al. [HMMV22], which is one of the papers this thesis is based on. The main results are the characterisations of DelP and DelNP in terms of team logic.

1.5 Overview

The content of this thesis is based on the papers “Counting of teams in first-order team logics” [HKM+19] and “Enumerating teams in first-order team logics” [HMMV22]. Compared to these papers there is one major new result, Theorem 4.13, which states that a certain decision problem defined over an inclusion logic formula is NP-hard, which implies DelNP-hardness for a related enumeration problem. Furthermore, there are some minor enhancements to results from these papers. For example, we dedicatedly show $\mathsf{DelP} = \mathsf{DelFO}(\subseteq)$. Overall these new results and the presentation of this thesis leads to a better overview about the relationships between the considered complexity classes and enhances the comparability between the decision, counting and enumeration settings.

In Chapter 2 we recall the fundamentals of propositional logic, first and second-order logic, team logic, counting and enumeration complexity. Furthermore we introduce a new framework that allows us to easily define corresponding decision, counting and enumeration problems in Section 2.5. This framework is very similar to the one used by Creignou et al. [CKP+19], but uses functions instead of relations, which feels slightly more natural in the context of counting and enumeration problems. In Section 2.6 we give a complete picture of the descriptive complexity results for team logics in the decision setting.

In Chapter 3 we study the descriptive complexity of counting problems in terms of team logic. We start by showing that the descriptive complexity classes

$\#\mathsf{FO}(=(\dots))$, $\#\mathsf{FO}(\perp)$ and $\#\mathsf{FO}(\subseteq)$ are closed under first-order reductions, which is the reducibility notion we consider in this context. Afterwards we show that $\#\mathsf{FO}(\mathrm{T}) \subseteq \#{\cdot}\mathsf{NP}$, for sets of atoms T with a certain property, which implies that this inclusion holds for the three mentioned classes. The first three sections are each dedicated to one of these logics. Section 3.1 contains the results for independence logic. Here we show $\#{\cdot}\mathsf{NP} \subseteq \#\mathsf{FO}(\perp)$ in two steps. In the first step, we show $\#{\cdot}\mathsf{NP} \subseteq \#\Sigma^1_1$ and in the second $\#\Sigma^1_1 = \#\mathsf{FO}(\perp)$. Together with the result from before, $\#\mathsf{FO}(\perp) \subseteq \#{\cdot}\mathsf{NP}$, we are able to conclude $\#\mathsf{FO}(\perp) = \#{\cdot}\mathsf{NP}$. In Section 3.2, we show that $\#{\cdot}\mathsf{FO}(=(\dots))$ contains a problem that is $\#{\cdot}\mathsf{NP}$-complete under Turing reductions. Furthermore, we show that the same problem is also complete for $\#{\cdot}\mathsf{FO}(=(\dots))$ under first-order reductions. In Section 3.3, we study the class $\#\mathsf{FO}(\subseteq)$. We show that $\#\mathsf{FO}(\subseteq) \subseteq \#\mathsf{P}$ and this inclusion is strict unless $\mathsf{P} = \mathsf{NP}$. However, we identify a problem that is included in $\#\mathsf{FO}(\subseteq)$ and also complete for $\#\mathsf{P}$ under Turing reductions. Furthermore, we show that $\#\mathsf{FO}(\subseteq) \subseteq \mathsf{TotP}$. We conjecture that this inclusion is strict as well, but have no proof. To give our conjecture more substance we present a problem that is hard for (and probably not included in) $\#\mathsf{FO}(\subseteq)$ and included in TotP. In Section 3.4, we identify complete problems for $\#{\cdot}\mathsf{NP}$, most notably the problem we showed is included in $\#\mathsf{FO}(=(\dots))$. In Section 3.5, we summarize our results from the Chapter in a class diagram.

In Chapter 4, we study the descriptive complexity of enumeration problems in terms of team logic. First, we show that $\mathsf{DelFO}(\perp) = \mathsf{DelFO}(=(\dots)) = \mathsf{DelNP}$ in three steps. In similar fashion, we capture DelP with inclusion logic, that is, we show $\mathsf{DelFO}(\subseteq) = \mathsf{DelP}$. We summarise the results up to this point in a class diagram, which looks almost identical to the one from the decision setting. In Section 4.1, we study the complexity of enumerating optimal (minimal, maximal etc.) solutions for dependence and independence logic formulas. We show that all except one of these problems are DelNP-complete. For the one exception, we show DelNP-hardness and the inclusion in $\mathsf{Del}^+\mathsf{NP}$, which is a more powerful version of DelNP. In Section 4.2, we study the complexity of optima problems for inclusion logic. Here, we show that enumerating maxima is in DelP, whereas enumerating minima is DelNP-complete. We give a summary of the Chapter in Section 4.3.

2 Preliminaries

In the first few section, we recall the main ideas of proposition logic, first-order logic, second-order logic, graph theory and complexity theory. For a thorough introduction to propositional, first-order and second-order logic we recommend "A Mathematical Introduction to Logic" [End72]. For the topics of graph theory and complexity theory we recommend "Computational Complexity" [Pap94].

As we consider several logics in this thesis (often in the same proof), it might sometimes be hard to determine from which logic a certain formula stems. To minimize confusion we always represent propositional formulas with χ, team logic formulas with φ, and first- and second-order logic formulas with ψ.

2.1 Propositional Logic

The *formulas of propositional logic* are constructed from variables and the constants 0 and 1, using the connectives $\wedge, \vee$ and $\neg$. We also refer to those formulas as *Boolean formulas.* For readability we denote an implication by $\chi_1 \rightarrow \chi_2$ and an equivalence by $\chi_1 \leftrightarrow \chi_2$, for propositional formulas χ_1, χ_2. That is, we use $\chi_1 \rightarrow \chi_2$ as a shorthand for the formula $\neg\chi_1 \vee \chi_2$ and $\chi_1 \leftrightarrow \chi_2$ for $(\chi_1 \rightarrow \chi_2) \wedge (\chi_2 \rightarrow \chi_1)$. Given a formula χ over variables $x_1, \dots, x_n$, we denote with $\mathsf{vars}(\chi) = \{x_1, \dots, x_n\}$ the set of variables of χ. When we use the term *literal* we mean either a (positive) variable or a negated variable. A formula χ is in *conjunctive normal form*, if it has the form $\chi = \bigwedge_i C_i$, where each C_i is a *clause*, that is, $C_i = (l_{i,1} \vee l_{i,2} \vee \dots \vee l_{i,|C_i|})$ and $l_{i,j}$ are literals. A *Horn formula* (respectively *DualHorn formula*) is a formula in conjunctive normal form, that contains at most one positive (respectively negative) literal in each clause. We define *quantified Boolean formulas* as the extension of propositional formulas by existential and universal quantifiers. In this thesis we only consider quantified Boolean formulas in prenex normal form. In a quantified Boolean formula (in prenex normal form), we say a variable x is bound if it lies in the scope of a quantifier $\exists x$ or $\forall x$, otherwise we say x is *free.* We denote with $\mathsf{free}(\chi)$ the set of free variables for a given formula χ. We also write $\chi(x_1, \dots, x_n)$ to denote that $x_1, \dots, x_n$ are free variables in χ. Note that in propositional formulas all variables are free.

Now we define different classes of propositional formulas. First of all we have the class BF, which stands for Boolean formula and consists of all propositional formulas. The class CNF contains all propositional formulas that are in conjunctive normal form and kCNF contains the formulas that are in CNF and have their clause size bounded by k. By HORN and DH we denote the class of Horn formulas and DualHorn formulas, respectively. Given a class P of propositional formulas, we define the new class Σ_1P which consists of quantified Boolean formulas in prenex normal form with only existential quantifiers where the quantifier-free part is in the class P. For a class P of quantified boolean formulas, we define the classes P^+ and P^- of formulas from P whose free variables only occur positively or negatively, respectively. Note the subtle difference between the classes $(\Sigma_1\mathrm{P})^-$ and $\Sigma_1(\mathrm{P}^-)$: Formulas in $(\Sigma_1\mathrm{P})^-$ have only negative free variables but may have positive and negative bound variables, whereas formulas in $\Sigma_1(\mathrm{P}^-)$ have only negative variables (bound and free). In this thesis we are only interested in classes of the type $(\Sigma_1\mathrm{P})^-$, therefore we omit the parentheses and write $\Sigma_1\mathrm{P}^-$ instead. For example, $\Sigma_1\mathrm{CNF}^-$ consists of all quantified Boolean formulas in prenex normal form with only existential quantifiers, where the quantifier-free part is in CNF and the free variables occur only negatively.

A *(propositional) assignment* β is a set of variables which we interpret as the set of variables with the truth value 1. By this definition the empty set is an assignment as well, which we will call *empty assignment*. When we consider an assignment β in the context of a formula χ, we assume that β is a suitable assignment for χ, that is, $\beta \subseteq \mathsf{free}(\chi)$. Given a formula χ we call the assignment $\beta = \mathsf{free}(\chi)$ *full assignment*. We denote by Θ the set of all propositional assignments and by $\Theta(\chi)$ the set of all assignments with $\beta \subseteq \mathsf{free}(\chi)$.

Remark 2.1. *Let χ be a* Horn *formula. Then there is a* DualHorn *formula $\widetilde{\chi}$ such that*

$$\beta \models \chi \iff \mathsf{vars}(\chi) \setminus \beta \models \widetilde{\chi}, \text{ for all } \beta \subseteq \mathsf{vars}(\chi)$$

and vice versa. The same holds for CNF^+ *and* CNF^- *formulas. The formula $\widetilde{\chi}$ is defined as follows. Let $\chi = \bigwedge_i \bigvee_j \ell_{i,j}$ be a formula in conjunctive normal form. We define the formula $\widetilde{\chi} \coloneqq \bigwedge_i \bigvee_j \widetilde{\ell_{i,j}}$, where*

$$\widetilde{\ell_{i,j}} \coloneqq \begin{cases} x_m & , \textit{ if } \ell_{i,j} = \neg x_m, \\ \neg x_m & , \textit{ if } \ell_{i,j} = x_m. \end{cases}$$

2.2 First-order logic

Now we turn to first-order logic (FO). We consider only (finite) relational vocabularies, that is, our vocabularies contain no constants or functions. Therefore we use the notions relational vocabulary and vocabulary as synonyms. Since the only terms in our formulas are variables, we skip the definitions of terms and define formulas over variables directly.

Let σ be a vocabulary, *first-order formulas in* $\mathsf{FO}(\sigma)$ are defined by the following grammar:

$$\psi ::= x = y \mid x \neq y \mid R(\overline{x}) \mid \neg R(\overline{x}) \mid (\psi \wedge \psi) \mid (\psi \vee \psi) \mid \exists x\ \psi \mid \forall x\ \psi,$$

where x, y are variables, $\overline{x}$ is a tuple of variables and R is a relation symbol in σ.

When comparing our definition of first-order logic to the "standard" definition from the literature one might notice some differences: We allow no function and constant variables. This is a common practice in descriptive complexity and we follow this approach to ensure compatibility to established results from the literature (for example from [Fag74, SST95]). Moreover, our negation only applies to relations (this includes equality) and not arbitrary formulas, which is sometimes called *weak negation* or *atomic negation*. We made this choice since in team semantics one gets a more powerful logic when allowing arbitrary negation, which is also called *strong negation* in this context. However, for classical first-order logic both definitions—with strong negation or atomic negation—are equivalent. As for propositional logic we use $\psi_1 \to \psi_2$ as an abbreviation for the formula $\neg\psi_1 \vee \psi_2$ and $\psi_1 \leftrightarrow \psi_2$ for $(\psi_1 \to \psi_2) \wedge (\psi_2 \to \psi_1)$ in first-order logic. This time we have to pay attention when using the implication symbol since $\neg\psi_1$ might not be a formula in our logic, as we only allow atomic negation. Despite that we allow arbitrary first-order formulas ψ_1 on the left side of an implication, in that case $\psi_1 \to \psi_2$ means $\widetilde{\psi_1} \vee \psi_2$, where $\widetilde{\psi_1}$ is the formula $\neg\psi_1$ in negation normal form (negation pushed inwards, such that the negations only apply to relations and equality atoms). Given a formula ψ with free variables $x_1 \dots x_n$ we define $\mathsf{free}(\psi) := \{x_1, \dots, x_n\}$ as the set of free variables in ψ. For better readability we often write $\psi(x_1, \dots, x_n)$, to denote that $x_1, \dots, x_n$ are the free variables in ψ. A formula with no free variables is called *sentence*.

Let $\sigma = \{R_1^{i_1}, \dots R_k^{i_k}\}$ be a vocabulary, where $i_1 \dots i_k \in \mathbb{N}$ are the corresponding arities of the relations. A *σ-structure* $\mathcal{A} = (A, (R_j^{\mathcal{A}})_{R_j \in \sigma})$ is a tuple, which consists of a finite set A—that we also call *universe* or *domain*—and an interpretation for every relation symbol in σ, that is, $R_j^{\mathcal{A}} \subseteq A^{i_j}$ for every $R_j \in \sigma$. As notation we use $\mathrm{dom}(\mathcal{A})$ to denote the universe of $\mathcal{A}$. For convenience, we assume without loss of generality that the universe of a given structure is always the set $\{0, \dots, n-1\}$ for some $n \in \mathbb{N}$. Let σ be a vocabulary, we denote with $\mathrm{STRUC}(\sigma)$ the set of all σ-structures. We use $\leq, +$ and $\times$ in this context as relation symbols and assume that these symbols are always interpreted in the same way. More formally, let σ be a vocabulary with $\leq\ \in \sigma$, then $\mathrm{STRUC}(\sigma)$ is the set of σ-structures $\mathcal{A}$ with $\leq^{\mathcal{A}} = \{(x,y) \mid 0 \leq x \leq y \leq n-1\}$. Analogously $+$ and $\times$ are interpreted in a structure $\mathcal{A}$ as $+^{\mathcal{A}} = \{(x,y,z) \mid 0 \leq x,y,z \leq n-1 \text{ and } x+y=z\}$ and $\times^{\mathcal{A}} = \{(x,y,z) \mid 0 \leq x,y,z \leq n-1 \text{ and } x \cdot y = z\}$, respectively.

A *(first-order) assignment* is a function $s\colon V \to A$ that maps variables from a set V to an arbitrary finite set A. With $s_\emptyset$ we denote the *empty assignment*, that is, $V = \emptyset$. For an assignment $s\colon V \to A$ we define the new assignment $s(a/x)$, where $a \in A$ and $x \in V$, which assigns the value a to variable x and agrees with s on all other variables:

$$s(a/x)(y) := \begin{cases} a, & \text{if } x = y, \\ s(y), & \text{if } x \neq y. \end{cases}$$

Let $\overline{x} = (x_1, \dots, x_n)$ be a tuple of variables and s be an assignment, we define $s(\overline{x})$ as the tuple of values that s assigns to the variables from $\overline{x}$, that is, $s(\overline{x}) := (s(x_1), \dots, s(x_n))$. For an assignment $s\colon V \to A$, a structure $\mathcal{A}$ and a formula ψ we say *s is an assignment of $\mathcal{A}$* if $A = \mathrm{dom}(\mathcal{A})$ and *s is an assignment of ψ* if $\mathsf{free}(\psi) = V$.

Definition 2.2. *Let σ be a vocabulary, $\mathcal{A}$ be a σ-structure, s be an assignment of $\mathcal{A}$, x, y be variables, $\overline{x}$ be a tuple of variables, R be a relation from σ and $\psi_1, \psi_2 \in \mathsf{FO}(\sigma)$. The* satisfaction relation $\models$ *is defined as follows:*

$$\begin{array}{lll}
\mathcal{A} \models_s x = y & \iff & s(x) = s(y), \\
\mathcal{A} \models_s x \neq y & \iff & s(x) \neq s(y), \\
\mathcal{A} \models_s R(\overline{x}) & \iff & s(\overline{x}) \in R^{\mathcal{A}}, \\
\mathcal{A} \models_s \neg R(\overline{x}) & \iff & s(\overline{x}) \notin R^{\mathcal{A}}, \\
\mathcal{A} \models_s \psi_1 \wedge \psi_2 & \iff & \mathcal{A} \models_s \psi_1 \textit{ and } \mathcal{A} \models_s \psi_2, \\
\mathcal{A} \models_s \psi_1 \vee \psi_2 & \iff & \mathcal{A} \models_s \psi_1 \textit{ or } \mathcal{A} \models_s \psi_2, \\
\mathcal{A} \models_s \exists x\, \psi_1 & \iff & \textit{there is } a \in \mathrm{dom}(\mathcal{A}) \textit{ such that } \mathcal{A} \models_{s(a/x)} \psi_1, \\
\mathcal{A} \models_s \forall x\, \psi_1 & \iff & \textit{for every } a \in \mathrm{dom}(\mathcal{A}) \textit{ we have that } \mathcal{A} \models_{s(a/x)} \psi_1.
\end{array}$$

From now on, when we mention an assignment s in the context of a formula ψ and/or a structure $\mathcal{A}$—for example with $\mathcal{A}, s \models \psi$—we assume that s is an assignment of ψ and/or that s is an assignment of $\mathcal{A}$, respectively. To ease notation we often omit the vocabulary if it is arbitrary or obvious in the context. In that case we write for example FO instead of $\mathsf{FO}(\sigma)$ or "let $\mathcal{A}$ be a structure" instead of "let $\mathcal{A}$ be a σ-structure".

2.3 Second-order logic

Second-order logic (SO) is an extension of first-order logic, that additionally allows quantification of relations. Moreover relations may occur free in SO formulas, which means that these relations are neither quantified nor defined in the structure but have to be given separately. Formally, *second-order formulas* in $\mathsf{SO}(\sigma)$ are defined by the following grammar:

$$\psi ::= x = y \mid x \neq y \mid R(\overline{x}) \mid \neg R(\overline{x}) \mid (\psi \wedge \psi) \mid (\psi \vee \psi) \mid \exists x\ \psi \mid \forall x\ \psi \mid \exists R\ \psi \mid \forall R\ \psi,$$

where x, y are variables, $\overline{x}$ is a tuple of variables and R is a relation symbol in σ or an arbitrary relation symbol. We write $\psi(R_1, \ldots, R_k)$ to denote that relation symbols $R_1, \ldots, R_k$ appear free in formula ψ. An $\mathsf{SO}(\sigma)$ formula with neither free relations nor free variables is called $\mathsf{SO}(\sigma)$-*sentence*. As for first-order logic, we may omit the vocabulary σ and just write SO. Let $\psi \in \mathsf{SO}$, $\mathcal{A}$ be a structure and R be a relation of arity i, we say R is a relation of $\mathcal{A}$, if $R \subseteq \mathrm{dom}(\mathcal{A})^i$ and R is a relation of ψ, if ψ has a free relational variable of arity i. With $\mathrm{REL}(\mathcal{A}), \mathrm{REL}(\psi)$ and $\mathrm{REL}(\mathcal{A}, \psi)$, we denote the set of all relations of $\mathcal{A}$, the set of all relations of ψ and $\mathrm{REL}(\mathcal{A}) \cap \mathrm{REL}(\psi)$, respectively.

Definition 2.3. *Let $\sigma, \mathcal{A}, s, x, y, \overline{x}, R, \psi_1$ and ψ_2 be defined as in Definition 2.2. Furthermore let P be a relation symbol of arity i, with $P \notin \sigma$ and $\overline{Q}$ be a tuple of relations of $\mathcal{A}$. The* (second-order) satisfaction relation $\models$ *is defined by the rules from Definition 2.2 with the following additions:*

$$\begin{aligned}
\mathcal{A}, \overline{Q} &\models_s P(\overline{x}) &&\iff s(\overline{x}) \in Q_P, \text{ where } Q_P \in \overline{Q},\\
\mathcal{A}, \overline{Q} &\models_s \exists P\ \psi_1 &&\iff \textit{there is a relation } Q_P \textit{ such that } \mathcal{A}, \overline{Q}, Q_P \models_s \psi_1,\\
\mathcal{A}, \overline{Q} &\models_s \forall P\ \psi_1 &&\iff \textit{for every relation } Q_P \textit{ it holds } \mathcal{A}, \overline{Q}, Q_P \models_s \psi_1,
\end{aligned}$$

where Q_P is the relation assigned to relation symbol P.

By abuse of notation we might use the same symbol for the relational variable and the relation itself, for example we write $\mathcal{A}, R \models_s \psi(R)$.

In this thesis we are especially interested in three fragments of SO-logic, namely *existential second-order logic* (also known as ESO or Σ_1^1), the logic $\mathsf{FO}^{\mathrm{rel}}$ and the set of so-called *myopic formulas*, as these logics have similarities to the team logics we want to analyse. Existential second-order logic allows—as the name suggests—only existential quantification of relations. The logic $\mathsf{FO}^{\mathrm{rel}}$ consists of first-order formulas with a free relational variable, whereas myopic formulas ψ_1 are $\mathsf{FO}^{\mathrm{rel}}$ formulas that have a certain form:

$$\psi_1(R) = \forall \overline{x} \big(R(\overline{x}) \rightarrow \psi_2(R, \overline{x}) \big),$$

where ψ_2 is a first-order formula with only positive occurrences of the relation R. $\mathsf{FO}^{\mathrm{rel}}$ and myopic formulas are technically first-order formulas but as they are both allowed to have a free relational variable we list them here as fragments of second-order logic.

2.4 Team Semantics

In team semantics (first-order) formulas are not evaluated on single assignments but on sets of assignments, which we call *teams*. Formally a team $X = \{s_1, \dots s_n\}$ is a finite set of assignments $s_1, \dots, s_n$ that share the same domain and codomain. The set containing only the empty assignment $\{s_\emptyset\}$ and the empty set are also teams, the latter we call *empty team* in this context. As for assignments, we say X *is a team of* $\mathcal{A}$ or X *is a team of* φ if the assignments from X are assignments of $\mathcal{A}$ or of φ respectively. We denote with $\mathrm{TEAM}(\mathcal{A}), \mathrm{TEAM}(\varphi)$ and $\mathrm{TEAM}(\mathcal{A}, \varphi)$, the set of all teams of $\mathcal{A}$, the set of all teams of φ and $\mathrm{TEAM}(\mathcal{A}) \cap \mathrm{TEAM}(\varphi)$ respectively. When considering a team X, a structure $\mathcal{A}$ and a formula φ in the same context, we assume that X is a team of $\mathcal{A}$ and φ. Given a structure $\mathcal{A}$ and a formula φ there are $|\mathrm{dom}(\mathcal{A})|^{\mathsf{free}(\varphi)}$ many assignments of $\mathcal{A}$ and φ, and therefore $2^{|\mathrm{dom}(\mathcal{A})|^{\mathsf{free}(\varphi)}}$ teams of $\mathcal{A}$ and φ. We call the team containing all those assignments *full team* and use $X_{\mathrm{full}}^{\mathcal{A},\varphi}$ to denote the full team of $\mathcal{A}$ and φ or X_{full} if the structure and formula are clear from the context.

Definition 2.4. *Let X be a team, x be a variable, A be a set and $F\colon X \to 2^A \setminus \emptyset$ be a function. We define the* supplementing team $X[F/x]$ *and the* duplicate team $X[A/x]$ *as follows:*

$$X[F/x] := \{\, s(a/x) \mid s \in X \text{ and } a \in F(s) \,\}$$
$$X[A/x] := \{\, s(a/x) \mid s \in X \text{ and } a \in A \,\}.$$

Example 2.5. *We consider the structure $\mathcal{A} = (\{0,1,2,3\})$ (consisting only of its universe) and the team X consisting of the assignments s_1, s_2 given by Table 2.1. Furthermore let $F\colon X \to \{0,1,2,3\}$ be a function with $F(s_1) = \{0,1\}$ and $F(s_2) = \{1\}$. The supplementing team $X[F/y]$ consists of the assignments given in Table 2.2 and the duplicating team $X[A/x]$ of the ones given in Table 2.3.*

Definition 2.6 (Team semantics for first-order formulas)**.** *Let σ be a vocabulary, $\mathcal{A}$ be a σ-structure, X be a team of $\mathcal{A}$, x, y be variables, $\overline{x}$ be a tuple of variables, R be a relation from σ and φ_1, φ_2 be σ-formulas. The* satisfaction relation $\models$ *is defined as follows:*

$$\begin{array}{lll}
\mathcal{A} \models_X x = y & \iff & \text{for every } s \in X \text{ we have that } s(x) = s(y), \\
\mathcal{A} \models_X x \neq y & \iff & \text{for every } s \in X \text{ we have that } s(x) \neq s(y), \\
\mathcal{A} \models_X R(\overline{x}) & \iff & \text{for every } s \in X \text{ we have that } s(\overline{x}) \in R^{\mathcal{A}}, \\
\mathcal{A} \models_X \neg R(\overline{x}) & \iff & \text{for every } s \in X \text{ we have that } s(\overline{x}) \notin R^{\mathcal{A}}, \\
\mathcal{A} \models_X \varphi_1 \wedge \varphi_2 & \iff & \mathcal{A} \models_X \varphi_1 \text{ and } \mathcal{A} \models_X \varphi_2, \\
\mathcal{A} \models_X \varphi_1 \vee \varphi_2 & \iff & \text{there are } Y, Z \subseteq X \text{ with} \\
 & & Y \cup Z = X, \mathcal{A} \models_Y \varphi_1 \text{ and } \mathcal{A} \models_Z \varphi_2, \\
\mathcal{A} \models_X \exists x\, \varphi_1 & \iff & \mathcal{A} \models_{X[F/x]} \varphi_1 \text{ for some } F\colon X \to \mathrm{dom}(A), \\
\mathcal{A} \models_X \forall x\, \varphi_1 & \iff & \mathcal{A} \models_{X[A/x]} \varphi_1.
\end{array}$$

Table 2.1

	x_1	x_2	x_3
s_1	0	2	0
s_2	1	3	3

Table 2.2

	x_1	x_2	x_3	y
s'_1	0	2	0	0
s'_2	0	2	0	1
s'_3	1	3	3	1

Table 2.3

	x_1	x_2	x_3	y
s''_1	0	2	0	0
s''_2	0	2	0	1
s''_3	0	2	0	2
s''_4	0	2	0	3
s''_5	1	3	3	0
s''_6	1	3	3	1
s''_7	1	3	3	2
s''_8	1	3	3	3

Example 2.7. *We consider the structure $\mathcal{A} = (\{0, 1, 2, 3\}, R^{\mathcal{A}})$, with $R^{\mathcal{A}} = \{0, 1\}$ and the team X given in Table 2.1. In Table 2.4 we give examples of formulas that are satisfied by $\mathcal{A}$ and X and of ones that are not.*

In contrast to classical first-order logic, team semantics comes with some special properties. As we have seen in Example 2.7 there are formulas φ, structures $\mathcal{A}$ and teams X such that neither $\mathcal{A} \models_X \varphi$ nor $\mathcal{A} \models_X \neg\varphi$. In other words the law of the excluded middle does not hold in team semantics. Since we are mainly interested in satisfiability in this thesis the law of the excluded middle will not be of much relevance for us. There are more such properties discussed by Väänänen [Vää07], where the following are especially relevant for us.

Proposition 2.8 ([Vää07])**.**

1. *First-order formulas have the* empty team property*, that is, for any $\varphi \in \mathsf{FO}$ and any structure $\mathcal{A}$ it holds $\mathcal{A} \models_\emptyset \varphi$.*
2. *First-order formulas are* flat*, that is, for any $\varphi \in \mathsf{FO}$, any structure $\mathcal{A}$ and any team X it holds $\mathcal{A} \models_X \varphi$, if and only if $\mathcal{A} \models_s \varphi$ for all $s \in X$.*

By the empty team property any formula φ is satisfiable. Especially every sentence φ is trivially true in the sense that for any structure $\mathcal{A}$, $\mathcal{A} \models_\emptyset \varphi$ holds. We therefore redefine the satisfaction relation for sentences: A σ-sentence φ is satisfied in $\mathcal{A}$, if $\mathcal{A} \models_{\{s_\emptyset\}} \varphi$ holds. To simplify the notation we write $\mathcal{A} \models \varphi$ in that case. As we will later see, the empty team property also holds for other logics we define. For sentences this will not be an issue because of our definition above, but we still have to handle formulas. We therefore decide to exclude the empty team from the solution sets when defining computational problems based on team logics later. This will cause some issues, since we want to express problems from other fields that allow the “empty solution” using team logics and vice versa. Therefore

Table 2.4

Satisfaction relation	Explanation
$\mathcal{A} \models_X R(x_1)$	The values that appear in X for variable x_1 are $s_1(x_1) = 0$ and $s_2(x_1) = 1$, both values are in $R^{\mathcal{A}}$ and therefore $\mathcal{A} \models_X R(x_1)$.
$\mathcal{A} \models_X \neg R(x_2)$	The values that appear in X for variable x_2 are $s_1(x_2) = 2$ and $s_2(x_2) = 3$. Both values are not in $R^{\mathcal{A}}$ and therefore $\mathcal{A} \models \neg R(x_2)$.
$\mathcal{A} \not\models_X R(x_3)$	We have $s_2(x_3) = 3$, which is not in $R^{\mathcal{A}}$. Therefore, $\mathcal{A} \models_X R(x_3)$ does not hold.
$\mathcal{A} \not\models_X \neg R(x_3)$	We have $s_1(x_3) = 0$, which is in $R^{\mathcal{A}}$. Therefore, $\mathcal{A} \models_X \neg R(x_3)$ does not hold.
$\mathcal{A} \models_X R(x_1) \vee R(x_3)$	We choose $Y = \{s_2\}$ and $Z = \{s_1\}$. Now we have $\mathcal{A} \models_Y R(x_1)$ and $\mathcal{A} \models_Z R(x_3)$.
$\mathcal{A} \models_X R(x_1) \wedge \neg R(x_2)$	Since $\mathcal{A} \models_X R(x_1)$ and $\mathcal{A} \models_X \neg R(x_2)$ hold we can conclude $\mathcal{A} \models_X R(x_1) \wedge \neg R(x_2)$.
$\mathcal{A} \models_X \exists y\, R(y)$	We choose the function F from Example 2.5. The values that appear in the corresponding supplementing team $X[F/y]$ for y are $s'_1(y) = 0$ and $s'_2(y) = s'_3(y) = 1$, which are both in $R^{\mathcal{A}}$ and therefore $\mathcal{A} \models_X \exists y\, R(y)$.
$\mathcal{A} \not\models_X \forall y\, R(y)$	Consider the duplicating team $X[A/x]$ from Example 2.5. As $s''_3(y) = 2$ appears as a value for y and is not in $R^{\mathcal{A}}$, we can conclude $\mathcal{A} \not\models_X \forall y\, R(y)$.

one of our tasks will be to find equivalent problems that somehow exclude the empty solution.

Because of the flatness property FO formulas are not that interesting in team semantics. We therefore extend FO formulas by new atoms which we will call *team atoms*. We start by defining the three most studied atoms which are also the focus of this thesis. These are the *dependency atom* $=(\overline{x}, y)$, the *independency atom* $\overline{x} \perp_{\overline{z}} \overline{y}$ and the *inclusion atom* $\overline{x} \subseteq \overline{y}$, where $\overline{x}, \overline{y}, \overline{z}$ are tuples of variables and y is a single variable.

Definition 2.9. *Let σ be a vocabulary, $\mathcal{A}$ be a σ-structure, X be a team of $\mathcal{A}$, y be a variable and $\overline{x}, \overline{y}, \overline{z}$ be tuples of variables. The* satisfaction relation for team atoms $\models$ *is defined as follows:*

$$\begin{array}{llll}
\mathcal{A} \models_X & =(\overline{x}, y) & \iff & \textit{for every } s,t \in X \textit{ we have that } s(\overline{x}) = t(\overline{x}) \textit{ implies} \\
 & & & s(y) = t(y), \\
\mathcal{A} \models_X & \overline{x}\perp_{\overline{z}}\overline{y} & \iff & \textit{for every } s,t \in X \textit{ with } s(\overline{z}) = t(\overline{z}) \textit{ there is } u \in X \\
 & & & \textit{such that } u(\overline{x}) = s(\overline{x}),\ u(\overline{y}) = t(\overline{y}) \text{ and } u(\overline{z}) = s(\overline{z}), \\
\mathcal{A} \models_X & \overline{x} \subseteq \overline{y} & \iff & \textit{for every } s \in X \textit{ there is } t \in X \textit{ such that } s(\overline{x}) = t(\overline{y}).
\end{array}$$

Intuitively, the dependence atom $=(\overline{x}, y)$ expresses dependency in the sense that if two assignments agree on $\overline{x}$ they must also agree on y. The independence atom $\overline{x}\perp\overline{y}$ expresses "total" independency, which means in our case that all values that are assigned to $\overline{x}$ and $\overline{y}$ by (potentially different) assignments must also appear in one assignment combined. This is not to be confused with the negation of the dependence atom, which expresses another kind of independecy not considered in this thesis. The inclusion atom $\overline{x} \subseteq \overline{y}$ expresses that for $\overline{x}$ only values may appear that also appear for $\overline{y}$. When considering the sets of those values $R_{\overline{x}}, R_{\overline{y}}$, we can observe a subset relation between those sets, that is, $R_{\overline{x}} \subseteq R_{\overline{y}}$.

Example 2.10. *We consider the structure $\mathcal{A} = (\{0,1\})$ (consisting only of its universe) and the team X consisting of the assignments s_1, s_2, s_3, s_4 given in Table 2.5. We have that $\mathcal{A} \models_X \varphi$ for $\varphi \in \{=(x_1,x_2), x_1\perp x_3, x_1 \subseteq x_3\}$ and $\mathcal{A} \not\models_x \varphi'$ for $\varphi' \in \{=(x_2,x_1), x_1\perp x_4, x_1 \subseteq x_2\}$. In Table 2.6 we explain why this is the case. Note that, for example, the formula $x_1\perp x_2$ is also satisfied by $\mathcal{A}$ and X, which means that at the same time x_1 and x_2 are independent but also x_2 depends on x_1 (since $\mathcal{A} \models_X =(x_1,x_2)$, see above) in our notion.*

Table 2.5

	x_1	x_2	x_3	x_4
s_1	0	0	0	1
s_2	0	0	1	1
s_3	1	0	0	0
s_4	1	0	1	0

For a vocabulary σ and a set of team atoms $\mathrm{T} \subseteq \{=(\dots), \perp, \subseteq\}$ we define $\mathsf{FO}(\sigma, \mathrm{T})$ as the logic of first-order formulas over σ that can contain atoms from T. For convenience we omit the curly brackets if T consists of only one atom and may omit σ as mentioned before. We give names to the logics we get by extending first-order logic by one of the three atoms we defined already: $\mathsf{FO}(=(\dots))$ is called *dependence logic*, $\mathsf{FO}(\perp)$ is called *independence logic* and $\mathsf{FO}(\subseteq)$ is called *inclusion logic.*

Table 2.6

Satisfaction relation	Explanation
$\mathcal{A} \models_X =(x_1, x_2)$	There are two interesting pairs of assignments since they agree on x_1, one being s_1, s_2 and the other being s_3, s_4. Since for both pairs the respective assignments also agree on x_2, the required implication holds. All other pairs do not agree on x_1 and therefore satisfy the implication trivially.
$\mathcal{A} \not\models_X =(x_2, x_1)$	Since $s_1(x_2) = s_3(x_2)$ but $s_1(x_1) \neq s_3(x_1)$ the formula $=(x_2, x_1)$ is not satisfied by the team X in $\mathcal{A}$.
$\mathcal{A} \models_X x_1 \perp x_3$	This holds since for any combination of values ((0, 0), (0, 1), (1, 0), (1, 1)), there is an assignment that assigns those values to x_1 and x_3. More formally, for any pair $s, t \in X$ one of the four assignments s_1, s_2, s_3, s_4 agrees with s on x_1 and with t on x_3.
$\mathcal{A} \not\models_X x_1 \perp x_4$	We have $s_1(x_1) = 0$ and $s_3(x_4) = 0$ but no assignment $s \in X$ that agrees on both.
$\mathcal{A} \models_X x_1 \subseteq x_3$	For any assignment $s \in X$ either $s(x_1) = s_1(x_3)$ or $s(x_1) = s_2(x_3)$.
$\mathcal{A} \not\models_X x_1 \subseteq x_2$	For the assignment s_3 there is no assignment $s \in X$ with $s_3(x_1) = s(x_2)$.

Proposition 2.11 ([Gal12, GH13, GV13, Vää07]).

1. *The dependence atom* $=(\dots)$ *and the inclusion atom* $\subseteq$ *can be expressed using the independence atom* $\perp$*:*

$$\begin{aligned} =(\overline{x}, y) &\equiv y \perp_{\overline{x}} y \\ \overline{x} \subseteq \overline{y} &\equiv \forall v_1 \forall v_2 \forall \overline{z} \Big((\overline{z} \neq \overline{x} \wedge \overline{z} \neq \overline{y}) \vee (v_1 \neq v_2 \wedge \overline{z} \neq \overline{y}) \\ &\qquad \vee \big((v_1 = v_2 \vee \overline{z} = \overline{y}) \wedge \overline{z} \perp (v_1, v_2) \big) \Big) \end{aligned}$$

2. *Let* $\mathrm{T} \subseteq \{=(\dots), \perp, \subseteq\}$*. Formulas of* $\mathsf{FO}(\mathrm{T})$ *have the empty team property.*

3. *Formulas of* $\mathsf{FO}(=(\dots))$ *are* downwards closed*, that is: If* $\mathcal{A} \models_X \varphi$ *and* $Y \subseteq X$*, then* $\mathcal{A} \models_Y \varphi$ *for any structure* $\mathcal{A}$*, formula* φ *and teams* X, Y*.*

4. *Formulas of* $\mathsf{FO}(\subseteq)$ *are* closed under union*, that is: If* $\mathcal{A} \models_X \varphi$ *and* $\mathcal{A} \models_Y \varphi$*, then* $\mathcal{A} \models_{X \cup Y} \varphi$ *for any structure* $\mathcal{A}$*, formula* φ *and teams* X, Y*.*

The union closure property of inclusion logic implies a really nice property: Given a formula $\varphi \in \mathsf{FO}(\subseteq)$ and a structure $\mathcal{A}$, there is a unique maximal team X with $\mathcal{A} \models_X \varphi$. Moreover, when in addition given a team X', the maximal team $X \subseteq X'$ with that property is unique as well.

The above atoms expressing team properties can be generalized, which we will do next. For this, we introduce some more notation: With the expression $\mathrm{rel}(X)$ we denote the translation of a team X to a relation, formally

$$\mathrm{rel}(X) = \{(s(x_1), \dots, s(x_k)) \mid s \in X\}.$$

For an assignment $s\colon V \to A$ and a set of variables $V' \subseteq V$, we denote by $s|_{V'}$ the *restriction of* s *to* V', that is, $s|_{V'}\colon V' \to A$ with $s|_{V'}(x) = s(x)$ for all $x \in V'$. Furthermore, for a team X over V and a set of variables $V' \subseteq V$ we denote by $X|_{V'}$ the *restriction of* X *to* V', that is, $X|_{V'} = \{s|_{V'} \mid s \in X\}$. With $\mathsf{vars}(\overline{x})$ we denote the set of variables in the tuple $\overline{x}$.

Definition 2.12. *Let* $i_1, \dots, i_n > 0$*, and* $\sigma = (R_1^{i_1}, \dots, R_n^{i_n})$ *be a vocabulary. A* generalized quantifier of type $(i_1, \dots, i_n)$ *is a class* Q *of* σ*-structures that is closed under isomorphism.*

Let Q *be a generalized quantifier of type* $(i_1, \dots, i_n)$*. We extend the syntax of first-order logic with an expression* $G_{\mathrm{Q}}(\overline{x_1}, \dots, \overline{x_n})$*, where each* $\overline{x_j}$ *is a tuple of variables of length* i_j*. We call* G_Q *a* generalized team atom (of type $(i_1, \dots, i_n)$)*, and define its satisfaction relation by:*

$$\mathcal{A} \models_X G_{\mathrm{Q}}(\overline{x_1}, \dots, \overline{x_n}) \iff \big(\mathrm{dom}(\mathcal{A}), \mathrm{rel}(X|_{\mathsf{vars}(\overline{x_1})}), \dots, \mathrm{rel}(X|_{\mathsf{vars}(\overline{x_n})})\big) \in \mathrm{Q}.$$

Of course one can express the dependence, independence and inclusion atoms in terms of generalized quantifiers. We do this exemplarily for the inclusion atom.

Example 2.13. *We define a generalized quantifier* Q *such that the corresponding generalized team atom* G_{Q} *expresses the inclusion atom* $\subseteq$ *(for single variables). The class* Q *consists of all structures* $\mathcal{A} = (A, R_1, R_2)$ *with the property* $R_1 \subseteq R_2$*. Clearly* Q *is closed under isomorphisms and we have that*

$$\begin{aligned}
\mathcal{A} \models_X G_{\mathrm{Q}}(x, y) &\iff (\mathrm{dom}(\mathcal{A}), \mathrm{rel}(X|_x), \mathrm{rel}(X|_y) \in \mathrm{Q})\\
&\iff \mathrm{rel}(X|_x) \subseteq \mathrm{rel}(X|_y)\\
&\iff \text{for every } s \in X \text{ there is } t \in X \textit{ such that } s(\overline{x}) = t(\overline{y})\\
&\iff \mathcal{A} \models_X x \subseteq y.
\end{aligned}$$

We denote by $\mathfrak{G}$ the set of all generalized team atoms. Given a set $\mathrm{T} \subseteq \mathfrak{G}$ of generalized team atoms, we define the logic $\mathsf{FO}(\mathrm{T})$ which consists of all first-order formulas extended by the atoms from T. Under the term *team logics*, we summarise every such extension of first-order logic. We say a set of generalized team atoms $\mathrm{T} \subseteq \mathfrak{G}$ is NP-*verifiable*, if the model checking problem $\mathrm{VERIFYTEAM}_\varphi$ is in NP for every formula $\varphi \in \mathsf{FO}(\mathrm{T})$. The set of all NP-verifiable generalized team atoms is denoted with $\mathfrak{G}_{\mathsf{NP}}$.

Problem:	$\mathrm{VERIFYTEAM}_\varphi$
Input:	structure $\mathcal{A}$, team X
Question:	$\mathcal{A} \models_X \varphi$ and $X \neq \emptyset$?

Proposition 2.14 ([Grä13])**.** *Let* T *be a set of generalized team atoms that can be evaluated in polynomial time. Then* $\mathrm{T} \subseteq \mathfrak{G}_{\mathsf{NP}}$.

Proof. This Proposition is a special case of a theorem by Grädel (see [Grä13]) which states that the model-checking problem $\mathrm{VERIFYTEAM}_{\mathsf{FO}(\mathrm{T})}$ is in NEXPTIME and furthermore in NP if the input formula has bounded width. In this context the width of a formula φ is the maximal number of free variables of its subformulas.

Problem:	$\mathrm{VERIFYTEAM}_{\mathsf{FO}(\mathrm{T})}$
Input:	$\varphi \in \mathsf{FO}(\mathrm{T})$, structure $\mathcal{A}$, team X
Question:	$\mathcal{A} \models_X \varphi$ and $X \neq \emptyset$?

Note that compared to the model-checking problem $\mathrm{VERIFYTEAM}_\varphi$ we defined before, in the problem $\mathrm{VERIFYTEAM}_{\mathsf{FO}(\mathrm{T})}$ the formula is part of the input and not fixed. Because in our case the formula is fixed, its width is bounded and therefore $\mathrm{VERIFYTEAM}_\varphi \in \mathsf{NP}$ for $\varphi \in \mathsf{FO}(\mathrm{T})$. □

Corollary 2.15. *Let* $\mathrm{T} \subseteq \{=(\dots), \perp, \subseteq\}$. *Then* T *is* NP-*verifiable.*

Proof. This follows from Proposition 2.14, since all three atoms $=(\dots), \perp$ and $\subseteq$ can be evaluated in polynomial time. □

2.5 Complexity Theory

In this section we introduce a framework for defining computational problems, which allows us to quickly compare problems from different settings. On top of that we define complexity classes and certain properties of them.

We formally define computational problems such as decision problems, counting problems and enumeration problems by functions that map instances to sets of solutions. For this let Σ be an alphabet with $|\Sigma| \geq 2$. We call any function $f\colon \Sigma^* \to 2^{\Sigma^*}$ a *computational problem function*, in short CPF. For a given $x \in \Sigma^*$ we call $f(x)$ the *set of solutions of* x *in* f and every element $y \in f(x)$ a *solution of* x *in* f. We say f is *polynomially bounded*, if there is a polynomial p such that for all $x \in \Sigma^*$ and all $y \in f(x)$ it holds that $|y| \leq p(|x|)$.

Now, given a (polynomially bounded) computational problem function f, we can easily define decision (D-f), counting (C-f) and enumeration (E-f) problems:

Problem:	D-f
Input:	$x \in \Sigma^*$
Question:	$f(x) \neq \emptyset$?

Problem:	C-f
Input:	$x \in \Sigma^*$
Output:	$\lvert f(x)\rvert$

Problem:	E-f
Input:	$x \in \Sigma^*$
Output:	$f(x)$

This framework also allows us to define other computation problems, such as the search problem which asks for a solution given an input (S-f) or the model-checking problem (V-f), which will turn out to be quite useful.

Problem:	S-f
Input:	$x \in \Sigma^*$
Output:	Any $y \in f(x)$

Problem:	V-f
Input:	$x, y \in \Sigma^*$
Question:	$y \in f(x)$?

For convenience, when defining arbitrary problems this way, we will define the corresponding function implicitly. For example we write "Let E-f be an enumeration problem" instead of "Let f be a polynomial bounded CPF and E-f be the corresponding enumeration problem". For this definition Σ can be arbitrary (with $|\Sigma| \geq 2$) but from now on we assume that $\Sigma = \{0, 1\}$. For readability, when we specifically define CPFs and computational problems, we will hide the underlying encoding over Σ. If the argument is not an encoding of the respective object, then the image value is the empty set. For example, in the definition of the function $\mathsf{sat}^{\mathrm{team}}_{\varphi}$ (see below), instead of $\mathsf{sat}^{\mathrm{team}}_{\varphi}\colon \Sigma^* \to 2^{\Sigma^*}$, we write $\mathsf{sat}^{\mathrm{team}}_{\varphi}\colon \mathrm{STRUC} \to 2^{\mathrm{TEAM}(\varphi)}$. Here we want to express that $\mathsf{sat}^{\mathrm{team}}_{\varphi}$ maps structures to sets of teams and, for this, assume suitable encodings of structures and teams over Σ^*, respectively. Later we will also define non-unary functions. For this we assume a suitable encoding of all the arguments of the respective function over Σ^* as well. If at some point the encoding becomes important, we write $\mathrm{enc}(Z)$ to specify the encoding of object Z over alphabet Σ.

As an example, we define the satisfiability problems that are the focus of this thesis. Let $\mathrm{T} \subseteq \mathfrak{S}$ and $\varphi \in \mathsf{FO}(\mathrm{T})$. We define the CPF $\mathsf{sat}_\varphi^{\mathrm{team}}\colon \mathrm{STRUC} \to 2^{\mathrm{TEAM}(\varphi)}$ with

$$\mathsf{sat}_\varphi^{\mathrm{team}}(\mathcal{A}) \coloneqq \{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \,\}\,.$$

The corresponding problems are defined as D-$\mathsf{sat}_\varphi^{\mathrm{team}}$, C-$\mathsf{sat}_\varphi^{\mathrm{team}}$, E-$\mathsf{sat}_\varphi^{\mathrm{team}}$ with:

Problem:	D-$\mathsf{sat}_\varphi^{\mathrm{team}}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Question:	$\mathsf{sat}_\varphi^{\mathrm{team}}(\mathcal{A}) \neq \emptyset$?

Problem:	C-$\mathsf{sat}_\varphi^{\mathrm{team}}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\lvert\mathsf{sat}_\varphi^{\mathrm{team}}(\mathcal{A})\rvert$

Problem:	E-$\mathsf{sat}_\varphi^{\mathrm{team}}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\mathsf{sat}_\varphi^{\mathrm{team}}(\mathcal{A})$

or if we replace $\mathsf{sat}_\varphi^{\mathrm{team}}(\mathcal{A})$ with its image

Problem:	D-$\mathsf{sat}_\varphi^{\mathrm{team}}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Question:	$\{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \,\} \neq \emptyset$?

Problem:	C-$\mathsf{sat}_\varphi^{\mathrm{team}}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\lvert \{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \,\} \rvert$

Problem:	E-$\mathsf{sat}_\varphi^{\mathrm{team}}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \,\}$

The corresponding model-checking problem would be defined as

Problem:	V-$\mathsf{sat}^{\text{team}}_{\varphi}$
Input:	$\mathcal{A} \in \text{STRUC}, X' \in \text{TEAM}(\varphi)$
Question:	$X' \in \{\, X \in \text{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi$ and $X \neq \emptyset \,\}$?

which is equivalent to the model-checking problem $\text{VERIFYTEAM}_{\varphi}$ we defined in Section 2.4. Note that in these problems the formula φ is fixed and not part of the input. In fact we just defined infinitely many CPFs, four for each formula φ. The size of the solutions is bounded by the size of the full team, that is, $|X_{\text{full}}| = |\text{dom}(\mathcal{A})|^{|\mathsf{free}(\varphi)|}$. Since φ is fixed, $|\mathsf{free}(\varphi)|$ is a constant and therefore $|\text{dom}(\mathcal{A})|^{|\mathsf{free}(\varphi)|}$ is polynomial. Thus, $\mathsf{sat}^{\text{team}}_{\varphi}$ is polynomially bounded.

We consider three types of *complexity classes*. A *decision complexity class* is a set of decision problems. Analogously we define *counting complexity classes* and *enumeration complexity classes* as sets of counting problems and enumeration problems respectively. Given a complexity class, one might want to compare the problems contained in it to each other in terms of complexity. For this, we introduce a comparison notion—called *reduction*—, which differs depending on the respective complexity class. We assume that the reader is familiar with the classes Σ^{P}_{k} (for $k \geq 0$) of the polynomial hierarchy, polynomial-time many-one reductions ($\leq^{\mathsf{P}}_{m}$) and log-space reductions ($\leq^{\log}_{m}$). We will define other complexity classes as well as reducibility notions later in Sections 2.6, 2.7 and 2.8. Since all the reducibility notions we define in this thesis are polynomial-time computable, we sometimes omit "polynomial-time" and, for example, only write "many-one reducible". This is not to be confused with the generalisations of the reductions, where the reduction function only needs to be computable.

Definition 2.16. *Let A be a computational problem and* C *be a complexity class. We say A is* C-hard with respect to reducibility notion $\leq$ *if for every $B \in \mathsf{C}$ it holds that $B \leq A$. Furthermore we say A is* C-complete with respect to reducibility notion $\leq$, *if $A \in \mathsf{C}$ and A is C-hard with respect to reducibility notion $\leq$.*

Given a complexity class, there are weaker and stronger reducibility notions and therefore hardness and completeness results have different impacts depending on the notion. Hence, some reductions are more suitable than others for a specific complexity class. Formally, for a reducibility notion $\leq$ to be suitable for a class C, we want C to be closed under $\leq$, as defined below.

Definition 2.17. *A complexity class* C *is* closed under some reducibility $\leq$, *if for any two computational problems A, B it holds that:*

$$\text{if } A \leq B \text{ and } B \in \mathsf{C} \text{ then } A \in \mathsf{C}.$$

When a complexity class is closed under a reduction we have some nice properties such as: upper bounds translate "from right to left" and lower bound "from left to right". These properties will help us later to show membership and hardness results and subset relationships between complexity classes.

Next we define the notion of oracle Turing machines. We start in the decision setting. A *(decision) oracle Turing machine M with oracle* D-f is a Turing machine with two new special tapes—the query tape and the answer tape—and two new states corresponding to those tapes. The machine may use the query tape like a normal tape. At any point in the computation the machine may ask the oracle for an answer to the word written on the query tape by going into the query state. The oracle outputs the answer to the query "$x \in$ D-f?" in one time step on the answer tape and the machine switches into the answer state. Afterwards, the computation of M continues. Given an oracle Turing machine M, we use $\mathrm{M}^{\text{D-}f}$ to denote that we give the machine access to the oracle D-f. A *counting oracle Turing machine* is defined analogously but instead of a decision problem it uses a counting problem C-f as an oracle. There is also a notion of oracle machines in the enumeration setting, which we will not define here but later in Section 2.8 due to its complexity. Let C be a complexity class defined via Turing machines (with some computational bounds) and D be an arbitrary complexity class, we denote with C^{D} the class of problems that can be solved by a machine with access to an oracle in D and the computational bounds of C. For example, P^{NP} is the set of problems that can be decided by a polynomial-time Turing machine with access to an NP-oracle.

Now, we define all (polynomially bounded) CPFs—and by this also implicitly computational problems—that we consider in this thesis. The idea of this is to have proper formal definitions of all CPFs in one place, but for readability we will restate the definitions of the corresponding problems at the places where we need them. We group these CPFs in different categories: The first group consists of CPFs we aim to achieve new results for, the second group consists of those that we only use as oracles and the third group contains CPFs that we only use in reductions. Any CPFs defined in this section are polynomially bounded, as one can easily show.

First group The CPF $\mathsf{sat}_\varphi^{\text{team}}$, which was defined above, belongs to the first group of CPFs. The other CPFs of this group are different variants of $\mathsf{sat}_\varphi^{\text{team}}$, where we restrict the solutions to those that are (inclusion) maximal/minimal or of maximum/minimum cardinality. Formally, given $\mathrm{T} \subseteq \mathfrak{G}$, a formula $\varphi \in \mathsf{FO}(\mathrm{T})$, a structure $\mathcal{A}$ and a team $X \neq \emptyset$ with $\mathcal{A} \models_X \varphi$, we say X is

$$\begin{aligned}
\textit{maximal satisfying} &\iff \forall X'\ X \subsetneq X' \text{ implies } \mathcal{A} \not\models_{X'} \varphi,\\
\textit{minimal satisfying} &\iff \forall X' \neq \emptyset\ X' \subsetneq X \text{ implies } \mathcal{A} \not\models_{X'} \varphi,\\
\textit{maximum satisfying} &\iff \forall X'\ |X'| > |X| \text{ implies } \mathcal{A} \not\models_{X'} \varphi \text{ and}
\end{aligned}$$

$$\textit{minimum satisfying} \iff \forall X' \neq \emptyset\ |X'| < |X| \text{ implies } \mathcal{A} \not\models_{X'} \varphi.$$

Since all of these terms describe satisfying teams we will sometimes omit the word "satisfying" and write for example "let X be a maximal team". When using the term *optimum*, we refer to either of the above defined terms.

Let $\mathrm{T} \subseteq \mathfrak{G}$ and $\varphi \in \mathsf{FO}(\mathrm{T})$, we define the CPFs

$$\mathsf{sat}^{\text{team}}_{\varphi} \colon \begin{cases} \text{STRUC} \to 2^{\text{TEAM}(\varphi)} \\ \mathcal{A} \mapsto \{\, X \in \text{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \,\}, \end{cases}$$

$$\mathsf{maxsat}^{\text{team}}_{\varphi} \colon \begin{cases} \text{STRUC} \to 2^{\text{TEAM}(\varphi)} \\ \mathcal{A} \mapsto \{\, X \in \text{TEAM}(\mathcal{A}, \varphi) \mid X \text{ is a maximal team for } \varphi \text{ in } \mathcal{A} \,\}, \end{cases}$$

$$\mathsf{minsat}^{\text{team}}_{\varphi} \colon \begin{cases} \text{STRUC} \to 2^{\text{TEAM}(\varphi)} \\ \mathcal{A} \mapsto \{\, X \in \text{TEAM}(\mathcal{A}, \varphi) \mid X \text{ is a minimal team for } \varphi \text{ in } \mathcal{A} \,\}, \end{cases}$$

$$\mathsf{cmaxsat}^{\text{team}}_{\varphi} \colon \begin{cases} \text{STRUC} \to 2^{\text{TEAM}(\varphi)} \\ \mathcal{A} \mapsto \{\, X \in \text{TEAM}(\mathcal{A}, \varphi) \mid X \text{ is a maximum team for } \varphi \text{ in } \mathcal{A} \,\}, \end{cases}$$

$$\mathsf{cminsat}^{\text{team}}_{\varphi} \colon \begin{cases} \text{STRUC} \to 2^{\text{TEAM}(\varphi)} \\ \mathcal{A} \mapsto \{\, X \in \text{TEAM}(\mathcal{A}, \varphi) \mid X \text{ is a minimum team for } \varphi \text{ in } \mathcal{A} \,\}. \end{cases}$$

Second group Let $\mathrm{T} \subseteq \mathfrak{G}$ and $\varphi \in \mathsf{FO}(\mathrm{T})$, we define the CPFs

$$\mathsf{extendteam}_{\varphi} \colon \begin{cases} \text{STRUC} \times \text{TEAM}(\varphi)^2 \to 2^{\text{TEAM}(\varphi)} \\ (\mathcal{A}, X, Y) \mapsto \left\{ X' \in \text{TEAM}(\mathcal{A}, \varphi) \;\middle|\; \begin{array}{l} \mathcal{A} \models_{X'} \varphi,\ X \subsetneq X' \\ \text{and } X' \cap Y = \emptyset \end{array} \right\}, \end{cases}$$

$$\mathsf{extendteam}'_{\varphi} \colon \begin{cases} \text{STRUC} \times \text{TEAM}(\varphi) \to 2^{\text{TEAM}(\varphi)} \\ (\mathcal{A}, X) \mapsto \{\, X' \in \text{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_{X'} \varphi, X \subsetneq X' \,\}, \end{cases}$$

$$\mathsf{extendcteam}_{\varphi} \colon \begin{cases} \text{STRUC} \times \text{TEAM}(\varphi)^2 \times \mathbb{N} \to 2^{\text{TEAM}(\varphi)} \\ (\mathcal{A}, X, Y, k) \mapsto \left\{ X' \in \text{TEAM}(\mathcal{A}, \varphi) \;\middle|\; \begin{array}{l} \mathcal{A} \models_{X'} \varphi,\ X \subsetneq X', \\ X' \cap Y = \emptyset \text{ and} \\ |X'| = k' \end{array} \right\}, \end{cases}$$

$$\mathsf{maxsubteam}_{\varphi} \colon \begin{cases} \text{STRUC} \times \text{TEAM}(\varphi) \to 2^{\text{TEAM}(\varphi)} \\ (\mathcal{A}, X) \mapsto \left\{ X' \in \text{TEAM}(\mathcal{A}, \varphi) \;\middle|\; \begin{array}{l} \mathcal{A} \models_{X'} \varphi,\ X' \subseteq X \text{ and} \\ \forall X'' \subseteq X \colon |X''| > |X'| \\ \implies \mathcal{A} \not\models_{X''} \varphi \end{array} \right\}. \end{cases}$$

The CPF $\mathsf{maxsubteam}_{\varphi}$ does not really fit in this category (or the other two categories) as we are only interested in the S-version, which we formally may not use as an oracle, but since we use it in similar fashion to an oracle we list it here.

Third group To improve readability we split this group into two blocks. The first block contains logic based CPFs. Let $\mathrm{T} \in \mathfrak{G}, \varphi \in \mathsf{FO}(\mathrm{T}), \psi \in \mathsf{SO}, \mathrm{P}_1, \mathrm{P}_2 \subseteq \Sigma_1\mathrm{BF}$ and $\mathrm{P}_3 \subseteq \mathrm{CNF}$. We define the CPFs

$$\mathsf{k\text{-}minsat}^{\mathrm{team}}_{\varphi}\colon \begin{cases} \mathrm{STRUC} \times \mathbb{N} \to 2^{\mathrm{TEAM}(\varphi)} \\ (\mathcal{A}, k) \mapsto \left\{ X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \,\middle|\, \begin{array}{l} \mathcal{A} \models_X \varphi,\ X \neq \emptyset \\ \text{and } |X| \leq k \end{array} \right\}, \end{cases}$$

$$\mathsf{anothersolminsat}^{\mathrm{team}}_{\varphi}\colon \begin{cases} \mathrm{STRUC} \times 2^{\mathrm{TEAM}(\varphi)} \to 2^{\mathrm{TEAM}(\varphi)} \\ (\mathcal{A}, M) \mapsto \left\{ X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \,\middle|\, \begin{array}{l} X \notin M \text{ and} \\ X \text{ is minimal} \\ \text{for } \varphi \text{ in } \mathcal{A} \end{array} \right\}, \end{cases}$$

$$\mathsf{k\text{-}minsat}^{\mathrm{rel}}_{\psi}\colon \begin{cases} \mathrm{STRUC} \times \mathbb{N} \to 2^{\mathrm{REL}(\psi)} \\ (\mathcal{A}, k) \mapsto \left\{ R \in \mathrm{REL}(\mathcal{A}, \psi) \,\middle|\, \begin{array}{l} \mathcal{A}, R \models \psi,\ R \neq \emptyset \\ \text{and } |R| \leq k \end{array} \right\}, \end{cases}$$

$$\mathsf{anothersolminsat}^{\mathrm{rel}}_{\psi}\colon \begin{cases} \mathrm{STRUC} \times 2^{\mathrm{REL}(\psi)} \to 2^{\mathrm{REL}(\psi)} \\ (\mathcal{A}, M) \mapsto \left\{ R \in \mathrm{REL}(\mathcal{A}, \psi) \,\middle|\, \begin{array}{l} R \notin M \text{ and } R \text{ is} \\ \text{minimal for } \psi \text{ in } \mathcal{A} \end{array} \right\}, \end{cases}$$

$$\mathsf{sat}^{\mathrm{rel}}_{\psi}\colon \begin{cases} \mathrm{STRUC} \to 2^{\mathrm{REL}(\psi)} \\ \mathcal{A} \mapsto \{\, R \in \mathrm{REL}(\mathcal{A}, \psi) \mid \mathcal{A}, R \models \psi \,\}, \end{cases}$$

$$\mathsf{sat}^{\mathrm{rel},\neg\emptyset}_{\psi}\colon \begin{cases} \mathrm{STRUC} \to 2^{\mathrm{REL}(\psi)} \\ \mathcal{A} \mapsto \{\, R \in \mathrm{REL}(\mathcal{A}, \psi) \mid \mathcal{A}, R \models \psi \text{ and } R \neq \emptyset \,\}, \end{cases}$$

$$\mathsf{sat}_{\mathrm{P}_1}\colon \begin{cases} \mathrm{P}_1 \to 2^{\Theta} \\ \chi \mapsto \{\, \beta \in \Theta(\chi) \mid \beta \models \chi \,\}, \end{cases}$$

$$\mathsf{sat}^{\neg\emptyset}_{\mathrm{P}_1}\colon \begin{cases} \mathrm{P}_1 \to 2^{\Theta} \\ \chi \mapsto \{\, \beta \in \Theta(\chi) \mid \beta \models \chi \text{ and } \beta \neq \emptyset \,\}, \end{cases}$$

$$(\mathsf{sat}_{\mathrm{P}_1}, \mathsf{sat}_{\mathrm{P}_2})\colon \begin{cases} \mathrm{P}_1 \times \mathrm{P}_2 \to 2^{\Theta} \\ (\chi_1, \chi_2) \mapsto \left\{ \beta \in \Theta(\chi_1) \cup \Theta(\chi_2) \,\middle|\, \begin{array}{l} \beta \models \chi_1 \text{ and} \\ \beta \models \chi_2 \end{array} \right\}, \end{cases}$$

$$\mathsf{sat}^{1}_{\mathrm{P}_3}\colon \begin{cases} \mathrm{P}_3 \to 2^{\Theta} \\ \chi \mapsto \left\{ \beta \in \Theta(\chi) \,\middle|\, \begin{array}{l} \text{In each clause } C_i \text{ of } \chi \text{ there} \\ \text{is exactly one literal } \ell_{i,j} \\ \text{that evaluates to 1 under } \beta \end{array} \right\}, \end{cases}$$

$$\mathsf{k\text{-}minsat}^{\neg\emptyset}_{\mathrm{DH}}\colon \begin{cases}\mathrm{DH}\times\mathbb{N}\to 2^{\Theta}\\ (\chi,k)\mapsto\{\,\beta\in\Theta(\chi)\mid\beta\models\chi,\beta\neq\emptyset\text{ and }|\beta|\leq k\,\},\end{cases}$$

$$\mathsf{anothersolminsat}^{\neg\emptyset}_{\mathrm{DH}}\colon \begin{cases}\mathrm{DH}\times 2^{\Theta}\to 2^{\Theta}\\ (\chi,B)\mapsto\left\{\,\beta\in\Theta(\chi)\;\middle|\;\begin{array}{l}\beta\notin B\text{ and }\beta\text{ is}\\ \text{minimal for }\chi\end{array}\right\},\end{cases}$$

$$\mathsf{anothersolmaxsat}^{\neg\mathrm{full}}_{\mathrm{HORN}}\colon \begin{cases}\mathrm{HORN}\times 2^{\Theta}\to 2^{\Theta}\\ (\chi,B)\mapsto\left\{\,\beta\in\Theta(\chi)\;\middle|\;\begin{array}{l}\beta\notin B\text{ and }\beta\text{ is}\\ \text{maximal for }\chi\end{array}\right\}.\end{cases}$$

Here, minimal solutions in the context of second-order formulas and DualHorn formulas are defined analogously to inclusion minimality in the team logic setting (with the special role of the empty solution). In the context of Horn formulas, maximal means inclusion maximal with respect to the set of all solutions that are not the full assignment, i.e. the set

$$\{\,\beta\in\Theta(\chi)\mid\beta\models\chi,\beta\neq\beta_{\mathrm{full}}\text{ and }\forall\beta'\neq\beta_{\mathrm{full}}\colon\beta\subsetneq\beta'\implies\beta'\not\models\chi\,\}.$$

The second block contains graph CPFs. Before we define those, we have to recall some notions of graph theory: We consider only undirected graphs. Let GRAPH be the set of all (undirected) graphs $G=(V,E)$, VERTEX $=\{V\mid G=(V,E)\in$ GRAPH$\}$ and EDGE $=\{E\mid G=(V,E)\in$ GRAPH$\}$. A *cycle cover* of a graph is a set of cycles such that every vertex of the graph is contained in at least one of the cycles. A cycle cover is vertex-disjoint, if each pair of cycles have no vertex in common. In the following we identify vertex disjoint cycle covers with the sets of its edges. A graph $G=(V,E)$ is *bipartite* if one can split its vertex set into two disjoint independent sets V_1,V_2. We call each such split (V_1,V_2) *bipartition.* We write $G=(V_1,V_2,E)$ to specify the bipartition (V_1,V_2) that was applied to the Graph $G=(V,E)$. With BIP we denote the set of all bipartite graphs. For a given graph $G=(V,E)$ a *matching* is a set of edges E' such that every two edges $e_1,e_2\in E'$ do not share a vertex, that is, $e_1\cap e_2=\emptyset$. A matching E' is called *perfect*, if every vertex $v\in V$ is part of an edge $e\in E'$. Now we have everything we need to define the second block of CPFs. Let $\mathrm{P}\subseteq\Sigma_1\mathrm{BF}$, we define the CPFs

$$\mathsf{is}^{\neg\mathrm{full}}\colon \begin{cases}\mathrm{GRAPH}\times\mathbb{N}\to 2^{\mathrm{VERTEX}}\\ (G,k)\mapsto\{\,V'\subsetneq V\mid\forall u,v\in V'\colon\{\,u,v\,\}\notin E,|V'|\geq k\,\},\end{cases}$$

$$\mathsf{cyclecover}\colon \begin{cases}\mathrm{GRAPH}\to 2^{\mathrm{EDGE}}\\ G\mapsto\{\,E'\subseteq E\mid E'\text{ is a vertex disjoint cycle cover of }G\,\},\end{cases}$$

$$\mathsf{matching}\colon \begin{cases}\mathrm{BIP}\to 2^{\mathrm{EDGE}}\\ G\mapsto\{\,E'\subseteq E\mid E'\text{ is a matching of }G\,\},\end{cases}$$

$$\mathsf{pmatching}\colon \begin{cases} \mathrm{BIP} \to 2^{\mathrm{EDGE}} \\ G \mapsto \{\, E' \subseteq E \mid E' \text{ is a perfect matching of } G \,\}, \end{cases}$$

$$(\mathsf{cyclecover}, \mathsf{sat}_{\mathrm{P}})\colon \begin{cases} \mathrm{GRAPH} \times \mathrm{P} \to 2^{\mathrm{EDGE}} \\ (G,\chi) \mapsto \left\{ E' \subseteq E \,\middle|\, \begin{array}{l} E' \text{ is a vertex disjoint cycle cover} \\ \text{of } G, \mathsf{free}(\chi) = E \text{ and } E' \models \chi \end{array} \right\}, \end{cases}$$

$$(\mathsf{matching}, \mathsf{sat}_{\mathrm{P}})\colon \begin{cases} \mathrm{BIP} \times \mathrm{P} \to 2^{\mathrm{EDGE}} \\ (G,\chi) \mapsto \left\{ E' \subseteq E \,\middle|\, \begin{array}{l} E' \text{ is a matching of} \\ G, \mathsf{free}(\chi) = E \text{ and } E' \models \chi \end{array} \right\}, \end{cases}$$

$$(\mathsf{pmatching}, \mathsf{sat}_{\mathrm{P}})\colon \begin{cases} \mathrm{BIP} \times \mathrm{P} \to 2^{\mathrm{EDGE}} \\ (G,\chi) \mapsto \left\{ E' \subseteq E \,\middle|\, \begin{array}{l} E' \text{ is a perfect matching of} \\ G, \mathsf{free}(\chi) = E \text{ and } E' \models \chi \end{array} \right\}. \end{cases}$$

This concludes the definitions of the considered CPFs. We close this section by recalling some completeness results.

Proposition 2.18.

1. $\mathsf{D\text{-}sat}_{\mathrm{BF}}$ *is* NP*-complete with respect to* $\leq_m^{\mathsf{P}}$ *reductions [GJ79].*
2. $\mathsf{D\text{-}sat}_{\mathrm{HORN}}$ *is* P*-complete with respect to* $\leq_m^{\log}$ *reductions [Kas86, GHR95].*
3. $\mathsf{D\text{-}sat}_{\mathrm{DH}}$ *is* P*-complete with respect to* $\leq_m^{\log}$ *reductions, by Remark 2.1.*

2.6 Decision Problems in Team Logics

In descriptive complexity we study complexity classes which consist of problems that are definable by certain logics. At first glance some description complexity results might seem confusing, since description complexity classes are often denoted by the same symbols as the corresponding logic itself. For example, Fagin's Theorem states $\Sigma_1^1 = \mathsf{NP}$, which does not mean that NP and existential second order logic are the same, but that NP contains exactly the problems definable by Σ_1^1 formulas. To avoid confusion, we now define the decision complexity classes based on the considered logics.

Problem:	$\mathsf{D\text{-}sat}_\psi^{\mathrm{rel}}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Question:	$\{\, R \in \mathrm{REL}(\mathcal{A}, \psi) \mid \mathcal{A}, R \models \psi \,\} \neq \emptyset$?

Definition 2.19. *The complexity class* $\mathsf{FO}^{\mathrm{rel}}$ *consists of all problems* $\mathsf{D\text{-}sat}_\psi^{\mathrm{rel}}$, *where* $\psi \in \mathsf{FO}^{\mathrm{rel}}$.

Definition 2.20. *The complexity class* Σ^1_1 *consists of all problems* $\text{D-}\mathsf{sat}^{\text{rel}}_{\psi}$*, where* $\psi \in \Sigma^1_1$.

Proposition 2.21. $\mathsf{FO}^{\text{rel}} = \Sigma^1_1$.

Proof. The relation $\mathsf{FO}^{\text{rel}} \subseteq \Sigma^1_1$ holds trivially since any FO^{rel} formula is also a Σ^1_1 formula. For the converse let $\text{D-}\mathsf{sat}^{\text{rel}}_{\psi_1} \in \Sigma^1_1$ via $\psi_1 \in \Sigma^1_1$. We assume without loss of generality that $\psi_1(R)$ is of the form $\psi_1(R) = \exists\overline{\text{Q}}\ \psi'_1(R, \overline{\text{Q}})$, where $\psi'_1(R, \overline{\text{Q}})$ is a first-order formula with free relational variables R and $\overline{\text{Q}}$. Now we have

$$\begin{aligned}\text{D-}\mathsf{sat}^{\text{rel}}_{\psi_1} &= \{\,\mathcal{A} \mid \text{there is a relation } R \text{ with } \mathcal{A}, R \models \psi_1(R)\,\}\\ &= \{\,\mathcal{A} \,\big|\, \text{there are relations } R, \overline{\text{Q}} \text{ with } \mathcal{A}, R, \overline{\text{Q}} \models \psi'_1(R, \overline{\text{Q}})\,\}\\ &= \{\,\mathcal{A} \mid \text{there is a relation } R' \text{ with } \mathcal{A}, R' \models \psi_2(R')\,\}\\ &= \text{D-}\mathsf{sat}^{\text{rel}}_{\psi_2},\end{aligned}$$

where ψ_2 is obtained by replacing any relational variable in ψ'_1 by R'. The idea of R' is to capture all relations $R, \overline{\text{Q}}$. The formal definition of R' is straightforward but technical, which is why we omit it. Note that $\psi_2 \in \mathsf{FO}^{\text{rel}}$ and therefore $\text{D-}\mathsf{sat}^{\text{rel}}_{\psi_1} = \text{D-}\mathsf{sat}^{\text{rel}}_{\psi_2} \in \mathsf{FO}^{\text{rel}}$. □

The motivation to define the class FO^{rel} arises from the corresponding counting class $\#\mathsf{FO}^{\text{rel}}$, which was first introduced by Saluja et al. with the aim to capture the counting class $\#\mathsf{P}$ [SST95] (we will revisit the definition of these counting classes and the corresponding result in Section 2.7). Note that (under reasonable assumptions) $\#\mathsf{FO}^{\text{rel}} \subsetneq \#\Sigma^1_1$, where $\#\Sigma^1_1$ is the counting class corresponding to Σ^1_1. We therefore introduced the class FO^{rel} to compare the results from the counting setting to the decision setting.

Problem:	$\text{D-}\mathsf{sat}^{\text{team}}_{\varphi}$
Input:	$\mathcal{A} \in \text{STRUC}$
Question:	$\{\,X \in \text{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset\,\} \neq \emptyset$?

Definition 2.22. *Let* $\text{T} \in \mathfrak{G}$*. The complexity class* $\mathsf{FO}(\text{T})$ *consists of all problems* $\text{D-}\mathsf{sat}^{\text{team}}_{\varphi}$*, where* $\varphi \in \mathsf{FO}(\text{T})$.

There are some other logics that help us to compare the complexity classes we defined via team logics to conventional complexity classes. These include *least fixed point logic* (LFP), *positive greatest fixed point logic* (posGFP) and FOT. We do not go into detail about those logics as this would go beyond the scope of this thesis. However, we stress that these logics match logics we study in terms of expressibility. For more information about them see [GH13] for LFP and posGFP

and [KY19] for FOT. We define the corresponding complexity classes analogously to the complexity classes $\mathsf{FO}^{\mathrm{rel}}$, Σ_1^1 and $\mathsf{FO}(\mathrm{T})$ (see Definitions 2.19, 2.20 and 2.22). The logics LFP and posGFP were thoroughly studied and it was shown that both coincide with P over finite ordered structures [Imm86, Var82]. The logic FOT is a variant of first-order team logic, which was defined with the aim to capture $\mathsf{FO}^{\mathrm{rel}}$ [KY19]. Having defined our logic-based complexity classes, we can show how the classes defined by team logics relate to those defined by predicate logics. This allows us to show that the team logic-based complexity classes capture certain classical complexity classes.

Proposition 2.23 (Fagin's Theorem, [Fag74])**.** *$\Sigma_1^1 = \mathsf{NP}$.*

For dependence and independence logic there is a connection to existential second-order logic which we can use to translate Fagin's Theorem to those logics.

Proposition 2.24 ([Gal12])**.** *For every σ-formula $\varphi(x_1,\dots,x_k)$ of $\mathsf{FO}(\perp)$, there is a σ-formula $\psi(R)$ of Σ_1^1 such that for all σ-structures $\mathcal{A}$ and teams X over $\{x_1,\dots,x_k\}$,*

$$\mathcal{A} \models_X \varphi \iff \mathcal{A}, \mathrm{rel}(X) \models \psi(R). \tag{$\star$}$$

Conversely, for every σ-formula $\psi(R)$ of Σ_1^1, there is a σ-formula φ of $\mathsf{FO}(\perp)$ such that $(\star)$ holds for all σ-structures $\mathcal{A}$ and non-empty teams X.

Proposition 2.25 ([KV09])**.** *For every σ-formula $\varphi(x_1,\dots,x_k)$ of $\mathsf{FO}(=(\dots))$, there is a σ-formula $\psi(R)$ of Σ_1^1 with R occurring only negatively such that for all σ-structures $\mathcal{A}$ and teams X over $\{x_1,\dots,x_k\}$,*

$$\mathcal{A} \models_X \varphi \iff \mathcal{A}, \mathrm{rel}(X) \models \psi(R). \tag{$\star$}$$

Conversely, for every σ-formula $\psi(R)$ of Σ_1^1 with R only occurring negatively, there is a σ-formula φ of $\mathsf{FO}(=(\dots))$ such that $(\star)$ holds for all σ-structures $\mathcal{A}$ and non-empty teams X.

To get a better understanding for how such translations work, we present an example for the case of dependence logic.

Example 2.26. *Let $\varphi :=\ =(x,y) \wedge z = y$ be a dependence logic formula. Consider the following Σ_1^1 formula*

$$\psi := \forall x_1 \forall y_1 \forall x_2 \forall y_2 \forall z \Bigg(\overbrace{\Big(\big(R(x_1,y_1,z) \wedge R(x_2,y_2,z) \wedge x_1 = x_2 \big) \to y_1 = y_2 \Big)}^{\text{expresses } =(x,y)} \\ \wedge \underbrace{\big(R(x_1,y_1,z) \to z = y_1 \big)}_{\text{expresses } z=y} \Bigg)$$

Note that R only appears in the left side of an implication and therefore only negatively. Now we have for all structures $\mathcal{A}$ and teams X

$$\mathcal{A} \models_X \varphi \iff \mathcal{A}, \mathrm{rel}(X) \models \psi(R).$$

For the other direction let $\psi' := \forall x \exists y \forall z \neg R(x,y,z) \vee y = z$. To construct the corresponding dependence logic formula we first translate ψ' to Skolem normal form:

$$\psi'_{\text{Skolem}} := \exists f \forall x \forall z\ \neg R(x, f(x), z)) \vee f(x) = z.$$

Note that we have introduced second-order logic without functional variables, hence the formula ψ'_{Skolem} is not a second-order formula by our definition. Nevertheless any second-order formula (by our definition) is equivalent to a formula in Skolem normal form [Vää07]. The corresponding dependence logic formula is defined as follows

$$\varphi'(a,b,c) := \forall x \forall z \exists y \big(\overbrace{=(x,y)}^{\text{expresses } f(x)=y} \wedge (\overbrace{a \neq x \vee b \neq y \vee c \neq z}^{\text{expresses } \neg R(x,y,z)} \vee y = z)\big)$$

Now we have for all structures $\mathcal{A}$ and teams $X \neq \emptyset$

$$\mathcal{A} \models_X \varphi' \iff \mathcal{A}, \mathrm{rel}(X) \models \psi'(R).$$

Corollary 2.27 ([Gal12, KV09])**.** $\mathsf{FO}(=(\dots)) = \mathsf{FO}(\perp) = \mathsf{NP}$.

Proof. By Propositions 2.24 and 2.25, over sentences both $\mathsf{FO}(\perp)$ and $\mathsf{FO}(=(\dots))$ are expressively equivalent to Σ^1_1: Every σ-sentence φ of $\mathsf{FO}(\perp)$ (or $\mathsf{FO}(=(\dots))$) is equivalent to a σ-sentence ψ of Σ^1_1, i.e., for any σ-structure $\mathcal{A}$,

$$\mathcal{A} \models \varphi \iff \mathcal{A} \models \psi,$$

and vice versa. Combining this with Fagin's Theorem 2.23 we can conclude:

$$\mathsf{FO}(=(\dots)) = \mathsf{FO}(\perp) = \Sigma^1_1 = \mathsf{NP}. \qquad \square$$

Amongst others, Corollary 2.27 states that $\mathsf{FO}(=(\dots)) = \mathsf{FO}(\perp)$, even though independence logic has a higher expressive power than dependence logic. The reason for this is very similar to the reason why $\mathsf{FO}^{\text{rel}} = \Sigma^1_1$ holds: Any decision problem that can be described by an $\mathsf{FO}(\perp)$ formula, can also be described by an $\mathsf{FO}(\perp)$ sentence. Furthermore, any such problem can be described by an $\mathsf{FO}(=(\dots))$ sentence, since over sentences independence logic and dependence logic are expressively equivalent. Note that the CPFs that are described by such a formula and a corresponding sentence are not equivalent. This is due to the reason that the solution set of the CPF defined by the sentence can only be the empty set or consist of the team $\{s_\emptyset\}$. For this reason the corresponding counting classes do not coincide, as we will later see.

Proposition 2.28 ([KY19])**.** *For any σ-formula $\varphi(x_1, \dots, x_k) \in \mathsf{FOT}$, there is a σ-formula $\psi(R)$ of $\mathsf{FO}^{\mathrm{rel}}$ such that for any σ-structure $\mathcal{A}$ and any team X over $\{x_1, \dots, x_k\}$,*

$$\mathcal{A} \models_X \varphi \iff \mathcal{A}, \mathrm{rel}(X) \models \psi(R) \qquad (\star)$$

Conversely, for $\sigma(R)$-formula $\psi(R)$ of $\mathsf{FO}^{\mathrm{rel}}$, there is a σ-formula φ of FOT such that $(\star)$ holds for all σ-structures $\mathcal{A}$ and non-empty teams X.

Corollary 2.29 ([KY19])**.** $\mathsf{FOT} = \mathsf{NP}$.

Proof. By Proposition 2.28 follows that $\mathsf{FOT} = \mathsf{FO}^{\mathrm{rel}}$. Furthermore by Propositions 2.21 and 2.23 we can conclude $\mathsf{FOT} = \Sigma_1^1 = \mathsf{NP}$. □

For inclusion logic there is a connection to posGFP and therefore to LFP, which can be use to show $\mathsf{FO}(\subseteq) = \mathsf{P}$.

Proposition 2.30 ([GH13])**.** *For any σ-formula $\varphi(x_1, \dots, x_k)$ of $\mathsf{FO}(\subseteq)$, there exists a σ-formula $\psi(R)$ of posGFP such that for all σ-structures $\mathcal{A}$ and teams X over $\{x_1, \dots, x_k\}$,*

$$\mathcal{A} \models_X \varphi \iff \mathcal{A}, \mathrm{rel}(X) \models \psi(R) \qquad (\star)$$

Conversely, for every σ-formula $\psi(R)$ of posGFP, there is a σ-formula φ of $\mathsf{FO}(\subseteq)$ such that $(\star)$ holds for all σ-structures $\mathcal{A}$ and non-empty teams X.

The result from Proposition 2.30 allows us to conclude $\mathsf{FO}(\subseteq) = \mathsf{P}$ over ordered structures. Here, "over ordered structures" means that any problem D-$\mathsf{sat}_\varphi^{\mathrm{team}} \in \mathsf{FO}(\subseteq)$ is defined over a vocabulary σ with $\leq\, \in \sigma$ and thereby that any input structure $\mathcal{A}$ for D-$\mathsf{sat}_\varphi^{\mathrm{team}}$ interprets the relation $\leq$.

Corollary 2.31 ([GH13])**.** *Over ordered structures we have* $\mathsf{FO}(\subseteq) = \mathsf{P}$.

Proof. By Proposition 2.30, over sentences $\mathsf{FO}(\subseteq)$ is expressively equivalent to posGFP. Since $\mathsf{posGFP} = \mathsf{LFP}$ (see [Imm86]), $\mathsf{FO}(\subseteq)$ is expressively equivalent to LFP over finite structures. Furthermore $\mathsf{LFP} = \mathsf{P}$ over ordered finite structures [Imm86, Var82], thus $\mathsf{FO}(\subseteq)$ captures P. □

There is also a connection between inclusion logic and myopic formulas, which will be useful later.

Proposition 2.32 ([GH13])**.** *Let $\psi(R)$ be a myopic σ-formula. Then there exists a σ-formula $\varphi(x_1, \dots, x_k) \in \mathsf{FO}(\subseteq)$ such that for all σ-structures $\mathcal{A}$ and all teams X over $\{x_1, \dots, x_k\}$ we have*

$$\mathcal{A} \models_X \varphi(\overline{x}) \iff \mathcal{A}, \mathrm{rel}(X) \models \psi(R).$$

In Figure 2.1, we summarise the relationships between the classes defined in this section.

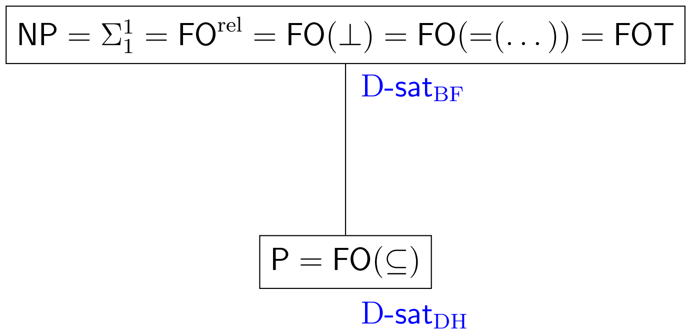
$\mathsf{NP} = \Sigma_1^1 = \mathsf{FO}^{\mathrm{rel}} = \mathsf{FO}(\perp) = \mathsf{FO}({=}(\dots)) = \mathsf{FOT}$
$\mathrm{D}\text{-}\mathsf{sat}_{\mathrm{BF}}$
$\mathsf{P} = \mathsf{FO}(\subseteq)$
$\mathrm{D}\text{-}\mathsf{sat}_{\mathrm{DH}}$

Figure 2.1: Class diagram of considered decision complexity classes. The edge denotes a subset relationship from bottom to top between its nodes. The problems marked in blue next to the classes are complete for the respective class (see Proposition 2.18).

2.7 Counting Complexity

In this section, we define counting complexity classes as well as the corresponding reducibility notions. We start with the conventional counting classes and get to the logic based classes afterwards.

Definition 2.33. *A counting problem* C-f *is in* FP*, if there is a (deterministic) polynomial-time Turing machine M such that, given input $x \in \{0,1\}^*$, M computes* C-$f(x)$*.*

Definition 2.34. *A counting problem* C-f *is in* #P *if there is a non-deterministic polynomial-time Turing machine M such that for all inputs $x \in \{0,1\}^*$,*

$$\text{C-}f(x) \textit{ is the number of accepting computation paths of } M \textit{ on input } x.$$

Definition 2.35. *A counting problem* C-f *is in* TotP *if there is a non-deterministic polynomial-time Turing machine M such that for all inputs $x \in \{0,1\}^*$,*

$$\text{C-}f(x) \textit{ is the number of computation paths of } M \textit{ on input } x \textit{ minus } 1.$$

The purpose of subtracting the value 1 is to allow TotP problems to map to the value 0.

Definition 2.36. *Let* C *be a decision complexity class. A counting problem* C-f *is said to be in* #·C *if there are* D-$g \in$ C *and a polynomial p such that for all $x \in \{0,1\}^*$:*

$$\text{C-}f(x) = |\{y \mid |y| \le p(|x|) \textit{ and } (x,y) \in \text{D-}g\}|.$$

In Definition 2.36, we defined #·C for arbitrary classes C, however in this thesis we are only really interested in one of these classes, namely #·NP. The class

$\#\cdot\mathsf{NP}$ is not to be confused with $\#\mathsf{NP} := \#\mathsf{P}^{\mathsf{NP}}$, which is defined by extending the definition of $\#\mathsf{P}$ by giving the machine M access to an NP oracle [Val79a]. It is straightforward to show $\#\cdot\mathsf{NP} \subseteq \#\mathsf{NP}$ but the converse does (most likely) not hold, since this would imply $\mathsf{NP} = \mathsf{coNP}$ [HV95].

In the next proposition we recall subset relationships between the counting classes we defined in this section so far. We consider all but one of those results folklore, hence we do not provide a reference for those results but give proof sketches instead.

Proposition 2.37 ([PZ06])**.** $\mathsf{FP} \subseteq \mathsf{TotP} \subseteq \#\mathsf{P} = \#\cdot\mathsf{P} \subseteq \#\cdot\mathsf{NP}$.

Proof. $\mathsf{FP} \subseteq \mathsf{TotP}$: Let $\text{C-}f \in \mathsf{FP}$. Then there is a machine M that on input x computes $\text{C-}f(x)$ in polynomial time. Now the following machine witnesses $\text{C-}f \in \mathsf{TotP}$: On input x compute $\text{C-}f(x)$ in polynomial time by simulating M. Afterwards create $\text{C-}f(x)$ (accepting or rejecting) branches.

$\mathsf{TotP} \subseteq \#\mathsf{P}$: It was shown that TotP is the closure with respect to parsimonious reductions of self-reducible counting problems from $\#\mathsf{P}$ whose decision version is in P [PZ06]. It follows that $\mathsf{TotP} \subsetneq \#\mathsf{P}$ unless $\mathsf{P} = \mathsf{NP}$.

$\#\mathsf{P} \subseteq \#\cdot\mathsf{P}$: Let $\text{C-}f \in \#\mathsf{P}$. This means that there is a non-deterministic polynomial-time Turing machine M such that $\text{C-}f(x)$ is the number of accepting paths of M on input x. Let p be a polynomial such that the longest computation path of M on input x is bounded by $p(|x|)$. The set of accepting computation paths of M on input x can be described as

$$\{y \mid y \leq p(|x|) \text{ and } (x,y) \in \text{D-}g\},$$

where $(x,y) \in \text{D-}g \iff$ M accepts x on computation path y. Note that D-g can be computed in polynomial time by simulating M on input x on path y. Now for all $x \in \{0,1\}^*$ we have

$$\begin{aligned} \text{C-}f(x) &= |\{y \mid \text{M accepts input } x \text{ on computation path } y\}| \\ &= |\{y \mid y \leq p(|x|) \text{ and } (x,y) \in \text{D-}g\}|. \end{aligned}$$

Hence $\text{C-}f \in \#\cdot\mathsf{P}$.

$\#\cdot\mathsf{P} \subseteq \#\mathsf{P}$: Let $\text{C-}f \in \#\cdot\mathsf{P}$. By definition there are $\text{D-}g \in \mathsf{P}$ and polynomial p such that

$$\text{C-}f(x) = |\{y \mid |y| \leq p(|x|) \text{ and } (x,y) \in \text{D-}g\}|.$$

We describe a machine M that witnesses $\text{C-}f \in \#\mathsf{P}$: Branch on all $y \leq p(|x|)$. At the end of each branch accept if and only if $(x,y) \in \text{D-}g$.

$\#\cdot\mathsf{P} \subseteq \#\cdot\mathsf{NP}$: This follows from $\mathsf{P} \subseteq \mathsf{NP}$. □

Next, we introduce two reducibility notions: On one hand we have *parsimonious reductions* under which the classes defined above are closed (see Lemma 2.39) and on the other hand we have *Turing reductions* under which these classes are (probably) not closed.

Definition 2.38. *Let* C-f_1, C-f_2 *be two counting problems. We say* C-f_1 *is* (polynomial-time) parsimonious reducible *to* C-f_2, C-$f_1 \leq^{\mathsf{P}}_{\text{par}}$ C-f_2, *if there is a function* $g\colon \{0,1\}^* \to \{0,1\}^*$ *such that*

1. C-$f_1(x) =$ C-$f_2(g(x))$ *for every* x *and*
2. g *can be computed in polynomial time.*

In the next lemma we examine $\leq^{\mathsf{P}}_{\text{par}}$ closure properties of certain counting classes. Again, we consider those results folklore and therefore provide only proof sketches.

Lemma 2.39. *The classes* $\#\mathsf{P}, \mathsf{TotP}$ *and* $\#\cdot\Sigma^{\mathsf{P}}_k$ *for* $k \geq 0$ *are closed under parsimonious reductions.*

Proof. We start by showing the closure under parsimonious reductions for $\#\mathsf{P}$. Let C-f_1, C-f_2 be two counting problems with C-$f_1 \leq^{\mathsf{P}}_{\text{par}}$ C-f_2 and C-$f_2 \in \#\mathsf{P}$. Since C-$f_1 \leq^{\mathsf{P}}_{\text{par}}$ C-f_2, there is a polynomial-time computable function g such that

$$\text{C-}f_1(x) = \text{C-}f_2(g(x)) \tag{1}$$

and since C-$f_2 \in \#\mathsf{P}$ there is a non-deterministic polynomial-time Turing machine M such that

$$\text{C-}f_2(x) = |\{y \mid \text{M accepts input } x \text{ on computation path } y\}|. \tag{2}$$

Combining (1) and (2) we conclude:

$$\begin{aligned}\text{C-}f_1(x) &= |\{y \mid \text{M accepts input } g(x) \text{ on computation path } y\}| \\ &= |\{y \mid \text{M}' \text{ accepts input } x \text{ on computation path } y\}|\end{aligned}$$

and therefore C-$f_1(x) \in \#\mathsf{P}$. Here, M$'$ is the Turing machine that first computes $g(x)$ and then acts like machine M on input $g(x)$.

We are very brief in the proofs for TotP and $\#\cdot\mathsf{C}$ since they work analogously to the case of $\#\mathsf{P}$.

Let C-$f_1 \leq^{\mathsf{P}}_{\text{par}}$ C-f_2 and C-$f_2 \in \mathsf{TotP}$. It holds that

$$\begin{aligned}\text{C-}f_1(x) &= \text{C-}f_2(g(x)) \\ &= |\{y \mid y \text{ encodes a computation path of M on input } g(x)\}| - 1 \\ &= |\{y \mid y \text{ encodes a computation path of M' on input } x\}| - 1\end{aligned}$$

Let $\text{C-}f_1 \leq^{\mathsf{P}}_{\text{par}} \text{C-}f_2$ and $\text{C-}f_2 \in \#\cdot\Sigma^{\mathsf{P}}_k$. This means that there is a $\text{D-}h \in \Sigma^{\mathsf{P}}_k$ such that

$$\begin{aligned}\text{C-}f_1(x) &= \text{C-}f_2(g(x)) \\ &= |\{y \mid |y| \leq p(|g(x)|) \text{ and } (g(x), y) \in \text{D-}h\}| \\ &= |\{y \mid |y| \leq p'(|x|) \text{ and } (x, y) \in \text{D-}h'\}|,\end{aligned}$$

where $p'(|x|) = p(|g(x)|)$ and $(x, y) \in \text{D-}h' \iff (g(x), y) \in \text{D-}h$. Note that $\text{D-}h' \in \Sigma^{\mathsf{P}}_k$, since $\text{D-}h \in \Sigma^{\mathsf{P}}_k$ and $\mathsf{P} \subseteq \Sigma^{\mathsf{P}}_k$. □

For the second reducibility notion—Turing reducibility—recall the definition of oracle Turing machines from Section 2.5.

Definition 2.40. *Let* $\text{C-}f_1, \text{C-}f_2$ *be two counting problems. We say* $\text{C-}f_1$ *is* (polynomial-time) Turing reducible *to* $\text{C-}f_2$ *(denoted by* $\text{C-}f_1 \leq^{\mathsf{P}}_{\text{T}} \text{C-}f_2$*), if* $\text{C-}f_1$ *can be computed by a Turing machine* $M^{\text{C-}f_2}$ *in polynomial time.*

Since the considered counting classes of this thesis are not closed under Turing reductions, we can not use this kind of reduction to show subset relations between those classes. However, we can use Turing reductions to compare FP^{C} to FP^{D}, for counting classes C, D: Suppose we have two counting problems $\text{C-}f_1 \in \mathsf{C}, \text{C-}f_2 \in \mathsf{D}$ with $\text{C-}f_1 \leq^{\mathsf{P}}_{\text{T}} \text{C-}f_2$. Furthermore, let $\text{C-}f_1$ be complete for C with respect to Turing reductions. Then, we can conclude $\mathsf{FP}^{\mathsf{C}} = \mathsf{FP}^{\text{C-}f_1} \subseteq \mathsf{FP}^{\text{C-}f_2} \subseteq \mathsf{FP}^{\mathsf{D}}$.

In the following result we recall one $\#\mathsf{P}$-complete and one TotP-complete problem, to have a representative problems for both classes.

Proposition 2.41. *[Pap94, PZ06]*

1. $\text{C-}\mathsf{sat}_{\text{BF}}$ *is* $\#\mathsf{P}$*-complete with respect to* $\leq^{\mathsf{P}}_{\text{par}}$ *reductions.*
2. $\text{C-}\mathsf{sat}_{\text{DNF}}$ *is* TotP*-complete with respect to* $\leq^{\mathsf{P}}_{\text{T}}$ *reductions.*

Here with DNF we denote the class of propositional formulas in disjunctive normal form, that is, $\text{DNF} := \{\varphi \in \text{BF} \mid \varphi = \bigvee_i \bigwedge_j \ell_{i,j}\}$. Note that we do not know whether or not $\text{C-}\mathsf{sat}_{\text{DNF}}$ is TotP-complete with respect to $\leq^{\mathsf{P}}_{\text{par}}$ reductions. As a matter of fact there is no problem known to be TotP-complete with respect to $\leq^{\mathsf{P}}_{\text{par}}$ reductions. For $\#\cdot\mathsf{NP}$, we will identify complete problems later in Section 3.4. For FP we have that any problem $\text{C-}f \in \mathsf{FP}$ is trivially also complete with respect to $\leq^{\mathsf{P}}_{\text{par}}$ reductions.

We already mentioned that for inclusion logic formulas there is always an unambiguous maximal (satisfying) team. In fact, we can even compute this team in polynomial time, as the next result states. For this, recall the problem $\text{S-}\mathsf{maxsubteam}_{\varphi}$.

Problem:	$\textsf{S-maxsubteam}_\varphi$
Input:	$\mathcal{A} \in \text{STRUC}, X \in \text{TEAM}(\varphi)$
Output:	$y \in \left\{ X' \in \text{TEAM}(\mathcal{A},\varphi) \,\middle|\, \begin{array}{l} \mathcal{A} \models_{X'} \varphi,\ X' \subseteq X \text{ and} \\ \forall X'' \subseteq X \colon \lvert X''\rvert > \lvert X'\rvert \\ \implies \mathcal{A} \not\models_{X''} \varphi \end{array} \right\}$

Proposition 2.42 ([Grä16])**.** $\textsf{S-maxsubteam}_\varphi \in \textsf{FP}$ *for any* $\varphi \in \textsf{FO}(\subseteq)$*.*

Now we turn to logic-based counting complexity classes. We define one counterpart to each of the logic-based decision complexity classes from Section 2.6.

Problem:	$\textsf{C-sat}^{\text{rel}}_\psi$
Input:	$\mathcal{A} \in \text{STRUC}$
Question:	$\lvert \{ R \in \text{REL}(\mathcal{A},\psi) \mid \mathcal{A}, R \models \psi \} \rvert$

Definition 2.43. *The counting class* $\#\textsf{FO}^{\text{rel}}$ *consists of all functions* $\textsf{C-sat}^{\text{rel}}_\psi$*, where* $\psi \in \textsf{FO}^{\text{rel}}(\sigma)$ *and* σ *is a vocabulary with* $\leq \in \sigma$*.*

As mentioned before the class $\#\textsf{FO}^{\text{rel}}$ was first introduced by Saluja et al. [SST95]. There is a difference between our definition and the one of Saluja: By our definition the formulas may only contain one free relational variable, whereas Saluja's definition allows an arbitrary number of free relational variables and individual variables. We stress that both definitions are equivalent as one can convert a formula $\psi(R_1,\dots,R_k,x_1,\dots,x_\ell)$ to a formula $\psi'(R)$ such that the number of solutions is preserved for any structure $\mathcal{A}$.

Definition 2.44. *The counting class* $\#\Sigma^1_1$ *consists of all functions* $\textsf{C-sat}^{\text{rel}}_\psi$*, where* $\psi \in \Sigma^1_1(\sigma)$ *and* σ *is a vocabulary with* $\{ \leq, +, \times \} \subseteq \sigma$*.*

Problem:	$\textsf{C-sat}^{\text{team}}_\varphi$
Input:	$\mathcal{A} \in \text{STRUC}$
Output:	$\lvert \{ X \in \text{TEAM}(\mathcal{A},\varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \} \rvert$

Definition 2.45. *Let* $\mathrm{T} \in \mathfrak{G}$*. The counting class* $\#\textsf{FO}(\mathrm{T})$ *consists of all functions* $\textsf{C-sat}^{\text{team}}_\varphi$*, where* $\varphi \in \textsf{FO}(\sigma,\mathrm{T})$ *and* σ *is a vocabulary with* $\{ \leq, +, \times \} \subseteq \sigma$*.*

As we see in the next two results, there are already characterisations of $\#\textsf{P}$ in the context of first-order logic and team logic.

Proposition 2.46 ([SST95])**.** $\#\textsf{FO}^{\text{rel}} = \#\textsf{P}$.

In Section 2.6 we mentioned that there is a logic, FOT, and its corresponding descriptive complexity class captures NP. We introduce its counting counterpart $\#\mathsf{FOT}$ as the set of problems C-$\mathsf{sat}^{\text{team}}_{\varphi}$, where $\varphi \in \mathsf{FOT}(\sigma)$ and σ is a vocabulary with $\leq \in \sigma$.

Corollary 2.47 ([HKM+19])**.** $\#\mathsf{FOT} = \#\mathsf{P}$.

Proof. Since there is a one-to-one correspondence between satisfying teams for FOT formulas and satisfying relations for the corresponding first-order formulas in the sense of Proposition 2.28, we can conclude $\#\mathsf{FOT} = \#\mathsf{FO}^{\text{rel}}$. Furthermore, $\#\mathsf{FO}^{\text{rel}} = \#\mathsf{P}$ (see Proposition 2.46). Hence we can conclude $\#\mathsf{FOT} = \#\mathsf{P}$. □

As our reducibility notion in the context of counting complexity classes defined via team logics, we choose (parsimonious) first-order reducibility. As we will later see in Theorem 3.1, all classes $\#\mathsf{FO}(\mathrm{T})$, for $\mathrm{T} \in \{=(\dots), \perp, \subseteq\}$, are closed under this type of reduction.

For the definition of first-order reductions we first need to introduce first-order queries. Let σ_1, σ_2 be two vocabularies with $\sigma_2 = (R_1^{i_1}, \dots, R_k^{i_k})$. An FO-*query* is a function $I\colon \mathrm{STRUC}(\sigma_1) \to \mathrm{STRUC}(\sigma_2)$ represented by the tuple $I = (\psi_0, \dots, \psi_k)$ of first-order σ_1-formulas, where for a structure $\mathcal{A} \in \mathrm{STRUC}(\sigma_1)$ there is $\ell \in \mathbb{N}$ such that

$$\begin{aligned} I(\mathcal{A}) &= (\mathrm{dom}(I(\mathcal{A})), R_1^{I(\mathcal{A})}, \dots, R_k^{I(\mathcal{A})}), \\ \mathrm{dom}(I(\mathcal{A})) &= \{\, (a_1, \dots, a_\ell) \mid \mathcal{A} \models \psi_0(a_1, \dots, a_\ell) \,\} \text{ and} \\ R_j^{I(\mathcal{A})} &= \left\{ (a_1^1, \dots, a_1^\ell, \dots, a_{i_j}^1, \dots, a_{i_j}^\ell) \;\middle|\; \mathcal{A} \models \psi_i(a_1^1, \dots, a_1^\ell, \dots, a_{i_j}^1, \dots, a_{i_j}^\ell) \right\} \end{aligned}$$

for all $j \leq k$. Intuitively I simply maps ℓ-ary tuples of elements of $\mathcal{A}$ to elements of $I(\mathcal{A})$. Since we use FO-queries in the context of team logics we also need to define what it means to apply an FO-query to a team X or an assignment s. Let $s\colon \{x_1^1, \dots, x_\ell^1, \dots, x_1^m, \dots, x_\ell^m\} \to \mathrm{dom}(\mathcal{A})$ be an assignment with $(s(x)_1^j, \dots, s(x)_\ell^j) \in \mathrm{dom}(I(\mathcal{A}))$ for $1 \leq j \leq m$, we define $I(s)((x_1^j, \dots, x_\ell^j)) = (s(x_1^j), \dots, s(x_\ell^j))$. For a team $X = \{s_1, \dots, s_n\}$, we define $I(X) = \{I(s) \mid s \in X\}$.

Definition 2.48. *Let* C-f_1, C-f_2 *be counting problems. We say* C-f_1 *is first-order reducible or* FO*-reducible to* C-f_2 *(denoted by* C-$f_1 \leq^{\mathsf{FO}}_{\text{par}}$ C-f_2*) if there are vocabularies* σ_1, σ_2 *and an* FO*-query* $I\colon \mathrm{STRUC}(\sigma_1) \to \mathrm{STRUC}(\sigma_2)$ *such that for all* σ_1*-structures* $\mathcal{A}$

$$\text{C-}f_1(\mathcal{A}) = \text{C-}f_2(I(\mathcal{A})).$$

2.8 Enumeration Complexity

By the definition of the considered CPFs, the solution set to an instance might contain an exponential number of solutions (compared to the size of the instance).

Hence, a machine enumerating such a solution set might be required to have an exponential running time. Furthermore, the space requirement of the machine might be exponential as well. This means that our classical complexity measures are not suitable for the enumeration setting and we have to define new ones.

Rather than analysing the runtime of the whole computation, we examine the *delay*, which is an upper bound for the time of the computation before the first solution is output (*precomputation*), the time between the outputs of each two consecutive solutions and the time between the output of the last solution and the termination of the algorithm (*postcomputation*).

Instead of Turing machines, we will use *random access machines (RAMs)* to be able to access the (potentially) exponential "memory" in polynomial time. For a formal introduction to RAMs we refer to the PhD Thesis of Strozecki [Str10] and the book "Computational Complexity: A Modern Approach" [AB09]. Here, we will not give a formal definition of RAMs, but instead give an intuition of how they work using the concept of Turing machines.

One can imagine a RAM as a Turing machine that can jump to certain cells in one time step. For this, the machine has one special index tape, where it can write down the index of the cell it wants to jump to. This is the same concept that is used to define log-time complexity classes [RV97]. Analogously to the definition of (decision) oracle Turing machines we define *(decision) oracle random access machines (ORAMs)* and write $\mathrm{M}^{\text{D-}f}$ to denote that we give the ORAM M access to the oracle D-f.

We say a RAM enumerates a problem E-f, if on input $x \in \Sigma$ it outputs all solutions $y \in f(x)$ without duplicates (in any order).

Definition 2.49. *The enumeration complexity class* DelP *contains all enumeration problems* E-f*, for which there is a RAM M and polynomial p such that for all inputs x, M enumerates the output set of* E-f *with delay* $p(|x|)$.

Definition 2.50. *Let* C *be a decision complexity class. The enumeration complexity class* DelC *consists of all enumeration problems* E-f*, for which there exists a RAM* $M^{\text{D-}g}$*, with* D-$g \in \mathsf{C}$ *and p be a polynomial such that for all inputs x, M enumerates the output set of* E-f *with* $p(|x|)$ *delay and all oracle queries are bounded by* $p(|x|)$.

The class Del^+C *is defined analogously without the bound on the size of the oracle queries.*

Example 2.51. *We show* E-$\mathsf{sat}_{\mathrm{BF}} \in \mathsf{DelNP}$.

Problem:	E-$\mathsf{sat}_{\mathrm{BF}}$
Input:	$\chi \in \mathrm{BF}$
Output:	$\{\beta \in \Theta(\chi) \mid \beta \models \chi\}$

Let χ be an input formula over the variables $x_1, \ldots, x_n$. To enumerate all satisfying assignments of χ, we assign the value 0 to the smallest variables that is not assigned yet and ask the oracle if resulting formula is still satisfiable. If the answer is "yes", we continue with the next variable, if it is "no" we stop looking for extensions of the current assignment. Afterwards we repeat the same for the value 1. At any point, where all variables are assigned we output the current assignment.

We now have to argue, that this method has polynomial delay, the oracles queries are polynomially bounded and that the oracle is in NP*. The last one is the easiest, since we all know by Cook's Theorem that* $\text{D-}\mathsf{sat}_{\mathrm{BF}} \in \mathsf{NP}$ *(see Proposition 2.18). The oracles queries have the same length as the input formula, therefore they are polynomial bounded. To get from one satisfying assignment to another we have to go up and down the whole tree of assignments once in the worst case, which takes polynomial time.*

Since the answer of the oracle from Example 2.51 gives us a glance at the whole tree of solutions, this kind of enumeration algorithm is often called *flashlight* or *torchlight search*. Our algorithms in Section 4 showing membership for DelP and DelNP will be based on torchlight search. We therefore call the corresponding oracles *torchlight* in such algorithms.

Definition 2.50 yields a second definition of DelP, which coincides with our first one, since the answers to oracle queries to an oracle $\text{D-}f \in \mathsf{P}$ can simply be computed in polynomial time [JPY88].

To determine the complexity of enumeration problems precisely we need a suitable definition of reducibility. The reductions we use are quite similar to Turing reductions. For this, we give a RAM access to an enumeration oracle.

An *enumeration oracle machine (EOM) with access to oracle* $\text{E-}f$ is defined similar to an ORAM but with an enumeration oracle instead of a decision oracle and the following difference: When the machine asks the oracle for an answer to a query x, the oracle outputs only one solution y from the solution set. The machine can then ask the oracle again and the answer is any solution $y' \neq y$. In that manner, as long as there are solutions to x that have not been output by the oracle before, the oracle answers queries to x with one of them. On the other hand, when all solutions have been given by the oracle it answers with the special symbol $\perp$ to the query x. The oracle may give its answers in any order. As for the other oracle machines we defined previously we write $\mathrm{M}^{\text{E-}f}$ to denote that we give the EOM M access to the oracle $\text{E-}f$. We say an EOM is *oracle-bounded* if the size of all queries is at most polynomial in the size of the input.

Definition 2.52. *Let* $\text{E-}f_1, \text{E-}f_2$ *be enumeration problems. We say that* $\text{E-}f_1$ *reduces to* $\text{E-}f_2$ *via* DelP *reductions,* $\text{E-}f_1 \leq_{\mathsf{DelP}} \text{E-}f_2$*, if there is an oracle-bounded EOM* $M^{\text{E-}f_2}$ *that enumerates* $\text{E-}f_1$ *with polynomial delay and independently of the order in which the* $\text{E-}f_2$*-oracle gives its answers.*

With DelP reductions we are using a quite strong reducibility notion in the enumeration setting. Fortunately, the classes we consider are still closed under this reducibility notion.

Proposition 2.53 ([CKP$^+$19]). *The class* $\mathsf{Del}\Sigma_k^{\mathsf{P}}$ *is closed under* DelP *reductions for any* $k \geq 0$.

By the definition of our complexity classes via decision oracles and with the power of our reducibility notion, we get a connection to the decision case that allows us to translate hardness results.

Theorem 2.54 ([HMMV22]). *Let* $k \geq 0, \text{D-}f$ *be a* Σ_k^{P}*-hard decision problem (with respect to* $\leq_m^{\mathsf{P}}$*) and* $\text{E-}g$ *be an enumeration problem such that* $\text{D-}f$ *can be decided by an EOM* $M^{\text{E-}g}$ *in polynomial time. Then it holds that* $\text{E-}g$ *is* $\mathsf{Del}\Sigma_k^{\mathsf{P}}$*-hard under* DelP *reductions.*

Proof. We show that for every $\text{E-}h \in \mathsf{Del}\Sigma_k^{\mathsf{P}}$ it holds that $\text{E-}h \leq_{\mathsf{DelP}} \text{E-}g$. For this, let $\text{E-}h \in \mathsf{Del}\Sigma_k^{\mathsf{P}}$. Then

there are $\text{D-}q \in \Sigma_k^{\mathsf{P}}$ and ORAM $\mathrm{M}_1^{\text{D-}q}$ that enumerates $\text{E-}h$ with polynomial delay

$\overset{(1)}{\Longrightarrow}$ there is an ORAM $\mathrm{M}_2^{\text{D-}f}$ that enumerates $\text{E-}h$ with polynomial delay

$\overset{(2)}{\Longrightarrow}$ there is an EOM $\mathrm{M}_3^{\text{E-}g}$ that enumerates $\text{E-}h$ with polynomial delay

$\Longleftrightarrow$ $\text{E-}h \leq_{\mathsf{DelP}} \text{E-}g$.

Since $\text{D-}f$ is Σ_k^{P}-hard, implication (1) holds: The machine M_2 can simulate M_1 but instead of asking the oracle $\text{D-}q$ it translates the oracle queries via the reduction function that witnesses $\text{D-}q \leq_m^{\mathsf{P}} \text{D-}f$ and asks $\text{D-}f$.

Furthermore, implication (2) holds, since $\text{D-}f$ can be decided in polynomial time by an EOM $\mathrm{M}^{\text{E-}g}$: The machine M_3 can simulate M_2 but whenever M_2 would ask the oracle $\text{D-}f$, M_3 instead simulates $\mathrm{M}^{\text{E-}g}$. □

The next corollary covers a special case of Theorem 2.54, which relates an enumeration problem $\text{E-}f$ to its decision counterpart $\text{D-}f$.

Corollary 2.55. *Let* $\text{E-}f$ *be an enumeration problem and* $k \geq 0$ *such that* $\text{D-}f$ *is* Σ_k^{P}*-hard (with respect to* $\leq_m^{\mathsf{P}}$*). Then we have that* $\text{E-}f$ *is* $\mathsf{Del}\Sigma_k^{\mathsf{P}}$*-hard under* DelP *reductions.*

Proof. The problem $\text{D-}f$ can be decided in polynomial time by just asking the oracle for a solution to the input $\text{E-}f$ once and output "yes" if the answer was a solution and "no" if the answer was $\perp$. Since the precondition of Theorem 2.54 is fulfilled, we can conclude that $\text{E-}f$ is $\mathsf{Del}\Sigma_k^{\mathsf{P}}$-hard under DelP reductions. □

Remark 2.56. *By Corollary 2.55 we can conclude that any enumeration problem* E-$f \in \mathsf{DelP}$ *is also* DelP*-hard under* DelP *reductions:*

$$
\begin{aligned}
\text{E-}f \in \mathsf{DelP} &\implies \text{D-}f \in \mathsf{P} && (2.1)\\
&\implies \text{D-}f \text{ is } \mathsf{P}\text{-hard with respect to } \leq_m^{\mathsf{P}} && (2.2)\\
&\implies \text{E-}f \text{ is } \mathsf{DelP}\text{-hard with respect to } \mathsf{DelP} \text{ reductions.} && (2.3)
\end{aligned}
$$

For (1) we can just run the algorithm the precomputation of the algorithm that enumerates E-f *and output "yes" if and only if we find a solution that way. The second implication (2) follows the fact that any decision problem (other than* Σ^* *and* $\emptyset$*) in* P *is* P*-hard under* $\leq_m^{\mathsf{P}}$ *reductions. Finally (3) follows from Corollary 2.55.*

Proposition 2.57 ([CKP+19])**.**

1. E-$\mathsf{sat}_{\mathrm{BF}}$ *is* DelNP*-complete with respect to* DelP *reductions.*
2. E-$\mathsf{sat}_{\mathrm{DH}} \in \mathsf{DelP}$ *and thereby* DelP*-complete with respect to* DelP *reductions (see Remark 2.56).*

Next, we define enumeration classes based on team logics. This time, we do not define these classes as sets of satisfiability problems like we did in the decision or counting setting, but as the "$\leq_{\mathsf{DelP}}$ closure" of such problems. This is due to the fact that in the enumeration case, the output is the solution set, and we might not be able to express arbitrary solution sets using teams.

Problem:	E-$\mathsf{sat}_\varphi^{\mathrm{team}}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \,\}$

Definition 2.58. *Let* $\mathrm{T} \in \mathfrak{G}$*.The class* $\mathsf{DelFO}(\mathrm{T})$ *is defined as follows*

$$\mathsf{DelFO}(\mathrm{T}) = \Big[\bigcup_{\varphi \in \mathsf{FO}(\mathrm{T})} \text{E-}\mathsf{sat}_\varphi^{\mathrm{team}} \Big]^{\leq_{\mathsf{DelP}}}.$$

Note that, by definition, $\mathsf{DelFO}(\mathrm{T})$ is closed under $\leq_{\mathsf{DelP}}$ reductions.

2.9 Modelling With First-Order Structures

In this section we define how we encode propositional formulas and strings as first-order structures. We then give examples of team logic formulas that are satisfied by a structure $\mathcal{A}$ and a team X if and only if $\mathcal{A}$ represents a propositional formula χ and X a satisfying assignment for χ. To represent formulas in conjunctive normal form as first-order structures we use the vocabulary $\tau_{\mathrm{CNF}} = (V^1, P^2, N^2)$. The intention of the predicate V is to be the set of all variables. Furthermore, the predicate P (respectively N) is the incidence relation between clauses and positive (respectively negated) variables. A formula $\chi(x_1, \dots, x_k) = \bigwedge_i C_i$ with $C_i = l_{i,1} \vee \cdots \vee l_{i,m_i}$ is encoded by the following τ_{CNF}-structure

$$\mathcal{A}_\chi = (\mathrm{dom}(\mathcal{A}), V^{\mathcal{A}}, P^{\mathcal{A}}, N^{\mathcal{A}})$$

defined as follows: The elements of $\mathrm{dom}(\mathcal{A})$ are numerical encodings of the variables and clauses in χ. By abuse of notation, we write

$$\mathrm{dom}(\mathcal{A}) = \{ x_1, \dots, x_k, C_1, \dots, C_n \},$$

where we identify the variables and clauses with their encodings. The interpretations of the predicate symbols are defined as:

- $x \in V^{\mathcal{A}}$ if and only if x is a variable in χ,
- $(C_i, x) \in P^{\mathcal{A}}$ if and only if there is a j such that $l_{i,j} = x$ and
- $(C_i, x) \in N^{\mathcal{A}}$ if and only if there is a j such that $l_{i,j} = \neg x$.

Note that by definition the set of clauses of χ is given by $\mathrm{dom}(\mathcal{A}) \setminus V$.

We give two examples of how we use this encoding to model CPFs of type $\mathsf{sat}_{\mathrm{C}}$, for $\mathrm{C} \subseteq \mathrm{CNF}$ in terms of team logics.

Example 2.59. *We model the* CPF $\mathsf{sat}_{2\mathrm{CNF}^+}$ *in terms of inclusion logic. For this let* $\chi(x_1, \dots, x_k) = \bigwedge C_i \in 2\mathrm{CNF}^+$, *where each* $C_i = \ell_{i,1} \vee \ell_{i,2}$ *for some* $\ell_{i,1}, \ell_{i,2} \in \{x_1, \dots, x_k\}$. *The corresponding* τ_{CNF}*-structure then is*

$$\mathcal{A}_\chi = (\{x_1, \dots, x_k, C_1, \dots, C_n\}, \{x_1, \dots, x_k\}, \{(C_i, x_j) \mid x_j = \ell_{i,1} \text{ or } x_j = \ell_{i,2}\}, \emptyset).$$

We now provide an inclusion logic formula $\varphi_{2\mathrm{CNF}^+}$ *such that for all structures* $\mathcal{A}_\chi$ *and all teams* X *it holds*

$$\mathcal{A}_\chi \models_X \varphi_{2\mathrm{CNF}^+} \iff \mathrm{rel}(X) \models \chi. \tag{$\star$}$$

The formula is defined as $\varphi_{2\mathrm{CNF}^+}(t) := \varphi^1_{2\mathrm{CNF}^+} \wedge \varphi^2_{2\mathrm{CNF}^+}(t)$, *where the first-order sentence* $\varphi^1_{2\mathrm{CNF}^+}$ *expresses that the input structure encodes a propositional formula in* $2\mathrm{CNF}^+$ *and the formula* $\varphi^2_{2\mathrm{CNF}^+}(t)$ *ensures that satisfying teams correspond to satisfying assignments for* χ. *Formally, these formulas are defined as follows:*

$$\begin{aligned}\varphi^1_{2\mathrm{CNF}^+} := \forall C \forall x \forall y \forall z\ & \Big(\neg N(C,x) \\ & \wedge \big(P(C,x) \rightarrow \neg V(C) \wedge V(x)\big) \\ & \wedge \big(P(C,x) \wedge P(C,y) \wedge P(C,z) \rightarrow (x = y) \vee (x = z) \vee (y = z)\big)\Big)\end{aligned}$$

$$\varphi^2_{2\mathrm{CNF}^+}(t) \coloneqq V(t) \land \forall C \exists x\big(\neg V(C) \to (P(C,x) \land x \subseteq t)\big)$$

We argue that $(\star)$ *holds: Let* X *be a satisfying team for* $\varphi^2_{2\mathrm{CNF}^+}(t)$ *in* $\mathcal{A}_\chi$*. This means that* $\mathrm{rel}(X)$ *is a set of variables that contains at least one variable for each clause, which in turn means that* $\mathrm{rel}(X)$ *is a satisfying assignment for* χ*. For the converse let* β *be a satisfying assignment for* χ *and* X *be the team with* $\mathrm{rel}(X) = \beta$*. Since* $\mathrm{rel}(X)$ *is satisfying for* χ*, in particular for every clause* C *of* χ *there is a variable* $x \in \mathrm{rel}(X)$ *such that* C *is satisfied by assigning the value* 1 *to* x*. This is exactly what the formula* $\varphi^2_{2\mathrm{CNF}^+}(t)$ *states, hence* $\mathcal{A}_\chi \models_X \varphi_{2\mathrm{CNF}^+}$*.*

As there is a one-to-one correspondence between satisfying teams for $\varphi_{2\mathrm{CNF}^+}$ *in* $\mathcal{A}_\chi$ *and satisfying assignments* β *for* χ*, we can conclude* $\text{D-}\mathsf{sat}_{2\mathrm{CNF}^+} = \text{D-}\mathsf{sat}^{\mathrm{team}}_{\varphi_{2\mathrm{CNF}^+}}$, $\text{C-}\mathsf{sat}_{2\mathrm{CNF}^+} = \text{C-}\mathsf{sat}^{\mathrm{team}}_{\varphi_{2\mathrm{CNF}^+}}$ *and* $\text{E-}\mathsf{sat}_{2\mathrm{CNF}^+} = \text{E-}\mathsf{sat}^{\mathrm{team}}_{\varphi_{2\mathrm{CNF}^+}}$*.*

By Example 2.59 we know that the CPFs $\mathsf{sat}_{2\mathrm{CNF}^+}$ and $\mathsf{sat}^{\mathrm{team}}_{\varphi_{2\mathrm{CNF}^+}}$ are the same (assuming the same encodings are used). Moreover, by Remark 2.1 we can conclude $\mathsf{sat}_{2\mathrm{CNF}^-} = \mathsf{sat}_{2\mathrm{CNF}^+} = \mathsf{sat}^{\mathrm{team}}_{\varphi_{2\mathrm{CNF}^+}}$ (when switching the predicates P and N in the respective encodings).

In the next example we show that we can express the same CPFs in terms of dependence logic. This time we define a formula $\varphi_{2\mathrm{CNF}^-}$ such that $\mathsf{sat}_{2\mathrm{CNF}^-} = \mathsf{sat}^{\mathrm{team}}_{\varphi_{2\mathrm{CNF}^-}}$.

Example 2.60. *Analogously to Example 2.59, we model the* CPF $\mathsf{sat}_{2\mathrm{CNF}^-}$*, but this time in terms of dependence logic. For this, let* $\chi(x_1, \ldots, x_k) = \bigwedge C_i \in 2\mathrm{CNF}^-$*, where each* $C_i = \ell_{i,1} \lor \ell_{i,2}$ *and* $\ell_{i,1}, \ell_{i,2} \in \{\neg x_1, \ldots, \neg x_k\}$*. The corresponding* τ_{CNF}*-structure is*

$$\mathcal{A}_\chi = (\{x_1, \ldots, x_k, C_1, \ldots, C_n\}, \{x_1, \ldots, x_k\}, \emptyset, \{(C_i, x_j) \mid x_j = \ell_{i,1} \text{ or } x_j = \ell_{i,2}\}).$$

We now provide a dependence logic formula $\varphi_{2\mathrm{CNF}^-}$ *such that for all structures* $\mathcal{A}_\chi$ *and all teams* X *it holds*

$$\mathcal{A}_\chi \models_X \varphi_{2\mathrm{CNF}^-} \iff \mathrm{rel}(X) \models \chi. \qquad (\star)$$

The formula is defined as $\varphi_{2\mathrm{CNF}^-}(t) \coloneqq \varphi^1_{2\mathrm{CNF}^-} \land \varphi^2_{2\mathrm{CNF}^-}(t)$*, where the first-order sentence* $\varphi^1_{2\mathrm{CNF}^-}$ *expresses that the input structure encodes a propositional formula in* $2\mathrm{CNF}^-$ *and the formula* $\varphi^2_{2\mathrm{CNF}^-}(t)$ *ensures that satisfying teams correspond to satisfying assignments for* χ*. Formally, these formulas are defined as follows:*

$$\begin{aligned}\varphi^1_{2\mathrm{CNF}^+} \coloneqq\ & \forall C \forall x \forall y \forall z \\ & \Big(\neg P(C,x) \\ & \land \big(N(C,x) \to \neg V(C) \land V(x)\big) \\ & \land \big(N(C,x) \land N(C,y) \land N(C,z) \to (x = y) \lor (x = z) \lor (y = z)\big)\Big)\end{aligned}$$

$$\varphi^2_{2\mathrm{CNF}^-}(t) \coloneqq V(t) \land \forall C\Big(\neg V(C) \to \big(\exists x(N(C,x) \land {=}(C,x) \land x \neq t)\big)\Big)$$

We argue that ($\star$) holds: Let X be a team with $\mathcal{A}_\chi \models_X \varphi_{2\mathrm{CNF}^-}$. This means that for any clause C in χ, there is a fixed variable $x \in \mathsf{vars}(\chi) \setminus \mathrm{rel}(X)$. In other words every C, and thus also the whole formula χ, is satisfied by $\mathrm{rel}(X)$. *For the other direction let β be a satisfying assignment for χ and X be the team with* $\mathrm{rel}(X) = \beta$. *Now we can choose one variable $x \in \mathsf{vars}(\chi) \setminus \mathrm{rel}(X)$ for each clause C of χ, such that C is satisfied by assigning the value 0 to x. As this property is stated by the formula we can conclude $\mathcal{A}_\chi \models_X \varphi_{2\mathrm{CNF}^-}$.*

As there is a one-to-one correspondence between satisfying teams for $\varphi_{2\mathrm{CNF}^-}$ in $\mathcal{A}_\chi$ and satisfying assignments β for χ, we can conclude $\text{D-}\mathsf{sat}_{2\mathrm{CNF}^-} = \text{D-}\mathsf{sat}^{\mathrm{team}}_{\varphi_{2\mathrm{CNF}^-}}$, $\text{C-}\mathsf{sat}_{2\mathrm{CNF}^-} = \text{C-}\mathsf{sat}^{\mathrm{team}}_{\varphi_{2\mathrm{CNF}^-}}$ *and* $\text{E-}\mathsf{sat}_{2\mathrm{CNF}^-} = \text{E-}\mathsf{sat}^{\mathrm{team}}_{\varphi_{2\mathrm{CNF}^-}}$.

To be able to also represent Σ_1CNF formulas, we adjust the vocabulary τ_{CNF} to obtain the new vocabulary $\tau_{\Sigma_1\mathrm{CNF}}$:

$$\tau_{\Sigma_1\mathrm{CNF}} = (F^1, B^1, P^2, N^2).$$

The predicate symbols P and N have the same intended meaning as in τ_{CNF}, but—since Σ_1CNF formulas have quantifiers—we now differentiate between bound variables (predicate B) and free variables (predicate F). Clauses then are the elements that are neither a bound nor a free variable. Now an arbitrary Σ_1CNF formula $\chi(x_1, \dots, x_k) = \exists y_1 \dots \exists y_l\ \psi(x_1, \dots, x_k, y_1, \dots, y_l)$ with $\psi = \bigwedge_{i=1}^n C_i$ and $C_i = l_{i,1} \lor \dots \lor l_{i,m_i}$ is encoded as the $\tau_{\Sigma_1\mathrm{CNF}}$-structure

$$\mathcal{A}_\chi = (\mathrm{dom}(\mathcal{A}), F^{\mathcal{A}}, B^{\mathcal{A}}, P^{\mathcal{A}}, N^{\mathcal{A}}),$$

defined as follows: The elements of $\mathrm{dom}(\mathcal{A}) = \{x_1, \dots, x_k, y_1, \dots, y_l, C_1, \dots, C_n\}$ are the numerical encodings of the variables and clauses in χ. The interpretations of the predicate symbols are defined as:

- $x \in F^{\mathcal{A}}$ if and only if x is a free variable in χ,
- $x \in B^{\mathcal{A}}$ if and only if x is a bound variable in χ,
- $(C_i, x) \in P^{\mathcal{A}}$ if and only if there is a j such that $l_{i,j} = x$ and
- $(C_i, x) \in N^{\mathcal{A}}$ if and only if there is a j such that $l_{i,j} = \neg x$.

We close this section by explaining how we encode binary strings as first-order structures. For this let $\tau_{\mathrm{string}} = (S^1)$ be a vocabulary. For any binary string $w = w_0 w_1 \dots w_{n-1} \in \{0,1\}^*$ we define the structure encoding w as

$$\mathcal{A}_w = (\{0, 1, \dots, n-1\}, S)$$

where $S(i) = w_i$ for all i.

3 Counting Problems in Team Logics

We aim to classify the counting classes of the three "main" team logics, which are $\#\mathsf{FO}(\perp)$, $\#\mathsf{FO}({=}(\dots))$ and $\#\mathsf{FO}(\subseteq)$. First we establish results that hold for all these logics and dedicate a section to each individual logic afterwards. At the end of this section we show $\#\cdot\mathsf{NP}$-completeness of two problems to give some of our results more context and then close with a summary of our results in form of a class diagram.

First we would like to point out that our reducibility notion of choice—first-order reductions—are suitable for our task. We do this by showing that our logics are closed under these kind of reductions, that is, C-$f_1 \leq^{\mathsf{FO}}_{\mathrm{par}}$ C-f_2 and C-$f_2 \in \#\mathsf{FO}(\perp)$ imply C-$f_1 \in \#\mathsf{FO}(\perp)$ (for $\#\mathsf{FO}({=}(\dots))$ and $\#\mathsf{FO}(\subseteq)$,respectively).

Theorem 3.1 ([HKM$^+$19])**.** $\#\mathsf{FO}(\mathrm{T})$ *is closed under first-order reductions for* $\mathrm{T} \subseteq \{{=}(\dots), \perp, \subseteq\}$.

Proof. Let C-f_1, C-f_2 be two functions and σ_1, σ_2 two vocabularies such that C-$f_1 \leq^{\mathsf{FO}}_{\mathrm{par}}$ C-f_2 via FO-query $I = (\psi_0, \dots, \psi_k)$. Furthermore, let C-$f_2 \in \#\mathsf{FO}(\mathrm{T})$, then there is a formula $\varphi(x_1, \dots, x_n) \in \mathsf{FO}(\sigma_2, \mathrm{T})$ such that C-$f_2(\mathcal{A})$ is equal to the number of teams X satisfying $\mathcal{A} \models_X \varphi(x_1, \dots, x_n)$ or in short C-f_2 = C-$\mathsf{sat}^{\mathrm{team}}_{\varphi}$.

We define a formula $\varphi^*(x_1^1, \dots, x_1^k, \dots, x_n^1, \dots, x_n^k) \in \mathsf{FO}(\sigma_1, \mathrm{T})$ such that for all σ_1-structures $\mathcal{A}$ and all teams X over $x_1^1, \dots, x_1^k, \dots, x_n^1, \dots, x_n^k$,

$$\mathcal{A} \models_X \varphi^*(x_1^1, \dots, x_1^k, \dots, x_n^1, \dots, x_n^k) \iff I(\mathcal{A}) \models_{I(X)} \varphi(x_1, \dots, x_n)$$

For readability we write $\overline{x}_i$ to denote the variables $x_i^1, \dots, x_i^k$. We first inductively define a formula φ' from φ as follows:

- $\varphi' \coloneqq \psi_R(\overline{x}_1, \dots, \overline{x}_{a_\ell})$ if $\varphi = R(x_1, \dots, x_{a_\ell})$
- $\varphi' \coloneqq \widetilde{\psi}_R(\overline{x}_1, \dots, \overline{x}_{a_\ell})$ if $\varphi = \neg R(x_1, \dots, x_{a_\ell})$
- $\varphi' \coloneqq \bigwedge_{i=1}^{k} x^i = y^i$ if $\varphi = (x = y)$
- $\varphi' \coloneqq \overline{x} \leq \overline{y}$ if $\varphi = x \leq y$

- $\varphi' := +(\overline{x}, \overline{y}, \overline{z})$ if $\varphi = +(x, y, z)$
- $\varphi' := \times(\overline{x}, \overline{y}, \overline{z})$ if $\varphi = \times(x, y, z)$
- $\varphi' := \bigwedge_{i=1}^{k} =(\overline{x}_1, \dots, \overline{x}_m, y^i)$ if $\varphi = \; =(x_1, \dots, x_m, y)$
- $\varphi' := (\overline{x}_1, \dots, \overline{x}_m) \subseteq (\overline{y}_1, \dots, \overline{y}_m)$ if $\varphi = (x_1, \dots, x_m) \subseteq (y_1, \dots, y_m)$
- $\varphi' := (\overline{y}_1, \dots, \overline{y}_{m_1}) \bot_{(\overline{x}_1, \dots, \overline{x}_{m_2})} (\overline{z}_1, \dots, \overline{z}_{m_1})$ if
  $\varphi = (y_1, \dots, y_{m_1}) \bot_{(x_1, \dots, x_{m_2})} (z_1, \dots, z_{m_1})$
- $\varphi' := \varphi_1' \wedge \varphi_2'$ if $\varphi = \varphi_1 \wedge \varphi_2$
- $\varphi' := \varphi_1' \vee \varphi_2'$ if $\varphi = \varphi_1 \vee \varphi_2$
- $\varphi' := \exists \overline{x}\; \psi_0(\overline{x}) \wedge \varphi_1'$ if $\varphi = \exists x\; \varphi_1$
- $\varphi' := \forall \overline{x}\; \psi_0(\overline{x}) \rightarrow \varphi_1'$ if $\varphi = \forall x\; \varphi_1$

The numerical predicates $\leq, +, \times$ are definable for tuples by Immerman [Imm99]. Now, we define $\varphi^*(\overline{x}_1, \dots, \overline{x}_n) := \varphi'(\overline{x}_1, \dots, \overline{x}_n) \wedge \bigwedge_i \psi_0(\overline{x}_i)$. We conclude that there is a formula $\varphi^*(\overline{x}_1, \dots, \overline{x}_n) \in \mathsf{FO}(\mathrm{T})$ over σ_1 such that

$$\begin{aligned}
\text{C-}f_1(\mathcal{A}) &= \text{C-}f_2(I(\mathcal{A}))) \\
&= |\{\, X \mid X \neq \emptyset \text{ and } I(\mathcal{A}) \models_{I(X)} \varphi(x_1, \dots, x_n) \,\}| \\
&= |\{\, X \mid X \neq \emptyset \text{ and } \mathcal{A} \models_{X} \varphi^*(\overline{x}_1, \dots, \overline{x}_n) \,\}| \\
&= \text{C-}\mathsf{sat}^{\text{team}}_{\varphi^*}(\mathcal{A})
\end{aligned}$$

and therefore that $\text{C-}f_1 \in \#\mathsf{FO}(\mathrm{T})$. □

Since $\#\mathsf{FO}(\mathrm{T})$ is closed under first-order reductions for $\mathrm{T} \subseteq \{=(\dots), \bot, \subseteq\}$ by Theorem 3.1, this result especially holds for the classes $\#\mathsf{FO}(\bot)$, $\#\mathsf{FO}(=(\dots))$ and $\#\mathsf{FO}(\subseteq)$.

Next we establish an upper bound that holds for all considered logics, which we reconsider for individual cases later.

Theorem 3.2 ([HKM+19])**.** *For any set* $\mathrm{T} \in \mathfrak{G}_{\mathsf{NP}}$*, we have that* $\#\mathsf{FO}(\mathrm{T}) \subseteq \#\cdot\mathsf{NP}$.

Proof. Let $\text{C-}\mathsf{sat}^{\text{team}}_{\varphi} \in \#\mathsf{FO}(\mathrm{T})$ with $\varphi(x_1, \dots, x_k) \in \mathsf{FO}(\mathrm{T})$. To count the number of (non-empty) teams X with $\mathcal{A} \models_X \varphi$ for a given input structure $\mathcal{A}$ with a $\#\cdot\mathsf{NP}$-algorithm, we first non-deterministically guess a team X and check in NP whether $\mathcal{A} \models_X \varphi$ holds. The latter can be done since $\mathrm{T} \subseteq \mathfrak{G}_{\mathsf{NP}}$. Therefore $\text{C-}\mathsf{sat}^{\text{team}}_{\varphi} \in \#\cdot\mathsf{NP}$. □

3.1 Counting Problems in Independence Logic

We show that the result $\mathsf{FO}(\perp) = \mathsf{NP}$ (see Corollary 2.27) in the decision setting can be translated to the counting setting, that is, $\#\mathsf{FO}(\perp) = \#\cdot\mathsf{NP}$. We already covered the direction $\#\mathsf{FO}(\perp) \subseteq \#\cdot\mathsf{NP}$ in Theorem 3.2, but we still have to show the converse. We do this in two steps, first we show $\#\cdot\mathsf{NP} \subseteq \#\Sigma_1^1$ and afterwards $\#\Sigma_1^1 = \#\mathsf{FO}(\perp)$.

Theorem 3.3 ([HKM+19])**.** $\#\cdot\mathsf{NP} \subseteq \#\Sigma_1^1$.

Proof. Let C-$f \in \#\cdot\mathsf{NP}$ via D-$g \in \mathsf{NP}$ and the polynomial p, with $p(n) = n^\ell + c$ for some $\ell, c \in \mathbb{N}$. By definition we have C-$f(x) = |\{\, y \mid |y| \leq p(|x|), (x, y) \in \text{D-}g \,\}|$. We encode tuples (x, y) with $x \in \{0,1\}^*$ and $y \in \{0,1\}^{|x|^k}$ by structures $\mathcal{A}_{(x,y)} = (\{0, \dots, |x| - 1\}, S, R^k)$ over the vocabulary $\tau_k = \tau_{\text{string}} \cup (R^k)$. Here x is encoded by the string relation S and y is encoded by R^k, with $R(i_0, \dots, i_{k-1}) = y_{i_0 + i_1 \cdot |x| + \cdots + i_k \cdot |x|^k}$. Such an encoding is possible because as mentioned before we can define the extension of the numerical predicates to tuples in FO (see [Imm99]). For strings x with $|x| \geq 2$ we can choose k such that $|x|^k \geq p(|x|)$ (strings of length 1 can be handled separately). Fix such a k.

Now consider the decision problem

$$\text{D-}g' := \left\{ \mathcal{A}_{(x,y)} \;\middle|\; \begin{array}{c} \mathcal{A}_{(x,y)} \in \text{STRUC}(\tau_k), y = y_0 \dots y_{|x|^k - 1}, \\ y_{p(|x|)} = \cdots = y_{|x|^k - 1} = 0 \text{ and } (x, y_0 \dots y_{p(|x|)-1}) \in \text{D-}g \end{array} \right\}.$$

For any given x, $\mathcal{A}_{(x,y)}$ is an element of D-g' if and only if the first $p(|x|)$ bits of y form an input z such that $(x, z) \in$ D-g and the rest of the bits are fixed to be 0. Thus we can write C-$f(x)$ as:

$$\text{C-}f(x) = |\{\, y \mid \mathcal{A}_{(x,y)} \in \text{D-}g' \,\}|.$$

Obviously D-$g' \in \mathsf{NP}$, which, by Fagin's Theorem (see Theorem 2.23), implies that there is a sentence $\psi \in \Sigma_1^1$ over τ_k such that

$$\mathcal{A}_{(x,y)} \in \text{D-}g' \iff \mathcal{A}_{(x,y)} \models \psi.$$

Viewing ψ as a formula over the vocabulary τ_{string} with free relational variable R of arity k (that encodes y) we have

$$\mathcal{A}_{(x,y)} \in \text{D-}g' \iff \mathcal{A}_x, R \models \psi(R)$$

for all $x \in \{0,1\}^*$, which yields

$$\text{C-}f(x) = |\{R \mid \mathcal{A}_x, R \models \psi(R)\}| = \text{C-}\mathsf{sat}_\psi^{\text{rel}}(\text{enc}(\mathcal{A}_x)) = \text{C-}\mathsf{sat}_\psi^{\text{rel}}(x).$$

Hence C-$f \in \#\Sigma_1^1$. □

Theorem 3.4 ([HKM$^+$19])**.** $\#\Sigma^1_1 = \#\mathsf{FO}(\perp)$.

Proof. Let $\psi(R) \in \Sigma^1_1$ be a sentence with a k-ary relation symbol R. We translate the formula $\psi(R)$ to a formula $\psi'(R')$ with a new $(k+1)$-ary relation R' such that $\text{C-}\mathsf{sat}^{\text{rel}}_{\psi}(x) = \text{C-}\mathsf{sat}^{\text{rel}}_{\psi'}(x)$ for all inputs x and $\psi'(R')$ is only satisfied by non-empty relations. The formula $\psi'(R')$ is defined as follows:

$$\begin{aligned}\psi'(R') = \exists \min \exists \max \forall x\\ &\Big(\leq(\min, x) \wedge \leq(x, \max)\\ &\wedge \forall x_1, \dots, x_{k+1}\big(R'(x_1, \dots, x_{k+1}) \to (\bigwedge_{i=1}^{k+1} x_i = \min \vee x_{k+1} = \max)\big)\\ &\wedge \big(\psi(\emptyset) \leftrightarrow R'(\min, \dots, \min)\big)\\ &\wedge \exists R\big((R(\overline{x}) \leftrightarrow R'(\overline{x}, \max)\big) \wedge \psi(R)\Big).\end{aligned}$$

Now we have that for any structure $\mathcal{A}$ and relation R:

$$\mathcal{A}, R \models \psi \iff \mathcal{A}, R' \models \psi',$$

where $R' = \{(\overline{x}, \max) \mid \overline{x} \in R\} \cup \{(\min, \dots, \min)\}$, when $\mathcal{A}, \emptyset \models \psi$ and $R' = \{(\overline{x}, \max) \mid \overline{x} \in R\}$ otherwise. Therefore it follows $\text{C-}\mathsf{sat}^{\text{rel}}_{\psi}(x) = \text{C-}\mathsf{sat}^{\text{rel}}_{\psi'}(x)$. Furthermore, ψ' is never satisfied by the empty relation as desired. By Proposition 2.24 and Definition 2.45, it follows that there is a formula $\varphi \in \mathsf{FO}(\perp)$ such that for all inputs x it holds $\text{C-}\mathsf{sat}^{\text{team}}_{\varphi}(x) = \text{C-}\mathsf{sat}^{\text{rel}}_{\psi'}(x)$ and therefore $\text{C-}\mathsf{sat}^{\text{team}}_{\varphi}(x) = \text{C-}\mathsf{sat}^{\text{rel}}_{\psi}(x)$.
The other direction can be shown in the same manner. □

Corollary 3.5. $\#\mathsf{FO}(\perp) = \#\cdot\mathsf{NP}$

Proof. $\#\mathsf{FO}(\perp) \subseteq \#\cdot\mathsf{NP}$ follows from Theorem 3.2 and Corollary 2.15, just choose $\mathrm{T} = \{\perp\}$. The converse follows from the composition of Theorem 3.4 and Theorem 3.3. □

Note that Theorem 3.4 and Corollary 3.5 also imply that $\#\Sigma^1_1 = \#\cdot\mathsf{NP}$, which can be seen as a counting version of Fagin's Theorem (see Theorem 2.23).

3.2 Counting Problems in Dependence Logic

For dependence logic we can not translate the result $\mathsf{FO}(=(\dots)) = \mathsf{NP}$ (see Corollary 2.27) from the decision setting to the counting setting like we did for independence logic. This is due to the fact that there are Σ^1_1 formulas—those where R occurs positively—that have not a corresponding dependence logic formula such that the solutions are preserved. Therefore we are not able to show $\#{\cdot}\mathsf{NP} \subseteq \#\mathsf{FO}(=(\dots))$, the other direction follows from Theorem 3.2 and Corollary 2.15 though.

Instead of showing the equality of those classes, we show that $\#\mathsf{FO}(=(\dots))$ contains the problem C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$, which is also $\#{\cdot}\mathsf{NP}$-complete under Turing reductions (see Section 3.4). This does not imply $\#{\cdot}\mathsf{NP} = \#\mathsf{FO}(=(\dots))$, but it implies the relativisation, that is, $\mathsf{FP}^{\#\cdot\mathsf{NP}} = \mathsf{FP}^{\#\mathsf{FO}(=(\dots))}$.

Problem:	C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$
Input:	$\chi \in \Sigma_1\mathrm{CNF}^-$
Output:	$\lvert \{ \beta \in \Theta(\chi) \mid \beta \models \chi \text{ and } \beta \neq \emptyset \} \rvert$

We like to explain the choice of the problem C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$ for this task: We can not simply work with a problem based on quantified boolean formulas with no free variables (like in the decision case), as there would not be much to count (the output would be either 0 or 1). Therefore we need to allow free variables in the problem definition. Here it is important that the solutions have a monotone property since we want to describe them with dependence logic formulas, which are downwards monotone. Thus we allow free variables only to occur negatively.

Theorem 3.6 ([HKM+19])**.** C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-} \in \#\mathsf{FO}(=(\dots))$.

Proof. By Proposition 2.25 it suffices to construct a Σ^1_1 formula $\psi(R)$ with R occurring only negatively such that for each $\tau_{\Sigma_1\mathrm{CNF}}$-structure $\mathcal{A}_\chi$, the number of relations R with $\mathcal{A}_\chi, R \models \psi(R)$ is equal to the number of satisfying assignments of the $\Sigma_1\mathrm{CNF}^-$ formula χ. Note a subtle point in this setting: The empty assignment is not counted by C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$. In the formula $\psi(R)$, the empty assignment corresponds to the empty relation, which, in turn, corresponds to the empty team in the translation to $\mathsf{FO}(=(\dots))$ given in Proposition 2.25. By definition we do not count the empty team in $\#\mathsf{FO}(=(\dots))$, and the empty assignment is thus not counted in this setting.

Let us now define the formula $\psi(R)$. First, let ψ_1 be an FO formula expressing that $\mathcal{A}_\chi$ is a correct encoding of a $\Sigma_1\mathrm{CNF}^-$ formula (i.e., F and B correspond to disjoint sets, free variables occur only negatively in the clauses, etc.). Formally ψ_1 is defined as follows:

$$\begin{aligned}\psi_1 :=& \forall x\big(\neg B(x) \lor \neg F(x)\big)\\ &\land \forall C \forall x\Big(N(C,x) \to \big(\neg B(C) \land \neg F(C) \land (B(x) \lor F(x))\big)\Big)\\ &\land \forall C \forall x\Big(P(C,x) \to \big(\neg B(C) \land \neg F(C) \land B(x)\big)\Big)\end{aligned}$$

Second, let ψ_2 be a formula expressing that R only assigns values to free variables and that each clause must be satisfied by the assignment, that is, formally,

$$\begin{aligned}\psi_2(R) \coloneqq \exists S \forall C \Bigg(\big(\neg F(C) \wedge \neg B(C)\big) \rightarrow \\ &\exists x \Big(N(C,x) \wedge \big((B(x) \wedge \neg S(x)) \vee (F(x) \wedge \neg R(x))\big)\Big) \\ &\vee \exists x \Big(P(C,x) \wedge (B(x) \wedge S(x))\Big)\Bigg),\end{aligned}$$

Finally, let

$$\psi(R) \coloneqq \psi_1 \wedge \psi_2(R)$$

It is easy to see that the formula $\psi(R)$ has the desired properties. □

We like to briefly discuss #FO(=(...)) versus #P: #FO(=(...)) ⊆ #P together with Theorem 3.6 would imply that $\text{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-} \in$ #P. This seems not to be the case, as it would mean that there is a #P-machine that answers $\text{D-sat}_{\mathrm{BF}}$ queries in every path. The converse—#P ⊆ #FO(=(...))—is also unlikely since TotP and therefore also #P contains the (non-monotone) problem $\text{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}}$ (see Theorem 3.13) which is probably not contained in #FO(=(...)), as FO(=(...)) is downwards closed. We therefore conjecture that #P (and TotP for that matter) are incomparable to #FO(=(...)).

Theorem 3.7 ([HKM+19]). *$\text{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$ is complete for #FO(=(...)) with respect to first-order reductions.*

Proof. By Theorem 3.6, $\text{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$ is contained in #FO(=(...)). It remains to show hardness. Let $\varphi(x_1, \dots, x_m) \in$ FO(=(...)) and $\mathcal{A}$ be a structure. We reduce computing the value of $\text{C-sat}^{\mathrm{team}}_{\varphi}$ to counting the number of satisfying assignments (apart from the empty assignment) of a suitable Boolean formula $\chi^{\varphi,\mathcal{A}} \in \Sigma_1\mathrm{CNF}^-$. By [Vää07], we may assume without loss of generality that φ is of the form

$$\forall y_1 \dots \forall y_k \exists y_{k+1} \dots \exists y_{k+l} \big(\bigwedge_t =\!(\overline{u}_t, w_t) \wedge \psi(y_1, \dots, y_{k+l}, x_1, \dots, x_m)\big),$$

where ψ is a quantifier-free FO formula, $w_t \in \{y_{k+1}, \dots, y_{k+l}\}$, and $\overline{u}_i$ is a tuple consisting of some of the variables $y_1, \dots, y_k$. The formula $\chi^{\varphi,\mathcal{A}}$ is defined over the set $\{p_s \mid s \in A^m \cup A^{m+1} \cup \dots \cup A^{m+k+l}\}$ of propositional variables. Observe that each such s can be identified with a partial first-order assignment over the domain $\{x_1, \dots, x_m, y_1, \dots, y_{k+l}\}$. Since φ is a fixed formula, the number of such assignments and, consequently, also the number of variables p_s is polynomial. The variables p_s for $s \in A^{m+1} \cup \dots \cup A^{m+k+l}$ will be existentially quantified in $\chi^{\varphi,\mathcal{A}}$, whereas the variables p_s for $s \in A^m$ will remain free and occur only negatively.

We now define the set of clauses $\mathfrak{C}$ of $\chi^{\varphi,\mathcal{A}}$. For every universally quantified variable y_i in φ, and for every $s \in A^{m+(i-1)}$ we introduce to $\mathfrak{C}$ the following set of clauses:

$$\{p_s \to p_{s'} \mid s' \in A^{m+i} \text{ and } s = s'|_{\{\overline{x},y_1,\dots,y_{i-1}\}}\} \tag{$\forall$}$$

For every existentially quantified variable y_i in φ, and for every $s \in A^{m+(i-1)}$ we introduce the following clause:

$$p_s \to \bigvee_{\substack{s' \in A^{m+i} \text{ and} \\ s = s'|_{\{\overline{x},y_1,\dots,y_{i-1}\}}}} p_{s'}. \tag{$\exists$}$$

The quantifier-free part of the formula φ also gives rise to clauses as follows. For each dependence atom $=\!(\overline{u}_t, w_t)$ we introduce the set of clauses:

$$\{\neg p_s \vee \neg p_{s'} \mid s, s' \in A^{m+k+l}, s(\overline{u}_t) = s'(\overline{u}_t) \text{ and } s(w_t) \neq s'(w_t)\}. \tag{$=\!(\dots)$}$$

Finally, for the FO formula ψ the team semantics satisfaction condition stipulates that all assignments $s \in A^{m+k+l}$ should satisfy ψ (since ψ is flat). This can be expressed by introducing the following two sets of clauses:

$$\{p_s \to \top \mid s \in A^{m+k+l} \text{ and } \mathcal{A} \models_s \psi\} \tag{$\top$}$$

and

$$\{p_s \to \bot \mid s \in A^{m+k+l} \text{ and } \mathcal{A} \not\models_s \psi\}. \tag{$\bot$}$$

Now define $\chi^{\varphi,\mathcal{A}} \in \Sigma_1\mathrm{CNF}^-$ as

$$\chi^{\varphi,\mathcal{A}} := \exists\{p_s \mid s \in \bigcup_{1 \le i \le k+l} A^{m+i}\} \bigwedge_{C \in \mathfrak{C}} C.$$

Clearly, there is a one-to-one-correspondence between teams X over the domain $\{x_1, \dots, x_m\}$ and assignments β of formula $\chi^{\varphi,\mathcal{A}}$. Furthermore, it is also easy to check that for all teams X with domain $\{x_1, \dots, x_m\}$

$$\mathcal{A} \models_X \varphi(x_1, \dots, x_m) \iff \beta \models \chi^{\varphi,\mathcal{A}},$$

where the Boolean assignment β is defined as $p_s \in \beta$ if and only if $s \in X$.

We now argue that the above can be done using a first-order reduction. First note that clauses of types $\forall$,$\exists$,$=\!(\dots)$ only depend on the input length n. We thus only need to give a closed formula in n (only using $+$ and $\times$) to calculate for every clause the incident literals.

As an illustration, we give the details for clauses of type $\forall$. In total, there are $\sum_{i=0}^{k} n^{m+i}$ clauses of this type, where each summand gives the number of clauses corresponding to one universal quantifier. Consider a clause C that is counted in the i-th summand. The clause C is of the form $p_s \to p_{s'}$, where $s' \in A^{m+i}$ and $s = s'|_{\{(\overline{x}, y_1, \dots, y_{i-1})\}}$. Let j be the position of C within the i-th summand. Then s' is the j-th element of A^{m+i} and s is defined accordingly. Clauses arising from existential quantifiers can be handled analogously.

For clauses corresponding to dependence atoms it is helpful to simplify the construction as follows: Consider the atom $=(\overline{u}_t, w_t)$. Instead of adding the clauses from $(=(\dots))$ we add a clause for each pair $s, s' \in A^{m+k+l}$ and check whether that pair falsifies the additional condition using an FO formula. If it does, we construct a fixed tautology instead of the clause $(\neg p_s \vee \neg p_{s'})$.

Clauses of type $\top$,$\bot$ depend not only on the input length but on the actual input. They can be handled simultaneously by adding clauses of the form $p_s \to p_s$ for type $\top$ and $p_s \to \bot \equiv \neg p_s$ for type $\bot$. Distinguishing between those two types only depends on the FO-definable property $\mathcal{A} \models_s \psi$. □

3.3 Counting Problems in Inclusion Logic

In this section we show $\#\mathsf{FO}(\subseteq) \subseteq \#\mathsf{P}$ and that this inclusion is also strict unless $\mathsf{P} = \mathsf{NP}$. Moreover we strengthen this result by showing $\#\mathsf{FO}(\subseteq) \subseteq \mathsf{TotP}$. To indicate that this inclusion might also be strict, we show that TotP contains the problem C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}}$ which is hard for $\#\mathsf{FO}(\subseteq)$ but probably not included in $\#\mathsf{FO}(\subseteq)$.

Theorem 3.8 ([HKM+19])**.** *$\#\mathsf{FO}(\subseteq) \subseteq \#\mathsf{P}$.*

Proof. To count the number of satisfying teams for a given input structure $\mathcal{A}$ and a formula in $\mathsf{FO}(\subseteq)$, we simply guess a team and verify that it satisfies the formula. The latter step can be done in polynomial time, since model-checking for $\mathsf{FO}(\subseteq)$ is in P by Corollary 2.31. □

The next lemma connects the counting setting to the decision setting. It allows us to answer $\#\mathsf{FO}(\subseteq)$ versus $\#\mathsf{P}$ depending on P versus NP.

Lemma 3.9. *Let* C-$f \in \#\mathsf{FO}(\subseteq)$*, then* D-$f \in \mathsf{P}$.

Proof. Let C-$f \in \#\mathsf{FO}(\subseteq)$. That means there is a formula $\varphi \in \mathsf{FO}(\subseteq)$ such that C-f = C-$\mathsf{sat}^{\mathrm{team}}_{\varphi}$ and hence D-f = D-$\mathsf{sat}^{\mathrm{team}}_{\varphi}$. A structure $\mathcal{A}$ is in D-$\mathsf{sat}^{\mathrm{team}}_{\varphi}$ if there is a team $X \neq \emptyset$ with $\mathcal{A} \models_X \varphi(\overline{x})$, which is equivalent to asking whether $\mathcal{A} \models_{\{s_\emptyset\}} \exists\overline{x}\, \varphi(\overline{x})$ holds. By Corollary 2.31, over ordered structures, the properties definable by $\mathsf{FO}(\subseteq)$-sentences are exactly the properties in P, hence it follows that D-$f \in \mathsf{P}$. □

By Lemma 3.9 we can conclude that $\#\mathsf{FO}(=(\dots)) \not\subseteq \#\mathsf{FO}(\subseteq)$ (unless $\mathsf{P} = \mathsf{NP}$): Suppose $\#\mathsf{FO}(=(\dots)) \subseteq \#\mathsf{FO}(\subseteq)$, then $\text{C-}\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\text{CNF}^-} \in \#\mathsf{FO}(\subseteq)$ and therefore $\text{D-}\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\text{CNF}^-} \in \mathsf{P}$. This is only possible if $\mathsf{P} = \mathsf{NP}$, since $\text{D-}\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\text{CNF}^-}$ is obviously NP-complete.

Corollary 3.10 ([HKM+19]). *If* $\mathsf{P} \neq \mathsf{NP}$, *then* $\#\mathsf{FO}(\subseteq) \neq \#\mathsf{P}$.

Proof. Suppose $\#\mathsf{FO}(\subseteq) = \#\mathsf{P}$. This means that $\text{C-}\mathsf{sat}_{3\text{CNF}} \in \#\mathsf{FO}(\subseteq)$. Then, by Lemma 3.9, $\text{D-}\mathsf{sat}_{3\text{CNF}}$ is in P, which implies $\mathsf{P} = \mathsf{NP}$ (since $\text{D-}\mathsf{sat}_{3\text{CNF}}$ is NP-complete), contradicting the assumption. □

Theorem 3.8 and Corollary 3.10 indicate that $\#\mathsf{FO}(\subseteq)$ is most likely a strict subclass of $\#\mathsf{P}$. Nevertheless, we show in the next theorem that $\#\mathsf{FO}(\subseteq)$ contains the problem $\text{C-}\mathsf{sat}^{\neg\emptyset}_{\text{DH}}$ which is complete for $\#\mathsf{P}$ with respect to Turing reductions. By this we also show that $\mathsf{FP}^{\#\mathsf{P}} = \mathsf{FP}^{\#\mathsf{FO}(\subseteq)}$.

Problem:	$\text{C-}\mathsf{sat}^{\neg\emptyset}_{\text{DH}}$
Input:	$\chi \in \text{DH}$
Output:	$\lvert \{ \beta \in \Theta(\chi) \mid \beta \models \chi \text{ and } \beta \neq \emptyset \} \rvert$

We briefly argue the $\#\mathsf{P}$-completeness for $\text{C-}\mathsf{sat}^{\neg\emptyset}_{\text{DH}}$: Valiant [Val79b] showed that $\text{C-}\mathsf{sat}_{2\text{CNF}^+}$ is $\#\mathsf{P}$-complete with respect to Turing reductions. This result can be translated to $\text{C-}\mathsf{sat}_{\text{DH}}$ since every 2CNF^+ formula is also a DualHorn formula. The empty assignment can be handled in a further reduction which is similar to the one we later use in Theorem 3.17.

Theorem 3.11 ([HKM+19]). $\text{C-}\mathsf{sat}^{\neg\emptyset}_{\text{DH}} \in \#\mathsf{FO}(\subseteq)$.

Proof. By Proposition 2.32 it suffices to give a myopic τ_{CNF}-formula ψ such that for all DualHorn formulas χ the number of satisfying assignments is equal to the number of relations R with $\mathcal{A}_\chi, R \models \psi(R)$. We define the formula as $\psi(R) := \psi_1 \wedge \psi_2(R)$, where ψ_1 expresses that $\mathcal{A}_\chi$ is an encoding of a DualHorn formula and ψ_2 ensures that the solutions are preserved. The formulas ψ_1, ψ_2 are defined as follows:

$$\begin{aligned}\psi_1 := \forall C \forall x \Big(&\big(P(C,x) \to \neg V(C) \wedge V(x)\big) \wedge \\ &\big(N(C,x) \to \neg V(C) \wedge V(x) \wedge \neg \exists y (N(C,y) \wedge x \neq y)\big)\Big)\end{aligned}$$

$$\begin{aligned}\psi_2(R) := \forall x \Bigg(R(x) \to V(x) \wedge \Big(\forall C \neg V(C) \to &\\ &\big((\neg \exists z N(C,z)\big) \to \big(\exists y P(C,y) \wedge R(y)\big) \\ &\wedge \big(N(C,x) \to (\exists y P(C,y) \wedge R(y))\big)\Big)\Bigg).\end{aligned}$$

Note that ψ is not a myopic formula but can be easily transformed into one, since ψ_2 is myopic and ψ_1 is independent from R.

Now suppose R satisfies the formula ψ_2. Let $x \in R$. It follows that all clauses that contain x or contain only positive literals are satisfied by R: If x is positively contained in a clause C, then it is already satisfied since $x \in R$. If x is negatively contained in C, then there must be another variable y that occurs positively in C (since each clause contains at most one negative literal) with $y \in R$. If C only contains positive literals, then there must be also one $y \in R$. This only works if there is at least one variable included in R. If R is empty in the first place the premise of the first implication is always false and therefore the conclusion can be anything. It follows that $\psi_2(\emptyset)$ is always true, which is no surprise since ψ_2 is a myopic formula. But since we are only looking for non-empty relations, non-empty assignments β, respectively, this is not a problem. Now for all assignments $\beta \neq \emptyset$ it holds that $\beta \models \chi \iff \mathcal{A}_\chi, \beta \models \psi_1 \wedge \psi_2(\beta) \iff \mathcal{A}_\chi, \beta \models \psi(\beta)$. □

In Section 3.2 we conjectured that #P and #FO(=(...)) are incomparable. Similarly we argue that #FO(⊆) and #FO(=(...)) are most likely incomparable, since C-$\mathsf{sat}^{\neg\emptyset}_{\mathrm{DH}}$ is non-monotone and by Theorem 3.11 included in #FO(⊆).

The properties of #FO(⊆), being (most likely a strict) subclass of #P and having easy decision versions, reminds us of the class TotP. We therefore investigate the relationship between these two classes.

Problem:	C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}}$
Input:	$\chi \in \Sigma_1\mathrm{DH}$
Output:	$\lvert \{ \beta \in \Theta(\chi) \mid \beta \models \chi \text{ and } \beta \neq \emptyset \} \rvert$

Theorem 3.12 ([HKM+19]). *C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}}$ is hard for #FO(⊆) with respect to first-order reductions.*

Proof. The proof is analogous to that of Theorem 3.7 (see also [DKdRV15]). As for dependence logic formulas, there is a normal form for inclusion logic formulas $\varphi(x_1, \dots, x_m)$ [Yan20]:

$$\exists y_1, \dots \exists y_k \forall y_{k+1} \big(\varphi'(y_1, \dots, y_{k+1}) \wedge \psi(x_1, \dots, x_m, y_1, \dots, y_k)\big),$$

where φ' is a conjunction of inclusion atoms and ψ is a quantifier free first-order formula. Thus, the only remaining ingredient needed for the proof is the fact that inclusion atoms $\overline{x} \subseteq \overline{y}$ can be expressed by adding the following type of DH clauses:

$$p_s \rightarrow \bigvee_{\substack{s' \in A^{m+k+l} \text{ and} \\ s(\overline{x}) = s'(\overline{y})}} p_{s'}.$$

Note that the other clause types (∀,∃,⊤,⊥) that were introduced in the proof of Theorem 3.7 are also DualHorn clauses. □

Theorem 3.13 ([HKM+19])**.** $\textsf{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}} \in \textsf{TotP}$.

Proof. Let $\chi \in \Sigma_1\mathrm{DH}$. The machine that witnesses membership in $\textsf{TotP}$ works as follows: Choose a free variable, assign the value 0 to it and simplify the formula (remove all clauses that are already satisfied and all positive occurrences of the variable). Compute in polynomial time if the resulting formula is satisfiable and do the same for the value 1 afterwards. Create one branch for each of those two formulas that is satisfying. In each branch do this recursively for the next variable that is not assigned yet. □

Corollary 3.14 ([HKM+19])**.** $\#\textsf{FO}(\subseteq) \subseteq \textsf{TotP}$.

Proof. Let $\mathrm{C}\text{-}f \in \#\textsf{FO}(\subseteq)$. By Theorem 3.12 it follows $\mathrm{C}\text{-}f \leq^{\textsf{FO}}_{\mathrm{par}} \textsf{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}}$ and thereby $\mathrm{C}\text{-}f \leq^{\textsf{P}}_{\mathrm{par}} \textsf{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}}$ (since $\leq^{\textsf{FO}}_{\mathrm{par}}$ implies $\leq^{\textsf{P}}_{\mathrm{par}}$). Furthermore, since $\textsf{TotP}$ is closed under $\leq^{\textsf{P}}_{\mathrm{par}}$ (see Lemma 2.39) and $\textsf{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}} \in \textsf{TotP}$ by Theorem 3.13, we can conclude $\mathrm{C}\text{-}f \in \textsf{TotP}$.

□

We do not know whether or not the inclusion $\#\textsf{FO}(\subseteq) \subseteq \textsf{TotP}$ is in fact strict, but as we have seen, there is a problem—namely $\textsf{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}}$—that is hard for $\#\textsf{FO}(\subseteq)$ and contained in $\textsf{TotP}$. It is not known if $\textsf{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}}$ is also contained in $\#\textsf{FO}(\subseteq)$ but quantifiers seem not to be expressible in inclusion logic.

3.4 Complete Problems for $\#\cdot\textsf{NP}$

In Section 3.2 we showed $\textsf{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-} \in \#\textsf{FO}(=(\dots))$. We give this result some more weight by showing that $\textsf{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$ is also $\#\cdot\textsf{NP}$-complete with respect to Turing reductions. For this, recall that for a class of quantified boolean formulas P, the problems $\textsf{C-sat}_{\mathrm{P}}$ and $\textsf{C-sat}^{\neg\emptyset}_{\mathrm{P}}$ are defined as follows:

Problem:	$\textsf{C-sat}_{\mathrm{P}}$
Input:	$\chi \in \mathrm{P}$
Output:	$\lvert \{ \beta \in \Theta(\chi) \mid \beta \models \chi \} \rvert$

Problem:	$\textsf{C-sat}^{\neg\emptyset}_{\mathrm{P}}$
Input:	$\chi \in \mathrm{P}$
Output:	$\lvert \{ \beta \in \Theta(\chi) \mid \beta \models \chi \text{ and } \beta \neq \emptyset \} \rvert$

We start by showing $\#\cdot\textsf{NP}$-completeness for $\textsf{C-sat}_{\Sigma_1\mathrm{CNF}}$ and show $\textsf{C-sat}_{\Sigma_1\mathrm{CNF}} \leq^{\textsf{P}}_{\mathrm{T}} \textsf{C-sat}_{\Sigma_1\mathrm{CNF}^-} \leq^{\textsf{P}}_{\mathrm{T}} \textsf{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$ in two separate results afterwards.

Lemma 3.15 ([HKM⁺19]). *$\mathsf{C\text{-}sat}_{\Sigma_1\mathrm{BF}}$ and $\mathsf{C\text{-}sat}_{\Sigma_1\mathrm{CNF}}$ are $\#\cdot\mathsf{NP}$-complete under parsimonious reductions.*

Proof. Aziz et al. [ACMS15] studied the problem $\mathsf{C\text{-}sat}_{\Sigma_1\mathrm{BF}}$ under the name *projected model counting* and showed its membership in $\#\cdot\mathsf{NP}$. Since $\mathsf{C\text{-}sat}_{\Sigma_1\mathrm{CNF}}$ is a restriction of $\mathsf{C\text{-}sat}_{\Sigma_1\mathrm{BF}}$, membership for $\mathsf{C\text{-}sat}_{\Sigma_1\mathrm{CNF}}$ also follows immediately. A simple adaptation of Cook's NP-completeness proof for $\mathsf{D\text{-}sat}_{\mathrm{BF}}$ [Coo71] shows that both problems are hard for $\#\cdot\mathsf{NP}$ with respect to parsimonious reductions. □

In the next result we show $\#\cdot\mathsf{NP}$-completeness for the problem $\mathsf{C\text{-}sat}_{\Sigma_1\mathrm{CNF}^-}$. For the hardness proof, we orient ourselves to some results from Valiant [Val79a, Val79b] . In "The complexity of computing the permanent" [Val79a] Valiant shows that the problem to compute the permanent of a given binary matrix is $\#\mathsf{P}$-complete. In the proof Valiant uses the fact that one can compute the permanent of a given binary matrix by counting (vertex-disjoint) cycle covers of a graph corresponding to the matrix. In "The complexity of Enumeration and reliability problems" Valiant shows $\#\mathsf{P}$-completeness for several counting problems—among them the problem $\mathsf{C\text{-}sat}_{2\mathrm{CNF}^+}$. For this the permanent problem (or the cycle cover problem) is reduced to $\mathsf{C\text{-}sat}_{2\mathrm{CNF}^+}$ with two intermediate steps. The complete chain of reductions looks as follows:

$$\begin{aligned}\mathsf{C\text{-}sat}_{3\mathrm{CNF}} &\leq^{\mathsf{P}}_{\mathrm{T}} \mathsf{C\text{-}cyclecover}\\ &\leq^{\mathsf{P}}_{\mathrm{T}} \mathsf{C\text{-}pmatching}\\ &\leq^{\mathsf{P}}_{\mathrm{T}} \mathsf{C\text{-}matching}\\ &\leq^{\mathsf{P}}_{\mathrm{T}} \mathsf{C\text{-}sat}_{2\mathrm{CNF}^+},\end{aligned}$$

Formally the problems C-cyclecover, C-pmatching, C-matching are defined as follows:

Problem:	C-cyclecover
Input:	$G = (V, E) \in \mathrm{GRAPH}$
Output:	$\lvert \{ E' \subseteq E \mid E' \text{ is a vertex disjoint cycle cover of } G \} \rvert$

Problem:	C-pmatching
Input:	$G = (V_1, V_2, E) \in \mathrm{BIP}$
Output:	$\lvert \{ E' \subseteq E \mid E' \text{ is a perfect matching of } G \} \rvert$

Problem:	C-matching
Input:	$G = (V_1, V_2, E) \in \mathrm{BIP}$
Output:	$\lvert \{ E' \subseteq E \mid E' \text{ is a matching of } G \} \rvert$

Note that this reduction can be modified to also work for C-$\mathsf{sat}_{\mathrm{2CNF}^-}$. To show $\#\cdot\mathsf{NP}$-hardness for C-$\mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$ the idea is to add a $\mathbf{\Sigma_1}$3CNF$^-$ formula to the input of each problem in the chain of reductions above and to express certain properties of the respective inputs in the added formula. We then count only the solutions to the input that also satisfy the added formula. Valiant's reductions are Turing reductions but some of them can be made parsimonious, in our case the extra power from the second input makes this much simpler.

Theorem 3.16 ([HKM$^+$19]). C-$\mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$ *is* $\#\cdot\mathsf{NP}$*-complete under Turing reductions.*

Proof. Membership follows from Lemma 3.15, since C-$\mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$ is a special case of C-$\mathsf{sat}_{\Sigma_1\mathrm{CNF}}$.

For hardness we first reduce C-$\mathsf{sat}_{\mathbf{\Sigma_1}\mathrm{3CNF}}$ to C-($\mathsf{sat}_{\mathrm{3CNF}}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$), and then apply the above mentioned chain of reductions with the added formulas to reduce C-($\mathsf{sat}_{\mathrm{3CNF}}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$) to C-($\mathsf{sat}_{\mathrm{2CNF}^-}, \mathsf{sat}_{\mathbf{\Sigma_1}\mathrm{3CNF}^-}$), which will be further reduced to C-$\mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$. Recall the definition of C-($\mathsf{sat}_{\mathrm{P}_1}, \mathsf{sat}_{\mathrm{P}_2}$) for two sets of quantified Boolean formulas $\mathrm{P}_1, \mathrm{P}_2$:

Problem:	C-($\mathsf{sat}_{\mathrm{P}_1}, \mathsf{sat}_{\mathrm{P}_2}$)
Input:	$\chi_1 \in \mathrm{P}_1, \chi_2 \in \mathrm{P}_2$
Output:	$\lvert \{ \beta \in \Theta(\chi_1) \cup \Theta(\chi_2) \mid \beta \models \chi_1 \text{ and } \beta \models \chi_2 \} \rvert$

All of these reductions will be parsimonious, except for the one from perfect matchings to imperfect matchings. This full chain of reductions looks as follows:

$$\begin{aligned}
\text{C-}\mathsf{sat}_{\mathbf{\Sigma_1}\mathrm{3CNF}} &\leq^{\mathsf{P}}_{\mathrm{par}} \text{C-}(\mathsf{sat}_{\mathrm{3CNF}}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}) && (3.1)\\
&\leq^{\mathsf{P}}_{\mathrm{par}} \text{C-}(\mathsf{cyclecover}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}) && (3.2)\\
&\leq^{\mathsf{P}}_{\mathrm{par}} \text{C-}(\mathsf{pmatching}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}) && (3.3)\\
&\leq^{\mathsf{P}}_{\mathrm{T}} \text{C-}(\mathsf{matching}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}) && (3.4)\\
&\leq^{\mathsf{P}}_{\mathrm{par}} \text{C-}(\mathsf{sat}_{\mathrm{2CNF}^-}, \mathsf{sat}_{\mathbf{\Sigma_1}\mathrm{3CNF}^-}) && (3.5)\\
&\leq^{\mathsf{P}}_{\mathrm{par}} \text{C-}\mathsf{sat}_{\Sigma_1\mathrm{CNF}^-} && (3.6)
\end{aligned}$$

We continue by proving each of these reductions.

(3.1) C-$\mathsf{sat}_{\mathbf{\Sigma_1}\mathrm{3CNF}} \leq^{\mathsf{P}}_{\mathrm{par}}$ C-($\mathsf{sat}_{\mathrm{3CNF}}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$): Let $\chi_0(x_1, \dots, x_k) \in \mathbf{\Sigma_1}$3CNF with $k \in \mathbb{N}$ and

$$\chi_0(x_1, \dots, x_k) = \exists y_1 \dots \exists y_\ell \bigwedge C_i \wedge \bigwedge D_i \wedge \bigwedge E_i,$$

where $\mathsf{vars}(C_i) \subseteq \{x_1, \dots, x_k\}$, $\mathsf{vars}(D_i) \subseteq \{y_1, \dots, y_\ell\}$, $\mathsf{vars}(E_i) \subseteq \{x_1, \dots, x_k\} \cup \{y_1, \dots, y_\ell\}$. We now construct two formulas $\chi_1 \in$ 3CNF and $\chi_2 \in \mathbf{\Sigma_1}$3CNF$^-$ such that $\text{C-}\mathsf{sat}_{\Sigma_1 3\mathrm{CNF}}(\chi_0) = \text{C-}(\mathsf{sat}_{3\mathrm{CNF}}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-})(\chi_1, \chi_2)$. Define

$$\chi_1(x_1, \dots, x_k, e_1, \dots, e_m) = \bigwedge \Big(C_i \wedge \bigwedge (\neg e_i \leftrightarrow E_i|_{\{x_1,\dots,x_k\}}) \Big)$$

and

$$\chi_2(e_1, \dots, e_m) = \exists y_1 \dots \exists y_\ell \bigwedge D_i \wedge \bigwedge (e_i \rightarrow E_i|_{\{y_1,\dots,y_\ell\}}),$$

where m is the number of the clauses E_i, and $C|_V$ denotes the restriction of the clause C to variables in V. More precisely, for a clause $C = \ell_1 \vee \ell_2 \vee \ell_3$ we define

$$C|_V \;:=\; \bigvee_{\substack{i \in \{1,2,3\} \\ \exists x \in V : \ell_i = x \text{ or } \ell_i = \neg x}} \ell_i.$$

Note that in these two formulas the new implications and equivalences can be trivially transformed to 3CNF formulas, and in χ_2 the free variables only occur negatively. Intuitively, using the new variables e_i, the formula χ_1 expresses that the assignment to the variables $x_1, \dots, x_k$ does not satisfy any literal in E_i, and thus, as expressed in χ_2, the clause E_i has to be satisfied by an appropriate assignment to the variables $y_1, \dots, y_\ell$. Since the assignments to the new variables e_i are uniquely determined by the assignments to the variables $x_1, \dots, x_k$, the formula $\chi_1 \wedge \chi_2$ has the same number of satisfying assignments as the original formula χ_0.

(3.2) $\text{C-}(\mathsf{sat}_{3\mathrm{CNF}}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}) \leq^{\mathsf{P}}_{\mathrm{par}} \text{C-}(\mathsf{cyclecover}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-})$: Let $\chi_0(x_1, \dots, x_k) \in$ 3CNF, $\chi_1(x_1, \dots, x_k) \in \Sigma_1\text{3CNF}^-$.

Problem:	$\text{C-}(\mathsf{cyclecover}, \mathsf{sat}_{\Sigma_1\mathrm{CNF}^-})$
Input:	$G = (V, E) \in \mathrm{GRAPH}, \chi \in \Sigma_1\mathrm{CNF}^-$
Output:	$\left\lvert \left\{ E' \subseteq E \;\middle|\; \begin{array}{l} E' \text{ is a vertex disjoint cycle cover} \\ \text{of } G, \mathsf{free}(\chi) = E \text{ and } E' \models \chi \end{array} \right\} \right\rvert$

We map χ_0 to an instance G of C-$\mathsf{cyclecover}$ similar to Valiant [Val79a]. In Valiant's reduction [Val79a], certain pairs of nodes are connected by so-called junctions, which are essentially two edges connecting the nodes in both directions. The goal then is to count only "good" cycle covers, namely those cycle covers that contain at most one edge per junction. In the original proof this is achieved by replacing junctions by a certain gadget. In our case, we can instead use a formula to express the crucial condition: A junction consisting of two edges e_1, e_2 is used appropriately if and only if one of the edges e_1 and e_2 is not contained in the cycle cover.

In Valiant's construction, each satisfying assignment of χ_0 corresponds to exactly one good cycle cover of G, and vice versa. In particular, for each variable x of χ_0, there is a certain edge e in G such that e is contained in each good cycle cover of G if and only if the variable x is assigned to 1 by the corresponding assignment.

Now, let χ_1' be the formula obtained from χ_1 by replacing all occurrences of the free variables by the corresponding edges. Let J be the set of junctions in G, each of which can be given as the set of its edges. Define

$$\chi_1'' \coloneqq \chi_1' \wedge \bigwedge_{\{j_1,j_2\}\in J} (\neg j_1 \vee \neg j_2)$$

Note that the free variables in χ_1'' only occur negatively. Now we have

$$\text{C-}(\mathsf{cyclecover}, \mathsf{sat}_{\Sigma_1\text{CNF}^-})(G, \chi_1'') = \text{C-}(\mathsf{sat}_{3\text{CNF}}, \mathsf{sat}_{\Sigma_1\text{CNF}^-})(\chi_0, \chi_1).$$

(3.3) C-$(\mathsf{cyclecover}, \mathsf{sat}_{\Sigma_1\text{CNF}^-}) \leq^{\mathsf{P}}_{\text{par}}$ C-$(\mathsf{pmatching}, \mathsf{sat}_{\Sigma_1\text{CNF}^-})$:

Problem:	C-$(\mathsf{pmatching}, \mathsf{sat}_{\Sigma_1\text{CNF}^-})$
Input:	$G = (V_1, V_2, E) \in \text{BIP}, \chi \in \Sigma_1\text{CNF}^-$
Output:	$\left\lvert\left\{ E' \subseteq E \;\middle\vert\; \begin{array}{l} E' \text{ is a perfect matching of} \\ G, \mathsf{free}(\chi) = E \text{ and } E' \models \chi \end{array} \right\}\right\rvert$

Following the 1-to-1 correspondence between cycle covers of directed graphs and perfect matchings of bipartite graphs, the reduction can be given as follows:

$$\big((V,E),\chi\big) \mapsto \big((V, \{v' \mid v \in V\}, \{\{v_1, v_2'\} \mid (v_1, v_2) \in E\}), \chi'\big),$$

where χ' is obtained from χ by replacing all occurrences of variables (v_1, v_2) by the corresponding new variables $\{v_1, v_2'\}$.

(3.4) C-$(\mathsf{pmatching}, \mathsf{sat}_{\Sigma_1\text{CNF}^-}) \leq^{\mathsf{P}}_{\text{T}}$ C-$(\mathsf{matching}, \mathsf{sat}_{\Sigma_1\text{CNF}^-})$: Let $G = (V_1, V_2, E)$ be a bipartite graph with $E = \{e_1, \ldots, e_n\}$ and $\chi(e_1, \ldots, e_n) \in \mathbf{\Sigma_1}\text{3CNF}^-$.

Problem:	C-$(\mathsf{matching}, \mathsf{sat}_{\Sigma_1\text{CNF}^-})$
Input:	$G = (V_1, V_2, E) \in \text{BIP}, \chi \in \Sigma_1\text{CNF}^-$
Output:	$\lvert \{ E' \subseteq E \mid E' \text{ is a matching of } G, \mathsf{free}(\chi) = E \text{ and } E' \models \chi \} \rvert$

For the reduction C-$\mathsf{pmatching} \leq^{\mathsf{P}}_{\text{T}}$ C-$\mathsf{matching}$, Valiant constructs bipartite graphs G_k for $1 \leq k \leq |V_1| + 1$ from G by adding copies of all nodes in V_1 as follows:

$$\begin{aligned} G_k &\coloneqq (V_{1,k}, V_2, E_k), \text{ where} \\ V_{1,k} &\coloneqq V_1 \cup \{u_{ij} \mid 1 \leq i \leq |V_1|, 1 \leq j \leq k\} \text{ and} \\ E_k &\coloneqq E \cup \big\{\{u_{ij}, v_i\} \mid 1 \leq i \leq |V_1|, 1 \leq j \leq k\big\} \end{aligned}$$

Let A_r be the number of matchings of G of size $|V_1| - r$. Then G_k has exactly $\sum_{r=0}^{|V_1|} A_r \cdot (k+1)^r$ matchings. Using the number of matchings of all graphs G_k yields a system of linear equations that allows us to compute A_0, the number of perfect matchings of G. Note that each matching of G corresponds to a number of matchings in each G_k (those consisting only of copies of the edges from the original matching).

To compute the number of perfect matchings E' of G with $E' \models \chi$, we now associate each graph G_k with a formula χ_k such that $E'' \models \chi_k$ holds for those matchings E'' of G_k corresponding to a matching E' of G with $E' \models \chi$. Let $e_i = \{v_1, v_2\}$ be an edge of G. A matching E'' of G_k corresponds to a matching E' of G that does not use edge e_i if and only if it does neither use the edge $\{v_1, v_2\}$ nor any of the edges e_{ij}, where $e_{ij} = \{u_{1,j}, v_2\}$. Formally this can be written as

$$E' \models \neg e_i \iff E'' \models \neg\{v_1, v_2\} \wedge \bigwedge_{1 \leq j \leq k} \neg e_{ij}.$$

Now in any clause $(\neg e_i \vee \ell_1 \vee \ell_2)$ where ℓ_1 and ℓ_2 are literals of bound variables of χ we can replace $\neg e_i$ by $\bigwedge_{1 \leq j \leq k} \neg e_{ij}$. The resulting formula is equivalent to

$$\bigwedge_{1 \leq j \leq k} (\neg e_{ij} \vee \ell_1 \vee \ell_2),$$

which is of the desired form. Similarly we can replace any clause of the form $(\neg e_{i_1} \vee \neg e_{i_2} \vee \ell_1)$ by $(\bigwedge_{1 \leq j \leq k} \neg e_{i_1,j} \vee \bigwedge_{1 \leq j \leq k} \neg e_{i_2,j} \vee \ell_1)$, resulting in the formula

$$\bigwedge_{(j_1,j_2) \in \{1,\dots,k\}^2} (\neg e_{i_1,j_1} \vee \neg e_{i_2,j_2} \vee \ell_1).$$

Analogously we can also handle clauses of the form $(\neg e_{i_1} \vee \neg e_{i_2} \vee \neg e_{i_3})$.

Let χ' be χ after applying the above changes. We have that any matching E'' of G_k corresponds to a matching of E' of G with $E' \models \chi$ if and only if $E'' \models \chi'$. Now, we can proceed as Valiant [Val79b]: Let A'_r be the number of matchings E' of G of size $|V_1| - r$ with $E' \models \chi$. Then G_k has exactly $\sum_{r=0}^{|V_1|} A'_r \cdot (k+1)^r$ matchings E'' with $E'' \models \chi'$. Using the number of such matchings for all graphs G_k we get a system of linear equations allowing us to compute A'_0, the number of perfect matchings E' of G with $E' \models \chi$.

(3.5) C-($\mathsf{matching}, \mathsf{\Sigma_1 3cnf}^-$) $\leq^{\mathsf{P}}_{\mathrm{par}}$ C-($\mathsf{sat}_{\mathrm{2CNF}^-}, \mathsf{sat}_{\mathsf{\Sigma_1}\mathrm{3CNF}^-}$): Let $G = (V_1, V_2, E)$ be a bipartite graph with $E = \{e_1, \dots, e_n\}$ and $\chi_0(e_1, \dots, e_n) \in \mathsf{\Sigma_1}\mathrm{3CNF}^-$. The reduction is completely analogously to the proof by Valiant: We define a 2CNF$^-$ formula $\chi_1(e_1, \dots, e_n)$ expressing that each node of the graph is only matched once as:

$$\chi_1(e_1, \dots, e_n) \coloneqq \bigwedge_{\substack{(e_1,e_2) \in E \times E \\ e_1 \neq e_2 \text{ and } e_1 \cap e_2 \neq \emptyset}} \neg e_1 \vee \neg e_2$$

Then

$$\text{C-}(\mathsf{matching}, \mathsf{sat}_{\mathsf{\Sigma_1}\mathrm{CNF}^-})(G, \chi_0) = \text{C-}(\mathsf{sat}_{\mathrm{2CNF}^-}, \mathsf{sat}_{\mathsf{\Sigma_1}\mathrm{3CNF}^-})(\chi_1, \chi_0).$$

Note that no change is made to the $\Sigma_1$3CNF$^-$ formula χ_0 in this reduction.

(3.6) C-$(\mathsf{sat}_{2\mathrm{CNF}^-}, \mathsf{sat}_{\Sigma_1 3\mathrm{CNF}^-}) \leq^{\mathsf{P}}_{\mathrm{par}}$ C-$\mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$: Let $\chi_0(x_1, \dots, x_n) \in 2\mathrm{CNF}^-$ and $\chi_1(x_1, \dots, x_n) \in \Sigma_1 3\mathrm{CNF}^-$. Clearly,

$$\text{C-}(\mathsf{sat}_{2\mathrm{CNF}^-}, \mathsf{sat}_{\Sigma_1 3\mathrm{CNF}^-})(\chi_0, \chi_1) = \text{C-}\mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}(\exists y_1 \dots \exists y_k(\chi_0 \wedge \chi_1')). \qquad \square$$

Theorem 3.17 ([HKM+19]). *The problem* C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$ *is* $\#\cdot\mathsf{NP}$*-complete under Turing reductions.*

Proof. Membership follows again from Lemma 3.15.

For hardness we give a Turing reduction from C-$\mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$. From Theorem 3.16 it follows that C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$ is $\#\cdot\mathsf{NP}$-hard. Let $\chi_0(x_1, \dots, x_n) \in \Sigma_1\mathrm{CNF}^-$. Replace every free variable x_i in χ_0 by the constant $\bot$, and simplify the formula by removing all literals that are equivalent to $\bot$ and all clauses that are equivalent to $\top$. Denote the resulting formula by χ_0', and

$$\chi_1 := \chi_0' \wedge (\neg x_{n+1} \vee \neg x_{n+2}),$$

where x_{n+1} and x_{n+2} are the only free variables in χ_1. Observe that if $\chi_0(x_1, \dots, x_n)$ is satisfiable then it is also satisfied by the empty assignment. We use C-$\mathsf{sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$ as an oracle to compute the number of satisfying assignments of χ_1, not counting the empty assignment. The answer can only be either 0 or 2. If the answer is 0, we conclude that $\chi_0(x_1, \dots, x_n)$ is not satisfiable and therefore the number of satisfying assignments of χ_0 is 0. If the answer is 2, we know that $\chi_0(x_1, \dots, x_n)$ is satisfiable. Now we can ask the oracle again for the number k of the satisfying assignments of $\chi_0(x_1, \dots, x_n)$. The actual number of satisfying assignments of $\chi_0(x_1, \dots, x_n)$ is then $k+1$ (as the oracle does not count the empty assignment). $\square$

3.5 Summary

In Figure 3.1, we present a class diagram which summarizes all our results regarding counting complexity classes. The diagram contains new counting classes we defined using team logics and compares them to previously known counting classes. We conjecture that all these subsets are in fact strict, but these conjectures are bound to assumptions (e.g. $\mathsf{P} \neq \mathsf{NP}$ or C-$\mathsf{sat}^{\neg\emptyset}_{\mathrm{DH}} \notin \#\mathsf{FO}(=(\dots)))$. Recall that by Proposition 2.47 $\#\mathsf{FOT} = \#\mathsf{P}$.

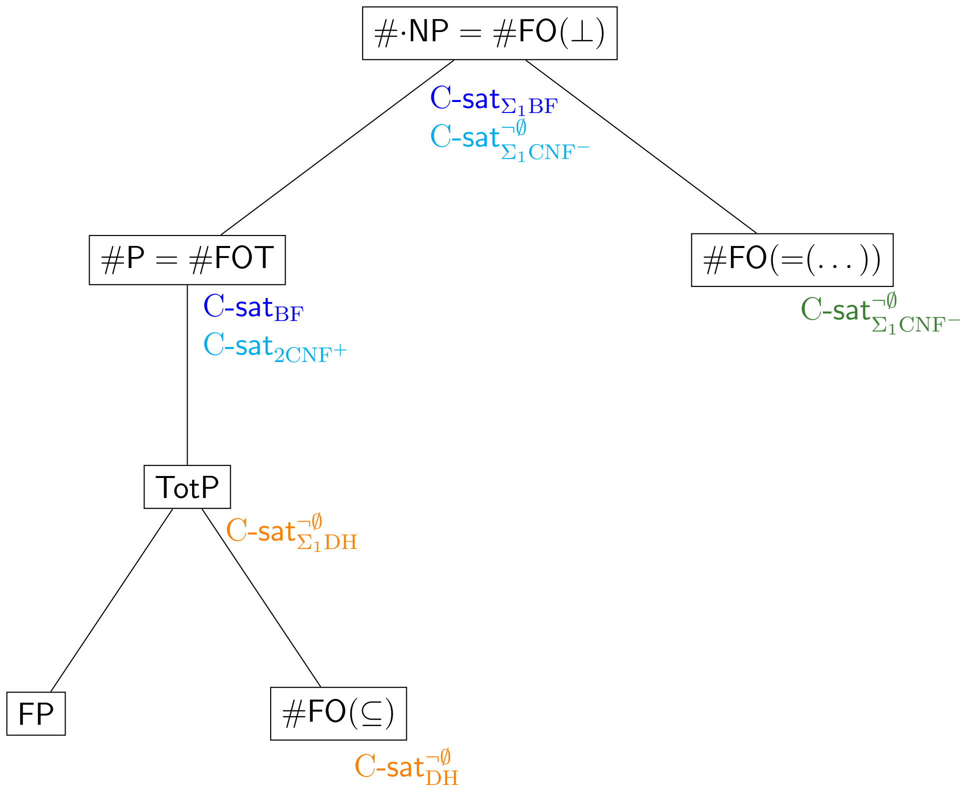
$\#\cdot\mathrm{NP} = \#\mathrm{FO}(\perp)$
$\text{C-sat}_{\Sigma_1\mathrm{BF}}$
$\text{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$
$\#\mathrm{P} = \#\mathrm{FOT}$
$\#\mathrm{FO}(=(\dots))$
$\text{C-sat}_{\mathrm{BF}}$
$\text{C-sat}_{2\mathrm{CNF}^+}$
$\text{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{CNF}^-}$
TotP
$\text{C-sat}^{\neg\emptyset}_{\Sigma_1\mathrm{DH}}$
FP
$\#\mathrm{FO}(\subseteq)$
$\text{C-sat}^{\neg\emptyset}_{\mathrm{DH}}$

Figure 3.1: Class diagram of considered counting complexity classes. An edge between two nodes in the diagram denotes a subset relation between the two nodes (from bottom to top). Orange marked problems are contained in the respective class, blue marked problems are complete for the respective class with respect to parsimonious reductions, light blue marked problems are complete for the respective class with respect to Turing reductions and green marked problems are complete with respect to first-order reductions.

4 Enumeration Problems in Team Logics

Like in the decision and counting setting we classify the enumeration complexity classes that we defined based on independence, dependence and inclusion logic. As we will see, the results here are very similar to the ones in the decision case, since by Theorem 2.54 there is a simple way to translate hardness from the decision to the enumeration setting. To study the framework of hardness in the enumeration setting a bit more, we analyse the complexity of satisfiability problems that are restricted to optimal solutions.

In this section, we mainly have two types of results, membership and hardness results. For showing membership results we provide an algorithm (a RAM) that has access to a (suitable) decision oracle and computes the desired solutions. Afterwards we show that every solution is output exactly once and argue that the delay satisfies the given bounds. To prove hardness for an enumeration problem we show hardness for a corresponding decision problem and translate it to the enumeration problem.

Some of our algorithms assume an order on the assignments. Here any total order, e.g. lexicographical order, is suitable. Furthermore, for a given team X we denote by $\mathsf{max}(X)$ the greatest assignment $s \in X$ with respect to our order. In our first result we show that E-$\mathsf{sat}^{\text{team}}_{\varphi} \in \mathsf{DelNP}$ for $\varphi \in \mathsf{FO}(\mathrm{T})$ and $\mathrm{T} \in \mathfrak{G}_{\mathsf{NP}}$.

Problem:	E-$\mathsf{sat}^{\text{team}}_{\varphi}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \,\}$

For this we provide an algorithm that has access to two oracles, namely V-$\mathsf{sat}^{\text{team}}_{\varphi}$ and D-$\mathsf{extendteam}_{\varphi}$.

Problem:	V-$\mathsf{sat}^{\text{team}}_{\varphi}$
Input:	$\mathcal{A} \in \mathrm{STRUC}, X' \in \mathrm{TEAM}(\varphi)$
Question:	$X' \in \{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \,\}$?

Problem:	D-$\mathsf{extendteam}_\varphi$
Input:	$\mathcal{A} \in \text{STRUC}, X \in \text{TEAM}(\varphi), Y \in \text{TEAM}(\varphi)$
Question:	$\left\{ X' \in \text{TEAM}(\mathcal{A}, \varphi) \;\middle|\; \begin{array}{l} \mathcal{A} \models_{X'} \varphi,\, X \subsetneq X' \\ \text{and } X' \cap Y = \emptyset \end{array} \right\} \neq \emptyset?$

We use the oracle D-$\mathsf{extendteam}_\varphi$ as our torchlight (like D-sat_{BF} in Example 2.51). That means we can query the oracle at any point whether the current team can be extended to a solution. The idea of the extra input Y is to keep track of solutions that have been output before. More precisely, we use Y to store certain assignments that may not appear any more for further solutions, since all solutions containing those assignments have been output before. Since Y is a set of assignments it is formally a team but Y does not correspond to any solutions.

Since $\text{T} \in \mathfrak{G}_{\mathsf{NP}}$, we know that V-$\mathsf{sat}_\varphi^{\text{team}} \in \mathsf{NP}$ and therefore that it is a suitable oracle for showing DelNP membership. Moreover D-$\mathsf{extendteam}_\varphi$ is in NP as well: Guess team X' and make the guesses needed to verify $\mathcal{A} \models_{X'} \varphi$ in polynomial time. The latter is possible since V-$\mathsf{sat}_\varphi^{\text{team}} \in \mathsf{NP}$. Afterwards check $X \subsetneq X'$ and $X' \cap Y = \emptyset$ (in polynomial time).

Formally, our algorithm may only use one oracle, but since D-$\mathsf{extendteam}_\varphi$ and V-$\mathsf{sat}_\varphi^{\text{team}}$ are both in NP we can reduce them to an NP-complete problem D-f which we can then use as oracle. For readability we give the algorithm direct access to the oracles D-$\mathsf{extendteam}_\varphi$ and V-$\mathsf{sat}_\varphi^{\text{team}}$ instead of D-f. Later we will provide more algorithms with two oracles which we will handle this way.

Theorem 4.1 ([HMMV22]). E-$\mathsf{sat}_\varphi^{\text{team}} \in \mathsf{DelNP}$ *for* $\varphi \in \mathsf{FO}(\text{T})$ *and* $\text{T} \subseteq \mathfrak{G}_{\mathsf{NP}}$.

Proof. We start with the empty team and add assignments step by step. At every step we ask the oracle V-$\mathsf{sat}_\varphi^{\text{team}}$ and output the current team depending on the answer. Afterwards we ask the oracle D-$\mathsf{extendteam}_\varphi$ whether we have to search for more solutions.

A formal description of our method is given in Algorithm 1. The algorithm gets a structure $\mathcal{A}$ and a team X as inputs and outputs all satisfying teams X' with $X \subsetneq X'$ and $X' \setminus X \subseteq \{ s \in \text{dom}(\mathcal{A})^{|\mathsf{free}(\varphi)|} \mid s > \mathsf{max}(X) \}$, that is, X' only contains new assignments that are greater (with respect to to our order) than the greatest assignment in X. The algorithm computes such teams by using recursive calls where in each call exactly one assignment $s > \mathsf{max}(X)$ is added to X. We run the algorithm on input $(\mathcal{A}, \emptyset)$ to get all satisfying teams.

The algorithm outputs every solution at least once: Suppose we would skip line 3 of the algorithm. Then the algorithm would branch on any assignments s that are greater than $\mathsf{max}(X)$ for the considered team X and therefore would consider all teams at one point. Line 3 stops some of these recursive calls from happening but only when the torchlight tells us that there are no solutions to be found in those calls. Therefore all solutions will be found.

Algorithm 1: $\textsf{E-sat}^{\text{team}}_{\varphi} \in \textsf{DelNP}$ for $\varphi \in \textsf{FO}(T)$

```
  Oracles: $\textsf{D-extendteam}_{\varphi}$ and $\textsf{V-sat}^{\text{team}}_{\varphi}$
  Function ESuperteams(structure $\mathcal{A}$, team $X$)
1     if $\textsf{V-sat}^{\text{team}}_{\varphi}(\mathcal{A}, X)$ then output $X$
2     $Y = \bigcup_{s<\max(X) \land s \notin X} s$
3     if $\textsf{D-extendteam}_{\varphi}(\mathcal{A}, X, Y)$ then
4         forall $s > \textsf{max}(X)$ do
5             ESuperteams($\mathcal{A}, X \cup \{ s \}$)
```

On the other hand in the recursive calls the teamsize grows and any two parallel recursive calls search through distinct sets of teams, therefore no solution will be output twice.

We now argue that the delay of the algorithm is polynomial. Suppose the only solution is the full team X_{full}, which is a worst case for both the precomputation and postcomputation. In that case the algorithm makes $|X_{\text{full}}|$ recursive calls until the first solution is output and therefore the precomputation takes polynomial time. After the output there are no other solutions to be found but the algorithm still has to end some recursive calls. An upper bound for the number of these calls is $|X_{\text{full}}| \cdot |X_{\text{full}}|$ and thus the postcompuation takes polynomial time as well. Now consider the time between the outputs of two consecutive solutions. Here, a worst case is that X_{full} and $\{s\}$ are the only solutions, where s is the highest assignment with respect to the considered order. In that case the number of recursive calls that have to be handled between the first and second solution would be bounded by $|X_{\text{full}}| \cdot |X_{\text{full}}|$ again. □

We have shown $\textsf{DelNP}$ membership of $\textsf{E-sat}^{\text{team}}_{\varphi}$ for formulas of team logics that allow arbitrary sets of $\textsf{NP}$-verifiable generalized atoms and by this deduce $\textsf{DelFO}(\text{T}) \subseteq \textsf{DelNP}$ for $\text{T} \subseteq \{\perp, =(\dots), \subseteq\}$. This result therefore holds especially for independence, dependence and inclusion logic. As we will see, the other direction only holds for independence and dependence logic. For this we show that there are formulas $\varphi \in \textsf{FO}(\text{T})$ for $\text{T} \in \{\perp, =(\dots)\}$ such that $\textsf{D-sat}^{\text{team}}_{\varphi}$ is $\textsf{NP}$-hard and use Corollary 2.55 afterwards to conclude $\textsf{DelNP}$-hardness for $\textsf{E-sat}^{\text{team}}_{\varphi}$. This yields the desired result $\textsf{DelNP} \subseteq \textsf{FO}(\text{T})$.

Problem:	$\textsf{D-sat}^{\text{team}}_{\varphi}$
Input:	$\mathcal{A} \in \text{STRUC}$
Question:	$\{ X \in \text{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi \text{ and } X \neq \emptyset \} \neq \emptyset$?

Theorem 4.2 ([HMMV22]). *Let* $\mathrm{T} \subseteq \{ =(\dots), \perp \}$ *with* $\mathrm{T} \neq \emptyset$. *There exists a formula* $\varphi \in \mathsf{FO}(\mathrm{T})$ *such that the problem* $\text{D-}\mathsf{sat}^{\text{team}}_{\varphi}$ *is* NP*-hard.*

Proof. We show the result for $\mathrm{T} = \{ \perp \}$. The proof for $\mathrm{T} = \{ =(\dots) \}$ works analogously by reducing from the NP-complete problem $\text{D-}\mathsf{sat}_{\Sigma_1\text{CNF}^-}$. Since the dependence atom can be expressed using the independence atom the case of $\{ \perp, =(\dots) \}$ is the same as $\{ \perp \}$.

We reduce from the NP-complete problem $\text{D-}\mathsf{sat}_{\text{CNF}}$ to the problems $\text{D-}\mathsf{sat}^{\text{rel}}_{\psi}$ and $\text{D-}\mathsf{sat}^{\text{rel},\neg\emptyset}_{\psi'}$ for some $\psi, \psi' \in \Sigma^1_1$.

By Proposition 2.24 we get that $\text{D-}\mathsf{sat}^{\text{team}}_{\varphi}$ is NP-hard, for a formula $\varphi \in \mathsf{FO}(\perp)$. Let us recall the definition of those problems. For a class P of quantified boolean formulas the problem $\text{D-}\mathsf{sat}_{\mathrm{P}}$ is defined as follows:

Problem:	$\text{D-}\mathsf{sat}_{\mathrm{P}}$
Input:	$\chi \in \mathrm{P}$
Question:	$\{ \beta \in \Theta(\chi) \mid \beta \models \chi \} \neq \emptyset$?

Furthermore, for second order formulas ψ, the problems $\text{D-}\mathsf{sat}^{\text{rel}}_{\psi}$ and $\text{D-}\mathsf{sat}^{\text{rel},\neg\emptyset}_{\psi}$ are defined in the following way:

Problem:	$\text{D-}\mathsf{sat}^{\text{rel}}_{\psi}$
Input:	$\mathcal{A} \in \text{STRUC}$
Question:	$\{ R \in \text{REL}(\mathcal{A}, \psi) \mid \mathcal{A}, R \models \psi \} \neq \emptyset$?

Problem:	$\text{D-}\mathsf{sat}^{\text{rel},\neg\emptyset}_{\psi}$
Input:	$\mathcal{A} \in \text{STRUC}$
Question:	$\{ R \in \text{REL}(\mathcal{A}, \psi) \mid \mathcal{A}, R \models \psi \text{ and } R \neq \emptyset \} \neq \emptyset$?

Let $\chi(x_1, \dots, x_n) = \bigwedge_i^m C_i$ be a propositional formula in conjunctive normal form, with $C_i = \bigvee_j l_{i,j}$. We encode χ via the τ_{CNF}-structure

$$\mathcal{A}_\chi = \{ \{ x_1, \dots, x_n, C_1, \dots, C_m \}, V^{\mathcal{A}}, P^{\mathcal{A}}, N^{\mathcal{A}} \}.$$

We define the Σ^1_1 formula $\psi(R) \coloneqq \psi_1 \wedge \psi_2(R)$, where ψ_1 and $\psi_2(R)$ are defined as follows:

$$\psi_1 \coloneqq \forall C \forall x \Big(\big(P(C,x) \to \neg V(C) \wedge V(x)\big) \wedge \big(N(C,x) \to \neg V(C) \wedge V(x)\big) \Big)$$

$$\psi_2(R) \coloneqq \forall C \Big(\neg V(C) \to \exists x \big(P(C,x) \wedge R(x)\big) \vee \big(N(C,x) \wedge \neg R(x)\big) \Big)$$

The formula ψ_1 ensures that the input structure is an encoding of a CNF formula and $\psi_2(R)$ that there is a one-to-one correspondence between relations satisfying R and satisfying assignments β for formula χ. Now, we have that

$$\exists R\colon \mathcal{A}_\chi, R \models \psi(R) \iff \exists \beta\colon \beta \models \chi,$$

showing $\text{D-}\mathsf{sat}_{\text{CNF}} \leq_m^{\mathsf{P}} \text{D-}\mathsf{sat}_\psi^{\text{rel}}$.

Next, we will show NP-hardness for $\text{D-}\mathsf{sat}_{\psi'}^{\text{rel},\neg\emptyset}$. This follows from an easy reduction from $\text{D-}\mathsf{sat}_\psi^{\text{rel}}$ to $\text{D-}\mathsf{sat}_{\psi'}^{\text{rel},\neg\emptyset}$ which holds for all $\psi \in \Sigma_1^1$. Let $\psi'(R) = \psi(R) \vee \psi(\emptyset)$. Now, for all structures $\mathcal{A}$ we claim that

$$\exists R\colon \mathcal{A}, R \models \psi(R) \iff \exists R' \neq \emptyset\colon \mathcal{A}, R' \models \psi'(R').$$

"$\implies$": If $\mathcal{A}, R \models \psi(R)$ holds for $R = \emptyset$, then $\mathcal{A}, R' \models \psi'(R')$ holds for any $R' \neq \emptyset$. If $\mathcal{A}, R \models \psi(R)$ for any $R \neq \emptyset$, then $\mathcal{A}, R \models \psi'(R)$ also holds.

"$\impliedby$": Since $\mathcal{A}, R \not\models \psi(R)$ for all R, we have $\mathcal{A}, \emptyset \not\models \psi(R)$ in particular. This immediately shows $\mathcal{A}, R \not\models \psi'(R)$ for all R. □

Corollary 4.3. *Let* $\mathrm{T} = \{\perp, {=}(\dots)\}$, *then there is a formula* $\varphi \in \mathsf{FO}(\mathrm{T})$ *such that* $\text{E-}\mathsf{sat}_\varphi^{\text{team}}$ *is* DelNP*-complete. Furthermore* $\mathsf{DelFO}(\mathrm{T}) = \mathsf{DelNP}$.

Proof. Again we only argue for the independence logic case, the proof for dependence logic case works analogously. By Corollary 2.15 the independence atom is NP-verifiable. Hence Theorem 4.1 is applicable and we can conclude $\text{E-}\mathsf{sat}_{\varphi'}^{\text{team}} \in \mathsf{DelNP}$ for all formulas $\varphi' \in \mathsf{FO}(\perp)$ and thus $\mathsf{DelFO}(\perp) \subseteq \mathsf{DelNP}$.

For the other direction we first conclude that by Theorem 4.2 and Corollary 2.55 there is a formula $\varphi \in \mathsf{FO}(\perp)$ such that $\text{E-}\mathsf{sat}_\varphi^{\text{team}}$ is DelNP-hard and thus DelNP-complete. Since $\text{E-}\mathsf{sat}_\varphi^{\text{team}}$ is DelNP-hard it follows that any enumeration problem $\text{E-}f \in \mathsf{DelNP}$ is reducible to $\text{E-}\mathsf{sat}_\varphi^{\text{team}}$ and therefore

$$\text{E-}f \in \left[\text{E-}\mathsf{sat}_\varphi^{\text{team}}\right]^{\leq_{\mathsf{DelP}}} \subseteq \left[\bigcup_{\varphi' \in \mathsf{FO}(\perp)} \text{E-}\mathsf{sat}_{\varphi'}^{\text{team}}\right]^{\leq_{\mathsf{DelP}}} = \mathsf{DelFO}(\perp).$$

Since $\mathsf{DelFO}(\perp) \subseteq \mathsf{DelNP}$ and any $\text{E-}f \in \mathsf{DelNP}$ is also in $\mathsf{DelFO}(\perp)$, we can conclude $\mathsf{DelFO}(\perp) = \mathsf{DelNP}$.

□

Recall that we defined $\mathsf{DelFO}(\perp)$ as $\left[\bigcup_{\varphi \in \mathsf{FO}(\perp)} \text{E-}\mathsf{sat}_\varphi^{\text{team}}\right]^{\leq_{\mathsf{DelP}}}$. We can now simplify this definition to $\mathsf{DelFO}(\perp) = \left[\text{E-}\mathsf{sat}_\varphi^{\text{team}}\right]^{\leq_{\mathsf{DelP}}}$, where φ is the formula from Corollary 4.3. The same holds for dependence logic.

Having classified DelNP, our next task is to capture DelP with inclusion logic. We start by showing $\text{E-}\mathsf{sat}_\varphi^{\text{team}} \in \mathsf{DelP}$ for arbitrary inclusion logic formulas. For this we take advantage of the fact that, by Proposition 2.42, $\text{S-}\mathsf{maxsubteam}_\varphi \in \mathsf{FP}$ for inclusion logic formulas φ.

Theorem 4.4 ([HMMV22]). *For any formula $\varphi \in \mathsf{FO}(\subseteq)$ it holds that* $\text{E-}\mathsf{sat}^{\text{team}}_{\varphi} \in \mathsf{DelP}$.

Proof. We will search through the set of all teams in a top down manner for solutions using $\text{S-}\mathsf{maxsubteam}_{\varphi}$ as our torchlight. Note that formally $\text{S-}\mathsf{maxsubteam}_{\varphi}$ can not be used as an oracle since it is not a decision problem, but as the problem is included in FP (see Proposition 2.42) we can compute it directly and therefore do not need an oracle.

We start by computing the maximal satisfying team X and output it. Afterwards we split the set of subteams of X in two halves, the ones that contain s and the ones that do not. Here s is one (arbitrary) assignment in X. We look for additional solutions in the second half by starting a recursive call of our method with the input $X \setminus s$. To find the solutions in the first half we again split the remaining teams in half, this time in the set of teams that contain s' (and s) and the ones that do not contain s' (but s). Then we start a recursive call for the second half and continue as above. If at some point the maximal team is the empty team we output nothing and do not start any new recursive calls.

Algorithm 2 is a formal description of this method. It is a recursive algorithm that computes all satisfying subteams $X' \neq \emptyset$ of X with $Y \subseteq X'$ on input $(\mathcal{A}, X, Y)$. To compute all satisfying subteams for a given $\mathcal{A}$, we run this algorithm on input $(\mathcal{A}, X_{\text{full}}, \emptyset)$.

Algorithm 2: $\text{E-}\mathsf{sat}^{\text{team}}_{\varphi} \in \mathsf{DelP}$ for $\varphi \in \mathsf{FO}(\subseteq)$

```
Function ESubteams(structure $\mathcal{A}$, teams $X, Y$)
1   $X \leftarrow \text{S-}\mathsf{maxsubteam}_{\varphi}(\mathcal{A}, X)$
2   if $X \neq \emptyset \wedge Y \subseteq X$ then
3       output $X$
4       for $s \in X$ do
5           $Y = Y \cup \{ s' \mid s' < s \wedge s' \in X \}$
6           ESubteams($\mathcal{A}, X \setminus \{ s \}, Y$)
```

The algorithm outputs every solution at least once: The first output X is the maximal satisfying team and due to union closure all other satisfying teams are subsets of X. In the recursive calls the set of subteams of X is systematically looked through for any new solutions. The search in a recursive call is only stopped when the maximal team is the empty team or $Y \not\subseteq X$, i.e. either there is no solution left to find in the corresponding subset or we have considered it before.

The algorithm outputs every solution at most once: Suppose that there is a team X that is output more than once. In the recursive call where X is output for the first time the algorithm starts new recursive calls and we know that these can not lead to the output X again, since only subsets of X are considered. That means the recursive calls that yield the solution X must originate from the same recursive call. On the other hand we know that this can not happen since the search space of these recursive calls are distinct, which leads to a contradiction. There are no more cases to consider which means that the claim was wrong. It follows that the algorithms outputs every solution exactly once as desired.

The precomputation takes polynomial time since S-$\mathsf{maxsubteam}_\varphi \in \mathsf{FP}$. For the postcomputation consider the worst case: For every solution that is output, only the first recursive call leads to another solution. Thus the algorithm outputs all solutions in direct succession at the beginning and has no solutions left to output afterwards. We now argue that there are only polynomially many recursive calls left at that point. The number of solutions that are output in this case is at most $|X_{\text{full}}|$. For a given solution X the algorithm makes at most $|X| \leq |X_{\text{full}}|$ recursive calls. Therefore, the algorithm makes at most $|X_{\text{full}}| \cdot |X_{\text{full}}|$ recursive calls which is polynomial. Every recursive call that does not lead to a solution only takes polynomial time, since it terminates at line 2 of the algorithm (and does not make any new recursive calls). It follows that the algorithm makes polynomially many recursive calls that each take polynomial time which in total takes polynomial time. For the delay between two outputs a similar worst case can be found. □

Corollary 4.5. E-$\mathsf{sat}^{\text{team}}_\varphi$ *is* DelP*-complete for any formula* $\varphi \in \mathsf{DelFO}(\subseteq)$*. Furthermore,* $\mathsf{DelFO}(\subseteq) = \mathsf{DelP}$*.*

Proof. By Theorem 4.4 we know that E-$\mathsf{sat}^{\text{team}}_\varphi \in \mathsf{DelP}$ for any formula $\varphi \in \mathsf{FO}(\subseteq)$ and therefore $\mathsf{DelFO}(\subseteq) \subseteq \mathsf{DelP}$. Furthermore any problem in DelP is also DelP-complete (see Remark 2.56). Hence, we can conclude that E-$\mathsf{sat}^{\text{team}}_\varphi$ is DelP-complete. Since E-$\mathsf{sat}^{\text{team}}_\varphi$ is DelP-hard it follows that any enumeration problem E-$f \in \mathsf{DelP}$ is reducible to E-$\mathsf{sat}^{\text{team}}_\varphi$ and therefore

$$\text{E-}f \in \left[\text{E-}\mathsf{sat}^{\text{team}}_\varphi\right]^{\leq_{\mathsf{DelP}}} \subseteq \left[\bigcup_{\varphi' \in \mathsf{FO}(\subseteq)} \text{E-}\mathsf{sat}^{\text{team}}_{\varphi'}\right]^{\leq_{\mathsf{DelP}}} = \mathsf{DelFO}(\subseteq).$$

Since $\mathsf{DelFO}(\subseteq) \subseteq \mathsf{DelP}$ and any E-$f \in \mathsf{DelP}$ is also in $\mathsf{DelFO}(\subseteq)$, we can conclude $\mathsf{DelFO}(\subseteq) = \mathsf{DelP}$. □

Similar to the independence and dependence cases we can now simplify the definition of $\mathsf{DelFO}(\subseteq)$ to $\left[\text{E-}\mathsf{sat}^{\text{team}}_\varphi\right]^{\leq_{\mathsf{DelP}}}$, where $\varphi \in \mathsf{FO}(\subseteq)$.

Figure 4.1 summarises the results we have established in the enumeration setting so far.

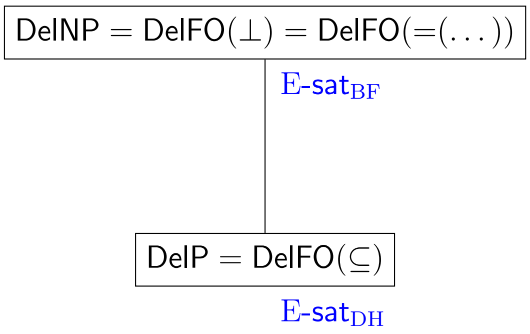

Figure 4.1: Class diagram of considered enumeration complexity classes. An edge between two nodes denotes a subset relation from bottom to top. The blue marked problems are complete for the respective class with respect to $\leq_{\mathsf{DelP}}$ reductions.

4.1 Enumeration of Optima in Team Logics

Having characterised the enumeration complexity classes defined in this work, we now like to study other satisfiability problems based on team logics. These problems arise when restricting the solutions to those that are minimal/maximal or a minimum/maximum. In this section we present results for different team logics and we dedicate the next section to inclusion logic. The problems we want to study in both sections are formally defined as follows:

Problem:	$\mathsf{E\text{-}maxsat}^{\mathrm{team}}_{\varphi}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid X$ is a maximal team for φ in $\mathcal{A} \,\}$

Problem:	$\mathsf{E\text{-}minsat}^{\mathrm{team}}_{\varphi}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid X$ is a minimal team for φ in $\mathcal{A} \,\}$

Problem:	$\mathsf{E\text{-}cmaxsat}^{\mathrm{team}}_{\varphi}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid X$ is a maximum team for φ in $\mathcal{A} \,\}$

Problem:	$\mathsf{E\text{-}cminsat}^{\mathrm{team}}_{\varphi}$
Input:	$\mathcal{A} \in \mathrm{STRUC}$
Output:	$\{\, X \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid X$ is a minimum team for φ in $\mathcal{A} \,\}$

We start by showing $\mathsf{E\text{-}cmaxsat}^{\mathrm{team}}_{\varphi} \in \mathsf{DelNP}$. For this we use a new torchlight oracle, namely $\mathsf{D\text{-}extendcteam}_{\varphi}$.

Problem:	D-$\mathsf{extendcteam}_\varphi$
Input:	$\mathcal{A} \in \text{STRUC}, X \in \text{TEAM}(\varphi), Y \in \text{TEAM}(\varphi), k \in \mathbb{N}$
Question:	$\left\{ X' \in \text{TEAM}(\mathcal{A},\varphi) \,\middle|\, \begin{array}{l} \mathcal{A} \models_{X'} \varphi,\ X \subsetneq X', \\ X' \cap Y = \emptyset \text{ and } \lvert X' \rvert = k' \end{array} \right\} \neq \emptyset?$

Note that D-$\mathsf{extendcteam}_\varphi$ is in NP: Use the same method as for D-$\mathsf{extendteam}_\varphi$ and check if $|X| = k$ afterwards.

Theorem 4.6 ([HMMV22]). E-$\mathsf{cmaxsat}_\varphi^{\text{team}} \in \mathsf{DelNP}$ *for* $\varphi \in \mathsf{FO}(\mathrm{T})$ *and* $\mathrm{T} = \mathfrak{G}_{\mathsf{NP}}$.

Proof. There is a recursive algorithm that on input $(\mathcal{A}, X, k)$ enumerates all satisfying superteams of X having cardinality k with polynomial delay. The algorithm is very similar to Algorithm 1. The only differences are that $|X| = k$ is checked before a team X is output and that D-$\mathsf{extendcteam}_\varphi$ is used as the torchlight oracle instead of D-$\mathsf{extendteam}_\varphi$. A formal description of the algorithm is given in Algorithm 3. Running the algorithm on input $(\mathcal{A}, \emptyset, k)$, where k is the highest cardinality of a satisfying team, enumerates all solutions. The highest cardinality can be computed by asking the D-$\mathsf{extendcteam}_\varphi$ oracle on input $(\mathcal{A}, \emptyset, \emptyset, i)$ for $i = |\text{dom}(\mathcal{A})|^{|\mathsf{free}(\varphi)|}, \dots, 1$. The highest cardinality is then the largest value i for which the answer of the oracle was "yes".

Algorithm 3: E-$\mathsf{cmaxsat}_\varphi^{\text{team}} \in \mathsf{DelNP}$ for $\varphi \in \mathsf{FO}(T)$

```
Oracles: D-$\mathsf{extendcteam}_\varphi$ and V-$\mathsf{sat}_\varphi^{\text{team}}$
Function
EnumerateCMaxTeams(structure $\mathcal{A}$, team $X$, natural number $k$)
1    if V-$\mathsf{sat}_\varphi^{\text{team}}(\mathcal{A}, X) \wedge |X| = k$ then output $X$
2    $Y = \bigcup_{s < \max(X) \wedge s \notin X} s$
3    else if D-$\mathsf{extendcteam}_\varphi(\mathcal{A}, X, Y, k)$ then
4        for $s > \mathsf{max}(X)$ do EnumerateCardMaxTeams($\mathcal{A}, X \cup \{ s \}, k$)
```

The algorithm outputs every solution exactly once and has polynomial delay. This can be argued similarly as in the proof of Theorem 4.1. □

As we will see next, the membership results for E-$\mathsf{sat}_\varphi^{\text{team}}$ and E-$\mathsf{cmaxsat}_\varphi^{\text{team}}$ can be translated to E-$\mathsf{minsat}_\varphi^{\text{team}}$ and E-$\mathsf{cminsat}_\varphi^{\text{team}}$, by making slight modifications to the algorithms.

Theorem 4.7 ([HMMV22]). E-$\mathsf{minsat}_\varphi^{\text{team}}$, E-$\mathsf{cminsat}_\varphi^{\text{team}} \in \mathsf{DelNP}$ *for* $\varphi \in \mathsf{FO}(\mathrm{T})$ *and* $\mathrm{T} \subseteq \mathfrak{G}_{\mathsf{NP}}$.

Proof. For E-$\mathsf{minsat}_\varphi^{\mathrm{team}}$ we can run a slightly modified version of Algorithm 1 on input $(\mathcal{A}, \emptyset)$, which was originally used for E-$\mathsf{sat}_\varphi^{\mathrm{team}}$. The only modification needed is that the new algorithm terminates after outputting a solution. This suffices to enumerate E-$\mathsf{minsat}_\varphi^{\mathrm{team}}$, since Algorithm 1 outputs the solutions in an ascending order in any recursive call. Thus, in any recursive call only one minimal team is considered and this team is always output first.

For E-$\mathsf{cminsat}_\varphi^{\mathrm{team}}$ we can run Algorithm 3 on input $(\mathcal{A}, \emptyset, k)$, where k is the lowest cardinality (instead of the maximal) of a satisfying team. This way we can enumerate E-$\mathsf{cminsat}_\varphi^{\mathrm{team}}$, since as stated in Theorem 4.6, Algorithm 3 enumerates all satisfying teams with cardinality k. The lowest cardinality can be computed analogously to the maximal cardinality using the oracle D-$\mathsf{extendcteam}_\varphi$.

Both algorithms inherit the properties that are needed for showing DelNP membership from their counterparts. □

Having shown DelNP membership, we can already conclude DelNP-completeness for certain formulas. This is due to the fact that Theorem 4.2 together with Theorem 2.54 induces DelNP-hardness.

Corollary 4.8. *Let* $\mathrm{T} = \{\perp, =(\dots)\}$*, then there is a formula* $\varphi \in \mathsf{FO}(\mathrm{T})$ *such that* E-$\mathsf{maxsat}_\varphi^{\mathrm{team}}$*,* E-$\mathsf{cmaxsat}_\varphi^{\mathrm{team}}$*,* E-$\mathsf{minsat}_\varphi^{\mathrm{team}}$*,* E-$\mathsf{cminsat}_\varphi^{\mathrm{team}}$ *are* DelNP*-hard. Furthermore, the latter three problems are in* DelNP *and thus* DelNP*-complete.*

Proof. Again we argue only for the independence logic case. By Theorem 4.2 there is a formula $\varphi \in \mathsf{FO}(\perp)$ such that D-$\mathsf{sat}_\varphi^{\mathrm{team}}$ is NP-hard. Furthermore, the Problem D-$\mathsf{sat}_\varphi^{\mathrm{team}}$ can be decided in polynomial time by an EOM $\mathrm{M}^{\mathrm{E}\text{-}f}$, where $f \in \{\mathsf{maxsat}_\varphi^{\mathrm{team}}, \mathsf{cmaxsat}_\varphi^{\mathrm{team}}, \mathsf{minsat}_\varphi^{\mathrm{team}}, \mathsf{cminsat}_\varphi^{\mathrm{team}}\}$: The machine just asks the oracle for the next solution. If the answer is a team then it outputs "yes" otherwise it outputs "no". By Theorem 2.54 it follows that E-f is DelNP-hard for $f \in \{\mathsf{maxsat}_\varphi^{\mathrm{team}}, \mathsf{cmaxsat}_\varphi^{\mathrm{team}}, \mathsf{minsat}_\varphi^{\mathrm{team}}, \mathsf{cminsat}_\varphi^{\mathrm{team}}\}$. Finally by the Theorems 4.6 and 4.7, the problems E-$\mathsf{cmaxsat}_\varphi^{\mathrm{team}}$, E-$\mathsf{minsat}_\varphi^{\mathrm{team}}$, E-$\mathsf{cminsat}_\varphi^{\mathrm{team}}$ are in DelNP and thus DelNP-complete. □

By Corollary 4.8 we have further characterisations of DelNP as

$$\mathsf{DelNP} = \left[\text{E-}\mathsf{cmaxsat}_\varphi^{\mathrm{team}}\right]^{\leq_{\mathsf{DelP}}} = \left[\text{E-}\mathsf{minsat}_\varphi^{\mathrm{team}}\right]^{\leq_{\mathsf{DelP}}} = \left[\text{E-}\mathsf{cminsat}_\varphi^{\mathrm{team}}\right]^{\leq_{\mathsf{DelP}}},$$

where φ is the formula from Corollary 4.8 or more generally as

$$\begin{aligned}\mathsf{DelNP} &= \Big[\bigcup_{\varphi \in \mathsf{FO}(\mathrm{T})} \text{E-}\mathsf{cmaxsat}_\varphi^{\mathrm{team}}\Big]^{\leq_{\mathsf{DelP}}} \\ &= \Big[\bigcup_{\varphi \in \mathsf{FO}(\mathrm{T})} \text{E-}\mathsf{minsat}_\varphi^{\mathrm{team}}\Big]^{\leq_{\mathsf{DelP}}}\end{aligned}$$

$$= \Big[\bigcup_{\varphi \in \mathsf{FO}(\mathrm{T})} \text{E-}\mathsf{cminsat}_{\varphi}^{\mathrm{team}} \Big]^{\leq_{\mathsf{DelP}}}$$

for $\mathrm{T} \in \{ \perp, =(\dots) \}$.

We have not talked much about the problem E-$\mathsf{maxsat}_{\varphi}^{\mathrm{team}}$ and concluded only DelNP-hardness so far. The reason for this is that there does not seem to be a suitable torchlight to show DelNP membership. A helpful torchlight would, given a team X, answer the query whether there is an inclusion maximal satisfying team X' that is a superset of X and contains no assignments from a set Y. This seems to be a Σ_2^{P}-query. Note that in contrast to here, for D-$\mathsf{extendcteam}_{\varphi}$ we did not need the maximality condition since we could simply ask for a team of size k. One way to get rid of the maximality condition would be to omit the set Y, but then there is no difference between asking for a maximal satisfying team and asking for any satisfying team since these questions are equivalent. Using only such an oracle it is not clear how to compute different solutions with polynomial delay: At some point in the algorithm the torchlight could tell us that there is a satisfying team X' that is a superset of our current team X, but the only solution X'' (a maximal satisfying team) with $X \subseteq X''$ has been output before. In that case—depending of the design of the algorithm—the algorithm would either output X'' again or go through the whole set of possible solutions without outputting a solution which could then lead to an exponential delay. In both cases we would fail to present a DelC algorithm for any class decision complexity class C.

Instead of showing DelNP membership we show $\mathsf{Del}^{+}\mathsf{NP}$ membership. This gives us the power to make queries which are not polynomially long with respect to the input. The idea is to modify the input structure for the torchlight after each output, such that the input structure contains all the information about the teams that were output before. By doing this (and by modifying the formula once) we can ensure that no team is output twice. To achieve this, given a structure $\mathcal{A} = (\{ 0, \dots, n-1 \}, R_0^{\mathcal{A}}, \dots, R_{m-1}^{\mathcal{A}})$ and a set of teams $\{ X_0, \dots, X_{j-1} \}$, we define a new structure $\mathcal{A}(X_0, \dots, X_{j-1})$ as follows:

$$\mathcal{A}' \coloneqq \big(\{ 0, \dots, \max(n-1, j-1) \}, R_0^{\mathcal{A}}, \dots, R_{m-1}^{\mathcal{A}}, U^{\mathcal{A}'}, S^{\mathcal{A}'}\big),$$

where $\mathcal{A}'$ is an abbreviation for $\mathcal{A}(X_0, \dots, X_{j-1})$, $U^{\mathcal{A}'} = \{ 0, \dots, n-1 \}$ and $S^{\mathcal{A}'} = \{ (i, a_0, \dots, a_{k-1}) \mid (a_0, \dots, a_{k-1}) \in \mathrm{rel}(X_i) \}$. The intention of this new structure $\mathcal{A}'$ is to store (previously output) teams in the relation $S^{\mathcal{A}'}$. Since the number of teams we want to store in $S^{\mathcal{A}'}$ might be larger than $|\mathrm{dom}(\mathcal{A})|$, we need to extend the universe for $\mathrm{dom}(\mathcal{A}')$. To be able to distinguish between elements from $\mathrm{dom}(\mathcal{A})$ and $\mathrm{dom}(\mathcal{A}')$ we use the relation $U^{\mathcal{A}'}$. Now by replacing $\mathcal{A}$ with $\mathcal{A}(X_0, \dots, X_{j-1})$ in our oracle queries, we give the oracle indirect access to the solutions $X_0, \dots, X_{j-1}$. Since now our structure contains all information about the solutions that have already been output, our oracle does not need its third input Y, which was only used for storing information about the sets of teams that have been searched through so far. Therefore we use the oracle D-$\mathsf{extendteam}'_\varphi$ which is defined analogously to D-$\mathsf{extendteam}_\varphi$ but without the third input Y.

Problem:	D-$\mathsf{extendteam}'_\varphi$
Input:	$\mathcal{A} \in \mathrm{STRUC}, X \in \mathrm{TEAM}(\varphi)$
Question:	$\{ X' \in \mathrm{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_{X'} \varphi, X \subsetneq X' \} \neq \emptyset$?

Clearly this oracle is in NP as it is a simpler version of D-$\mathsf{extendteam}_\varphi$.

Theorem 4.9 ([HMMV22]). *For* $\mathrm{T} \in \{\perp, =(\dots), \subseteq\}$ *and* $\varphi \in \mathsf{FO}(\mathrm{T})$ *we have that* E-$\mathsf{maxsat}^{\mathrm{team}}_\varphi \in \mathsf{Del}^+\mathsf{NP}$.

Proof. By Proposition 2.11 it suffices to show this result for $\mathrm{T} = \{\perp\}$. Let $\varphi_0(x_0, \dots, x_{k-1})$ be a fixed $\mathsf{FO}(\perp)$ formula over $\sigma = \{ R_0, \dots, R_{m-1} \}$. We enumerate all inclusion maximal satisfying teams for φ_0 and an input structure $\mathcal{A} = (\{ 0, \dots, n-1 \}, R_0^{\mathcal{A}}, \dots, R_{m-1}^{\mathcal{A}})$ with the help of the oracles D-$\mathsf{extendteam}'_{\varphi_1}$ and V-$\mathsf{sat}^{\mathrm{team}}_{\varphi_1}$, for a formula φ_1 we are about to define.

The algorithm works as follows: After every output, the output itself is "added" to the current structure $\mathcal{A}$. To access the information about the solutions that are encoded in the current structure $\mathcal{A}$, we use the oracle D-$\mathsf{extendteam}'_{\varphi_1}$ for a new $\mathsf{FO}(\perp)$ formula $\varphi_1 \coloneqq \varphi_2 \wedge \varphi_3$, where φ_2, φ_3 are defined as follows:

$$\varphi_2(\overline{x}) \coloneqq \bigwedge_{x \in \overline{x}} U(x) \wedge \varphi_0'(\overline{x}),$$
$$\varphi_3(\overline{x}) \coloneqq \forall i \exists \overline{y}\ \neg S(i, \overline{y}) \wedge \overline{y} \subseteq \overline{x},$$

where $\overline{x}, \overline{y}$ are tuples of k variables, U, S are two new relations and φ_0' is the formula that we obtain from φ_0 by replacing each subformula $\exists z\ \varphi$ with $\exists z\ (U(z) \wedge \varphi)$ and each subformula $\forall z\ \varphi$ with $\forall z\ (\neg U(z) \vee \varphi)$. The point of relation U is to ensure, that only elements from the original universe are used for φ_0', for the free variables $x_0, \dots, x_{k-1}$ and indirectly also for the variables $y_0, \dots, y_{k-1}$. Recall that we can express the inclusion atom with the independence atom (see Proposition 2.11) and therefore it is appropriate to use an inclusion atom here.

Now, for a set of teams $\{ X_0, \dots, X_{j-1} \}$ and a team X with $\mathcal{A}(X_0, \dots, X_{j-1}) \models_X \varphi_1$, it follows that:

1. $\mathcal{A} \models_X \varphi$ and
2. $X \not\subseteq X'$ for all $X' \in \{ X_0, \dots, X_{j-1} \}$.

The first statement holds, since X satisfies φ_2 and $U^{\mathcal{A}'} = \{0, \dots, n-1\} = \mathrm{dom}(\mathcal{A})$. The second statement holds, since every satisfying team X must contain one assignment that is not in X' for each team $X' \in \{ X_0, \dots, X_{j-1} \}$ to satisfy φ_3. Both taken together imply that if X is maximal for φ_1, then it is maximal for φ_0 as well.

Before we can provide the algorithm we have to deal with one technical detail. Since our solutions are produced step by step we need to define the structures $\mathcal{A}(X_0, \dots, X_{j-1})$ inductively:

$$\begin{aligned}
\mathcal{A}(\emptyset) &\coloneqq \big(\{ 0, \dots, n-1 \}, R_0^{\mathcal{A}}, \dots, R_{m-1}^{\mathcal{A}}, \{0, \dots, n-1\}, \emptyset\big)\\
\mathcal{A}(X) &\coloneqq \big(\{ 0, \dots, \max(n-1, j) \}, R_0^{\mathcal{A}}, \dots, R_{m-1}^{\mathcal{A}}, \{0, \dots, n-1\}, S\big), \text{ where}\\
&\quad S \coloneqq S^{\mathcal{A}} \cup \{ (j, a_0, \dots, a_{k-1}) \mid (a_0, \dots, a_{k-1}) \in \mathrm{rel}(X) \}.
\end{aligned}$$

Now we can write $\mathcal{A}(X_0, \dots, X_{j-1})$ as $\mathcal{A}(\emptyset)(X_0) \dots (X_{j-1})$. To enumerate all maximal satisfying teams for φ_0 and $\mathcal{A}$, we run Algorithm 4 on input $(\mathcal{A}(\emptyset), \emptyset)$.

Algorithm 4: $\textsf{E-maxsat}^{\text{team}}_{\varphi_0} \in \textsf{Del}^{+}\textsf{NP}$ for $\varphi_0 \in \textsf{FO}(T)$

```
Oracles: $\textsf{D-extendteam}'_{\varphi_1}$ and $\textsf{V-sat}^{\text{team}}_{\varphi_1}$
Function EMaximalTeams(structure $\mathcal{A}$, team $X$)
1   if $\textsf{D-extendteam}'_{\varphi_1}(\mathcal{A}, X)$ then
2       $s \leftarrow X_{\text{full}} \setminus X$ with $\textsf{D-extendteam}'_{\varphi_1}(\mathcal{A}, X \cup \{ s \})$ or
         $\textsf{V-sat}^{\text{team}}_{\varphi_1}(\mathcal{A}, X \cup \{ s \})$
3       EMaximalTeams($\mathcal{A}, X \cup \{ s \}$)
4   else if $X \neq \emptyset$ then
5       output $X$
6       EMaximalTeams($\mathcal{A}(X), \emptyset$)
```

Since after each output the oracle query is (polynomially) extended, it might be exponential in size compared to the original input structure $\mathcal{A}$ at some point. Therefore this algorithm shows $\textsf{Del}^{+}\textsf{NP}$ membership (rather than $\textsf{DelNP}$ membership). Moreover, the sizes of the universe and S might be exponential with respect to the input.

By the design of formula φ_1 and the structures $\mathcal{A}(X_0, \dots X_{j-1})$, the oracles lead us only to new solutions and therefore no solution is output twice. After every output X the algorithm starts at the bottom again with a smaller search space since X is no longer considered. It follows that all solutions are obtained since the search space still contains all solutions that have not been output before.

The precomputation, the time between two consecutive outputs and the postcomputation have all the same (polynomial) bound: Starting at the empty team the algorithm adds one assignment at a time to the current team. In the worst case this has to be repeated $|\mathrm{dom}(\mathcal{A})|^{|\textsf{free}(\varphi)|}$ times. To add one assignment the algorithm has to check all assignments in the worst case, which again is bounded by $|\mathrm{dom}(\mathcal{A})|^{|\mathsf{free}(\varphi)|}$. The resulting delay is therefore bounded by $(|\mathrm{dom}(\mathcal{A})|^{|\mathsf{free}(\varphi)|})^2$ which is polynomial.

In order for this to work we have to deal with some technical details, which were not addressed in the algorithm for readability. We choose a special encoding of $\mathcal{A}' = \mathcal{A}(X_0, \ldots, X_{j-1})$ to make sure that we are able to extend the oracle queries in polynomial time. For this, we encode the relations $R_0^{\mathcal{A}}, \ldots, R_{m-1}^{\mathcal{A}}, U^{\mathcal{A}'}, S^{\mathcal{A}'}$ as lists of tuples and the universe as the set of values which occur as first entry in $S^{\mathcal{A}'}$. To extend $\mathcal{A}'$, we now only need to add tuples to the relation $S^{\mathcal{A}'}$. This leads to the problem, that the universe has fewer than n elements before the first n teams are output. We therefore add the elements $(i, n, \ldots, n)$ for all $i < n$ to $S^{\mathcal{A}'}$. This ensures that at least the elements $\{\,0, \ldots, n-1\,\}$ are included in the universe. Note that this does not have any influence on whether a team satisfies φ_1 or not, since $n \notin U^{\mathcal{A}'}$ and therefore n is not a possible value for the free variables $x_0, \ldots, x_{k-1}$ in a satisfying team.

As in the proofs of Theorem 4.1 and 4.6, we use two oracles here which is not consistent with the definition of a $\mathsf{DelC}/\mathsf{Del^+C}$ problem. This time we can not simply reduce them to another oracle, since the oracles' queries can be exponentially long. Carrying out the reduction could therefore take exponential time. We solve this problem by using the marked union of D-$\mathsf{extendteam}'_{\varphi_1}$ and V-$\mathsf{sat}^{\mathrm{team}}_{\varphi_1}$ as our oracle. This means that our oracle gets a structure $\mathcal{A}$, a team X and a bit b as input. If b is 0 then the question is "$\mathcal{A} \models_X \varphi_1$ and $X \neq \emptyset$?" and if b is 1 the question is "$\exists X' \mathcal{A} \models_{X'} \varphi_1$ and $X \subseteq X'$?". Note that this problem is in NP since V-$\mathsf{sat}^{\mathrm{team}}_{\varphi_1}$ and D-$\mathsf{extendteam}'_{\varphi_1}$ are both in NP and NP is closed under marked union.

Before an oracle query, the oracle tape now looks like this:

$$R_0^{\mathcal{A}} \mid \cdots \mid R_{m-1}^{\mathcal{A}} \mid U^{\mathcal{A}'} \mid S_0 \mid \cdots \mid S_{j-1} \mid X \mid 0/1,$$

where $S_i = \{\,(i, a_0, \ldots, a_{k-1}) \mid (i, a_0, \ldots, a_{k-1}) \in S^{\mathcal{A}'}\,\}$, X is the current team (which might be empty) and 0 or 1 which specifies the oracle query.

□

4.2 Enumeration of Optima in Inclusion Logic

For enumeration of optima in inclusion logic we can summarise our results in the following way: Enumeration of maxima is easy (in DelP), whereas enumeration of minima is hard (DelNP-complete). We will first show the easy cases in the next theorem and will handle the minima cases in separate theorems afterwards.

For the maxima cases, E-$\mathsf{maxsat}^{\mathrm{team}}_{\varphi}$ and E-$\mathsf{cmaxsat}^{\mathrm{team}}_{\varphi}$, the solution sets are either empty or consist of one single team, the team $X_{\max}$. Since this team can be computed in polynomial time (see Proposition 2.42) we can conclude that $\text{E-}\mathsf{maxsat}_\varphi^{\text{team}}, \text{E-}\mathsf{cmaxsat}_\varphi^{\text{team}} \in \mathsf{DelP}$.

Theorem 4.10 ([HMMV22])**.** $\text{E-}\mathsf{maxsat}_\varphi^{\text{team}}, \text{E-}\mathsf{cmaxsat}_\varphi^{\text{team}} \in \mathsf{DelP}$ *for* $\varphi \in \mathsf{FO}(\subseteq)$.

Proof. Compute team $X = \text{S-}\mathsf{maxsubteam}_\varphi(\mathcal{A}, X_{\text{full}})$ and output X, when $X \neq \emptyset$ or $\bot$ otherwise. □

Corollary 4.11. *For any* $\varphi \in \mathsf{FO}(\subseteq)$ *the problems* $\text{E-}\mathsf{maxsat}_\varphi^{\text{team}}$ *and* $\text{E-}\mathsf{cmaxsat}_\varphi^{\text{team}}$ *are* DelP*-complete.*

Proof. By Theorem 4.10 $\text{E-}\mathsf{maxsat}_\varphi^{\text{team}}$ and $\text{E-}\mathsf{cmaxsat}_\varphi^{\text{team}}$ are in DelP for any formula $\varphi \in \mathsf{FO}(\subseteq)$. As DelP membership implies DelP-hardness (see Remark 2.56), we can conclude that $\text{E-}\mathsf{maxsat}_\varphi^{\text{team}}$ and $\text{E-}\mathsf{cmaxsat}_\varphi^{\text{team}}$ are DelP-complete. □

Corollary 4.11 leads to additional characterisations of DelP as

$$\mathsf{DelP} = \left[\text{E-}\mathsf{cmaxsat}_\varphi^{\text{team}}\right]^{\leq_{\mathsf{DelP}}} = \left[\text{E-}\mathsf{maxsat}_\varphi^{\text{team}}\right]^{\leq_{\mathsf{DelP}}},$$

where $\varphi \in \mathsf{FO}(\subseteq)$.

In the next result, we show NP-hardness for the decision problem $\text{D-}\mathsf{k\text{-}minsat}_\varphi^{\text{team}}$ for an inclusion logic formula φ.

Problem:	$\text{D-}\mathsf{k\text{-}minsat}_\varphi^{\text{team}}$
Input:	$\mathcal{A} \in \text{STRUC}, k \in \mathbb{N}$
Question:	$\{\, X \in \text{TEAM}(\mathcal{A}, \varphi) \mid \mathcal{A} \models_X \varphi, X \neq \emptyset \text{ and } \lvert X\rvert \leq k \,\} \neq \emptyset$?

By this and Theorem 2.54, we can conclude DelNP-hardness for $\text{E-}\mathsf{cminsat}_\varphi^{\text{team}}$. We reduce from the problem $\text{D-}\mathsf{is}^{\neg\text{full}}$—a variation of the independent set decision problem, where the full vertex set V is not allowed as a solution—to $\text{D-}\mathsf{k\text{-}minsat}_\varphi^{\text{team}}$ with two intermediate steps.

Problem:	$\text{D-}\mathsf{is}^{\neg\text{full}}$
Input:	$G = (V, E) \in \text{GRAPH}, k \in \mathbb{N}$
Question:	$\{\, V' \subsetneq V \mid \forall u, v \in V' \colon \{\, u, v \,\} \notin E, \lvert V'\rvert \geq k \,\} \neq \emptyset$?

Note that $\text{D-}\mathsf{is}^{\neg\text{full}}$ is NP-complete: We can reduce from the standard independent set problem, where $V' = V$ is allowed, by just adding one new vertex which is connected to all old vertices. Finally recall the following two problems for our reduction.

Problem:	$\text{D-}\mathsf{k\text{-}minsat}_{\text{DH}}^{\neg\emptyset}$
Input:	$\chi \in \text{DH}, k \in \mathbb{N}$
Question:	$\{\, \beta \in \Theta(\chi) \mid \beta \models \chi, \beta \neq \emptyset \text{ and } \lvert\beta\rvert \leq k \,\} \neq \emptyset$?

Problem:	$\mathsf{D\text{-}k\text{-}minsat}_{\psi}^{\mathrm{rel}}$
Input:	$\mathcal{A} \in \mathrm{STRUC}, k \in \mathbb{N}$
Question:	$\{\, R \in \mathrm{REL}(\mathcal{A}, \psi) \mid \mathcal{A}, R \models \psi, R \neq \emptyset \text{ and } \lvert R\rvert \leq k \,\} \neq \emptyset$?

Theorem 4.12 ([HMMV22]). *There is a* $\varphi \in \mathsf{FO}(\subseteq)$ *such that* $\mathsf{D\text{-}k\text{-}minsat}_{\varphi}^{\mathrm{team}}$ *is* NP*-hard.*

Proof. We reduce from the NP-complete problem $\mathrm{D\text{-}}\mathsf{is}^{\neg\mathrm{full}}$, showing that there are a myopic formula $\psi \in \Sigma_1^1$ and a formula $\varphi \in \mathsf{FO}(\subseteq)$ such that

$$\mathrm{D\text{-}}\mathsf{is}^{\neg\mathrm{full}} \underset{(1)}{\leq_m^{\mathsf{P}}} \mathsf{D\text{-}k\text{-}minsat}_{\mathrm{DH}}^{\neg\emptyset} \underset{(2)}{\leq_m^{\mathsf{P}}} \mathsf{D\text{-}k\text{-}minsat}_{\psi}^{\mathrm{rel}} \underset{(3)}{\leq_m^{\mathsf{P}}} \mathsf{D\text{-}k\text{-}minsat}_{\varphi}^{\mathrm{team}}.$$

For (1) an arbitrary $(G = (V, E), k)$ is mapped to $(\chi = \bigwedge_{\{\,i,j\,\}\in E} x_i \vee x_j, \lvert V\rvert - k)$. Intuitively, assigning a variable x_i to 0 in χ corresponds to picking the vertex i in G for an independent set. The formula χ expresses that at most one of the variables in any clause may be set to 0, corresponding to the condition that at most one of the endpoints of an edge can be in an independent set. Obviously, there is a one-to-one correspondence between independent sets V' of G of size at least k and satisfying assignments of χ of weight at most k. Note that χ is a DualHorn formula.

(2): As we have seen before in Theorem 3.11, there is a formula ψ such that for all DualHorn formulas χ and all assignments $\beta \neq \emptyset$ it holds that

$$\beta \models \chi \iff \mathcal{A}_{\chi}, \beta \models \psi(\beta).$$

Note that compared to the counting case (see Theorem 3.11) where only the number of solutions to the DualHorn formula must be equal to the number of solutions to the $\mathsf{FO}(\subseteq)$ formula, in this case the sizes of the solutions must be preserved. Fortunately, the given translation does both, as the solutions for both χ and $\mathcal{A}_{\chi}$ are exactly the same. Thereby the identity function is a suitable reduction function.

Finally, (3) follows from Proposition 2.32, since ψ can be easily transformed to a myopic formula. □

Note that the reduction from the previous proof also works if we use $2\mathrm{CNF}^{+}$ formulas instead of DualHorn formulas, since the given formula $\chi = \bigwedge_{\{\,i,j\,\}\in E} x_i \vee x_j$ is a $2\mathrm{CNF}^{+}$ formula.

The next result is very similar to the previous: We show NP-hardness for the problem $\mathrm{D\text{-}}\mathsf{anothersolminsat}_{\varphi}^{\mathrm{team}}$ and conclude DelNP-hardness for $\mathrm{E\text{-}}\mathsf{minsat}_{\varphi}^{\mathrm{team}}$ afterwards. Note that we previously tried to show $\mathrm{E\text{-}}\mathsf{minsat}_{\varphi}^{\mathrm{team}} \in \mathsf{DelP}$ [HMMV22] by providing a suitable algorithm, which together with our result here implies $\mathsf{DelP} = \mathsf{DelNP}$ and hence $\mathsf{P} = \mathsf{NP}$. We stress that the provided algorithm does in fact enumerate $\mathrm{E\text{-}}\mathsf{minsat}_{\varphi}^{\mathrm{team}}$ but with exponential delay and hence does not yield $\mathrm{E\text{-}}\mathsf{minsat}_{\varphi}^{\mathrm{team}} \in \mathsf{DelP}$.

Problem:	$\textsf{D-anothersolminsat}^{\text{team}}_{\varphi}$
Input:	$\mathcal{A} \in \text{STRUC}$, $M = \{X_1, \ldots, X_k\} \in 2^{\text{TEAM}(\varphi)}$
Question:	$\left\{ X \in \text{TEAM}(\mathcal{A}, \varphi) \,\middle|\, \begin{array}{l} X \notin M \text{ and } X \text{ is} \\ \text{minimal for } \varphi \text{ in } \mathcal{A} \end{array} \right\} \neq \emptyset?$

This time we reduce the NP-complete problem $\textsf{D-sat}^1_{3\text{CNF}^+}$ (see [Sch78]) to $\textsf{D-anothersolminsat}^{\text{team}}_{\varphi}$ with three intermediate steps.

Problem:	$\textsf{D-sat}^1_{3\text{CNF}^+}$
Input:	$\chi \in 3\text{CNF}^+$
Question:	$\left\{ \beta \in \Theta(\chi) \,\middle|\, \begin{array}{l} \text{In each clause } C_i \text{ of } \chi \text{ there} \\ \text{is exactly one literal } \ell_{i,j} \\ \text{that evaluates to 1 under } \beta \end{array} \right\} \neq \emptyset?$

Let us recall the following problems for the reduction.

Problem:	$\textsf{D-anothersolmaxsat}^{\neg\text{full}}_{\text{HORN}}$
Input:	$\chi \in \text{HORN}$, $B = \{\beta_1, \ldots, \beta_k\} \in 2^{\Theta}$
Question:	$\{\beta \in \Theta(\chi) \mid \beta \notin B \text{ and } \beta \text{ is maximal for } \chi\} \neq \emptyset?$

Problem:	$\textsf{D-anothersolminsat}^{\neg\emptyset}_{\text{DH}}$
Input:	$\chi \in \text{DH}$, $B = \{\beta_1, \ldots, \beta_k\} \in 2^{\Theta}$
Question:	$\{\beta \in \Theta(\chi) \mid \beta \notin B \text{ and } \beta \text{ is minimal for } \chi\} \neq \emptyset?$

Problem:	$\textsf{D-anothersolminsat}^{\text{rel}}_{\psi}$
Input:	$\mathcal{A} \in \text{STRUC}$, $M = \{R_1, \ldots, R_k\} \in 2^{\text{REL}(\psi)}$
Question:	$\left\{ R \in \text{REL}(\mathcal{A}, \psi) \,\middle|\, \begin{array}{l} R \notin M \text{ and } R \text{ is} \\ \text{minimal for } \psi \text{ in } \mathcal{A} \end{array} \right\} \neq \emptyset?$

Theorem 4.13. *There is a formula $\varphi \in \mathsf{FO}(\subseteq)$ such that* $\textsf{D-anothersolminsat}^{\text{team}}_{\varphi}$ *is NP-hard.*

Proof. We reduce the NP-complete Problem $\textsf{D-sat}^1_{3\text{CNF}^+}$ to $\textsf{D-anothersolminsat}^{\text{team}}_{\varphi}$ for a $\varphi \in \mathsf{FO}(\subseteq)$ with a few intermediate steps. The complete chain of reduction looks as follows:

$$\begin{aligned} \textsf{D-sat}^1_{3\text{CNF}^+} &\leq^{\mathsf{P}}_m \textsf{anothersolmaxsat}^{\neg\text{full}}_{\text{HORN}} &(1)\\ &\leq^{\mathsf{P}}_m \textsf{anothersolminsat}^{\neg\emptyset}_{\text{DH}} &(2)\\ &\leq^{\mathsf{P}}_m \textsf{D-anothersolminsat}^{\text{rel}}_{\psi} &(3)\\ &\leq^{\mathsf{P}}_m \textsf{D-anothersolminsat}^{\text{team}}_{\varphi} &(4) \end{aligned}$$

(1): Kavvadias et al. showed this reduction already for $\mathsf{anothersolmaxsat}_{\mathrm{HORN}}$—the version where the full assignment is allowed as a solution [KSS00]. Since the formula they give in their reduction is never satisfied by the full assignment the same reduction is also suitable for showing NP-hardness for $\mathsf{anothersolmaxsat}^{\neg\mathrm{full}}_{\mathrm{HORN}}$.
(2): By Remark 2.1 we already know that for any Horn formula χ there is a DualHorn formula $\widetilde{\chi}$ such that for all assignments β:

$$\beta \models \chi \iff \mathsf{vars}(\varphi) \setminus \beta \models \widetilde{\chi}.$$

Moreover if β is maximal satisfying for χ, then $\mathsf{vars}(\varphi) \setminus \beta$ is minimal satisfying for $\widetilde{\chi}$. All taken together we can conclude:

$$\begin{aligned}&(\chi, \{\beta_1, \ldots, \beta_k\}) \in \text{D-}\mathsf{anothersolmaxsat}^{\neg\mathrm{full}}_{\mathrm{HORN}}\\ \iff &(\chi', \{\mathsf{vars}(\varphi) \setminus \beta_1, \ldots, \mathsf{vars}(\varphi) \setminus \beta_k\}) \in \text{D-}\mathsf{anothersolminsat}^{\neg\emptyset}_{\mathrm{DH}}.\end{aligned}$$

(3): By using the formula $\psi(R) = \psi_1 \wedge \psi_2(R)$ from the proofs of Theorems 3.11 and 4.12 we get the following equivalence:

$$\begin{aligned}&(\chi, \{\beta_1, \ldots, \beta_k\}) \in \text{D-}\mathsf{anothersolminsat}^{\neg\emptyset}_{\mathrm{DH}}\\ \iff &(\mathcal{A}_\chi, \{\beta_1, \ldots, \beta_k\}) \in \text{D-}\mathsf{anothersolminsat}^{\mathrm{rel}}_{\psi},\end{aligned}$$

for any DualHorn formula χ and sets of minimal assignments $\{\beta_1, \ldots, \beta_k\}$.
(4): This follows from Proposition 2.32. □

Corollary 4.14. *There are formulas $\varphi, \varphi' \in \mathsf{FO}(\subseteq)$ such that* E-$\mathsf{cminsat}^{\mathrm{team}}_{\varphi}$, E-$\mathsf{minsat}^{\mathrm{team}}_{\varphi'}$ *are* DelNP*-complete.*

Proof. By Theorem 4.7 the problems E-$\mathsf{cminsat}^{\mathrm{team}}_{\varphi}$, E-$\mathsf{minsat}^{\mathrm{team}}_{\varphi'}$ are in DelNP for any formulas $\varphi, \varphi' \in \mathsf{FO}(\subseteq)$.

DelNP-hardness for E-$\mathsf{cminsat}^{\mathrm{team}}_{\varphi}$ follows from Theorem 2.54 together with Theorem 4.12, as D-k-$\mathsf{minsat}^{\mathrm{team}}_{\varphi}$ can trivially be decided in polynomial time by a RAM $\mathrm{M}^{\text{E-}\mathsf{cminsat}^{\mathrm{team}}_{\varphi}}$: Simply get a solution from the oracle, output "yes" if the cardinality of the solution is at most k and "no" otherwise.

For E-$\mathsf{minsat}^{\mathrm{team}}_{\varphi'}$ DelNP-hardness follows from Theorems 2.54 and 4.13, since D-$\mathsf{anothersolminsat}^{\mathrm{team}}_{\varphi}$ can be decided in polynomial time by a RAM $\mathrm{M}^{\text{E-}\mathsf{minsat}^{\mathrm{team}}_{\varphi'}}$: Ask the oracle for $|M| + 1$ solutions, if any of the answers was $\perp$ output "no", otherwise output "yes". □

By Corollary 4.14 we have yet additional characterisations of DelNP as

$$\mathsf{DelNP} = \left[\text{E-}\mathsf{cminsat}^{\mathrm{team}}_{\varphi}\right]^{\leq_{\mathsf{DelP}}} = \left[\text{E-}\mathsf{minsat}^{\mathrm{team}}_{\varphi'}\right]^{\leq_{\mathsf{DelP}}},$$

where φ, φ' are the formulas from Corollary 4.14 or as

$$\mathsf{DelNP} = \Bigg[\bigcup_{\varphi \in \mathsf{FO}(\subseteq)} \text{E-}\mathsf{cminsat}^{\mathrm{team}}_{\varphi}\Bigg]^{\leq_{\mathsf{DelP}}} = \Bigg[\bigcup_{\varphi \in \mathsf{FO}(\subseteq)} \text{E-}\mathsf{minsat}^{\mathrm{team}}_{\varphi}\Bigg]^{\leq_{\mathsf{DelP}}}.$$

4.3 Summary

We have already seen a class diagram of the enumeration complexity classes we considered in this paper at the beginning of this section. In Sections 4.2 and 4.1 we identified additional complete problems for the mentioned classes, the results are summarised Table 4.1.

Table 4.1

	$\mathsf{FO}(\subseteq)$	$\mathsf{FO}(\perp), \mathsf{FO}(=(\dots))$
E-$\mathsf{sat}_{\varphi}^{\text{team}}$	DelP-complete	DelNP-complete
E-$\mathsf{maxsat}_{\varphi}^{\text{team}}$	DelP-complete	in $\mathsf{Del}^{+}\mathsf{NP}$,$\mathsf{DelNP}$-hard
E-$\mathsf{cmaxsat}_{\varphi}^{\text{team}}$	DelP-complete	DelNP-complete
E-$\mathsf{minsat}_{\varphi}^{\text{team}}$	DelNP-complete	DelNP-complete
E-$\mathsf{cminsat}_{\varphi}^{\text{team}}$	DelNP-complete	DelNP-complete

As we have seen, we can characterise the considered enumeration complexity classes as the $\leq_{\mathsf{DelP}}$ closure of any problem that is complete for the respective class. Consequently we can characterise DelNP via inclusion logic, for example as $\left[\text{E-}\mathsf{cminsat}_{\varphi}^{\text{team}}\right]^{\leq_{\mathsf{DelP}}}$ for a certain formula $\varphi \in \mathsf{FO}(\subseteq)$.

5 Conclusion

Even though the expressive power of independence logic is higher than the one of dependence logic, the descriptive decision complexity classes corresponding to those logics coincide. This is due to the facts that over sentences these logics are equal in terms of expressive power and that sentences suffice to describe decision problems. In the counting and enumeration settings this is no longer true. Sentences are not of much use in these settings, since here the solutions matter and need to be described by a formula. However, since the enumeration and decision setting are closely connected by definition, their classdiagrams look almost identical. On the other hand, in the counting setting we see a different picture. Here, the descriptive complexity of the classes corresponding to independence and dependence logic differ. Similarly, the classes $\#\mathsf{FO}^{\mathrm{rel}}$ and $\#\Sigma_1^1$ are not equal even though their decision counterparts are. With inclusion logic we can capture P, but we have no corresponding result like $\#\mathsf{FO}(\subseteq) = \mathsf{FP}$ in the counting setting. Moreover, we stress that this equality does most likely not hold since this would imply $\mathrm{C}\text{-}\mathsf{sat}_{\mathrm{DH}}^{\neg\emptyset} \in \mathsf{FP}$ which yields $\mathsf{FP} = \mathsf{FP}^{\#\mathsf{P}}$ (and thereby $\mathsf{FP} = \#\mathsf{PH}$ [TW92]), as $\mathrm{C}\text{-}\mathsf{sat}_{\mathrm{DH}}^{\neg\emptyset}$ is $\#\mathsf{P}$-complete under Turing reductions.

We like to mention a few results that follow from our results or should be easy to show. In Section 3.1 we showed $\#\mathsf{FO}(\perp) = \#\cdot\mathsf{NP}$ which, together with Theorem 3.4, implies that also $\#\Sigma_1^1$ captures $\#\cdot\mathsf{NP}$. We think that this result can be generalized to $\#\Sigma_k^1 = \#\cdot\Sigma_k^{\mathsf{P}}$, when providing a suitable definition of $\#\Sigma_k^1$. Furthermore, characterisations of decision classes with "classical logics" $(\Sigma_1^1, \mathsf{FO}^{\mathrm{rel}}, \mathsf{LFP})$ should be translatable to the enumeration setting. For this, one would have to define corresponding enumeration classes first, which could be done analogously to our definition of $\mathsf{DelFO}(\mathrm{T})$ (see Definition 2.58). In the enumeration setting we focused on the delay classes, but there are other complexity measures. One of these measures is incremental delay. Analogous to the $\mathsf{Del}\Sigma_k^{\mathsf{P}}$ hierarchy, Creignou et al. introduced an $\mathsf{Inc}\Sigma_k^{\mathsf{P}}$ hierarchy with corresponding incremental delay classes [CKP+19]. We like to point out that all our DelP and DelNP completeness results imply IncP and IncNP-completeness, respectively. Moreover, the problem $\mathrm{E}\text{-}\mathsf{maxsat}_{\varphi}^{\mathrm{team}}$ is also IncNP-complete, even though we were not able to show DelNP-completeness.

As we have seen, we can define infinitely many different team logics with the help of generalized team atoms. We mainly studied three specific ones of them, which leaves the others open for further studies. Maybe some more general results can be found that hold for all generalized team atoms or all that are NP-verifiable. Other individual atoms that might be of interest are the strong negation we mentioned in Chapter 2.4 and Boolean disjunction. Furthermore, there are "strict" versions of the existential quantifier and the disjunction (see for example [Gal12]) which might lead to some interesting results.

In Section 3.4 we showed that C-$\mathsf{sat}_{\Sigma_1\mathrm{CNF}^-}$ is $\#\cdot\mathsf{NP}$-complete by adjusting a reduction from Valiant that showed $\#\mathsf{P}$-completeness for C-$\mathsf{sat}_{2\mathrm{CNF}^+}$. We think that the technique from the proof can be used to identify further $\#\cdot\mathsf{NP}$-complete problems.

We studied the complexity of optimal solution problems in the enumeration setting, but not in the counting or decision setting. In the decision setting these problems all coincide with general satisfiability but the counting versions of these problems might be of interest. Especially C-$\mathsf{maxsat}^{\mathrm{team}}_{\varphi}$, as the problem E-$\mathsf{maxsat}^{\mathrm{team}}_{\varphi}$ has some similarities with the problem E-CIRCUMSCRIPTION that was studied by Creignou et al. [CKP$^+$19]. They showed that E-CIRCUMSCRIPTION is DelNP-hard and included in $\mathsf{Del}^+\mathsf{NP}$ (like E-$\mathsf{maxsat}^{\mathrm{team}}_{\varphi}$) and has some other interesting properties [CKP$^+$19]. The corresponding "natural torchlight oracle" is Σ^{P}_2-complete (we suspect the same to be true for E-$\mathsf{maxsat}^{\mathrm{team}}_{\varphi}$) and the counting version C-CIRCUMSCRIPTION is $\#\cdot\mathsf{coNP}$-complete. We therefore think it might be interesting to study the corresponding counting and torchlight problems.

Our definition of $\mathsf{DelFO}(\mathrm{T})$ has the benefit that is comes with $\leq_{\mathsf{DelP}}$ closure but we think it could be enhanced. First of all, $\leq_{\mathsf{DelP}}$ reductions are a very strong reducibility notion, a weaker reduction like $\leq^{\mathsf{P}}_{m}$ should suffice to define the same class. No matter which reducibility notion we choose, the "closure type" definition has the drawback that it introduces new resource bounds, which does not seem very fitting for descriptive complexity classes. Therefore we are very intrigued whether there is a definition of $\mathsf{DelFO}(\mathrm{T})$ that disposes of the closure entirely.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.

[ACMS15] Rehan Abdul Aziz, Geoffrey Chu, Christian J. Muise, and Peter J. Stuckey. #∃SAT: Projected model counting. In *SAT*, volume 9340 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2015.

[ÀJ93] Carme Àlvarez and Birgit Jenner. A very hard log-space counting class. *Theor. Comput. Sci.*, 107(1):3–30, 1993.

[AMR20] Marcelo Arenas, Martin Muñoz, and Cristian Riveros. Descriptive complexity for counting complexity classes. *Log. Methods Comput. Sci.*, 16(1), 2020.

[AO96] Eric Allender and Mitsunori Ogihara. Relationships among PL, #L, and the determinant. *RAIRO Theor. Informatics Appl.*, 30(1):1–21, 1996.

[CKP$^+$19] Nadia Creignou, Markus Kröll, Reinhard Pichler, Sebastian Skritek, and Heribert Vollmer. A complexity theory for hard enumeration problems. *Discret. Appl. Math.*, 268:191–209, 2019.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158. ACM, 1971.

[DHKV21] Arnaud Durand, Anselm Haak, Juha Kontinen, and Heribert Vollmer. Descriptive complexity of #P functions: A new perspective. *J. Comput. Syst. Sci.*, 116:40–54, 2021.

[DKdRV15] Arnaud Durand, Juha Kontinen, Nicolas de Rugy-Altherre, and Jouko Väänänen. Tractability frontier of data complexity in team semantics. In *GandALF*, volume 193 of *EPTCS*, pages 73–85, 2015.

[End72] Herbert B. Enderton. *A mathematical introduction to logic.* Academic Press, 1972.

[Fag74] Ronald Fagin. Generalized first-order spectra, and polynomial time recognizable sets. *SIAM-AMS Proceedings*, 7:43–73, 1974.

[Gal12] Pietro Galliani. Inclusion and exclusion dependencies in team semantics - on some logics of imperfect information. *Ann. Pure Appl. Logic*, 163(1):68–84, 2012.

[GH13] Pietro Galliani and Lauri Hella. Inclusion logic and fixed point logic. In *CSL*, volume 23 of *LIPIcs*, pages 281–295. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[GHR95] Raymond Greenlaw, H James Hoover, and Walter L Ruzzo. *Limits to parallel computation: P-completeness theory.* Oxford University Press on Demand, 1995.

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[Grä13] Erich Grädel. Model-checking games for logics of imperfect information. *Theor. Comput. Sci.*, 493:2–14, 2013.

[Grä16] Erich Grädel. Games for inclusion logic and fixed-point logic. In *Dependence Logic*, pages 73–98. Springer, 2016.

[GV13] Erich Grädel and Jouko A. Väänänen. Dependence and independence. *Studia Logica*, 101(2):399–410, 2013.

[Hen61] L. Henkin. Some remarks on infinitely long formulas. *Journal of Symbolic Logic*, 30(1):167–183, 1961.

[HKM$^+$19] Anselm Haak, Juha Kontinen, Fabian Müller, Heribert Vollmer, and Fan Yang. Counting of teams in first-order team logics. In *MFCS*, volume 138 of *LIPIcs*, pages 19:1–19:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[HMMV22] Anselm Haak, Arne Meier, Fabian Müller, and Heribert Vollmer. Enumerating teams in first-order team logics. *Ann. Pure Appl. Log.*, 173(10):103163, 2022.

[HS89] Jaakko Hintikka and Gabriel Sandu. Informational independence as a semantical phenomenon. *Studies in logic and the foundations of mathematics*, 126:52–70, 1989.

[HV95] Lane A. Hemaspaandra and Heribert Vollmer. The satanic notations: counting classes beyond #P and other definitional adventures. *SIGACT News*, 26(1):2–13, 1995.

[Imm82] Neil Immerman. Relational queries computable in polynomial time (extended abstract). In *STOC*, pages 147–152. ACM, 1982.

[Imm86] Neil Immerman. Relational queries computable in polynomial time. *Inf. Control.*, 68(1-3):86–104, 1986.

[Imm87] Neil Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16(4):760–778, 1987.

[Imm99] Neil Immerman. *Descriptive Complexity.* Graduate texts in computer science. Springer, 1999.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988.

[Kas86] Simon Kasif. On the parallel complexity of some constraint satisfaction problems. In *AAAI*, pages 349–353. Morgan Kaufmann, 1986.

[KPSZ01] Aggelos Kiayias, Aris Pagourtzis, Kiron Sharma, and Stathis Zachos. Acceptor-definable counting classes. In *Panhellenic Conference on Informatics*, volume 2563 of *Lecture Notes in Computer Science*, pages 453–463. Springer, 2001.

[KSS00] Dimitris J. Kavvadias, Martha Sideri, and Elias C. Stavropoulos. Generating all maximal models of a boolean expression. *Inf. Process. Lett.*, 74(3-4):157–162, 2000.

[KV09] Juha Kontinen and Jouko A. Väänänen. On definability in dependence logic. *Journal of Logic, Language and Information*, 18(3):317–332, 2009.

[KY19] Juha Kontinen and Fan Yang. Logics for first-order team properties. In *WoLLIC*, volume 11541 of *Lecture Notes in Computer Science*, pages 392–414. Springer, 2019.

[Pag01] Aris Pagourtzis. On the complexity of hard counting problems with easy decision version. In *Proceedings of the 3rd Panhellenic Logic Symposium*, PLS, pages 21–29, 2001.

[Pap94] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[PZ06] Aris Pagourtzis and Stathis Zachos. The complexity of counting functions with easy decision version. In *MFCS*, volume 4162 of *Lecture Notes in Computer Science*, pages 741–752. Springer, 2006.

[RV97] Kenneth W. Regan and Heribert Vollmer. Gap-languages and log-time complexity classes. *Theor. Comput. Sci.*, 188(1-2):101–116, 1997.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226. ACM, 1978.

[SST95] Sanjeev Saluja, K. V. Subrahmanyam, and Madhukar N. Thakur. Descriptive complexity of #P functions. *J. Comput. Syst. Sci.*, 50(3):493–505, 1995.

[Str10] Yann Strozecki. *Enumeration complexity and matroid decomposition.* PhD thesis, Université Paris Diderot - Paris 7, 2010.

[Tod91] Seinosuke Toda. *Computational complexity of counting complexity classes.* PhD thesis, Tokyo Institute of Technology, 1991.

[TW92] Seinosuke Toda and Osamu Watanabe. Polynomial time 1-turing reductions from #ph to #p. *Theor. Comput. Sci.*, 100(1):205–221, 1992.

[Vää07] Jouko A. Väänänen. *Dependence Logic - A New Approach to Independence Friendly Logic*, volume 70 of *London Mathematical Society student texts.* Cambridge University Press, 2007.

[Val79a] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[Val79b] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[Val92] Leslie G. Valiant. *Why is Boolean Complexity Theory so Difficult?*, page 84–94. London Mathematical Society Lecture Note Series. Cambridge University Press, 1992.

[Var82] Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *STOC*, pages 137–146. ACM, 1982.

[Vin91] V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *SCT*, pages 270–284. IEEE Computer Society, 1991.

[Yan20] Fan Yang. Axiomatizing first order consequences in inclusion logic. *Math. Log. Q.*, 66(2):195–216, 2020.