Chapter 4

Descriptive complexity

Descriptive complexity theory is a field of finite model theory which studies the connection between logical definability and computational complexity. Two central problems considered in this field are the model checking and the satisfiability testing. The satisfiability testing for a logic $L$ and a class of finite structures $M$ is the following problem: Given a sentence $\phi \in L$, determine whether or not there is a structure in $M$ which satisfies $\phi$. The model checking problem for a logic $L$ and a class of finite structures $M$ is to determine whether $M \models \phi$ holds or not for a given formula $\phi \in L$ and a finite structure $M \in M$. Closely related problem to the model checking problem is the query evaluation problem: Given a formula $\phi(x_1, \ldots, x_k)$ and a structure $M$, it is to calculate the relation defined by the formula $\phi(x_1, \ldots, x_k)$, i.e. the set of tuples $\bar{a} \in M^k$ for which hold $(M, \bar{a}) \models \phi(x_1, \ldots, x_k)$. The query evaluation reduces to a polynomially many model checking problems.

The complexities of these problems are measured as a function of the size of the input. The taxonomy for measuring the computational complexity of different query languages was developed by Vardi in [37]. One usually differentiates between three complexities: Data complexity; The formula is fixed and the structure is given as an input. Expression complexity; The structure is fixed and the formula is given as a input. Combined complexity; Both the structure and the formula are given as input. Usually, combined complexity is exponentially higher than the data complexity. The computational complexity of the query evaluation for various logics is presented in the table 4.1 [38].

Definition 4.0.6. Suppose $M$ is a class of structures and $\Phi$ is a class formulas. The model checking problem for $M$ and $\Phi$, denoted with $MC(M, \Phi)$, is to determine whether $M \models \phi$ holds for given $M \in M$ and $\phi \in \Phi$.

When we fix the formula $\phi$ and the class of structures $M$ is clear from the context, we denote the model checking $MC(\ M, \{\phi\})$ with $MC(\phi)$.

We will consider both data- and combined complexity of the model checking problem of $D$-formulas over finite teams. We will give a characterization of the
Table 4.1: Data- and combined complexity for different logics.

<table>
<thead>
<tr>
<th>Logic</th>
<th>Data Complexity</th>
<th>Expression Complexity</th>
<th>Combined Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO</td>
<td>AC^0</td>
<td>PSPACE-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>FP</td>
<td>PTIME-complete</td>
<td>EXPTIME-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>(\Sigma_1^1)</td>
<td>NP-complete</td>
<td>NEXPTIME-complete</td>
<td>NEXPTIME-complete</td>
</tr>
<tr>
<td>PFP</td>
<td>PSPACE-complete</td>
<td>EXPSPACE-complete</td>
<td>EXPSPACE-complete</td>
</tr>
</tbody>
</table>

Data complexity for quantifier-free formulas in terms of coherence and number of conjunctions in the formula. We will observe that disjunction increases the computational complexity of the model checking for dependence logic formulas. In case of classical disjunction, the effect is linear, as you just have to check if one of the disjuncts is satisfied. In dependence logic, however, we have a weaker form of disjunction and it is more complex than the classical one. We will show that there is a leap in computational complexity between disjunctions of size one and two when considering the combined complexity of the model checking problem for dependence atoms.

### 4.1 Computational complexity theory

Computational complexity theory focuses on classifying problems according to their inherent difficulty. One measures the difficulty of a given problem by the amount of resources it take to solve it, such as the running time, or the memory space used by the algorithm. The complexity is measured as a function of the size of the input. A problem is regarded as inherently difficult if all the algorithms solving the problem require a large amount of resources. The following notation is commonly adopted:

**Definition 4.1.1.** Suppose \(f\) and \(g\) are functions from \(\mathbb{N}\) to \(\mathbb{N}\). We write

\[ f \in \mathcal{O}(g(n)), \]

if there are positive constants \(c\) and \(k\), such that \(0 \leq f(n) \leq cg(n)\) for all \(n > k\). We write \(f \in o(g(n))\) if

\[ \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0. \]

**Definition 4.1.2.** A function is called time-contractible, if there exists a Turing machine \(M\) which, given a string consisting of \(n\) ones, stops after exactly \(f(n)\) steps. Analogously, \(f\) is space-contractible, if there is a Turing machine \(M\) that stops after using exactly \(f(n)\) cells.
4.1. Computational complexity theory

Notice that the definitions for time-and space constructible functions may slightly vary depending on the source. Most of the commonly used functions such as polynomial-functions are both time- and space-constructible.

We will denote the class of languages recognized by a deterministic Turing machine working in time $f(n)$, where $n$ is the length of the input by $\text{TIME}(f(n))$ and languages recognized by a deterministic Turing machine using at most $f(n)$ memory cells, where $n$ is the length of the input by $\text{SPACE}(f(n))$. And respectively $\text{NTIME}(f(n))$ and $\text{NSPACE}(f(n))$ for the non-deterministic versions of these classes.

**Definition 4.1.3.**

$$\text{LOGSPACE} = \bigcup_{k \in \mathbb{N}} \text{SPACE}(k \cdot \log(n)),$$

$$\text{NL} = \bigcup_{k \in \mathbb{N}} \text{NSPACE}(k \cdot \log(n)),$$

$$\text{PTIME} = \bigcup_{k \in \mathbb{N}} \text{TIME}(n^k),$$

$$\text{NP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k),$$

$$\text{PSPACE} = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k),$$

$$\text{EXPTIME} = \bigcup_{k \in \mathbb{N}} \text{TIME}(2^{n^k}),$$

$$\text{NEXPTIME} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(2^{n^k}),$$

$$\text{EXPSPACE} = \bigcup_{k \in \mathbb{N}} \text{SPACE}(2^{n^k}).$$

It is known that

$$\text{LOGSPACE} \subseteq \text{NL} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME} \subseteq \text{EXPSPACE}.$$  

The strictness of this hierarchy is open for most of the cases. The strictness of some of the containments follow from the Hierarchy theorems for time and space see [33] and [13]:

**Theorem 4.1.4.** [18](Deterministic Time hierarchy theorem) Suppose $f(n)$ is a time-constructible function. Then $\text{TIME}(f(n)) \subset \text{TIME}(f^2(n))$.

**Theorem 4.1.5.** [6](Nondeterministic Time Hierarchy Theorem) Suppose $g(n)$ is a time-constructible function, and $f(n + 1) = o(g(n))$. Then $\text{NTIME}(f(n)) \subset \text{NTIME}(g(n))$. 


Chapter 4. Descriptive complexity

**Theorem 4.1.6.** (Space Hierarchy Theorem) Suppose \( f(n) \) is a space-constructible function, such that \( f(n) > \log n \) and \( f(n) = o(g(n)) \). Then \( \text{SPACE}(f(n)) \subseteq \text{SPACE}(g(n)) \).

Savitch’s theorem establishes a relation between deterministic and non-deterministic space.

**Theorem 4.1.7.** [34](Savitch’s theorem) Suppose \( f(n) \) is a function, such that \( f(n) > \log n \). Then \( \text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n)) \).

The Hierarchy theorems for time and space and Savitch’s Theorem give the following known inequalities:

**Corollary 4.1.8.**

\[
\begin{align*}
\text{NL} & \subset \text{PSPACE}, \\
\text{PSPACE} & \subset \text{EXPSPACE}, \\
\text{PTIME} & \subset \text{EXPTIME}, \\
\text{NP} & \subset \text{NEXPTIME}.
\end{align*}
\]

The Hierarchy theorems for time and space give us the last three inequalities. Savitch’s Theorem shows that \( \text{NL} \subseteq \text{SPACE}(\log 2n) \), while the Space Hierarchy Theorem shows that \( \text{SPACE}(\log 2n) \subseteq \text{SPACE}(n) \). Thus giving us that \( \text{NL} \subseteq \text{PSPACE} \).

Measuring computational complexity using Turing machines as computational model one has to encode finite structures into strings. For this, one has to assume some ordering of the structure. The encoding in itself is not relevant as long as certain conditions are met. For example, the size of the encoding should be at most polynomial to the size of the structure.

We denote all the finite \( \tau \)-structures with \( M^\tau \). Let us fix an encoding \( \text{Bin} \) of finite \( \tau \)-structures into binary words. We denote the encoding of a structure \( M \) with \( \text{Bin}(M) \).

**Definition 4.1.9.** Let \( \phi \in L(\tau) \)-formula. Then let

\[ \mathcal{K}(\phi) = \{ M \mid M \in M^\tau \land M \models \phi \} \].

**Definition 4.1.10.** Suppose \( \mathcal{K} \) is a class of \( \tau \)-structures. Then \( \mathcal{L}(\mathcal{K}) \) is the language defined by \( \mathcal{K} \) under the encoding \( \text{Bin} \);

\[ \mathcal{L}(\mathcal{K}) = \{ \text{Bin}(M) \mid M \in \mathcal{K} \} \].

Also each language can be seen as a class of structures in the following way.
4.1. Computational complexity theory

**Definition 4.1.11.** Suppose $L$ is a language over some finite vocabulary $\tau$. A word $w = v_0 \ldots v_n \in L$ can be seen as a structure $W(w)$ over vocabulary $\leq \cup \{ P_v \mid v \in \Sigma \}$ with universe $\{v_0, \ldots, v_n\}$ and with the natural interpretation for $\leq$ and $P_v = \{i \mid v_i = a\}$. Now let $W(L)$ be defined the following way for each language $L$:

$$W(L) = \{W(w) \mid w \in L\}.$$ 

Thus, mathematical structures can be encoded into languages which can be then given as inputs for Turing machines. On the other hand, languages can be turned into classes of mathematical structures, which can be then characterized using logic. When there is a perfect match between a definability in logic $L$ and computability in a complexity class $C$, we say that the logic $L$ characterizes the complexity class $C$:

**Definition 4.1.12.** Let $L$ be a logic and $C$ a complexity class. We say that the logic $L$ characterizes a complexity class $C$, if for any vocabulary $\tau$ and any class of $\tau$-structures $K$ the following two conditions are equivalent:

1. $K$ is definable in $L$,
2. $L(K) \in C$.

We denote it by $L \equiv C$ and $\equiv \leq$ for equivalence over ordered structures.

Fagin initiated the field of descriptive complexity theory in [9]. He showed that $\Sigma^1_1$ characterizes the non-deterministic polynomial time. Another classical result is by Immerman and Vardi, that polynomial time is characterized by first order logic extended by a least fixed point operator (FO(LFP)) over ordered structures [23], [37]. It is considered an important open problem to find a characterizing logic for PTIME over all finite structures.

**Theorem 4.1.13.** ([9]) Fagin’s theorem:

$$\Sigma^1_1 \equiv NP.$$ 

**Theorem 4.1.14.** ([23], [37]) Immerman-Vardi theorem:

$$FO(LFP) \equiv \leq PTIME.$$ 

Many of the main classes are given logical characterizations, e.g. first-order logic extended by deterministic transitive closure operator ($FO(DTC)$) characterizes LOGSPACE over ordered structures and first-order logic extended by the transitive closure operator ($FO(TC)$) characterizes NL over ordered structures. Second-order logic extended with second order transitive closure operator ($SO(TC^2)$) characterizes PSPACE over all finite structures.

A problem $L$ is said to be complete for a class $C$ if every other problem in $C$ can be reduced to $L$. The theory of complete problems was initiated by Cook in
Chapter 4. Descriptive complexity

[7]. He showed that determining the satisfiability of a given boolean first order formula is complete for NP. Soon after, Karp showed the NP-completeness of 21 famous combinatoric- and graph-theoretic problems in [25]. For more NP-complete problems see [12].

**Definition 4.1.15.** A function $f : A \to B$ is $C$-computable if there is a Turing machine $M$ working in $C$, which for each input $a \in A$ returns the value $f(a)$.

**Definition 4.1.16.** Suppose $L_1 \subseteq A$ and $L_2 \subseteq B$. $L_1$ is $C$-reducible to $L_2$, if there is a $C$-computable function $f : A \to B$, such that for all $\alpha \in A$

$$\alpha \in L_1 \iff f(\alpha) \in L_2.$$ 

We denote this by $L_1 \leq_C L_2$.

**Definition 4.1.17.** Let $L$ be a language, $\leq_r$ is a reducibility relation and $C$ is a complexity class. We say that $L$ is complete for $C$ with respect to $r$-reductions if the following two conditions are met:

1. $L \in C$,
2. for all $L' \in C$, $L' \leq_r L$.

NP-complete problems are usually considered with respect to polynomial time reductions.

**Definition 4.1.18.** Boolean satisfiability problem (SAT) is a decision problem to determine whether a given propositional first order formula is satisfiable. The variables are boolean and may occur positively or negatively in the formula. The formulas are assumed to be in the conjunctive normal form. The problem is to determine, whether there is an assignment, which evaluates the given formula true. There are several variations of SAT from which we consider the following two:

- 2-SAT: At most 2 disjuncts in each clause.
- 3-SAT: At most 3 disjuncts in each clause.

2-SAT is shown to be complete for NL (for proof see) [33] and 3-SAT complete for NP [25]. Another well-know NP-complete problem is the 3-colorability of a graph [12].

**Definition 4.1.19.** $k$-colorability of a graph ($k$-COL) is a decision problem to determine whether the vertices of a given graph can be colored with $k$ colors in such a way that, if two vertices share an edge, then they are colored with different colors. In other words, if there is a function $\xi : V \to \{0, \ldots, 1, k - 1\}$, such that if $(v, w) \in E$, then $\xi(v) \neq \xi(w)$.

We will use reductions to 2-SAT, 3-SAT and $k$-COL to show completeness of certain model checking problems of $D$-formulas.
4.2 Computational complexity of dependence logic formulas

We are interested in finding a connection between the complexity of the model checking problem and the syntactic form of $\phi$. The model checking for all $D$-formulas is in NP. We will show that coherent formulas can be verified in logarithmic space.

Recall, that the formulas of dependence logic are verified in terms of sets of assignments. A formula $\phi \in D$ defines a collection of pairs $(M, X)$, where $M$ is a $\tau$-structure and $X$ is a team with range $M$, such that $Fr(\phi) = dom(X)$. Thus every formula of dependence logic can be seen as a collection of $\tau \cup \{R\}$-structures, where $R$ is a $|Fr(\phi)|$-ary relation symbol interpreting the team $X$. When we consider sentences, the team is always the same team containing just the empty assignment, $\{\emptyset\}$, and it can be naturally left out, but for quantifier-free formulas, which we are mainly focused in, we have to take the team into consideration. When given a structure $M$ and a team $\mathcal{X}$, the pair $(M, \mathcal{X})$ is coded as a structure $(M, Rel(\mathcal{X}))$.

**Theorem 4.2.1.** $D \equiv NP$.

**Proof.** Suppose $\phi \in D(\tau)$ defines the class $K(\phi)$ of pairs $(M, X)$, where $M$ is a finite $\tau$-structure and $X$ is a team with range $M$ and domain $Fr(\phi)$. By Theorem 2.1.5 $K(\phi)$ is definable in $\Sigma_1^1$ as a collection of $\tau \cup \{R\}$-structures $K(\phi^*)$, where $R$ is interpreted as $Rel(\mathcal{X})$.

Furthermore by Theorem 4.1.13, it holds that $((M, Rel(\mathcal{X}))) \in K(\phi^*)$ can be decided in NP. Thus $(M, \mathcal{X}) \in K(\phi)$ can be decided in NP.

Other direction: Suppose $K \in NP$. Then by Theorem 4.1.13 $K$ is definable in $\Sigma_1^1$. By Theorem 2.1.6, $K$ is also definable in $D$. \qed

There are several related results on the computational complexity of fragments of $\Sigma_1$ and for partially ordered quantifiers: Gottlob, Kolaitis and Schwentick characterize $\Sigma_1$-formulas with respect to their quantifier prefixes over directed, undirected and undirected graphs with self-loops in [14]. Grädel considers certain fragments of SO, which collapse to their existential fragments, which in the presence of a successor relation provide characterizations for LOGSPACE, NL and PTIME [17].

Blass and Gurevich observe the connection between NP-computability and definability with Henkin quantifiers [2]. When the prefixes of the FO formulas are linearly ordered, it is just the classical quantification of FO and the formula can be verified in LOGSPACE. They show that all non-linear Henkin quantifiers can express NP-complete problems as long as the existentially quantified variables range over the whole universe. They impose constraints on the existentially
quantified variables and show that
\[
(\forall x_1 \ldots \forall x_{1k} \exists y_1) \\
(\forall x_{21} \ldots \forall x_{2k} \exists y_2)
\]  
(4.1)

can express NP-complete problems as long as the variables \(y_1\) and \(y_2\) range over at least a three element set. When \(y_1\) and \(y_2\) are boolean variables they can be verified in NL. Furthermore, they show that the following two quantifiers are enough to express NP-complete problems:
\[
(\forall x_1 \exists \alpha_1) \\
(\forall x_2 \exists \alpha_2) \\
(\forall x_3 \exists \alpha_3)
\]  
(4.2)
\[
(\forall x_1 \exists \gamma_1) \\
(\forall x_2 \exists \gamma_2)
\]  
(4.3)
where, \(\alpha_i, i \leq 3\), are boolean and \(\gamma_i, i \leq 2\) range over three element domain.

One method for obtaining results for fragments of dependence logic would have been to map the fragments of dependence logic into fragments of \(\Sigma_1^1\) or into fragments of first-order logic defined with Henkin quantifiers, for which the computational complexity is known. We did not find this approach fruitful since at least the straightforward translations of formulas did not seem to give any non-trivial results.

It is known that the data complexity of the model checking of first order formulas can be done in LOGSPACE (see [16] for proof).

**Theorem 4.2.2.** Suppose \(\phi \in FO(\tau)\). Then \(MC(\phi) \in LOGSPACE\).

**Theorem 4.2.3.** Suppose \(\phi \in D(\tau)\) is a \(k\)-coherent formula for some \(k \in \mathbb{N}\). Then \(MC(\phi) \in LOGSPACE\).

**Proof.** Suppose \(\phi \in D(\tau)\) is a \(k\)-coherent formula and that it defines a class \(K(\phi)\) of pairs \((M, \mathcal{X})\), where \(M\) is a \(\tau\)-structure and \(\mathcal{X}\) is a team of range \(M\) and domain \(Fr(\phi)\). Then by 3.2.5 there is \(FO(\tau \cup \{R\})\)-sentence \(\phi^*\), where \(R\) is \(|Fr(\phi)|\)-ary relation symbol interpreted as \(Rel(\mathcal{X})\), such that
\[
(M, \mathcal{X}) \in K(\phi) \iff (M, Rel(\mathcal{X})) \in K(\phi^*)
\]  
(4.4)

By 4.2.2, it holds that \((M, Rel(\mathcal{X})) \in K(\phi^*)\) can be decided LOGSPACE. Thus \((M, \mathcal{X}) \in K(\phi)\) can be decided in LOGSPACE. \(\square\)

### 4.3 Quantifier-free formulas

We will give characterization for the data complexity of quantifier-free formulas in terms of numbers of disjunctions in the formula and coherence. We will point
4.3. Quantifier-free formulas

out three thresholds, namely when the complexity of the model checking is in LOGSPACE, NL or in NP. We will also give complete instances for NL and NP.

Suppose $\phi(x_1, \ldots, x_k) \in D$ is a quantifier free formula. We will show that the following claims hold:

1. If $\phi$ is coherent, then $MC(\phi) \in \text{LOGSPACE}$ by Theorem 4.2.3.
2. If $\phi =: \theta \lor \psi$, where $\theta$ and $\psi$ are 2-coherent formulas, then $MC(\phi) \in NL$ and contains NL-complete instances.
3. Formulas of form $\theta \lor \psi$, where $\theta$ is 2-coherent and $\psi$ a 3-coherent formula contain NP-complete instances.

First, we will show that the model checking problem for disjunctions of 2-coherent formulas can be reduced to 2-satisfiability problem.

**Theorem 4.3.1.** Suppose $\phi$ and $\psi$ are 2-coherent $D$-formulas. Then

$$MC(\phi \lor \psi) \leq \text{LOGSPACE } 2 - \text{SAT}.$$ 

**Proof.** Suppose we are given a team $\mathcal{X} = \{s_1, \ldots, x_k\}$. We will go through all the two-element subsets $\{s_i, s_j\} \subseteq \mathcal{X}$, and construct an instance of 2-SAT in the following way:

- If $\{s_i, s_j\} \not \models \phi$, then $(x_i \lor x_j) \in C$.
- If $\{s_i, s_j\} \not \models \psi$, then $(\neg x_i \lor \neg x_j) \in C$.

We let $\Theta_X = \bigwedge_{\phi \in C} \phi$. Clearly, $\Theta_X$ is a proper instance of 2-SAT. We will next show that there is an assignment $S$, which satisfies $\Theta_X$ if and only if $\mathcal{X} \models \phi \lor \psi$ holds: Suppose there is an assignment $S : \text{Var}(\Theta_X) \rightarrow \{0, 1\}$, which evaluates $\Theta_X$ true. Let us define the partition of $\mathcal{X}$ in the following way:

- $Z = \{s_i \in \mathcal{X} \mid S(x_i) = 1\}$.
- $\mathcal{Y} = \mathcal{X} \setminus Z$.

Clearly it holds that $\mathcal{X} = Z \cup \mathcal{Y}$. Let us show that $Z \models \psi$ and $\mathcal{Y} \models \phi$ hold:

Suppose $s_i, s_j \in Z$. Since $S$ satisfies $\Theta_X$, $(\neg x_i \lor \neg x_j)$ cannot be a clause in $C_X$. By the construction above, it follows that $\{s_i, s_j\} \models \psi$ holds. Now, by 2-coherence of $\psi$ it follows that $Z \models \psi$.

Suppose $s_i, s_j \in \mathcal{Y}$. Since $S$ was assumed to satisfy $C_X$, $(x_i \lor x_j)$ cannot be a clause in $C_X$. It follows by the construction above that $\{s_i, s_j\} \models \phi$ holds. Again, from 2-coherence of $\phi$ it follows that $\mathcal{Y} \models \phi$.

The other direction: Suppose $\mathcal{X} \models \phi \lor \psi$ holds. Then, by Definition 2.0.10 it holds that there is a division of $\mathcal{X}$ into two sets $Z$ and $\mathcal{Y}$, such that $\mathcal{X} = Z \cup \mathcal{Y}$, $Z \cap \mathcal{Y} = \emptyset$, $\mathcal{Y} \models \phi$ and $Z \models \psi$. Let $S$ be defined the following way:
• $S(x_i) = 1$, if $s_i \in Z$.

• $S(x_i) = 0$, if $s_i \in Y$.

Clearly it holds that $S : \text{Var}(\Theta_X) \rightarrow \{0,1\}$ is a function. Let us show that $S$ satisfies $\Theta_X$: Suppose $\theta \in C$ of form $(x_i \lor x_j)$. Then $\{s_i, s_j\}$ fails $\phi$ by the construction of $\Theta_X$. Then $s_i$ and $s_j$ cannot be both in $Y$, since $Y$ was supposed to satisfy $\phi$. Thus, either $s_i$ or $s_j$ must be in $Z$. Then, it holds that $S(x_i) = 1$ or $S(x_j) = 1$, which implies that $S(x_i \lor x_j) = 1$.

Suppose $\theta$ is $(\neg x_i \lor \neg x_j)$. Then, by the construction of $\Theta_X$, it holds that $\{s_i, s_j\}$ fails $\psi$. Then, $s_i$ and $s_j$ cannot be both in $Z$, since $Z$ was supposed to satisfy $\psi$. Thus either $s_i$ or $s_j$ must be in $Y$. Then, it holds that $S(x_i) = 0$ or $S(x_j) = 0$, which implies that $S(\neg x_i \lor \neg x_j) = 1$.

Last, the complexity of this reduction is in LOGSPACE: We go through the 2-element subsets of the team $X$ and check if they fail $\phi$ or $\psi$. Since $\phi$ and $\psi$ were coherent, the model checking for the sub-teams can be done in LOGSPACE by Theorem 4.2.3.

**Corollary 4.3.2.** Suppose $\phi$ and $\psi$ are 2-coherent $D$-formulas. Then

$$MC(\phi \lor \psi) \in NL.$$ 

Next we will show that the set of formulas of form $\phi \lor \psi$, where $\phi$ and $\psi$ are 2-coherent contain NL-complete instances.

### 4.3.1 A complete instance for non-deterministic logarithmic space

We will reduce the problem 2-SAT to the model checking problem of the formula $=(x,y) \lor =(z,v)$.

**Theorem 4.3.3.** $2-SAT \leq_{\text{LOGSPACE}} MC(=(x,y) \lor =(z,v))$.

**Proof.** Suppose $\theta(p_0, \ldots, p_{m-1})$ is an instance of 2-SAT of the form $\bigwedge_{i \in I} E_i$, where each conjunct $E_i = (A_{i_1} \lor A_{i_2})$, $i \in I$, where $A_{i_j}$, $j \leq 1$, are positive or negative boolean variables.

We will construct a team $X$, such that the following are equivalent:

1. $X \models =(x,y) \lor =(z,v)$.

2. $\theta(p_0, \ldots, p_{m-1})$ is satisfiable.

For each conjunct $E_i$, $i \in I$, we create a team $X_{E_i}$ where we code the information required to satisfy $E_i$. Now, $E_i$ will be satisfied if one of the disjuncts will be true. Thus it has two conditions for being satisfied. We will code these conditions into the team we construct in the following way:
4.3. Quantifier-free formulas

We will have a variable \( z \) denote the clause \( E_i \), \( x \) denote the variables of the clause, \( y \) the truth value of the corresponding variable and \( v \) that makes sure we choose at least one of the assignments form each \( X_{E_i} \) into the sub-set of \( X \) which eventually codes the assignment which evaluates \( \theta \) true. Each disjunct \( A_{ij} \) gives a rise to one assignment. Now \( X \) is the union \( \bigcup_{i \in I} X_{E_i} \).

For example, the team \( X_{E_i} \) for a clause \((p_k \lor p_j)\) is the one in Table 4.2. The team for the whole instance of 2-SAT:

\[
(A_{01} \lor A_{02}) \land (A_{11} \lor A_{12}) \land \ldots \land (A_{I_1} \lor A_{I_2})
\]

is the one in Table 4.3, where \( t(A_i) = 1 \) if \( A_i \) is a positive variable and \( t(A_i) = 0 \), if \( A_i \) is a negated variable.

<table>
<thead>
<tr>
<th>( z )</th>
<th>( x )</th>
<th>( y )</th>
<th>( v )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i )</td>
<td>( p_k )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( i )</td>
<td>( p_j )</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.2: Team for \((p_k \lor p_j)\).

<table>
<thead>
<tr>
<th>( z )</th>
<th>( x )</th>
<th>( y )</th>
<th>( v )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( A_{01} )</td>
<td>( t(A_{01}) )</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>( A_{02} )</td>
<td>( t(A_{02}) )</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>( A_{11} )</td>
<td>( t(A_{11}) )</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>( A_{12} )</td>
<td>( t(A_{02}) )</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>( A_{21} )</td>
<td>( t(A_{21}) )</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>( A_{22} )</td>
<td>( t(A_{22}) )</td>
<td>2</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( n )</td>
<td>( A_{I_1} )</td>
<td>( t(A_{I_1}) )</td>
<td>1</td>
</tr>
<tr>
<td>( n )</td>
<td>( A_{I_2} )</td>
<td>( t(A_{I_2}) )</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.3: Team \( \bigcup_{i \in I} X_{E_i} \).

Suppose \( \theta(p_0, \ldots, p_{m-1}) \) is satisfiable. Then there exists an assignment \( F : \{p_0, \ldots, p_{m-1}\} \rightarrow \{0, 1\} \), such that \( F \) evaluates \( \theta(p_0, \ldots, p_{m-1}) \) true. We define the partition of the team \( X \) into two sets in the following way: For each \( s \in X \), \( s \in X_1 \) if the following condition holds:

\[
(s(x) = p_i) \rightarrow F(p_i) = s(y).
\]

(4.5)

Otherwise \( s \in X_2 \).
Condition (4.5) guarantees that the tuples that agree with the assignment \( F \) are chosen to \( \mathcal{X}_1 \). Since \( F \) evaluates \( \wedge_{i \in I} E_i \) to true, it evaluates every conjunct \( E_i \) true. As the satisfying conditions of each \( E_i \) are coded into \( \mathcal{X}_{E_i} \), the condition (4.5) is satisfied by at least one of the assignments in each \( \mathcal{X}_{E_i} \). Thus there will be at most one tuple from each \( \mathcal{X}_{E_i} \) in \( \mathcal{X}_2 \). Thus \( \mathcal{X}_2 \) trivially satisfies \( = (z, v) \) since all tuples in \( \mathcal{X}_2 \) disagree on \( z \). Next we will show that \( \mathcal{X}_1 \) satisfies \( = (x, y) \): Let \( s, s' \in \mathcal{X}_1 \), such that \( s(x) = s'(x) = p_i \). Then by (4.5) it follows that \( s(y) = F(p_i) = s'(y) \) holds. Thus \( \mathcal{X}_1 \models = (x, y) \).

The other direction: Suppose \( \mathcal{X} \models = (x, y) \lor = (z, v) \). There is a partition of \( \mathcal{X} \) into \( \mathcal{X}_1 \) and \( \mathcal{X}_2 \), such that \( \mathcal{X}_1 \models = (x, y) \) and \( \mathcal{X}_2 \models = (z, v) \). We will define the assignment \( F : \{p_0, \ldots, p_m\} \rightarrow \{0, 1\} \) in the following way:

1. If there exists \( s \in \mathcal{X}_1 \), such that \( s(x) = p_i \), then \( F(p_i) = s(y) \).
2. If \( \forall s \in \mathcal{X}_1 \) it holds \( s(x) \neq p_i \), then \( F(p_i) = 1 \).\(^1\)

Let us show that \( F : \{p_0, \ldots, p_m\} \rightarrow \{0, 1\} \) is a function, which evaluates \( \Theta(p_0, \ldots, p_{m-1}) \) true:

1. Clearly, \( \text{Dom}(F) = \{p_0, \ldots, p_{m-1}\} \) and \( \text{Range}(F) = \{0, 1\} \).
2. \( F \) is a function: Let \( p_i \in \{p_0, \ldots, p_m\} \). Suppose there exists \( s, s' \in \mathcal{X}_1 \), such that \( s(x) = s'(x) = p_i \) holds. Since \( \mathcal{X}_1 \models = (x, y) \) holds, it follows that \( s(y) = s'(y) \) holds. Suppose there are no \( s \in \mathcal{X}_1 \), such that \( s(x) = p_i \). Then by definition of \( F \) it holds that \( F(p_i) = 1 \).
3. \( F \) evaluates \( \Theta \) true: Note that \( z \) is constant and \( v \) is assigned different value by each tuple in each \( \mathcal{X}_{E_i} \). Thus \( \mathcal{X}_1 \) contains at least one of the tuples from each \( \mathcal{X}_{E_i} \). Let \( s_0 \in \mathcal{X}_{E_i} \), such that \( s_0 \in \mathcal{X}_1 \). Since each tuple codes a satisfying condition of \( E_i \) it follows that \( F \) evaluates one of the disjuncts in \( E_i \) true. Thus \( S(E_i) = 1 \).

Each conjunct of \( \Theta \) gives rise to a constant size team of two assignments with domain \( \{x, y, z, v\} \). Thus the team \( \mathcal{X} \) can be constructed in LOGSPACE when given \( \Theta \).

\( \Box \)

The problem 2-SAT is complete for NL [33]. Thus we have the following corollary.

**Corollary 4.3.4.** \( MC(= (x, y) \lor = (z, v)) \) is complete for NL.

\(^1\)If for all the assignments \( s \in \mathcal{X}_1 \) holds \( s(x) \neq p_i \), then the value of \( p_i \) is not relevant to the satisfiability of \( \Theta \). Thus the value of \( p_i \) can be chosen 0 or 1.
4.3. Quantifier-free formulas

4.3.2 A complete instance for non-deterministic polynomial time

We will show that the set of formulas of form $\phi \lor \psi$, where $\phi$ is 2-coherent and $\psi$ is 3-coherent formula contains NP-complete instances. We will show that 3-SAT can be reduced to the model checking problem of the formula $=(x, y) \lor =(z, u) \lor = (z, v)$.

Recall that an instance $\theta \in$ 3-SAT is a first-order formula in conjunctive normal form, where each conjunct has at most three variables: $\bigwedge_{i \in I} E_i$, where $I$ is finite. Each $E_i$ is of form $(A_{i_0} \lor A_{i_2} \lor A_{i_3})$, where $A_i$ is either a positive or a negated boolean variable. $\theta$ is accepted if there is an assignment, which evaluates $\theta$ true. The reduction is analogous with the reduction given in Theorem 4.3.3.

**Theorem 4.3.5.** 3-SAT $\leq$ LOGSPACE MC($=(x, y) \lor = (z, v) \lor = (z, v)$).

*Proof.* Suppose $\theta(p_0, \ldots, p_{m-1})$ is an instance of 3-SAT with conjuncts $E_i, i \in I$. We will construct a team $\mathcal{X}$, such that the following are equivalent:

- $\mathcal{X} | = (x, y) \lor = (z, v) \lor = (z, v)$.
- $\theta(p_0, \ldots, p_{m-1})$ is satisfiable.

For each conjunct $E_i, i \in I$, we create a team $\mathcal{X}_{E_i}$ where we code all the satisfying conditions of the clause $E_i$. Let $\mathcal{X} = \bigcup_{i \in I} \mathcal{X}_{E_i}$. For example, a clause $E_i = (p_l \lor \neg p_j \lor \neg p_k)$ will be satisfied if $p_l = 1$ or $p_j = 0$ or $p_k = 0$. The team for $(p_l \lor \neg p_j \lor \neg p_k)$ is the one in Table 4.4.

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$p_l$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$p_j$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>$p_k$</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.4: A team for $(p_l \lor \neg p_j \lor \neg p_k)$.

Suppose $\theta(p_0, \ldots, p_{m-1})$ is satisfiable. Then there exists an assignment $F : \{p_0, \ldots, p_{m-1}\} \rightarrow \{0, 1\}$, such that $F$ evaluates $\theta(p_0, \ldots, p_{m-1})$ true. We define $\mathcal{X}_1$ in the following way: For all $s \in \mathcal{X}$, $s \in \mathcal{X}_1$ if

$s(x) = p_l \rightarrow F(p_l) = s(y)$ \hspace{1cm} (4.6)

Since $F$ evaluates $\bigwedge_{i \in I} E_i$ true, it evaluates every conjunct $E_i$ true. Furthermore, since we coded all the satisfying conditions of $E_i$ into $\mathcal{X}_{E_i}$, it holds that at least one assignment from each $\mathcal{X}_{E_i}$ satisfies the condition (4.6). Thus $\mathcal{X}_1$ contains at least one assignment from each $\mathcal{X}_{E_i}$. Thus the two "leftover"-assignment form
each $X_E_i$ can be easily divided into $X_2$ and $X_3$ in such a way that $= (z, v)$ holds in both of them. We just place one of the assignments into $X_2$ and one into $X_3$.

Let us show that $X_1 = (x, y)$: Suppose $s, s' \in X_1$, such that $s(x) = s'(x) = p_i$. Then, by (4.6), it follows that $s(y) = s'(y) = F(p_i)$. Thus $X_1 = (x, y)$.

The other direction: Suppose $X = (x, y) \lor (z, v) \lor (z, v)$ holds. Then by the truth definition of the disjunction, it follows that $X$ can be partitioned into three sets $X_1, X_2$ and $X_3$, such that $X_1 = (x, y), X_2 = (z, v)$ and $X_3 = (z, v)$ hold. Let $F$ be defined in the following way for each variable $p_i$:

- If $\exists s \in X_1$, such that $s(x) = p_i$, then $F(p_i) = s(y)$.
- If $\forall s \in X_1$ it holds $s(x) \neq p_i$, then $F(p_i) = 1$.

Let us show that $F : \{p_0, \ldots, p_m\} \to \{0, 1\}$ is a function, which evaluates $\theta(p_0, \ldots, p_{m-1})$ true:

1. Clearly, $F$ is well defined and the domain of $F$ is $\{p_0, \ldots, p_{m-1}\}$ and the range is $\{0, 1\}$.

2. $F$ is a function: Let $p_i \in \{p_0, \ldots, p_m\}$. Suppose there exists $s, s' \in X_1$, such that $s(x) = s'(x) = p_i$ holds. Since $X_1 = (x, y)$ holds, it follows that $s(y) = s'(y) = F(p_i)$ holds. If there exists no $s \in X_1$, such that $s(x) = p_i$, then it holds by the definition of $F$, that $F(p_i) = 1$.

3. We will show that $S$ evaluates each $E_i, i \in I$, true: Note that $z$ is constant and $v$ gets different value by each tuple in each $X_{E_i}$. Thus $X_1$ must contain at least one of the tuples from each $X_{E_i}$. Since each tuple in $X_{E_i}$ codes a satisfying condition for $E_i$ it means that $F$ agrees with one of the satisfying conditions for $E_i$. Thus $F$ satisfies $E_i$.

Each conjunct of theta gives rise to a constant size team of three assignments with domain $\{x, y, z, v\}$. Thus given $\theta$, $X$ can be constructed in LOGSPACE.

By Theorem 4.2.1 it holds that $MC(= (x, y) \lor = (z, v) \lor = (z, v)) \in \text{NP}$. It is well-know that 3-SAT is complete for NP [25]. Thus we have the following corollary.

**Corollary 4.3.6.** $MC(= (x, y) \lor = (z, v) \lor = (z, v))$ is complete for NP.

### 4.4 Combined complexity of dependence logic formulas over finite teams

We will study the combined complexity of classes of 2-coherent formulas over all finite teams and the effect of disjunction on these classes. We will show that when we allow disjunction over two formulas the combined complexity of the
model checking for these classes is in $NL$ and that it becomes $NP$-complete when we allow disjunctions over three formulas. When we measure the combined complexity for a given set of formulas and set of structures, both the structure and the formula are coded as input.

**Definition 4.4.1.** Suppose $M$ is a class of finite structures and $\Phi$ is a class of formulas. Let $D_i(M, \Phi)$ be the decision problem with input $(\mathcal{X}, \phi_0, \ldots, \phi_{i-1})$, where $\mathcal{X} \in M$ and $\phi_j \in \Phi$ for all $j < i$. Problem is to determine whether $\mathcal{X} \models \bigvee_{j \leq i} \phi_j$.

**Theorem 4.4.2.** Suppose $C$ is a complexity class, such that $LOGSPACE \subseteq C$, $M$ is a set of finite teams closed under sub-teams and $\Phi$ is a class of 2-coherent formulas, such that $MC(M, \Phi) \in C$. Then

$$D_2(M, \Phi) \leq_C 2SAT.$$ 

*Proof.* Suppose $\mathcal{X}$ is a team, $\phi_0$ and $\phi_1 \in \Phi$. Let $Cl$ be the set of clauses defined the following way: For each 2-element subset $\{s_i, s_j\} \subseteq \mathcal{X}$:

- If $\{s_i, s_j\} \not\models \phi_0$ holds, then $(x_i \lor x_j) \in Cl$.
- If $\{s_i, s_j\} \not\models \phi_1$ holds, then $(\neg x_i \lor \neg x_j) \in Cl$.

Let $C_\mathcal{X} = \bigwedge_{\phi \in Cl} \phi$. Clearly, $C_\mathcal{X}$ is a proper instance of $2SAT$. Next we will show that $\mathcal{X} \models (\phi_0 \lor \phi_1)$ holds if and only if $C_\mathcal{X}$ is satisfiable.

Suppose there is an assignment $S : Var(C_\mathcal{X}) \to \{0, 1\}$, which evaluates $C_\mathcal{X}$ true. Let $\mathcal{X}_0$ and $\mathcal{X}_1$ be defined in the following way:

$$\mathcal{X}_0 = \{s_j \mid S(x_j) = 0\},$$

$$\mathcal{X}_1 = \mathcal{X} \setminus \mathcal{X}_0.$$ 

Clearly, it holds that $\mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_1$. Suppose $s_i, s_j \in \mathcal{X}_1$. Since $S$ was assumed to satisfy $C_\mathcal{X}$, it holds that $(x_i \lor x_j)$ cannot be a clause in $C_\mathcal{X}$. Thus $\{s_i, s_j\} \not\models \phi_0$ holds by the construction of $C_\mathcal{X}$. By 2-coherence of $\phi_0$, it follows that $\mathcal{X}_0 \models \phi_0$ holds. Suppose $s_i, s_j \in \mathcal{X}_1$. Again, since $S$ was assumed to satisfy $C_\mathcal{X}$, it holds that $(\neg x_i \lor \neg x_j)$ cannot be a clause in $C_\mathcal{X}$. Thus $\{s_i, s_j\} \not\models \phi_1$ holds by the construction of $C_\mathcal{X}$. Again, by 2-coherence of $\phi_1$, it follows that $\mathcal{X}_1 \models \phi_1$ holds. Thus $\mathcal{X} \models (\phi_0 \lor \phi_1)$.

The other direction: Suppose $\mathcal{X} \models (\phi_0 \lor \phi_1)$. Then by Definition 2.0.10, there are $\mathcal{X}_0$ and $\mathcal{X}_1$, such that $\mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_1$ and $\mathcal{X}_i \models \phi_i$, for $i \in \{0, 1\}$. Let $S : Var(C_\mathcal{X}) \to \{0, 1\}$ be defined in the following way:

- $S(x_i) = 0$, if $s_i \in \mathcal{X}_0$.
- $S(x_i) = 1$, if $s_i \in \mathcal{X}_1$. 


Clearly, $S$ is well defined. Let us show that $S$ satisfies $C_X$. Suppose $(x_i \lor x_j)$ is a clause in $C_X$. Then $\{s_i, s_j\}$ fails $\phi_0$ by the construction of $C_X$. Thus $s_i$ and $s_j$ cannot be both in $X_0$. Thus $x_i$ or $x_j$ must be in $X_1$. Thus $S(x_i) = 1$ or $S(x_j) = 1$. In both cases, it holds that $S(x_i \lor x_j) = 1$.

Suppose $(\neg x_i \lor \neg x_j)$ is a clause in $C_X$. Then by the construction of $C_X$, it follows that $\{s_i, s_j\}$ fails $\phi_1$. Then $s_i$ and $s_j$ cannot be both in $X_1$. Thus $x_i$ or $x_j$ must be in $X_0$. Thus $S(x_i) = 0$ or $S(x_j) = 0$. In both cases, it holds that $S(\neg x_i \lor \neg x_j) = 1$.

Last, the complexity of the reduction is in $C$: We go through the 2-element subsets of the team $X$, which can be done in LOGSPACE. For each subset $\{s, s'\}$ we check if they fail $\phi_0$ or $\phi_1$. Since we assumed that $M$ was closed under sub-teams, especially the 2-element sub-teams and that $MC(M, \Phi) \in C$, $\{s, s'\} \models \phi_i$ can be decide in $C$ for each 2-element sub-team and for each $\phi_i$, $i \leq 2$.

The following corollary states that if the combined complexity of the model checking for a pair $MC(M, \Phi)$ is already as high as NL, then allowing one disjunction over the formulas of $\Phi$ does not increase the combined complexity of the model checking.

**Corollary 4.4.3.** Suppose $M$ is a set of teams closed under sub-teams, $\Phi$ is a class of 2-coherent formulas, $C$ is a complexity class, such that $NL \subseteq C$ and $MC(M, \Phi) \in C$. Then $D_2(M, \Phi) \in C$.

### 4.4.1 Dependence atoms

We will consider the combined complexity of the model checking for dependence atoms over finite teams. We will show that the combined complexity of disjunctions of conjunctions of dependence atoms becomes NP-complete for disjunctions larger than 2. Note, that we consider disjunctions of a single dependence atom. Thus all the formulas considered in this section are coherent. It is essential to the reduction that we have unbounded number of variables in use.

**Definition 4.4.4.** Let $M_n^k$ be the set of all teams with domain $\{x_1, \ldots, x_k\}$ and range $\{1, \ldots, n\}$. Let

$$\mathcal{M} = \bigcup_{k, n \in \mathbb{N}} M_n^k.$$ 

Let $\mathcal{T}$ be the set of all finite conjunctions of dependence atoms over variables $\{x_i \mid i \in \mathbb{N}\}$.

**Theorem 4.4.5.** $MC(\mathcal{M}, \mathcal{T}) \in LOGSPACE$.

**Proof.** Suppose we are given a team $X$ and some finite conjunction of dependence atoms $\bigwedge_{i \in m} = (X_i, y_i)$. We use $log(n)$ many memory space as a counter to go through all the conjuncts of the formula. We will check for each $i \in m$ whether
4.4. Combined complexity of dependence logic formulas over finite teams

$\mathcal{X} \models = (X_i, y_i)$. If this holds, we increase the counter to $i+1$ and check again $\mathcal{X} \models = (X_{i+1}, y_{i+1})$. We proceed this way until some of the conjuncts fail or the counter reaches $m$. If the counter reaches $m$, then all the conjuncts are satisfied and the machine accepts the input. If some conjunct is failed, the machine halts, and does not accept the input.

Since the size of the conjunction $m$ is always smaller than the input $n$ (the formula is part of the input), $\log(n)$ memory space is enough to go through all the conjuncts. The model checking for a single dependence atom can be checked in LOGSPACE by Theorem 4.2.3. The algorithm checking for a dependence in a relation is the same for all dependence atoms, we just check from different places (for different variables) with different dependence atoms. Thus, $MC(\mathcal{M}, T)$ is in LOGSPACE. 

Next we restrict the problem $D_k(\mathcal{M}, T)$ so that we only consider disjunction over a single formula. We denote this problem with $D^*_k(\mathcal{M}, T)$. Thus the the problem is to decide if $\mathcal{X} \models \phi \vee \phi$ for given $\mathcal{X} \in \mathcal{M}$ and $\phi \in T$. We will show that $k - \text{COL}$ can be reduced to $D^*_k(\mathcal{M}, T)$

**Theorem 4.4.6.** $k - \text{COL} \leq \text{LOGSPACE} D^*_k(\mathcal{M}, T)$.

**Proof.** Given a graph $G = (V, E)$ we construct a team $\mathcal{X}_G$ and a conjunction of dependence atoms $\phi$ in the following way:

- For each $v \in V$ we add an assignment $s_v \in \mathcal{X}_G$.

- Suppose there is some order of the vertices of $E$. For the $n$:th pair $(v_i, v_j) \in E$ we add a new conjunct $=(x_{i_n}, x_{j_n})$ to $\phi$, and extend all the assignments in $\mathcal{X}_G$ with variables $x_{i_n}$ and $x_{j_n}$. We let $s_v(x_{i_n}) = s_w(x_{i_n})$ and $s_v(x_{j_n}) \neq s_w(x_{j_n})$, thus we assign values in a way that $\{s_v, s_w\} \neq = (x_{i_n}, x_{j_n})$. The values for $s_v(x)$ and $s_w(x)$ for all other variables $x$ are set to 0.

- Every time we process an edge $(v, w)$ of the relation $E$, we use new variables and assign new values (excluding 0), which have not been used before in the construction.

We will show that the following two conditions are equivalent:

1. $G$ is $k$-colorable.
2. $\mathcal{X}_G \models \bigvee_k \phi$.

Suppose $G$ is $k$-colorable. Then, there is a function $\xi : V \to \{1, \ldots, k\}$, such that if $(v, w) \in E$, then $\xi(v) \neq \xi(w)$. Let the partition of $\mathcal{X}$ into $k$ subsets $\mathcal{X}_i$, $i \leq k$ be defined in the following way:

$\mathcal{X}_i = \{s_v \mid \xi(v) = i\}$. 

One can observe that \( X_G = \bigcup_i X_i \) holds. Next we will show that \( X_i \models \phi \) holds for all \( X_i, i \leq k \).

Suppose \( s_v, s_w \in X_i \) and \( = (x, y) \) is a conjunct in \( \phi \). Suppose \( = (x, y) \) was generated because \( (v, w) \in E \). Then \( s_v(x) = s_w(x) \) and \( s_v(y) \neq s_w(y) \), which means that \( \{s_v, s_w\} \not\models \phi \), which is a contradiction with the assumption. Thus \( = (x, y) \) was not generated because \( (v, w) \in E \). Then, by the construction of \( X_G \), it holds that \( s_v(x) = s_v(y) = s_w(x) = s_w(y) = 0 \). Then it holds that \( \{s_i, s_j\} \not\models = (x, y) \).

The other direction: Suppose \( X_G \models \bigvee_k \phi \) holds. Then by the definition of disjunction it follows that there are sets \( X_i, i \leq k \), such that \( X_G = \bigcup_i X_i \) and \( X_i \models \phi \) for all \( i, i \leq k \). Let \( \xi \) be defined in the following way:

- \( \xi(v) = i \), if \( s_v \in X_i \).

Clearly, \( \xi : Var(X) \to \mathbb{N} \) is well defined. Suppose \( (v, w) \in E \). Then, by the construction of \( X_G \) there is a conjunct in \( = (x, y) \) in \( \phi \), such that \( \{s_v, s_w\} \not\models = (x, y) \). Then \( s_v \) and \( s_w \) cannot be in the same set \( X_i \), thus \( \xi(v) \neq \xi(w) \).

We also show the other direction, i.e. \( D^*_k(M, T) \) can be reduced to \( k - \text{COL} \).

**Theorem 4.4.7.** \( D^*_k(M, T) \) \( \leq \text{LOGSPACE} \ k - \text{COL} \).

**Proof.** Given a team \( X \) and a \( k \)-disjunction of conjunctions of dependence atoms \( \bigvee_k \bigwedge_{i \in m} = (X_i, y_i) \) we create a graph \( G_X = (V, E) \) in the following way:

- For each \( s_i \in X \) we add \( v_i \in V \).
- If \( \{s_i, s_j\} \) fails one of the dependence atoms \( = (X_n, y_n) \), we add the pairs \( (v_i, v_j) \) and \( (v_j, v_i) \) into \( E \).

Now, the following are equivalent:

1. \( G_X \) is \( k \)-colorable.
2. \( X \models \bigvee_{i \in k} \phi_i \).

The proof is analogous to the proof of Theorem 4.4.6.

**Corollary 4.4.8.** \( D^*_k(M, T) \) is \( \text{NP-complete} \) for \( k > 2 \).