Complexity in Dependence Logic

Diplomarbeit
von
Johannes Ebbing

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

July 14, 2010

Erklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt habe.

Johannes Ebbing

Abstract. In this thesis, we will investigate the complexity of model checking on an extended version of modal logic ML, called modal dependence logic (MDL). MDL is the extension of ML by the so called *dependence atom* denoted by $=(\cdot)$. It was introduced in 2007 by Jouko Väänänen. Let $p_1, \dots, p_n, q$ be atomic propositions, then $=(p_1, \dots, p_n; q)$ means that the value of q is determined solely by $p_1, \dots, p_n$. The model checking problem for modal logic in general is solvable in polynomial time. We will see that the MDL model checking problem ($\mathsf{MDL\text{-}MC}$) in gerneral is NP-complete. In the second part we will restrict the set of operators $\{\neg(\text{atomic}), \vee, \wedge, \Box, \Diamond, =(\cdot)\}$ and we will show, that the complexity of $\mathsf{MDL\text{-}MC}$ stays NP-complete if we leave at least $\{\vee, \wedge, =(\cdot)\}$, $\{\vee, \Box, =(\cdot)\}$ or $\{\Diamond, =(\cdot)\}$ in set of operators. Furthermore we will show that there are some tractable cases. $\mathsf{MDL\text{-}MC}$ is tractable if we leave out either the $=(\cdot)$ atom or if we leave out the disjunction $\vee$ and the $\Diamond$ operator. The case where we have some subset of $\{\neg, \vee, =(\cdot)\}$ is only shown for dependece atoms of the form $=(p_1, \dots, p_k; q)$ where $k \in \mathbb{N}$ is fixed in advance.

Acknowledgements

First of all I would like to thank Professor Heribert Vollmer for the supervision and the support of my diploma thesis. I thank Michael Thomas, Arne Meier, and Peter Lohmann for their help, especially I thank Peter for his support in developing the results of this thesis and for proofreading.
I thank my wife Melanie and my parents for motivating and supporting me during my studies of mathematics. For proofreading I also want to thank Ulrich von der Ohe, Kai Lämmle and my wife Melanie.
But most of all I want to thank the one who made all this possible. My God Jesus Christ.

Männer werden müde und matt, und Jünglinge straucheln und fallen;
aber die auf den HERRN harren, kriegen neue Kraft,
dass sie auffahren mit Flügeln wie Adler, dass sie laufen und nicht matt werden,
dass sie wandeln und nicht müde werden.

Jesaja 40 Vers 30 + 31

Contents

1 Introduction

Model Checking is a method to automatically verify systems. It has become more important to computer science because many systems such as integrated circuits can be modeled as so called *Kripke structures* which are also called *transition systems* in the literature. The behavior of systems is described by logics like CTL (computation tree logic) or LTL (linear temporal logic). For further information about model checking over CTL see [CGP99].
In this thesis we will consider a new kind of logic, the so called modal dependence logic (MDL). MDL is an extension of modal logic (ML), where modal logic is extended by the so called *dependence atom* denoted by $=(p_1, \ldots, p_n; q)$. The concept of dependence in first order logic was first introduced by Jouko Väänänen in [Vää07]. Later he combined the concept of dependence with modal logic in [Vää08]. Modal logic itself can be seen as a restriction of CTL where only the operators **AX** (which corresponds to $\Box$ in this thesis) and **EX** (which corresponds to $\Diamond$ in this thesis) are allowed.
With the dependence atom, we can make statements about dependencies in different worlds or situations. $=(p_1, \ldots, p_n; q)$ means, that the value of q only depends on the vector $(p_1, \ldots, p_n)$. For this purpose we consider sets of situations or worlds, the so called *teams*. We use the words *world* and *situation* synonymously.
For example, if we play roulettes, we play a round, setting on red and we notice that we win. In this case, it makes no sense, to state a dependence on this single round like *"The win of a round is determined by whether I set my money on the color red."* as a formula $=(\text{set money on red}; \text{win})$. On a single round this would be always true. Of course it is possible, to set the money on red and to lose the round. So we would have to make more observations on the game to verify, whether the formula holds or not. Consider the table which represents a team of game situations

round	set money on red	win
1	true	true
2	false	false
3	true	false

In round 3 our formula would not hold any longer. What we do is checking a formula on a given model, in this case a team of game situations.
In the so called *satisfiability problem* of modal dependence logic (MDL-SAT) we want to determine whether a given MDL formula is satisfiable. Sevenster showed in [Sev09] that the satisfiability problem is complete for nondeterministic exponential time and in [LV10] Peter Lohmann and Heribert Vollmer give results about operator fragments of MDL-SAT obtained by restricting the set of allowed operators.
In this thesis, we will consider the computational complexity of *model checking* over

MDL. Because ML is a restriction of CTL, we have that MDL is a restriction of a kind of CTL where we add the dependence atom to get DCTL the so called *dependence* CTL. In this way this thesis can be seen as a first approach to examine fragments of DCTL. In [CGP99] it is shown, that model checking over CTL is tractable. We will show, that it is in general no longer tractable with dependence. In the following table, we have listed the comlexity of model checking over MDL. The MDL formulas are built over a set of operators $\{\neg, \vee, \wedge, \Box, \Diamond, =\!(\cdot)\}$. In the table a "+" means, that we allow this operator in the formula, a "−" means, that all considered formulae do not contain this operator, and an "∗" means that it does not matter whether this operator is in the formula or not. In the first row, we have all operators allowed, what would correspond to the MDL model checking problem (MDL-MC) where all operators are allowed. The second row corresponds to the model checking problem on operators without $=\!(\cdot)$, i.e. model checking on modal logic formulas, which is in P as we mentioned above.

Operators						**Complexity**	**Reference**
$\neg$	$\vee$	$\wedge$	$\Box$	$\Diamond$	$=\!(\cdot)$		
$+$	$+$	$+$	$+$	$+$	$+$	NP-complete	Theorem 3.1
$*$	$*$	$*$	$*$	$*$	$-$	$\in P$	Theorem 4.8
$*$	$+$	$+$	$*$	$*$	$+$	NP-complete	Theorem 3.1
$*$	$+$	$*$	$+$	$*$	$+$	NP-complete	Theorem 4.4
$*$	$*$	$*$	$*$	$+$	$+$	NP-complete	Theorem 4.2
$*$	$-$	$*$	$*$	$-$	$*$	$\in P$	Theorem 4.6
$*$	$+$	$-$	$-$	$-$	$(+)$[1]	$\in P$	Theorem 4.20
$*$	$+$	$-$	$-$	$-$	$+$	open	

In the following part of this thesis, we restrict the set of operators $\{\neg, \vee, \wedge, \Box, \Diamond, =\!(\cdot)\}$ to so called *operator fragments* to determine the complexity of model checking over these operator fragments. It turns out, that we have three operators $\vee, \Diamond$ and $=\!(\cdot)$ which make the problem NP-hard and we have to banish at least one of them to get model checking complexities that are tractable.

[1]We limit the dependence atom to constant length, i.e. all dependence atoms are of the form $=\!(p_1, \dots, p_k; q)$ where $k \in \mathbb{N}$ is fixed and $p_1, \dots, p_k, q$ are atomic propositions.

2 Preliminaries

At first we will examine the fundamentals which will be used in this thesis. We begin by the definition of *modal logic* (ML). We explain *Kripke structures* over which modal logic is interpreted. As we will see, modal logic itself can be interpreted as a special restriction of *computation tree logic* (CTL). The next step is to introduce the concept of dependence. We get a new operator, called the *dependence atom*, denoted by $=(p_1, \ldots, p_n; q)$, where $p_1, \ldots, p_n, q$ are atomic propositions. That means, that the value of q only depends on the p_i. We combine the concept of dependence with modal logic, and we obtain a new form of logic, called *modal dependence logic* (MDL). Then we will introduce the concept of model checking. Model checking works on so called *initial sets* of states in the Kripke structure. The task is to determine whether a given formula is satisfied by a given initial set of the Kripke structure.

Definition 2.1. (Kripke structure)
A *Kripke structure* (or *frame*) over a set of atomic propositions AP (or AP-Kripke structure) is a tuple $W = (S, R, \pi)$, where

1. $S \neq \emptyset$
2. $R \subseteq S \times S$
3. $\pi : S \to \mathcal{P}(AP)$

If $s \in S$, s is called *world*, *state*, *node*, or *situation*. We call S the *set of worlds*. R is called the *transition relation* between the worlds, and π is called the *labeling function*.

2.1 Modal Logic

As CTL formulas, modal logic formulas describe properties of Kripke structures. Modal logic can be considered as an extension of propositional logic. Similar to propositional logic, the formulas of modal logic are built from a set of atomic propositions AP, the operators $\neg, \vee, \wedge$, but moreover ML also contains temporal quantifiers $\Box, \Diamond$. With these operators, we can make statements that a ML formula can be *possibly* true (denoted by the $\Diamond$ operator) or that a formula will be true *in any case* (denoted by $\Box$).
Modal logic also can be understood as a constrained form of CTL. In CTL, we have more quantifiers, allowing us to make statements e.g. whether a formula will be true at some state in the future. In ML we only consider quantifiers ($\Box$ and $\Diamond$), that allow to make statements about the next step[2].

[2]For this reason in some literature $\Box$ is called AX that means for **a**ll states in the ne**x**t step, and $\Diamond$ is called EX, which means there **e**xist states in the ne**x**t step.

2.1.1 Syntax

Definition 2.2. (The syntax of modal logic)
Let AP be a set of atomic propositions and $p \in AP$. The syntax of modal logic formulas is defined by the grammar:

$$\varphi ::= p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Box \varphi \mid \Diamond \varphi$$

Note, that $\neg$ is only used for atomic propositions in ML.

2.1.2 Semantics

Definition 2.3. (Truth of teams)
Let $W = (S, R, \pi)$ an AP-Kripke structure, $A \subseteq S$ a set of initial states, and φ a ML formula over AP and $p \in AP$. A *successor* of $T \subseteq S$ is an element $r \in R[T] := \{a \in S | \exists a' \in T, (a', a) \in R\}$. Inductive over the structure of φ we define

$$\begin{array}{lll}
W, A \models p & :\Leftrightarrow & p \in \pi(s), \forall s \in A \\
W, A \models \neg p & :\Leftrightarrow & p \notin \pi(s), \forall s \in A \\
W, A \models \varphi \vee \psi & :\Leftrightarrow & \exists A_1, A_2 \subseteq A, A_1 \cup A_2 = A, \\
 & & W, A_1 \models \varphi \text{ and } W, A_2 \models \psi \\
W, A \models \varphi \wedge \psi & :\Leftrightarrow & W, A \models \varphi \text{ and } W, A \models \psi \\
W, A \models \Box \varphi & :\Leftrightarrow & W, R[A] \models \varphi \\
W, A \models \Diamond \varphi & :\Leftrightarrow & \exists A' \subseteq R[A] \text{ s. t. } A' \models \varphi \text{ and} \\
 & & \forall a \in A \exists a' \in A' \text{ with } (a, a') \in R.
\end{array}$$

We call a formula φ with $K, A \models \varphi$ *K, A-true* or *true on K and A*. On a K, A-true formula φ, we also say K, A *satisfy* φ. When is is clear from the context on which Kripke structure we operate, we leave out K and only say φ *is true on A* or $A \models \varphi$.

Note that the definition of $\vee$ differs from the "classical" disjunction, defined by

$$K, A \models \varphi \tilde{\vee} \psi \text{ iff } K, A \models \varphi \textit{ or } K, A \models \psi.$$

It holds that $K, A \models \varphi \tilde{\vee} \psi \Rightarrow K, A \models \phi \vee \psi$, because the subsets A_1 and A_2 are chosen either $A_1 = A$ and $A_2 = \emptyset$ in the case of $K, A \models \varphi$, or $A_1 = \emptyset$ and $A_2 = A$ in the case of $K, A \models \psi$. As we can easily verify $K, \emptyset \models \psi$ for any modal logic formula ψ holds.

2.1.3 Example

We interpret the states of the Kripke structure as situations in a game. The elements of the transition relation can be interpreted as moves. In this way $\Box f$ would mean *"For every move a player does, f will be true"*. The other quantifier $\Diamond f$ means *"There is a way to move so that f will be true."*. Initially, we designate which player begins moving, and then we process the formula step by step. We can interpret this formula in the "Nine Men's Morris" game in the placing phase. Consider the following modal logic formula

$$F := \Box \Diamond \varphi.$$

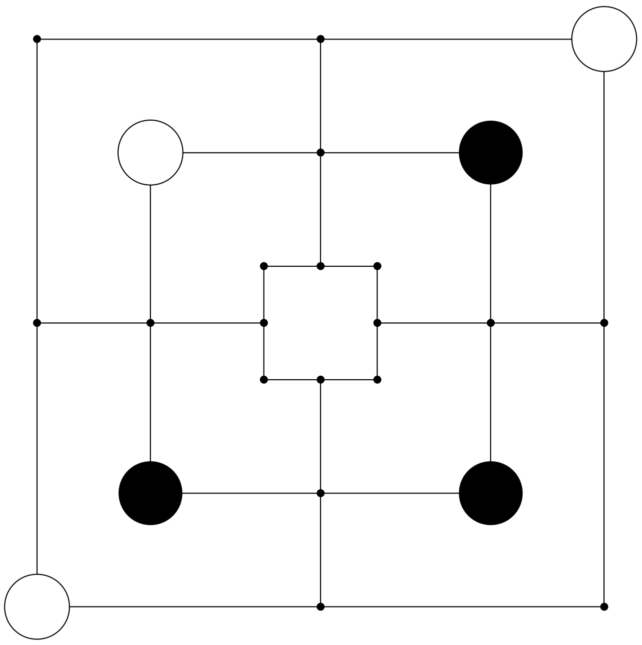


Figure 2.1: A “Nine Men’s Morris” game, with a double bind in the placing phase.

When it is white’s turn F would mean:

For every move white does, there is a way for black to get a mill.

In this case, φ is true if and only if there is a mill of black tokens. Consider the initial situation as shown in figure 2.1, where we have a Nine Men’s Morris game in the placing phase. Here we see, that no matter what move white draws, black can get a mill. We construct a Kripke structure, representing the possible moves of the players. Remember, that a move of any player represents a transition between two nodes in the structure and the initial situation in this picture is represented by the root of the structure, node a. Now it is the white player’s turn. Consider the labeled Kripke structure in figure 2.2 over the variable set $\{\varphi\}$. Negated variables are not labeled in this structure.

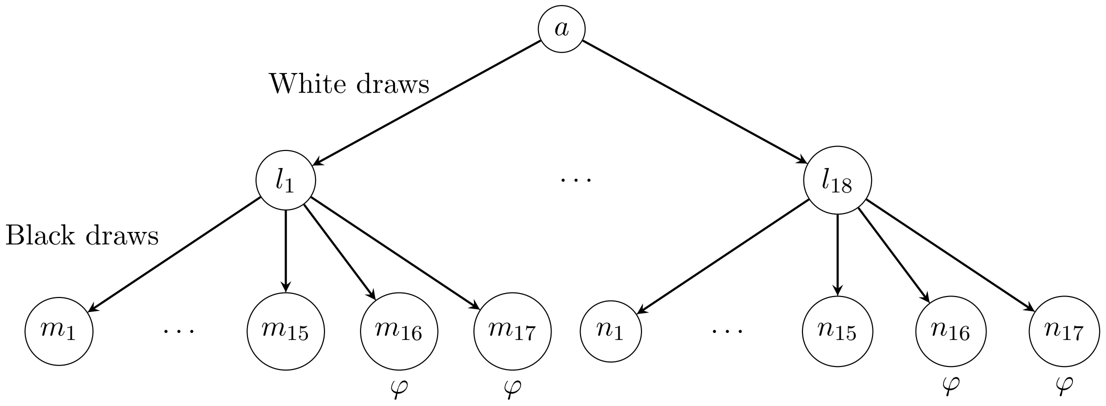


Figure 2.2: The Kripke structure K, representing the possible moves of the "Nine Men's Morris" game situation.

Note that there are many moves for the white player, for simplification we do not explicitely display them, but indicate them by dots.
In this structure the following formulas hold:

$$\begin{array}{lll}
K, \{a\} & \models F(= \Box \Diamond \varphi) & \\
K, \{l_i\} & \models \Diamond \varphi & \text{for } i \in \{1, \dots, 18\} \\
K, \{m_i\} & \models \varphi & \text{for } i \in \{16, 17\} \\
K, \{m_i\} & \models \neg\varphi & \text{for } i \in \{1, \dots, 15\} \\
K, \{n_i\} & \models \varphi & \text{for } i \in \{16, 17\} \\
K, \{n_i\} & \models \neg\varphi & \text{for } i \in \{1, \dots, 15\}
\end{array}$$

Of course, not all strategies appear useful, e.g. n_i for $i \in \{1, \dots, 15\}$ would mean, that although there was an opportunity to get a mill, but the player moving the black tokens did not use it.
This example illustrates the expressive power of modal logic compared with propositional logic, where it is not possible to express future situations.

2.2 The Concept of Dependence

In dependence logic we want to express whether atomic propositions determine each other. For this purpose, we consider sets of different worlds for making statements about dependence. We introduce a new operator, called the *dependence atom* $=(p_1, \dots, p_n; q)$. That means, that the value of the variable q depends only on the vector $p = p_1, \dots, p_n$ in all worlds considered. Now it becomes clear why we consider more than one world. If we did not, we would look statically at a configuration of atomic propositions and it would not make sense to make statements about dependence if we do not have any comparative configurations.

We will give a formal definition of the truth of the dependence atom.

Definition 2.4. (The dependence atom)
Let $K = (S, R, \pi)$ an AP-Kripke structure, $A \subseteq S$. Let $p_1, \ldots, p_n, q \in AP$. Then $=(p_1, \ldots, p_n; q)$ is called a *dependence atom.* The *truth* of the dependence atom on K, A is defined by

$$\begin{aligned} K, A \models {=}(p_1, \ldots, p_n; q) \text{ iff } & \forall m_1, m_2 \in A \text{ with} \\ & \pi(m_1) \cap \{p_1, \ldots, p_n\} = \pi(m_2) \cap \{p_1, \ldots, p_n\} : \\ & q \in \pi(m_1) \Leftrightarrow q \in \pi(m_2) \end{aligned}$$

For example, consider a game of Blackjack, where the goal is to get an entire card value of 21, which is the maximum that may not be exceeded. In a certain game we might notice, that we won a round and we have a total card score of 21, but from this single observation, we can not make a statement of dependence just like

If you get 21 in a Blackjack game, you automatically win.

There is a small chance that the croupier also gets an overall card value of 21, in this situation we would not have won. That makes clear, that it takes more than one event or observation or world to make a non-trivial statement about the game rules using the dependence atoms.
But we could make a statement about several games. The possibility of winning a round in Blackjack depends on more than the simple total score. There is a probability of getting more than 21, which is called "breaking". In this case, we lose automatically. There is also a chance that the croupier gets a higher total card score, in this case we also lose. So we might want to express victory of a round as follows:

$$=(\text{break}, \text{higher score}; \text{win})$$

This formula is interpreted as

"The win of this round depends only on having a higher total card score than the croupier, and on whether our score is higher than 21 or not.".

Example 2.5. (The Dependence operator)
Let P be the total card value of the player, and C be the total card value of the croupier. Moreover p_1 denotes, whether our score is higher than 21, and p_2 denotes, whether we have a higher score than the croupier. q displays whether we have won this round. Consider the following table:

	P	C	P > 21	P > C	victory
			p_1	p_2	q
r_1	18	20	false	false	false
r_2	18	17	false	true	true
r_3	22	19	true	true	false
r_4	19	23	false	false	true

In this example, our formula F would be

$$=(p_1, p_2; q)\,.$$

Now, if our observation consists of the rounds $\{r_1, r_2, r_3\}$, the formula would hold. But the last round conflicts with the formula, because in round 1, we have the same configuration of variables p_1, p_2, but the value of the variable q differs. The reason for this conflict is that we did not considered the possibility, that the croupier can also break. We see that we have to modify our formula again to make it fit to the rules. We have

$$\begin{aligned} \{r_1, r_2, r_3\} &\models\ =(p_1, p_2; q) \\ \{r_1, r_2, r_3, r_4\} &\not\models\ =(p_1, p_2; q) \\ \{r_2, r_3, r_4\} &\models\ =(p_1, p_2; q)\,. \end{aligned}$$

2.3 Modal Dependence Logic

Modal dependence logic can be considered as an extension of modal logic. It combines dependence logic and modal logic. As modal logic it works on Kripke structures, and we can make statements about dependence in Kripke structures.

2.3.1 Syntax

According to the definition of the modal logic syntax, we define the syntax of modal *depencence* logic (MDL).

Definition 2.6. (MDL syntax)
Let AP be a set of atomic propositions and $q_1, \ldots, q_n, p \in AP$. Then the following grammar defines the MDL syntax:

$$\varphi \ ::= p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Box\varphi \mid \Diamond\varphi \mid =(q_1, \ldots, q_n; p)$$

From this, we can make statements, about a dependence over different worlds in Kripke structures. As an example, we could state

$$\Psi := \Diamond \Box =(a; b)\,.$$

We could interpret this statement for example in a game as

"There is a way to move, so that it does not matter how all the other players interact, if player b wins depends only on how player a interacted.".

2.3.2 Semantics

The truth of a formula is very similar to the modal logic truth definition. In an intuitive way, we can see the definition of truth in MDL as an extension of the modal logic truth. The truth of a formula φ is denoted in a set of worlds $A \subseteq S$ of the structure $K = (S, R, \pi)$ by $K, A \models \varphi$. Let $p_1, \dots, p_n, q \in AP$.

$$\begin{array}{lll}
K, A \models q & :\Leftrightarrow & q \in \pi(s), \forall s \in A \\
K, A \models \neg q & :\Leftrightarrow & q \notin \pi(s), \forall s \in A \\
K, A \models \varphi \vee \psi & :\Leftrightarrow & \exists M_1, M_2 \subseteq A, M_1 \cup M_2 = A, \\
 & & K, M_1 \models \varphi \text{ and } K, M_2 \models \psi \\
K, A \models \varphi \wedge \psi & :\Leftrightarrow & K, A \models \varphi \text{ and } K, A \models \psi \\
K, A \models \Box \varphi & :\Leftrightarrow & K, R[A] \models \varphi \\
K, A \models \Diamond \varphi & :\Leftrightarrow & \exists A' \subseteq R[A] \\
 & & \text{so that } A' \models \varphi \\
K, A \models =(p_1, \dots, p_n; q) & :\Leftrightarrow & \forall s, \bar{s} \in A \text{ where} \\
 & & \pi(s) \cap \{p_1, \dots, p_n\} = \pi(\bar{s}) \cap \{p_1, \dots, p_n\} : \\
 & & p_n \in \pi(s) \Leftrightarrow p_n \in \pi(\bar{s})
\end{array}$$

2.3.3 Example

Now we are able to combine statements about moves that can happen, or about events that will happen in the next step with dependence statements. Consider the formula $\Box \Diamond =(p; q)$. We interpret the formula as follows:

"For every move the player makes (whether he draws a card or not), it is possible that the players victory only depends on whether the croupier's total card value is smaller than the one of the player."

As in example 2.5 above, p stands for the proposition that the total card value of the player is higher than the total card value of the croupier, and q stands for the proposition that the player wins this round.
If this formula is true for a certain game situation, the player should be intended to draw a card. Consider the initial situation, that the player has a total score of 11. In this case it is not possible to break (having a card score of more than 21), because, the highest card value is 10 (aces count 11 or 1, which the player decides). Consider figure 2.3, where P is the player's score, and C is the croupier's score. We have labeled p, q and the ratio of P and C. Note that a node b_3 does not exist, because it is not possible to get more than 21 points by drawing one card if the player has 11.
We want to check, whether $\Box \Diamond =(p; q)$ is true or not on the initial set $A := \{a\}$. It follows, that $\Diamond =(p; q)$ has to be true on $R[A] = \{b_1, b_2\}$, and $=(p; q)$ has to be true on some subset of $R[R[A]]$. The set $R[R[A]]$ with its labeling is listed in table 2.1. We have, that

$$S_{max}(\Box \Diamond =(p; q)) = \{\{c_1, c_3, c_4, d_1, d_3\}, \{c_2, c_3, d_1, d_2\}, \{c_1, c_3, c_4, d_1, d_3\}\}$$

on A. Because $S_{max}(\Box \Diamond =(p; q)) \neq \emptyset$, $\Box \Diamond =(p; q)$ is true on the initial set A over this Kripke structure.

state	p	q
c_1	false	false
c_2	false	true
c_3	true	true
c_4	false	false
d_1	true	true
d_2	false	true
d_3	false	false

Table 2.1: List of states of $R[R[\{a\}]]$ of figure 2.3 on which $=(p;q)$ has to be evaluated.

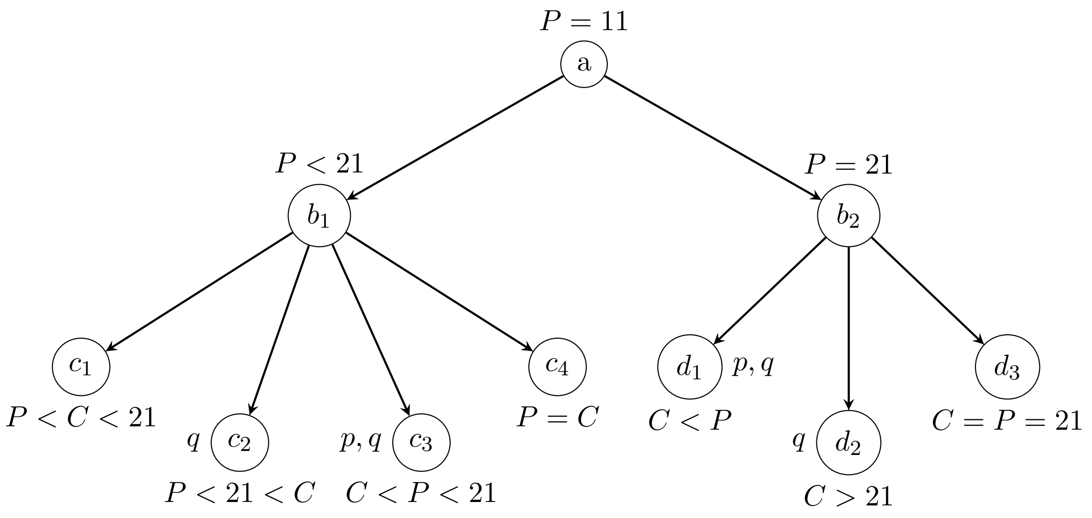


Figure 2.3: The Kripke structure K, representing the possible draws in a Black Jack game situation.

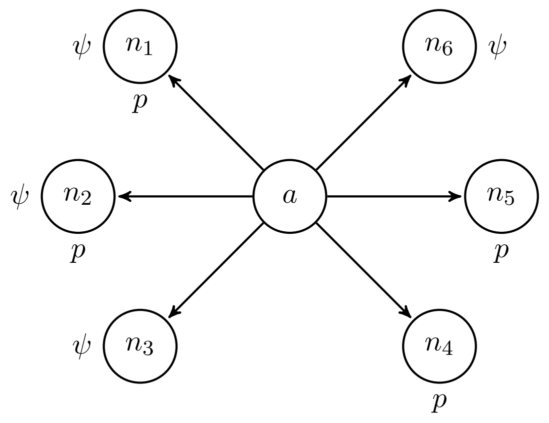


Figure 2.4: Model Checking example, where $A = \{a\}$ and $F = \Diamond(=(p) \vee q)$.

2.4 Model Checking

Model checking in modal dependence logic is a technique for verifying MDL formulas in Kripke structures from a set of initial states. Given a Kripke structure $K = (S, R, \pi)$, representing a finite-state concurrent system and a modal dependence logic formula F, expressing some desired specification. The task is to find the set of all sets of states that satisfy F:

$$M = \{s \in \mathcal{P}(S) | K, s \models F\}.$$

Normally, one set of states of the concurrent system is designated as *initial state set*. The system satisfies the specification provided if the initial set of states is in the set M.

2.4.1 Examples

We will illustrate model checking on a small example. The formula is $F := \Box(=(p) \vee q)$. The meaning of the formula is, that there is a successing set A' of our initial set A and there is a decomposition into two subsets A_1 and A_2 of A', such that p is constant in A_1, and ψ is true in A_2. Formally

$$\exists A_1, A_2 \subseteq A', A_1 \cup A_2 = A' : (K, A_1 \models =(p)) \text{ and } (K, A_2 \models q).$$

Let K be the Kripke structure corresponding to figure 2.4.

First of all, we can list those states, that are labeled with q. Next, we determine those states, which satisfy $=(p)$. Let us define a set $S_{max}(\varphi)$, that contains all maximal sets M with $M \models \varphi$. (We will later define the S_{max} sets formally.) From this, we have

$$\begin{aligned} S_{max}(\psi) &= \{\{n_1, n_2, n_3, n_6\}\} \\ S_{max}(=(p)) &= \{\{n_1, n_2, n_4, n_5\}, \{n_3, n_6\}\} \end{aligned}$$

Now we have to choose one of the sets $\{n_1, n_2, n_4, n_5\}$ or $\{n_3, n_6\}$ and join it with $\{n_1, n_2, n_3, n_6\}$ from $S_{max}(\psi)$ to obtain a successing set A'. That leads us to

$$\{n_1, n_2, n_3, n_6\} \cup \{n_1, n_2, n_4, n_5\} = \{n_1, n_2, n_3, n_4, n_5, n_6\}$$
$$\{n_1, n_2, n_3, n_6\} \cup \{n_3, n_6\} = \{n_1, n_2, n_3, n_6\}.$$

Because $S_{max}(\Box(=(p) \vee \psi)) \ni A'$ with $\{a\} \subseteq A'$ follows, that $K, \{a\} \models \Box(=(p) \vee \psi)$.

3 Complexity Results on MDL-MC

The main result of this section is that model checking on modal dependence logic is NP-complete. We split the proof up into two smaller propositions: the upper bound, that MDL-MC is not more complex than NP and the hardness, by reducing 3-SAT.

Theorem 3.1. Model checking for modal dependence logic is NP-complete.

3.1 The Model Checking Problem on MDL is in NP

It is easy to see, that MDL-MC $\in$ NP by considering the following top-down algorithm, checking the formula F on the Kripke structure S, with the initial set of states M.

Algorithm 3.2. (MDL-MC $\in$ NP)

```
bool check($S$,$F$,$M$)
  if $F = A \vee B$
    guess a fragmentation $\bar{A}$,$\bar{B}$ with $\bar{A} \cup \bar{B} = M$ ;
    return (check $\big(S, A, \bar{A}\big)$ and check $\big(S, B, \bar{B}\big)$ );
  endif

  if $F = A \wedge B$
    return check( $(S, A, M)$ and check $(S, B, M)$ );
  endif

  if $F = \Box A$
    find all successing states $M'$ of $M$ ;
    return check $(S, A, M')$ ;
  endif

  if $F = \Diamond A$
    guess successing set of states $M'$ ;
    return check $(S, A, M')$ ;
  endif
  if $F = {=}(p_1, \ldots, p_n; q)$
    for $1 \leq i \leq |M|$
      Save for vertice $s_i \in M$ the tuple $(p_1, \ldots, p_n, q)$ ;
      for $i \leq j \leq |M|$
        if the values for $p'_1, \ldots, p'_n$ from $s_j = (p'_1, \ldots, p'_n, q')$ fit
        the saved ones, check that  $q' = q$. ;
        if $q' \neq q \Rightarrow$ return false;
      end
    end
    return true;
  endif

  if $F = p$
    for each $s_i \in M$ check that $p \in \pi(s_i)$
      if $p \notin \pi(s_i) \Rightarrow$ return false;
    return true;
  endif

  if $F = \neg p$
    for each $s_i \in M$ check, if $p \notin \pi(s_i)$
      if $p \in \pi(s_i) \Rightarrow$ return false;
    return true;
  endif
end
```

Thereby, we have shown

Proposition 3.3. MDL-MC is in NP.

3.2 The Model Checking Problem for MDL is NP-hard

Now we have to show the NP-hardness of the problem.

Proposition 3.4. MDL-MC is NP-hard.

Proof. We reduce 3-SAT onto MDL-MC. The 3-SAT-problem is the decision, if a 3-CNF formula is in the set

$$\text{3-SAT} := \{\varphi | \varphi \text{ is a 3-CNF formula, that is satisfiable}\}.$$

To show: 3-SAT $\leq_m^p$ MDL-MC.
Let Φ be a 3-CNF formula:

$$\Phi := C_1 \wedge \ldots \wedge C_n, C_i := \bigvee_{k=1}^{3} L_{k,i}.$$

with variables $x_1, \ldots, x_m$. We construct a Kripke structure, a formula we have to check and a set of states for the MDL-MC problem. The following sets are defined:

$$\tilde{x}_i := (\{j | x_i \text{ occurs in } C_j \text{ positive.}\}, \{j | x_i \text{ occurs in } C_j \text{ negative.}\})$$

Let the Kripke structure $W = (S, R, \pi)$ be defined as $S = \{s_1, \ldots, s_n\}$ and

$$\pi(s_i) \supseteq \begin{cases} \{r_j, p_j\}, & \text{iff } x_j \text{ occurs in } C_i \text{ positive.} \\ \{r_j, \neg p_j\} & \text{iff } x_j \text{ occurs in } C_i \text{ negative.} \end{cases} \tag{3.1}$$

Now we define the formula

$$\Psi := \bigvee_{j=1}^{m} \gamma_j \text{ with } \gamma_j := r_j \wedge {=}(p_j)\,.$$

Moreover, let $A := \{s_1, \ldots, s_n\}$ be the set, which we have to check under Ψ into the given Kripke structure.

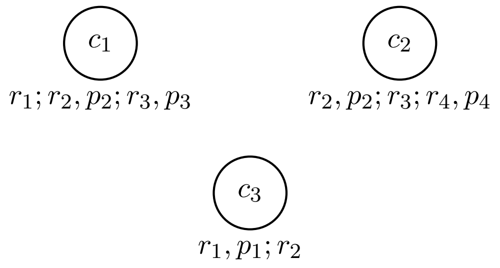
$S_{max}(\phi)$ is the set of valid maximal teams of ϕ. Generally speaking, in MDL it is a set of state-sets. $S_{max}(\phi) := \{M | W, M \models \phi, \text{and } (M \subseteq M' \models \phi \Rightarrow M' = M)\}$

From the definition of the $\vee$ operator follows:

$$A \models \phi \vee \psi \Leftrightarrow \exists B \in S_{max}(\phi), \exists C \in S_{max}(\psi) \text{ with } A \subseteq B \cup C.$$

We also have to show, that for every satisfying configuration of Φ, a choice of state sets $M_i \in S_{max}(\gamma_i)$ exists, so that $\bigcup_{i=1}^{m} M_i \supseteq A$ hold and vice versa.

Example 3.5. Let Φ be the 3-CNF formula, $(\bar{x_1} \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2)$. The associated Kripke structure would be as follows:



Note, that in this special case, we do not need any edges at all. The reason is, that we do not need the operators $\Box$ and $\Diamond$.
A satisfying configuration of Φ would be $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$ and the formula, we have to check is $F = \bigvee_{i=1}^{4} r_i \wedge {=}(p_i)$. The sets $\tilde{x}_1, \ldots, \tilde{x}_4$ are:

$$\begin{aligned} \tilde{x}_1 &= (\{3\}, \{1\}) \\ \tilde{x}_2 &= (\{1,2\}, \{3\}) \\ \tilde{x}_3 &= (\{1\}, \{2\}) \\ \tilde{x}_4 &= (\{2\}, \emptyset) \end{aligned}$$

Along these sets we can see, which clause will be satisfied, by choosing a variable to be 1 or 0. If we assign the configuration above, we set $x_1 = 1$, so we can see in $\tilde{x}_1$, that C_3 will be true. The sets associated with the configuration are $\{3\}, \{3\}, \{1\}, \{2\}$. And we can see, that the union of all of these sets is $\{1, 2, 3\}$, what means, that every clause is satisfied. Thereby we have, that Φ is satisfied.
On the other hand, we can choose some subsets, whose union is $\{1, 2, 3\}$, and show, that this choice has a corresponding configuration, satisfying Φ. For example let us choose from $\tilde{x}_1$ the second set $\{1\}$, as well as for $\tilde{x}_2$ and $\tilde{x}_3$. The choice of the set of $\tilde{x}_4$ does not matter, so we choose $\emptyset$. The union of the set we have chosen is $\{1, 2, 3\}$ and by definition the corresponding configuration to this choice is $x_1 = x_2 = x_3 = x_4 = 0$. Now we can easily verify, that this is a valid configuration.

Claim (Correctness)
$\Phi \in \textsf{3-SAT} \Leftrightarrow W, A \models \Psi$.

Proof. " $\Rightarrow$" Let I be a valid configuration for Φ. Now there are two cases for each variable to choose a set, satisfying γ_i.
Case I: On one hand if $I(x_i) = 1$, it follows, that all clauses containing x_i not negated are true. Let $C_{l_1}, \ldots, C_{l_k}$ be these Clauses, $l_1, \ldots, l_k \in \{1, \ldots, n\}$. From (3.1) we have, that $\pi(s_{l_i}) \supseteq \{r, p\}, i \in \{1, \ldots, k\}$. We choose $p_i = 1$, such that

$$\forall s_j \in M_i : \pi(s_j) \supseteq \{r_i, p_i\}$$

holds.
According to the definition of the π function, these M_i match the choice of the tuple $\tilde{x}_i$. For this reason such a M_i can always be found. Note, that if there is an i for which x_i does not satisfy any clause (no matter if $I(x_i) = 1$ or $I(x_i) = 0$), the formula Φ is equivalent to a formula Φ', where we just leave the variable x_i out.
Case II: On the other hand if $I(x_i) = 0$, we choose analog for every i the sets $M_i \in S_{max}(\gamma_i)$ for which

$$\forall s_j \in M_i : \pi(s_j) \supseteq \{r_i, \neg p_i\}$$

holds. For every s_j, we have found a containing set M_i. Since $I \models \Phi$, every clause C_j in Φ has to be true. From this we have:

$$\bigcup_{i=1}^{m} M_i = A.$$

Hence $\forall i : M_i \in S_{max}(\gamma_i)$ it follows

$$M_i \models \gamma_i \text{ and } W, A \models \Psi, i \in \{1, \ldots, m\}.$$

" $\Leftarrow$" Let $M_i \in S_{max}(\gamma_i)$ mit $\bigcup_{i=1}^{m} M_i \supseteq A$ be given. Because every γ_i are built like $r_i \wedge {=}(p_i)$, for every set $M_i \in S_{max}(\gamma_i)$ holds: $\forall s_j \in M_i : r_i \in \pi(s_j)$. Moreover p_i has to be constant because of the ${=}(p_i)$ operator. From this we have either:

$$\forall s_j \in M_i : r_i, p_i \in \pi(s_j)$$

or

$$\forall s_j \in M_i : r_i, \neg p_i \in \pi(s_j).$$

Now we check, if $\forall s_j \in M_i : \pi(s_j) \supseteq \{r_i, \neg p_i\}$ holds. In this case, let $I(x_i) = 0$. In every other case let $I(x_i) = 1$. Hence, I is a valid configuration, because from definition follows

$$\begin{aligned} \pi(s_j) \supseteq \{r_i, p_j\} &\Leftrightarrow x_i \text{ occurs in clause } C_i \text{ positive.} && \Rightarrow C_i \text{ comes true.} \\ \pi(s_j) \supseteq \{r_i, \neg p_j\} &\Leftrightarrow x_i \text{ occurs in clause } C_i \text{ negative.} && \Rightarrow C_i \text{ comes true.} \end{aligned}$$

Altogether the configuration I is chosen for every variable x_i in a way, that C_i comes true. It follows

$$\forall_i : I \text{ satisfies } C_i \Rightarrow I \text{ satisfies } \Phi.$$

□

4 Complexity Results on Operator Fragments for MDL-MC

We have seen, that MDL-MC is NP-complete. The proof is based mainly on the property of guessing sets to satisfy the disjunction. In the following section, we will examine some cases, where only fragments of the operator set $\{\neg, \vee, \wedge, \Box, \Diamond, =(\cdot)\}$ are allowed. Especially the case, where no disjunction is allowed is interesting, because the NP-hardness proof will not work in the above way without disjunction.

4.1 NP-hard Fragments

To introduce the *operator fragments*, we will define it in an intuitional way.

Definition 4.1. (Operator fragments)
Let $M \subseteq \{\neg, \vee, \wedge, \Box, \Diamond, =(\cdot)\}$, then MDL(M) is the set of formulas over AP built from the operators in M. The MDL-MC(M)-problem is the problem of model checking over formulas of MDL(M).

Let $M := \{\neg, \wedge, \Box, \Diamond, =(\cdot)\}$, then MDL-MC is NP-hard, as we see in the next Theorem. This is the interesting case, because here, we do not need the disjunction.

Theorem 4.2. MDL-MC($\{\Diamond, =(\cdot)\}$) is NP-complete.

Proof. We have already shown, that MDL-MC is in NP. Because MDL-MC(M) is a subproblem of MDL-MC, it is also in NP.
To show: MDL-MC is NP-hard.
Again we reduce 3-SAT $\leq_m^p$ MDL-MC(M). Let Φ be a given 3-CNF-formula.

$$\Phi := C_1 \wedge \ldots \wedge C_n \text{ with } C_i := L_{i_1} \vee L_{i_2} \vee L_{i_3} \text{ for } i = 1, \ldots, n.$$

$$\forall i : L_i \in \{x_1, \ldots, x_m\}.$$

The structure $K = (S, L, \pi)$ is defined as follows:

$$S := \{c_1, \ldots, c_n, s_1, \ldots, s_m, \bar{s}_1, \ldots, \bar{s}_m\}, \text{ thus } card(S) = 2m + n.$$

$$S \times S \supseteq L \supseteq \begin{cases} \{(c_i, s_j)\}, & \text{if in clause } C_i \text{ the literal } x_j \text{ occurs.} \\ \{(c_i, \bar{s}_j)\}, & \text{if in clause } C_i \text{ the literal } \bar{x}_j \text{ occurs.} \end{cases}$$

$$\begin{tikzcd}
& c_i \arrow[dr, bend left] & \\
\bar{s}_j & & s_j
\end{tikzcd}$$

$$\Phi_1 = (L_1 \vee L_2 \vee L_3) \wedge \ldots \wedge \underbrace{(\ldots \vee x_j \vee \ldots)}_{C_i} \wedge \ldots \wedge (L_k \vee L_l \vee L_m)$$

$$\begin{tikzcd}
& c_i \arrow[dl, bend right] & \\
\bar{s}_j & & s_j
\end{tikzcd}$$

$$\Phi_2 = (L_1 \vee L_2 \vee L_3) \wedge \ldots \wedge \underbrace{(\ldots \vee \bar{x}_j \vee \ldots)}_{C_i} \wedge \ldots \wedge (L_k \vee L_l \vee L_m)$$

π is defined by

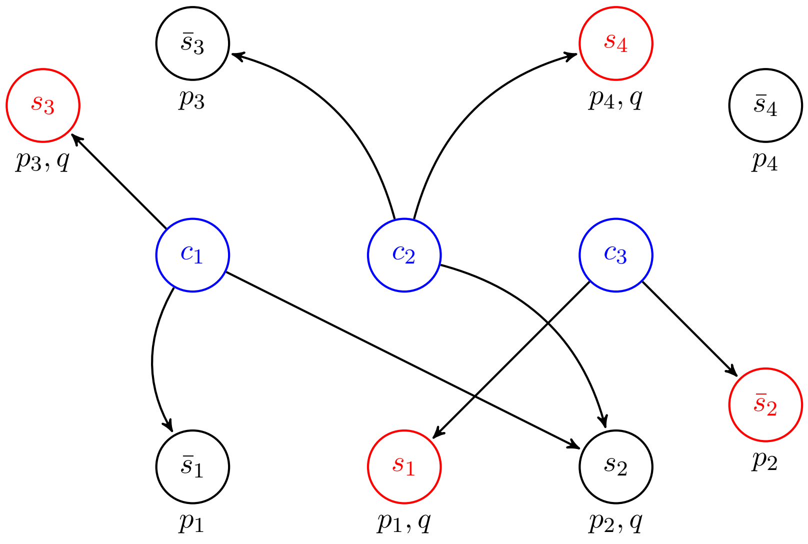
$$\pi(s_i) \supseteq \{p_i, q\}$$
$$\pi(\bar{s}_i) \supseteq \{p_i\}$$

The formula F we have to check is

$$\Psi := \Diamond{=}(p_1, \ldots, p_m; q)\,.$$

Moreover let $A := \{c_1, \ldots, c_n\}$ bet the set of initial states which we want to check on the formula F.

Example 4.3. Let Φ be the 3-CNF formula, given by $\Phi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2)$. The MDL(M)-formula we have to check unfolds to $F = \Diamond{=}(p_1, p_2, p_3, p_4; q)$. The labeled Kripke structure would be:



A satisfying configuration for Φ could be e.g.: $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$. Now, there has to be a successor set of states B for the set of blue colored states $A = \{c_1, c_2, c_3\} \models F$ and $B \models {=}(p_1, \ldots, p_4; q)$. In this case this set would be the set of red colored nodes $B = \{s_1, \bar{s}_2, s_3, s_4\}$.

Claim (Correctness)
$\Phi \in$ 3-SAT $\Leftrightarrow K, A \models \Psi$.

Proof. " $\Rightarrow$": Let Γ be a configuration, that satisfies $\Gamma \models \Phi$. It follows $\forall i : \Gamma \models C_i$. Choose as a successing set B of A as follows:

$$B \supseteq \begin{cases} \{x_k\}, & \text{iff. } \exists l : (c_l, x_k) \in M \text{ and } \Gamma(x_k) = 1 \\ \{\bar{x}_k\}, & \text{iff. } \exists l : (c_l, \bar{x}_k) \in M \text{ and } \Gamma(x_k) = 0 \end{cases}$$

From this we have, that every element in B has a predecessor c_l for applicable l.
Now we have to show, that every state c_i has a successor in B. Hence, every clause has a literal L_k with $\Gamma(L_k) = 1$, every state c_i has to have a successor in B. Of course $B \models {=}(p_1, \ldots, p_m; q)$ has to hold. Hence, $\forall k : \Gamma(s_k) = 0$ oder $\Gamma(s_k) = 1$ holds, B contains either x_k or $\bar{x}_k$. From the definition of π it is necessary, that

$$\pi(s_k) \cap \{p_1, \ldots, p_m\} = \pi(\bar{s}_l) \cap \{p_1, \ldots, p_m\} \Leftrightarrow k = l.$$

That means, that the configuration of the p_i for two different elements of B can not be the same. It follows

$$B \models {=}(p_1, \ldots, p_m; q) \Rightarrow K, A \models F.$$

" $\Leftarrow$" Assuming $K, A \models F$. Let B be a set of successing states of A, for wich is true $B \models {=}(p_1, \ldots, p_m; q)$. According to the above argument B never contains s_k *and* $\bar{s}_k$. Define Γ as follows:

$$\Gamma(x_k) = \begin{cases} 1, & \text{iff. } x_k \in B \\ 0, & \text{iff. } x_k \notin B. \end{cases}$$

Let c_i be the predecessor node of s_k. It follows, that $\Gamma(x_k) = 1 \Rightarrow \Gamma \models C_i$. Let c_l be the predecessor node of $\bar{s}_t$. In this case follows $\Gamma(x_t) = 0 \Rightarrow \Gamma \models C_i$.
Hence $K, A \models F$ altogether it follows

$$\forall i : \Gamma \models C_i \Rightarrow \Gamma \models \Phi.$$

$\square$

Theorem 4.4. Let M be $\{\vee, \Box, =\!(\cdot)\}$. Then $\mathsf{MDL\text{-}MC}(\mathrm{M})$ is NP-complete.

Proof. Again it is clear, that $\mathsf{MDL\text{-}MC}(M) \in \mathsf{NP}$, using the subset argument. To prove the hardness, we will reduce $3SAT$ onto this problem, analogously to the theorems above.
To show: $3SAT \leq_m^p \mathsf{MDL\text{-}MC}(M)$.
Let Φ be a given $3CNF$-formula, defined similar as above with

$$\Phi := C_1 \wedge \ldots \wedge C_n \text{ with } C_i := L_{i_1} \wedge L_{i_2} \wedge L_{i_3}, \text{ for } i = 1, \ldots, n.$$

$$\forall i, j : L_{i_j} \in \{x_1, \ldots, x_m, \bar{x}_1, \ldots, \bar{x}_m\}.$$

We define the structure $K = (S, R, \pi)$ as follows:

$$\begin{array}{rll} S := \{ & s_1, \ldots, & s_n, \\ & r_1^1, \ldots, & r_1^m, \\ & \vdots & \vdots \\ & r_n^1, \ldots, & r_n^m, \\ & \bar{r}_1^1, \ldots, & \bar{r}_1^m, \\ & \vdots & \vdots \\ & \bar{r}_n^1, \ldots, & \bar{r}_n^m\}. \end{array}$$

$$R \supseteq \begin{cases} \{(s_i, r_i^1)\} \text{ if } x_1 \text{ or } \bar{x}_1 \text{ occur } C_i. & \textbf{Case I} \\ \{(s_i, r_i^1), (s_i, \bar{r}_i^1)\} \text{ if } x_1 \text{ and } \bar{x}_1 \text{ does not occur in } C_i. & \textbf{Case II} \\ \{(r_i^j, r_i^{j+1})\} \text{ if } (x_{j+1} \text{ or } \bar{x}_{j+1}) \text{ and } (x_j \text{ or } \bar{x}_j) \text{ occur in } C_i. & \textbf{Case III} \\ \{(r_i^j, r_i^{j+1}), (r_i^j, \bar{r}_i^{j+1})\} \text{ if } x_{j+1} \text{ and } \bar{x}_{j+1} \text{ do not occur} & \\ \quad \text{in } C_i \text{ but } x_j \text{ or } \bar{x}_j \text{ do occur in } C_i. & \textbf{Case IV} \\ \{(r_i^j, r_i^{j+1}), (\bar{r}_i^j, r_i^{j+1})\} \text{ if } x_j \text{ and } \bar{x}_j \text{ does not occur} & \\ \quad \text{in } C_i\text{, but } x_{j+1} \text{ or } x_{j+1} \text{ does occur in } C_i. & \textbf{Case V} \\ \{(r_i^j, r_i^{j+1}), (\bar{r}_i^j, \bar{r}_i^{j+1})\} \text{ if both} x_j \text{ and } \bar{x}_j & \\ \quad \textit{and } x_{j+1} \text{ and } x_{j+1} \text{ do not occur in } C_i. & \textbf{Case VI} \end{cases}$$

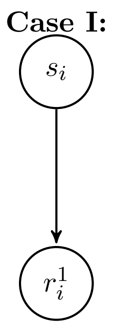


Figure 4.1: x_1 occurs in C_i.

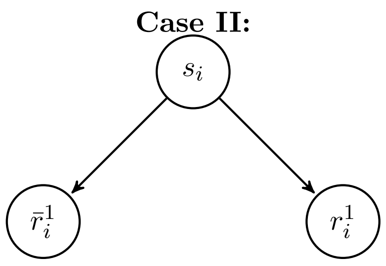


Figure 4.2: x_1 does not occur in C_i.

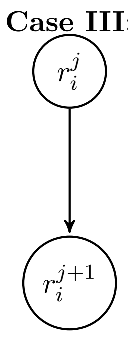


Figure 4.3: x_j and x_{j+1} occur in C_i.

Case IV:

$$\begin{tikzcd}
 & r_i^j \arrow[dl] \arrow[dr] & \\
\bar{r}_i^{j+1} & & r_i^{j+1}
\end{tikzcd}$$

Figure 4.4: x_j occurs in C_i, but x_{j+1} does not occur in C_i.

Case V:

$$\begin{tikzcd}
\bar{r}_i^j \arrow[dr] & & r_i^j \arrow[dl] \\
 & r_i^{j+1} &
\end{tikzcd}$$

Figure 4.5: x_j does not occur in C_i, but x_{j+1} does occur in C_i.

Case VI:

$$\begin{tikzcd}
\bar{r}_i^j \arrow[d] & r_i^j \arrow[d] \\
\bar{r}_i^{j+1} & r_i^{j+1}
\end{tikzcd}$$

Figure 4.6: x_j and x_{j+1} do not occur in C_i.

The labeling function π denotes, if a variable occurs negated or not negated in some clause or even if it does not occur anyway in a specific clause. We define π accordingly to the negation of the variables in the clauses. In the cases **II, IV, V, VI**, where we have more than one state in one variable level at one clause, we will label one of the states with p_i, and the other one with $\bar{p}_i$. This guarantees, that if a variable x_j does not occur in a clause C_i, there will be no configuration found, satisfying C_i, because $=\!(p_i)$ for both of these states will not hold. The formal defininition of π is given by

$$\pi(s_i) := \emptyset \tag{4.1}$$

$$\pi(r_i^j) := \begin{cases} \emptyset & \text{iff } x_j \text{ occurs negated in } C_i. \\ \{p_j\} & \text{otherwise.} \end{cases} \tag{4.2}$$

$$\pi(\bar{r}_i^j) := \emptyset. \tag{4.3}$$

The formula we have to check is

$$\Psi := \bigvee_{i=1}^{m} \gamma_i \text{ with } \gamma_i := \Box^i =\!(p_i)\,.$$

with the initial state set $A := \{s_1, \ldots, s_n\}$.

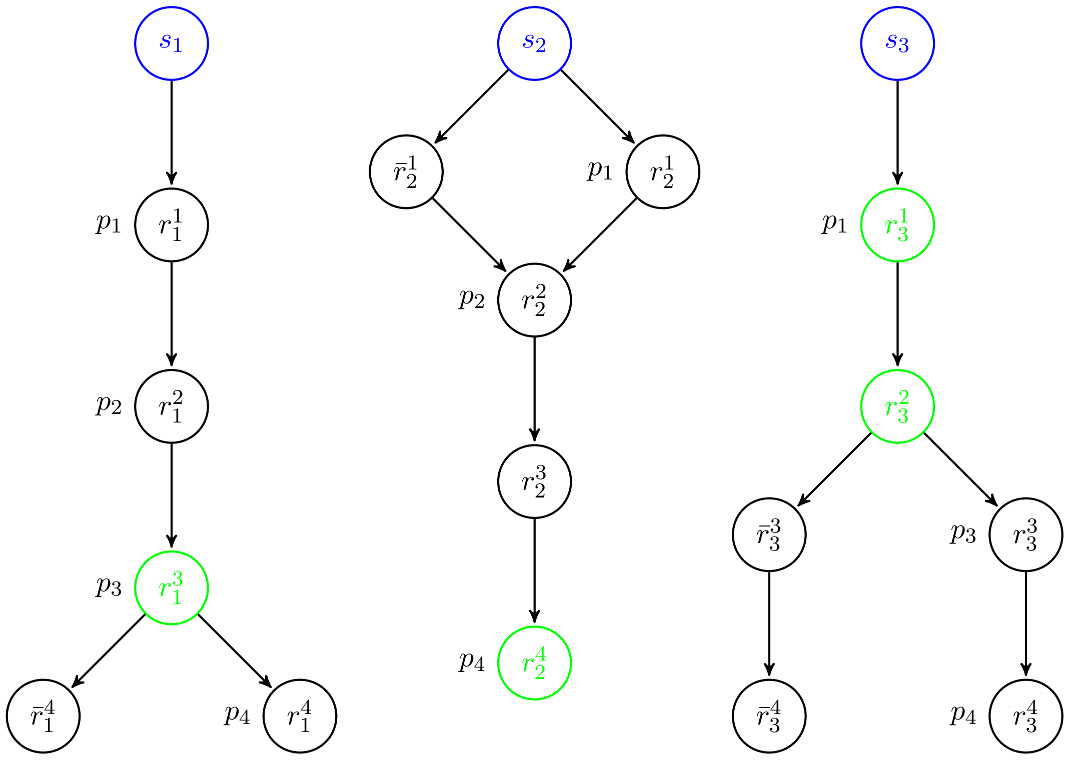
Consider the next example, where the structure is labeled. The blue colored states correspond to the clauses in the 3-CNF formula. Some of the states $\{r_1^1, r_1^2, r_1^3, r_2^2, r_2^3, r_3^1, r_3^2\}$ do not matter, so we left them out.

Example 4.5. Let Φ be a 3-CNF-formula:

$$\Phi := (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2)$$

The corresponding structure would have four "levels". Propositions, which are false in any state are not labeled. In this case Ψ would be

$$\Psi = \underbrace{\Box =\!(p_1)}_{\gamma_1} \vee \underbrace{\Box\Box =\!(p_2)}_{\gamma_2} \vee \underbrace{\Box\Box\Box =\!(p_3)}_{\gamma_3} \vee \underbrace{\Box\Box\Box\Box =\!(p_4)}_{\gamma_4}.$$



A satisfying configuration for example could be $x_1 = x_3 = x_4 = 1$ and $x_2 = 0$. Surely $S_{max}(\Psi) = \bigcup_{i=1}^{m} S_{max}(\gamma_i)$. We have to find nodes n_i in every tree T_i, so that $n_i \in S_{max}(\Psi)$. $S_{max}(\gamma_i) := S^i_{max}$. The S_{max} sets for this example are

$$
\begin{aligned}
S^1_{max} &= \{\underbrace{\{s_1\}}_{M_1}, \{s_3\}\} \\
S^2_{max} &= \{\underbrace{\{s_1, s_2\}}_{M_2}, \{s_3\}\} \\
S^3_{max} &= \{\{s_1\}, \underbrace{\{s_2\}}_{M_3}\} \\
S^4_{max} &= \{\underbrace{\{s_2\}}_{M_4}, \emptyset\}.
\end{aligned}
$$

We see, that $M_1, \ldots, M_4 \subseteq A$ and $M_1 \cup M_2 \cup M_3 \cup M_4 = A$ as required from the definition of disjunction to satisfy Ψ.
The green colored states indicate, which variable satisfies the associated clause. To satisfy Φ, we must have a green colored state in every tree. The $\Box$ operator allows only to color every state in a specific level of a clause green, or no one.

Claim (Correctness)
$\Phi \in$ 3-SAT $\Leftrightarrow K, A \models \Psi$.

Proof. "$\Rightarrow$": Let Γ be a configuration with $\Gamma \models \Phi$. For every $\Gamma(x_j) = 1$ set for the corresponding $=\!(p_j)$ term $p_j = 1$ in Ψ. Analog set for every $\Gamma(x_j) = 0$ in the $=\!(p_j)$ term $p_j = 0$. Because $\Gamma \models \Phi \Rightarrow \Gamma \models C_i$, for every $i \in \{1, \dots, n\}$ there exists variables x_j satisfying C_i. We will show now, that for every tree T_k in our structure, there is at least one node s_k with $s_k \in S_{max}(\Psi)$.
As we found a union of sets for the disjunction in the NP-hardness proof of proposition 3.4, again we have to find a union of sets, satisfying the disjunction:

$$K, A \models \Psi = \bigvee_{i=1}^{m} \gamma_i \Rightarrow \exists M_1, \dots, M_m \subseteq A : \bigcup_{i=1}^{m} M_i = A \text{ and } K, M_i \models \gamma_i.$$

From the definition of $\gamma = \Box^i =\!(p_i)$, we have the possibility of choosing states in a way that p_i is true in all states considered, or choosing states in a way that $\neg p_i$ is true in all states considered. For this purpose, we define an abbreviation $\gamma_j^0 := \Box^j \neg p_j$ and $\gamma_j^1 := \Box^j p_j$.
We choose these sets as follows:

$$\forall j \in \{1, \dots, m\} : M_j \supseteq \begin{cases} \{s | \{s\} \models \gamma_j^0\} \in S_{max}(\gamma_j) \text{ iff } x_j \text{ occurs in } C_i \text{ and } \Gamma(x_j) = 0 \\ \{s | \{s\} \models \gamma_j^1\} \in S_{max}(\gamma_j) \text{ iff } x_j \text{ occurs in } C_i \text{ and } \Gamma(x_j) = 1 \end{cases}$$

Clearly, $M_1, \dots, M_m \subseteq A$. We have to show, that $M_1 \cup \ldots \cup M_m = A$. For this purpose, we show

$$\forall i \in \{1, \dots, n\} : s_i \in A \Rightarrow \exists j \in \{1, \dots, m\} : s_i \in M_j.$$

Let $i \in \{1, \dots, n\}$ be arbitrary, $s_i \in A = \{s_1, \dots, s_n\}$ a state and L_j a literal, satisfying the corresponding C_i, $\Gamma(L_j) = 1$.
W.l.o.g. let $L_j = x_j$. in this case, we choose p_j to be true. From the definition of π (4.2) follows, that in the i-th level of the structure, there is a node labeled p_j.
The other case, where $L_j = \bar{x}_j$, we choose p_j to be false, and in the i-th level of the structure is a node, labeled with $\neg p_j$. From this follows, that $s_i \in M_j$.

"$\Leftarrow$": Let $M_1, \dots, M_m \subseteq A$ be a decomposition of A with $M_1 \cup \ldots \cup M_m = A$ and $K, M_j \models \gamma_i$. Let $s_i \in M_j$ be arbitrary. If $\pi(r_i^j) = \bar{p}_j$ set $x_j = 0$, else if $\pi(r_i^j) = p_j$ set $x_j = 1$. Let $\pi(r_i^j) = \bar{p}_j$. From (4.2) we know, that in this case, x_j occurs in negated form in C_i. Then $x_j = 0 \models C_i \Rightarrow \Gamma \models C_i$. On the other hand, let $\pi(r_i^j) = p_j$, then we know from (4.2), that x_j occurs in C_i not negated. Then $x_j = 1 \models C_i \Rightarrow \Gamma \models C_i$. Because $M_1 \cup \ldots \cup M_m = A = \{s_1, \dots, s_n\}$ follows, that

$$\begin{aligned} \forall i \in \{1, \dots, n\} :& \Gamma \models C_i \\ \Leftrightarrow& \Gamma \models \Phi \\ \Leftrightarrow& \Phi \in \mathsf{3\text{-}SAT}. \end{aligned}$$

□

4.2 Fragments in P

In the section above we discussed the NP-hard operator fragments of $\{\neg, \vee, \wedge, \Box, \Diamond, =(\cdot)\}$. The obvious question is, if there are fragments, that are efficiently solvable. This section is about such efficiently solvable operator fragments. We will give polynomial model checking algorithms for these operators.

Theorem 4.6. Let $M \subseteq \{\neg, \wedge, \Box, =(\cdot)\}$. Then MDL-MC(M) is solvable in polynomial time.

Proof. As we have seen in the proof of proposition 3.3, there is an NP-top down algorithm for the MDL-MCproblem. To verify the MDL-MC($\{\neg, \wedge, \Box, =(\cdot)\}$) problem, we will consider algorithm 3.2 and leave out all nondeterministic steps to obtain a P-algorithm for the subset. We will see, that the following algorithm fits the fragment $\{\neg, \wedge, \Box, =(\cdot)\}$.

Algorithm 4.7. (MDL-MC($\{\neg, \wedge, \Box, =(\cdot)\}$) $\in$ P)

```
bool check($S$,$F$,$M$)
  if $F = A \wedge B$
    return check( $(S, A, M)$ and check $(S, B, M)$ );
  endif

  if $F = \Box A$
    for all $s' \in S$
      if $(s, s') \in R$
        $M'$ <- $M' \cup \{s'\}$
      end;
    end;
    return check $(S, A, M')$ ;
  endif

  if $F = =(p_1, \ldots, p_n; q)$
    for $1 \leq i \leq |M|$
      Save for vertice $s_i \in M$ the tuple $(p_1, \ldots, p_n, q)$ ;
      for $i \leq j \leq |M|$
        if the values for $p'_1, \ldots, p'_n$ from $s_j = (p'_1, \ldots, p'_n, q')$ fit
        the saved ones, check that  $q' = q$. ;
        if $q' \neq q \Rightarrow$ return false;
      end
    end
    return true;
  endif
  if $F = p$
    for each $s_i \in M$ check that $p \in \pi(s_i)$
      if $p \notin \pi(s_i) \Rightarrow$ return false;
    return true;
  endif

  if $F = \neg p$
    for each $s_i \in M$ check, if $p \notin \pi(s_i)$
      if $p \in \pi(s_i) \Rightarrow$ return false;
    return true;
  endif
end
```

Model checking on teams works with this top down algorithm, because we have a given set M, where we determine whether the formula F holds or not. In the algorithm above we change this set and again determine whether F is true on al successing set of M. Model checking with a bottom up algorithm as it is given in [CGP99] does not work on teams. For a dependence atom the algorithm had to find all sets on which it is true, but there can be up to $2^{|S|}$ many teams the algorithm had to check.

$\square$

In the next Theorem 4.8, we leave the dependence operator out. For this reason, it is useful to consider only singleton sets as initial states. For succinctness, we will leave the brackets { and } out. On this note, $K, a \models F$ means $K, \{a\} \models F$.
Remember, that in case of singleton sets, there is no difference between the classical disjunction and the disjunction we defined by $\vee$.

Theorem 4.8. Let $M \subseteq \{\neg, \vee, \wedge, \Box, \Diamond\}$ then MDL-MC(M) is in P.

Proof. We will first show, that MDL-MC(M) $\in$ P for singleton sets, and then we generalize the theorem to arbitrary initial sets. The algorithms differ in that they are bottom up and we do not have sets of states as argument, but single states.

Algorithm 4.9. ($\psi = \neg\psi_1$ on singletons)

```
void check_psi_Eq_Not_psi1(K,$\psi_1$,a)
  if not $label(a) \ni \psi_1$
    $label(a)$ <- $label(a) \cup \psi$
  endif;
end;
```

Algorithm 4.10. ($\psi = \psi_1 \wedge \psi_2$ on singletons)

```
void check_psi_Eq_psi1_and_psi2(K,$\psi_1, \psi_2$,a)
   if $label(a) \ni \psi_1$ and $label(a) \ni \psi_2$
     $label(a)$ <- $label(a) \cup \psi$ ;
   endif;
end;
```

Remember, from the definition of $\vee$ follows, that

$$K, M \models \Psi_1 \vee \Psi_2 \text{ iff } \exists_{M_1,M_2 \subseteq M, M_1 \cup M_2 = M} : K, M_1 \models \Psi_1 \text{ and } K, M_2 \models \Psi_2.$$

Considering singleton sets $\{a\}$ leads us to either $M_1 = \{a\}$ so that $K, a \models \Psi_1$ or $M_2 = \{a\}$ so that $K, a \models \Psi_2$. Here we see, that this is similar to

$$a \models \Psi_1 \vee \Psi_2 \text{ iff } a \models \Psi_1 \text{ or } a \models \Psi_2.$$

From this point it is possible, to give an P algorithm, analog to algorithm 4.10.

Algorithm 4.11. ($\Psi = \Psi_1 \vee \Psi_2$ on singletons)

```
void check_psi_Eq_psi1_or_psi2(K,$\psi_1, \psi_2$,a)
   if $label(a) \ni \psi_1$ or $label(a) \ni \psi_2$
     $label(a)$ <- $label(a) \cup \psi$ ;
   endif;
end;
```

The following algorithm processes every possible successor in S, and if there is any successor $s \in S$, where $\psi_1 \in \pi(s)$ is not true, we notice, that $K, a \not\models \psi$ by setting the variable successors to false.

Algorithm 4.12. ($\psi = \Box\, \psi_1$)

```
void check_psi_Eq_Box_psi1(K,$\psi_1$,a)
   boolean successors = true;
   forall $s \in S$ do
     if $(a, s) \in R$ and $\psi_1 \notin label(s)$
       successors = false;
     endif;
   end;
   if successors == true
     $label(s)$ <- $label(s) \cup \psi$ ;
   endif;
end;
```

Now we define an algorithm, that never occured in the proofs of the fragments above, because every fragment containing the $\Diamond$ operator, has been NP-complete. Only on singleton intial state sets, the complexity falls to at least P. Since we only examine singleton sets, the P-algorithm resembles the algorithm 4.12.

Algorithm 4.13. ($\psi = \Diamond \psi_1$)

```
void check_psi_Eq_Diamond_psi1(K,$\psi_1$,a)
  boolean successors = false;
  forall $s \in S$ do
    if $(a, s) \in R$ and $\psi_1 \in label(s)$
      successors = true;
    endif;
  end;
  if successors == true
    $label(s)$ <- $label(s) \cup \psi$ ;
  endif;
end;
```

Now we want to generalize these algorithms, so that they work over countable sets. For this purpose, we will create a preprocessing algorithm, which processes every node of M one by one.

Algorithm 4.14. (Generalizing algorithm, over state set M)

```
void check_F $(K, F, M)$
  if  $F = \neg\psi$
    for every $m \in M$ do
      check_psi_Eq_Not_psi1 $(K, \psi, m)$
    end;
  endif;

  if  $F = \psi_1 \wedge \psi_2$
    for every $m \in M$ do
      check_psi_Eq_psi1_and_psi2 $(K, \psi_1, \psi_2, m)$
    end;
  endif;

  if  $F = \psi_1 \vee \psi_2$
    for every $m \in M$ do
      check_psi_Eq_psi1_or_psi2 $(K, \psi_1, \psi_2, m)$
    end;
  endif;
  if  $F = \Box\psi_1$
    for every $m \in M$ do
      check_psi_Eq_Box_psi1 $(K, \psi_1, m)$
    end;
  endif;
  if  $F = \Diamond\psi_1$      for every $m \in M$ do
      check_psi_Eq_Diamond_psi1 $(K, \psi_1, m)$
    end;
  endif;
end;
```

This algorithm decides which of the above algorithms to take and also generalizes the buttom up singleton algorithms above, by processing a set of states M one by one.

$\square$

We will show now, that MDL-MC($\{\neg, \vee, =\!(\cdot)\}$) is in P, when we make a constraint on the dependence atom, that all dependencies in the formula F are of the form $=\!(p_1, \ldots, p_j; p)$ with $p \in P, j \leq k$ and fixed $k \in \mathbb{N}$. To show this theorem, we will decompose it into two smaller propositions.

Theorem 4.15. Let us assume, that every $=\!(\cdot)$ atom is of the form $=\!(p_1, \ldots, p_j; q), k \in \mathbb{N}, j \leq k$ fixed with $q \in AP$ and $M := \{\neg, \vee, =\!(\cdot)\}$. Then MDL-MC(M) is in P.

We show, that even the whole $\{\neg, \vee, =\!(\cdot)\}$ fragment with unrestricted $=\!(\cdot)$ atoms is in P as long as a specific number of dependence atoms depending on the size of the Kripke structure is not exceeded.

Lemma 4.16. Let $F := =\!(p_1, \ldots, p_n; q)$ and A the set of initial states. Then follows

$$\exists_{M_1, M_2 \in S_{max}(F)} : M_1 \cup M_2 = A.$$

Proof. Choose a state $s_1 \in A$. If $A \models =\!(p_1, \ldots, p_n; q)$, there is nothing more to show. Otherwise there is a set $M_1 \in S_{max}(F)$ with $s \in M_1$ (at least $M_1 = \{s_1\}$). Define $M_2 := A \backslash M_1$. Clearly $A = M_1 \cup M_2$, so we have to show, that $\exists_{M_2' \supseteq M_2} : M_2' \in S_{max}(F)$. For all states $s_2 \in M_2$ with $\pi(s_2) \cap \{p_1, \ldots, p_n\} \neq \pi(s) \cap \{p_1, \ldots, p_n\}$ follows $\{s_2\} \models F$. For all other states $s_2' \in M_2$ with $\pi(s_2) \cap \{p_1, \ldots, p_n\} = \pi(s) \cap \{p_1, \ldots, p_n\}$ follows, that $\pi(s_2') \cap \{q\} \neq \pi(s_1) \cap \{q\}$. It follows

$$\forall_{s_1, s_2 \in M_2} : \pi(s_1) \cap \{p_1, \ldots, p_n\} = \pi(s_2) \cap \{p_1, \ldots, p_n\} \Rightarrow \pi(s_1) \cap \{q\} = \pi(s_2) \cap \{q\}.$$

Altogether follows $M_2 \models F$. $\square$

Lemma 4.17. Let $\gamma_1 := =\!(p_1, \ldots, p_n; q^i)$, $\gamma_2 := =\!(p_1', \ldots, p_m'; q^k)$ and $\Gamma := \gamma_1 \vee \gamma_2$. Then it follows, that $\forall_{M \in S_{max}(\gamma_j), N \in S_{max}(\Gamma)} : |M| \leq |N|$ for $j \in \{1, 2\}$.

Proof. From the definition we have, that $A \models \Gamma$ iff $\exists_{M_1, M_1 \in A, M_1 \cup M_2 = A} : M_1 \models \gamma_1$ and $M_2 \models \gamma_2$. Moreover we have, that $S_{max}(\gamma_j) := \{M \subseteq A | M \models \gamma_j \text{ and } (M \subseteq M' \models \gamma_j \Rightarrow M = M')\}, j \in \{1, 2\}$. So for every $S_{max}(\Gamma) \ni M$ there exists two sets $M_j \in S_{max}(\gamma_j), j \in \{1, 2\}$. It follows, that $M_1 \models \gamma_1$, $M_2 \models \gamma_2$ and $M_1 \cup M_2 = M$. Altogether we have, that $|M_1| \leq |M|$ and $|M_2| \leq |M|$. $\square$

Proposition 4.18. Let $M := \{\neg, \vee, =(\cdot)\}$. Let m be the number of dependence atoms in the formula F, and $n := |S|$ of the Kripke structure $K := (S, A, \pi)$. If $n < 2^m$, $\mathsf{MDL\text{-}MC}(\mathrm{M})$ is in P.

Proof. To prove the statement, we have to find a decomposition for all the $\vee$ operators in the formula. We show now, that every dependence atom covers at least half of the states in our initial set, that have not been covered yet.
Initially, we will process the atomic propositions in the formula. Let $p_1, \ldots, p_n \in AP$ be all atomic propositions of the formula F. Clearly, it is possible to write every $\mathsf{MDL}(\{\neg, \vee, =(\cdot)\})$ formula in the form

$$F = \underbrace{p_1 \vee \ldots \vee p_k \vee \neg p_{k+1} \vee \ldots \vee \neg p_n}_{F''} \vee F',$$

where F' is the part of the formula containing the dependence operators and $1 \leq k \leq n$ is the number of positive atomic propositions. Hence F'' is an $\mathsf{MDL}(\{\neg, \vee, \wedge, \Box, \Diamond\})$ formula, and from theorem 4.8 we have that $\mathsf{MDL\text{-}MC}(\{\neg, \vee, \wedge, \Box, \Diamond\})$, F is processable in P.
We have to examine F' on the Kripke structure. We consider all dependence atoms $=(p_1^i, \ldots, p_{n_i}^i; q^i), i \in \{1, \ldots, m\}$ of F. From lemma 4.16 we have, that fo every $\gamma_i := =(p_1^i, \ldots, p_{n_i}^i; q^i), i \in \{1, \ldots, m\}$ atom, there exists at least two sets M_1 and M_2 in $S_{max}(\gamma_i)$ with $M_1 \cup M_2 = A$. We want to choose the set, containing the more uncovered states of A. We can do this in P, by counting the number of q^i.
For the disjunction of two dependence atoms $\Gamma := \gamma_1 \vee \gamma_2$ with $\gamma_1 := =(p_1^{(1)}, \ldots, p_{n_i}^{(1)}; q^{(1)})$ and $\gamma_2 := =(p_1^{(2)}, \ldots, p_{n_i}^{(2)}; q^{(2)})$ we have from lemma 4.17, that the sets M_1, M_2, we choose for γ_1, γ_2 are contained in $S_{max}(\Gamma)$.
Note, that we can only cover the whole set A by convering at least half of the remainder set, because if there are only two sets left to cover, we have at least one set in $S_{max}(F'')$, that contains this last node. We show by induction over m, that it is possible to cover all nodes in this way.
Induction basis: In the case of $m = 0$ we have from $n < 2^m$ that we have no nodes. In this case there is nothing to show. Even in the case $m = 1$ we have only one single world, where every dependence atom is true, as mentioned in the fundamentals.
Induction hypothesis: We assume, that there are less than 2^m worlds for which a valid fraction $A_1, \ldots, A_m = A$ exists.
Induction step: We make the step from $m \to m + 1$. We have to show, that there is a covering $A_0, \ldots, A_m$ of A, that means $A_0 \cup \ldots \cup A_m = A$. So we have one more dependence atom, and from $n < 2^m \Rightarrow 2 \cdot n < 2^{m+1}$ follows, that we have up to double as many worlds. From lemma 4.16 we have, that we can chosse a set $A_0 \subseteq A$ with $2 \cdot |A_0| \geq A$. So

there are at most $\lfloor |A|/2 \rfloor$ many worlds left to be covered by the remaining dependence atoms. With the induction hypothesis follows, that $A_0 \cup \ldots \cup A_m = A$.

$\square$

We have seen, that if we have enough dependence atoms in our formula, we can solve it in P. Now we consider the case, in which we have fewer dependence atoms in our formula than states in the Kripke structure. We use this fact, that there are only a few dependence atoms, by testing all possible configuration for the dependence atoms. We call a dependence atom k-ary if it is of the form $=(p_1, \ldots, p_k; q)$ where $p_1, \ldots, p_n$ are atomic propositions and $k \in \mathbb{N}$ fixed. If all of the dependence atoms are k-ary then it is possible to do model checking in P as we will see in the next proposition.

Proposition 4.19. Let m be the number of dependence atoms in the formula F, $n := |S|$ and assume that $2^m \leq n$. Moreover, let us assume, that all dependence atoms in our formula F are k-ary of the form $=(p_1, \ldots, p_k; q)$ with $p_1, \ldots, p_k, q \in AP$ and $k \in \mathbb{N}$ fix. Then $\mathsf{MDL\text{-}MC}(\{\neg, \vee, =(\cdot)\})$ is in P.

Proof. Let $F_k = =(p_1, \ldots, p_k; q)$ be a k-ary dependence atom. We show by induction over k, that $|S_{max}(F_k)| \leq 2^{2^k}$.
Induction basis: $k = 0 \Rightarrow F_0 = =(q)$. $S_{max}(F_0)$ consists at most of two sets, $M_1 = \{s \in S | q \in \pi(s)\}$ and $M_2 = \{s \in S | q \notin \pi(s)\}$. Note that $|S| < 2$ if $M_1 = M_2 = \emptyset$.
Induction hypothesis: Assume for fix $k \in \mathbb{N}$ there is $|S_{max}(F_k)| = 2^{2^k}$ so we have 2^k ways to choose a configuration for $p_k := (p_1, \ldots, p_k)$.
Induction step: Let $F_{k+1} = =(p_1, \ldots, p_{k+1}; q)$ the dependence atom. From the induction hypothesis we have $2^{k+1} = 2 \cdot 2^k$ ways to choose the vector $p_{k+1} = (p_1, \ldots, p_{k+1})$. For every configuration of the vector p, q can be either 0 or 1. It follows that we have up to $2^{2 \cdot 2^k} = 2^{2^{k+1}}$ different sets in $S_{max}(F_{k+1})$.
From $2^m \leq n$ follows that $m \leq \lceil log(n) \rceil$. So we have less or equal $\lceil log(n) \rceil$ many dependence atoms. Let S^i_{max} be the S_{max} set of the i-th dependence atom. We now have to check every combination of elements of S^i_{max} if

$$\forall i \in \{1, \ldots, m\} \exists M_i : \bigcup_{i=1}^{m} M_i = A$$

is possible. The number of these combinations is

$$\begin{aligned}
&(2^{2^k})^{\lceil log(n) \rceil} \\
=&2^{2^k \cdot \lceil log(n) \rceil} \\
=&2^{2^k \cdot \lceil log(n) \rceil} \\
=&(2^{\cdot \lceil log(n) \rceil})^{2^k} \\
\in& n^{O(1)}.
\end{aligned}$$

$\square$

With the proposition 4.18 and proposition 4.19 we have shown the following theorem.

Theorem 4.20. Let $M := \{\neg, \vee, =(\cdot)\}$ be the operator fragment where all dependence atoms are k-ary, with a fixed $k \in \mathbb{N}$. Then $\mathsf{MDL\text{-}MC}(M)$ is in P.

Proof. Let $K = (S, R, \pi)$ be an AP-Kripke structure, m the number of dependence atoms of the given formula and $n := |S|$. Then there are two cases to check.
Case I: $n < 2^m$. This is proposition 4.18.
Case II: $n \geq 2^m$. This is proposition 4.19.

$\square$

5 Conclusion and Open Problems

We have shown in theorem 3.1 that the MDL-MC problem in general is NP-complete. There are some operator fragments such as $\{\vee, \wedge, =\!(\cdot)\}$, $\{\vee, \Box, =\!(\cdot)\}$, and $\{\Diamond, =\!(\cdot)\}$ where model checking stays NP-complete (shown in 3.1, 4.2, and 4.4). In the cases where we disallow $\vee$ and $\Diamond$ the fragment becomes tractable. In theorem 4.20 we only were able to show the tractability for the fragment $\{\neg, \vee, =\!(\cdot)\}$ on k-ary dependence atoms, $k \in \mathbb{N}$. It would be interesting to investigate the complexity over the unrestricted fragment $\{\neg, \vee, =\!(\cdot)\}$. On the other hand a further step could be to analyse the complexity by only allowing k-ary dependence atoms in other fragments as well. Furthermore we could make restrictions on the length of the considered formula or on the size of the Kripke structure.

We have shown the completeness for the NP cases, so one possibility for further research would be the hardness of the P fragments.

As we mentioned above MDL can be seen as a restriction of DCTL which we obtain by combining the concept of dependence with CTL. A next step would be to expand the complexity analysis onto DCTL. In DCTL we have a higher expressive power. By this we would have a statement aboud model checking on logics that are used in practice as it is the case in [CGP99].

Bibliography

[CGP99] E.M. Clarke, O. Grumberg, and D.A. Peled, *Model checking*, Springer, 1999.

[LV10] Peter Lohmann and Heribert Vollmer, *Complexity results for modal dependence logic*, 19th EACSL Annual Conference on Computer Science Logic (A. Dawar and H. Veith, eds.), LNCS, vol. 6247, Springer, 2010, To appear, pp. 411–425.

[Sev09] Merlijn Sevenster, *Model-theoretic and computational properties of modal dependence logic*, Journal of Logic and Computation **19** (2009), no. 6, 1157–1173.

[Vää07] Jouko Väänänen, *Dependence logic: A new approach to independence friendly logic*, London Mathematical Society student texts, no. 70, Cambridge University Press, 2007.

[Vää08] ———, *Modal dependence logic*, New Perspectives on Games and Interaction (Krzysztof R. Apt and Robert van Rooij, eds.), Texts in Logic and Games, vol. 4, Amsterdam University Press, 2008, pp. 237–254.