Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Theoretische Informatik

Enumeration in team based logics

Masterarbeit

Hannover, 16.05.2017

Christian Reinbold
Matrikelnr.: 2942360

Erstprüfer:	Prof. Dr. Heribert Vollmer
Zweitprüfer:	Dr. rer. nat. Arne Meier
Betreuer:	Dr. rer. nat. Arne Meier

Abstract

Classical logics as first order logic or modal logic may be extended by a dependence atom $=(P,Q)$, expressing that Q functionally depends on P. In this thesis we research the enumeration complexity for dependence logics, considering the propositional and modal case respectively. We show that the existence of a **DelayP**-agorithm for the full propositional dependence logic implies P = NP. Hence, we focus on propositional dependence logic without the split junction operator. By restricting the team size according to a parameter and exploiting the invariance of the dependence atom regarding a group action of $\mathbb{F}_2^n$ on teams, we are able to construct a **DelayFPT**-algorithm that outputs all satisfying teams ordered by their cardinality. If the team size is restricted by a computable function f instead of a parameter, the algorithm performs with polynomial delay as long f itself is bounded by a polynomial. Furthermore, we show that a **DelayP**-algorithm sorting the output by its cardinality cannot be obtained if the restricting function f exceeds polynomial growth. A combination of our approach with merge sort techniques allows us to efficiently enumerate the solutions for DNF-formulas which are built with the classical disjunction operator. In addition to the previous algorithm, we present an alternative with incremental delay that performs in polynomial space.

For modal dependence logic, we show that the fragments constructed by either removing the $\Diamond$ or $\vee$ operator are not enumerable with polynomial delay unless P = NP. If both operators are absent, we can present reductions so that the problem of enumerating all maximal teams is equivalent to the problem of enumerating all maximal independent sets in undirected graphs. At last, we show that any polynomial verifiable fragment of modal dependence logic permits an enumeration algorithm with polynomial delay and polynomial space. If the outputted teams should be sorted by cardinality, we have to drop the space restriction, but preserve polynomial delay.

Kurzzusammenfassung

Klassische Logiken wie die Prädikatenlogik erster Stufe oder Modallogik können um ein Abhängigkeitsatom $=(P, Q)$ ergänzt werden, das die Abhängigkeit von Q bzgl. P ausdrückt. In der vorliegenden Arbeit untersuchen wir die Komplexität von Zählproblemen für um Abhängigkeiten ergänzte Logiken, wobei wir uns auf den aussagenlogischen sowie modalen Fall beschränken. Wir zeigen, dass die Existenz eines **DelayP**-Algorithmus für das vollständige, um Abhängigkeiten ergänzte, Fragment der Aussagenlogik P = NP impliziert. Aus diesem Grund betrachten wir zunächst nur Formeln ohne „split junction"-Operator. Die Beschränkung der Teamgröße durch einen Parameter sowie die Invarianz des Abhängigkeitsatoms unter einer Gruppenwirkung von $\mathbb{F}_2^n$ auf Teams ermöglichen es uns, einen **DelayFPT**-Algorithmus zu konstruieren, der alle erfüllenden Teams sortiert nach ihrer Kardinalität ausgibt. Falls die Teamgröße anstatt durch einen Parameter durch eine berechenbare Funktion f beschränkt wird, ist der Delay des Algorithmus polynomiell - vorausgesetzt f selbst ist durch ein Polynom beschränkt. Ist letzteres nicht der Fall, so können wir zeigen, dass kein polynomieller Algorithmus mit polynomiellen Delay existieren kann, der alle erfüllenden Teams nach Größe sortiert ausgibt. Die Kombination unseres Verfahrens mit Mergesort-Techniken erlaubt es uns, die Lösungen für DNF-Formeln mit klassischer Disjunktion effizient aufzuzählen. Ferner präsentieren wir einen alternativen Algorithmus mit größerem, inkrementellen Delay, dessen Platzbedarf dafür aber polynomiell beschränkt ist.

Im modalen Fall zeigen wir, dass unter der Annahme P ≠ NP die durch Entfernen des $\Diamond$- oder $\vee$-Operators entstehenden Logikfragmente nicht mit polynomiellen Delay aufzählbar sind. Lassen wir keinen der beiden Operatoren zu, so können wir die Zählprobleme maximaler erfüllender Teams sowie maximaler Co-Cliquen in ungerichten Graphen ineinander überführen. Zum Schluss zeigen wir, dass für jedes polynomiell verifizierbare Fragment der um Abhängigkeiten ergänzte Modallogik ein **DelaySpaceP**-Algorithmus existiert. Falls die Teams ihrer Größe nach sortiert ausgegeben werden sollen, können wir weiterhin einen **DelayP**-Algorithmus angeben, dessen Platzbedarf jedoch nicht durch ein Polynom beschränkt ist.

Acknowledgements

First of all I would like to thank Professor Heribert Vollmer and Doctor Arne Meier for entrusting me with their research topic and providing valuable tips whereupon to direct my research efforts to. Further I would like to thank my former and ongoing fellow students Thorben Funke and Sonja Menßen for detailed proofreading when both of them had little time left due to the commencement of their work on PhD and master's thesis respectively. I am particularly grateful for Sonja "recruiting" me for her symphonic wind orchestra which provided a counterbalance in difficult times when everything seemed to fall apart.

Contents

1 Introduction 1

2 Preliminaries 3
- 2.1 Team based Propositional Logic . . . 3
- 2.2 Team based Modal Logic . . . 6
- 2.3 Enumeration problems . . . 10
- 2.4 Group action . . . 13

3 Enumeration in Propositional Logic 15
- 3.1 NP-complete enumeration problems . . . 17
- 3.2 Enumerating in $\mathcal{PDL}\backslash\{\vee\}$ . . . 19
  - 3.2.1 Simplifying $\mathcal{PDL}\backslash\{\vee\}$-formulas . . . 20
  - 3.2.2 The group action of flipping bits . . . 21
  - 3.2.3 Constructing $\mathcal{T}_k^0$ . . . 28
  - 3.2.4 The algorithm . . . 33
  - 3.2.5 Consequences of sorting by cardinality . . . 34
  - 3.2.6 Limiting memory space . . . 36
- 3.3 Enumeration for DNF-formulas . . . 41
  - 3.3.1 Sorting orbits . . . 42
  - 3.3.2 Sorting satisfying teams for product terms . . . 45
  - 3.3.3 The algorithm . . . 47

4 Enumeration in Modal Logic 49
- 4.1 NP-complete enumeration problems . . . 50
- 4.2 Enumerating polynomial verifiable fragments . . . 52
- 4.3 Enumerating maximal teams in $\mathcal{MDL}\backslash\{\vee,\Diamond\}$ . . . 54

5 Conclusion 59

A Referenced problems 61

Bibliography 63

Index 63

1 Introduction

Functional dependencies are a fundamental concept when describing the behaviour of various systems, claiming that if the outcome of event A is equal in two different scenarios, then the outcome of event B has to be identical as well. Hence B may be described by a function in A. Although the mapping from A to B may be complex or remains unknown completely—consider black box testing—in most cases its existence is obvious. For instance, deterministic behaviour is expressed by a functional dependency of the output relating to the input. Many data base schemes can be described by functional dependencies solely. For this reason straightforward constructs to express functional dependencies in modelling languages are invaluable.

In 2007 Vääanänen extended first-order logic by the dependence atom $=(P,Q)$, meaning that if several assignments agree on P, then they also have to agree on Q [Vä07]. As functional dependency only is meaningful when considering at least two scenarios, formulas in the so called *dependence logic* are evaluated not on single assignments but on sets of assignments, which are called teams [Hod97]. Later on, Väänänen introduced the dependence atom to modal logic, bringing forth *modal dependence logic* [Vä08].

The complexity of model checking and satisfiability in dependence logic is well researched [FG06, LV13, EL12]. However, in practical applications one is not only interested in the existence of solutions for a given formula, but in a set of all possible solutions. Consider a specification describing the requirements for a system. Then we would like to enumerate all implementations fulfilling the specification and select the one coinciding with our interests most. The mere existence of such an implementation is of no help in this regard. Depending on the problem, it is beneficial if the solutions are outputted in a predefined order. For instance, if small implementations are favoured due to limited capacity, the output should be ordered by its size.

In general, the amount of possible solutions is not bounded by a polynomial in the input size. Consequently, one cannot expect enumeration algorithms performing in polynomial time. However, by considering the elapsed time between two outputs, that is *delay*, instead of the total runtime, we obtain a classification

which is independent of the solution space [JYP88, Sch09, Cre+13]. The class **DelayP** covers all problems that permit a deterministic enumeration algorithm with polynomial delay. Observe that polynomial delay does not imply polynomial space consumption. If the memory usage should be bounded by a polynomial, one has to consider the class **DelaySpaceP**. Incremental delay is obtained when the delay of an enumeration algorithm is bounded by a polynomial in the input size and the amount of previously outputted solutions. Hence, the delay increases with the amount of found solutions. Problems permitting a deterministic enumeration algorithm with incremental delay are contained in **IncP**.

In this thesis we investigate the complexity of enumerating satisfying teams for various fragments of propositional and modal logic extended by the dependence atom. First, we identify fragments for which we do not expect to obtain enumeration algorithms with polynomial delay, as this would imply P = NP. In particular, the full propositional dependence logic is such a fragment. On that account we mainly consider formulas without the split junction operator $\vee$ and show that polynomial delay can be achieved if the size of outputted teams is restricted to a polynomial in the input.

Furthermore, we show that this restriction is necessary when the outputted teams should be sorted by their cardinality. Otherwise the delay cannot be polynomial. If we adapt parametrised complexity theory to enumeration problems as in [Cre+13] and take the size of outputted teams as a parameter, we are able to present an enumeration algorithm that performs in FPT-delay. Moreover, we extend our approach such that DNF-formulas with the classical disjunction operator $\veebar$ are covered as well. When polynomial space restrictions have to be fulfilled, we prove that polynomial delay cannot be achieved. However, incremental delay is realisable. We present an appropriate enumeration algorithm that outputs teams of low cardinality first.

For modal dependence logic, we show that the fragments constructed by either removing the $\Diamond$ or $\vee$ operator are not enumerable with polynomial delay unless P = NP. If both operators are absent, we can present reductions so that the problem of enumerating all maximal teams is equivalent to the problem of enumerating all maximal independent sets in undirected graphs. Note that the latter permits a **DelayP**-algorithm [JYP88]. At last, we show that any polynomial verifiable fragment of modal dependence logic permits a **DelaySpaceP**-algorithm, outputting teams in lexicographical order, and a **DelayP**-algorithm which sorts the output by its cardinality.

2 Preliminaries

In this chapter we establish the basic concepts used in this thesis. First we introduce the notion of teams regarding various logics. Then we present fundamental definitions for enumeration problems and extend them by parametrisations. At last, we recall some definitions and properties regarding group actions which are central for section 3.2.

The definitions and theorems of section 2.1 base on [DKV16] with slight modifications concerning notation. Section 2.3 merges concepts found in [Cre+13] and [Sch09, chapter 2].

2.1 Team based Propositional Logic

The semantics of functional dependencies becomes relevant only when pairing assignments. For this reason we introduce the concept of evaluating propositional formulas on sets of assignments, that is teams. When doing so, the evaluation on singletons is supposed to recover the semantics of the classical evaluation on assignments. The obvious approach of extending the semantics of the negation operator to teams, that is a team satisfies $\neg\varphi$ iff it falsifies φ, lacks an important property used throughout the thesis, namely downward closure. Instead, we use another interpretation that is derived from the semantics for teams satisfying atomic propositions. A team T satisfies an atomic proposition x iff all assignments in T assign x to 1. On the contrary, we say that T satisfies $\neg x$ iff all assignments in T assign x to 0. Note that this differs from the previous approach, which would require only one assignment of T assigning x to 0 in order to satisfy $\neg x$. On the downside, the latter negation may only be applied to atomic propositions and not to any formula. As a consequence, the definition of propositional logic has to be altered such that $\neg$ may only occur in front of variables. Due to De Morgan's laws, this restriction does not influence the expressivity of propositional logic.

Definition 2.1.1 (Propositional Logic) A propositional formula φ is defined by the grammar

$$\varphi ::= x \mid \neg x \mid 0 \mid 1 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi,$$

where x denotes an arbitrary variable. The set of all variables occurring in φ is denoted by $\mathrm{Var}(\varphi)$. The class of all formulas derived from this grammar is denoted by $\mathcal{PL}$.

Now we will specify the notion of teams and its interpretation on propositional formulas.

Definition 2.1.2 (Team) Let $\mathcal{V}$ be a set of variables. An *assignment* over $\mathcal{V}$ is a mapping

$$s\colon \mathcal{V} \to \{0,1\}.$$

We set

$$2^{\mathcal{V}} := \{s : s \text{ assignment over } \mathcal{V}\}.$$

A *team* T over $\mathcal{V}$ is a subset $T \subseteq 2^{\mathcal{V}}$. Consequently, the set of all teams over $\mathcal{V}$ is denoted by $\mathcal{P}\left(2^{\mathcal{V}}\right)$. If X is a subset of $\mathcal{V}$, we set

$$T\big|_X := \left\{s\big|_X : s \in T\right\},$$

where $s\big|_X$ is the restriction of s on X. If T has cardinality $k \in \mathbb{N}$, we say that T is a *k-team*. If φ is a formula, then a team over $\mathrm{Var}(\varphi)$ is called a *team for φ*. Equally, *assignments for φ* are defined to be assignments over $\mathrm{Var}(\varphi)$.

Definition 2.1.3 (Team based Propositional Logic) A team based propositional formula φ is defined by the grammar of Definition 2.1.1 extended by the rules

$$\varphi ::= \varphi \dot{\vee} \varphi \mid \varphi \circledcirc \varphi \mid =(P,Q),$$

where P, Q are sets of arbitrary variables. We write $=(x_1, x_2, \ldots, x_n)$ as a shorthand for $=(\{x_1, x_2, \ldots, x_{n-1}\}, \{x_n\})$. The size of P is called the *arity* of $=(P,Q)$.

Note that many other team based operators and atoms are dealt with in literature, for instance in [DKV16]. We do list only those concepts which are relevant in this thesis.

Notation 2.1.4 Let P be a set of productions. Then we set

$$\mathcal{F}(P) := \{\varphi \text{ formula} : \varphi \text{ is derived from productions in } P \text{ solely}\}.$$

For example, we have $\mathcal{PL} = \mathcal{F}(x, \neg, 0, 1, \wedge, \vee)$, where we abbreviate the productions by their corresponding atoms and operators.

If $Q = \{q_1, \dots, q_r\}$ is another set of productions, we set

$$\mathcal{F}(P)(q_1, \dots, q_r) := \mathcal{F}(P \cup Q) \quad \text{and}$$
$$\mathcal{F}(P)\backslash Q := \mathcal{F}(P \setminus Q).$$

We write $\mathcal{PDL} := \mathcal{PL}(=(\cdot))$ for the formulas of *Propositional Dependence Logic*.

Definition 2.1.5 Let φ be a team based propositional formula and T be a team over a superset of $\mathrm{Var}(\varphi)$. We define $T \models \varphi$ inductively by

$$\begin{aligned}
T \models x &:\Leftrightarrow s(x) = 1 \quad \forall s \in T, \\
T \models \neg x &:\Leftrightarrow s(x) = 0 \quad \forall s \in T, \\
T \models 1 &:\Leftrightarrow \text{true}, \\
T \models 0 &:\Leftrightarrow T = \emptyset, \\
T \models \varphi \wedge \psi &:\Leftrightarrow T \models \varphi \text{ and } T \models \psi, \\
T \models \varphi \vee \psi &:\Leftrightarrow \exists R, S \subseteq T : R \cup S = T \text{ and } R \models \varphi \quad \text{(lax split junction)} \\
&\qquad \text{and } S \models \psi, \\
T \models \varphi \dot{\vee} \psi &:\Leftrightarrow \exists R, S \subseteq T : R \cup S = T \text{ and } R \cap S = \emptyset \quad \text{(strict split junction)} \\
&\qquad \text{and } R \models \varphi \text{ and } S \models \psi, \\
T \models \varphi \mathbin{⩒} \psi &:\Leftrightarrow T \models \varphi \text{ or } T \models \psi, \quad \text{(classical disjunction)} \\
T \models =(P, Q) &:\Leftrightarrow \forall s, t \in T : s\big|_P = t\big|_P \Rightarrow s\big|_Q = t\big|_Q. \quad \text{(functional dependency)}
\end{aligned}$$

We say that T *satisfies* φ iff $T \models \varphi$ holds.

Note that we have $T \models (x \wedge \neg x)$ iff $T = \emptyset$. This observation motivates the definition for $T \models 0$.

Lemma 2.1.6 *Let φ be a team based propositional formula and s be an assignment for φ. Let $\varphi' \in \mathcal{PL}$ be the formula obtained from φ by replacing all occurrences of $\vee$, $\dot{\vee}$ and $⩒$ by $\vee$ in classical propositional logic as well as $=(P, Q)$ by 1. Then it holds that*

(i) $\emptyset \models \varphi$,

(ii) $\{s\} \models \varphi \Leftrightarrow s \models \varphi'$.

In particular, the evaluation in classical propositional logic occurs as the special case of evaluating singletons in team based propositional logic.

Proof: (i) follows immediately by induction. Due to (i), the team based definitions for $\vee$, $\dot{\vee}$ and $⩒$ translate to the disjunction operator in classical propositional logic. Note that when evaluating a formula on a team T, its sub-formulas are

evaluated on subsets of T only. For this reason $=(P,Q)$ is evaluated on singletons or the empty set. In both cases $=(P,Q)$ is satisfied trivially and we may replace the dependency atom by 1. This proves (ii). □

For all formulas φ considered in this thesis it holds that if a team T satisfies φ, then already all subsets of T have to satisfy φ. This is a valuable property of team based logics when it comes to enumerating satisfying teams. If T is satisfying, we can output all subsets of T without the computational cost of verifying them. On the contrary, if we have ascertained that T falsifies φ, we do not need to check the teams containing T anymore. We formalise this property now.

Definition 2.1.7 (Downward closure)

- A team based propositional formula φ is called *downward closed*, if for every team T it holds that
$$T \models \varphi \Rightarrow \forall S \subseteq T : S \models \varphi.$$
- An atom is called *downward closed* if its corresponding atomic formula is downward closed.
- An operator $\circ$ of arity k is called *downward closed* if $\circ(\varphi_1,\dots,\varphi_k)$ is downward closed for all downward closed formulas φ_i, $i = 1,\dots,k$.
- A class ϕ of team based propositional formulas is called *downward closed* if all formulas in ϕ are downward closed.

Lemma 2.1.8 *All atoms and operators from Definition 2.1.1 and 2.1.3 are downward closed.*

Proof: As we will see later on in Remark 2.2.6, team based propositional logic is a special case of team based modal logic. Hence, the claim follows from the modal version of this lemma (see Lemma 2.2.9). □

Remark 2.1.9 Let P be a set of productions corresponding to downward closed atoms or operators. By a simple induction it follows that $\mathcal{F}(P)$ is downward closed. In particular $\mathcal{PL}$ and $\mathcal{PDL}$ are downward-closed.

2.2 Team based Modal Logic

Modal logic extends propositional logic by introducing two modal operators $\Diamond$ and $\Box$. Formulas are not evaluated on assignments but on Kripke models, which do include a temporal component. A Kripke model consists of a set of worlds, where each world defines a set of variables holding in its context. On that account

a world corresponds to an assignment in propositional logic and teams are given by sets of worlds. In contrast to propositional logic, pairs of worlds may be related in order to model dependencies. The semantics of the modal operators is given by these relations. We formalise these concepts by the following definitions.

Definition 2.2.1 (Modal Logic) A modal formula φ is defined by the grammar

$$\varphi ::= x \mid \neg x \mid 0 \mid 1 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi,$$

where x denotes an arbitrary variable. The set of all variables occurring in φ is denoted by $\mathrm{Var}(\varphi)$. The class of all formulas derived from this grammar is denoted by $\mathcal{ML}$.

Team based modal formulas are obtained by extending the grammar of $\mathcal{ML}$ as in Definition 2.1.3. We write $\mathcal{MDL} := \mathcal{ML}(=(\cdot))$ for the formulas of *Modal Dependence Logic*.

As before when defining $\mathcal{PL}$, we permit the negation operator in front of variables only. Due to De Morgan and the equivalence of $\Diamond$ and $\neg\Box\neg$ in the classical definition of $\mathcal{ML}$, the expressive power of $\mathcal{ML}$ is not affected by this slight modification.

Definition 2.2.2 (Kripke model & Team) Let $\mathcal{V}$ be a set of variables. Then a *Kripke model* $K = (W, R, \pi)$ over $\mathcal{V}$ is given by

- a set of worlds W,
- a binary relation $R \subseteq W \times W$,
- a mapping $\pi\colon W \to \mathcal{P}(\mathcal{V})$ that determines the variables holding in a world.

If φ is a formula, then a Kripke model over $\mathrm{Var}(\varphi)$ is called a Kripke model *for* φ.

A *team* T of K is a subset of worlds in W. Consequently, the set of all teams of K is denoted by $\mathcal{P}(K) := \mathcal{P}(W)$. The set of successors for a world w or a team T is given by

$$\mathrm{succ}(w) := \{v \in W : (w, v) \in R\},$$
$$\mathrm{succ}(T) := \bigcup_{w \in T} \mathrm{succ}(w).$$

Definition 2.2.3 Let φ be a team based modal formula and $K = (W, R, \pi)$ be a Kripke model over a superset of $\mathrm{Var}(\varphi)$. Let T be a team of K. We define $K, T \models \varphi$ inductively by

$$
\begin{aligned}
K, T \models x \quad &:\Leftrightarrow x \in \pi(w) \quad \forall w \in T,\\
K, T \models \neg x \quad &:\Leftrightarrow x \notin \pi(w) \quad \forall w \in T,\\
K, T \models 1 \quad &:\Leftrightarrow \text{true},\\
K, T \models 0 \quad &:\Leftrightarrow T = \emptyset,\\
K, T \models \varphi \wedge \psi \quad &:\Leftrightarrow K, T \models \varphi \text{ and } K, T \models \psi,\\
K, T \models \varphi \vee \psi \quad &:\Leftrightarrow \exists R, S \subseteq T : R \cup S = T \text{ and } K, R \models \varphi \text{ and } K, S \models \psi,\\
K, T \models \varphi \dot{\vee} \psi \quad &:\Leftrightarrow \exists R, S \subseteq T : R \cup S = T \text{ and } R \cap S = \emptyset \text{ and } K, R \models \varphi \text{ and } K, S \models \psi,\\
K, T \models \varphi \otimes \psi \quad &:\Leftrightarrow K, T \models \varphi \text{ or } K, T \models \psi,\\
K, T \models \Diamond \varphi \quad &:\Leftrightarrow \exists T' \subseteq \mathrm{succ}(T)\ \forall w \in T : \mathrm{succ}(w) \cap T' \neq \emptyset \text{ and } K, T' \models \varphi,\\
K, T \models \Box \varphi \quad &:\Leftrightarrow K, \mathrm{succ}(T) \models \varphi,\\
K, T \models =(P, Q) \quad &:\Leftrightarrow \forall w, v \in T : \pi(w) \cap P = \pi(v) \cap P \Rightarrow \pi(w) \cap Q = \pi(v) \cap Q.
\end{aligned}
$$

We say that T *satisfies* φ *respecting* K iff $K, T \models \varphi$ holds.

Definition 2.2.4 Let ϕ be a class of team based modal formulas. We write $\mathcal{K}(\phi)$ for the set containing all valid pairings of formulas in ϕ and Kripke models, precisely:

$$\mathcal{K}(\phi) := \{(\varphi, K) : \varphi \in \phi \text{ and } K \text{ is a Kripke model for } \varphi\}.$$

Problem 2.2.5 (Model checking) Let ϕ be a class of team based modal formulas. Then we define the *model checking problem in* ϕ by

$$\mathrm{MC}(\phi) := \{(K, T, \varphi) : (\varphi, K) \in \mathcal{K}(\phi),\ T \in \mathcal{P}(K) \text{ with } K, T \models \varphi\}.$$

Remark 2.2.6 Model checking in team based propositional logic is a special case of model checking in team based modal logic. Let φ be a team based propositional formula and T be a team for φ. We define the Kripke model $K = (W, R, \pi)$ with $W = 2^{\mathrm{Var}(\varphi)}$, $R = \emptyset$ and $\pi(s) = \{x \in \mathrm{Var}(\varphi) : s(x) = 1\}$ for all $s \in W$. Then it is easy to see that

$$T \models \varphi \Leftrightarrow K, T \models \varphi,$$

where on the right hand side φ is regarded as a team based modal formula.

Remark 2.2.7 Let φ be a team based modal formula. Then a simple induction yields $K, \emptyset \models \varphi$ for every Kripke model K for φ. This corresponds to the first claim of Lemma 2.1.6. However, we do not obtain a modal version of the second claim without loosing the property that the reduction $\varphi \mapsto \varphi'$ is polynomial.

In [Sev09, section 4] a non polynomial reduction from φ to $\varphi' \in \mathcal{ML}$ is presented such that $K, \{w\} \models \varphi \Leftrightarrow K, w \models \varphi'$ holds for every Kripke model K over $\mathrm{Var}(\varphi)$ and every world w in K. In fact we question the existence of a polynomial reduction with this property, as this would imply P = NP.

To prove this claim, we point out that model checking in team based modal logic can be reduced to model checking on singletons by checking $\Box\varphi$ on $\{w\}$, where w is a new world appended to the Kripke model which relates to all worlds contained in the team we wish to check. Consequently, a polynomial reduction would result in a polynomial reduction of model checking in $\mathcal{MDL}$ on model checking in $\mathcal{ML}$. By [EL12], checking $\mathcal{MDL}$ formulas is NP-complete, whereas checking $\mathcal{ML}$ formulas is in P.

Definition 2.2.8 (Downward closure) A team based modal formula φ is called *downward closed*, if for every model K for φ and team T of K it holds that

$$K, T \models \varphi \Rightarrow \forall S \subseteq T : K, S \models \varphi.$$

The downwards closure property of atoms, operators and classes of team based modal formulas is defined similar to Definition 2.1.7.

Lemma 2.2.9 *All atoms and operators defined for team based modal logic are downward closed.*

Proof: It is easy to see that the atoms x, $\neg x$, 0, 1, $=(\cdot)$ are downward closed. Let φ, ψ be two downward closed formulas and K be a Kripke model over the union of $\mathrm{Var}(\varphi)$ and $\mathrm{Var}(\psi)$. Let T be a team with $K, T \models \varphi \wedge \psi$. Then we have $K, T \models \varphi$ and $K, T \models \psi$, so that $K, S \models \varphi$ and $K, S \models \psi$ holds for every subset $S \subseteq T$. It follows that $K, S \models \varphi \wedge \psi$. For this reason $\wedge$ is downward closed. The proof for $\otimes$ is similar.

Now let T be a team with $K, T \models \varphi \vee \psi$. Then $R, S \subseteq T$ with $R \cup S = T$ exist such that $K, R \models \varphi$ and $K, S \models \psi$. Let T' be a subset of T. Set $R' := R \cap T'$ and $S' := S \cap T'$. Obviously it holds that $R' \cup S' = T'$. As φ and ψ are downward closed, we conclude $K, R' \models \varphi$ and $K, S' \models \psi$. In other words $K, T' \models \varphi \vee \psi$, proving that $\vee$ is downward closed. Similarly, we obtain that $\dot{\vee}$ is downward closed.

Next we cover the modal operators. Let T be a team with $K, T \models \Diamond\varphi$. Then there exists a subset T' of $\mathrm{succ}(T)$ such that $\mathrm{succ}(w) \cap T' \neq \emptyset$ for all $w \in T$ and $K, T' \models \varphi$. Let $S \subseteq T$ and set $S' := T' \cap \mathrm{succ}(S)$. We still have $\mathrm{succ}(w) \cap S' \neq \emptyset$ for all $w \in S$. Due to the downward closure of φ it follows that $K, S' \models \varphi$. Consequently, $K, S \models \Diamond\varphi$ and $\Diamond$ is downward closed. Because $\mathrm{succ}(\cdot)$ preserves inclusion, we obtain that $\Box$ is downward closed. □

Remark 2.2.10 With the same argument as in Remark 2.1.9 it follows that $\mathcal{ML}$ and $\mathcal{MDL}$ are downward-closed.

2.3 Enumeration problems

Now we formalise the concept of enumerating solutions for a given problem. In this thesis we extend the classical notion of enumeration problems by ordered outputs and parametrisations.

When implementing algorithms later on, we are going to make intensive use of data structures with logarithmic costs for standard operations. Those data structures require random access operations in constant time which are not available in Turing machines. On that account we use the computation model based on random-access machines (RAM) presented in [Cre+17] instead of the standard computation model based on Turing machines. A detailed description of the RAM computation model may be found in [Str10, section 1.2.2]. The space occupied by a RAM is given by the total amount of used registers, provided that the content of each register is polynomially bounded in the size of the input.

Definition 2.3.1 (Enumeration problem) Let Σ be a finite alphabet and $(S, \leq)$ a partially ordered set (poset) of possible solutions. An *enumeration problem* is a triple $E = (Q, \mathrm{Sol}, \leq)$ such that

1. $Q \subset \Sigma^*$ is a decidable language,
2. $\mathrm{Sol}\colon Q \to \mathcal{P}(S)$ is a computable function.

For an element $x \in Q$ we call x an *instance* and $\mathrm{Sol}(x)$ its *set of solutions*. If $\leq$ is the trivial poset given by

$$x \leq y :\Leftrightarrow x = y,$$

we omit it and write $E = (Q, \mathrm{Sol})$.

Notation 2.3.2 Let $(X, \leq)$ be a poset. Let $x, y \in X$. Then we write $x < y$ for

$$x \leq y \text{ and } x \neq y.$$

Definition 2.3.3 (Enumeration algorithm) Let $E = (Q, \mathrm{Sol}, \leq)$ be an enumeration problem. A deterministic algorithm $\mathcal{A}$ is an *enumeration algorithm* for E if for every input $x \in Q$

1. $\mathcal{A}$ terminates,
2. $\mathcal{A}$ outputs the set $\mathrm{Sol}(x)$ without duplicates,
3. for every $s, t \in \mathrm{Sol}(x)$ with $s < t$ the solution s is outputted before t.

It is common that the amount of solutions for a given instance of an enumeration problem is not polynomially bounded by the size of the instance. Consequently one cannot expect that enumeration algorithms perform in polynomial time.

However, when the solutions are not required all at once but consecutively, it is reasonable to analyse the delay until obtaining the next solution. Note that this delay may be polynomial even if the amount of all solutions is not.

Definition 2.3.4 (Delay) Let $\mathcal{A}$ be an enumeration algorithm for the enumeration problem $E = (Q, \text{Sol}, \leq)$ and $x \in Q$. The *i-th delay* of $\mathcal{A}$ is defined as the elapsed time between outputting the i-th and $(i+1)$-th solution of $\text{Sol}(x)$, where the 0-th and $(|\text{Sol}(x)|+1)$-th solutions are considered to be the start and the end of the computation respectively. The 0-th delay is called *precalculation delay* and the $|\text{Sol}(x)|$-th delay is called *postcalculation delay*.

Definition 2.3.5 Let $\mathcal{A}$ be an enumeration algorithm for an enumeration problem $E = (Q, \text{Sol}, \leq)$.

1. $\mathcal{A}$ is an ***IncP**-algorithm* if there exists a polynomial p such that the i-th delay on input $x \in Q$ is bounded by $p(|x| + i)$.
2. $\mathcal{A}$ is a ***DelayP**-algorithm* if there exists a polynomial p such that all delays on input $x \in Q$ are bounded by $p(|x|)$.
3. $\mathcal{A}$ is a ***DelaySpaceP**-algorithm* if it is a **DelayP**-algorithm using polynomial amount of space with respect to the size of the input.

Definition 2.3.6 (IncP, DelayP, DelaySpaceP) The class **IncP** contains all enumeration problems admitting an **IncP**-algorithm. We define the classes **DelayP** and **DelaySpaceP** accordingly.

Many classical problems like Sat or Clique require the existence of a solution with a specific property for a given instance. As a consequence, those problems give rise to enumeration problems. Vice versa, we can obtain classical problems from enumeration problems.

Problem 2.3.7 Let $E = (Q, \text{Sol}, \leq)$ be an enumeration problem. Then we define the problem $\textsc{Decide}(E) \subseteq Q$ as follows:

$$x \in \textsc{Decide}(E) :\Leftrightarrow \text{Sol}(x) \neq \emptyset \quad \text{for } x \in Q.$$

Remark 2.3.8 For any enumeration problem $E = (Q, \text{Sol}, \leq)$ we have

$$E \in \textbf{DelayP} \Rightarrow \textsc{Decide}(E) \in P.$$

In order to prove this claim, we consider the RAM $R(x)$ on input $x \in Q$ that aborts after outputting the first element $\text{Sol}(x)$. Because of $E \in \textbf{DelayP}$ we can assume

that $R(x)$ performs in polynomial time. The following Turing machine M decides Decide(E):

Input: $x \in Q$

Simulate $R(x)$;
if Output of $R(x)$ is nonempty **then**
  accept;
else
  reject;
end

By [CR73], $R(x)$ can be simulated in polynomial time, proving Decide$(E) \in$ P.

Now we introduce the parametrised version of enumeration problems. The extensions are similar to those when extending P to FPT.

Definition 2.3.9 (Parametrised enumeration problem) A problem $(Q, \text{Sol}, \leq)$ together with a polynomial time computable *parametrisation* $\kappa\colon \Sigma^* \to \mathbb{N}$ is called a *parametrised enumeration problem* $E = (Q, \kappa, \text{Sol}, \leq)$. As before, if $\leq$ is omitted, we assume $\leq$ to be trivial.

Definition 2.3.10 (Slice) Let $E = (Q, \kappa, \text{Sol}, \leq)$ be a parametrised enumeration problem and $\ell \in \mathbb{N}$. Then the *ℓ-th slice E_ℓ of E* is the (non-parametrised) enumeration problem given by

$$E_\ell := (Q_\ell, \text{Sol}\big|_{Q_\ell}, \leq\big|_{Q_\ell}),$$

where $Q_\ell := \{x \in Q : \kappa(x) = \ell\}$ is the ℓ-th slice of the parametrised problem (Q, κ). For $I \subseteq \mathbb{N}$, *the union E_I of slices of E* is defined by

$$E_I := (Q_I, \text{Sol}\big|_{Q_I}, \leq\big|_{Q_I}) \quad \text{with} \quad Q_I := \bigcup_{i \in I} Q_i.$$

We say that the union is *finite* if I is finite.

Definition 2.3.11 Let $\mathcal{A}$ be an enumeration algorithm for a parametrised enumeration problem $E = (Q, \kappa, \text{Sol}, \leq)$. If there are a polynomial p and a computable function $f\colon \mathbb{N} \to \mathbb{N}$ such that the i-th delay on input $x \in Q$ is bounded by $f(\kappa(x)) \cdot p(|x| + i)$, then $\mathcal{A}$ is an ***IncFPT**-algorithm*. We call $\mathcal{A}$ a **DelayFPT**-algorithm if all delays on input $x \in Q$ are bounded by $f(\kappa(x)) \cdot p(|x|)$.

Definition 2.3.12 (IncFPT, DelayFPT) The class **IncFPT** contains all enumeration problems that admit an **IncFPT**-algorithm. The class **DelayFPT** is defined accordingly.

2.4 Group action

The following section provides a compact introduction to group actions on sets. The proofs of the following theorems may be found in any established lecture book in algebra, for instance [Rot95].

Definition 2.4.1 (Group action, [Rot95, p. 55]) Let G be a group with identity element e and X be a set. A *group action of G on X*, denoted by $G \circlearrowright X$, is a mapping $G \times X \to X$, $(g, x) \mapsto gx$, with

(i) $ex = x \quad \forall x \in X$

(ii) $(gh)x = g(hx) \quad \forall g, h \in G,\ x \in X.$

Remark 2.4.2 Let G be a group and X a set.

- The mapping $(g, h) \mapsto gh$ for $g, h \in G$ defines a group action of G on itself.
- A group action $G \circlearrowright X$ induces a group action of G on $\mathcal{P}(X)$ by

$$gS := \{gs : s \in S\} \quad \forall g \in G,\ S \subseteq X.$$

  Note that this group action preserves the cardinality of sets.

Definition 2.4.3 (Orbit, [Rot95, p. 56]) Let $G \circlearrowright X$ be a group action and $x \in X$. Then the *orbit of x* is given by

$$Gx := \{gx : g \in G\} \subseteq X.$$

Proposition 2.4.4 (Partitioning, [Rot95, p. 56]) *Let $G \circlearrowright X$ be a group action and $x, y \in X$. Then either $Gx = Gy$ or $Gx \cap Gy = \emptyset$. Consequently, the orbits of $G \circlearrowright X$ partition the set X.*

Definition 2.4.5 (Stabiliser, [Rot95, p. 56]) Let $G \circlearrowright X$ be a group action and $x \in X$. The *stabiliser subgroup of x* is given by

$$G_x := \{g \in G : gx = x\}$$

and indeed is a subgroup of G.

Proposition 2.4.6 (Orbit-Stabiliser theorem, [Rot95, Theorem 3.19]) *Let G be a finite group acting on a set X. Let $x \in X$. Then the mapping $gG_x \mapsto gx$ is a bijection from G/G_x to Gx. In particular it holds that $|Gx| \cdot |G_x| = |G|$.*

Proposition 2.4.7 (Cauchy-Frobenius lemma, [Rot95, Theorem 3.22]) *Let G be a finite group acting on a set X. Then the number of orbits is given by*

$$\frac{1}{|G|} \sum_{g \in G} |\{x \in X : gx = x\}|.$$

3 Enumeration in Propositional Logic

This chapter focuses on researching the problem of enumerating all satisfying teams for various fragments of team based propositional logic. After introducing the problem EnumTeamPl and its parametrised version P-EnumTeamPl, we present fragments for which the existence of efficient enumeration algorithms would yield P = NP. Then we develop two enumeration algorithms for $\mathcal{PDL}\backslash\{\vee\}$, either guaranteeing polynomial delay or incremental delay in polynomial space. At last, we propose a modification of the previous **DelayP**-algorithm such that formulas in disjunctive normal form (DNF) are supported.

Problem 3.0.1 (EnumTeamPl) Let ϕ be a class of team based propositional formulas and $f : \mathbb{N} \to \mathbb{N}$ be a computable function. Then we define

$$\text{EnumTeamPl}(\phi,\ f) := (\phi,\ \text{Sol}_f)$$

where

$$\text{Sol}_f(\varphi) := \left\{\emptyset \neq T \in \mathcal{P}\left(2^{\text{Var}(\varphi)}\right) : T \models \varphi,\ |T| \leq f(|\varphi|)\right\} \quad \text{for } \varphi \in \phi.$$

Problem 3.0.2 (P-EnumTeamPl) Let ϕ be a class of team based propositional formulas. Then we define

$$\text{P-EnumTeamPl}(\phi) := (\phi \times \mathbb{N},\ \kappa,\ \text{Sol})$$

where

$$\kappa((\varphi, k)) := k \quad \text{and}$$
$$\text{Sol}((\varphi, k)) := \left\{\emptyset \neq T \in \mathcal{P}\left(2^{\text{Var}(\varphi)}\right) : T \models \varphi,\ |T| \leq k\right\}$$

for $(\varphi, k) \in \phi \times \mathbb{N}$.

Notation 3.0.3 We write $\text{EnumTeamPl}(\phi)$ for $\text{EnumTeamPl}(\phi,\ n \mapsto 2^n)$. Since $|T| \leq 2^{|\varphi|}$ holds for every team T for φ, we effectively eliminate the cardinality constraint.

As we shall see, the order in which the teams are outputted plays an important role in the following reasoning. There are two natural orders on teams to consider.

Definition 3.0.4 (Order of cardinality) Let R, S be two teams. Then we define a partial order on the set of all teams by

$$R \leq_{\text{size}} S :\Leftrightarrow |R| < |S| \text{ or } R = S.$$

When a formula φ is given, we assume to have a total order $\leq$ on $2^{\text{Var}(\varphi)}$ such that comparing two elements is possible in $\mathcal{O}(|\text{Var}(\varphi)|)$ and iterating over the set of all assignments is feasible with delay $\mathcal{O}(|\text{Var}(\varphi)|)$. When interpreting each assignment as a binary-encoded integer, we obtain an appropriate order on $2^{\text{Var}(\varphi)}$ by translating the order on $\mathbb{N}_0$. If necessary, one can demand that adjacent assignments differ in only one place by using the order induced by the Gray code. Now we are able to define the second order.

Definition 3.0.5 (Lexicographical order) Let $R = \{r_1, \ldots, r_n\}$, $S = \{s_1, \ldots, s_m\}$ be two teams such that $r_1 < \cdots < r_n$ and $s_1 < \cdots < s_m$. Set

$$i := \max\{j \in \mathbb{N}_0 : j \leq \min(n, m),\ r_\ell = s_\ell\ \forall \ell \in \{1, \ldots, j\}\}.$$

Then we define a partial order on $\mathcal{P}\left(2^{\text{Var}(\varphi)}\right)$ by

$$R \leq_{\text{lex}} S :\Leftrightarrow \begin{cases} n \leq m, & i = \min(n, m) \\ r_{i+1} < s_{i+1}, & \text{else.} \end{cases}$$

Remark 3.0.6 The lexicographical order is a total order that does not extend the order of cardinality. For example we have $\{00, 01, 10\} <_{\text{lex}} \{00, 10\}$ when assignments are ordered according to their integer representation.

Problem 3.0.7 Let ϕ be a class of team based propositional formulas and let $f : \mathbb{N} \to \mathbb{N}$ be a computable function. Then we define

$$\textsc{EnumTeamPlLex}(\phi,\ f) := (\phi,\ \text{Sol}_f,\ \leq_{\text{lex}}),$$
$$\textsc{EnumTeamPlSize}(\phi,\ f) := (\phi,\ \text{Sol}_f,\ \leq_{\text{size}})$$

with Sol_f as in Problem 3.0.1. P-EnumTeamPlLex and P-EnumTeamPlSize are defined accordingly.

3.1 NP-complete enumeration problems

The purpose of this section is to identify fragments ϕ of team based propositional logic for that EnumTeamPl(ϕ, $\cdot$) and P-EnumTeamPl(ϕ) cannot be enumerated in polynomial delay respectively FPT-delay unless P = NP.

Definition 3.1.1 Let E be an enumeration problem. We say that E is NP-*complete* if Decide(E) is NP-complete. A parametrised enumeration problem E_p is NP-*complete* if $I \subseteq \mathbb{N}$, $|I| < \infty$, exists such that the finite union of slices $(E_p)_I$ of E_p is NP-complete.

Remark 3.1.2 The readers being familiar with parametrised complexity theory may notice that a parametrised enumeration problem $E = (Q, \kappa, \text{Sol}, \leq)$ is NP-complete iff the corresponding parametrised problem (Decide(E), κ) is para-NP-complete under fpt-reductions, provided that (Decide(E), κ) is nontrivial and contained in para-NP ([see FG06, Theorem 2.14]).

Lemma 3.1.3 *Let E be an* NP*-complete [parametrised] enumeration problem. Then it holds that*

$$E \in \textbf{\textit{DelayP}}\ [\textbf{\textit{DelayFPT}}] \Rightarrow \text{P} = \text{NP}.$$

Proof: Let E be a non-parametrised enumeration problem such that Decide(E) is NP-complete. If $E \in$ **DelayP** holds, Remark 2.3.8 yields Decide(E) $\in$ P. Accordingly, we obtain P = NP.

Now let E be a parametrised enumeration problem. Consequently, we find $I \subseteq \mathbb{N}$, $|I| < \infty$, such that E_I is NP-complete. Let $\mathcal{A}$ be a **DelayFPT**-algorithm for E. By restricting $\mathcal{A}$ to inputs with parameter values in I, we obtain a **DelayP**-algorithm for E_I. On that account P = NP follows by applying the lemma to the non-parametrised enumeration problem E_I.

□

Theorem 3.1.4 *Let ϕ be a downward closed class of team based propositional formulas such that a polynomial time computable function $f : \mathcal{PL} \to \phi$ exists that embeds $\mathcal{PL}$ in ϕ, meaning*

$$\forall \varphi \in \mathcal{PL}: \quad Var(\varphi) = Var(f(\varphi)') \quad \textit{and} \quad s \models f(\varphi)' \Leftrightarrow s \models \varphi\ \forall s \in 2^{Var(\varphi)},$$

where $\varphi \mapsto \varphi'$ is defined as in Lemma 2.1.6. Then it holds that EnumTeamPl(ϕ, f) *and* P-EnumTeamPl(ϕ) *are* NP*-complete for all computable functions $f : \mathbb{N} \to \mathbb{N}$.*

Proof: First note that a formula in ϕ is satisfiable iff it is satisfied by a 1-team. For this reason we have

$$\textsc{Decide}(\textsc{EnumTeamPl}(\phi,\ f)) = \textsc{Decide}(\textsc{EnumTeamPl}(\phi,\ g))$$

for all computable functions $f, g\colon \mathbb{N} \to \mathbb{N}$. Furthermore for $I \subseteq \mathbb{N}$ it follows that

$$\varphi \in \textsc{Decide}(\textsc{EnumTeamPl}(\phi) \text{ and } k \in I \Leftrightarrow (\varphi, k) \in \textsc{Decide}(\text{P-}\textsc{EnumTeamPl}(\phi)_I)$$

for all $\varphi \in \phi$ and $k \in \mathbb{N}$. Consequently, it is enough to show that $\textsc{EnumTeamPl}(\phi)$ is NP-complete.

Let $\varphi \in \phi$. Then we have

$$\begin{aligned}
\varphi \in \textsc{Decide}(\textsc{EnumTeamPl}(\phi)) & \\
&\Leftrightarrow \exists \emptyset \neq T \in \mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right) : T \models \varphi \\
&\Leftrightarrow \exists s \in 2^{\mathrm{Var}(\varphi)} : \{s\} \models \varphi && (\phi \text{ downward closed}) \\
&\Leftrightarrow \exists s \in 2^{\mathrm{Var}(\varphi)} : s \models \varphi' && (\text{Lemma 2.1.6}) \\
&\Leftrightarrow \varphi' \in \textsc{Sat}.
\end{aligned}$$

We obtain $\textsc{Decide}(\textsc{EnumTeamPl}(\phi)) \leq_m^P \textsc{Sat}$ via $\varphi \mapsto \varphi'$. As we have $\textsc{Sat} \in \mathrm{NP}$, it follows that $\textsc{Decide}(\textsc{EnumTeamPl}(\phi)) \in \mathrm{NP}$.

In order to show the NP-hardness, we reduce the NP-hard problem 3CNF-$\textsc{Sat}$ to $\textsc{Decide}(\textsc{EnumTeamPl}(\phi))$. Let φ be a 3CNF-formula. Then we have $\varphi \in \mathcal{PL}$ and the reduction is given by f because of

$$\begin{aligned}
\varphi \in \text{3CNF-}\textsc{Sat} &\Leftrightarrow \exists s \in 2^{\mathrm{Var}(\varphi)} : s \models \varphi \\
&\Leftrightarrow \exists s \in 2^{\mathrm{Var}(f(\varphi)')} : s \models f(\varphi)' \\
&\Leftrightarrow \exists s \in 2^{\mathrm{Var}(f(\varphi))} : \{s\} \models f(\varphi) \\
&\Leftrightarrow \exists \emptyset \neq T \in \mathcal{P}\left(2^{\mathrm{Var}(f(\varphi))}\right) : T \models f(\varphi) \\
&\Leftrightarrow f(\varphi) \in \textsc{Decide}(\textsc{EnumTeamPl}(\phi)).
\end{aligned}$$

□

Corollary 3.1.5 *Let P be a set of productions such that*

$$\{x, \neg x, 0, 1, \wedge\} \subseteq P, \quad \{\vee, \dot{\vee}, \otimes\} \cap P \neq \emptyset.$$

Unless P = NP *holds, it follows that* $\textsc{EnumTeamPl}(\mathcal{F}(P),\ f) \notin \boldsymbol{DelayP}$ *for all computable functions* $f\colon \mathbb{N} \to \mathbb{N}$ *and* $\text{P-}\textsc{EnumTeamPl}(\mathcal{F}(P)) \notin \boldsymbol{DelayFPT}$.

Proof: Set $\phi = \mathcal{F}(P)$. Due to Remark 2.1.9, ϕ is downward closed. We can embed $\mathcal{PL}$ in ϕ by replacing the operator $\vee$ in $\mathcal{PL}$ by one operator from the set $\{\vee, \dot{\vee}, \circledcirc\} \cap P$. It follows that $\varphi = f(\varphi)'$ by Lemma 2.1.6. Now we can apply Theorem 3.1.4 and obtain the claim by Lemma 3.1.3. □

As a result of the previous corollary we do not expect to obtain **DelayP**- or **DelayFPT**-algorithms for $\mathcal{PDL}$. Consequently, the largest fragments of $\mathcal{PDL}$ which possibly do not result in NP-complete enumeration problems are $\mathcal{PDL}\backslash\{\wedge\}$, $\mathcal{PDL}\backslash\{\neg\}$ and $\mathcal{PDL}\backslash\{\vee\}$. Note that the enumeration problems for $\mathcal{PDL}(\dot{\vee})\backslash\{\vee\}$ and $\mathcal{PDL}(\circledcirc)\backslash\{\vee\}$ still are NP-complete.

3.2 Enumerating in $\mathcal{PDL}\backslash\{\vee\}$

Now we would like to deduce an approach of enumerating satisfying teams for the fragment $\mathcal{PDL}\backslash\{\vee\}$ of propositional logic. The delay of the resulting algorithm is polynomial in terms of the size of the input and the outputted teams. As teams may grow exponentially large according to the input size, the delay will not be polynomial in the classical sense of **DelayP**. Hence we go over to **DelayFPT** and set the maximal cardinality of outputted teams as the parameter. Note that the drawback of having polynomial delay in the output is minor. When subsequent algorithms process the outputted teams, they have to input them first, requiring at least linear time in the output size.

In fact, we will see that we cannot obtain a **DelayP**-algorithm when the output is sorted by cardinality. This sorting, however, is an inherent characteristic of our algorithm as satisfying teams of cardinality k are constructed by analysing those of cardinality $k-1$.

Before diving into details, we introduce some notation used in this chapter.

Notation 3.2.1 Let $\varphi \in \mathcal{PDL}\backslash\{\vee\}$ be fixed and $k \in \mathbb{N}_0$. Then we set

$$\begin{aligned} n &:= |\mathrm{Var}(\varphi)|, \\ \mathcal{T}_k &:= \left\{T \in \mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right) : T \models \varphi, |T| = k\right\}, \\ \mathcal{T}_k^0 &:= \{T \in \mathcal{T}_k : (\forall x \in \mathrm{Var}(\varphi) : x \mapsto 0) \in T\}, \\ t_k &:= |\mathcal{T}_k|, \\ t_k^0 &:= |\mathcal{T}_k^0|. \end{aligned}$$

We write $\mathrm{Var}(\varphi) = \{x_1, \ldots, x_n\}$. An assignment $s \in 2^{\mathrm{Var}(\varphi)}$ is depicted as a sequence of 0 and 1, precisely:

$$s = s(x_1)s(x_2)\ldots s(x_n).$$

Example 3.2.2 For $\varphi := =(x_1, x_2)$ we have:

$$\begin{aligned} n &= 2, \\ \mathcal{T}_2 &= \{\{00,10\},\{00,11\},\{01,10\},\{01,11\}\}, \\ \mathcal{T}_2^0 &= \{\{00,10\},\{00,11\}\}, \\ \mathcal{T}_3 &= \mathcal{T}_3^0 = \emptyset. \end{aligned}$$

Example 3.2.3 (Robot Soccer) Consider the following scenario: A team of students wishes to stage a match of robot soccer. They have to assemble two teams of robots, where each robot consists of a COM-unit and a mobility-unit. Both units are manufactured by two different companies C_0 and C_1. In order to enable co-operation, robots of the same team have to be equipped with the same COM-units. Furthermore each robot is assigned to one of two maintenance rooms. As the space in the maintenance rooms is limited, each room provides tools for maintaining the communication-unit of only one company. The same applies for the mobility-unit. Consequently, all robots assigned to the same maintenance room have to be equipped with the same units. Due to rumours concerning the reliability of the mobility-units manufactured by C_0, the students wish to install mobility-units of company C_1 only. In order to minimise the risk of failure, as many robots as possible has to be configured differently. However, each pair of two robots still have to be compatible concerning the previous restrictions.

We model each configuration by four binary properties $x_1, \ldots, x_4$, where x_1 describes the team membership of a robot and x_2 identifies the maintenance room the robot is assigned to. The third and fourth property provide the manufacturer of the COM- and mobility-unit. A set of configurations T is conform with the scenario iff

$$T \models \varphi \quad \text{with} \quad \varphi := x_4 \wedge =(x_1, x_3) \wedge =(\{x_2\},\{x_3, x_4\}).$$

For this reason, the students are interested in enumerating all satisfying teams T for φ. We will come back to this example later on to illustrate the algorithm presented in the next sections.

3.2.1 Simplifying $\mathcal{PDL}\backslash\{\vee\}$-formulas

Let $\varphi \in \mathcal{PDL}\backslash\{\vee\}$. As the only logical connective in φ is $\wedge$, there exist $I, J \subseteq \mathrm{Var}(\varphi)$, $L \subseteq \mathbb{N}$ and $P_\ell, Q_\ell \subseteq \mathrm{Var}(\varphi)$ for all $\ell \in L$ such that

$$\varphi \equiv \left(\bigwedge_{x \in I} x \right) \wedge \left(\bigwedge_{x \in J} \neg x \right) \wedge \left(\bigwedge_{\ell \in L} =(P_\ell, Q_\ell) \right).$$

W.l.o.g. let $I \cap J = \emptyset$. If $I \cap J \neq \emptyset$, we immediately see that there is no nonempty team satisfying φ. Now let $x \in I \cup J$ and T be a team with $T \models \varphi$. Then all assignments of T have to assign x to the same value. Therefore we can remove x from P_ℓ and Q_ℓ for all $\ell \in L$ without changing the semantics of the formula. Consequently we can split φ into two parts

$$\psi := \left(\bigwedge_{x \in I} x\right) \wedge \left(\bigwedge_{x \in J} \neg x\right) \quad \text{and} \quad \theta := \bigwedge_{\ell \in L} {=}(P_\ell, Q_\ell)$$

with $\mathrm{Var}(\psi) \cap \mathrm{Var}(\theta) = \emptyset$. Then satisfying teams for φ may be obtained by extending the assignments of the satisfying teams for θ. Variables in I are assigned to 1, whereas variables in J are assigned to 0. That being so, it is enough to consider formulas of the form

$$\varphi := \bigwedge_{\ell \in L} {=}(P_\ell, Q_\ell) \tag{3.1}$$

for the remainder of the chapter.

Example 3.2.4 We proceed with Example 3.2.3. The formula

$$x_4 \wedge {=}(x_1, x_3) \wedge {=}(\{x_2\}, \{x_3, x_4\})$$

may be reduced to

$${=}(x_1, x_3) \wedge {=}(x_2, x_3).$$

The team $\{000, 010\}$ satisfies the latter formula and yields the team $\{0001, 0101\}$, which satisfies the primary formula.

3.2.2 The group action of flipping bits

By the semantics of $=(\cdot)$ we see that flipping the bit at a fixed position in all assignments of a team T is an invariant for $T \models {=}(P, Q)$. For example, the team $\{00, 10\}$ satisfies ${=}(x_1, x_2)$. The remaining 2-team satisfying the formula is $\{01, 11\}$. Note that this team may be constructed from the previous one by flipping the value of x_2. Accordingly, it would be enough to compute the satisfying team $\{00, 10\}$ and construct the remaining 2-team by flipping bits. The concept of computing a minor set of satisfying k-Teams and constructing the remaining ones by flipping bits is the main concept of our algorithm for ensuring FPT-delay.

Notation 3.2.5 By identifying each assignment s with the vector $(s(x_1), \ldots, s(x_n))$, we obtain a bijection of sets

$$\mathbb{F}_2^n \leftrightarrow 2^{\mathrm{Var}(\varphi)}.$$

We will switch between interpreting an element as an assignment or a $\mathbb{F}_2$-vector as necessary, leading to expressions like $s+t$ for assignments s and t. Those may seem confusing at first, but become obvious when interpreting s and t as vectors. Vice versa we will consider $\mathbb{F}_2$-vectors as assignments that may be contained in a team. When both notations are to be used, this is indicated by taking $s \in \mathbb{F}_2^n \cong 2^{\mathrm{Var}(\varphi)}$ instead of simply writing $s \in \mathbb{F}_2^n$ or $s \in 2^{\mathrm{Var}(\varphi)}$.

Definition 3.2.6 (Group action of flipping bits) By Remark 2.4.2 the group action of $(\mathbb{F}_2^n, +)$ on itself induces a group action of $\mathbb{F}_2^n$ on $\mathcal{P}(\mathbb{F}_2^n)$. On that account we obtain a group action $\mathbb{F}_2^n \curvearrowright \mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right)$, called *group action of flipping bits*.

Remark 3.2.7 Let e_i be the vector of $\mathbb{F}_2^n$ with 1 as i-th component and 0 in the other components. Then the operation of e_i on $\mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right)$ corresponds to flipping the value for x_i in each assignment of a team.

Theorem 3.2.8 *Let $k \in \mathbb{N}$. The restriction of $\mathbb{F}_2^n \curvearrowright \mathcal{P}(\mathbb{F}_2^n)$ on $\mathcal{T}_k$ yields a group action $\mathbb{F}_2^n \curvearrowright \mathcal{T}_k$.*

Proof: As the axioms of group actions still hold on a subset of $\mathcal{P}(\mathbb{F}_2^n)$, it remains to show that

$$zT \in \mathcal{T}_k \quad \forall z \in \mathbb{F}_2^n,\ T \in \mathcal{T}_k.$$

Let $z \in \mathbb{F}_2^n$ and $T \in \mathcal{T}_k$. By Remark 2.4.2 we have $|zT| = k$. Let $P \subseteq \mathrm{Var}(\varphi)$ and $s, t \in 2^{\mathrm{Var}(\varphi)}$. If $s', t' \in 2^{\mathrm{Var}(\varphi)}$ arise from s, t by flipping the value for a variable x_i, then obviously

$$s\big|_P = t\big|_P \Leftrightarrow s'\big|_P = t'\big|_P.$$

It follows that

$$T \models =(P, Q) \Leftrightarrow zT \models =(P, Q) \quad \forall P, Q \subseteq \mathrm{Var}(\varphi).$$

When assuming that φ has the form of (3.1), $T \models \varphi$ clearly implies $zT \models \varphi$, proving $zT \in \mathcal{T}_k$. □

Lemma 3.2.9 *Let $T \in \mathcal{T}_k$, $k \in \mathbb{N}$. Then it holds that*

$$\mathbb{F}_2^n T \cap \mathcal{T}_k^0 \neq \emptyset.$$

For this reason, $\mathcal{T}_k^0$ contains a representative system for the orbits of $\mathbb{F}_2^n \curvearrowright \mathcal{T}_k$.

Proof: Take $s \in T \subseteq 2^{\mathrm{Var}(\varphi)} \cong \mathbb{F}_2^n$. Then $sT \in \mathcal{T}_k^0$ since $z + z = \vec{0}$ holds for all $z \in \mathbb{F}_2^n$. □

The previous lemma states that we can compute $\mathcal{T}_k$ from $\mathcal{T}_k^0$ by generating orbits. Next we want to present and analyse an algorithm for enumerating those orbits. The results are given in Theorem 3.2.13.

Definition 3.2.10 Let $\vec{0} \neq s = (s_1, \ldots, s_n) \in \mathbb{F}_2^n$ and $\mathcal{B} \subseteq \mathbb{F}_2^n \setminus \{\vec{0}\}$. Then we define

$$\begin{aligned} \operatorname{last}(s) &:= \max\{i \in \{1, \ldots, n\} : s_i = 1\}, \\ \operatorname{last}(\mathcal{B}) &:= \{\operatorname{last}(s) : s \in \mathcal{B}\}. \end{aligned}$$

Definition 3.2.11 Let $\mathcal{B}$ be a subset of $\mathbb{F}_2^n$. Then the subspace generated by $\mathcal{B}$ is defined by

$$\operatorname{span}(\mathcal{B}) := \{b_1 + \cdots + b_r : r \in \mathbb{N}_0,\ b_i \in \mathcal{B}\ \forall i \in \{1, \ldots, r\}\}.$$

Lemma 3.2.12 *Let U be a subspace of the $\mathbb{F}_2$-vector space $\mathbb{F}_2^n$. Let $\mathcal{B} \subseteq U \setminus \{\vec{0}\}$ be a maximal subset with*

$$b \neq b' \Rightarrow \operatorname{last}(b) \neq \operatorname{last}(b') \quad \forall b, b' \in \mathcal{B}. \tag{3.2}$$

Then $\mathcal{B}$ is a basis for U.

Proof: First we show that any set $A \subseteq U \setminus \{\vec{0}\}$ satisfying (3.2) is linearly independent. We conduct an induction over $|A|$. For $|A| = 1$, the claim is obvious. Because of (3.2) there exists an element $a_0 \in A$ with $\operatorname{last}(a_0) > \operatorname{last}(a)$ for all $a_0 \neq a \in A$. When considering the $\operatorname{last}(a_0)$-th component, clearly the equation

$$a_0 = \sum_{a_0 \neq a \in A} \lambda_a a, \quad \lambda_a \in \mathbb{F}_2$$

has no solution. As $A \setminus \{a_0\}$ is linearly independent by induction hypothesis, it follows that A is linearly independent.

Now assume that $\mathcal{B}$ does not generate U. We take an element $s \in U \setminus \operatorname{span}(\mathcal{B})$ with minimal $\operatorname{last}(s)$. As $\mathcal{B}$ is a maximal subset fulfilling (3.2), we have $\operatorname{last}(b) = \operatorname{last}(s)$ for a suitable element $b \in \mathcal{B}$. But then $s - b \in U \setminus \operatorname{span}(\mathcal{B})$ with $\operatorname{last}(s - b) < \operatorname{last}(s)$ contradicts the minimality of s. □

Theorem 3.2.13 *Let $T \in \mathcal{T}_k$, $k \in \mathbb{N}$. Then $\mathbb{F}_2^n T$ can be enumerated with delay $\mathcal{O}(k^3 n)$.*

Proof: W.l.o.g. let $T \in \mathcal{T}_k^0$. Note that T may have a nontrivial stabiliser subgroup so that duplicates occur when simply applying each $z \in \mathbb{F}_2^n$ to T. However, Proposition 2.4.6 states that we can enumerate the orbit of T without duplicates when applying a representative system for $\mathbb{F}_2^n/(\mathbb{F}_2^n)_T$.

When considering $\mathbb{F}_2^n$ as a vector space over $\mathbb{F}_2$, the subspaces of $\mathbb{F}_2^n$ correspond to the subgroups of $(\mathbb{F}_2^n, +)$. In view of this, any basis for a complement of the stabiliser subgroup $(\mathbb{F}_2^n)_T$ of T in $\mathbb{F}_2^n$ generates a representative system for $\mathbb{F}_2^n/(\mathbb{F}_2^n)_T$.

Algorithm 1: Enumerating orbits

Input: a team T with $\vec{0} \in T$
Output: the orbit $\mathbb{F}_2^n T$ of T where each outputted team is sorted

```
1  B_last ← ∅;                                          /* Assume that B_last is sorted */
2  for 0⃗ ≠ s ∈ T do                                     /* < k iterations */
3      if last(s) ∈ B_last then continue;               /* O(n) */
4      failed ← false;
5      for t ∈ T do                                     /* ≤ k iterations */
6          if s + t ∉ T then failed ← true;             /* O(kn) */
7      end
8      if not failed then B_last ← B_last ∪ {last(s)};  /* O(n) */
9  end
10 C_last ← {1,...,n} \ B_last;                         /* O(n) */
11 for s ∈ span({e_i : i ∈ C_last}) do
12     Compute sT;                                      /* O(kn) */
13     Sort sT;                                         /* O(kn log k) */
14     output sT;
15 end
```

Take a basis $\mathcal{B}$ of $(\mathbb{F}_2^n)_T$ as in Lemma 3.2.12. Set

$$\mathcal{C} := \{e_i : i \in \{1,\dots,n\} \setminus \mathrm{last}(\mathcal{B})\},$$

where e_i denotes the i-th standard vector of $\mathbb{F}_2^n$. By construction of $\mathcal{C}$, we can arrange the elements of $\mathcal{B} \cup \mathcal{C}$ so that the matrix containing these elements as columns has triangular shape with 1-entries on its diagonal. Consequently, $\mathcal{B} \cup \mathcal{C}$ is a basis for $\mathbb{F}_2^n$ and $\mathcal{C}$ is a basis for a complement of $(\mathbb{F}_2^n)_T$.

Now it remains to construct $\mathcal{B}$ as desired. For $s \in 2^{\mathrm{Var}(\varphi)} \cong \mathbb{F}_2^n$ we have

$$s \in (\mathbb{F}_2^n)_T \Rightarrow sT = T \Rightarrow s = s + \vec{0} \in T.$$

Hence we can compute $(\mathbb{F}_2^n)_T$ by checking $sT = T$ for $|T| = k$ elements. In fact it is enough to check $sT \subseteq T$ as we have $|sT| = |T|$. We obtain $\mathcal{B}$ by inserting each element of $(\mathbb{F}_2^n)_T \setminus \{\vec{0}\}$ preserving (3.2) into $\mathcal{B}$. This shows that Algorithm 1 outputs $\mathbb{F}_2^n T$ without duplicates. The delay is dominated by the precalculation delay, which is $\mathcal{O}(k^3 n)$. Note that we sort the k assignments of each team in ascending order before returning it. □

Example 3.2.14 Let $n = 3$ and $T = \{000, 100, 010, 110\}$. Note that T satisfies the reduced formula from Example 3.2.4 describing the robot soccer scenario. We

compute the orbit $\mathbb{F}_2^3 T$ of T by algorithm 1. We check $sT = T$ for all nonzero assignments s in T:

$$\begin{aligned} 100 \cdot T &= \{100, 000, 110, 010\} = \{000, 100, 010, 110\} = T, \\ 010 \cdot T &= \{010, 110, 000, 100\} = \{000, 100, 010, 110\} = T, \\ 110 \cdot T &= \{110, 010, 100, 000\} = \{000, 100, 010, 110\} = T. \end{aligned}$$

On that account we obtain

$$\begin{aligned} \mathcal{B}_{\text{last}} &= \{\text{last}(100), \text{last}(010), \text{last}(110)\} = \{1, 2\}, \\ \mathcal{C}_{\text{last}} &= \{3\}, \\ \text{span}(\{e_i : i \in \mathcal{C}_{\text{last}}\}) &= \{000, 001\}. \end{aligned}$$

Then the orbit of T is given by

$$\begin{aligned} 000 \cdot T &= \{000, 100, 010, 110\}, \\ 001 \cdot T &= \{001, 101, 011, 111\}. \end{aligned}$$

Finally we would like to relate t_k to t_k^0. The larger the quotient t_k/t_k^0, the more computation costs are saved by generating orbits instead of computing $\mathcal{T}_k$ immediately.

Theorem 3.2.15 *Let $k \in \mathbb{N}$ with $t_k \neq 0$. Then it holds that*

$$\frac{t_k}{t_k^0} = \frac{2^n}{k}.$$

Proof: Because of $t_k \neq 0$ and Lemma 3.2.9 it follows that $t_k^0 \neq 0$. For this reason we can choose $T \in \mathcal{T}_k^0$. We claim

$$|\mathbb{F}_2^n T \cap \mathcal{T}_k^0| = \frac{k}{|(\mathbb{F}_2^n)_T|}. \tag{3.3}$$

For any $s \in 2^{\text{Var}(\varphi)} \cong \mathbb{F}_2^n$ it holds that

$$sT \in \mathcal{T}_k^0 \Leftrightarrow \exists t \in T : s + t = \vec{0} \Leftrightarrow \exists t \in T : s = t \Leftrightarrow s \in T. \tag{3.4}$$

Consequently we have

$$\mathbb{F}_2^n T \cap \mathcal{T}_k^0 = \{sT : s \in T\} =: TT.$$

Let $r, s \in T$. Both elements yield the same team $rT = sT$ iff $s \in r(\mathbb{F}_2^n)_T$ so that for any fixed $r \in T$ we find exactly $|r(\mathbb{F}_2^n)_T| = |(\mathbb{F}_2^n)_T|$ ways of expressing rT in the form

of sT, where $s \in T$ by (3.4). When iterating over the k elements sT, $s \in T$, each team in TT is counted $|(\mathbb{F}_2^n)_T|$ times. It follows that

$$|TT| = \frac{k}{|(\mathbb{F}_2^n)_T|},$$

proving (3.3).

By Lemma 3.2.9 we find a representative system $R \subseteq \mathcal{T}_k^0$ for the orbits of $\mathbb{F}_2^n \cup \mathcal{T}_k$. With Equation (3.3) and the Orbit-Stabiliser theorem (see Proposition 2.4.6) we obtain

$$\begin{aligned}
t_k &= \sum_{T \in R} |\mathbb{F}_2^n T| && \text{(by Proposition 2.4.4)} \\
&= \sum_{T \in \mathcal{T}_k^0} \frac{|\mathbb{F}_2^n T|}{|\mathbb{F}_2^n T \cap \mathcal{T}_k^0|} \\
&= \sum_{T \in \mathcal{T}_k^0} \frac{|(\mathbb{F}_2^n)_T|}{k} \cdot |\mathbb{F}_2^n T| && \text{(by (3.3))} \\
&= \sum_{T \in \mathcal{T}_k^0} \frac{|(\mathbb{F}_2^n)_T|}{k} \cdot \frac{2^n}{|(\mathbb{F}_2^n)_T|} && \text{(by Proposition 2.4.6)} \\
&= \frac{2^n}{k} \sum_{T \in \mathcal{T}_k^0} 1 \\
&= \frac{2^n}{k} t_k^0.
\end{aligned}$$

□

Example 3.2.16 Consider the reduced formula

$$\varphi := =(x_1, x_3) \wedge =(x_2, x_3)$$

from Example 3.2.4. Then the orbits of $\mathcal{T}_k$, $k \in \mathbb{N}$, and their corresponding stabiliser subgroups are given in Figure 1. Teams located in $\mathcal{T}_k^0$ are coloured red. Note that the amount of red teams in each orbit of $\mathcal{T}_k$ matches k divided by the cardinality of the stabiliser subgroup. Furthermore we have

$$\frac{t_1}{t_1^0} = \frac{8}{1} = \frac{2^3}{1}, \quad \frac{t_2}{t_2^0} = \frac{16}{4} = \frac{2^3}{2}, \quad \frac{t_3}{t_3^0} = \frac{8}{3} = \frac{2^3}{3}, \quad \frac{t_4}{t_4^0} = \frac{2}{1} = \frac{2^3}{4}.$$

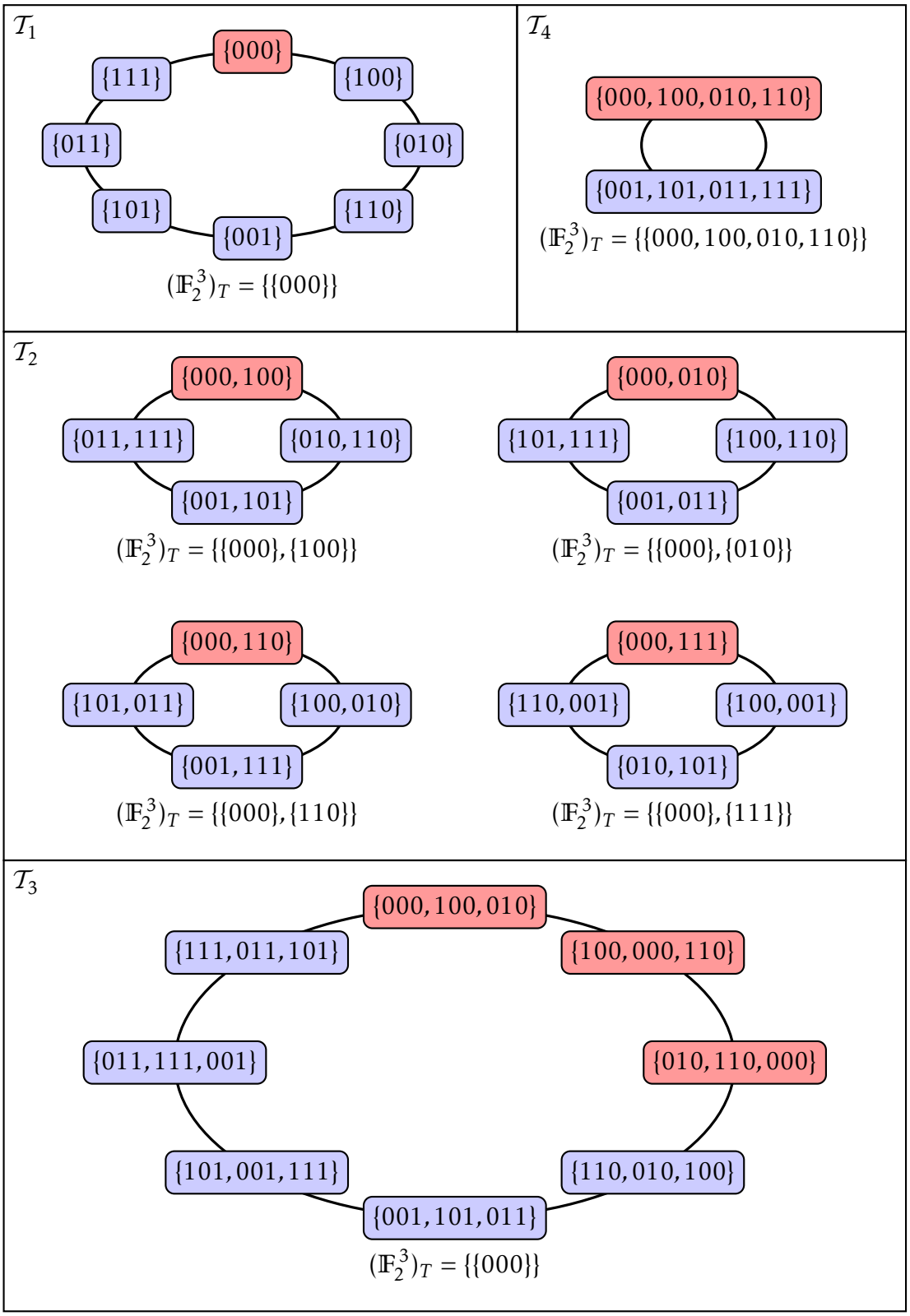

Figure 1: Orbits of $\mathcal{T}_k$ with $\varphi := =(x_1, x_3) \wedge =(x_2, x_3)$.

3.2.3 Constructing $\mathcal{T}_k^0$

Now that we are able to construct all satisfying k-teams from a representative system, the next step is the construction of $\mathcal{T}_k^0$. For this purpose, the concept of coherence presented by Kontinen will prove useful. The proofs are omitted here and may be found in [Kon13].

Definition 3.2.17 ([Kon13, Definition 3.1]) Let ϕ be a team based propositional formula. Then ϕ is *k-coherent* iff for all teams T it holds that

$$T \models \phi \;\Leftrightarrow\; R \models \phi \;\forall R \subseteq T \text{ with } |R| = k.$$

Proposition 3.2.18 ([Kon13, Proposition 3.3]) *The atom $=(\cdot)$ is 2-coherent.*

Proposition 3.2.19 ([Kon13, Proposition 3.4]) *Let ϕ and ψ be k-coherent formulas. Then $\phi \wedge \psi$ is k-coherent.*

Notation 3.2.20 Let $T = \{s_1, \dots, s_k\}$ be a team with $s_1 < \cdots < s_k$, $k \geq 2$. Then write

$$\begin{aligned} T^1_{\text{red}} &:= \{s_1, \dots, s_{k-1}\}, \\ T^2_{\text{red}} &:= \{s_1, \dots, s_{k-2}, s_k\}, \\ \max(T) &:= s_k. \end{aligned}$$

The following lemma provides a powerful tool for constructing the sets $\mathcal{T}_k^0$.

Lemma 3.2.21 *Let T be as above and $k := |T| \geq 3$. Then the following are equivalent:*

(i) $T \in \mathcal{T}_k^0$,

(ii) $T^1_{\text{red}}, T^2_{\text{red}} \in \mathcal{T}_{k-1}^0$ *and* $\{\vec{0}, s_{k-1} + s_k\} \in \mathcal{T}_2^0$.

Proof: After simplifying φ we may assume that φ is a conjunction of dependence atoms. In particular, φ is 2-coherent by Proposition 3.2.18 and 3.2.19.

(i) $\Rightarrow$ (ii): Let $T \in \mathcal{T}_k$. Any subset of cardinality 2 contained in T^1_{red} or T^2_{red} is a subset of T. The 2-coherence of φ yields $T^i_{\text{red}} \models \varphi$ for $i \in \{1, 2\}$. Furthermore $\vec{0} = s_1 \in T^i_{\text{red}}$ and $|T^i_{\text{red}}| = k - 1$ holds. This gives us $T^1_{\text{red}}, T^2_{\text{red}} \in \mathcal{T}^0_{k-1}$. Again by the 2-coherence of φ, we obtain that $\{s_{k-1}, s_k\} \in \mathcal{T}_2$. Applying the group action $\mathbb{F}_2^n \curvearrowright \mathcal{T}_2$ shows that $\{\vec{0}, s_{k-1} + s_k\} \in \mathcal{T}_2^0$.

(ii) $\Rightarrow$ (i): First note that $\vec{0} \in T^1_{\text{red}} \subset T$ and $|T| = |T^1_{\text{red}}| + 1 = k$. Assume $T \not\models \varphi$. Then by 2-coherence T has a subset R with cardinality 2 and $R \not\models \varphi$. In particular, it holds that $R \not\subseteq T^1_{\text{red}}, T^2_{\text{red}}$, implying $R = \{s_{k-1}, s_k\}$. This contradicts $s_{k-1}R = \{\vec{0}, s_{k-1} + s_k\} \in \mathcal{T}_2^0$. □

Algorithm 2: Constructing $\mathcal{T}_k^0$

Input: $k \in \mathbb{N}$, $k \geq 2$
Dependencies: if $k > 2$: $\mathcal{D}_2[\{\vec{0}\}]$, $\mathcal{D}_{k-1}$ of the previous iteration
Result: $\mathcal{T}_k^0$

```
1  T_k^0 ← ∅;
2  D_k ← new Map(Team, List(Assignment));
3  if k = 2 then
4      D_2[{0⃗}] ← ∅;
5      for 0⃗ ≠ s ∈ 2^Var(φ) do                          /* ≤ 2^n iterations */
6          if {0⃗, s} ⊨ φ then                            /* O(|φ|) */
7              D_2[{0⃗}] ← D_2[{0⃗}] ∪ {s};               /* O(n^2) */
8              T_2^0 ← T_2^0 ∪ {0⃗, s};                   /* O(n^2) */
9          end
10     end
11 else
12     for (T, L) ∈ D_{k−1} do
13         for r ∈ L do                                  /* t^0_{k−1} iterations */
14             T' ← T ∪ {r};
15             D_k[T'] ← ∅;
16             for s ∈ L with s > r do                   /* ≤ 2^n iterations */
17                 if r + s ∈ D_2[{0⃗}] then               /* O(n^2) */
18                     D_k[T'] ← D_k[T'] ∪ {s};           /* O(k^2 n^2) */
19                     T_k^0 ← T_k^0 ∪ {T' ∪ {s}};        /* O(k^2 n^2) */
20                 end
21             end
22         end
23     end
24 end
```

Algorithm 2 computes the sets $\mathcal{T}_k^0$ by exploiting the previous lemma. In order to ensure fast list operations, we sort teams by the lexicographical order as given in Definition 3.0.5. When the assignments of each team are saved in ascending order—which is easy to guarantee—the cost of comparing two k-teams is $\mathcal{O}(kn)$. We do not store duplicates, restricting the size of lists containing teams to

$$\left|\mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right)\right| = \binom{2^n}{k} \leq (2^n)^k = 2^{kn}. \tag{3.5}$$

By managing those lists in AVL Trees presented in [AVL62], the standard list operations as searching, insertion and deletion are realised in

$$\mathcal{O}\left(\log(2^{kn}) \cdot kn\right) = \mathcal{O}\left(k^2 n^2\right). \tag{3.6}$$

With these considerations in mind, we begin proving the correctness and performance of the algorithm.

Lemma 3.2.22 *Let $k \geq 2$. For $T \in \mathcal{T}_k^0$ it holds that $\max(T) \in \mathcal{D}_k[T^1_{\mathrm{red}}]$. Vice versa, if $s \in \mathcal{D}_k[T]$, then it follows that $T \cup \{s\} \in \mathcal{T}_k^0$ and $s > \max(T)$.*

Proof: We conduct an induction over k.

Induction basis ($k = 2$): Let $T \in \mathcal{T}_2^0$. It follows that $T^1_{\mathrm{red}} = \{\vec{0}\}$. As we have $\{\vec{0}, \max(T)\} \models \varphi$, in line 7 $\max(T)$ is inserted into $\mathcal{D}_2[T^1_{\mathrm{red}}]$. Now let $s \in 2^{\mathrm{Var}(\varphi)}$ and $T \in \mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right)$ such that $s \in \mathcal{D}_2[T]$. The only team occurring in $\mathcal{D}_2$ is $T = \{\vec{0}\}$. We have $s \in \mathcal{D}_2[T]$ iff $T \cup \{s\} = \{\vec{0}, s\} \models \varphi$ and $s \neq \vec{0}$. The claim follows.

Induction step ($k - 1 \to k$): Let $T = \{s_1, \ldots, s_k\} \in \mathcal{T}_k^0$, $s_1 < \cdots < s_k$. By induction hypothesis and Lemma 3.2.21 it holds that $s_{k-1} \in \mathcal{D}_{k-1}[T^1_{\mathrm{red}} \setminus \{s_{k-1}\}]$. Accordingly, the loop body of line 13 is invoked with $T \leftarrow T^1_{\mathrm{red}} \setminus \{s_{k-1}\}$, $r \leftarrow s_{k-1}$, $T' \leftarrow T^1_{\mathrm{red}}$. Furthermore by Lemma 3.2.21 the loop body of line 16 is invoked with $s \leftarrow s_k$, passing the check in line 17. As a result, $s_k = \max(T)$ is inserted into $\mathcal{D}_k[T^1_{\mathrm{red}}]$.

Now let $s \in 2^{\mathrm{Var}(\varphi)}$ and $T \in \mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right)$ such that $s \in \mathcal{D}_k[T]$. Then by the construction of $\mathcal{D}_k$ there exist a team T' and $r \in \mathcal{D}_{k-1}[T']$ with $r < s$, $s \in \mathcal{D}_{k-1}[T']$, $T' \cup \{r\} = T$ and $\{\vec{0}, r + s\} \in \mathcal{T}_2^0$. The induction hypothesis yields $T' \cup \{r\}, T' \cup \{s\} \in \mathcal{T}_{k-1}^0$ and $r > \max(T')$, implying $s > r = \max(T)$. As s and r are the largest elements of $T \cup \{s\}$, it follows that $(T \cup \{s\})^1_{\mathrm{red}} = T' \cup \{r\}$ and $(T \cup \{s\})^2_{\mathrm{red}} = T' \cup \{s\}$. By Lemma 3.2.21 we obtain $T \cup \{s\} \in \mathcal{T}_k^0$. □

Corollary 3.2.23 *Algorithm 2 constructs the sets $\mathcal{T}_k^0$ correctly.*

Proof: Every team T inserted into $\mathcal{T}_k^0$ by Algorithm 2 has the form $T' \cup \{s\}$ with $s \in \mathcal{D}_k[T']$. Then by Lemma 3.2.22 it follows that $s > \max(T)$ and $T \in \mathcal{T}_k^0$. Note

that the decomposition of T is unique because of $s > \max(T)$. On that account T is inserted only once.

Now let $T \in \mathcal{T}_k^0$. Lemma 3.2.22 states that $\max(T) \in \mathcal{D}_k[T_{\text{red}}^1]$. After inserting $\max(T)$ into $\mathcal{D}_k[T_{\text{red}}^1]$, Algorithm 2 inserts $T_{\text{red}}^1 \cup \max(T) = T$ into $\mathcal{T}_k^0$. □

Corollary 3.2.24 *Algorithm 2 requires $t_{k-1}^0 \cdot 2^n \cdot \mathcal{O}(k^2|\varphi|^2)$ time on input $k \in \mathbb{N}$.*

Proof: Note

$$\{\vec{0}, s\} \models =(P, Q) \;\Leftrightarrow\; s \models \left(\bigvee_{x \in P} x\right) \vee \left(\bigwedge_{y \in Q} \neg y\right) \quad \forall \vec{0} \neq s \in 2^{\mathrm{Var}(\varphi)}.$$

As a consequence, checking $\{\vec{0}, s\} \models \varphi$ can be accomplished in linear time by evaluating a $\mathcal{PL}$-formula of length $\mathcal{O}(|\varphi|)$. Accessing the list $\mathcal{D}_k[T]$ for a team T is in $\mathcal{O}(k^2 n^2)$ by (3.6). When managing the list in an AVL Tree, the list operations are realised in $\mathcal{O}(\log 2^n) = \mathcal{O}(n)$, which is contained in $\mathcal{O}(k^2 n^2)$.

As the decomposition for $T \in \mathcal{T}_{k-1}^0$ into T' and s with $s > \max(T')$ is unique, applying Lemma 3.2.22 yields that the loop body of line 13 is invoked t_{k-1}^0 times.

Taking into account that $n \leq |\varphi|$, we obtain the claim by adding up all costs. □

Example 3.2.25 We construct the sets $\mathcal{T}_k^0$ for the reduced formula

$$\varphi := =(x_1, x_3) \wedge =(x_2, x_3)$$

from Example 3.2.4. Trivially, $\mathcal{T}_1^0 = \{\{000\}\}$ holds. When computing $\mathcal{T}_2^0$, we have to identify all nonzero assignments that satisfy

$$(x_1 \vee \neg x_3) \wedge (x_2 \vee \neg x_3).$$

Obviously, the satisfying assignments are 100, 010, 110 and 111. We obtain

$$\mathcal{D}_2[\{000\}] = \{100, 010, 110, 111\}.$$

Figure 2 illustrates the construction of the remaining lists and the resulting sets $\mathcal{T}_k^0$. We are able to verify that the orbits presented in Example 3.2.16 are exactly those of $\mathcal{T}_k$, $k \in \mathbb{N}$. Each orbit contains at least one element of $\mathcal{T}_k^0$ and every team in $\mathcal{T}_k^0$ can be recovered in one orbit of Figure 1.

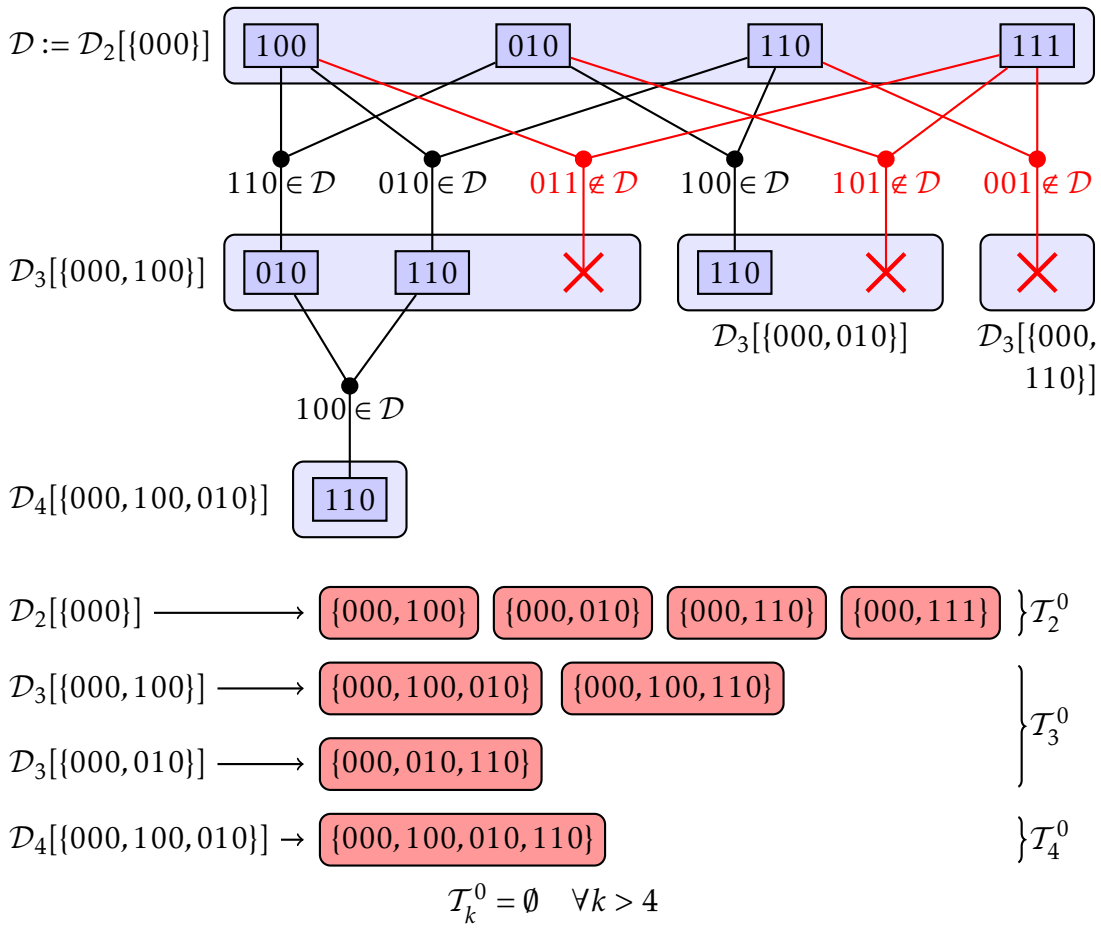

Figure 2: Construction of $\mathcal{T}_k^0$ with $\varphi := =(x_1, x_3) \wedge =(x_2, x_3)$.

Algorithm 3: Enumerating satisfying teams in $\mathcal{PDL}\backslash\{\vee\}$ ordered by cardinality

Input: a team based propositional formula φ as in Equation (3.1), $k \in \mathbb{N}$
Output: all teams T for φ with $T \models \varphi$, $1 \leq |T| \leq k$

```
1  T_1^0 ← {{0⃗}};
2  for ℓ = 2,...,k+1 do
3      simultaneously do
4          while T_{ℓ-1}^0 ≠ ∅ do
5              Choose T ∈ T_{ℓ-1}^0;
6              for T' ∈ F_2^n T (Algorithm 1) do
7                  output T';
8                  T_{ℓ-1}^0 ← T_{ℓ-1}^0 \ {T'};
9              end
10         end
11     and do
12         Compute T_ℓ^0 by Algorithm 2;
13     end
14     if T_ℓ^0 = ∅ then break;
15 end
```

3.2.4 The algorithm

Although by Corollary 3.2.24 Algorithm 2 does not perform in polynomial time on input $k \in \mathbb{N}$, we can ensure polynomial delay when distributing its execution over the process of outputting all satisfying teams of cardinality $k-1$. For this reason we investigate the costs of computing $\mathcal{T}_k^0$ divided by t_{k-1}. With Corollary 3.2.24 and

$$k-1 = \frac{t_{k-1}^0 \cdot 2^n}{t_{k-1}},$$

which is a transformation of the equation in Theorem 3.2.15, we obtain

$$\begin{aligned}\frac{\text{computation\_costs}(\mathcal{T}_k^0)}{t_{k-1}} &= \frac{t_{k-1}^0 \cdot 2^n \cdot \mathcal{O}(k^2|\varphi|^2)}{t_{k-1}} \\ &= (k-1)\cdot\mathcal{O}(k^2|\varphi|^2) \\ &= \mathcal{O}(k^3|\varphi|^2).\end{aligned}$$

Consequently, the delay of Algorithm 3 is bounded by $\mathcal{O}(k^3|\varphi|^2)$. Note that the delay of generating the orbits $\mathbb{F}_2^n T$, which is $\mathcal{O}(k^3 n)$ by Theorem 3.2.13, and the cost of removing elements in $\mathcal{T}_k^0$, which is $\mathcal{O}(k^2 n^2)$, are contained in $\mathcal{O}(k^3|\varphi|^2)$.

Theorem 3.2.26 *Algorithm* 3 *enumerates all satisfying teams T for φ with $1 \leq |T| \leq k$ without duplicates.*

Proof: It is easy to see that all dependencies in Algorithm 2 and 3 are resolved in time. Due to Proposition 2.4.4 and Lemma 3.2.9, every satisfying team is outputted at least once. By removing every outputted element in line 8, no orbit is outputted twice, preventing duplicates. □

We conclude:

Theorem 3.2.27 *The following holds:*

(i) P-EnumTeamPlSize($\mathcal{PDL}\backslash\{\vee\}$) $\in$ ***DelayFPT***,

(ii) EnumTeamPlSize($\mathcal{PDL}\backslash\{\vee\}, f$) $\in$ ***DelayP*** *for any polynomial time computable function $f \in n^{\mathcal{O}(1)}$.*

3.2.5 Consequences of sorting by cardinality

In the previous section we have seen that the restriction on polynomial teams is sufficient to obtain a **DelayP**-algorithm for $\mathcal{PDL}\backslash\{\vee\}$. As we will see in this section, the restriction is not only sufficient but also necessary when the output is sorted by its cardinality. Consequently, the algorithm presented above is optimal regarding output size.

Lemma 3.2.28 *Let $k \geq 2$ and*

$$\varphi(x_1, \ldots, x_k) := \bigwedge_{i=1}^{k-1} {=}(x_i, x_k) \in \mathcal{PDL}\backslash\{\vee\}.$$

Then for any team $T \neq \emptyset$ with $T \models \varphi$ and $|T| \geq 3$ it holds that $\left|T\big|_{\{x_k\}}\right| = 1$.

Proof: Let T be a team with $T \models \varphi$ and $|T| \geq 3$. Set

$$T\big|_{x_k=i} := \{s \in T : s(x_k) = i\} \quad \forall i \in \{0, 1\}.$$

Assume that $\left|T\big|_{\{x_k\}}\right| = 1$ does not hold. Then w.l.o.g. $T\big|_{x_k=0} \neq \emptyset$ and $\left|T\big|_{x_k=1}\right| > 1$. Take $r \in T\big|_{x_k=0}$. For any $s \in T\big|_{x_k=1}$ it holds that $\{r, s\} \models \varphi$ because φ is downward closed. In particular, $\{r, s\} \models {=}(x_i, x_k)$ for all $i \in \{1, \ldots, k-1\}$, yielding $r(x_i) \neq s(x_i)$ because of $r(x_k) \neq s(x_k)$. On that account s is uniquely determined by r, contradicting $\left|T\big|_{x_k=1}\right| > 1$. □

Theorem 3.2.29 *Let f be a polynomial time computable function. Then we have*

$$\textsc{EnumTeamPlSize}(\mathcal{PDL}\backslash\{\vee\}, f) \in \textbf{\textit{DelayP}} \Leftrightarrow f \in n^{\mathcal{O}(1)}.$$

Proof: $\Leftarrow$: immediately follows from Theorem 3.2.27
$\Rightarrow$: Let $f \notin n^{\mathcal{O}(1)}$. Assume that $\textsc{EnumTeamPlSize}(\mathcal{PDL}\backslash\{\vee\}, f) \in$ **DelayP** holds via an algorithm with a delay bounded by n^c, $c \in \mathbb{N}$. Then there exists $k \in \mathbb{N}$ such that

$$z := \min\{f(k), 2^{k-1}\} > 4^c \cdot k^c \geq k \geq 3.$$

Let φ be as in Lemma 3.2.28. Obviously, there exist teams $T_0, T_1 \in \mathcal{T}_z$ with $s(x_k) = i$ for all $s \in T_i$, $i \in \{0,1\}$. Since the elements in $\mathcal{T}_z$ have to be outputted in succession and $\left|T\big|_{\{x_k\}}\right| = 1$ for any $T \in \mathcal{T}_z$, we can choose T_0 and T_1 such that both teams are outputted in consecutive order. However, both teams differ in at least z bits describing the evaluation at x_k. For this reason the delay is at least

$$z > (4k)^c \geq (|\varphi|)^c,$$

contradicting that the delay is bounded by n^c. □

Corollary 3.2.30 $\textsc{EnumTeamPlSize}(\mathcal{PDL}\backslash\{\vee\}) \notin$ ***DelayP***.

Proof: Since $(n \mapsto 2^n) \notin n^{\mathcal{O}(1)}$, the claim follows immediately from Theorem 3.2.29. □

The trick of examining the symmetric difference of consecutive teams gives rise to the previous theorem. Unfortunately this trick cannot be applied to arbitrary orders and certainly fails for the lexicographical order. In order to prove this claim, consider Theorem 3.2.31 with $S = 2^{\mathrm{Var}(\varphi)}$, $X = \left\{T \in \mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right) : T \models \varphi\right\}$.

Theorem 3.2.31 *Let $S = \{s_1, \ldots, s_n\}$ be a total ordered, finite set and $X \subseteq \mathcal{P}(S)$ be a downward closed set, meaning*

$$T \in X \Rightarrow R \in X \; \forall R \subseteq T.$$

When X is ordered lexicographically in respect to the order on S, the symmetric difference $\triangle$ between two consecutive elements in X is at most 3.

Proof: W.l.o.g, we assume that $\{s_i\} \in X$ for all $i \in \{1, \ldots, n\}$. We prove the claim by induction over n. The induction basis for $n = 1$ is obvious. Consider the induction step from $n-1$ to n.

Set

$$X_i := \{T \in X : s_1, \dots, s_{i-1} \notin T,\ s_i \in T\},$$
$$\overline{X_i} := \{T \setminus \{s_i\} : T \in X_i\}$$

for $i \in \{1, \dots, n\}$. The subsets X_i together with $\{\emptyset\}$ form a partition of X. Furthermore it is easy to see that the sets $\overline{X_i}$ are downward closed and that $T < T'$ for all $T \in X_i$, $T' \in X_j$ with $i < j$.

By induction hypothesis the symmetric difference for consecutive elements in $\overline{X_i}$ is at most 3. As the order on X_i corresponds to the order on $\overline{X_i}$, the same applies to X_i. It remains to check the consecutive elements located in different X_i's. For any element $T = \{s_{i_1}, \dots, s_{i_k}\} \in X_i$ with $s_{i_1} < \dots < s_{i_k}$ we have $T \geq \{s_{i_1}\} = \{s_i\} \in X_i$ and $T \leq \{s_{i_1}, s_{i_k}\} \in X_i$. Consequently:

$$\{s_i\} = \min_{T \in X_i} T, \quad \left|\max_{T \in X_i} T\right| \leq 2 \quad \forall i \in \{1, \dots, n\}.$$

In fact, it holds that

$$\left|\emptyset \mathbin{\triangle} \min_{T \in X_1} T\right| \leq 0 + 1 < 3,$$
$$\left|\max_{T \in X_i} T \mathbin{\triangle} \min_{T \in X_{i+1}} T\right| \leq 2 + 1 = 3 \quad \forall i \in \{1, \dots, n-1\}.$$

□

3.2.6 Limiting memory space

Next we examine the memory usage of Algorithm 3. Throughout the execution, $\mathcal{D}_2[\{\vec{0}\}]$, $\mathcal{D}_k$ and $\mathcal{T}_k^0$ have to be saved. However the size of those lists increases exponentially when raising the size k of the outputted teams or the amount of variables occurring in the formula φ. By Equation (3.5) Algorithm 3 requires space $\mathcal{O}(2^{2^n})$, respectively, $\mathcal{O}(2^n)$ when fixing the parameter k. In fact, any algorithm that saves a representative system for the orbits of $\mathbb{F}_2^n \circlearrowright \mathcal{T}_k$ cannot perform in polynomial space by the following theorem. For this reason we have to discard the group action of flipping bits when limiting memory space to polynomial sizes.

Theorem 3.2.32 *Let $1 \neq k \in \mathbb{N}$ and $n \in \mathbb{N}$. We set $\varphi := =(x_1, x_2, \dots, x_n)$. Then the amount of orbits of $\mathbb{F}_2^n \circlearrowright \mathcal{T}_k$ is not polynomial in n.*

Proof: Note that each orbit of $\mathbb{F}_2^{n-1}$ on the set of all k-teams over $n-1$ variables maps to an orbit of $\mathbb{F}_2^n \circlearrowright \mathcal{T}_k$ by extending all assignments of a team so that x_n is assigned to the same value. As we have

$$f(n) \in n^{\mathcal{O}(1)} \Leftrightarrow f(n-1) \in n^{\mathcal{O}(1)}$$

for any function $f : \mathbb{N} \to \mathbb{N}$, we may assume that φ is equivalent to 1 with $|\mathrm{Var}(\varphi)| = n$.

By the Cauchy-Frobenius lemma (see Proposition 2.4.7) the amount of orbits is at least

$$\frac{|\{T \in \mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right) : |T| = k\}|}{2^n}$$

when neglecting all summands except the one for $\vec{0} \in \mathbb{F}_2^n$. That is why the number of orbits in $\mathcal{T}_k$ has to be larger than $\binom{2^n}{k}/2^n$, which already increases exponentially in n.

□

In the previous sections we had to limit the cardinality of outputted teams for obtaining polynomial delay. As the following theorem shows, this measure is necessary as well when demanding polynomial space.

Theorem 3.2.33 *Let ϕ be any fragment of team based propositional logic and f be a function with $f \notin n^{\mathcal{O}(1)}$ such that for any $n \in \mathbb{N}$ there exists a formula $\varphi_n \in \phi$ in n variables with at least $2^{f(n)}$ satisfying teams. Then it follows that* EnumTeamPl(ϕ) *cannot be enumerated in polynomial space.*

Proof: Any enumeration algorithm enumerating $\mathrm{Sol}(\varphi_n)$ has to output $2^{f(n)}$ different teams. The same amount of configurations have to be adopted. In order to distinguish these, the configurations are encoded by at least $f(n)$ bits. However, when considering a RAM performing in polynomial space, the contents of all registers may be encoded by a polynomial amount of bits. For this reason a RAM enumerating $\mathrm{Sol}(\varphi_n)$ cannot perform in polynomial space. □

Corollary 3.2.34 *The problem* EnumTeamPl($\mathcal{PDL}\backslash\{\vee\}$) *cannot be enumerated in polynomial space.*

Proof: Let $n \in \mathbb{N}$. Set $\varphi_n := =(x_1, x_2, \ldots, x_n)$. All teams T with $s(x_n) = 0$ for all assignments $s \in T$ satisfy φ_n. For this reason at least $2^{2^{n-1}}$ satisfying teams exist. Because of $2^{n-1} \notin n^{\mathcal{O}(1)}$, the claim follows by the previous theorem. □

We now present an algorithm enumerating EnumTeamPlSize($\mathcal{PDL}\backslash\{\vee\}, f$) for any $f \in n^{\mathcal{O}(1)}$ in polynomial space. Compared to Algorithm 3, it saves memory space by recomputing the satisfying teams of lower cardinality instead of storing them in a list. As a downside we have to accept incremental delays.

Notation 3.2.35 We define a unary relation hasNext on $2^{\mathrm{Var}(\varphi)}$ by

$$s \in \mathtt{hasNext} :\Leftrightarrow \exists t \in 2^{\mathrm{Var}(\varphi)} : s < t.$$

For any $s \in \mathtt{hasNext}$ let $\mathtt{next}(s)$ be the unambiguous assignment greater than s such that $s < t < \mathtt{next}(s)$ does not hold for any assignment t. We denote the smallest element in $2^{\mathrm{Var}(\varphi)}$ by s_{first}. The largest element is denoted by s_{last}.

As already mentioned when defining the lexicographical order, we assume that hasNext, next and s_{first} may be determined in $\mathcal{O}(n)$ time.

Lemma 3.2.36 *Let T be a team with cardinality k. Then $T \models \varphi$ can be checked in $\mathcal{O}(k^2|\varphi|)$ time.*

Proof: Because of the 2-coherence of φ it is enough to check all 2-subteams of T. By the proof of Corollary 3.2.24 checking a 2-team is accomplished in $\mathcal{O}(|\varphi|)$ time. As T has $\mathcal{O}(k^2)$ 2-subteams, the claim follows. □

Notation 3.2.37 Let $k \in \mathbb{N}$. We write $\mathcal{M}_k$ for the set of teams T is assigned to during the k-th iteration of the outer loop of Algorithm 4.

Lemma 3.2.38 *Let $S \in \mathcal{M}_k$ be a nonempty set such that $s := \max(S) \in \mathtt{hasNext}$. Then it follows that*

$$S \setminus \{s\} \cup \{\mathtt{next}(s)\} \in \mathcal{M}_k.$$

In particular, we have

$$S \setminus \{s\} \cup \{t\} \in \mathcal{M}_k \quad \forall t \in 2^{Var(\varphi)} \text{ with } t \geq s.$$

Proof: We conduct an induction over $k - |S|$.

Induction basis ($|S| = k$): Because of $|S| \not< k$ and $s \in \mathtt{hasNext}$ line 9 is executed and T is assigned to $S \setminus \{s\} \cup \{\mathtt{next}(s)\}$.

Induction step ($|S| + 1 \to |S|$, $1 \leq |S| < k$): If $S \not\models \varphi$, line 9 is executed as before and the claim follows. If $S \models \varphi$, line 7 is executed. It follows that $S \cup \{\mathtt{next}(s)\} \in \mathcal{M}_k$. By induction hypothesis it follows that $S \cup \{s_{\mathrm{last}}\} \in \mathcal{M}_k$. When executing the body of the while loop with T assigned to $S \cup \{s_{\mathrm{last}}\}$, the block beginning at line 11 is executed, assigning T to $S \setminus \{s\} \cup \{\mathtt{next}(s)\}$. □

Lemma 3.2.39 *Let $k \in \mathbb{N}$ and S be a team with $S \models \varphi$ and $|S| \leq k$. Then it follows that $S \in \mathcal{M}_k$.*

Algorithm 4: Enumerating satisfying teams in polynomial space ordered by cardinality

Input: a team based propositional formula φ as in Equation (3.1)
Output: all teams T for φ with $T \models \varphi$, $1 \leq |T| \leq f(|\varphi|)$

```
1  for k = 1,...,f(|φ|) do
2      T ← {s_first};
3      while true do
4          if |T| = k and T ⊨ φ then output T;
5          s ← max(T);
6          if |T| < k and T ⊨ φ and s ∈ hasNext then
7              T ← T ∪ {next(s)};
8          else if s ∈ hasNext then
9              T ← T \ {s} ∪ {next(s)};
10         else if |T| > 1 then
11             T ← T \ {s};
12             s ← max(T);
13             T ← T \ {s} ∪ {next(s)};
14         else
15             break;
16         end
17     end
18 end
```

Proof: We conduct an induction over $|S|$.

Induction basis ($|S| = 1$): Clearly $\{s_{\text{first}}\} \in \mathcal{M}_k$. By Lemma 3.2.38 every 1-team is contained in $\mathcal{M}_k$.

Induction step ($|S| - 1 \to |S|$, $1 < |S| \leq k$): Let $s = \max(S)$. Since φ is downward closed, it follows that $S \setminus \{s\} \models \varphi$. The induction hypothesis yields $S \setminus \{s\} \in \mathcal{M}_k$. Consequently the while loop is executed with T assigned to $S \setminus \{s\}$. Line 9 is executed, assigning T to a team $S \setminus \{s\} \cup \{t\}$, where t is an appropriate assignment with $t \leq s$. Lemma 3.2.38 yields $S \in \mathcal{M}_k$. □

Lemma 3.2.40 *Let $f \in n^{\mathcal{O}(1)}$ be a polynomial time computable function. Then there exists a polynomial p such that the i-th delay of Algorithm 4 is bounded by $i^2 p(|\varphi|)$.*

Proof: Note that the delay is constant when outputting the 2^n singletons that satisfy φ trivially. Hence we assume that $i \geq 2^n$. It is easy to verify that any team T is assigned to is lexicographically larger than the previous value for T. For this

reason the number of iterations of the inner while loop is bounded by $|\mathcal{M}_k|$.

As T is not assigned to teams with greater cardinality when the current value for T does not satisfy φ, it follows that $S \setminus \{\max(S)\} \models \varphi$ for any $S \in \mathcal{M}_k$ with $|S| > 1$. Consequently

$$|\mathcal{M}_k| \leq 2^n \sum_{l=0}^{k-1} t_l \leq i \sum_{l=0}^{k-1} t_l.$$

Let S be the $(i+1)$-th outputted element. Set $k = |S|$. We have $S \in \mathcal{M}_k$ and $|S| > 1$. By outputting teams of lower cardinality first we guarantee that $i \geq \sum_{l=1}^{k-1} t_l$. Furthermore S is outputted in the k-th iteration of the outer loop. Consequently the inner while loop has been executed at most $k \cdot i^2$ times before outputting S. Since k is bounded by a polynomial in $|\varphi|$, by Lemma 3.2.36 it follows that the body of the inner while loop can be executed in polynomial time. We conclude that the i-th delay is bounded by $i^2 p(|\varphi|)$, where p is an appropriate polynomial.

Now let i be the total amount of outputted teams. Then the number of iterations of the inner while loop is bounded by

$$\begin{aligned}\sum_{k=1}^{f(|\varphi|)} |\mathcal{M}_k| &\leq f(|\varphi|) \cdot |\mathcal{M}_{f(|\varphi|)}| \\ &\leq f(|\varphi|) \cdot 2^n \sum_{l=0}^{f(|\varphi|)} t_l \\ &\leq f(|\varphi|) \cdot i^2.\end{aligned}$$

Accordingly, we can choose p such that even the postcalculation delay is bounded by $i^2 p(|\varphi|)$. □

Theorem 3.2.41 *Let $f \in n^{\mathcal{O}(1)}$ be a polynomial time computable function. Then Algorithm 4 is an **IncP**-algorithm for* EnumTeamPlSize($\mathcal{PDL}\setminus\{\vee\}, f$) *which performs in polynomial space.*

Proof: The algorithm saves only one team of cardinality $\leq f(|\varphi|)$ and one assignment for which $(f(|\varphi|)+1)$ registers are required. By Lemma 3.2.39 it is clear that the algorithm outputs the satisfying teams ordered by cardinality. Lemma 3.2.40 states that the delays conform to the definition of **IncP**. □

3.3 Enumeration for DNF-formulas

In this section we modify the enumeration algorithm for $\mathcal{PDL}\backslash\{\vee\}$ such that satisfying teams for DNF-formulas in $\mathcal{PDL}(\mathbin{⩒})\backslash\{\vee\}$ can be enumerated. As in classical propositional logic, formulas in disjunctive normal form are disjunctions of conjunctions combining atoms. Formally:

Definition 3.3.1 We define the fragment

$$\mathcal{CPDL} := \mathcal{PDL}(\mathbin{⩒})\backslash\{\vee\},$$

which contains the $\mathcal{PDL}$-formulas, where the disjunction operator $\vee$ is replaced by the classical disjunction operator $\mathbin{⩒}$. The fragment DNF-$\mathcal{CPDL}$ of DNF-formulas in $\mathcal{CPDL}$ is defined by

$$\text{DNF-}\mathcal{CPDL} := \{\varphi_1 \mathbin{⩒} \varphi_2 \mathbin{⩒} \cdots \mathbin{⩒} \varphi_r : r \in \mathbb{N},\ \varphi_i \in \mathcal{PDL}\backslash\{\vee\} \text{ for all } i \in \{1,\dots,r\}\}.$$

The naive approach of outputting the satisfying teams of φ_1 first, then the remaining ones of φ_2 second and so forth is flawed as we cannot guarantee polynomial delay. Consider $\varphi_1 \mathbin{⩒} \varphi_2$ such that $\mathrm{Sol}(\varphi_1) = \mathrm{Sol}(\varphi_2)$. Then we have to execute the enumeration algorithm for φ_2 in case there exist teams satisfying φ_2 and falsifying φ_1. However, we will not find any such team so that the postcalculation delay is dominated by the costs of enumerating the whole set $\mathrm{Sol}(\varphi_2)$.

Our approach bases on merging the outputs $\mathrm{Sol}(\varphi_i)$, $i \in \{1,\dots,r\}$, which is possible in polynomial delay if the output for each φ_i is presorted according to a total order. To see this, consider the set of the last computed elements in $\mathrm{Sol}(\varphi_i)$ for all $i \in \{1,\dots,r\}$ and output the minimal element of this set. Then advance the enumeration of those $\mathrm{Sol}(\varphi_i)$ for which the last computed element corresponds to the outputted one. Since the merged output is sorted as well, we may use it as input for another merge. In fact the presorted sets $\mathrm{Sol}(\varphi_i)$ are constructed by merging their orbits with respect to the group action of flipping bits. Hence, it remains to presort the orbits.

So far we did not specify the total order used for merging. Since we wish to output all teams ordered by their cardinality, we extend $\leq_{\mathrm{size}}$ appropriately such that orbits may be efficiently enumerated regarding the extended order. As we will see in the upcoming section, this is possible when extending $\leq_{\mathrm{size}}$ by the lexicographical order on teams induced by the lexicographical order on assignments. Remember that the lexicographical order on assignments is given by the natural order on $\mathbb{N}_0$ when interpreting each assignment as a binary encoded integer. Formally, if s, t are two assignments over n variables $\{x_1,\dots,x_n\}$, then it holds that

$$s \leq t \ \Leftrightarrow\ \sum_{i=1}^{n} s(x_i)2^i \leq \sum_{i=1}^{n} t(x_i)2^i.$$

Definition 3.3.2 (Size-Lex-order) Let R, S be two teams. Then we define the *Size-Lex-order* $\leq_{\text{size-lex}}$ on the set of all teams by

$$R \leq_{\text{size-lex}} S :\Leftrightarrow |R| < |S| \text{ or } (|R| = |S| \text{ and } R \leq_{\text{lex}} S),$$

where $\leq_{\text{lex}}$ is the lexicographical order induced by the lexicographical order on assignments. Note that $\leq_{\text{size-lex}}$ is total and extends $\leq_{\text{size}}$.

3.3.1 Sorting orbits

We replace the algorithm for enumerating the orbit of a fixed team (Algorithm 1) by another procedure such that the outputted teams are sorted according to the Size-Lex-order. The trick of the method presented below is to partition the orbit with regard to the minimal assignment of each team. This yields parts which not only are easy to compute and to sort, but also are compatible with the Size-Lex-order, meaning that, if $T_1 <_{\text{size-lex}} T_2$ holds for two teams T_1, T_2 located in different parts P_1, P_2, then $T_1 <_{\text{size-lex}} T_2$ holds for all pairs $(T_1, T_2) \in P_1 \times P_2$.

We would like to point out the interesting observation that—in order to enumerate orbits—we utilise the fact of satisfying assignments for DNF-formulas in $\mathcal{PL}$ being efficiently enumerable (Proposition A.5). Hence we reduce the problem of enumerating teams for DNF-formulas in $\mathcal{PDL}(\veebar)\backslash\{\vee\}$ to its classical counterpart.

For the remaining part of this section, let $\mathcal{V}$ be a set of variables with cardinality $n \in \mathbb{N}$.

Definition 3.3.3 Let $T \in \mathcal{P}\left(2^{\mathcal{V}}\right)$ and $s \in 2^{\mathcal{V}}$. Then define

$$\begin{aligned} \mathcal{M}(T, s) &:= \{R \in \mathbb{F}_2^n T : \min(R) = s\}, \\ m(T) \quad &:= \{s \in 2^{\mathcal{V}} : \mathcal{M}(T, s) \neq \emptyset\}. \end{aligned}$$

Lemma 3.3.4 *Let $T \in \mathcal{P}\left(2^{\mathcal{V}}\right)$ and $s \in 2^{\mathcal{V}}$. Then it follows that*

$$\mathcal{M}(T, s) = \{sR : R \in \mathcal{M}(T, \vec{0}),\ \min(sR) = s\}.$$

Proof: The containment $\supseteq$ is obvious. Let $R \in \mathcal{M}(T, s)$. We set $S = sR$. Then $\vec{0}$ is contained in S, yielding $S \in \mathcal{M}(T, \vec{0})$. Because of $R = sS$ and $\min(sS) = \min(R) = s$ the containment $\subseteq$ follows. □

Remark 3.3.5 The sets $\mathcal{M}(T, \cdot)$ partition the orbit $\mathbb{F}_2^n T$. By (3.4) in the proof of Theorem 3.2.15 the set $\mathcal{M}(T, \vec{0})$ is given by $\mathcal{M}(T, \vec{0}) = \{sT : s \in T\}$. Hence, $\mathcal{M}(T, \vec{0})$ can be computed in time $\mathcal{O}(|T|n)$ and has at most $|T|$ elements. Consequently,

$\mathcal{M}(T,\vec{0})$ is small enough to be sorted in polynomial time. Thanks to the previous lemma, any part $\mathcal{M}(T,s)$ may be constructed by computing the teams in $s\mathcal{M}(T,\vec{0})$, sorting the assignments of each team and removing all teams with s not being the minimal assignment. That is why $\mathcal{M}(T,s)$ has at most $|T|$ elements and can be computed in $\mathcal{O}(|T|^2 n \log|T|)$.

Simply iterating over all $s \in 2^{\mathcal{V}}$ and outputting the sets $\mathcal{M}(T,s)$ does not guarantee polynomial delay. If exponentially many, subsequent sets are empty, we do not output any team while iterating over a set of exponential size. For this reason we have to implement an efficient algorithm enumerating $m(T)$. Each assignment in $m(T)$ guarantees at least one outputted team which buys enough time to compute the next nonempty set $\mathcal{M}(T,\cdot)$.

Lemma 3.3.6 *Let $T \in \mathcal{P}\left(2^{\mathcal{V}}\right)$. Then it holds that*

$$m(T) = \{s \in \mathbb{F}_2^n \cong 2^{\mathcal{V}} : \exists R \in \mathcal{M}(T,\vec{0})\ \forall t \in R \setminus \{\vec{0}\} : s_{\text{last}(t)} = 0\}.$$

Proof: Let $R \in \mathcal{M}(T,\vec{0})$ and $s \in \mathbb{F}_2^n \cong 2^{\mathcal{V}}$. We claim that

$$\min(sR) = s \ \Leftrightarrow\ s_{\text{last}(t)} = 0\ \forall t \in R \setminus \{\vec{0}\}. \tag{3.7}$$

W.l.o.g. $s \neq \vec{0}$ and $|R| > 1$, as (3.7) is obvious for these cases. We prove the contraposition, which is

$$\min(sR) \neq s \ \Leftrightarrow\ \exists t \in R \setminus \{\vec{0}\} : s_{\text{last}(t)} = 1.$$

$\Leftarrow$: Let $t \in R \setminus \{\vec{0}\}$ such that $s_{\text{last}(t)} = 1$. Set $u := s + t \in sR$. Obviously $u_i = s_i$ holds for all $i \in \mathbb{N}$ with $i > \text{last}(t)$. Furthermore we have

$$u_{\text{last}(t)} = s_{\text{last}(t)} + t_{\text{last}(t)} = 1 + 1 = 0 < 1 = s_{\text{last}(t)},$$

so that $u < s$ follows because assignments are ordered lexicographically. On that account we obtain $\min(sR) \leq u < s$.

$\Rightarrow$: Let $\min(sR) \neq s$. Then we have $u := \min(sR) < s$. Set $\vec{0} \neq t := s + u \in s + sR = R$. For all $i \in \mathbb{N}$ with $i > \text{last}(t)$ it holds that

$$t_i = s_i + u_i = 0,$$

hence $s_i = u_i$. Furthermore, $t_{\text{last}(t)} = 1$ yields $s_{\text{last}(t)} \neq u_{\text{last}(t)}$. Because u is lexicographically smaller than s, we obtain $s_{\text{last}(t)} = 1$.

Now, for every $s \in 2^{\mathcal{V}}$ it holds that

$$\begin{aligned} s \in m(T) &\Leftrightarrow \{sR : R \in \mathcal{M}(T, \vec{0}),\ \min(sR) = s\} \neq \emptyset && \text{(by Lemma 3.3.4)} \\ &\Leftrightarrow \exists R \in \mathcal{M}(T, \vec{0}) : \min(sR) = s \\ &\Leftrightarrow \exists R \in \mathcal{M}(T, \vec{0})\ \forall t \in R \setminus \{\vec{0}\} : s_{\text{last}(t)} = 0. && \text{(by (3.7))} \end{aligned}$$

□

Lemma 3.3.7 *Let T be a team over $\mathcal{V}$ with cardinality k. Then the set $m(T)$ may be enumerated lexicographically with delay $\mathcal{O}(k^2 n)$.*

Proof: Notice that the condition

$$\exists R \in \mathcal{M}(T, \vec{0})\ \forall t \in R \setminus \{\vec{0}\} : s_{\text{last}(t)} = 0$$

translates to the classical propositional DNF-formula

$$\psi := \bigvee_{R \in \mathcal{M}(T, \vec{0})} \bigwedge_{i \in \text{last}(R \setminus \{\vec{0}\})} \neg x_i.$$

By Remark 3.3.5, $\mathcal{M}(T, \vec{0})$ is computed in $\mathcal{O}(kn)$ and contains at most k elements. For any $R \in \mathcal{M}(T, \vec{0})$ the set $\text{last}(R \setminus \{\vec{0}\})$ may be computed by searching the last set bit in each assignment of R. This costs $\mathcal{O}(kn)$ because of $|R| = |T| = k$. Overall, ψ is constructed in $\mathcal{O}(k^2 n)$ time. By Lemma 3.3.6 enumerating $m(T)$ is equivalent to enumerating the solutions of ψ, which is achieved with delay $\mathcal{O}(kn)$ due to Proposition A.5. □

Theorem 3.3.8 *Let T be a team over $\mathcal{V}$ with cardinality k. Then Algorithm 5 enumerates $\mathbb{F}_2^2 T$ in Size-Lex-order with delay $\mathcal{O}(k^2 n \log k)$.*

Proof: Since all teams in $\mathbb{F}_2^2 T$ have the same cardinality, it is enough to verify that the output is sorted according to the lexicographical order. This is guaranteed by sorting the sets $\mathcal{M}(T, \cdot)$ and iterating over $m(T)$ lexicographically. Because of $\{\mathcal{M}(T, s) : s \in m(T)\}$ being a partition of $\mathbb{F}_2^n T$, the orbits of T are outputted without duplicates.

It remains to analyse the delay. As we iterate over $m(T)$, in each loop pass at least one element is outputted. Consequently the delay is bounded by the costs of executing the loop body. Comparing two k-teams is possible in $\mathcal{O}(kn)$. As the sets $\mathcal{M}(T, \cdot)$ contain at most k elements, they may be sorted in $\mathcal{O}(k^2 n \log k)$. The other costs annotated in Algorithm 5 are taken from Remark 3.3.5 and Lemma 3.3.7.

□

Algorithm 5: Enumerating orbits in Size-Lex-order

Input: a team T
Output: the orbit $\mathbb{F}_2^n T$ of T in Size-Lex-order

```
1 for s ∈ m(T) do                                   /* O(k²n) per element */
2 |   Compute M(T,s);                               /* O(k²n log k) */
3 |   Sort M(T,s);                                  /* O(k²n log k) */
4 |   for R ∈ M(T,s) do
5 |   |   output R;
6 |   end
7 end
```

3.3.2 Sorting satisfying teams for product terms

Before proceeding, it is advised to recall Notation 3.2.1. Let φ be a product term of a DNF-formula, that is $\varphi \in \mathcal{PDL}\backslash\{\vee\}$. In this section we enumerate all satisfying teams for φ according to the Size-Lex-Order. W.l.o.g. assume that φ is simplified as described in section 3.2.1. In particular, $\mathcal{T}_k$ is invariant with respect to the group action of flipping bits, so that the orbits generated by $\mathcal{T}_k^0$ partition $\mathcal{T}_k$. Similar to Algorithm 6 we compute and output the sets $\mathcal{T}_{k-1}$ while constructing $\mathcal{T}_k^0$. However, the sets $\mathcal{T}_{k-1}$ have to be sorted accordingly. This is achieved by merging all orbits of $\mathcal{T}_{k-1}$. Since the amount of orbits grows exponentially by Theorem 3.2.32, we have to implement a slightly more sophisticated merging routine to guarantee polynomial delay.

Theorem 3.3.9 *Algorithm 6 enumerates $\mathcal{T}_k$ in Size-Lex-order with delay $\mathcal{O}(k^2n^2)$.*

Proof: Let R, S be teams outputted successively. Between outputting R and S at most two new teams are added to the list $\mathcal{L}$. For this reason S has to be one of:

- $\min(\mathcal{L})$ immediately after removing R from $\mathcal{L}$,
- The successor of R in $\mathbb{F}_2^n R$,
- The team $T_{\text{new\_orbit}}$ is assigned to when R is outputted.

Except for the last case $R < S$ is obvious. It is easy to verify that $T_{\text{new\_orbit}} < \min(\mathcal{T}_k^0)$ holds at all times except in line 1 and 7. If $T_{\text{new\_orbit}} \leq \min(\mathcal{L})$ holds in line 4, the team $T_{\text{new\_orbit}}$ is assigned to becomes the new minimal team in $\mathcal{L}$ and $T_{\text{new\_orbit}}$ is assigned to $\min(\mathcal{T}_k^0)$. Consequently, we have $\min(\mathcal{L}) < T_{\text{new\_orbit}}$ after executing the if statement. It follows that $R < S$ for the last case. Hence, all teams are outputted in lexicographical order and duplicates cannot occur. Furthermore the algorithm terminates, as one k-team is outputted for each loop pass and the set of k-teams is finite.

Algorithm 6: Enumerating satisfying k-teams in Size-Lex-order

Input: a team based propositional formula φ as in Equation (3.1), $k \in \mathbb{N}$, $\mathcal{T}_k^0$
Output: all k-teams T for φ with $T \models \varphi$ in Size-Lex-order

```
1  T_new_orbit ← min(T_k^0);                                   /* O(kn) */
2  T_k^0 ← T_k^0 \ {T_new_orbit};                              /* O(k²n²) */
3  L ← new List(Team);
4  while L not empty or T_new_orbit ≠ null do
5      if L empty or (T_new_orbit ≠ null and T_new_orbit ≤ min(L)) then
6          L ← L ∪ {T_new_orbit};                              /* O(k²n²) */
7          T_new_orbit ← min T_k^0;                            /* O(kn) */
8          T_k^0 ← T_k^0 \ {T_new_orbit};                      /* O(k²n²) */
9      end
10     T ← min(L);                                             /* O(kn) */
11     L ← L \ {T};                                            /* O(k²n²) */
12     output T;
13     T_next ← successor of T in F_2^n T;                     /* O(k²n log k) by Theorem 3.3.8 */
14     if T_next ≠ null then
15         L ← L ∪ {T_next};                                   /* O(k²n²) */
16     end
17 end
```

Next we show that the set of outputted teams is $\mathcal{T}_k$. Let T be an outputted team. Then T is located in an orbit generated by a team in $\mathcal{T}_k^0$. Because $\mathcal{T}_k$ is invariant with respect to the group action (see Theorem 3.2.8), it follows that $T \in \mathcal{T}_k$. Vice versa, let $T \in \mathcal{T}_k$. By Lemma 3.2.9 a team $T_0 \in \mathcal{T}_k^0$ exists such that $T \in \mathbb{F}_2^n T_0$. W.l.o.g. let T_0 be minimal with this property. As the algorithm does not terminate before every team in $\mathcal{T}_k^0$ is processed, at some point T_0 has to be inserted into $\mathcal{L}$. Considering line 15, all teams in $\mathbb{F}_2^n T_0$ are outputted eventually. In particular T is outputted.

It remains to analyse the delay. The size of $\mathcal{L}$ is bounded by $|\mathcal{T}_k^0|$. Hence, by managing $\mathcal{L}$ in an AVL Tree the standard list operations are realised in $\mathcal{O}(k^2n^2)$ by Equation (3.6). The depth of the tree is bounded by kn. On that account extracting the minimal item costs $\mathcal{O}(kn)$. Since $|2^{\mathrm{Var}(\varphi)}| = 2^n$, we can assume that $k \leq 2^n$ and $\log k \leq n$. Consequently, each loop pass is executed in $\mathcal{O}(k^2n^2)$ steps. One team is outputted per pass, resulting in a delay of $\mathcal{O}(k^2n^2)$. □

Theorem 3.3.10 *Algorithm 7 enumerates all satisfying teams of φ with cardinality $\leq k$ in Size-Lex-Order. The delay is bounded by $\mathcal{O}(k^3|\varphi|^2)$.*

Algorithm 7: Enumerating satisfying teams in Size-Lex-order

Input: a team based propositional formula φ as in Equation (3.1), $k \in \mathbb{N}$
Output: all teams T for φ with $T \models \varphi$, $1 \leq |T| \leq k$, in Size-Lex-order

```
1 T_1^0 ← {{0⃗}};
2 for ℓ = 2,...,k+1 do
3 |   simultaneously do
4 |   |   Enumerate T_{ℓ−1} by Algorithm 6;
5 |   and do
6 |   |   Compute T_ℓ^0 by Algorithm 2;
7 |   end
8 |   if T_ℓ^0 = ∅ then break;
9 end
```

Proof: The correctness of the algorithm is obvious. By distributing the computation of $\mathcal{T}_\ell^0$, we obtain the delay $\mathcal{O}(k^3|\varphi|^2)$. The details are found in section 3.2.4. By Theorem 3.3.9, the sets $\mathcal{T}_{\ell-1}$ are enumerated with delay $\mathcal{O}(k^2n^2)$. On that account the overall delay is given by $\mathcal{O}(k^3|\varphi|^2)$. □

3.3.3 The algorithm

Let $\varphi \in$ DNF-$\mathcal{CPDL}$ with product terms $\varphi_1, \ldots, \varphi_r$, $r \in \mathbb{N}$. Since a team over $\mathrm{Var}(\varphi)$ satisfies φ iff it satisfies at least one of the product terms, the union of the sets

$$\mathrm{Sol}_i := \{T \in \mathcal{P}\left(2^{\mathrm{Var}(\varphi)}\right) : T \models \varphi_i,\ 1 \leq |T| \leq k\} \quad \forall i \in \{1, \ldots, r\}$$

equals the set of all satisfying teams of φ with cardinality $\leq k$. However, by now we cannot enumerate Sol_i immediately as the teams outputted by Algorithm 7 with input φ_i are defined over $\mathrm{Var}(\varphi_i)$ instead of $\mathrm{Var}(\varphi)$. For this reason we have to execute the enumeration algorithm with a modified set of variables, such that all outputted teams are defined over $\mathrm{Var}(\varphi)$. Hence, the amount of variables occurring in the enumeration algorithm for φ_i may not be bounded by $|\varphi_i|$ anymore, invalidating the claim concerning the delay in Theorem 3.3.10. We are able to recover it by replacing $|\varphi_i|$ by $\max\{|\varphi_i|, |\mathrm{Var}(\varphi)|\}$. Consequently, the delay when enumerating Sol_i in lexicographical order is

$$\mathcal{O}(k^3(\max\{|\varphi_i|, |\mathrm{Var}(\varphi)|\})^2) \subseteq \mathcal{O}(k^3|\varphi|^2).$$

Algorithm 8 computes the union $\bigcup_i \mathrm{Sol}_i$ through merging. Clearly, its delay is bounded by

$$|\varphi| \cdot \mathcal{O}(k^3|\varphi|^2) \subseteq \mathcal{O}(k^3|\varphi|^3).$$

Algorithm 8: Enumerating satisfying teams in DNF-$\mathcal{CPDL}$

Input: $\varphi = \varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_r$ such that $\varphi_i \in \mathcal{PDL} \setminus \{\vee\}$ for all $i \in \{1, \ldots, r\}$, $k \in \mathbb{N}$
Output: all teams T for φ with $T \models \varphi$, $1 \leq |T| \leq k$, in Size-Lex-order

```
1  for i = 1,...,r do                                   /* ≤ |φ| iterations */
2  |  T_i ← min(Sol_i);                                 /* O(k³|φ|²) */
3  end
4  while {i ∈ {1,...,r} : T_i ≠ null} ≠ ∅ do
5  |  I ← {i ∈ {1,...,r} : T_i ≠ null};                /* O(|φ|) */
6  |  I_min ← {i ∈ I : T_i ≤ T_j for all j ∈ I};       /* O(k|φ|²) */
7  |  Choose i_0 ∈ I_min;
8  |  output T_{i_0};
9  |  for i ∈ I_min do                                  /* ≤ |φ| iterations */
10 |  |  T_i ← successor of T_i in Sol_i;               /* O(k³|φ|²) */
11 |  end
12 end
```

We conclude:

Theorem 3.3.11 *The following holds:*

(i) P-EnumTeamPlSize(*DNF-$\mathcal{CPDL}$*) $\in$ ***DelayFPT***,

(ii) EnumTeamPlSize(*DNF-$\mathcal{CPDL}$*, f) $\in$ ***DelayP*** *for all polynomial time computable functions* $f \in n^{\mathcal{O}(1)}$.

4 Enumeration in Modal Logic

After studying team based propositional logic, we are going to investigate enumeration problems for team based modal logic. Based on complexity results for model checking problems, we derive NP-completeness and **DelayP**-membership for various fragments of $\mathcal{MDL}$. At last, we show that the problem of enumerating all maximal independent sets of a graph is equivalent to enumerating all maximal, satisfying teams for $\mathcal{MDL}\backslash\{\vee, \Diamond\}$-formulas.

In contrast to propositional logic, teams in modal logic are only defined when a Kripke model is given. For this reason instances of the upcoming problem contain a Kripke model in addition to a team based modal formula. This modification has considerable consequences concerning complexity since the size of teams, which are subsets of the provided Kripke model, becomes linear in respect of the input. First, the parametrisation for bounding the team size becomes superfluous. Second, we may efficiently enumerate satisfying teams for fragments being larger than in the previous chapter—for instance $\mathcal{ML}$. Although $\mathcal{PL} \subseteq \mathcal{ML}$ holds, we do not obtain a contradiction to $\textsc{EnumTeamPl}(\mathcal{PL}) \notin$ **DelayP**, if P $\neq$ NP. The reason for this is that the reduction in Remark 2.2.6 is not polynomial.

Problem 4.0.1 (EnumTeamMl) Let ϕ be a class of team based modal formulas. Then we define

$$\textsc{EnumTeamMl}(\phi) := (\mathcal{K}(\phi), \text{ Sol})$$

where

$$\text{Sol}((\varphi, K)) := \{\emptyset \neq T \in \mathcal{P}(K) : K, T \models \varphi\} \quad \text{for } (\varphi, K) \in \mathcal{K}(\phi).$$

When a Kripke model $K = (W, R, \pi)$ is given, we may assume that W is totally ordered. As an analogy to the orders defined in the beginning of chapter 3, we define the lexicographical order and the order of cardinality on subsets of worlds, namely teams, as well.

Problem 4.0.2 Let ϕ and Sol be as above. Then we define

$$\textsc{EnumTeamMlLex}(\phi) := (Q, \text{ Sol}, \leq_{\text{lex}}),$$
$$\textsc{EnumTeamMlSize}(\phi) := (Q, \text{ Sol}, \leq_{\text{size}})$$

where $\leq_{\text{lex}}$ and $\leq_{\text{size}}$ are the lexicographical order respectively the order of cardinality on teams.

In section 3.2 we made extensive use of the downward closure property, saving computation time by omitting teams with proper subsets falsifying a given formula. As a side effect of downward closure, most of the outputted teams carry no relevant information. Since all satisfying teams are contained in at least one maximal team, it is enough to determine all satisfying teams that are maximal in respect of inclusion. This observation motivates the following definitions.

Definition 4.0.3 Let φ be a downward closed, team based modal formula and K be a Kripke model for φ. Then a team $T \in \mathcal{P}(K)$ satisfying φ is called *maximal*, if $K, T' \not\models \varphi$ for all $T' \in \mathcal{P}(K)$ with $T \subsetneq T'$.

Problem 4.0.4 (EnumMaxTeamMl) Let ϕ be a downward closed class of team based modal formulas. Then we define

$$\text{EnumMaxTeamMl}(\phi) := (\mathcal{K}(\phi), \text{ Sol})$$

where

$$\text{Sol}((\varphi, K)) := \{\emptyset \neq T \in \mathcal{P}(K) : K, T \models \varphi \text{ and } T \text{ is maximal}\} \quad \text{for } (\varphi, K) \in \mathcal{K}(\phi).$$

4.1 NP-complete enumeration problems

We are going to identify fragments ϕ of team based modal logic for that the corresponding enumeration problem is NP-complete in the sense of section 3.1. When considering team based propositional logic, we obtained appropriate results by reducing from 3CNF-Sat. In this section we are going to reduce from NP-complete model checking problems which are provided in [EL12].

Theorem 4.1.1 *Let P be a set of productions such that $\{\wedge, \Box\} \subseteq P$. Then it holds that*

$$\text{MC}(\mathcal{F}(P)) \leq_m^p \text{Decide}(\text{EnumTeamMl}(\mathcal{F}(P)).$$

If $\mathcal{F}(P)$ is downward closed, it follows that

$$\text{MC}(\mathcal{F}(P)) \in \text{NP} \Rightarrow \text{Decide}(\text{EnumTeamMl}(\mathcal{F}(P)) \in \text{NP}.$$

Proof: Let φ be a formula in $\mathcal{F}(P)$, $K = (W, R, \pi)$ be a Kripke model for φ and $T \in \mathcal{P}(K)$ be a team. Let z be a new variable. We define a formula φ' and a Kripke

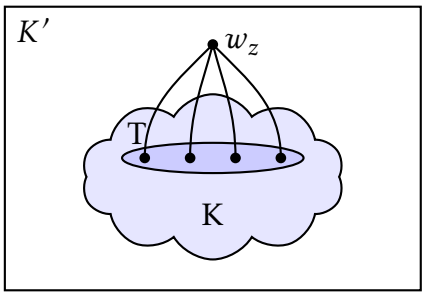

Figure 3: Construction of the expanded Kripke model K'.

model $K' = (W', R', \pi')$ over $\mathrm{Var}(\varphi) \cup \{z\}$ as follows:

$$\begin{aligned}
\varphi' &:= (z \wedge \Box\varphi) \in \mathcal{F}(P),\\
W' &:= W \cup \{w_z\}, \text{ where } w_z \notin W \text{ is a new world},\\
R' &:= R \cup \{(w_z, w) : w \in T\},\\
\pi'(w) &:= \pi(w) \quad \forall w \in W,\\
\pi'(w_z) &:= \{z\}.
\end{aligned}$$

The Kripke model K' is illustrated in Figure 3. Note that φ' and K' may be constructed in polynomial time with respect to the size of φ, K and T. We have

$$\begin{aligned}
(K, T, \varphi) \in \mathrm{MC}(\mathcal{F}(P)) &\Leftrightarrow K, T \models \varphi\\
&\Leftrightarrow K', \{w_z\} \models \Box\varphi\\
&\Leftrightarrow \exists T' \in \mathcal{P}(K') \setminus \{\emptyset\} : T' = \{w_z\} \text{ and } K', T' \models \Box\varphi\\
&\Leftrightarrow \exists T' \in \mathcal{P}(K') \setminus \{\emptyset\} : K', T' \models z \text{ and } K', T' \models \Box\varphi\\
&\Leftrightarrow \exists T' \in \mathcal{P}(K') \setminus \{\emptyset\} : K', T' \models \varphi'\\
&\Leftrightarrow (\varphi', K') \in \textsc{Decide}(\textsc{EnumTeamMl}(\mathcal{F}(P)),
\end{aligned}$$

proving the correctness of the reduction.

If $\mathcal{F}(P)$ is downward closed, $\textsc{Decide}(\textsc{EnumTeamMl}(\mathcal{F}(P))$ can be decided by checking all singletons. The claim follows immediately.

□

Corollary 4.1.2 *Let P be a set of productions such that $\{\wedge, \Box\} \subseteq P$ and $\mathcal{F}(P)$ is downward closed. If* $\mathrm{MC}(\mathcal{F}(P))$ *is* NP*-complete, then it follows that* $\textsc{EnumTeamMl}(\mathcal{F}(P))$ *is* NP*-complete.*

Proof: By the previous theorem it follows immediately that the decision problem $\textsc{Decide}(\textsc{EnumTeamMl}(\mathcal{F}(P))$ is NP-complete. □

Corollary 4.1.3 *Unless* P = NP, *it holds that* EnumTeamMl(ϕ) $\notin$ ***DelayP*** *for all fragments* $\phi \in \{\mathcal{MDL}\backslash\{\Diamond\}, \mathcal{MDL}\backslash\{\vee\}\}$.

Proof: The fragments $\mathcal{MDL}\backslash\{\Diamond\}$ and $\mathcal{MDL}\backslash\{\vee\}$ are downward closed. Both contain $\wedge$ and $\Box$. The corresponding model checking problems are NP-complete by [EL12]. On that account the requirements of the previous corollary are fulfilled, yielding that EnumTeamMl(ϕ) is NP-complete. The claim follows by Lemma 3.1.3. $\square$

We conclude that $\mathcal{MDL}\backslash\{\Diamond, \vee\}$, $\mathcal{MDL}\backslash\{\wedge\}$ and $\mathcal{MDL}\backslash\{\Box\}$ are the largest fragments possibly not resulting in NP-complete enumeration problems.

4.2 Enumerating polynomial verifiable fragments

In this section we are going to enumerate fragments of $\mathcal{MDL}$ for which the corresponding model checking problem is efficiently decidable. We present several enumeration algorithms for the problems defined in the beginning of the chapter. It is crucial that every algorithm becomes a **DelayP**-algorithm when queries such as "$K, T \models \varphi$?" can be resolved in polynomial time. By Ebbing and Lohmann, who presents appropriate model checking algorithms in [EL12], we obtain **DelayP**-algorithms for $\mathcal{ML}$ and $\mathcal{MDL}\backslash\{\vee, \Diamond\}$, or for $\mathcal{MDL}\backslash\{\wedge, \Diamond, \Box\}$ and $\mathcal{MDL}\backslash\{\wedge, \vee\}$ if the arity of $=(\cdot)$ is bounded by a constant.

Theorem 4.2.1 *Let* ϕ *a downward-closed class of team based modal formulas. If the model checking problem* MC(ϕ) *is decidable in polynomial time, then it holds that* EnumTeamMlLex(ϕ) $\in$ ***DelaySpaceP***.

Proof: Let (K, φ) be an instance of EnumTeamMlLex(ϕ). Let $W = \{w_1, \ldots, w_n\}$ be the ordered set of worlds of K such that $w_1 < w_2 < \cdots < w_n$. Then we set $\mathtt{next}(w_i) = w_{i+1}$ for all $i < n$. We show that Algorithm 9 is a **DelaySpaceP**-algorithm for EnumTeamMlSize(ϕ). Note that it is a modal variant of Algorithm 4, running the outer loop only once for $k = n$.

Since φ is downward closed, the proofs regarding Lemma 3.2.38 and 3.2.39 may be adopted, yielding that T iterates over all teams $S \in \mathcal{P}(K)$ with $K, S \models \varphi$ and $|S| < n$. Because $|S| < n$ trivially holds, all satisfying teams are outputted. Furthermore it is easy to verify that any team T is assigned to is lexicographically larger than the previous value for T.

We write $|T|$ for the size of the team currently assigned to T. Let us assume that T is assigned to an arbitrary team with $|T| = k$. We estimate an upper bound for the amount of loop passes before a team is outputted or the algorithm terminates.

Algorithm 9: Enumerating satisfying teams in lexicographical order

Input: $\varphi \in \phi$ and a Kripke model K
Output: all teams $T \in \mathcal{P}(K)$ with $K, T \models \varphi$ in lexicographical order

```
1  T ← {w_1};
2  while true do
3  |  if K, T ⊨ φ then output T;
4  |  w ← max(T);
5  |  if |T| < n and K, T ⊨ φ and w ≠ w_n then
6  |  |  T ← T ∪ {next(w)};
7  |  else if w ≠ w_n then
8  |  |  T ← T \ {w} ∪ {next(w)};
9  |  else if |T| > 1 then
10 |  |  T ← T \ {w};
11 |  |  w ← max(T);
12 |  |  T ← T \ {w} ∪ {next(w)};
13 |  else
14 |  |  break;
15 |  end
16 end
```

W.l.o.g. it holds that $K, T \not\models \varphi$ for every subsequent value for T. Consequently line 6 never is executed so that $|T| \leq k$ holds for all future values for T. The condition $w \neq w_n$ holds for at most n loop passes. For this reason $|T|$ is decremented after at most n loop passes. A simple induction over $|T|$ shows that the algorithm terminates after at most nk passes.

Since k is bounded by n, polynomial delay follows iff each loop pass performs in polynomial time. However, this is obvious because the only critical operation of checking $K, T \models \varphi$ may be implemented efficiently due to $\mathrm{MC}(\phi) \in \mathrm{P}$. The space consumption is given by the largest team assigned to T plus the space requirements for checking $K, T \models \varphi$, which is polynomial. □

Theorem 4.2.2 *Let ϕ a downward-closed class of team based modal formulas. If the model checking problem* $\mathrm{MC}(\phi)$ *is decidable in polynomial time, then it holds that* $\textsc{EnumTeamMlSize}(\phi) \in \boldsymbol{DelayP}$.

Proof: Using the same notation as in the proof of Theorem 4.2.1, we show that Algorithm 10 is a **DelayP**-algorithm for $\textsc{EnumTeamMlSize}(\phi)$. If defining $w > \max(\emptyset)$ to be true for all $w \in W$, each team $T \in \mathcal{P}(K)$ may be uniquely disassembled into

Algorithm 10: Enumerating satisfying teams ordered by cardinality

Input: $\varphi \in \phi$ and a Kripke model K
Output: all teams $T \in \mathcal{P}(K)$ with $K, T \models \varphi$ ordered by cardinality

```
1  L ← new Queue (Team);
2  L.push(∅);
3  while L is not empty do
4  |   T ← L.pop();
5  |   if T ≠ ∅ then output T;
6  |   for w ∈ W such that T = ∅ or w > max(T) do
7  |   |   T' ← T ∪ {w};
8  |   |   if K, T' ⊨ φ then L.push(T');
9  |   end
10 end
```

$T' \in \mathcal{P}(K)$ and $w \in W$ such that $|T'| = |T| - 1$ and $w > \max(T)$. On that account every team is inserted at most once into $\mathcal{L}$, preventing duplicates. Furthermore only satisfying teams are inserted. Note that $\emptyset$ is a satisfying team by Remark 2.2.7. Due to the downwards closure of φ, an induction over the size of teams implies that each satisfying team is inserted once into $\mathcal{L}$.

Next we show that teams in $\mathcal{L}$ are ordered by cardinality. We assume the contrary claim. Then we find teams T_1, T_2 such that $|T_2| < |T_1|$ but T_2 has been inserted into $\mathcal{L}$ after T_1. W.l.o.g. let T_1 be the first team inserted into $\mathcal{L}$ such that a team T_2 with the properties as above exists. We disassemble T_1 and T_2 into T_1', w_1, T_2' and w_2. Then T_2' had to be inserted after T_1' and it holds that $|T_2'| < |T_1'|$. However T_1' is inserted before T_1 as T_1 is pushed to $\mathcal{L}$ when T_1' is popped. This contradicts the existence of T_1.

Summing up, all satisfying teams are pushed to $\mathcal{L}$ such that the cardinality increases. Seeing that teams are outputted as soon as they are popped, it follows that Algorithm 10 is an enumeration algorithm for EnumTeamMlSize(ϕ). During each loop pass except the first, one team is outputted. For this reason the delay is polynomial. □

4.3 Enumerating maximal teams in $\mathcal{MDL}\backslash\{\vee, \Diamond\}$

One of the first papers introducing complexity classes for enumeration algorithms covers the problem of enumerating maximal independent sets of graphs [JYP88].

Showing that this problem is equivalent to EnumMaxTeamMl($\mathcal{MDL}\backslash\{\vee, \diamond\}$) is the purpose of this section. On that account we introduce the notion of polynomial time reductions being used in [DG07].

Definition 4.3.1 (m-reducible) Let $E_i = (Q_i, \mathrm{Sol}_i, \leq_i)$ be enumeration problems over posets S_i of possible solutions, $i \in \{1, 2\}$. We say that E_1 *is m-reducible to* E_2, in symbols $E_1 \leq_m E_2$, iff two computable functions $f : Q_1 \to Q_2$ and $g : S_2 \to S_1$ exist such that g is an order isomorphism between $\mathrm{Sol}(f(x))$ and $\mathrm{Sol}(x)$ for all $x \in Q_1$. That is:

- $g\big|_{\mathrm{Sol}(f(x))}$ is a bijection between $\mathrm{Sol}(f(x))$ and $\mathrm{Sol}(x)$,
- $r \leq_2 s \Leftrightarrow g(r) \leq_1 g(s)$ for all $r, s \in S_2$.

The problems are said to be *m-equivalent*, in symbols $E_1 \equiv_m E_2$, iff $E_1 \leq_m E_2$ and $E_2 \leq_m E_1$. If f and g are polynomial time computable, we write $E_1 \leq_m^p E_2$. Similarly we write $E_1 \equiv_m^p E_2$, iff $E_1 \leq_m^p E_2$ and $E_2 \leq_m^p E_1$.

Remark 4.3.2 By composing functions it is easy to see that $\leq_m^p$ is transitive. In addition, the complexity classes for enumeration problems introduced in Definition 2.3.6 are $\leq_m^p$-closed. Let E_1, E_2 be as above such that $E_1 \leq_m^p E_2$ holds via functions f and g. If an enumeration algorithm for E_2 is given, we obtain an enumeration algorithm for E_1 with the same complexity as follows:

Input: $x \in Q_1$

$y \leftarrow f(x)$;
for $s \in \mathrm{Sol}(y)$ **do** /* Apply enumeration algorithm for E_2 */
  output $g(s)$;
end

Note that one can easily relax the condition of g being an order isomorphism without loosing the properties described above. For instance in [Cre+17, Definition 10] a more general type of reduction is given. However, Definition 4.3.1 suffices to comprehend the upcoming proofs.

Problem 4.3.3 (MaxIndependentSet) Let Q be the set of undirected graphs. We define the enumeration problem

$$\textsc{MaxIndependentSet} := (Q, \mathrm{Sol}),$$

where $\mathrm{Sol}(G)$ is the set of all maximal independent sets of $G \in Q$.

Now we are ready to prove the previously mentioned equivalence in two steps, presenting reductions for both directions. The reduction from MaxIndependentSet to EnumMaxTeamMl($\mathcal{MDL}\backslash\{\vee, \diamond\}$) can be accomplished by using dependency atoms and conjunctions solely.

Lemma 4.3.4 *It holds that*

$$\textsc{MaxIndependentSet} \leq_m^p \textsc{EnumMaxTeamML}(\mathcal{F}(\wedge, =(\cdot))).$$

Proof: Let $G = (V, E)$ be an undirected graph. The adjacent nodes of an edge $e \in E$ are denoted by x_e^0 and x_e^1. We construct a Kripke model $K = (V, \emptyset, \pi)$ over $2E$ variables $\{x_e : e \in E\} \cup \{y_e : e \in E\}$ by setting

$$\pi(w) = \bigcup_{e \in E} \begin{cases} \{x_e\}, & w = x_e^0 \\ \{x_e, y_e\}, & w = x_e^1 \\ \emptyset, & \text{else.} \end{cases}$$

It follows immediately that

$$K, T \models =(x_e, y_e) \Leftrightarrow \{x_e^0, x_e^1\} \not\subseteq T \quad \forall T \in \mathcal{P}(K)$$

for every edge $e \in E$. Consequently, the teams satisfying

$$\varphi := \bigwedge_{e \in E} =(x_e, y_e)$$

exactly are the independent sets of G. The reduction is given by $f(G) = (K, \varphi)$ and $g \equiv \mathrm{id}_{\mathcal{P}(V)}$. □

The next lemma states that formulas in $\mathcal{MDL}\backslash\{\vee, \Diamond\}$ are "$\{1,2\}$-coherent" in the sense that satisfying teams are given by the satisfying teams of cardinality 1 and 2. For this reason reducing $\textsc{EnumMaxTeamML}(\mathcal{MDL}\backslash\{\vee, \Diamond\})$ to $\textsc{MaxIndependentSet}$ is accomplished by encoding all satisfying singletons as nodes and all falsifying 2-teams as edges.

Lemma 4.3.5 *Let $\varphi \in \mathcal{MDL}\backslash\{\vee, \Diamond\}$ and K be a Kripke model over $Var(\varphi)$. Then for all teams $T \in \mathcal{P}(K)$ it holds that*

$$K, T \models \phi \;\Leftrightarrow\; K, R \models \phi \;\forall R \subseteq T \text{ with } |R| \leq 2.$$

Proof: We conduct an induction over $|\varphi|$.

Induction basis: The claim follows trivially for the 1-coherent atoms x, $\neg x$, 0 and 1 respectively the 2-coherent atom $=(\cdot)$.

Induction step: The conjuction operator preserves coherence. If $\varphi = \psi \wedge \theta$, the claim follows by the induction hypothesis. The case $\varphi = \Box\psi$ remains. Let $T \in \mathcal{P}(K)$ be a team. As $\mathcal{MDL}\backslash\{\vee, \Diamond\}$ is downward closed, it is enough to show that

$$K, R \models \varphi \;\forall R \subseteq T \text{ with } |R| \leq 2 \;\Rightarrow\; K, T \models \varphi$$

respectively

$$K,T \not\models \varphi \Rightarrow \exists R \subseteq T \text{ with } |R| \leq 2 : K,R \not\models \varphi.$$

Assume that $K,T \not\models \varphi$. Hence $K,\mathrm{succ}(T) \not\models \psi$. By induction hypothesis we obtain a subset $R' \subseteq \mathrm{succ}(T)$ of cardinality ≤ 2 such that $K,R' \not\models \psi$. By taking a predecessor in T for each node in R' we find a subset $R \subseteq T$ such that $R' \subseteq \mathrm{succ}(R)$ and $|R| \leq 2$. In particular it holds that $K,R \not\models \varphi$. □

Lemma 4.3.6 *It holds that*

$$\textsc{EnumMaxTeamMl}(\mathcal{MDL}\backslash\{\vee,\Diamond\}) \leq_m^p \textsc{MaxIndependentSet}.$$

Proof: Let $\varphi \in \mathcal{MDL}\backslash\{\vee,\Diamond\}$ and $K = (W,R,\pi)$ be a Kripke model for φ. Inductively, we compute the falsifying 1- and 2-teams $\mathcal{F}_\psi$ for each subformula ψ of φ by the equalities

$$\begin{aligned}
\mathcal{F}_0 &= W \cup \{\{v,w\} : v,w \in W\},\\
\mathcal{F}_1 &= \emptyset,\\
\mathcal{F}_x &= \{\{v,w\} \subseteq W : x \notin \pi(v)\},\\
\mathcal{F}_{\neg x} &= \{\{v,w\} \subseteq W : x \in \pi(v)\},\\
\mathcal{F}_{=\!(P,Q)} &= \{\{v,w\} \subseteq W : \pi(v) \cap P = \pi(w) \cap P \text{ and } \pi(v) \cap Q \neq \pi(w) \cap Q\},\\
\mathcal{F}_{\psi\wedge\theta} &= \mathcal{F}_\psi \cup \mathcal{F}_\theta,\\
\mathcal{F}_{\Box\psi} &= \{T \subseteq W : |T| \leq 2 \text{ and } R \not\subseteq \mathrm{succ}(T)\ \forall R \in \mathcal{F}_\psi\}.
\end{aligned}$$

Note that the last equality holds due to Lemma 4.3.5.

Now we set

$$\begin{aligned}
V &:= W \setminus \left(\bigcup_{\{w\} \in \mathcal{F}_\varphi} \{w\} \right),\\
E &:= \{\{v,w\} \subseteq V : v \neq w \text{ and } \{v,w\} \in \mathcal{F}_\varphi\},
\end{aligned}$$

and obtain the undirected graph $G = (V,E)$. By construction and Lemma 4.3.5 it holds for $T \subseteq W$:

$$\begin{aligned}
K,T \models \varphi &\Leftrightarrow R \not\subseteq T\ \forall R \in \mathcal{F}_\varphi\\
&\Leftrightarrow T \subseteq V \text{ and } \mathcal{P}(T) \cap E = \emptyset\\
&\Leftrightarrow T \text{ is an independent set of } G.
\end{aligned}$$

For this reason the reduction is given by $f((K,\varphi)) = G$ and $g \equiv \mathrm{id}_{\mathcal{P}(K)}$. □

Theorem 4.3.7 *The following enumeration problems are m-equivalent via polynomial time computable reductions:*

- EnumMaxTeamMl($\mathcal{F}(\wedge, =(\cdot))$),
- EnumMaxTeamMl($\mathcal{MDL}\backslash\{\vee, \Diamond\}$),
- MaxIndependentSet.

In particular it holds that

$$\textsc{EnumMaxTeamMl}(\mathcal{MDL}\backslash\{\vee, \Diamond\}) \in \boldsymbol{DelayP}.$$

Proof: In this section we have proven that

$$\begin{aligned}\textsc{EnumMaxTeamMl}(\mathcal{MDL}\backslash\{\vee, \Diamond\}) &\leq_m^p \textsc{MaxIndependentSet} \\ &\leq_m^p \textsc{EnumMaxTeamMl}(\mathcal{F}(\wedge, =(\cdot))).\end{aligned}$$

Because of $\mathcal{F}(\wedge, =(\cdot)) \subseteq \mathcal{MDL}\backslash\{\vee, \Diamond\}$ we obtain

$$\textsc{EnumMaxTeamMl}(\mathcal{F}(\wedge, =(\cdot))) \leq_m^p \textsc{EnumMaxTeamMl}(\mathcal{MDL}\backslash\{\vee, \Diamond\}).$$

and the m-equivalences follow by the transitivity of $\leq_m^p$.
Johnson, Yannakakis, and Papadimitriou have shown in [JYP88] that

$$\textsc{MaxIndependentSet} \in \textbf{DelayP}.$$

Remark 4.3.2 yields $\textsc{EnumMaxTeamMl}(\mathcal{MDL}\backslash\{\vee, \Diamond\}) \in \textbf{DelayP}$. □

5 Conclusion

In this thesis we presented a novel approach to enumerate solutions in dependence logics. We applied elemental group theory in order to reduce the computational effort by an exponential factor with respect to the input size. As a result, we constructed an enumeration algorithm with polynomial delay regarding input and team size so that all satisfying teams for a formula in the fragment $\mathcal{PDL}\backslash\{\vee\}$ are outputted. In particular, we ordered the output by its cardinality and have shown that if this order is enforced, the algorithm is optimal regarding the team size. That is, no enumeration algorithm with polynomial delay exists if the team size is not restricted to a polynomial in the input. Furthermore we elucidated why a **DelayP**-algorithm for the full propositional dependence logic is unlikely, as its existence would yield P = NP.

By guaranteeing that all satisfying teams of equal cardinality are sorted according to the lexicographical order on teams, which itself is induced by the lexicographical order on assignments, we were able to modify our approach so that DNF-formulas in $\mathcal{PDL}(\textcircled{\vee})\backslash\{\vee\}$ are supported. Again, we stated that a **DelayP**-algorithm for arbitrary $\mathcal{PDL}(\textcircled{\vee})\backslash\{\vee\}$-formulas cannot exist unless P = NP.

Next, we considered the space consumption of our approach and noticed that it is at least exponential due to saving representative systems. If one wishes to reduce the space requirements to a polynomial level, we provide an **IncP**-algorithm. In addition, we have proven that such a result is not possible without restricting the team size. Any algorithm enumerating satisfying teams of arbitrary size features at least exponential space requirements.

By now we mostly presented algorithms sorting the output by its cardinality. It remains open to identify the enumeration complexity of $\mathcal{PDL}\backslash\{\vee\}$ when other orders— for instance the lexicographical order—are considered. Furthermore we did not investigate the fragment of $\mathcal{PDL}$ without conjunction yet. Exploiting the group action of flipping bits may be fruitful, because it is an invariant concerning satisfaction when formulas are simplified properly. However, in contrast to $\mathcal{PDL}\backslash\{\vee\}$, the 2-coherence property is lost so that one has to find other ways to compute representative systems.

When considering modal dependence logic, we have shown that the fragments constructed by either removing the $\Diamond$ or $\vee$ operator are not enumerable with polynomial delay unless P = NP. If both operators are absent, we demonstrated that the problem of enumerating all maximal teams is equivalent to the problem of enumerating all maximal independent sets in undirected graphs. In particular, both problems permit a **DelayP**-algorithm. Furthermore we developed a **DelaySpaceP**-algorithm in order to enumerate solutions for polynomial verifiable fragments of modal dependence logic. Another **DelayP**-algorithm presented in this thesis guarantees that the output is sorted by its cardinality. However, in this case we have to accept non polynomial space requirements. The question if the fragments $\mathcal{MDL}\backslash\{\wedge\}$ and $\mathcal{MDL}\backslash\{\Box\}$ permit **DelayP**-algorithms still remains unanswered.

A Referenced problems

Problem A.1 (Sat) The problem Sat is given by

$$\text{Sat} := \{\varphi : \varphi \text{ is a satisfiable propositional logic formula}\}.$$

Problem A.2 (3CNF-Sat) The problem 3CNF-Sat is given by

$$\text{3CNF-Sat} := \left\{ \varphi := \bigwedge_{i \in I} \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3} : |I| < \infty,\ \ell_{i,j} \text{ literal},\ \varphi \text{ satisfiable} \right\}.$$

Remark A.3 It is well known that Sat and 3CNF-Sat are NP-complete.

Problem A.4 (DNF-EnumSatLex) Let ϕ be the set of classical propositional formulas in disjunctive normal form and

$$\text{Sol}(\varphi) := \left\{ s \in 2^{\text{Var}(\varphi)} : s \models \varphi \right\} \quad \forall \varphi \in \phi.$$

We define the enumeration problem

$$\text{DNF-EnumSatLex} := (\phi,\ \text{Sol},\ \leq_{\text{lex}}),$$

where $\leq_{\text{lex}}$ is the lexicographical order on assignments.

Proposition A.5 *There exists an enumeration algorithm for* DNF-EnumSatLex *such that the solutions for a DNF-formula φ are enumerated with delay $\mathcal{O}(i \cdot (|Var(\varphi)| + \ell))$, where i is the amount of product terms in φ and ℓ the size of the largest product term.*

Proof: Let φ be a DNF-formula and $I \subset \mathbb{N}$ be a finite set such that

$$\varphi = \bigvee_{i \in I} \psi_i,$$

where the ψ_i are conjunctions of literals. First we show that the sets

$$\mathcal{S}_i := \{ s \in 2^{\text{Var}(\varphi)} : s \models \psi_i \} \quad \forall i \in I$$

may be enumerated with delay $\mathcal{O}(|\psi_i| + |\mathrm{Var}(\varphi)|)$. W.l.o.g we set $i = 1$ and $\psi = \psi_1$. Let $J, K \subseteq 2^{\mathrm{Var}(\varphi)}$ be subsets such that

$$\psi \equiv \bigwedge_{x \in J} x \wedge \bigwedge_{y \in K} \neg x.$$

If $J \cap K \neq \emptyset$, then ψ is not satisfiable and we may stop enumerating solutions immediately. Otherwise we output only assignments s with $s(J) \subseteq \{1\}$ and $s(K) \subseteq \{0\}$. The remaining variables $\mathrm{Var}(\varphi) \setminus (J \cup K)$ are assigned in lexicographical order when outputting $\mathcal{S}_1$.

At last $\mathrm{Sol}(\varphi) = \cup_{i \in I} \mathcal{S}_i$ may be enumerated by merging the lists $\mathcal{S}_i$. The algorithm is given by

for $i \in I$ **do**
  $s_i \leftarrow \min(\mathcal{S}_i)$;
end
while $\{i \in I : s_i \neq \textbf{null}\} \neq \emptyset$ **do**
  $L \leftarrow \{i \in I : s_i \neq \textbf{null}\}$;
  $L_{\min} \leftarrow \{\ell \in L : s_\ell \leq s_{\tilde{\ell}} \text{ for all } \tilde{\ell} \in L\}$;
  Choose $\ell_0 \in L_{\min}$;
  output s_{ℓ_0};
  for $\ell \in L_{\min}$ **do**
    $s_\ell \leftarrow$ successor of s_ℓ in $\mathcal{S}_\ell$;
  end
end

□

Bibliography

[AVL62] G.M. Adel'son-Vel'skii and E.M. Landis. 'An algorithm for the organization of information'. In: *Soviet Mathematics Doklady* 3 (1962), pp. 1259–1263.

[CR73] Stephen A. Cook and Robert A. Reckhow. 'Time Bounded Random Access Machines'. In: *Journal of Computer and System Sciences* 7.4 (1973), pp. 354–375. DOI: 10.1016/S0022-0000(73)80029-7.

[Cre+13] Nadia Creignou et al. 'Paradigms for Parameterized Enumeration'. In: *Mathematical Foundations of Computer Science*. Vol. 8087. Lecture Notes in Computer Science. Springer, 2013, pp. 290–301. DOI: 10.1007/978-3-642-40313-2_27.

[Cre+17] Nadia Creignou et al. 'On the Complexity of Hard Enumeration Problems'. In: *Language and Automata Theory and Applications*. Vol. 10168. Lecture Notes in Computer Science. Springer, 2017, pp. 183–195. ISBN: 978-3-319-53733-7. DOI: 10.1007/978-3-319-53733-7_13.

[DG07] Arnaud Durand and Etienne Grandjean. 'First-Order Queries on Structures of Bounded Degree are Computable with Constant Delay'. In: *ACM Transactions on Computational Logic* 8.4 (2007). DOI: 10.1145/1276920.1276923.

[DKV16] Arnaud Durand, Juha Kontinen, and Heribert Vollmer. 'Expressivity and complexity of dependence logic'. In: *Dependence Logic: Theory and Applications*. Birkhäuser, 2016, pp. 5–32. DOI: 10.1007/978-3-319-31803-5_2.

[EL12] Johannes Ebbing and Peter Lohmann. 'Complexity of Model Checking for Modal Dependence Logic'. In: *SOFSEM 2012: Theory and Practice of Computer Science*. Vol. 7147. Lecture Notes in Computer Science. Springer, 2012, pp. 226–237. ISBN: 978-3-642-27660-6. DOI: 10.1007/978-3-642-27660-6_19.

[FG06] Jörg Fluhm and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[Hod97] Wilfrid Hodges. 'Compositional Semantics for a Language of Imperfect Information'. In: *Logic Journal of the IGPL* 5.4 (1997), pp. 539–563. DOI: 10.1093/jigpal/5.4.539.

[JYP88] David S Johnson, Mihalis Yannakakis, and Christos H Papadimitriou. 'On generating all maximal independent sets'. In: *Information Processing Letters* 27.3 (1988), pp. 119–123. DOI: 10.1016/0020-0190(88)90065-8.

[Kon13] Jarmo Kontinen. 'Coherence and Computational Complexity of Quantifier-free Dependence Logic Formulas'. In: *Studia Logica* 101.2 (2013), pp. 267–291. DOI: 10.1007/s11225-013-9481-8.

[LV13] Peter Lohmann and Heribert Vollmer. 'Complexity Results for Modal Dependence Logic'. In: *Studia Logica* 101.2 (2013), pp. 343–366. DOI: 10.1007/s11225-013-9483-6.

[Rot95] Joseph J. Rotman. *An Introduction to the Theory of Groups*. Vol. 148. Graduate Texts in Mathematics. Springer, 1995.

[Sch09] Johannes Schmidt. 'Enumeration: Algorithms and Complexity'. MA thesis. Leibniz Universität Hannover & Université de la Méditerranée Aix-Marseille II, 2009. URL: https://www.thi.uni-hannover.de/fileadmin/forschung/arbeiten/schmidt-da.pdf.

[Sev09] Merlijn Sevenster. 'Model-theoretic and Computational Properties of Modal Dependence Logic'. In: *Journal of Logic and Computation* 19.6 (2009), pp. 1157–1173. DOI: 10.1093/logcom/exn102.

[Str10] Yann Strozecki. 'Enumeration complexity and matroid decomposition'. PhD thesis. Université Paris Diderot - Paris 7, 2010. URL: http://www.prism.uvsq.fr/~ystr/these_strozecki.

[Vä07] Jouko Vääanänen. *Dependence Logic*. Vol. 70. London Mathematical Society Student Texts. Cambridge University Press, 2007.

[Vä08] Jouko Väänänen. 'Modal Dependence Logic'. In: *New Perspectives on Games and Interaction*. Texts in Logic and Games 4 (2008), pp. 237–254.

Index

$\leq_{\text{lex}}$ 16
$\leq_m$, $\leq_m^p$ 55
$\leq_{\text{size}}$ 16
$\leq_{\text{size-lex}}$ 42
$\equiv_m$, $\equiv_m^p$ 55
$2^{\mathcal{V}}$ 4
3CNF-Sat 61

$\mathcal{CPDL}$ 41

Decide(E) 11
DelayFPT 12
DelayP 11
DelaySpaceP 11
DNF 15
DNF-$\mathcal{CPDL}$ 41
DNF-EnumSatLex 61

EnumMaxTeamMl(ϕ) 50
EnumTeamMl(ϕ) 49
EnumTeamMlLex(ϕ) 50
EnumTeamMlSize(ϕ) 50
EnumTeamPl(ϕ) 15
EnumTeamPlLex(ϕ, f) 16
EnumTeamPl(ϕ, f) 15
EnumTeamPlSize(ϕ, f) 16

$\mathcal{F}(P)$ 4

$G \circlearrowright X$ 13
G_x 13
Gx 13

hasNext 38

IncFPT 12
IncP 11

$\mathcal{K}(\phi)$ 8
$K, T \models \varphi$ 8

last($\cdot$) 23

MaxIndependentSet 55
max(T) 28
MC(ϕ) 8
$\mathcal{MDL}$ 7
$\mathcal{ML}$ 7
$m(T)$ 42
$\mathcal{M}(T, s)$ 42

n 19
next($\cdot$) 38

$\mathcal{P}(K)$ 7
$\mathcal{P}\left(2^{\mathcal{V}}\right)$ 4
$\mathcal{PDL}$ 5
P-EnumTeamPl(ϕ) 15
P-EnumTeamPlLex(ϕ) 16
P-EnumTeamPlSize(ϕ) 16
$\mathcal{PL}$ 4
poset 10

RAM 10

Sat 61
s_{first}, s_{last} 38

span(·) 23
succ(·) 7

$T \models \varphi$ 5
$T\big|_X$ 4
$\mathcal{T}_k, \mathcal{T}_k^0$ 19
t_k, t_k^0 19
$T_{\text{red}}^1, T_{\text{red}}^2$ 28

Var(φ) 4, 7

Erklärung zu dieser Arbeit

Hiermit erkläre ich, die vorliegende Masterarbeit selbstständig erstellt zu haben. Weiterhin versichere ich, dass sämtliche von mir verwendeten Hilfsmittel und Quellen im Literaturverzeichnis aufgeführt sind.

Christian Reinbold, Hannover, den 16.05.2017