Advanced Topics in Computational Complexity

Jonni Virtema

11.1.2016

This is a handout for the course Advanced Topics in Computational Complexity held in the Winter term 2015 in the Leibniz Universität Hannover.

Course Details

In the course we survey computational complexity of various propositional and modal logics. We concentrate on the recently developed logics for which the semantics is defined via *team semantics*. We will introduce these new logics and prove some fundamental results related to their computational complexity and expressive power. For example, we study the related model checking, satisfiability, and validity problems, and prove related complexity results.

Prerequisites:

- Basics of discrete math and naive set theory. Sets, relations, etc.
- Some previous knowledge on logics (propositional logic, modal logic, or first-order logic). Induction on the structure of formulae.
- Some knowledge about computational complexity. Basic complexity classes P, NP, PSPACE, etc. Some complete problems for these classes 3SAT, QBF, etc.

Content of the course:

- Flexible depending on the preliminaries that are needed and how much time things take.
- Introduction to logical modeling of things: Expressive power and computational complexity. Satisfiability, validity, model checking.
- First-order logic and second-order logic
- Introduction to first-order dependence logic.
- Modal logics with team semantics.
  - Syntax, semantics, and basic properties.
  - Dependence atoms and inclusion atoms.
  - Computational complexity and expressive power.
- Propositional logics with team semantics.
  - Syntax, semantics, and basic properties.
  - Dependence atoms and inclusion atoms.
  - Computational complexity and expressive power.

Practicalities:

- Lectures on Thursday 15.00–16.45 (15 minute brake)
- Lectures + Exercises on Monday 10.15–12.00 (15 minute break?), about 50% lecture, 50% exercise
- Exercises for the following Monday are given on the previous Thursday at the latest.
- Last lecture maybe on Monday 21.12.2015
- Oral exam at some point early 2016 (in principle only one possibility for the exam)
- Questions: Room 226 or virtema@thi.uni-hannover.de

Course material:

- This handout (will evolve during the course and is not complete)
- Examples, proofs, etc. on the white board
- Survey article: **Expressivity and Complexity of Dependence Logic** by Arnaud Durand, Juha Kontinen, and Heribert Vollmer. (available online).
- Book: **Dependence Logic – A new Approach to Independence Friendly Logic** by Jouko Väänänen. (online draft)
- Chapter: **Modal dependence logic**, in: New Perspectives on Games and Interaction, Krzysztof Apt, Robert van Rooij (eds.) Texts in Logic and Games, vol 5 Amsterdam University Press, 2008, 237-254. By Jouko Väänänen. (available online)

1 Logical modelling

The most fundamental concepts in logic are models and formulae. Models can be thought as abstractions of systems and formulae can be seen as abstractions of properties of these systems. Logic can then be used, for example, to formalize deduction and query answering related to real life systems.

Examples:

Structures abstract	Formulas abstract	Example logics
Networks (e.g., transport)	Properties of the network (e.g., route from Hannover to Berlin)	FO, MSO
Programs	Spesification of the program (e.g., fairness properities, no deadlocks)	CTL, μ-calculus, PDL
Hardware	Spesification of the hardware	CTL, μ-calculus, PDL

In applications the goal is to automatize task!

1. Construct a mathematical model of your system.
2. Translate desired properties to some mathematical language.
3. Use some pre-made machinery to see whether the desired properties hold in your system.

2 Expressive power and computational complexity

Two of the most fundamental aspects of any logic are its expressive power and computational complexity, i.e., what properties can be expressed with a given logic and how much resources, i.e., time and space, is needed to verify whether a described property holds or not. An ideal logic would of course be at the same time highly expressive and still computationally feasible. Unfortunately these concepts seem to be the opposite sides of the same coin. In most cases a more expressive logic also means a computationally more complex logic. And indeed, if the expressive power of a logic is great enough, many questions concerning the logic become even undecidable.

2.1 Expressive power and definability

The most common way to study the expressive power of a given logic is to compare it to the expressive power of other known logics. Almost every logic can be seen as fragments of second-order logic, whereas most modal logics are fragments of first-order logic. All the logics studied in this course can be seen as fragments of existential second-order logic. Thus we are interested in the question which fragment of existential second-order logic a given logic captures. In addition to relative expressive power of logics we are of course interested in the concrete expressive power of a given logic, i.e., we are interested in the question "Which properties can be defined in a given logic?".

2.2 Decidability and complexity

Undoubtedly the proliferation of computers and the rise of computer science in general in the end of the 20th century has had a seminal effect for the importance of the study of decidability and computational complexity, also in the field of logic. In modern computer science logic is used, among other things, to model and to verify existing real life systems, in automatic reasoning and, for example, in handling queries in huge medical databases. Hence the need for computationally feasible logics is now greater than ever.

We consider the decidability and complexity of three essential problems concerning any logic. The satisfiability problem, the validity problem, and the model checking problem. The **satisfiability problem** concentrates on the questions like: "Given a specification of a system is it possible to realize that specification?". In logical terms this translates to: "Given a sentence of some logic does there exist a model that satisfies it?".

The **validity problem** deals with problems of the sort: "Given a specification decide whether it always holds?". In logical terms this translates to: "Given a sentence of some logic, does every model satisfy the sentence?".

The **model checking problem** deals with problems of the sort: "Given a system and a specification, does the system satisfy the specification?". In logical terms this translates to: "Given a sentence of some logic and a model, does the model satisfy the sentence?". The model checking problem has two inputs, a finite model and a formula.

Hence we can identify three different complexity measures for this problem. We can fix the model, the formula or neither.

3 Relations and structures

A vocabulary, denoted by symbols τ and σ, is a collection of constant symbols, and function and relation symbols with prescribed arities. A vocabulary is *relational* if it does not contain any function symbols, i.e., a vocabulary is relational if it contains only constant symbols and relation symbols. We mainly considered only relational vocabularies. We denote constant symbols by $c_1, c_2, \dots$, relation symbols by $R_1, R_2, \dots$, and function symbols by $f_1, f_2, \dots$.

Definition 1. *A τ-structure is a tuple*

$$\mathfrak{A} = \big(A, (c_i^{\mathfrak{A}})_{c_i \in \tau}, (R_i^{\mathfrak{A}})_{R_i \in \tau}, (f_i^{\mathfrak{A}})_{R_i \in \tau}\big),$$

where A, called the domain of the structure, is a nonempty set and where

- *the interpretation $c_i^{\mathfrak{A}}$ of each constant symbol $c_i \in \tau$ is an element from the set A and*
- *the interpretation $R_i^{\mathfrak{A}}$ of each relation symbol $R_i \in \tau$ with arity k is a k-ary relation on the set A, i.e., a subset of A^k.*
- *the interpretation $f_i^{\mathfrak{A}}$ of each function symbol $f_i \in \tau$ with arity k is a funtion with domain A^k and co-domain A.*

We denote structures by uppercase letters of the typeface Fraktur, e.g $\mathfrak{A}$ and $\mathfrak{B}$, and the corresponding domains by the corresponding uppercase letters of the typeface Latin, e.g. A and B. Classes of structures are usually denoted by uppercase letters of a calligraphic font, e.g. $\mathcal{C}$ and $\mathcal{D}$. A structure $\mathfrak{A}$ is finite if A is a finite set. When τ is a vocabulary and $\mathcal{L}$ some logic, we denote by $\mathcal{L}(\tau)$ the set of all $\mathcal{L}$ formulae with vocabulary τ. When the vocabulary is clear from the context, we often write $\mathcal{L}$ instead of $\mathcal{L}(\tau)$. The lowercase letters x, y, z, and $\{x_i\}_{i\in\mathbb{N}}$ are reserved for first-order variables, while the uppercase letter X, Y, Z, and $\{X_i\}_{i\in\mathbb{N}}$ are used for second-order variables, i.e., relation variables. We sometimes use $\boldsymbol{x}$ and $\boldsymbol{X}$ to denote a tuple of variables of the corresponding type and of finite length.

An *assignment* over a model $\mathfrak{A}$ is a finite function that maps first-order variables to elements of A and a k-ary second-order variables to subsets of A^k. In the following, we denote by χ a variable that is either a first-order variable or a relation variable. If s is an assignment, x a first-order variable and $a \in A$ then $s(a/x)$ denotes the assignment with domain $\mathrm{dom}(s) \cup \{x\}$ such that

$$s(a/x)(\chi) = \begin{cases} a & \text{if } \chi = x, \\ s(\chi) & \text{otherwise.} \end{cases}$$

Analogously, if s is an assignment, X an k-ary second-order variable and $B \subseteq A^k$ then $s(B/X)$ denotes the assignment with domain $\mathrm{dom}(s) \cup \{X\}$ such that

$$s(B/X)(\chi) = \begin{cases} B & \text{if } \chi = X, \\ s(\chi) & \text{otherwise.} \end{cases}$$

For the modified assignment $s(a_1/x_1)(a_2/x_2)\ldots(a_n/x_n)$ we use the shorthand notations $s(a_1/x_1, \ldots, a_n/x_n)$ and $s(\boldsymbol{a}/\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{a} = (a_1, \ldots, a_n)$. If s is an assignment and V a set of variables, we denote by $s \upharpoonright V$ the assignment with domain $\mathrm{dom}(s) \cap V$ that agrees with s.

4 First-order and second-order logic

In this section we recall the definitions of first-order logic and second-order logic.

The set of τ-terms is defined as follows:

1. Every first-order variable x is a τ-term.
2. If $t_1, \ldots, t_n$ are τ-terms and $f \in \tau$ is an n-ary function symbol, then $f(t_1, \ldots, t_n)$ is a τ-term.

An assignment s assigns a value $t^{\mathfrak{A}}\langle s\rangle$ for every τ-term t in a model $\mathfrak{A}$ as follows:

- for a contant symbol $c \in \tau$: $c^{\mathfrak{A}}\langle s\rangle := c^{\mathfrak{A}}$
- for a first-order variable x: $x^{\mathfrak{A}}\langle s\rangle := s(x)$
- for an k-ary function symbol $f \in \tau$: $f(t_1, \ldots, t_k)^{\mathfrak{A}}\langle s\rangle := f^{\mathfrak{A}}(t_1^{\mathfrak{A}}\langle s\rangle, \ldots, t_k^{\mathfrak{A}}\langle s\rangle)$, where $t_1, \ldots, t_k$ are τ-terms

The syntax for first-order logic on a vocabulary τ, $\mathrm{FO}(\tau)$, is defined by the following grammar.

$$\varphi ::= t_1 = t_2 \mid R(t_1, \ldots, t_n) \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \exists x\varphi \mid \forall x\varphi,$$

where $R \in \tau$ is an n-ary relation symbol and $t_1, \ldots, t_n$ are τ-terms. The formulae $\varphi \rightarrow \psi$ and $\varphi \leftrightarrow \psi$ are shorthand notations for $(\neg\varphi \vee \psi)$ and $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$, respectively. The syntax for second-order logic, $SO(\tau)$, extends the syntax for first-order logic by quantification of second-order variables, i.e., relation variables. In addition to the above rules for first-order logic, we have the following three rules, for each $n \in \mathbb{N}$:

$$\varphi \quad ::= \quad X(t_1, \ldots, t_n) \mid \exists X\varphi \mid \forall X\varphi,$$

where X is an n-ary relation variable.

Existential second-order logic is the fragment of second-order logic of the form

$$\exists X_1 \ldots \exists X_n\, \varphi,$$

where φ is a formula of first-order logic (enriched with second-order variable symbols).

The set of *free variables* of an SO (or FO) formula φ, $\mathrm{fr}(\varphi)$, is defined inductively as follows.

$$\begin{aligned}
\mathrm{fr}(x) &:= \{x\}\\
\mathrm{fr}(f(t_1,\dots,t_n)) &:= \mathrm{fr}(t_1)\cup\dots\cup\mathrm{fr}(t_n)\\
\mathrm{fr}(t_1=t_2) &:= \mathrm{fr}(t_1)\cup\mathrm{fr}(t_2),\\
\mathrm{fr}\big(R(t_1,\dots,t_n)\big) &:= fr(t_1)\cup\dots\cup\mathrm{fr}(t_n),\\
\mathrm{fr}\big(X(t_1,\dots,t_n)\big) &:= \{X\}\cup\mathrm{fr}(t_1)\cup\dots\cup\mathrm{fr}(t_n),\\
\mathrm{fr}(\neg\varphi) &:= \mathrm{fr}(\varphi),\\
\mathrm{fr}\big((\varphi\wedge\psi)\big) &:= \mathrm{fr}(\varphi)\cup\mathrm{fr}(\psi),\\
\mathrm{fr}\big((\varphi\vee\psi)\big) &:= \mathrm{fr}(\varphi)\cup\mathrm{fr}(\psi),\\
\mathrm{fr}(\exists x\varphi) &:= \mathrm{fr}(\varphi)\setminus\{x\},\\
\mathrm{fr}(\forall x\varphi) &:= \mathrm{fr}(\varphi)\setminus\{x\},\\
\mathrm{fr}(\exists X\varphi) &:= \mathrm{fr}(\varphi)\setminus\{X\},\\
\mathrm{fr}(\forall X\varphi) &:= \mathrm{fr}(\varphi)\setminus\{X\}.
\end{aligned}$$

A formula φ is a *sentence* if $\mathrm{fr}(\varphi)=\emptyset$

The semantics for first-order logic and second-order logic is defined by the following clauses. Note that when stating $\mathfrak{A},s\models\varphi$, we always assume that $\mathrm{fr}(\varphi)\subseteq\mathrm{dom}(s)$.

$$\begin{aligned}
\mathfrak{A},s\models t_1=t_2 &\quad\text{iff}\quad t_1^{\mathfrak{A}}\langle s\rangle=t_2^{\mathfrak{A}}\langle s\rangle.\\
\mathfrak{A},s\models R(t_1,\dots,t_n) &\quad\text{iff}\quad \big(t_1^{\mathfrak{A}}\langle s\rangle,\dots,t_n^{\mathfrak{A}}\langle s\rangle\big)\in R^{\mathfrak{A}}.\\
\mathfrak{A},s\models\neg\varphi &\quad\text{iff}\quad \mathfrak{A},s\not\models\varphi.\\
\mathfrak{A},s\models(\varphi\wedge\psi) &\quad\text{iff}\quad \mathfrak{A},s\models\varphi\text{ and }\mathfrak{A},s\models\psi.\\
\mathfrak{A},s\models(\varphi\vee\psi) &\quad\text{iff}\quad \mathfrak{A},s\models\varphi\text{ or }\mathfrak{A},s\models\psi.\\
\mathfrak{A},s\models\exists x\varphi &\quad\text{iff}\quad \mathfrak{A},s(a/x)\models\varphi\text{ for some }a\in A.\\
\mathfrak{A},s\models\forall x\varphi &\quad\text{iff}\quad \mathfrak{A},s(a/x)\models\varphi\text{ for all }a\in A.
\end{aligned}$$

For second-order logic we also have the following additional cases.

$$\begin{aligned}
\mathfrak{A},s\models X(t_1,\dots,t_n) &\quad\text{iff}\quad \big(t_1^{\mathfrak{A}}\langle s\rangle,\dots,t_n^{\mathfrak{A}}\langle s\rangle\big)\in s(X).\\
\mathfrak{A},s\models\exists X\varphi &\quad\text{iff}\quad \mathfrak{A},s(B/X)\models\varphi\text{ for some }B\subseteq A^k,\\
&\qquad\quad\text{where }k\text{ is the arity of the relation variable }X.\\
\mathfrak{A},s\models\forall X\varphi &\quad\text{iff}\quad \mathfrak{A},s(B/X)\models\varphi\text{ for all }B\subseteq A^k,\\
&\qquad\quad\text{where }k\text{ is the arity of the relation variable }X.
\end{aligned}$$

We sometimes use $\models_{\mathrm{FO}}$ to denote the satisfaction relation of first-order logic.

4.1 Expressive power formally

Definition 2. *Let $\varphi,\psi\in\mathrm{FO}(\tau)$ (or in any of the logics defined above). We say that φ and ψ are equivalent if for every τ-structure $\mathfrak{A}$ and every assignment s the following* *holds:*

$$\mathfrak{A}, s \models \varphi \quad \Leftrightarrow \quad \mathfrak{A}, s \models \psi.$$

Definition 3. *A formula $\varphi \in \mathrm{SO}(\tau)$ is in* negation normal form *if negation symbols that occurs in φ directly precede an atomic formula (i.e., $t_1 = t_2$ or $R(t_1, \ldots, R_n)$)*

Lemma 1. *For every formula $\varphi \in \mathrm{SO}(\tau)$ ($\mathrm{FO}(\tau)$, resp.) there exists an equivalent formula in $\varphi \in \mathrm{SO}(\tau)$ ($\mathrm{FO}(\tau)$, resp.) that is in negation normal form.*

Proof. Easy proof by induction.

We write $\mathfrak{A} \models \varphi$ if and only if $\mathfrak{A}, s \models \varphi$ holds for every assignment s. Every sentence defines the class of structure in which the sentence is true.

Definition 4. *Let $\varphi \in \mathrm{FO}(\tau)$ and let $\mathcal{C}$ be a class of τ-structures. We say that the formula φ* defines *the class of structure $\mathcal{C}$ if and only if*

$$\{\mathfrak{A} \mid \mathfrak{A} \models \varphi\} = C.$$

We say that $\mathcal{C}$ is definable in $\mathrm{FO}(\tau)$ if and only if there exists a sentence in $\mathrm{FO}(\tau)$ that defines $\mathcal{C}$.

Definition 5. *Let Γ be a set of $\mathrm{FO}(\tau)$-formulae and let $\mathcal{C}$ be a class of τ-structures. We say that the formula Γ* defines *the class of structure $\mathcal{C}$ if and only if*

$$\{\mathfrak{A} \mid \textit{for every } \varphi \in \Gamma\text{: } \mathfrak{A} \models \varphi\} = C.$$

We say that $\mathcal{C}$ is definable by a set of $\mathrm{FO}(\tau) - formulae$ if and only if there exists a set $\mathrm{FO}(\tau)$-formulae that defines $\mathcal{C}$.

4.2 Computational complexity more formally

Definition 6. *Let $\mathcal{K}$ be a complexity class, $\mathcal{L}$ a logic and $enc : \mathcal{L} \to \{0,1\}^*$ some efficient encoding that maps $\mathcal{L}$-formulae to binary strings. We say that* the satisfiability problem *of $\mathcal{L}$ is in $\mathcal{K}$ (is $\mathcal{K}$-hard) if the language*

$$\{enc(\varphi) \mid \textit{there exists a model } \mathfrak{A} \textit{ such that } \mathfrak{A} \models \varphi\}$$

belongs in $\mathcal{K}$ (is $\mathcal{K}$-hard).

Theorem 1. *The satisfiability problem for*

- *first-order logic is undecidable (Π^0_1-complete)*
- *existential second-order logic is undecidable (Π^0_1-complete)*
- *three variable fragment of first-order logic is undecidable (Π^0_1-complete)*
- *two variable fragment of first-order logic is* NEXPTIME*-complete (relational vocabulary)*

Definition 7. *Let $\mathcal{K}$ be a complexity class, $\mathcal{L}(\tau)$ a logic and enc : $\mathcal{L} \to \{0,1\}^*$ some efficient encoding that maps $\mathcal{L}$-formulae to binary strings. We say that* the validity problem *of $\mathcal{L}$ is in $\mathcal{K}$ (is $\mathcal{K}$-hard) if the language*

$$\{enc(\varphi) \mid \mathfrak{A} \models \varphi \text{ holds for every model } \mathfrak{A}\}$$

belongs in $\mathcal{K}$ (is $\mathcal{K}$-hard).

Remember that a problem is $\mathcal{K}$-*complete* if it is both in $\mathcal{K}$ and $\mathcal{K}$-hard. For a logic $\mathcal{L}$, we denote by $\mathsf{SAT}(\mathcal{L})$ and $\mathsf{VAL}(\mathcal{L})$ the satisfiability problem and the validity problem of $\mathcal{L}$, respectively.

Theorem 2. *The validity problem for*

- *first-order logic is undecidable (Σ^0_1-complete)*
- *existential second-order logic is undecidable (Σ^0_1-complete)*
- *three variable fragment of first-order logic is undecidable (Σ^0_1-complete)*
- *two variable fragment of first-order logic is co*$\mathsf{NEXPTIME}$*-complete*

In the definition below $\mathfrak{A}$ is always a finite model.

Definition 8. *Let $\mathcal{K}$ be a complexity class, $\mathcal{L}$ a logic and and enc some efficient encoding that maps $\mathcal{L}$-formulae and finite models to binary strings.*

- *The data complexity of $\mathcal{L}$ is in $\mathcal{K}$ if for every sentence φ of $\mathcal{L}$ the language*

  $$\{enc(\mathfrak{A}) \mid \mathfrak{A} \models \varphi\}$$

  belongs to $\mathcal{K}$, and it is $\mathcal{K}$-hard if there exists a sentence φ such that

  $$\{enc(\mathfrak{A}) \mid \mathfrak{A} \models \varphi\}$$

  is a $\mathcal{K}$-hard problem.
- *The expression complexity of $\mathcal{L}$ is in $\mathcal{K}$ if for every finite structure $\mathfrak{A}$ the language*

  $$\{enc(\varphi) \mid \mathfrak{A} \models \varphi\}$$

  belongs to $\mathcal{K}$, and it is $\mathcal{K}$-hard if there exists a finite model $\mathfrak{A}$ such that

  $$\{enc(\varphi) \mid \mathfrak{A} \models \varphi\}$$

  is a $\mathcal{K}$-hard problem.
- *The combined complexity of $\mathcal{L}$ is in $\mathcal{K}$ (is K-hard) if the language*

  $$\{\big(enc(\mathfrak{A}), enc(\varphi)\big) \mid \mathfrak{A} \models \varphi\}$$

  belongs to $\mathcal{K}$ (is $\mathcal{K}$-hard).

Theorem 3. *The model checking problem for*

- *first-order logic is* PSPACE*-complete for combined complexity*
- *first-order logic is* PSPACE*-complete for expression complexity*
- *first-order logic is in* P *for data complexity (in* LOGSPACE*)*

Algorithm 1 APTIME algorithm for MC(FO)

```
function MC($\mathfrak{A}, s, \varphi, I$)
    if $\varphi = \psi_1 \wedge \psi_2$ then
        if $I = 1$ then
            universally choose $i \in \{1, 2\}$
            return MC($\mathfrak{A}, \psi_i, s, I$)
        else if $I = 0$ then
            existentially choose $i \in \{1, 2\}$
            return MC($\mathfrak{A}, \psi_i, s, I$)
    else if $\varphi = \psi_1 \vee \psi_2$ then
        if $I = 1$ then
            existentially choose $i \in \{1, 2\}$
            return MC($\mathfrak{A}, \psi_i, s, I$)
        else if $I = 0$ then
            universally choose $i \in \{1, 2\}$
            return MC($\mathfrak{A}, \psi_i, s, I$)
    else if $\varphi = \neg\psi$ then
        if $I = 1$ then
            return MC($\mathfrak{A}, \psi, s, 0$)
        else if $I = 0$ then
            return MC($\mathfrak{A}, \psi, s, 1$)
    else if $\varphi = \exists x\psi$ then
        if $I = 1$ then
            existentially choose $a \in A$
            return MC($\mathfrak{A}, \psi, s(a/x), 1$)
        else if $I = 0$ then
            universally choose $a \in A$
            return MC($\mathfrak{A}, \psi, s(a/x), 0$)
    else if $\varphi = \forall x\psi$ then
        if $I = 1$ then
            universally choose $a \in A$
            return MC($\mathfrak{A}, \psi, s(a/x), 1$)
        else if $I = 0$ then
            existentially choose $a \in A$
            return MC($\mathfrak{A}, \psi, s(a/x), 0$)
    else if $\varphi = R(x_1, \ldots, x_n)$ then
        if $I = 1$ and $s((x_1), \ldots, s(x_n)) \in R^{\mathfrak{A}})$ then
            return true
        else if $I = 0$ and $(s(x_1), \ldots, s(x_n)) \notin R^{\mathfrak{A}}$ then
            return true
        else
            return false
    else if $\varphi = \neg R(x_1, \ldots, x_n)$ then
        if $I = 1$ and $s((x_1), \ldots, s(x_n)) \notin R^{\mathfrak{A}})$ then
            return true
        else if $I = 0$ and $(s(x_1), \ldots, s(x_n)) \in R^{\mathfrak{A}}$ then
            return true
        else
            return false
```

5 Dependence logic

Dependence logic is a new logical framework for formalising and studying various notions of dependence and independence that are important in many scientific disciplines such as experimental physics, social choice theory, computer science, and cryptography. Dependence logic extends first-order logic by dependence atoms

$$=\!(x_1, \ldots, x_n, y) \tag{1}$$

expressing that the value of the variable y is functionally determined on the values of $x_1, \ldots, x_n$. Satisfaction for formulas of dependence logic is defined using sets of assignments (*teams*) and not in terms of single assignments as in first-order logic.

Definition 9. *Let τ be a relational vocabulary. The syntax for dependence logic $\mathrm{D}(\tau)$ is defined by the following grammar:*

$$\begin{aligned} \varphi \quad ::= \quad & x_1 = x_2 \mid \neg\, x_1 = x_2 \mid R(x_1, \ldots, x_n) \mid \neg R(x_1, \ldots, x_n) \mid \\ & =\!(x_1, \ldots, x_n) \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \forall x \varphi \mid \exists x \varphi. \end{aligned}$$

The set of *free variables* $\mathrm{fr}(\varphi)$ of a D-formula φ is defined inductively as follows. The cases for atomic formulae, Boolean connectives and first-order quantifiers are defined as in first-order logic. For dependence atoms we have the additional clause

$$\mathrm{fr}\big(=\!(x_1, \ldots, x_n)\big) := \{x_1, \ldots, x_n\}.$$

As usual, we say that a formula φ is a sentence if $\mathrm{fr}(\varphi) = \emptyset$.

The semantics for dependence logic is defined via models and teams, i.e., sets of assignments. Let A be a set and $\{x_1, \ldots, x_n\}$ a finite (possibly empty) set of variables. A *team* X of A with the domain

$$\mathrm{dom}(X) = \{x_1, \ldots, x_n\}$$

is any set of assignments from $\mathrm{dom}(X)$ into A. If X is a team of $\mathfrak{A}$, B a set, and $F\colon X \to \mathcal{P}(A) \setminus \{\emptyset\}$ a function, we use

- $X(F/x)$ to denote the team $\{s(a/x) \mid s \in X, a \in F(s)\}$, and
- $X(B/x)$ to denote the team $\{s(b/x) \mid s \in X \text{ and } b \in B\}$.

In the following definitions we always assume that the domain of X contains $\mathrm{fr}(\varphi)$.

Definition 10. *Let $\mathfrak{A}$ be a model and X a team of A. The satisfaction relation $\mathfrak{A} \models_X \varphi$ for dependence logic is defined as follows:*

$$\begin{aligned}
\mathfrak{A} \models_X R(x_1,\ldots,x_n) &\text{ iff } \forall s \in X : \big(s(x_1),\ldots,s(x_n)\big) \in R^{\mathfrak{A}}.\\
\mathfrak{A} \models_X \neg R(x_1,\ldots,x_n) &\text{ iff } \forall s \in X : \big(s(x_1),\ldots,s(x_n)\big) \notin R^{\mathfrak{A}}.\\
\mathfrak{A} \models_X {=}(x_1,\ldots,x_n) &\text{ iff } \forall s,t \in X : s(x_1) = t(x_1),\ldots,s(x_{n-1}) = t(x_{n-1})\\
&\qquad \text{implies that } s(x_n) = t(x_n).\\
\mathfrak{A} \models_X \psi \wedge \varphi &\text{ iff } \mathfrak{A} \models_X \psi \text{ and } \mathfrak{A} \models_X \varphi.\\
\mathfrak{A} \models_X \psi \vee \varphi &\text{ iff } \mathfrak{A} \models_Y \psi \text{ and } \mathfrak{A} \models_Z \varphi,\\
&\qquad \text{for some } Y \text{ and } Z \text{ such that } Y \cup Z = X.\\
\mathfrak{A} \models_X \exists x\psi &\text{ iff } \mathfrak{A} \models_{X(F/x)} \psi \text{ for some function } F\colon X \to \mathcal{P}(A) \setminus \{\emptyset\}.\\
\mathfrak{A} \models_X \forall x\psi &\text{ iff } \mathfrak{A} \models_{X(A/x)} \psi.
\end{aligned}$$

Note 1. Above we gave the so-called lax semantics for the existential quantifier. However the following variant, called strict semantics, is equivalent with respect to team semantics in dependence logic:

We denote by $\exists_s$ variants of $\exists$ for which the semantics is defined as follows:

$$\mathfrak{A} \models_X \exists_s x\psi \quad \text{iff} \quad \mathfrak{A} \models_{X(F/x)} \psi \text{ for some function } F\colon X \to A.$$

For each formula φ of dependence logic, the following holds

$$\mathfrak{A} \models_X \exists_s x\varphi \Leftrightarrow \mathfrak{A} \models_X \exists x\varphi.$$

We say that a sentence φ of dependence logic is *true in a model* $\mathfrak{A}$, and write $\mathfrak{A} \models \varphi$, if $\mathfrak{A} \models_{\{\emptyset\}} \varphi$ holds.

The following proposition reveals that dependence logic is a conservative extension of first-order logic, i.e., the semantics of dependence logic coincide with that of first-order logic with regards to formulae that are syntactically first-order.

Proposition 1. *Let φ be a formula of dependence logic without dependence atoms, i.e., φ is syntactically a first-order formula. Then for every model $\mathfrak{A}$, team X of $\mathfrak{A}$ and assignment s of $\mathfrak{A}$:*

1. $\mathfrak{A} \models_{\{s\}} \varphi \quad \Leftrightarrow \quad \mathfrak{A}, s \models_{\mathrm{FO}} \varphi$.
2. $\mathfrak{A} \models_X \varphi \quad \Leftrightarrow \quad \mathfrak{A}, t \models_{\mathrm{FO}} \varphi$, *for every $t \in X$.*

Let X be a team with domain $\{x_1,\ldots,x_k\}$ and $V \subseteq \{x_1,\ldots,x_k\}$. We denote by $X \upharpoonright V$ the team

$$\{s \upharpoonright V \mid s \in X\}$$

with domain V. The following proposition shows that truth of a dependence logic formula depends only on the interpretations of the variables occurring free in the formula.

Proposition 2. *Let φ be any formula of dependence logic and $V \supseteq \mathrm{fr}(\varphi)$. Now*

$$\mathfrak{A} \models_X \varphi \quad \Leftrightarrow \quad \mathfrak{A} \models_{X \upharpoonright V} \varphi.$$

The following fact is a fundamental property of all formulae of dependence logic.

Proposition 3 (Downward closure). *Let φ be a formula of dependence logic, $\mathfrak{A}$ a model, and $Y \subseteq X$ teams of $\mathfrak{A}$. Then*

$$\mathfrak{A} \models_X \varphi \quad \Rightarrow \quad \mathfrak{A} \models_Y \varphi.$$

Theorem 4.

1. *The data complexity for* D *is* NP*-complete.*
2. *The expression complexity for* D *is* NEXPTIME*-complete.*
3. *The combined complexity of* D *is* NEXPTIME*-complete.*

6 Equivalence, flatness, and strict semantics

Definition 11. *Let φ and ψ be formulae of dependence logic. We say that ψ is a* logical consequence *of φ,*

$$\varphi \Rightarrow \psi,$$

if for all models $\mathfrak{A}$ and teams X with $\mathrm{dom}(X) \supseteq \mathrm{fr}(\varphi) \cup \mathrm{fr}(\psi)$ such that $\mathfrak{A} \models_X \varphi$ it holds that also $\mathfrak{A} \models_X \psi$. We say that φ is logically equivalent *with ψ if both $\varphi \Rightarrow \psi$ and $\psi \Rightarrow \varphi$ hold.*

Definition 12. *For logics L and L', we write $L \leq L'$ if for every sentence $\varphi \in L$ there exists an equivalent sentence $\varphi' \in L'$. We write $L \equiv L'$, if both $L \leq L'$ and $L' \leq L$ hold. We write $L < L'$, if $L \leq L'$ holds but $L' \leq L$ does not.*

Definition 13. *We say that a formula φ passes the* Flattness Test *if, for all $\mathfrak{A}$ and X,*

$$\mathfrak{A} \models_X \varphi \Leftrightarrow \forall s \in X : \mathfrak{A} \models_{\{s\}} \varphi.$$

Proposition 4. *Passing the Flattness Test is preserved by logical equivalence.*

Proposition 5. *Any formula of dependence logic that is equivalent to a first-order formula satisfies the Flattness Test.*

We now define the so-called strict variants of disjunction and existential quantifier for dependence logic.

Definition 14. *We denote by $\vee_s$ and $\exists_s$ variants of $\vee$ and $\exists$ for which the semantics is defined as follows:*

$$\begin{aligned} \mathfrak{A} \models_X \psi \vee_s \varphi \quad &\textit{iff} \quad \mathfrak{A} \models_Y \psi \textit{ and } \mathfrak{A} \models_Z \varphi, \\ &\qquad \textit{for some } Y \cap Z = \emptyset \textit{ such that } Y \cup Z = X. \\ \mathfrak{A} \models_X \exists_s x \psi \quad &\textit{iff} \quad \mathfrak{A} \models_{X(F/x)} \psi \textit{ for some function } F \colon X \to A. \end{aligned}$$

Proposition 6. *Let φ and ψ be formula of dependence logic. Then $\exists_s\varphi$ is logically equivalent with $\exists\varphi$, and $(\varphi \vee_s \psi)$ is logically equivalent with $(\varphi \vee \psi)$.*

Proof. Proof follows easily form downward closure.

In the next sections, we will use the strict semantics defined above for the existential quantifier $\exists$.

7 Finite variable logics

In this section we define k-variable fragments of first-order logic, first-order logic with counting, existential second-order logic, and dependence logic. Existential second-order logic, ESO, is the fragment of second-order logic for which the formulae are of the form

$$\exists X_1 \dots \exists X_n \varphi, \tag{2}$$

where $X_1 \dots X_n$ are second-order variables (i.e. relation variables) and φ is a formula of first-order logic. Monadic second-order logic, MSO, is the fragment of second-order logic in which all relation variables have arity 1 (i.e. all relation variables range over sets). Existential monadic second order logic, EMSO, is the fragment of MSO of the form (2).

We start by defining an expansion of first-order logic called first-order logic with counting. A counting quantifier is an expression of the form

$$\exists^{\geq i} x,$$

where x is a first-order variable and $i \in \mathbb{N}$. First-order logic with counting, denoted by FOC, is the extension of first-order logic with all counting quantifiers. In other words, the syntax of FOC extends the syntax of first-order logic by the clause

$$\varphi ::= \exists^{\geq i} x\, \varphi,$$

where x is a first-order variable and $i \in \mathbb{N}$. The semantics for FOC is defined in the same manner as for first-order logic with the following additional clause for counting quantifiers:

$$\mathfrak{A}, s \models \exists^{\geq i} x\, \varphi \quad \text{iff} \quad |\{a \in A \mid \mathfrak{A}, s(a/x) \models \varphi\}| \geq i.$$

Note that a counting quantifier

$$\exists^{\leq i} x$$

is regarded as a shorthand notation for the expression

$$\neg\exists^{\geq i+1} x.$$

The k-variable first-order logic, denoted by FO^k, is the syntactic fragment of first-order logic in which only variables form the set $\{x_1, \dots, x_k\}$ can appear in formulae. In the case $k = 2$ we denote these variables by x and y. The k-variable fragment FOC^k of FOC is defined analogously. The k-variable existential second-order logic, ESO^k, is the syntactic fragment of ESO in which only first-order variables from the set $\{x_1, \ldots, x_k\}$ can appear in formulae. Note that there is no restrictions concerning the usage of second-order variables. Also the alternative notation $\Sigma^1_1(\mathrm{FO}^k)$ is used for ESO^k. By $\Sigma^1_1(\mathrm{FOC}^k)$ we denote the extension of ESO^k with counting quantifiers for $x_1, \ldots, x_k$.

The syntax for k-variable dependence logic is given below.

Definition 15. *Let τ be a relational vocabulary. The set of formulae for* k-variable dependence logic $\mathrm{D}^k(\tau)$ *is defined by the following grammar:*

$$\begin{aligned}\varphi \quad ::= \quad & x_i = x_j \mid \neg\, x_i = x_j \mid R(x_{i_1}, \ldots, x_{i_n}) \mid \neg R(x_{i_1}, \ldots, x_{i_n}) \mid \\ & =\!(x_{i_1}, \ldots, x_{i_n}) \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \forall x \varphi \mid \exists x \varphi,\end{aligned}$$

where $R \in \tau$ is an n-ary relation symbol and $x_i, x_j, x_{i_1}, \ldots, x_{i_n} \in \{x_1, \ldots, x_k\}$.

The following self-evident proposition follows directly from Proposition 1.

Proposition 7. *For every $k \in \mathbb{N}$, $\mathrm{FO}^k \leq \mathrm{D}^k$.*

Proposition 8. *For each $k \in \mathbb{N}$, there is a sentence φ_k in D^1 such that*

$$\mathfrak{A} \models \varphi_k \quad \text{iff} \quad |A| \leq k.$$

Proof. For each $k \in \mathbb{N}$, let φ_k denote the formula

$$\forall x \,(\bigvee_{1 \leq i \leq k} =\!(x)).$$

The following equivalences hold.

$$\begin{aligned}& \mathfrak{A} \models \varphi_k \\ \Leftrightarrow \quad & \mathfrak{A} \models_{\{\emptyset\}(A/x)} \bigvee_{1 \leq i \leq k} =\!(x) \\ \Leftrightarrow \quad & \text{there exist teams } X_1, \ldots, X_k \text{ such that } X_1 \cup \cdots \cup X_k = \{\emptyset\}(A/x) \text{ and} \\ & \text{such that, for every } i \leq k,\ \mathfrak{A} \models_{X_i} =\!(x) \\ \Leftrightarrow \quad & \text{there exist teams } X_1, \ldots, X_k \text{ such that } X_1 \cup \cdots \cup X_k = \{\emptyset\}(A/x) \text{ and} \\ & \text{such that, for every } i \leq k,\ |X_i| \leq 1 \\ \Leftrightarrow \quad & |A| \leq k.\end{aligned}$$

Hence the claim follows.

As an immediate corollary of Proposition 8, we obtain the following result.

Corollary 1. *For each $k \in \mathbb{N}$ it holds that $\mathrm{D}^1 \not\leq \mathrm{FO}^k$.*

Proof. The expressive power of FO^k can be characterized by the the so-called k-pebble game. By using the k-pebble game, it is easy to show that on the empty vocabulary FO^k cannot differentiate a model of size k from a model of size $k+1$. Hence, the claim follows from Proposition 8.

Proposition 9. *Let τ be a finite unary vocabulary. Then the equivalence $\mathrm{MSO}(\tau) \equiv \mathrm{FO}(\tau)$ holds.*

Proposition 10. $\mathrm{D}^1 < \mathrm{FO}$.

Proof. By proof of Lemma 2 and by Theorem 5 from Section 8, we have that

$$\mathrm{D}^1 \leq \mathrm{EMSO}(\mathrm{FOC}^1).$$

Hence, it suffices to show that

$$\mathrm{EMSO}(\mathrm{FOC}^1) < \mathrm{FO}.$$

Without loss of generality, we can restrict our attention to structures of unary vocabulary. By Proposition 9, we have that $\mathrm{EMSO} \leq \mathrm{FO}$ on unary vocabularies. Furthermore, it is well known that $\mathrm{FO} \equiv \mathrm{FOC}$. Hence

$$\mathrm{EMSO}(\mathrm{FOC}^1) \leq \mathrm{FO}$$

follows. Now, since clearly nonemptiness of a binary relation is expressible in FO, but is not expressible in $\mathrm{EMSO}(\mathrm{FOC}^1)$, it follows that

$$\mathrm{EMSO}(\mathrm{FOC}^1) < \mathrm{FO}.$$

Therefore $\mathrm{D}^1 < \mathrm{FO}$.

We have seen that the expressive power of one-variable dependence logic is strictly below the expressive power of first-order logic. This however does not hold for the two-variable fragments. Next we will describe non-first-order properties that can be expressed in D^2.

Proposition 11. *The following properties can be expressed in D^2.*

1. *For unary relation symbols P and Q, we can express that $|P| \leq |Q|$ and hence that $|P| = |Q|$.*
2. *We can express that the model is infinite.*

Proof. For claim 1, we will show that a model $\mathfrak{A}$ satisfies the D^2 formula

$$\varphi := \forall x \exists y \Big({=}(y,x) \wedge \big(\neg P(x) \vee Q(y)\big)\Big)$$

if and only if $|P^{\mathfrak{A}}| \leq |Q^{\mathfrak{A}}|$. Notice first that, by the semantics of universal and existential quantifiers,

$$\mathfrak{A} \models \varphi$$

if and only if there exists a function $F : \{\emptyset\}(A/x) \to A$ such that

$$\mathfrak{A} \models_{\{\emptyset\}(A/x,F/y)} \Big({=}(y,x) \wedge \big(\neg P(x) \vee Q(y)\big)\Big).$$

This holds if and only if there exists an injective function $F : \{\emptyset\}(A/x) \to A$ such that

$$\mathfrak{A} \models_{\{\emptyset\}(A/x,F/y)} \big(\neg P(x) \vee Q(y)\big).$$

Moreover, this holds if and only if there exists an injective function $F\colon A \to A$ such that $F[P^{\mathfrak{A}}] \subseteq Q^{\mathfrak{A}}$. This, of course, is equivalent with the claim that $|P^{\mathfrak{A}}| \leq |Q^{\mathfrak{A}}|$.

For part 2, we use a similar idea as above. We will show that a model $\mathfrak{A}$ satisfies

$$\psi := \forall x \exists y \Big({=}(y,x) \wedge \exists x \big({=}(x) \wedge \neg x = y\big)\Big)$$

if and only is A is infinite. Again, notice that

$$\mathfrak{A} \models \psi$$

if and only if there exists a function $F : \{\emptyset\}(A/x) \to A$ such that

$$\mathfrak{A} \models_{\{\emptyset\}(A/x,F/y)} \Big({=}(y,x) \wedge \exists x \big({=}(x) \wedge \neg x = y\big)\Big).$$

This holds if and only if there exists an injective function $F : \{\emptyset\}(A/x) \to A$ such that

$$\mathfrak{A} \models_{\{\emptyset\}(A/x,F/y)} \exists x \big({=}(x) \wedge \neg x = y\big).$$

Furthermore, this is equivalent to the claim that there exists an injective function $F\colon A \to A$ and $a \in A$ such that $a \notin F[A]$. Clearly this is equivalent with the claim that A is infinite.

8 Fragments of existential second-order logic

In this section we study the relationship of finite variable dependence logics with fragments of existential second-order logic. We show that for each D^k-sentence there exists an equivalent sentence in $\Sigma^1_1(\mathrm{FOC}^k)$. This translation is optimal since clearly $\Sigma^1_1(\mathrm{FOC}^{k-1})$ does not suffice, e.g nonemptiness of a k-ary relation can be expressed in D^k but not in $\Sigma^1_1(\mathrm{FOC}^{k-1})$. Moreover we establish a translation into $\Sigma^1_1(\mathrm{FO}^{k+1})$.

We will first show how to translate D^k-sentences into equivalent sentences of $\Sigma^1_1(\mathrm{FOC}^k)$. We will then modify this translation to show how to translate D^k-sentences into equivalent sentences of $\Sigma^1_1(\mathrm{FO}^{k+1})$. Notice that the counting quantifier $\exists^{\leq 1}$ (or in fact the expression $\neg\exists^{\geq 2}$) is the only counting quantifier used in the translation below.

Lemma 2. *Assume that $k \geq 1$. Let τ be a relational vocabulary and let $R \notin \tau$ be a k-ary relation symbol. For every formula $\varphi \in \mathrm{D}^k(\tau)$ there exists a sentence*

$$\varphi^* \in \Sigma_1^1(\mathrm{FOC}^k)(\tau \cup \{R\})$$

such that for every model $\mathfrak{A}$ and team X of $\mathfrak{A}$ with $\mathrm{dom}(()\, X) = \{x_1, \ldots, x_k\}$

$$\mathfrak{A} \models_X \varphi \quad \textit{iff} \quad \big(\mathfrak{A}, \mathrm{rel}(X)\big) \models \varphi^*, \tag{3}$$

where $(\mathfrak{A}, \mathrm{rel}(X))$ is the expansion $\mathfrak{A}'$ of $\mathfrak{A}$ into vocabulary $\tau \cup \{R\}$ defined by

$$R^{\mathfrak{A}'} := \mathrm{rel}(X)\,^1.$$

Proof. Fix $k \geq 1$. We will define a translation

$$tr_k : \mathrm{D}^k(\tau) \mapsto \Sigma_1^1(\mathrm{FOC}^k)(\tau \cup \{R\})$$

inductively. Below we always assume that the quantified relations S and T are fresh, i.e., they are assumed not to appear in $tr_k(\psi)$ or $tr_k(\vartheta)$. Notice that for every D^k-formula φ

$$tr_k(\varphi) = \exists S_1 \ldots \exists S_n \varphi',$$

for some k-ary relation variables $S_1 \ldots S_n$, $n \in \mathbb{N}$, and for some syntactically FOC^k-formula φ'. The translation tr_k is defined as follows.

1. If φ is a literal then $tr_k(\varphi)$ is defined as

$$\forall x_1 \ldots \forall x_k \big(R(x_1, \ldots, x_k) \to \varphi(x_1, \ldots, x_k)\big).$$

2. Assume that φ is a dependence atom $=\!(\boldsymbol{x}, x_i)$, where $\boldsymbol{x} \in \{x_1, \ldots, x_k\}^t$ and $t \in \mathbb{N}$.
   (a) If x_i occurs in $\boldsymbol{x}$ then $tr_k(\varphi)$ is defined as

$$\exists x_1 (x_1 = x_1).$$

   (b) If x_i does not occur in $\boldsymbol{x}$ then $tr_k(\varphi)$ is defined as

$$\forall \boldsymbol{x}\, \exists^{\leq 1} x_i\, \exists \boldsymbol{z}\, R(x_1, \ldots, x_k),$$

   where $\boldsymbol{z}$ is exactly the sequence of variables in $\{x_1, \ldots, x_k\} \setminus \{x_i\}$ (in alphabetical order) that do not occur in $\boldsymbol{x}$.
3. Assume that $tr_k(\psi) = \exists S_1 \ldots \exists S_n \psi'$ and $tr_k(\vartheta) = \exists T_1 \ldots \exists T_m \vartheta'$, where ψ' and ϑ' are syntactically FOC^k-formulae. Furthermore, assume that $S_1, \ldots S_n, T_1, \ldots, T_m$ are all distinct.

[1] By $\mathrm{rel}(X)$, we mean the canonical representation of the team X as a relation, i.e., $(a_1, \ldots, a_n) \in \mathrm{rel}(X)$ if and only if $s \in X$, where $s(x_i) = a_i$, for every $1 \leq i \leq n$.

(a) If φ is of the form $(\psi \vee \vartheta)$, then $tr_k(\varphi)$ is defined as

$$\exists S \exists T \exists S_1 \ldots \exists S_n \exists T_1 \ldots \exists T_m \Big(\forall x_1 \ldots \forall x_k \Big(R(x_1, \ldots, x_k) \\ \rightarrow \big(S(x_1, \ldots, x_k) \vee T(x_1, \ldots, x_k) \big) \Big) \wedge \psi'(S/R) \wedge \vartheta'(T/R) \Big).$$

(b) If φ is of the form $(\psi \wedge \vartheta)$, then $tr_k(\varphi)$ is defined as

$$\exists S_1 \ldots \exists S_n \exists T_1 \ldots \exists T_m \big(\psi' \wedge \vartheta' \big).$$

4. If φ is of the form $\exists x_i \psi$ and $tr_k(\psi) = \exists S_1 \ldots \exists S_n \psi'$, where ψ' is syntactically an FOC^k-formula, then $tr_k(\varphi)$ is defined as

$$\exists S \exists S_1 \ldots \exists S_n \Big(\forall x_1 \ldots \forall x_k \big(R(x_1, \ldots, x_k) \rightarrow \exists x_i S(x_1, \ldots, x_k) \big) \wedge \psi'(S/R) \Big).$$

5. If φ is of the form $\forall x_i \psi$ and $tr_k(\psi) = \exists S_1 \ldots \exists S_n \psi'$, where ψ' is syntactically an FOC^k-formula, then $tr_k(\varphi)$ is defined as

$$\exists S \exists S_1 \ldots \exists S_n \Big(\forall x_1 \ldots \forall x_k \big(R(x_1, \ldots, x_k) \rightarrow \forall x_i S(x_1, \ldots, x_k) \big) \wedge \psi'(S/R) \Big).$$

A straightforward induction on φ shows that for every model $\mathfrak{A}$ and every team X of $\mathfrak{A}$ such that $\mathrm{dom}(()\, X) = \{x_1, \ldots, x_k\}$, we have that

$$\mathfrak{A} \models_X \varphi \quad \text{iff} \quad \big(\mathfrak{A}, \mathrm{rel}(X)\,\big) \models tr_k(\varphi).$$

We will proof here the cases 2b and 4. Assume that φ is $=\!(\boldsymbol{x}, x_i)$, $\boldsymbol{x} \in \{x_1, \ldots, x_k\}^t$ and $t \in \mathbb{N}$. Furthermore, assume that x_i does not occur in $\boldsymbol{x}$. Let $\boldsymbol{z}$ denote the sequence of variables in $\{x_1, \ldots, x_k\} \setminus \{x_i\}$ that do not occur in the tuple $\boldsymbol{z}$. Let $\mathfrak{A}$ be a model and X a team of $\mathfrak{A}$ with domain $\{x_1, \ldots, x_m\}$. The claim follows from the following equivalences.

$$\begin{aligned}
\mathfrak{A} \models_X =\!(\boldsymbol{x}, x_i) \quad &\Leftrightarrow \quad \text{For every } s_1, s_2 \in X\text{: } s_1(\boldsymbol{x}) = s_2(\boldsymbol{x}) \Rightarrow s_1(x_i) = s_2(x_i). \\
&\Leftrightarrow \quad \text{For every assignment } s \text{ defined by } s(\boldsymbol{x}) = \boldsymbol{a} \text{ there exists} \\
&\qquad\ \text{at most one } b \in A \text{ such that there exists an expansion} \\
&\qquad\ \text{of the assignment } s(b/x_i) \text{ in } X. \\
&\Leftrightarrow \quad \big(\mathfrak{A}, \mathrm{rel}(X)\,\big) \models \forall \boldsymbol{x}\, \exists^{\leq 1} x_i\, \exists \boldsymbol{z}\, R(x_1, \ldots, x_k).
\end{aligned}$$

For the proof of case 4, assume that φ is $\exists x_i \psi$ and $tr_k(\psi) = \exists S_1 \ldots \exists S_n \psi'$, where ψ' is syntactically an FOC^k-formula. Let $\mathfrak{A}$ be a model and X a team of $\mathfrak{A}$ with domain

$\{x_1, \ldots, x_k\}$. The claim follows by the following equivalences.

$$
\begin{aligned}
\mathfrak{A} \models_X \exists x_i \psi \Leftrightarrow\ & \mathfrak{A} \models_{X(F/x_i)} \psi, \text{ for some function } F : X \to A. \\
\Leftrightarrow\ & \big(\mathfrak{A}, \mathrm{rel}(X(F/x_i))\,\big) \models tr_k(\psi), \text{ for some function } F : X \to A. \\
\Leftrightarrow\ & \big(\mathfrak{A}, \mathrm{rel}(X)\,\big) \models \exists S \Big(\forall x_1 \ldots \forall x_k \big(R(x_1, \ldots, x_k) \\
& \qquad \to \exists x_i S(x_1, \ldots, x_k) \big) \wedge tr_k(\psi)(S/R) \Big). \\
\Leftrightarrow\ & \big(\mathfrak{A}, \mathrm{rel}(X)\,\big) \models \exists S \Big(\forall x_1 \ldots \forall x_k \big(R(x_1, \ldots, x_k) \\
& \qquad \to \exists x_i S(x_1, \ldots, x_k) \big) \wedge \exists S_1 \ldots \exists S_n \psi'(S/R) \Big). \\
\Leftrightarrow\ & \big(\mathfrak{A}, \mathrm{rel}(X)\,\big) \models \exists S \exists S_1 \ldots \exists S_n \Big(\forall x_1 \ldots \forall x_k \big(R(x_1, \ldots, x_k) \\
& \qquad \to \exists x_i S(x_1, \ldots, x_k) \big) \wedge \psi'(S/R) \Big).
\end{aligned}
$$

The first equivalence is due to the semantics of existential quantifiers and the second follows from the induction hypothesis. The third equivalence follows from the definition of $X(F/x_i)$. The fourth and the last equivalence is trivial.

Theorem 5. *For every $k \geq 1$ it holds that $\mathrm{D}^k \leq \Sigma_1^1(\mathrm{FOC}^k)$.*

Proof. Let τ be a relational vocabulary and $k \geq 1$. Let φ be a $\mathrm{D}^k(\tau)$-sentence and

$$\varphi^* = \exists R_1, \ldots R_n \psi$$

the related $\Sigma_1^1(\mathrm{FOC}^k)(\tau \cup \{R\})$-sentence given by Lemma 2. The following conditions are equivalent.

1. $\mathfrak{A} \models \varphi$.
2. $\mathfrak{A} \models_X \varphi$ for some team X such that $\mathrm{dom}(()\,X) = \{x_1, \ldots, x_k\}$.
3. $\big(\mathfrak{A}, \mathrm{rel}(X)\,\big) \models \varphi^*$ for some team X such that $\mathrm{dom}(X) = \{x_1, \ldots, x_k\}$.
4. $(\mathfrak{A}, R) \models \exists R_1 \ldots \exists R_n \big(\exists x_1 \ldots \exists x_k R(x_1, \ldots, x_k) \wedge \psi \big)$ for some nonempty $R \subseteq A^k$.
5. $\mathfrak{A} \models \exists R \exists R_1 \ldots \exists R_n \big(\exists x_1 \ldots \exists x_k R(x_1, \ldots, x_k) \wedge \psi \big)$.

The equivalence of 1 and 2 follows from Proposition 2 and the fact that $\mathrm{fr}(\varphi) = \emptyset$. By Lemma 2, conditions 2 and 3 are equivalent. The equivalence of 3 and 4 follows from the fact that the relations $\mathrm{rel}(X)$ with $\mathrm{dom}(X) = \{x_1, \ldots, x_k\}$ are exactly nonempty subsets of A^k. The conditions 4 are 5 clearly equivalent.

The following theorem follows from the proofs of Lemma 2 and Theorem 5 with a small modification.

Theorem 6. *For every $k \geq 1$ it holds that $\mathrm{D}^k < \Sigma_1^1(\mathrm{FO}^{k+1})$.*

Proof. For each $k \geq 1$, the translation

$$tr_k : \mathrm{D}^k(\tau) \mapsto \Sigma_1^1(\mathrm{FOC}^k)(\tau \cup \{R\})$$

defined in the proof of Lemma 2 can be easily modified to a translation

$$tr'_k : \mathrm{D}^k(\tau) \mapsto \Sigma_1^1(\mathrm{FO}^{k+1})(\tau \cup \{R\})$$

by replacing case 2b of the translation tr_k by the following case 2b′.

2. Assume that φ is a dependence atom $=\!(\boldsymbol{x}, x_i)$, where $\boldsymbol{x} \in \{x_1, \ldots, x_k\}^t$ and $t \in \mathbb{N}$.
   (b′) If x_i does not occur in $\boldsymbol{x}$ then $tr_k(\varphi)$ is defined as

$$\forall \boldsymbol{x}\, \forall x_i \forall x_{k+1} \Big(\big(\exists \boldsymbol{z}\, R(x_1, \ldots, x_k) \\ \wedge \exists \boldsymbol{z}\, R(x_1, \ldots, x_{i-1}, x_{k+1}, x_{i+1}, \ldots, x_k) \big) \to x_i = x_{k+1} \Big),$$

   where $\boldsymbol{z}$ is exactly the sequence of variables in $\{x_1, \ldots, x_k\} \setminus \{x_i\}$ (in alphabetical order) that do not occur in $\boldsymbol{x}$.

Clearly the proofs of Lemma 2 and Theorem 5 can be straightforwardly modified to handle the translation tr'_k.

9 Complexity of model checking

In this section we study variants of the model checking problem for finite variable dependence logic. We show that, for $k \geq 2$, the data complexity and the combined complexity for D^k is NP-complete. We give a reduction from the dominating set problem, an NP-complete problem from graph theory, to the model checking problem with a fixed D^2-sentence.

We first recall the definitions for graphs and dominating sets. A finite graph is a structure

$$G = (V, E),$$

where V is a finite set and E is a subset of the set

$$\{\{u, v\} \mid u, v \in V\}.$$

The elements of V are called *vertices* and the elements in E are called *edges*. A set $D \subseteq V$ is called a *dominating set* for the graph G if for every vertex $v \in V$ either $v \in D$ holds or there exists some vertex $u \in D$ such that $\{u, v\} \in E$ holds.

We are now ready to describe the NP-complete problem to be used in the proof of Proposition 12 to show that the data complexity for D^2 is NP-hard.

Definition 16. *The dominating set problem is the following decision problem: Given a graph G and a number $n \in \mathbb{N}$, does there exist a dominating set for G of size at most n.*

Theorem 7. *The dominating set problem is* NP*-complete.*

Every graph $G = (V, E)$ can be naturally identified with a first-order structure $\mathfrak{A}_G = (A_G, R_G)$ of vocabulary $\{R\}$, where

$$A_G := V \quad \text{and} \quad R_G := \{(u, v) \in A_G^2 \mid \{u, v\} \in E\}.$$

We call $\mathfrak{A}_G$ the *first-order structure corresponding to* G. Let P be a unary relation symbol and $\mathfrak{A}_G^*$ some expansion of $\mathfrak{A}_G$ to the vocabulary $\{R, P\}$. We call the structure $\mathfrak{A}_G^*$ a *first-order structure corresponding to* (G, n) if the cardinality of the extension of the proposition symbol P in $\mathfrak{A}_G^*$ is $\min\{n, |V|\}$.

Lemma 3. *There exists a polynomial time algorithm that constructs from any given finite graph* $G = (V, E)$ *and a natural number* n *a first-order structure* $\mathfrak{A}_{(G,n)}$ *corresponding to* (G, n).

Proof. Straightforward.

We are now ready to show that the data complexity for D^2 is NP-hard.

Proposition 12. *The data complexity for* D^2 *is* NP*-hard.*

Proof. We will prove the NP-hardness by a polynomial time reduction from the dominating set problem to the model checking problem with a fixed D^2-sentence. By Lemma 3, there is a polynomial time algorithm that constructs from each input (G, n) to the dominating set problem a first-order structure $\mathfrak{A}_{(G,n)}$ corresponding to (G, n). Let φ_{dsp} denote the following D^2 sentence

$$\forall x \exists y \Big(\big(E(x, y) \lor x = y \big) \land \exists x \big({=}(x, y) \land P(x) \big) \Big).$$

It is straightforward to check that the following conditions are equivalent for any finite graph $G = (V, E)$ and $n \in \mathbb{N}$.

1. G has a dominating set of size at most n.
2. $\mathfrak{A} \models \varphi_{dsg}$ for some first-order structure $\mathfrak{A}$ corresponding to (G, n).
3. $\mathfrak{A} \models \varphi_{dsg}$ for all first-order structures $\mathfrak{A}$ corresponding to (G, n).

Hence the following conditions are equivalent for every input (G, n) to the dominating set problem.

1. G has a dominating set of size at most n.
2. $\mathfrak{A}_{(G,n)} \models \varphi_{dsg}$.

Now, since the dominating set problem is an NP-complete problem and $\mathfrak{A}_{(G,n)}$ is constructed from G by a polynomial time algorithm, it follows that the data complexity of D^2 is NP-hard.

Since finite variable fragments of independence-friendly logic and dependence logic translate into finite variable fragments of existential second-order logic, the following theorem provides us upper bounds for the complexities of the model checking problems for finite variable independence-friendly logics and dependence logics.

Theorem 8. *The expression complexity and the combined complexity for* ESO^k *is* NP-*complete, for every* $k \geq 1$.

For finite variable dependence logics we have the following theorem.

Theorem 9. *The following hold for every* $k \geq 2$.

1. *The data complexity for* D^k *is* NP-*complete.*
2. *The expression complexity for* D^k *is in* NP.
3. *The combined complexity for* D^k *is* NP-*complete.*

Proof. Fix $k \geq 2$. We will first show the lower bounds for the data complexity and the combined complexity for D^k. By Proposition 12, the data complexity for D^2 is NP-hard. Therefore the data complexity of D^k is NP-hard. Now since the combined complexity is bound below by the data complexity, it follows that the combined complexity for D^k is NP-hard. For the upper bounds, note first that, by Theorem 8, the combined complexity for ESO^n is in NP, for every $n \in \mathbb{N}$. Therefore, and since the translation from D^k to ESO^{k+1} defined in Theorem 6 is clearly polynomial, we conclude that the combined complexity for D^k is in NP. Since the data complexity and the expression complexity is bound above by the combined complexity, it follows that both the data complexity and the expression complexity for D^k is in NP.

Furthermore, by Proposition 10, each D^1 sentence is equivalent to some first-order sentence. Therefore and since the data complexity for FO is in AC^0 (and in P), it follows that the data complexity for D^1 is also in AC^0.

10 Modal logic

Let Φ be a set of atomic propositions. The set of formulae for *modal logic* $\mathrm{ML}(\Phi)$ is generated by the following grammar

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

Note that, since negations are allowed only in front of proposition symbols, $\Box$ and $\Diamond$ are *not* interdefinable.

Definition 17. *Let* Φ *be a set of atomic proposition symbols. A* Kripke model K *over* Φ *is a tuple* $\mathrm{K} = (W, R, V)$, *where* W *is a nonempty set of* worlds, $R \subseteq W \times W$ *is a binary relation, and* $V : \Phi \to \mathcal{P}(W)$ *is a* valuation.

Definition 18. *Let* K *be a Kripke model and* w *a point of* K*. The satisfaction relation* $\mathrm{K}, w \models \varphi$ *for* ML *is defined as follows.*

$$\begin{aligned}
\mathrm{K}, w \models p &\quad\Leftrightarrow\quad w \in V(p)\\
\mathrm{K}, w \models \neg p &\quad\Leftrightarrow\quad w \notin V(p)\\
\mathrm{K}, w \models (\varphi \wedge \psi) &\quad\Leftrightarrow\quad \mathrm{K}, w \models \varphi \textit{ and } K, w \models \psi.\\
\mathrm{K}, w \models (\varphi \vee \psi) &\quad\Leftrightarrow\quad \mathrm{K}, w \models \varphi \textit{ or } K, w \models \psi.\\
\mathrm{K}, w \models \Diamond\varphi &\quad\Leftrightarrow\quad \mathrm{K}, w' \models \varphi \textit{ for some } w' \textit{ such that } wRw'.\\
\mathrm{K}, w \models \Box\varphi &\quad\Leftrightarrow\quad \mathrm{K}, w' \models \varphi \textit{ for all } w' \textit{ such that } wRw'.
\end{aligned}$$

Theorem 10 (Finite Tree Property). *A modal formula is satisfiable if and only if it is satisfiable on a finite tree.*

Theorem 11.

1. *The model checking problem for modal logic is* P*-complete for combined complexity (in time* $|\varphi| \times |W| \times |W|$*).*
2. *The satisfiability problem for modal logic is* PSPACE*-complete.*
3. *The validity problem for modal logic is* PSPACE*-complete.*

Definition 19. *For a collection of proposition symbols* Φ*, let* $\mathcal{L}^1_R(\Phi)$ *be the first-order language (with equality) which has unary predicates* $P_0, P_1, \ldots$ *corresponding to the proposition symbols* $p_0, p_1, \ldots$ *in* Φ*, and an binary relation symbol* R *for the modal operators* $\Box$ *and* $\Diamond$

Definition 20 (Standard translation). *Let* $x, y \in \{x_1, x_2\}$ *be distinct first-order variables. Then the standard translation* ST_x *taking modal formulas to first-order formulas in* $\mathcal{L}^1_R(\Phi)$ *is defined as follows:*

$$\begin{aligned}
ST_x(p) &= P(x),\\
ST_x(\neg p) &= \neg P(x),\\
ST_x(\varphi \vee \psi) &= ST_x(\varphi) \vee ST_x(\psi),\\
ST_x(\varphi \wedge \psi) &= ST_x(\varphi) \wedge ST_x(\psi),\\
ST_x(\Diamond\varphi) &= \exists y\big(R(x,y) \wedge ST_y(\varphi)\big),\\
ST_x(\Diamond\varphi) &= \forall y\big(R(x,y) \rightarrow ST_y(\varphi)\big).
\end{aligned}$$

Proposition 13. *Let* K, w *be a pointed Kripke model and* φ *a formula of modal logic. Then*

$$\mathrm{K}, w \models \varphi \Leftrightarrow \mathrm{K}, s(w/x) \models ST_x(\varphi).$$

10.1 Bisimulation and definability in Kripke semantics

A *pointed Kripke model* is a pair (K, w) such that K is a Kripke model over Φ and w is a state in K.

Definition 21. *Let (K, w) and (K', w') be pointed Kripke models. We say that (K, w) and (K', w') are* modally equivalent*, in symbols $K, w \equiv K', w'$, if for every $\varphi \in \mathrm{ML}(\Phi)$*

$$K, w \models \varphi \quad \Longleftrightarrow \quad K', w' \models \varphi.$$

Definition 22. *A bisimulation between models $\mathrm{K} = (W, R, V)$ and $\mathrm{K}' = (W', R', V')$ is a nonempty binary relation $E \subseteq W \times W'$ such that whenever wEw' holds we have that:*

- *Atomic harmony: $\mathrm{K}, w \models p \Leftrightarrow \mathrm{K}', w \models p$, for every proposition symbol p.*
- *Zig: If $R(w, v)$, then there exists a point $v' \in W'$ such that vEv' and $R(w', v')$.*
- *Zag: If $R'(w', v')$, then there exists a point $v \in W$ such that vEv' and $R(w, v)$.*

If there exists a bisimulation E between the models K and K' such that the domain of E is W and the range of E is W', we say that K and K' are bisimilar*. We say that pointed models K, w and K', w' are* bisimilar*, if wEw' holds for some bisimulation E.*

Proposition 14. *If the pointed models K, w and K', w' are bisimilar then the equivalence $\mathrm{K}, w \equiv \mathrm{K}', w'$ holds.*

However the converse of the above proposition does not hold.

Example 1. Define $W = \{(i, j) \in \mathbb{N}^2 \mid j \leq i\}$, $W' = W \cup \{\omega, i \mid i \in \mathbb{N}\}$,

$$R = \{\big((i, j), (i, j + 1)\big) \in W^2 \mid i, j \in W\} \cup \{\big((0, 0), (j, 0)\big) \mid 0 \neq j \in W\},$$

and $R' = R \cup \{\big((\omega, j), (\omega, j + 1) \mid j \in \mathbb{N}\} \cup \{\big((0, 0), (\omega, 0)\big)\}$. Let $V(p) = V'(p) = \emptyset$, for all propositions p, and let $\mathrm{K} = (W, R, V)$ and $\mathrm{K}' = (W', R', V')$. Clearly, for every modal formula φ,

$$\mathrm{K}, (0, 0) \models \varphi \Leftrightarrow \mathrm{K}', (0, 0) \models \varphi$$

holds. However $\mathrm{K}, (0, 0)$ and $\mathrm{K}', (0, 0)$ are NOT bisimilar.

The converse holds for many restricted classes of pointed models.

Proposition 15. *Let K and K' be*

1. *finite Kripke models*
2. *image finite Kripke models (i.e. for every point w the set $\{v \mid wRv\}$ is finite)*

then the pointed models (K,w) and (K'w) are bisimilar if and only if they are modally equivalent.

Definition 23. *A first order formula $\varphi(x)$ is invariant for bisimulation if for all models $\mathfrak{A}$ and $\mathfrak{A}'$, and all points w of $\mathfrak{A}$ and w' of $\mathfrak{A}'$, and all bisimulations E between $\mathfrak{A}$ and $\mathfrak{A}'$ such that wEw', we have that*

$$\mathfrak{A}, s(w/x) \Leftrightarrow \mathfrak{A}', s(w'/x).$$

Theorem 12. *The following are equivalent for all first-order formulas $\varphi(x)$ in one free variable x.*

1. *$\varphi(x)$ is invariant for bisimulation.*
2. *$\varphi(x)$ is equivalent to the standard translation of a modal formula.*

The expressive power of modal logic ML with respect to Kripke semantics can be completely characterized in terms of a finite variant of bisimulation called k-bisimulation.

The *modal depth* $\mathrm{md}(\varphi)$ of a formula of $\mathrm{ML}(\Phi)$ is defined as follows:

$$\begin{aligned}&\mathrm{md}(p) = \mathrm{md}(\neg p) = 0 \text{ for } p \in \Phi,\\ &\mathrm{md}(\varphi \wedge \psi) = \mathrm{md}(\varphi \vee \psi) = \max\{\mathrm{md}(\varphi)\,, \mathrm{md}(\psi)\},\\ &\mathrm{md}(\Diamond\varphi) = \mathrm{md}(\Box\varphi) = \mathrm{md}(\varphi) + 1.\end{aligned}$$

Definition 24. *Let k be a natural number, and let (K, w) and (K', w') be pointed Φ-models. We say that (K, w) and (K', w') are* k-equivalent*, in symbols $K, w \equiv_k K', w'$, if for every $\varphi \in \mathrm{ML}(\Phi)$ with $\mathrm{md}(\varphi) \leq k$*

$$K, w \models \varphi \quad \Longleftrightarrow \quad K', w' \models \varphi.$$

Definition 25. *Let $k \in \mathbb{N}$, and let (K, w) and (K', w') be pointed Φ-models. We write $K, w \rightleftarrows_k K', w'$ if (K, w) and (K', w') are k-bisimilar. The k-bisimilarity relation $\rightleftarrows_k$ can be defined recursively as follows:*

- *$K, w \rightleftarrows_0 K', w'$ if and only if the equivalence $K, w \models p \iff K', w' \models p$ holds for all $p \in \Phi$.*
- *$K, w \rightleftarrows_{k+1} K', w'$ if and only if $K, w \rightleftarrows_0 K', w'$, and*
  - *for every v such that wRv there is v' such that $w'R'v'$ and $K, v \rightleftarrows_k K', v'$, and*
  - *for every v' such that $w'R'v'$ there is v such that wRv and $K, v \rightleftarrows_k K', v'$.*

A class $\mathcal{K}$ of pointed Φ-models is *closed under k-bisimulation* if it satisfies the following condition:

- $(K, w) \in \mathcal{K}$ and $K, w \rightleftarrows_k K', w'$ implies that $(K', w') \in \mathcal{K}$.

We will also make use of the fact that for every pointed Φ-model (K, w) and every $k \in \mathbb{N}$ there is a formula that characterizes (K, w) completely up to k-equivalence. These *Hintikka formulas* (or *characteristic formulas*) are defined as follows:

Definition 26. *Assume that Φ is a finite set of proposition symbols. Let $k \in \mathbb{N}$ and let (K, w) be a pointed Φ-model. The k-th* Hintikka formula *$\chi^k_{K,w}$ of (K, w) is defined recursively as follows:*

- $\chi^0_{K,w} := \bigwedge\{p \mid p \in \Phi, w \in V(p)\} \wedge \bigwedge\{\neg p \mid p \in \Phi, w \notin V(p)\}$.
- $\chi^{k+1}_{K,w} := \chi^k_{K,w} \wedge \bigwedge_{wRv} \Diamond\chi^k_{K,v} \wedge \Box \bigvee_{(w,v)} \chi^k_{K,v}$.

It is easy to see that $\mathrm{md}\big(\chi^k_{K,w}\big) = k$, and $K, w \models \chi^k_{K,w}$ for every pointed Φ-model (K, w). Moreover, the Hintikka formula $\chi^k_{K,w}$ captures the essence of k-bisimulation:

Proposition 16. *Let Φ be a finite set of proposition symbols, $k \in \mathbb{N}$, and (K, w) and (K', w') pointed Φ-models. Then*

$$K, w \equiv_k K', w' \quad\iff\quad K, w \rightleftarrows_k K', w' \quad\iff\quad K', w' \models \chi^k_{K,w}.$$

The characterization for the expressive power of ML with respect to Kripke-semantics can now be stated as follows:

Proposition 17 (van Benthem, Gabbay). *Assume that Φ is a finite set of proposition symbols. A class $\mathcal{K}$ of pointed Φ-models is definable in* ML *if and only if there is $k \in \mathbb{N}$ such that $\mathcal{K}$ is closed under k-bisimulation.*

11 Modal logics with team semantics

The syntax of *modal logic with intuitionistic disjunction* $\mathrm{ML}(\varovee)(\Phi)$ is obtained by extending the syntax of $\mathrm{ML}(\Phi)$ by the grammar rule

$$\varphi ::= (\varphi \varovee \varphi).$$

Team semantics for modal logic is defined via *Kripke models* and *teams*. In the context of modal logic, teams are subsets of the domain of the model.

Definition 27. *Let* $\mathrm{K} = (W, R, V)$ *be a Kripke model. A subset T of W is called a* team *of* K. *Furthermore, define*

$$R[T] := \{w \in W \mid vRw \textit{ holds for some } v \in T\},$$
$$R^{-1}[T] := \{w \in W \mid wRv \textit{ holds for some } v \in T\}.$$

For teams $T, S \subseteq W$, we write $T[R]S$ if $S \subseteq R[T]$ and $T \subseteq R^{-1}[S]$. Thus, $T[R]S$ holds if and only if for every $w \in T$ there exists some $v \in S$ such that wRv, and for every $v \in S$ there exists some $w \in T$ such that wRv.

We are now ready to define team semantics for modal logic and modal logic with intuitionistic disjunction. Similar to the case of first-order logic (with teams and assignments), the team semantics of modal logic, in a rather strong sense, coincides with the traditional semantics of modal logic defined via pointed Kripke models.

Definition 28. *Let* K *be a Kripke model. The satisfaction relation* $\mathrm{K}, T \models \varphi$ *for* ML *is defined as follows.*

$$
\begin{array}{rcl}
\mathrm{K}, T \models p & \Leftrightarrow & w \in V(p) \text{ for every } w \in T. \\
\mathrm{K}, T \models \neg p & \Leftrightarrow & w \notin V(p) \text{ for every } w \in T. \\
\mathrm{K}, T \models (\varphi \wedge \psi) & \Leftrightarrow & \mathrm{K}, T \models \varphi \text{ and } K, T \models \psi. \\
\mathrm{K}, T \models (\varphi \vee \psi) & \Leftrightarrow & \mathrm{K}, T_1 \models \varphi \text{ and } \mathrm{K}, T_2 \models \psi, \\
 & & \text{for some } T_1, T_2 \text{ such that } T_1 \cup T_2 = T. \\
\mathrm{K}, T \models \Diamond\varphi & \Leftrightarrow & \mathrm{K}, T' \models \varphi \text{ for some } T' \text{ such that } T[R]T'. \\
\mathrm{K}, T \models \Box\varphi & \Leftrightarrow & \mathrm{K}, T' \models \varphi, \text{ where } T' = R[T].
\end{array}
$$

For $\mathrm{ML}(\otimes)$*, we have the following additional clause:*

$$
\mathrm{K}, T \models (\varphi \otimes \psi) \quad \Leftrightarrow \quad \mathrm{K}, T \models \varphi \text{ or } \mathrm{K}, T \models \psi.
$$

Proposition 18. *Let* $\varphi \in \mathrm{ML}$*,* K *be a Kripke model and* T *a team of* K*. Then*

$$
\mathrm{K}, T \models \varphi \quad \textit{iff} \quad \forall w \in T : \mathrm{K}, w \models_{\mathrm{ML}} \varphi.
$$

Here $\models_{\mathrm{ML}}$ *refers to the ordinary satisfaction relation of modal logic defined via pointed Kripke models.*

The syntax for *modal dependence logic* $\mathrm{MDL}(\Phi)$ is obtained by extending the syntax of $\mathrm{ML}(\Phi)$ by propositional dependence atoms

$$
\varphi ::= \mathrm{dep}(p_1, \ldots, p_n, q)\,, \quad \text{where } p_1, \ldots, p_n, q \in \Phi.
$$

The syntax for *extended modal dependence logic* $\mathrm{EMDL}(\Phi)$ is obtained by extending the syntax of $\mathrm{ML}(\Phi)$ by *modal dependence atoms*

$$
\varphi ::= \mathrm{dep}(\varphi_1, \ldots, \varphi_n, \psi)\,, \quad \text{where } \varphi_1, \ldots, \varphi_n, \psi \text{ are } \mathrm{ML}(\Phi)\text{-formulae.}
$$

The intuitive meaning of the modal dependence atom $\mathrm{dep}(\varphi_1, \ldots, \varphi_n, \psi)$ is that inside a team the truth value of the formula ψ is completely determined by the truth values of the formulae $\varphi_1, \ldots, \varphi_n$. The semantics for these dependence atoms is defined as follows.

$$
\begin{array}{rcl}
\mathrm{K}, T \models \mathrm{dep}(\varphi_1, \ldots, \varphi_n, \psi) & \Leftrightarrow & \forall w, v \in T : \bigwedge_{i=1}^{n} \big(\mathrm{K}, \{w\} \models \varphi_i \Leftrightarrow \mathrm{K}, \{v\} \models \varphi_i\big) \\
 & & \text{implies } \big(\mathrm{K}, \{w\} \models \psi \Leftrightarrow \mathrm{K}, \{v\} \models \psi\big).
\end{array}
$$

Proposition 19 (Downwards closure). *Let* φ *be a formula of* $\mathrm{ML}(\otimes)$*,* MDL *or* EMDL*. Let* K *be a Kripke model and let* $S \subseteq T$ *be teams of* K*.*

$$
\textit{If } \mathrm{K}, T \models \varphi \textit{ then also } \mathrm{K}, S \models \varphi.
$$

Proposition 20. *Truth of* $\mathrm{ML}(\varovee)$*-formulae (*MDL *and* EMDL *respectively) are preserved under taking disjoint unions, i.e., if* K *and* K′ *are Kripke models, T is a team of* K *and* $\mathrm{K} \uplus \mathrm{K}'$ *denotes the disjoint union of* K *and* K′ *then*

$$\mathrm{K}, T \models \varphi \quad \Leftrightarrow \quad \mathrm{K} \uplus \mathrm{K}', T \models \varphi,$$

for every $\varphi \in \mathrm{ML}(\varovee)$.

Proof. By induction.

12 Dependency quantified Boolean formulae

Deciding whether a given quantified Boolean formula is valid is a canonical PSPACE-complete problem. *Dependency quantified Boolean formulae* are variants of quantified Boolean formulae for which the corresponding decision problem is NEXPTIME-complete.

A *propositional variable* (i.e., proposition symbol) is a variable that is assigned either 0 or 1, i.e., either *false* or *true*. We use symbols $p, q, p_1, \ldots, p_n$, etc., for propositional variables. If $C = \{p_{i_1}, \ldots, p_{i_n}\}$ is a finite subset of propositional variables and $i_j < i_{j+1}$ for each $j < n$, we let $\boldsymbol{c}$ denote the tuple $(p_{i_1}, \ldots, p_{i_n})$.

The formulae of *propositional logic* are built from propositional variables by the following grammar:

$$\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi), \quad \text{where } p \text{ is a propositional variable.}$$

Quantified Boolean formulae extend propositional logic by allowing a prenex quantification of propositional variables. Formally, the set of *quantified Boolean formulae* is built from formulae of propositional logic by the following grammar:

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

where p is a propositional variable θ is formula of propositional logic. The semantics for quantified Boolean formulae is defined via assignments $s : \mathrm{PROP} \to \{0, 1\}$ in the obvious way. When s is an assignment and $\boldsymbol{p} = (p_1, \ldots, p_n)$ is tuple of propositional variables, we denote by $s(\boldsymbol{p})$ the tuple $\big(s(p_1), \ldots, s(p_n)\big)$. Moreover, when p is a propositional variable and $b \in \{0, 1\}$ is a truth value, we denote by $s(p \mapsto b)$ the modified assignment defined as follows:

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

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

Theorem 13. *The validity problem of* QBF *is* PSPACE*-complete.*

A *simple quantified Boolean formula* is a closed quantified Boolean formula of the type

$$\varphi := \forall p_1 \dots \forall p_n \exists q_1 \dots \exists q_k \theta,$$

where θ is a propositional formula and the propositional variables p_i, q_j are all distinct. Any tuple $(C_1, \dots, C_k)$ such that $C_1, \dots, C_k \subseteq \{p_1, \dots, p_n\}$ is called a *constraint* for φ. If $C_1 \subseteq C_2 \subseteq \dots \subseteq C_k$, we call the constraint $(C_1, C_2, \dots, C_k)$ *simple.*

Intuitively, a constraint $C_j = \{p_1, p_3\}$ carries the same meaning as the dependence atom $\text{dep}(p_1, p_3, q_j)$.

Definition 29. *A simple quantified Boolean formula* $\forall p_1 \dots \forall p_n \exists q_1 \dots \exists q_k \theta$ *is* valid under a constraint $(C_1, \dots, C_k)$, *if for each* $i \le k$ *there exists a function*

$$f_i : \{0,1\}^{|C_i|} \to \{0,1\}$$

such that, for each assignment $s : \{p_1, \dots, p_n\} \to \{0,1\}$,

$$s\Big(q_1 \mapsto f_1\big(s(\boldsymbol{c}_1)\big), \dots, q_k \mapsto f_k\big(s(\boldsymbol{c}_k)\big)\Big) \models \theta.$$

A *dependency quantified Boolean formula* is a pair $(\varphi, \boldsymbol{C})$, where φ is a simple quantified Boolean formula and $\boldsymbol{C}$ is a constraint for φ. We say that $(\varphi, \boldsymbol{C})$ is *valid*, if φ is valid under the constraint $\boldsymbol{C}$

It is easy to see that there is a close connection between quantified Boolean formulae and simple quantified Boolean formulae with simple constraints.

Example 2. Let θ be a formula of propositional logic with propositional variables p_1, p_2, q_1, q_2. The following are equivalent:

1. The simple quantified Boolean formula $\forall p_1 \forall p_2 \exists q_1 \exists q_2 \theta$ is valid under the constraint $(\{p_1\}, \{p_1, p_2\})$.
2. The closed quantified Boolean formula $\forall p_1 \exists q_1 \forall p_2 \exists q_2 \theta$ is valid.

More generally, the following holds:

Proposition 21. *There exists a polynomial time computable function that associates each closed quantified Boolean formula* φ *to a simple quantified Boolean formula with a simple constraint* φ^* *such that* φ *is valid iff* φ^* *is valid, and vice versa.*

Proof. Let $\varphi := \forall p_1 \dots \forall p_n \exists q_1 \dots \exists q_k \theta$ be a simple quantified Boolean formula and let $(C_1, \dots, C_k)$ be a simple constraint for φ. Define $C_{k+1} := \{p_1, \dots p_n\} \setminus C_k$. It is straightforward to check that φ is valid under the constraint $(C_1, \dots, C_k)$ if and only if the quantified Boolean formula $\forall \boldsymbol{c}_1 \exists q_1 \dots \forall \boldsymbol{c}_k \exists q_k \forall \boldsymbol{c}_{k+1} \theta$ is valid.

For the converse direction, let φ be any formula of QBF. Without loss of generality, we may assume that each variable quantified in φ is quantified exactly once. Thus

$$\varphi = \forall \boldsymbol{p}_1 \exists q_1 \dots \forall \boldsymbol{p}_k \exists q_k \forall \boldsymbol{p}_{k+1} \theta,$$

for some (possibly empty) tuples of propositional variables $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k+1}$, for some propositional variables $q_1, \ldots, q_k$, and for some propositional formula θ. Define

$$C_i := \{p \in \text{PROP} \mid p \text{ occurs in the tuple } \boldsymbol{p}_j \text{ for some } j \leq i\}, \quad \text{for each } i \leq k.$$

Clearly $(C_1, \ldots, C_k)$ is a simple constraint for the simple quantified Boolean formula $\forall \boldsymbol{p}_1 \ldots \forall \boldsymbol{p}_{k+1} \exists q_1 \ldots \exists q_k \theta$. Furthermore, it is straightforward to check that φ is valid if and only if $\forall \boldsymbol{p}_1 \ldots \forall \boldsymbol{p}_{k+1} \exists q_1 \ldots \exists q_k \theta$ is valid under the constraint $(C_1, \ldots, C_k)$.

Let DQBF denote the set of all dependency quantified Boolean formulae and let SDQBF be the fragment of DQBF in which the constraints are simple.

Theorem 14. *The validity problem of* DQBF *is* NEXPTIME*-complete.*

The following corollary follows from Theorem 13 and Proposition 21

Corollary 2. *The validity problem of* SDQBF *is* PSPACE*-complete.*

13 Satisfiability, validity, and model checking in modal logics with team semantics

We first define the concepts of satisfiability and validity in the context of team semantics:

Definition 30. *A formula φ of* MDL $\big($EMDL *and* ML($\oslash$)*, resp.*$\big)$ *is* satisfiable*, if there exists a Kripke model* K *and a non-empty team T of* K *such that* $\mathrm{K}, T \models \varphi$.

Definition 31. *The formula φ is* valid*, if* $\mathrm{K}, T \models \varphi$ *holds for every Kripke model* K *(such that the proposition symbols in φ are mapped by the valuation of* K*) and every team T of* K.

The satisfiability problem and the validity problem is then defined in the obvious manner:

Satisfiability problem:
Input: A binary encoding of a formula of a given logic.
Output: Is the formula is satisfiable?

Validity problem:
Input: A binary encoding of a formula of a given logic.
Output: Is the formula is valid?

The variant of the model checking problem that we are concerned in this section is the following:

Model checking problem:
Input: A binary encodings of a formula ψ of MDL $\big($EMDL and ML($\oslash$), resp.$\big)$, of a finite Kripke model K, and of a team T of K.
Output: Does $\mathrm{K}, T \models \psi$ hold?

Let φ be a formula of MDL, EMDL or ML($\varovee$). The set nbSubf(φ) of *non-Boolean subformulae* of φ is defined recursively as follows.

$$\begin{aligned}
&\mathrm{nbSubf}(\neg p) := \mathrm{nbSubf}(p) := \{p\},\\
&\mathrm{nbSubf}(\triangle\varphi) := \{\triangle\varphi\} \cup \mathrm{nbSubf}(\varphi) \quad \text{for } \triangle \in \{\Diamond, \Box\},\\
&\mathrm{nbSubf}(\varphi \circ \psi) := \mathrm{nbSubf}(\varphi) \cup \mathrm{nbSubf}(\psi) \quad \text{for } \circ \in \{\varovee, \vee, \wedge\},\\
&\mathrm{nbSubf}\big(\mathrm{dep}(\varphi_1, \dots, \varphi_n, \psi)\big) := \mathrm{nbSubf}(\varphi_1) \cup \dots \cup \mathrm{nbSubf}(\varphi_n) \cup \mathrm{nbSubf}(\psi).
\end{aligned}$$

Proposition 22. *For every formula $\varphi \in \mathrm{EMDL}(\Phi)$ there exists an equivalent formula*

$$\varphi^* = \bigovee_{i \in I} \varphi_i, \tag{4}$$

where I is a finite set of indices and $\varphi_i \in \mathrm{ML}(\Phi)$, for each $i \in I$. Furthermore, for each $i \in I$, the size of φ_i is only exponential in the size of φ and $|\mathrm{nbSubf}(\varphi_i)| \leq 3 \times |\varphi|$.

Proof. We will first define an exponential translation $\varphi \mapsto \varphi^+$ from EMDL(Φ) to ML($\varovee$)(Φ). The cases for proposition symbols, Boolean connectives and modalities are trivial, i.e.,

$$\begin{array}{rcl}
p & \mapsto & p,\\
\neg p & \mapsto & \neg p,\\
(\varphi \wedge \psi) & \mapsto & (\varphi^+ \wedge \psi^+)\\
(\varphi \vee \psi) & \mapsto & (\varphi^+ \vee \psi^+)\\
\Diamond\varphi & \mapsto & \Diamond\varphi^+\\
\Box\varphi & \mapsto & \Box\varphi^+.
\end{array}$$

The only interesting case is the case for the dependence atom. We define that

$$\mathrm{dep}(\varphi_1, \dots, \varphi_n, \psi) \quad \mapsto \quad \bigvee_{a_1,\dots,a_n \in \{\bot, \top\}} \big(\bigwedge_{i \leq n} \varphi_i^{a_i} \wedge (\psi \varovee \psi^\bot)\big),$$

where $\varphi^\top$ denotes φ and $\varphi^\bot$ denotes the ML(Φ) formula obtained from $\neg\varphi$ by pulling all negations to the atomic level. Notice that the size of φ^+ is $\leq c \times |\varphi| \times 2^{|\varphi|}$, for some constant c. Thus the size of φ^+ is at most exponential with respect to the size of φ. From φ^+ it is easy to obtain an equivalent ML($\varovee$)(Φ)-formula φ^* of the form (4).

Let F be the set of all selection functions f that select, separately for each occurrence, either the left disjunct ψ or the right disjunct θ of each subformula of the form $(\psi \varovee \theta)$ of φ^+. Let φ_f^+ denote the formula obtained from φ^+ by substituting each occurrence of a subformula of type $(\psi \varovee \theta)$ by $f\big((\psi \varovee \theta)\big)$. Define

$$\varphi^* := \bigovee_{f \in F} \varphi_f^+.$$

It is straightforward to prove that φ^* is equivalent to φ^+ and hence to φ. Since, for each $f \in F$, φ^+_f is obtained from φ^+ by substituting subformulae of type $(\psi \varovee \theta)$ with either ψ or θ, it is clear that the size of φ^+_f is bounded above by the size of φ^+. Recall that the size of φ^+ is at most exponential with respect to the size of φ. Therefore, for each $f \in F$, the size of φ^+_f is at most exponential with respect to the size of φ.

We say that the modal operator $\Diamond$ in $\Diamond\theta$ *dominates* an intuitionistic disjunction if $\varovee$ occurs in θ. To see that $|\mathrm{nbSubf}(\varphi^+_f)| \leq 3 \times |\varphi|$, for each $f \in F$, notice first that in the translation $\varphi \mapsto \varphi^+$ the only case that can increase the number of non-Boolean subformulae is the case for the dependence atom. Each $\varphi_i^\perp$ and $\psi^\perp$ may introduce new non-Boolean subformulae. Thus it is straightforward to see that $|\mathrm{nbSubf}(\varphi^+)| \leq 2 \times |\mathrm{nbSubf}(\varphi)|$. Furthermore, notice that the number of modal operators that dominate an intuitionistic disjunction in φ^+ is less or equal to the number of modal operators in φ. Let k denote the number of modal operators in φ. It is easy to see that $|\mathrm{nbSubf}(\varphi^+_f)| \leq |\mathrm{nbSubf}(\varphi^+)| + k$, for each $f \in F$. Now since $k \leq |\varphi|$ and $|\mathrm{nbSubf}(\varphi)| \leq |\varphi|$, we obtain that $|\mathrm{nbSubf}(\varphi^+_f)| \leq 3 \times |\varphi|$, for each $f \in F$.

Note that by a more careful bookkeeping, a slightly better upper bound could be obtained.

Proposition 23. $\mathrm{ML}(\varovee)$ *has the* $\varovee$*-disjunction property with respect to satisfiability, i.e., for every* $\varphi, \psi \in \mathrm{ML}(\varovee)$ *it holds that* $(\varphi \varovee \psi)$ *is satisfiable if and only if either* φ *is satisfiable or* ψ *is satisfiable.*

Proposition 24. *Let* φ *be an* ML*-formula and let* $k = |\mathrm{nbSubf}(\varphi)|$*. Then, if* φ *is satisfiable it is satisfiable in a Kripke model of size* $\leq 2^k$*.*

Proposition 25. $\mathrm{ML}(\varovee)$ *has the* $\varovee$*-disjunction property with respect to validity, i.e., for every* $\varphi, \psi \in \mathrm{ML}(\varovee)$ *it holds that* $(\varphi \varovee \psi)$ *is valid if and only if either* φ *is valid or* ψ *is valid.*

Proof. The direction from right to left is trivial. We will prove here the direction form left to right. Assume that $(\varphi \varovee \psi)$ is valid. For the sake of a contradiction, assume that neither φ nor ψ is valid. Thus there exist Kripke models K and K$'$, and teams T and T' of K and K$'$, respectively, such that $\mathrm{K}, T \not\models \varphi$ and $\mathrm{K}', T' \not\models \psi$. From Corollary 20 it follows that $\mathrm{K} \uplus \mathrm{K}', T \not\models \varphi$ and $\mathrm{K} \uplus \mathrm{K}', T' \not\models \psi$, where $\mathrm{K} \uplus \mathrm{K}'$ denotes the disjoint union of K and K$'$. Since the formulae of $\mathrm{ML}(\varovee)$ are downwards closed (Proposition 19), we conclude that $\mathrm{K} \uplus \mathrm{K}', T \cup T' \not\models \varphi$ and $\mathrm{K} \uplus \mathrm{K}', T \cup T' \not\models \psi$. Thus $\mathrm{K} \uplus \mathrm{K}', T \cup T' \not\models (\varphi \varovee \psi)$. This contradicts the fact that $(\varphi \varovee \psi)$ is valid.

Lemma 4. *Let* $\varphi \in \mathrm{ML}$ *and let* $k = |\mathrm{nbSubf}(\varphi)|$*. Then,* φ *is valid if and only if* $\mathrm{K}, w \models \varphi$ *holds for every Kripke model* $\mathrm{K} = (W, R, V)$ *and* $w \in W$ *such that* $|W| \leq 2^k$*.*

Lemma 5. *The decision problem whether a given formula of* ML *is valid in small models is in co*NP*.*

Proof. If a formula $\varphi \in \mathrm{ML}$ is not valid in small models, then there is some $k \leq |\varphi|$ and a pointed Kripke model K, w with domain of size k such that $\mathrm{K}, w \not\models \varphi$. The size of K, w is clearly polynomial in $|\varphi|$, and thus it can be guessed nondeterministically in polynomial time with respect to $|\varphi|$. The model checking problem for modal logic is in P, and thus $\mathrm{K}, w \not\models \varphi$ can be verified in polynomial time with respect to $|\mathrm{K}| + |\varphi|$ and thus in polynomial time with respect to $|\varphi|$.

Theorem 15. *The validity problem for* EMDL *is in* $\mathsf{NEXPTIME}^{\mathsf{NP}}$.

Proof. For deciding whether a given EMDL formula is valid, we give a nondeterministic exponential time algorithm that has an access to an NP oracle. The oracle is used to decide whether a given ML-formula is valid in small models. For each $\varphi \in$ EMDL, let φ^+ denote the equivalent exponential size $\mathrm{ML}(\varovee)$-formula from the proof of Proposition 22. Clearly φ^+ is computable from φ in exponential time. Furthermore let φ^* denote the $\mathrm{ML}(\varovee)$-formula of the form $\bigovee_{f \in F} \varphi^+_f$ of Proposition 22. Clearly the size of each $f \in F$ is polynomial with respect to the size of φ^+, and thus exponential with respect to the size of φ

We are now ready to give a $\mathsf{NEXPTIME}^{\mathsf{NP}}$ algorithm for the validity problem of EMDL. Let φ be an EMDL formula. First compute φ^+ from φ. Then nondeterministically guess a selection function g of φ^+ and compute φ^+_g; clearly this can be done in $\mathsf{NEXPTIME}$. We then give

$$\psi_g := \big(\bigwedge_{i \leq 2^{3 \times |\varphi|}} (p \vee \neg p) \big) \wedge \varphi^+_g$$

as an input to an NP oracle that decides whether the ML-formula ψ_g is valid in small models. Clearly ψ_g is computable from φ^+_g in exponential time with respect to the size of φ. The algorithm outputs "No" if the oracle outputs "No" and "Yes" if the oracle outputs "Yes". Clearly this algorithm is in $\mathsf{NEXPTIME}^{\mathsf{NP}}$.

Now by Proposition 22, φ is valid if and only if φ^* is valid, and furthermore, by Proposition 25, φ^* is valid if and only if φ^+_f is valid for some $f \in F$. By Proposition 22, $|\mathrm{nbSubf}(\varphi^+_f)| \leq 3 \times |\varphi|$, for every $f \in F$. Thus by Lemma 4, for every $f \in F$, φ^+_f is valid if and only if φ^+_f is true on all pointed models with domain of size at most $2^{3 \times |\varphi|}$. Thus, for every $f \in F$, φ^+_f is valid if and only if the formula

$$\big(\bigwedge_{i \leq 2^{3 \times |\varphi|}} (p \vee \neg p) \big) \wedge \varphi^+_f$$

is valid in small models. Therefore the algorithm decides the validity problem of EMDL.

Corollary 3. *The validity problem for* MDL *is in* $\mathsf{NEXPTIME}^{\mathsf{NP}}$.

Theorem 16. *The satisfiability problems for* MDL *and* EMDL *are in* $\mathsf{NEXPTIME}$.

Proof. For each $\varphi \in$ EMDL, let φ^+ denote the equivalent exponential size $\mathrm{ML}(\varovee)$-formula from the proof of Proposition 22. Clearly φ^+ is computable from φ in exponential time. Furthermore let φ^* denote the $\mathrm{ML}(\varovee)$-formula of the form $\bigovee_{f \in F} \varphi^+_f$ of

Proposition 22. Clearly the size of each $f \in F$ is polynomial with respect to the size of φ^+, and thus exponential with respect to the size of φ

We are now ready to give a NEXPTIME algorithm for the satisfiability problem of EMDL. Let φ be an EMDL formula. First compute φ^+ from φ. Then nondeterministically guess a selection function g of φ^+ and compute φ^+_g; clearly this can be done in NEXPTIME. We then quess a model $\mathfrak{A}$ such that $|A| \leq 2^{3\times|\varphi|}$, a point w of $\mathfrak{A}$, and check whether

$$\mathfrak{A}, w \models \varphi^+_g.$$

Since the size of $\mathfrak{A}$ is only exponential with respect to $|\varphi|$, the guessing of $\mathfrak{A}$ and w can be done in NEXPTIME with respect $|\varphi|$. Finally, since model checking of ML is in P, checking whether $\mathfrak{A} \models_X \varphi^+_g$ holds can be done in exponential time with respect to $|\varphi|$. Our algorithm for satisfiability returns the same value as the query $\mathfrak{A} \models_X \varphi^+_g$. Clearly this algorithm is in NEXPTIME.

Now by Proposition 22, φ is satisfiable if and only if φ^* is satisfiable, and furthermore, by Proposition 23, φ^* is satisfiable if and only if φ^+_f is satisfiable for some $f \in F$. By Proposition 22, $|\mathrm{nbSubf}(\varphi^+_f)| \leq 3 \times |\varphi|$, for every $f \in F$. Thus by Lemma 24, for every $f \in F$, φ^+_f is satisfiable if and only if $\mathfrak{B}, v \models \varphi^+_f$ holds for some pointed model $\mathfrak{B}, v$ with domain of size at most $2^{3\times|\varphi|}$. Therefore the algorithm decides the validity problem of EMDL.

We will next establish that the validity problem for MDL and EMDL coincide.

Definition 32. *Let φ be a formula of* MDL *or* EMDL. *The set* $\mathrm{depSub}(\varphi)$ *of* dependence subformulae *of φ is defined recursively as follows:*

$$\begin{aligned}
&\mathrm{depSub}(\neg p) := \mathrm{depSub}(p) := \emptyset, \quad \mathrm{depSub}(\mathrm{dep}(\varphi_1, \dots, \varphi_n)) := \{\varphi_1, \dots, \varphi_n\},\\
&\mathrm{depSub}(\varphi \vee \psi) := \mathrm{depSub}(\varphi \wedge \psi) := \mathrm{depSub}(\varphi) \cup \mathrm{depSub}(\psi)\,,\\
&\mathrm{depSub}(\Box\varphi) := \mathrm{depSub}(\Diamond\varphi) := \mathrm{depSub}(\varphi)\,.
\end{aligned}$$

Proposition 26. *Let $\Phi \subseteq \Psi$ be sets proposition of symbols such that Ψ is infinite. There exists a polynomial time function that associates each* $\mathrm{EMDL}(\Phi)$*-formula φ with a corresponding* $\mathrm{MDL}(\Psi)$*-formula φ^* such that φ is valid iff φ^* is valid.*

Proof. Fix $\varphi \in \mathrm{EMDL}(\Phi)$. We will show how to construct φ^*. Let k denote the modal depth of φ and stipulate that $\mathrm{depSub}(\varphi) = \{\psi_1, \dots \psi_n\}$. Let $p_1, \dots, p_n \in \Psi$ be fresh distinct proposition symbols that do not occur in φ. Define

$$\varphi^* := \Big(\bigwedge_{0 \leq i \leq k} \Box^i \bigwedge_{1 \leq j \leq n} (p_j \leftrightarrow \psi_j)\Big) \rightarrow \varphi^+,$$

where $\varphi \mapsto \varphi^+$ is defined recursively as follows. For (negated) proposition symbols the translation is the identity. For dependence atoms, Boolean connectives and modalities, we define

$$\begin{aligned}
&(\psi_1 \circ \psi_2) \mapsto (\psi_1^+ \circ \psi_2^+),\ \triangle\psi \mapsto \triangle\psi^+, \quad \text{where } \circ \in \{\wedge, \vee\} \text{ and } \triangle \in \{\Diamond, \Box\},\\
&\mathrm{dep}\big(\psi_{i_1}, \dots, \psi_{i_j}\big) \mapsto \mathrm{dep}\big(p_{i_1}, \dots, p_{i_j}\big)\,.
\end{aligned}$$

Clearly φ^* is an $\mathrm{MDL}(\Psi)$-formula and clearly the size of φ^* is only polynomial in the size of φ. It is straightforward to show that φ is valid iff φ^* is valid. The direction from right to left is trivial. The converse direction relies on the fact that MDL and EMDL are downwards closed.

Corollary 4. *Let $\mathcal{C}$ be a complexity class that is closed under polynomial time reductions. The validity problem for* MDL *is $\mathcal{C}$-complete iff the validity problem for* EMDL *is $\mathcal{C}$-complete.*

However, the expressive powers of MDL and EMDL differ.

Proposition 27. *There does not exist a formula of* MDL *that is equivalent with the* EMDL *formula* $\mathrm{dep}(\Diamond p)$.

Proof. Let $\Phi = \{p\}$ and $K = (W, R, V)$ be a Kripke model over Φ such that $W = \{a, b\}$, $R = \{(b, b)\}$ and $V(a) = V(b) = \{p\}$. We will define a translation $\varphi \mapsto \varphi^*$ from MDL-formulas to ML-formulas and show that on this model each formula $\varphi \in \mathrm{MDL}$ is equivalent to the ML formula φ^*. We define that

$$(\psi \wedge \theta)^* := (\psi^* \wedge \theta^*),\ (\psi \vee \theta)^* := (\psi^* \vee \theta^*),\ (\Box\psi)^* := \Box\psi^*,$$
$$(\Diamond\psi)^* := \Diamond\psi^*,\ p^* := p,\ (\neg p)^* := \neg p,\ \mathrm{dep}(p, \dots, p)^* := p.$$

Note that since $\Phi = \{p\}$ each dependence atom in φ is of type $\mathrm{dep}(p, \dots, p)$, and

$$K, T \models \mathrm{dep}(p, \dots, p)\,,$$

for all $T \subseteq W$. Hence it is a trivial inductive proof, that for all $T \subseteq W$ and $\varphi \in \mathrm{MDL}$

$$K, T \models \varphi \text{ iff } K, T \models \varphi^*. \tag{5}$$

We will now show that the EMDL-formula $\mathrm{dep}(\Diamond p)$ is not expressible in MDL. For contradiction, assume that there exists a MDL-formula ψ that is equivalent to $\mathrm{dep}(\Diamond p)$. Clearly $K, \{a\} \models \mathrm{dep}(\Diamond p)$ and $K, \{b\} \models \mathrm{dep}(\Diamond p)$. Hence $K, \{a\} \models \psi$ and $K, \{b\} \models \psi$. Therefore, by (5) $K, \{a\} \models \psi^*$ and $K, \{b\} \models \psi^*$. Now since ψ^* is an ML-formula by Proposition 18 we have that $K, \{a, b\} \models \psi^*$. Hence by (5) we have that $K, \{a, b\} \models \psi$ and finally that $K, \{a, b\} \models \mathrm{dep}(\Diamond p)$. This is a contradiction since clearly $K, \{a, b\} \not\models \mathrm{dep}(\Diamond p)$.

14 Bisimulation in team semantics

We start by defining k-bisimulation in the context of team semantics; the definition is directly based on the k-bisimulation relation $\rightleftarrows_k$ for Kripke semantics.

Definition 33. *Let (K, T) and (K', T') be team-pointed Kripke models, and $k \in \mathbb{N}$. We say that K, T and K', T' are* team k-bisimilar *and write $K, T\ [\rightleftarrows_k]\ K', T'$ if*

1. for every $w \in T$ there exists some $w' \in T'$ such that $K, w \rightleftarrows_k K, w'$, and
2. for every $w' \in T'$ there exists some $w \in T$ such that $K, w \rightleftarrows_k K, w'$.

It is well known that $K, w \rightleftarrows_k K', w'$ implies $K, w \rightleftarrows_n K', w'$ for all $n \leq k$. Using this it is easy to prove that the same holds also for team k-bisimilarity:

Lemma 6. *Let (K,T) and (K',T') be team-pointed Kripke models, and $k \in \mathbb{N}$. If $K, T\ [\rightleftarrows_k]\ K', T'$, then $K, T\ [\rightleftarrows_n]\ K', T'$ for all $n \leq k$.*

We say that a class of team-pointed Kripke models $\mathcal{K}$ is *closed under team k-bisimulation* if it satisfies the condition:

– $(K,T) \in \mathcal{K}$ and $K, T\ [\rightleftarrows_k]\ K', T'$ implies that $(K', T) \in \mathcal{K}$.

The next lemma shows that team k-bisimulation satisfies the natural counterparts of the back-and-forth properties that we used in defining $\rightleftarrows_k$, as well as a couple of other useful properties related to team semantics.

Lemma 7. *Let $k \in \mathbb{N}$, and assume that (K,T) and (K',T') are team-pointed Kripke models such that $K, T\ [\rightleftarrows_{k+1}]\ K', T'$. Then*

1. for every S s.t. $T[R]S$ there is S' s.t. $T'[R']S'$ and $K, S\ [\rightleftarrows_k]\ K', S'$;
2. for every S' s.t. $T'[R']S'$ there is S s.t. $T[R]S$ and $K, S\ [\rightleftarrows_k]\ K', S'$;
3. $K, S\ [\rightleftarrows_k]\ K', S'$ for $S = R[T]$ and $S' = R'[T']$;
4. for all $T_1, T_2 \subseteq T$ s.t. $T = T_1 \cup T_2$ there are $T_1', T_2' \subseteq T'$ s.t. $T' = T_1' \cup T_2'$, and $K, T_i\ [\rightleftarrows_{k+1}]\ K', T_i'$ for $i \in \{1,2\}$.

Proof. (i) Assume that $T[R]S$. We define

$$S' := \{v' \in R'[T'] \mid \exists v \in S : K, v \rightleftarrows_k K', v'\}.$$

We will first show that $K, S\ [\rightleftarrows_k]\ K', S'$. By the definition of S', we have $\forall v' \in S' \exists v \in S : K, v \rightleftarrows_k K', v'$. On the other hand, if $v \in S$, then there is $w \in T$ such that wRv. Furthermore, since $K, T\ [\rightleftarrows_{k+1}]\ K', T'$, there is $w' \in T'$ such that $K, w \rightleftarrows_{k+1} K', w'$, whence by the definition of $\rightleftarrows_{k+1}$, there is $v' \in W'$ such that $w'R'v'$ and $K, v \rightleftarrows_k K', v'$. By the definition of S', v' is in S'. Thus we see that $\forall v \in S \exists v' \in S' : K, v \rightleftarrows_k K', v'$.

To see that $T'[R']S'$ holds, note first that $S' \subseteq R'[T']$ by its definition. Assume then that $w' \in T'$. Since $K, T\ [\rightleftarrows_{k+1}]\ K', T'$, there is $w \in T$ such that $K, w \rightleftarrows_{k+1} K', w'$. Furthermore, since $T[R]S$, there is $v \in S$ such that wRv, and consequently there is $v' \in R'[w']$ such that $K, v \rightleftarrows_k K', v'$. By the definition of S' we have now $v' \in S'$. Thus we conclude that $w' \in R'^{-1}[S']$.

(ii) The claim is proved in the same way as (i).

(iii) If $v \in R[T]$, then there is $w \in T$ such that wRv. By the assumption $K, T\ [\rightleftarrows_{k+1}$ $]\ K', T'$, there is $w' \in T'$ such that $K, w \rightleftarrows_{k+1} K', w'$. Hence, there is v' such that $w'R'v'$ and $K, v \rightleftarrows_k K', v'$. As $w'R'v'$, we have $v' \in R'[T']$. Thus, we conclude that $\forall v \in R[T] \exists v' \in R'[T'] : K, v \rightleftarrows_k K', v'$. Using a symmetrical argument, we see that $\forall v' \in R'[T'] \exists v \in R[T] : K, v \rightleftarrows_k K', v'$.

(iv) Let $T_1, T_2 \subseteq T$ be such that $T = T_1 \cup T_2$. Define now

$$T_i' := \{w' \in T' \mid \exists w \in T_i : K, w \rightleftarrows_{k+1} K', w'\},$$

for $i \in \{1, 2\}$. Then by the definition of T_i', $\forall w' \in T_i' \exists w \in T_i : K, w \rightleftarrows_{k+1} K', w'$. On the other hand, if $w \in T_i$, then $w \in T$, whence there is $w' \in T'$ such that $K, w \rightleftarrows_{k+1} K', w'$. By the definition of T_i', then w' is in T_i'. Thus we conclude that $\forall w \in T_i \exists w' \in T_i' : K, w \rightleftarrows_{k+1} K', w'$, as desired.

Our goal is to prove that definability in $\mathrm{ML}(\varovee)$ can be characterized by downward closure and closure under team k-bisimulation. We already know that all $\mathrm{ML}(\varovee)$-definable classes are downward closed (see Proposition 19). The next step is to prove that $\mathrm{ML}(\varovee)$-definable classes are closed under team k-bisimulation for some k.

Theorem 17. *Let Φ be a set of proposition symbols, and let $\mathcal{K}$ be class of team-pointed Kripke models over Φ. If $\mathcal{K}$ is definable in* $\mathrm{ML}(\varovee)$*, then there is a $k \in \mathbb{N}$ such that $\mathcal{K}$ is closed under k-bisimulation.*

Proof. Assume that $\varphi \in \mathrm{ML}(\varovee)$. We prove by induction on φ that the class $\|\varphi\|$ is closed under k-bisimulation, where $k = \mathrm{md}(\varphi)$.

- Let $\varphi = p \in \Phi$, and assume that $K, T \models \varphi$ and $K, T\ [\rightleftarrows_k]\ K', T'$ for $k = 0$. Then $K, w \models p$ for all $w \in T$, and for each $w' \in T'$ there is $w \in T$ such that $K, w \rightleftarrows_0 K', w'$. Thus, for all $w' \in T'$, $K', w' \models p$, whence $K', T' \models \varphi$.
- The case $\varphi = \neg p$ is similar to the previous one.
- Let $\varphi = \psi \vee \theta$, and assume that $K, T \models \varphi$ and $K, T\ [\rightleftarrows_k]\ K', T'$, where $k = \mathrm{md}(\varphi) = \max\{\mathrm{md}(\psi), \mathrm{md}(\theta)\}$. Then there are $T_1, T_2 \subseteq T$ such that $T = T_1 \cup T_2$, $K, T_1 \models \psi$ and $K, T_2 \models \theta$.
  By Lemma 7(iv), there are subteams $T_1', T_2' \subseteq T'$ such that $T' = T_1' \cup T_2'$ and $K, T_i\ [\rightleftarrows_k]\ K', T_i'$ for $i \in \{1, 2\}$, whence $K, T_1\ [\rightleftarrows_m]\ K', T_1'$ and $K, T_2\ [\rightleftarrows_n]\ K', T_2'$, where $m = \mathrm{md}(\psi)$ and $n = \mathrm{md}(\theta)$. By induction hypothesis, $K', T_1' \models \psi$ and $K', T_2' \models \theta$. Thus, we conclude that $K', T' \models \varphi$.
- The cases $\varphi = \psi \wedge \theta$ and $\varphi = \psi \varovee \theta$ are straightforward.
- Let $\varphi = \Diamond\psi$, and assume that $K, T \models \varphi$ and $K, T\ [\rightleftarrows_k]\ K', T'$, where $k = \mathrm{md}(\varphi) = \mathrm{md}(\psi) + 1$. Then there is a team S on K such that $T[R]S$ and $K, S \models \psi$. By Lemma 7(i), there is a team S' such that $T'[R']S'$ and $K, S\ [\rightleftarrows_{k-1}]\ K', S'$. By induction hypothesis, $K', S' \models \psi$, and consequently $K', T' \models \varphi$.
- Let $\varphi = \Box\psi$, and assume that $K, T \models \varphi$ and $K, T\ [\rightleftarrows_k]\ K', T'$, where $k = \mathrm{md}(\varphi) = \mathrm{md}(\psi) + 1$. Then $K, R[T] \models \psi$, and by Lemma 7(iii), $K, R[T]\ [\rightleftarrows_{k-1}]\ K', R'[T']$. Thus, by induction hypothesis, $K', R'[T'] \models \psi$, and consequently $K', T' \models \varphi$.

Next we prove that downward closure and closure under team k-bisimulation are together a sufficient condition for $\mathrm{ML}(\varovee)$-definability.

Theorem 18. *Let Φ be a finite set of proposition symbols and let $\mathcal{K}$ be class of team-pointed Kripke models over Φ. Assume that $\mathcal{K}$ is downward closed and closed under k-bisimulation for some $k \in \mathbb{N}$. Then $\mathcal{K}$ is definable in* $\mathrm{ML}(\varovee)$.

Proof. Let φ be the formula

$$\bigovee_{(K,T)\in\mathcal{K}} \bigvee_{w\in T} \chi^k_{K,w},$$

where $\chi^k_{K,w}$ is the k-th Hintikka-formula of the pair (K, w). Note that since Φ is finite, there are only finitely many different Hintikka-formulas $\chi^k_{K,w}$. Thus, the disjunction $\bigvee_{w\in T}$ and the intuitionistic disjunction $\bigovee_{(K,T)\in\mathcal{K}}$ in φ are essentially finite, whence $\varphi \in \mathrm{ML}(\ovee)$. We will now prove that φ defines $\mathcal{K}$.

Assume first that $(K_0, T_0) \in \mathcal{K}$. By Proposition 18, $K_0, \{v\} \models \chi^k_{K_0,v}$ for each $v \in T_0$. Thus, $K_0, T_0 \models \bigvee_{w\in T_0} \chi^k_{K_0,w}$, and consequently, $K_0, T_0 \models \varphi$.

Assume for the other direction that $K_0, T_0 \models \varphi$. Then there is a pair $(K, T) \in \mathcal{K}$ such that $K_0, T_0 \models \bigvee_{w\in T} \chi^k_{K,w}$. Thus, there are subsets T_w, $w \in T$, of T_0 such that $T_0 = \bigcup_{w\in T} T_w$, and $K_0, T_w \models \chi^k_{K,w}$. By Proposition 18, $K_0, v \models \chi^k_{K,w}$ for every $v \in T_w$. Let $T' = \{w \in T \mid T_w \neq \emptyset\}$. Since $\mathcal{K}$ is downward closed, we have $(K, T') \in \mathcal{K}$. Observe now that for every $v \in T_0$ there is $w \in T'$ such that $K_0, v \models \chi^k_{K,w}$, and for every $w \in T'$ there is $v \in T_0$ such that $K_0, v \models \chi^k_{K,w}$. By Proposition 16 this means that $K, T'\ [\rightleftarrows_k]\ K_0, T_0$. Since $\mathcal{K}$ is closed under k-bisimulation, we conclude that $(K_0, T_0) \in \mathcal{K}$.

Putting Proposition 19, Theorem 17 and Theorem 18 together, we finally get the promised characterization for the expressive power of $\mathrm{ML}(\ovee)$.

Theorem 19. *If* K, T *and* K', T' *are team* k*-bisimilar and* φ *is a formula of* $\mathrm{ML}(\ovee)$, MDL *or* EMDL *of modal depth at most* k*, then*

$$\mathrm{K}, T \models \varphi \quad \Leftrightarrow \quad \mathrm{K}', T' \models \varphi.$$

Theorem 20. *A class* $\mathcal{C}$ *of team-pointed Kripke models is definable by* $\mathrm{ML}(\ovee)$ *iff* $\mathcal{C}$ *is downward closed and closed under team* k*-bisimulation for some* k*.*

Theorem 21. *A class* $\mathcal{C}$ *of team-pointed Kripke models is definable by* $\mathrm{ML}(\ovee)$ *iff* $\mathcal{C}$ *is first-order definable, downward closed, and closed under team bisimulation.*

15 EMDL is equivalent to ML($\ovee$)

By Proposition 22, we know that $\mathrm{ML}(\ovee)$ is at least as expressive as EMDL. In this section we show that the converse is also true.

Theorem 22. $\mathrm{ML}(\ovee) \leq \mathrm{EMDL}$.

Before proving Theorem 22, we introduce some auxiliary concepts, and prove a couple of lemmas concerning them.

Let Ψ be a finite set of $\mathrm{ML}(\Phi)$-formulas, and let K be a Kripke model over Φ and w a state in K. The *Ψ-type* of w in K is defined as

$$\mathrm{tp}_\Psi(K, w) := \{\psi \in \Psi \mid K, w \models \psi\}.$$

Furthermore, the *Ψ-type* of a team T of K is just the set of Ψ-types of its elements:

$$\mathrm{TP}_{\Psi}(K,T) := \{\mathrm{tp}_{\Psi}(K,w) \mid w \in T\}.$$

Each Ψ-type $\Gamma \subseteq \Psi$ can be defined by a formula: Let

$$\theta_{\Gamma} := \bigwedge_{\psi \in \Gamma} \psi \wedge \bigwedge_{\psi \in \Psi \setminus \Gamma} \psi^{\perp}$$

where $\psi^{\perp}$ denotes the formula obtained from $\neg\psi$ by pushing the negations in front of proposition symbols. Then it is easy to see that $\mathrm{tp}_{\Psi}(K,w) = \Gamma$ if and only if $K, w \models \theta_{\Gamma}$.

Lemma 8. *Assume that $(K,T), (K',T') \in \mathcal{KT}(\Phi)$, and let Ψ be a finite set of* $\mathrm{ML}(\Phi)$*-formulas.*

1. *For each $\psi \in \Psi$, $K, T \models \psi$ if and only if $\psi \in \bigcap \mathrm{TP}_{\Psi}(K,T)$.*
2. *If $K, T \models \bigovee \Psi$ and $\mathrm{TP}_{\Psi}(K',T') \subseteq \mathrm{TP}_{\Psi}(K,T)$, then $K', T' \models \bigovee \Psi$.*

Proof. (ii) Assume that $K, T \models \bigovee \Psi$ and $\mathrm{TP}_{\Psi}(K',T') \subseteq \mathrm{TP}_{\Psi}(K,T)$. Thus, $K, T \models \psi$ for some $\psi \in \Psi$, and by claim (i), $\psi \in \bigcap \mathrm{TP}_{\Psi}(K,T)$. Since $\mathrm{TP}_{\Psi}(K',T') \subseteq \mathrm{TP}_{\Psi}(K,T)$, it follows that $\psi \in \bigcap \mathrm{TP}_{\Psi}(K',T')$. Thus, $K', T' \models \psi$, and consequently $K', T' \models \bigovee \Psi$.

Consider next the formula $\gamma := \bigwedge_{\psi \in \Psi} \mathrm{dep}(\psi)$. It says that the truth value of each ψ in Ψ is constant, whence $K, T \models \gamma$ if and only if $|\mathrm{TP}_{\Psi}(K,T)| \leq 1$. Define now recursively

$$\gamma^0 := p \wedge \neg p, \qquad \gamma^{k+1} := (\gamma^k \vee \gamma)$$

It is straightforward to show by induction that for all $k \in \mathbb{N}$, $K, T \models \gamma^k$ if and only if $|\mathrm{TP}_{\Psi}(K,T)| \leq k$.

Lemma 9. *Let Ψ be a finite set of* $\mathrm{ML}(\Phi)$*-formulas. If $(K,T) \in \mathcal{KT}(\Phi)$, $T \neq \emptyset$, then there is a formula $\xi_{K,T} \in \mathrm{EMDL}(\Phi)$ such that for every $(K',T') \in \mathcal{KT}(\Phi)$*

$$K', T' \models \xi_{K,T} \quad \Longleftrightarrow \quad \mathrm{TP}_{\Psi}(K,T) \not\subseteq \mathrm{TP}_{\Psi}(K',T').$$

Proof. Let $|\mathrm{TP}_{\Psi}(K,T)| = k+1$. We define

$$\xi_{K,T} := \Big(\bigvee_{\Gamma \in X} \theta_{\Gamma}\Big) \vee \gamma^k,$$

where $X = \mathcal{P}(\Psi) \setminus \mathrm{TP}_{\Psi}(K,T)$. Now given a pair $(K',T') \in \mathcal{KT}(\Phi)$ we have

$$\begin{aligned} K', T' \models \xi_{K,T} \iff & \text{ there are } T_1, T_2 \text{ such that } T_1 \cup T_2 = T' \text{ and} \\ & \qquad \mathrm{TP}_{\Psi}(K',T_1) \subseteq X \text{ and } |\mathrm{TP}_{\Psi}(K',T_2)| \leq k \\ \iff & |\mathrm{TP}_{\Psi}(K,T) \cap \mathrm{TP}_{\Psi}(K',T')| \leq k \\ \iff & \mathrm{TP}_{\Psi}(K,T) \not\subseteq \mathrm{TP}_{\Psi}(K',T'). \end{aligned}$$

Proof. **of Theorem 22**. Let φ be an $\mathrm{ML}(\ovee)(\Phi)$-formula. By the normal form derived in the proof of Theorem 18, we may assume that φ is of the form $\bigovee \Psi$, where Ψ is a finite set of $\mathrm{ML}(\Phi)$-formulas.

Let η be the formula

$$\bigwedge_{(K,T)\in\overline{\|\varphi\|}} \xi_{K,T},$$

where $\overline{\|\varphi\|} = \mathcal{KT}(\Phi) \setminus \|\varphi\|$ and $\xi_{K,T}$ is as in Lemma 9. Since Ψ is finite, there are finitely many different formulas of the form $\xi_{K,T}$. Thus, the conjunction in η is essentially finite, and hence η is in EMDL.

To prove that $\|\eta\| = \|\varphi\|$, let $(K_0, T_0) \in \mathcal{KT}(\Phi)$. Assume first that $(K_0, T_0) \in \|\varphi\|$, and consider any pair $(K,T) \in \overline{\|\varphi\|}$. It follows from Lemma 8 that $\mathrm{TP}_\Psi(K,T) \not\subseteq \mathrm{TP}_\Psi(K_0, T_0)$, whence by Lemma 9, $K_0, T_0 \models \xi_{K,T}$. Thus we see that $(K_0, T_0) \in \|\eta\|$.

Assume then that $(K_0, T_0) \notin \|\varphi\|$. Since $\mathrm{TP}_\Psi(K_0, T_0) \subseteq \mathrm{TP}_\Psi(K_0, T_0)$, it follows from Lemma 9 that $K_0, T_0 \not\models \xi_{K_0,T_0}$. Thus we conclude that $(K_0, T_0) \notin \|\eta\|$.

Combining Proposition 22 and Theorem 22, we see that the expressive power of EMDL and $\mathrm{ML}(\ovee)$ coincide. This means that the characterization for the expressive power of $\mathrm{ML}(\ovee)$ given in Theorem 20 is true for EMDL, too.

Corollary 5. EMDL $\equiv \mathrm{ML}(\ovee)$.

Corollary 6. *A class $\mathcal{K} \subseteq \mathcal{KT}(\Phi)$ is definable in* EMDL *if and only if $\mathcal{K}$ is downward closed and there is a $k \in \mathbb{N}$ such that $\mathcal{K}$ is closed under k-bisimulation.*

16 Propositional dependence logic

In this section we define the basic concepts and results related to propositional dependence logic.

16.1 Syntax and semantics

We denote by PROP the set of all proposition symbols. Let D be a finite, possibly empty, subset of PROP. A function $s : D \to \{0,1\}$ is called an *assignment*. A set X of assignments $s : D \to \{0,1\}$ is called a *propositional team*. The set D is the *domain* of X. Note that the empty team $\emptyset$ does not have a unique domain; any subset of PROP is a domain of the empty team. We denote by 2^D the set of *all assignments* $s : D \to \{0,1\}$.

Let Φ be a set of proposition symbols. The syntax for propositional logic $\mathrm{PL}(\Phi)$ is defined as follows.

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

By $\models_{\mathrm{PL}}$, we denote the ordinary satisfaction relation of propositional logic defined via assignments as usual. We will now give team semantics for propositional logic. We will see below (Proposition 28) that the team semantics and the ordinary semantics for propositional logic defined via assignments, in a rather strong sense, coincide.

Definition 34. *Let Φ be a set of atomic propositions and let X be a propositional team. The satisfaction relation $X \models \varphi$ is defined as follows. Note that, we always assume that the proposition symbols that occur in φ are also in the domain of X.*

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

Proposition 28. *Let φ be a formula of propositional logic and let X be a propositional team. Then $X \models \varphi$ iff $\forall s \in X : s \models_{PL} \varphi$.*

The syntax of *propositional dependence logic* $\mathrm{PD}(\Phi)$ is obtained by extending the syntax of $\mathrm{PL}(\Phi)$ by the grammar rules

$$\varphi ::= \mathrm{dep}(p_1, \dots, p_n, q)\,, \qquad \text{where } p_1, \dots, p_n, q \in \Phi.$$

The intuitive meaning of the *propositional dependence atom* $\mathrm{dep}(p_1, \dots, p_n, q)$ is that, inside a team the truth value of the proposition symbol q is completely determined by the truth values of the proposition symbols $p_1, \dots, p_n$. The semantics for the propositional dependence atom is defined as follows:

$$\begin{aligned}
X \models \mathrm{dep}(p_1, \dots, p_n, q) \quad \Leftrightarrow \quad &\forall s, t \in X : s(p_1) = t(p_1), \dots, s(p_n) = t(p_n)\\
&\text{implies that } s(q) = t(q).
\end{aligned}$$

The next proposition is very useful.

Proposition 29 (Downwards closure). *Let φ be a* PD*-formula and let $Y \subseteq X$ be propositional teams. Then $X \models \varphi$ implies $Y \models \varphi$.*

16.2 Satisfiability, validity, and model checking in team semantics

We will next define satisfiability and validity in the context of propositional team semantics: A formula φ of propositional dependence logic is *satisfiable*, if there exists a non-empty propositional team X such that $X \models \varphi$. A formula φ of propositional dependence logic is *valid*, if $X \models \varphi$ holds for every team X such that the proposition symbols that occur in φ are in the domain of X.

The satisfiability problem and the validity problem is then defined in the obvious manner: Given a binary encoding of a formula, decide whether the formula is satisfiable (valid, respectively). The variant of the model checking problem that is the following: Given binary encodings of a formula φ of propositional dependence logic and of a finite propositional team X, decide whether $X \models \varphi$.

Theorem 23. *The model checking problem for* PD *is* NP*-complete.*

Theorem 24. *The satisfiability problem for* PD *is* NP*-complete.*

Proof. Hardness directly from SAT. Inclusion straightforward algorithm.

Recall that 2^D is the set of exactly all assignments $s : D \to \{0,1\}$. The following lemma follows directly from the fact that PD is downward closed, i.e., Proposition 29.

Lemma 10. *Let φ be a* PD*-formula and let D be the set of exactly all proposition symbols that occur in φ. Then φ is valid if and only if $2^D \models \varphi$.*

We shall first show that the validity problem for PD is in NEXPTIME.

Lemma 11. *The validity problem for* PD *is in* NEXPTIME.

Proof. Let φ be a PD-formula and let D denote the set of proposition symbols that occur in φ. By Lemma 10, φ is valid if and only if $2^D \models \varphi$. The size of 2^D is $2^{|D|}$ and thus $\leq 2^{|\varphi|}$. Therefore 2^D can be clearly constructed from φ in exponential time. By Theorem 23, there exists an NP algorithm (with respect to $|2^D| + |\varphi|$) for checking whether $2^D \models \varphi$. Clearly this algorithm works in NEXPTIME with respect to the size of φ. Therefore, we conclude that the validity problem for PD is in NEXPTIME.

We will then show that the validity problem for PD is NEXPTIME-hard. We show a reduction from the validity problem of DQBF to the validity problem of PD.

Lemma 12. *The validity problem for* PD *is* NEXPTIME*-hard.*

Proof. We will associate each DQBF-formula μ with a corresponding PD-formula φ_μ. Let

$$\mu = \big(\forall p_1 \dots \forall p_n \exists q_1 \dots \exists q_k\, \theta, (C_1, \dots, C_k)\big)$$

be a DQBF-formula. Recall that θ is a formula of propositional logic and that by $\boldsymbol{c}_i$ we denote the canonically ordered tuple of the variables in C_i. For each set of Boolean variables C_i, $i \leq k$, we stipulate that $\boldsymbol{c}_i = (p_{i_1}, \dots, p_{i_{n_i}})$. Thus n_i denotes the cardinality of C_i. We then denote by D_μ the set of propositional variables in μ, i.e., $D_\mu := \{p_1, \dots, p_n, q_1, \dots, q_k\}$. Define

$$\varphi_\mu := \theta \vee \bigvee_{i \leq k} \operatorname{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big)\,.$$

We will show that μ is valid if and only if the PD-formula φ_μ is valid. Since the validity problem for DQBF is NEXPTIME-complete (Theorem 14) and φ_μ is polynomial with respect to μ, it follows that the validity problem for PD is NEXPTIME-hard. By Lemma 10, it suffices to show that μ is valid if and only if $2^{D_\mu} \models \varphi_\mu$.

Assume first that μ is valid, i.e., that $\forall p_1 \dots \forall p_n \exists q_1 \dots \exists q_k\, \theta$ is valid under the constraint $(C_1, \dots, C_k)$. Therefore, for each $i \leq k$, there exists a function $f_i : \{0,1\}^{|C_i|} \to \{0,1\}$ such that

$$\text{for every assignment } s : \{p_1, \dots, p_n\} \to \{0,1\} : \quad s' \models \theta, \tag{6}$$

where $s' := s\Big(q_1 \mapsto f_1\big(s(\boldsymbol{c}_1)\big), \dots, q_k \mapsto f_k\big(s(\boldsymbol{c}_k)\big)\Big)$. Our goal is to show that

$$2^{D_\mu} \models \theta \vee \bigvee_{i \le k} \mathrm{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big) .$$

It suffices to show that there exist some $Y, Z_1, \dots Z_k \subseteq 2^{D_\mu}$ such that $Y \cup Z_1 \cup \dots \cup Z_k = 2^{D_\mu}$, $Y \models \theta$, and $Z_i \models \mathrm{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big)$, for each $i \le k$. We define the team Z_i, for each $i \le k$, by using the function f_i. Define

$$Z_i := \{s \in 2^{D_\mu} \mid s(q_i) \neq f_i\big(s(p_{i_1}), \dots, s(p_{i_{n_i}})\big)\}, \quad \text{for each } i \le k.$$

Since propositional variables have only 2 possible values, we conclude that, for each $i \le k$, $Z_i \models \mathrm{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big)$. Thus

$$\bigcup_{1 \le i \le k} Z_i \models \bigvee_{i \le k} \mathrm{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big) . \tag{7}$$

Note that $s(q_i) = f_i\big(s(p_{i_1}), \dots, s(p_{i_{n_i}})\big)$ holds for every $s \in (2^{D_\mu} \setminus Z_i)$ and every $i \le k$. Define then

$$Y := 2^{D_\mu} \setminus \bigcup_{1 \le i \le k} Z_i.$$

Clearly, for every $s \in Y$ and $i \le k$, it holds that $s(q_i) = f_i\big(s(p_{i_1}), \dots, s(p_{i_{n_i}})\big)$. Recall that, for each $i \le k$, $\boldsymbol{c}_i = (p_{i_1}, \dots, p_{i_{n_i}})$. Thus from (6), it follows that $s \models \theta$, for every $s \in Y$. Since θ is a PL-formula, we conclude by Proposition 28 that $Y \models \theta$. From this together with (7), we conclude that $2^{D_\mu} \models \varphi_\mu$.

Assume then that $2^{D_\mu} \models \varphi_\mu$. Thus there exist some $Y_1, \dots, Y_k, Z$ such that $Y_1 \cup \dots \cup Y_k \cup Z = 2^{D_\mu}$, $Z \models \theta$, and

$$Y_i \models \mathrm{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big) , \text{ for each } i \le k.$$

Assume that we have picked $Y_1, \dots, Y_k, Z$ such that Z is minimal. We will show that then $Z \models \mathrm{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big)$, for each $i \le k$. Assume for the sake of a contradiction that, for some $i \le k$, there exist $s, t \in Z$ such that

$$s(p_{i_1}) = t(p_{i_1}), \dots, s(p_{i_{n_i}}) = t(p_{i_{n_i}}) \text{ but } s(q_i) \neq t(q_i).$$

Clearly either $Y_i \cup \{s\} \models \mathrm{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big)$ or $Y_i \cup \{t\} \models \mathrm{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big)$. This contradicts the fact that Z was assumed to be minimal.

We will then show that, for every $a_1, \dots, a_n \in \{0, 1\}$, there exists some assignment s in Z that expands

$$(p_1, \dots, p_n) \mapsto (a_1, \dots, a_n).$$

Fix $a_1, \dots, a_n \in \{0, 1\}$. Now, for every $i \le k$, since $Y_i \models \mathrm{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big)$, it follows that for any two $s', s'' \in Y_i$ that expand $(p_1, \dots, p_n) \mapsto (a_1, \dots, a_n)$, it holds that $s'(q_i) = s''(q_i)$. Thus, for each $i \le k$, there exists a truth value $b_i \in \{0, 1\}$ such that there is no expansions of $(p_1, \dots, p_n, q_i) \mapsto (a_1, \dots, a_n, b_i)$ in Y_i. Therefore, the assignment $(p_1, \dots, p_n, q_1, \dots, q_k) \mapsto (a_1, \dots, a_n, b_1, \dots, b_k)$ is not in Y_i, for any $i \leq k$. Thus the assignment $(p_1, \dots, p_n, q_1, \dots, q_k) \mapsto (a_1, \dots, a_n, b_1, \dots, b_k)$ is in Z. Hence, for every $a_1, \dots, a_n \in \{0, 1\}$, there exists some expansion of $(p_1, \dots, p_n) \mapsto (a_1, \dots, a_n)$ in Z.

Now, for each $i \leq k$, we define a function $f_i : \{0, 1\}^{n_i} \to \{0, 1\}$ as follows. Define

$$f_i(a_1, \dots, a_{n_i}) := s(q_i),$$

where s is an assignment in Z that expands $(p_{i_1}, \dots p_{i_{n_i}}) \mapsto (a_1, \dots, a_{n_i})$. Since $Z \models \operatorname{dep}\big(p_{i_1}, \dots, p_{i_{n_i}}, q_i\big)$, for each $i \leq k$, the functions f_i are well defined. Recall that, for each $i \leq k$, $\boldsymbol{c}_i = (p_{i_1}, \dots p_{i_{n_i}})$. Now since θ is syntactically a PL-formula and since $Z \models \theta$, it follows from Proposition 28 that $s' \models \theta$, for each $s' \in Z$. Clearly the functions f_i, for $i \leq k$, are as required in (6). Thus we conclude that (6) holds. Thus μ is valid.

By Lemmas 11 and 12, we obtain the following:

Theorem 25. *The validity problem for* PD *is* NEXPTIME*-complete.*

Corollary 7. *The validity problem for* MDL *and* EMDL *is* NEXPTIME*-hard.*

Theorem 26. *The satisfiability problem for* MDL *and* EMDL *is* NEXPTIME*-hard.*

Proof. The proof is similar to that of Theorem 12. First force exponential size model with 2^{D_μ} as leaves, and then a small adjustment of the proof of Theorem 12.