Interactions and Complexity in Multi-Agent Justification Logic

by

Antonis Achilleos

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

2015
This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

Sergei Artemov

__________________________

Date
Chair of Examining Committee

Robert Haralick

__________________________

Date
Executive Officer

Melvin Fitting

__________________________

Stathis Zachos

__________________________

Samuel Buss

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK
Abstract

Interactions and Complexity in Multi-Agent Justification Logic

by

Antonis Achilleos

Advisor: Distinguished Professor Sergei Artemov

Justification Logic is the logic which introduces justifications to the epistemic setting. In contrast to Modal Logic, when an agent believes (or knows) a certain claim, in Justification Logic we assume the agent believes the claim because of a certain justification. Therefore, instead of having formulas that represent the belief of a claim (ex. $\Box \phi$ or $K\phi$), we have formulas that represent that the belief of a claim follows from a provided justification (ex. $t : \phi$). The original Justification Logic is $LP$, the Logic of Proofs, and was introduced by Artemov in 1995 as a link between Intuitionistic Truth and Gödel proofs in Peano Arithmetic.

The complexity of Justification Logic was first studied by Kuznets in 2000. He demonstrated that for many justification logics, their derivability problem (and thus their satisfiability problem) is in the second level of the Polynomial
Hierarchy, a result which was shown to be tight and which was later extended to more justification logics. In fact, so far, given reasonable assumptions, every single-agent justification logic whose complexity has been settled has its satisfiability problem in the second level of the Polynomial Hierarchy. This result is nicely contrasted to Modal Logic, as the corresponding modal systems are \textsc{PSPACE}-complete.

We investigate the complexity of Justification Logic and Modal Logic when we allow multiple agents whose justifications affect each other – by including some combination of the axioms $t:_i \phi \rightarrow t:_j \phi$ and $t:_i \phi \rightarrow t!:_j t:_i \phi$ (modal cases: $\Box_i \phi \rightarrow \Box_j \phi$). We discover complexity jumps new for the field of Justification Logic: in addition to logics with their satisfiability problem in the second level of the polynomial hierarchy (as is the usual case until now), there are logics that have \textsc{PSPACE}-complete, \textsc{EXP}-complete and even \textsc{NEXP}-complete satisfiability problems.

It is notable how the behavior of several of these justification logics mirrors the behavior of the corresponding multi-modal logics when we restrict modal formulas (in negation normal form) to use no diamonds. Thus we first study the complexity of such diamond-free modal logics and then we deduce complexity properties for the justification logic systems. On the other hand, it is similarly notable how certain lower complexity bounds – the \textsc{NEXP}-
hardness bound and the general $\Sigma^p_2$-hardness bound we present – are more
dependent on the behavior of the justifications. The complexity results are
interesting for Modal Logic as well, as we give hardness results that hold
even for the diamond-free, 1-variable fragments of these multi-modal logics
and then we determine the complexity of these logics in a general case.
I want to thank many people. Here I will thank only a few.

First, I want to thank all the members of my committee, Professors Sergei Artemov, Melvin Fitting, Stathis Zachos, and Sam Buss. Their guidance and suggestions were invaluable, both before and after the completion of this thesis. I was glad to have each of those individuals as a member of my committee.

I am thankful to my advisor, Sergei Artemov, for his unwavering support and his advice all these years. His belief that my work was significant was sometimes stronger than mine and always a great encouragement.

Stathis Zachos is perhaps the person who has influenced my academic path more than anyone. I have known him for the past fourteen years; he was there on the first day I went to college and on the day I defended my PhD thesis. I am grateful to him for the insight he always has to offer and for his sound guidance, both academic and otherwise.
Lina Garcia, who used to be the Assistant Program Officer of the Computer Science department of the Graduate Center was the first person I met when I first came to the department. Her assistance was invaluable even after she left her position in the program. After Lina left, I relied heavily on Dilvania Rodriguez and Amauris Vega, who replaced her.

I am grateful to my coauthors and friends, Michalis Lampis and Valia Mitsou. They are great to have, both as coauthors and friends.

Of course, I thank my parents, who always supported my decisions for my future, even if those resulted in a PhD thesis.

Finally, I want to thank my wife, Matina. I still do not know why she got married to me and why she has yet to change her mind. I thank her for her patience, because it has been long and far; I thank her for her belief in me, because I know I can be weird; I thank her for being there, because it meant a lot.

This list is not complete, of course. There are more people to thank and more reasons to thank the ones above. As anyone who knows me can tell, I tend to use few words and these are enough. So, thank you, everyone else.
Contents

Acknowledgments vi

1 Introduction 1

2 Definitions 11

2.1 Modal Logic . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

2.2 Justification Logic . . . . . . . . . . . . . . . . . . . . . . . . 31

2.2.1 Syntax . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32

2.2.2 Axiomatizations . . . . . . . . . . . . . . . . . . . . . . . 33

2.2.3 Semantics . . . . . . . . . . . . . . . . . . . . . . . . . . 36

2.2.4 Explicit Modal Logic . . . . . . . . . . . . . . . . . . . . 48

2.2.5 Two versions of the *-calculus and admissible evidence 50

2.3 Computational Complexity . . . . . . . . . . . . . . . . . . . 55

2.3.1 Short Background on Complexity Theory . . . . . . . . 56

2.3.2 The Complexity of Modal Logic . . . . . . . . . . . . . 66
3 The Complexity of Single-agent Justification Logic 76

3.1 Reflected Fragments and the $\ast$-calculus ............... 77
3.2 Tableaux through Mkrtychev Models for One Agent ........ 82
3.3 Consistency and Fitting Models .............................. 85
3.4 A Lower Complexity Bound ................................. 96

4 Justification Logic and Modal Logic with Multiple Agents 108

4.1 Multi-Agent Justification Logic ............................. 108
  4.1.1 Semantics ........................................... 114
  4.1.2 Some Examples and Graphical Representations ........ 121
  4.1.3 The $\ast$-calculus. .................................. 126
4.2 The Complexity of the $\ast$-calculus on Multiple Agents .... 129
4.3 Tableaux through Mkrtychev Models ........................... 132
4.4 A Small Model Theorem ................................. 133

5 The Complexity of Interacting Agents: Conversion 138

5.1 Multi-Modal Logic .......................... 138
  5.1.1 Relating Diamond-free Modal Logic and Justification Logic 140
  5.1.2 Work on Dependent Modal Logic ........... 143
  5.1.3 Hardness Without Diamonds ....................... 148
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.4 A General Characterization</td>
<td>168</td>
</tr>
<tr>
<td>5.2 The Complexity of Multi-Agent Justification Logic with Conversion</td>
<td>178</td>
</tr>
<tr>
<td>6 The Complexity of Interacting Agents: Conversion and Verification</td>
<td>185</td>
</tr>
<tr>
<td>6.1 Two-agent Justification Logic</td>
<td>186</td>
</tr>
<tr>
<td>6.1.1 Tableaux and Satisfiability - the Method and the Tools</td>
<td>186</td>
</tr>
<tr>
<td>6.1.2 Complexity Results</td>
<td>191</td>
</tr>
<tr>
<td>6.2 Tableau Procedures for Multi-Agent Justification Logic</td>
<td>197</td>
</tr>
<tr>
<td>6.2.1 Inside the Second Level of the Polynomial Hierarchy</td>
<td>206</td>
</tr>
<tr>
<td>6.2.2 More Hardness with Verification</td>
<td>207</td>
</tr>
<tr>
<td>6.3 A NEXP-complete Justification Logic</td>
<td>212</td>
</tr>
<tr>
<td>6.3.1 Proof of Theorem 6.3.1</td>
<td>215</td>
</tr>
<tr>
<td>6.3.2 Justification Logic is harder than Modal Logic</td>
<td>231</td>
</tr>
<tr>
<td>7 Ending Remarks</td>
<td>236</td>
</tr>
<tr>
<td>Bibliography</td>
<td>241</td>
</tr>
</tbody>
</table>
List of Figures

2.1 The computation tree of an alternating TM (of exactly two
choices a each step). .............................................. 59
2.2 A hierarchy of complexity classes .............................. 65
2.3 The complexity of Unimodal Logic: in the figure, unless oth-
erwise specified, modal satisfiability (or validity, provability)
refers to Unimodal Logic (one modality). ....................... 75
3.1 The complexity of single-agent Justification and Modal Logic . 107
4.1 The logic $J_7$ graphically. ..................................... 124
4.2 The logic $J_8$ or $J_9$, depending on how we mark the edges. . 124
4.3 The logic $J_{12}$ graphically. ................................... 125
4.4 The logic $J_{13}$ graphically. ................................... 125
4.5 The logic $J_{14}$ graphically. ................................... 125
4.6 The logic $J_{15}$ can be any of these two, or $J_{14}$. ............ 126
4.7 The logic $J_{16}$ graphically. ................................... 126
LIST OF FIGURES

4.8 First results for multi-agent Justification Logic . . . . . . . . . 137

5.1 The complexity of Diamond-free Modal Logic . . . . . . . . . 179

5.2 The complexity of Diamond-free Modal Logic and Multi-Agent
Justification Logic with Conversion . . . . . . . . . . . . . . . 184

6.1 The logic $J_1$ graphically. . . . . . . . . . . . . . . . . . . . . . . 208

6.2 The logic $J_2$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 208

6.3 Complexity characterizations for Multi-Agent Justification Logic211

6.4 Graphical representation of EXP-hard logic $J_H$. . . . . . . 213

7.1 The complexity of Multi-Agent Justification Logic. . . . . . . 240
## List of Tables

2.1 A table of normal modal logics ........................................ 15
2.2 Propositional Tableau rules ............................................. 24
2.3 Modal Tableau rules ..................................................... 26
2.4 The $^F$-calculus for $J$: where $\mathcal{F} = (W, R)$ .............. 52
2.5 The $^*$-calculus for $J$ ............................................... 54

3.1 Tableau rules for $J$, JT, J4, and LP ................................ 83
3.2 Tableau for JD .......................................................... 86
3.3 Tableau for JD4 .......................................................... 93

4.1 The $^F$-calculus for $J$ ............................................... 128
4.2 Tableau rules for $(N, C, \leftrightarrow, F)_{CS}$ when no agent has the Consistency axiom .................................................. 133

5.1 Tableau rules for K. Notice that we do not use a truth prefix. . 150
5.2 The rules for $D_2 \oplus_\subseteq K$ ..................................... 151
5.3 Tableau rules for the diamond-free fragment of $D_2 \oplus_\subseteq K_4$ . . . 152
5.4 Tableau rules for the diamond-free fragment of $D \oplus_\subseteq K_4$ . . . 153
5.5 Tableau rules for the diamond-free fragment of $D_{4_2} \oplus_\subseteq K_4$ . . 153
5.6 Tableau rules for the diamond-free fragment of $(N, \subset, F)$ . . . 170
5.7 Tableau rules for the diamond-free fragment of $(N, \subset, F)$ . . . 180

6.1 Tableau rules for $(JD \times JD)_{CS}$ . . . . . . . . . . . . . . . . . . . . . 187
6.2 Tableau rules for logics without interactions. We use two of these rules: the ones that correspond to $J_1$ and $J_2$. . . . . . . . . . . . . . 192
6.3 Tableau rules for when $\times_\circ = \times_{\Pi}$. . . . . . . . . . . . . . . . . . . . . 194
6.4 Tableau rules for $\times_\circ = \times_1$. . . . . . . . . . . . . . . . . . . . . . . . . . . 194
6.5 Tableau rules for $(JT \times_1 JD)_{CS}$ and $(LP \times_1 JD)_{CS}$. . . . . . . 196
6.6 The tableau rules for $J$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 203
6.7 Tableau rules for $J_1, J_2$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 209
6.8 Tableau rules for $M_H$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 233
Chapter 1

Introduction

Justification Logic is a relatively new field, which complements Modal Logic, introducing justifications to modal epistemology. It is a family of logics which models the way justifications interact with statements and can be viewed as an explicit counterpart of Epistemic Modal Logic. It is often the case that we want to express statements of the form “agent A knows/believes $\phi$ because of justification $t$” and Justification Logic offers the means to formalize situation where either the distinction between different justifications is important, or a given claim is provided together with an appropriate justification for it. This allows for a finer analysis than the one provided by Modal Logic.

The first justification logic was $\mathbf{LP}$, the Logic of Proofs, introduced by Artemov in 1995 [Art01]. $\mathbf{LP}$ is an explicit version of $\mathbf{S4}$, using what are called justification terms instead of modal boxes – thus making the assertion that a statement is known/provable explicit by providing a valid justification/proof.
CHAPTER 1. INTRODUCTION

Its purpose was to link Intuitionistic Logic and Peano Arithmetic, explaining the relationship between the intuitionistic view of truth as something proven and classical proofs in PA. Thus, in LP, a justification represented an actual proof in PA.

Since then Justification Logic has grown into a broad system of explicit versions of Modal Logic, where “proofs” are generalized to “justifications”, allowing for a more relaxed and inclusive interpretation of justification terms as justifications for statements an agent believes. This allows us to have explicit versions of many epistemic modal logics, such as $K$, $D$, $T$, $K4$, $D4$, $S4$, $KD45$, $S5$, which correspond to $J$, $JD$, $JT$, $J4$, $JD4$, $LP$, $JD45$, and $JD45$ respectively [Art01, Bre00, Pac05, Rub06a, Rub06b, Rub06d, Rub06c]. It is known that all theorems of one of those justification logics can be translated to theorems of the corresponding modal logic by replacing each term by a box ($\Box$). On the other hand, for every modal theorem we can find a theorem of the corresponding justification logic by replacing each box by an appropriate term. This was already proven in the original paper by Artemov for LP and $S4$, as part of the main result [Art01]. By now Justification Logic has its own versions of logics with Public Announcement, actions, and probability [BKS11, Ren11, Mil14, KMOS] and its own type system [AB07, PP14].

Justification formulas are formed using propositional connectives and jus-
CHAPTER 1. INTRODUCTION

The need for Multi-Agent Justification Logic. Yavorskaya in [Yav08] presents two-agent versions of LP with interactions between the agents (agent 1 and agent 2). To describe the interactions, she introduced new operators on terms. In $\text{LP}^2$ the two agents do not interact with each other at all. In $\text{LP}^2_\downarrow$ one of the agents (agent 2) is more knowledgeable than the other: a justification of $\phi$ for agent 1 can be converted to a justification of $\phi$ for agent 2. In $\text{LP}^2_\uparrow$, agent 2 is aware of the knowledge of agent 1: if agent 1 has justification $t$ of a formula $\phi$, then agent 2 has justification of the fact that agent 1 has justification $t$ of a formula $\phi$. $\text{LP}^2_{\uparrow\downarrow}$ and $\text{LP}^2_{\downarrow\uparrow}$ respectively extend these logics by also incorporating the converse of these interactions (swapping 1 and 2). We claim this is an important approach that will further illuminate the role of justification in epistemological situations.

We extend Yavorskaya’s system to allow multiple (more than two) agents, so that each agent can be based on a different justification logic. Our main goal is to develop and study the complexity of a family of multi-agent justification logics with interactions among the agents. Here, interactions are not really actions, but interdependencies among the justifications of different agents, as characterized by the Conversion axiom ($t :_i \phi \rightarrow t :_j \phi$) and the
Verification axiom \( (t \, ::_i \phi \rightarrow t \, ::_j \phi) \). The Conversion axiom indicates that some agent’s (say \( i \)'s) justifications are also justifications accepted by some other agent (in this case \( j \)). The Verification axiom indicates that if agent \( i \) has justification \( t \) for a certain statement (we can call it belief), then agent \( j \) is aware of this and \( j \)'s justification for this fact is just the ability to check \( i \)'s justification. In other words, \( j \) can verify \( i \)'s justifications. An analog is that when someone presents us with a proof of a fact, that proof is the justification of that fact, while the justification of the fact that the proof is a valid proof of the fact is checking (verifying) the actual proof. The resulting system is thus particularly flexible and can model several diverse epistemological situations. The Conversion interaction is symbolized by \( \supset \) and the Verification interaction by \( \rightarrow \); thus if \( i \supset j \), then \( t \, ::_i \phi \rightarrow t \, ::_j \phi \) is an axiom and if \( i \rightarrow j \), then \( t \, ::_i \phi \rightarrow t \, ::_j \phi \) is an axiom.

**Example: Knowledge and Belief.** An agent has somehow obtained two pieces of evidence, the first being evidence for \( \phi \) and the second for \( \neg \phi \). After an additional inquiry the agent discovers that the second piece of evidence has been compromised whereas the first was confirmed. On this basis, the agent attains the knowledge of \( \phi \). Lets attempt to model this situation in Bi-modal and Two-agent Justification logic. Bi-modal logic is
insufficient to model this situation: we need to distinguish between two types of belief: $B$ and $K$, where $K$ indicates knowledge and $B$ some kind of belief. Then, initially the agent would have the beliefs $B\phi$, $B\neg\phi$, while the fact that the agent determines the first evidence as confirmed can be formalized as $K(B\chi \rightarrow \chi)$. We can already see that Modal logic’s language presents difficulties in expressing the desired distinction between the two pieces of evidence. From $B\phi$ we can derive $KB\phi$ (we assume that the agent has at least knowledge of the evidence they have obtained) and from $K(B\phi \rightarrow \phi)$ we can derive $KB\phi \rightarrow K\phi$; from the two we derive $K\phi$ and then $\phi$. Similarly we can derive $\neg\phi$ and we reach an inconsistency.

We can formalize the scenario in a natural and intrinsically faithful way using a two-agent justification logic, where $s :_1 X$ stands for ‘$s$ is an evidence for $X’$ and $t :_2 X$ denotes ‘$t$ is a conclusive/knowledge producing evidence for $X’$, equipped with Verification: $1 \subseteq 2$. The situation can be formalized by the set \{u :_1 \phi, v :_1 \neg\phi, c :_2 (u :_1 X \rightarrow X)\}. This set is consistent, which can easily be satisfied in a model (defined later on). We can derive knowledge of $\phi$ in the following way: $u :_1 \phi \rightarrow!u :_2 u :_1 \phi$ is an instance of the Verification axiom and together with $u :_1 \phi$ yields $!u :_2 u :_1 \phi$ and this in turn together with $c :_2 (u :_1 \phi \rightarrow \phi)$ and the application axiom (which is the counterpart of the $K$ axiom of Modal Logic and which we present later; an instance of it is
CHAPTER 1. INTRODUCTION

$c : 2 ( u : 1 \phi \to \phi ) \to !u : 2 u : 1 \phi \to [ c : !u ] : 2 \phi$ yield $[ c : !u ] : 2 \phi$. Then, $\phi$ is known with justification $[ c : !u ]$.

The above scenario can also be reformulated as a situation of two agents, where the second has more reliable sources than the first (enough to accept only evidence that yield knowledge) and the first one reports to the second one the two pieces of evidence for $\phi$ and $\neg \phi$; the second agent would then have obtained information about the reliability of the evidence that the first agent provides, i.e. that the second piece of evidence is compromised, while the first one is confirmed. The analysis would then be the same.\footnote{Of course, we could also handle the situation in Modal Logic by using more modalities. This treatment would not appropriately reflect the nature of the issue, though, and it would be hard to apply the analysis in other similar situations.}

**Example: Common Awareness through Interactions.** Another interesting situation arises when there are several agents who accept different views from each other, but each agent is aware of the other’s views. For example, there may be three agents from three respective religions, based on three respective holy books. Each agent may be completely aware of the contents of all three books, so each agent is completely aware of every agent’s beliefs, but does not necessarily embrace them. Then, the underlying logic would have three agents, $a, b, c$, such that $a \subset b \subset c \subset a$. Furthermore,
distinguishing between justifications makes sense, for example, when two (or all) of the agents accept a certain prophet’s teachings.

**Similarly:** During a trial, two lawyers, A and B, present evidence to support their case. A presents witness $a$ who claims A’s client is right, while B presents witness $b$ who claims B’s client is right (and lets assume they both believe their respective witnesses’ claims). Furthermore, A also presents document $d$ that strongly support that whoever is right is entitled to receive the sum of $10$ from the other. Both lawyers accept the document and their respective witness’ claims as valid evidence, while they are aware of (and reject) each other’s beliefs on the case. Similarly to the above, this scenario can be formalized by a two-agent logic where both agents are equipped with Consistency and $1 \subset 2 \subset 1$.

**The complexity of Justification Logic and Modal Logic.** Determining the complexity of the satisfiability problem of a logic is an important goal. It is important from a logical point of view because we feel that one only really knows a logic and what its formulas represent after we have determined its complexity. It is important from a computational perspective, because having an algorithm for a logic’s satisfiability (or derivability, or model checking...) means that you have a general method to solve any problem you can
formalize in that logic.

Justification Logic has intriguing complexity properties, especially when contrasted to the complexity properties of Modal Logic. For Modal Logic, Ladner has shown that satisfiability (and thus, provability) for $K$, $D$, $T$, $K4$, $D4$, $S4$ is $\text{PSPACE}$-complete \cite{Lad77}. On the other hand, provability for the corresponding justification logics is in the second level of the Polynomial Hierarchy and specifically in $\Pi_2^p$ (\cite{Kuz00, Kuz08a, Ach14b}), while Krupski has shown that LP-provability for formulas of the form $t : \phi$ is in $\text{coNP}$ \cite{Kru06}). Furthermore, there is an easily recognizable class of terms $\mathcal{T}$ so that if a formula $\phi$ is provable then $t : \phi$ is provable for some $t \in \mathcal{T}$ and for $t \in \mathcal{T}$, the provability of $t : \phi$ is in $\mathcal{P}$ \cite{AK13}). Of course, this does not simplify theoremhood of $S4$ or of any of the other modal logics (which is $\text{PSPACE}$-complete), but it demonstrates the complexity-theoretic difference between determining the provability of a modal statement and determining the provability of a modal statement when given appropriate evidence. It further demonstrates that Justification Logic internalizes not only formal, Hilbert-style proofs of its theorems, but also proofs in a computationally significant sense: justification terms in $\mathcal{T}$ can act as witnesses for a polynomial-time algorithm which verifies theoremhood of justified formulas.

We study the complexity of the derivability (equivalently, the satisfia-
bility) problem for Justification Logic in a multi-agent setting. We thus introduce and study multi-agent logics which are combinations of two or more different justification logics. During this process we discover several unexpected complexity jumps. This is a different situation from what is the case for all (pure) justification logics whose complexity has been studied. Specifically, we discover \textbf{PSPACE}-complete, \textbf{EXP}-complete, and even \textbf{NEXP}-complete justification logics. This latter result is perhaps the most surprising, as it gives us the first justification logic with known complexity higher than its corresponding modal logic (given standard complexity-theoretic assumptions).

Justification Logic is naturally connected to Modal Logic. This is also true with respect to its complexity properties. Although at first glance, the techniques and results concerning the complexity of Justification Logic seem completely foreign to their counterparts for Modal Logic, when we work with agents with consistent beliefs and especially in a multi-agent setting, this view changes. In fact, we can see that Justification Logic when tested for satisfiability tends to behave, at least in part, a lot like Modal Logic if there were no diamonds. Therefore, we have great interest in studying the complexity of diamond-free fragments of Modal Logic (in negation normal form), which we do.
CHAPTER 1.  INTRODUCTION

Overview:  We first give the basics by introducing Justification Logic and Modal Logic and providing a very brief overview of the complexity-theoretic classes and tools we will use. Following this we present old and new results and techniques relative to the complexity of single-agent Justification Logic. We proceed to define a multi-agent family of justification logics with the interactions described above by generalizing the two-agent versions of LP by Yavorskaya and we give basic facts and definitions for Multi-modal Logic. Afterwards, we examine the complexity of the satisifiability problem for these logics. We first give some immediate results: a general \( \text{NEXP} \) upper bound based on a small-model theorem; and an immediate \( \Sigma^p_2 \) upper bound for logics without agents of consistent beliefs, using Kuznets’ techniques from [Kuz00]. Then we examine the complexity of Diamond-free Modal Logic and Justification Logic with just the Conversion interaction; we manage to give a complete picture for this situation. Following this we cover the cases of only two agents and, again, we are able to provide the whole picture with respect to the complexity of each logic. Afterwards we determine a wide class of logics for which the satisifiability problem remains in the second level of the polynomial hierarchy; we also prove tightness for the general upper bound: we provide a \( \text{NEXP} \)-complete Justification Logic. We conclude with general remarks, open problems, and possible future directions.
Chapter 2
Definitions

In this chapter we give background from fields that are necessary for the study of the complexity of Multi-Agent Justification Logic and Modal Logic. Not surprisingly, these include Complexity Theory, Justification Logic, and Modal Logic. We go through these topics in reverse order.

2.1 Modal Logic

Modal Logic is the logic of necessity and possibility, of knowledge and belief, of obligation and permission, or the logic of temporal reasoning. Having its roots back in Aristotle, who examined the rules of necessary truth and possible truth, Modal Logic was reintroduced through the work of Lewis (ex. [Lewa, Lewb]) and became widespread after the introduction of Kripke semantics [Kri63]. For an overview of Modal Epistemic Logic, the reader can see [FHMV95, PBV].
CHAPTER 2. DEFINITIONS

Syntax

To construct the formulas of Modal Logic we first need propositional variables, of which we assume there is a (countably) infinite amount: $p_1, p_2, \ldots$. We call the set of propositional variables $Prop$. Then, we construct modal formulas using propositional connectives ($\land, \lor, \to, \neg$), modal operators ($\Box, \Diamond$), and the constant $\bot$; that is, if $\phi, \psi$ are formulas, then so are $\phi \land \psi$, $\phi \lor \psi$, $\phi \to \psi$, $\neg \phi$, $\Box \phi$, $\Diamond \phi$. The set of modal formulas is $L_M$ ($= L_M^1$ in a multimodal context). For a formula $\phi$ (modal like here or otherwise), let $\text{sub}(\phi)$ be the set of all its subformulas. It is a simple observation that $|\text{sub}(\phi)| \leq |\phi|$,\(^1\) as each subformula uniquely corresponds either to a propositional variable or to an instance of a connective.

There are several ways to save on symbols, and depending on convenience, we may assume a more restricted language and that certain symbols are defined from others (i.e. $\bot := p \land \neg p$, or $\neg \phi := \phi \to \bot$, or $\Diamond := \neg \Box \neg$, etc).

Axiomatizations

To present the family of (Uni)Modal Logic, we follow an axiomatic approach.

Usually when one presents a modal logic (or several) in order to study the

\(^1\)Throughout this thesis, we assume some reasonable definition for $|\phi|$; to make this more precise, it makes sense to define $|\phi|$ as the number of symbols in $\phi$ (with or without parentheses).
complexity of its (their) satisfiability problem, they do this by presenting the semantics first and the restrictions that correspond to each of these logics. This makes sense, as it is easier to present Modal Logic this way and we usually work on the models of Modal Logic anyway when we are interested in complexity issues. As we want to draw parallels to Justification Logic, though, and Justification Logic is better presented using an axiomatic approach, it seems more appropriate to present Modal Logic in an axiomatic way as well.

Normal modal logics use a sufficient amount of propositional tautological schemes (we can take all tautologies, or a finite axiomatization of Propositional Logic) and the axiom

\[ K: \Box \phi \land \Box (\phi \rightarrow \psi) \rightarrow \Box \psi. \]

Furthermore, they use two rules: Modus Ponens and the Necessitation Rule:

\[ \frac{\phi}{\Box \phi}, \]

which claims that all provable formulas are provably necessary (or known, believed, etc).²

²When we derive from assumptions, we need to specify that \( \Box \phi \) is derivable from \( \phi \) only when \( \phi \) can be derived independently (without assumptions). That is, we would state the rule above as \( \frac{\phi}{\Gamma \vdash \Box \phi} \).
modal logic, K. We can extend K with more axioms and thus introduce more normal modal logics. The axioms are:

\[ D: \neg \Box \bot; \]
\[ T: \Box \phi \rightarrow \phi; \]
\[ 4: \Box \phi \rightarrow \Box \Box \phi; \]
\[ 5: \neg \Box \phi \rightarrow \Box \neg \Box \phi. \]

Modal logic D is K extended by the axiom D, T is K extended by T, K4 is K extended by 4, D4 is K extended by both D and 4 (so it is D extended by 4), S4 is K extended by both T and 4, KD45 is D4 extended by 5, and S5 is S4 extended by 5. It is probably safe to claim as a general rule that if the name of a modal logic is a sequence of letters, these letters correspond to the axioms by which this logic extends K – with S4, S5 as obvious exceptions. For example, KT45 is S5.

Observation 1. All these modal logics are consistent. Simply notice that we can map all modal formulas to propositional formulas by removing all boxes. Then, modal axioms are mapped to propositional tautologies and the rules preserve theoremhood. Therefore, if \( \bot \) can be derived in a modal logic, then it can be derived in the propositional calculus, which is obviously not true.
CHAPTER 2. DEFINITIONS

<table>
<thead>
<tr>
<th>Modal Logic</th>
<th>Modal Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>D</td>
<td>K  D</td>
</tr>
<tr>
<td>T</td>
<td>K  T</td>
</tr>
<tr>
<td>K4</td>
<td>K  4</td>
</tr>
<tr>
<td>D4</td>
<td>K  D  4</td>
</tr>
<tr>
<td>S4</td>
<td>K  T 4</td>
</tr>
<tr>
<td>S5</td>
<td>K  T 4  5</td>
</tr>
<tr>
<td>KD45</td>
<td>K  D 4  5</td>
</tr>
</tbody>
</table>

Table 2.1: A table of normal modal logics

This argument can be repeated for all logics we present in this thesis and thus will be omitted.

Semantics

Modal Logic comes with many kinds of semantics, but the most prevalent ones are Kripke semantics. A Kripke model is a set of worlds or states, an accessibility relation and a truth valuation for every state. There, \( \square \phi \) is interpreted as “\( \phi \) is true in all accessible worlds” and \( \Diamond \phi \) is interpreted as “\( \phi \) is true in at least one accessible world.” In other words, knowledge/belief/necessity is the same as truth in all situations one can imagine/considers possible.

A Kripke model is a triple \( \mathcal{M} = (W, R, \mathcal{V}) \), where \( R \subseteq W \times W \) and for every propositional variable \( p \), \( \mathcal{V}(p) \subseteq W \). Then, \( (W, R) \) is called a frame and \( R \) is called an accessibility relation. We define the truth relation \( \models \) between models, worlds and formulas in the following recursive way:
\[ M, a \not\models \bot; \]
\[ M, a \models p \text{ iff } a \in \mathcal{V}(p); \]
\[ M, a \models \neg \phi \text{ iff } M, a \not\models \phi; \]
\[ M, a \models \phi \land \psi \text{ iff both } M, a \models \phi \text{ and } M, a \models \psi; \]
\[ M, a \models \phi \lor \psi \text{ iff } M, a \models \phi \text{ or } M, a \models \psi; \]
\[ M, a \models \Diamond \phi \text{ iff there is some } b \in W \text{ such that } aRb \text{ and } M, b \models \phi; \]
\[ M, a \models \Box \phi \text{ iff for all } b \in W \text{ such that } aRb \text{ it is the case that } M, b \models \phi. \]

Each modal logic is associated with a specific class of frames. That is, there are certain restrictions (or closure conditions) we impose on the kinds of frames we can use for the models of a modal logic.

- \( K \) is associated with the class of all frames – there are no restrictions on \( K \)-models.
- \( D \) is associated with the class of frames for which the accessibility relation is serial (for each world \( a \) there is a world \( b \) s.t. \( aRb \)).
- \( T \) is associated with the class of frames for which the accessibility relation is reflexive (for each world \( a, aRa \)).
• K4 is associated with the class of frames for which the accessibility relation is transitive (for every $a, b, c$ worlds, $aRb$ and $bRc$ imply that $aRc$).

• D4 is associated with the class of frames for which the accessibility relation is both serial and transitive.

• S4 is associated with the class of frames for which the accessibility relation is reflexive and transitive.

• S5 is associated with the class of frames for which the accessibility relation is reflexive, transitive and euclidean (if $aRb, c$, then $bRc$); equivalently, it is reflexive, symmetric, and transitive: it is an equivalence relation.

• KD45 is associated with the class of frames for which the accessibility relation is serial, transitive, and euclidean.

In fact, for any logic that results from a combination of these axioms, we can straightforwardly give frame conditions, as each axiom gives its own condition for the accessibility relation: $D$ gives seriality; $T$ gives reflexivity; 4 gives transitivity; 5 gives euclidean relations.

Let $M$ be one of the modal logics above. A modal formula $\phi$ is called
Chapter 2. Definitions

M-satisfiable (or just satisfiable if M is clear from the context) if there is a M-model M and a state a of that model, such that M, a ⊨ φ. In that case we say that φ is true/satisfied at a, or that M satisfies φ at a. φ is M-valid if it is true at all worlds of all M-models; it is called valid for model M if it is true at all worlds of M.

Theorem 2.1.1 (Completeness). Each modal logic introduced above is sound and complete with respect to its corresponding Kripke models.

Proof. To prove soundness, it suffices to demonstrate that for every instance A of an axiom of modal logic M, model M = (W, R, V) for M, and u ∈ W, M, u ⊨ A: if M, u ⊨ φ and M, u ⊨ φ → ψ, then it must also be the case that M, u ⊨ ψ; furthermore, if φ is true in all worlds, then so is □φ; therefore, the derivation rules preserve validity in a model. We do not need to concern ourselves with propositional axioms: each of those is a substitution instance of a propositional tautology, so they always hold at u. We examine the remaining axioms:

K: If M, u ⊨ □φ ∧ □(φ → ψ), then for every uRv, M, v ⊨ φ and M, v ⊨ φ → ψ, so for every uRv, M, v ⊨ ψ, concluding that M, u ⊨ □ψ.

D: M, u ⊨ □⊥ exactly when there are no accessible worlds from u; since R is serial, there is at least one v ∈ W, such that uRv.
T: Since $uRu$, if $\mathcal{M}, u \models \Box \phi$, then $\mathcal{M}, u \models \phi$.

4: If $\mathcal{M}, u \models \Box \phi$, for every $uRv$, if for some $w \in W$, $vRw$, then $uRw$ (since $R$ is transitive), so $\mathcal{M}, w \models \phi$; therefore, $\mathcal{M}, v \models \Box \phi$; therefore $\mathcal{M}, u \models \Box \Box \phi$.

5: If $\mathcal{M}, u \models \neg \Box \phi$, then there is some $w \in W$ so that $uRw$ and $\mathcal{M}, w \models \neg \phi$; if $uRv$, then $vRw$ (since $R$ is euclidean), so $\mathcal{M}, v \models \neg \Box \phi$; therefore, $\mathcal{M}, u \models \Box \neg \Box \phi$.

Completeness will be proven using a canonical model construction. Let $W$ be the set of all maximal consistent subsets of $L_M$ – that is, maximal subsets $S$ of $L_M$ so that $S \not\vdash M \bot$. We know that $W$ is not empty, because $M$ is consistent. For every $\Gamma \in W$, let $\Gamma^\# = \{\phi \in L_M | \Box \phi \in \Gamma\}$. $R$ is a binary relation on $W$, such that $\Gamma R \Delta$ if and only if $\Gamma^\# \subseteq \Delta$. Finally, $\mathcal{V} : \text{Prop} \rightarrow 2^W$ is such that $\mathcal{V}(p) = \{\Gamma \in W | p \in \Gamma\}$. The canonical model is $\mathcal{M} = (W, R, \mathcal{V})$.

Define the relation between worlds of the canonical models and formulas of $L_M$, $\models$, as in the definition of models.

**Lemma 2.1.2** (Truth Lemma). For all $\Gamma \in W$, $\phi \in L_M$, $\mathcal{M}, \Gamma \models \phi$ if and only if $\phi \in \Gamma$. 
Proof. By induction on the structure of \( \phi \). The cases for \( \phi = p \), a propositional variable, \( \bot \), or \( \psi_1 \rightarrow \psi_2 \), are immediate from the definition of \( \mathcal{V} \) and \( \models \). We examine the case when \( \phi = \Box \psi \). If \( \Box \psi \notin \Gamma \), then \( \neg \Box \psi \in \Gamma \); if \( \mathcal{M}, \Gamma \models \Box \psi \), then for every \( \Gamma R \Delta \), \( \psi \in \Delta \). Therefore, \( \Gamma^\# \cup \{ \neg \psi \} \) is inconsistent – meaning that there is a finite \( \Phi \subseteq \Gamma^\# \) so that \( \neg \psi \land \land \Phi \vdash \bot \), so \( \vdash \psi \lor \neg \land \Phi \), resulting in \( \vdash \Box (\psi \lor \neg \land \Phi) \), i.e. \( \vdash \Box (\land \Phi \rightarrow \psi) \); this gives \( \vdash \Box \land \Phi \rightarrow \Box \psi \). In other words, \( \Gamma \vdash \Box \psi \), so \( \Gamma \cup \{ \neg \Box \psi \} \) is inconsistent, therefore \( \Box \psi \in \Gamma \).

For the other direction, if \( \Box \psi \in \Gamma \), then for every \( \Delta \in \mathcal{W} \) such that \( \Gamma R \Delta \), \( \psi \in \Delta \), so \( \Delta \models \psi \) – which means that \( \mathcal{M}, \Gamma \models \Box \psi \) and completes the proof. \( \square \)

The canonical model is, indeed, a model for \( \mathcal{M} \): The canonical model is, of course, a Kripke model. To establish that it is also a model for \( \mathcal{M} \) (an \( \mathcal{M} \)-model), we must show that the conditions expected from \( R \) in an \( \mathcal{M} \)-model are satisfied in the canonical model.

If \( \mathcal{M} \) has axiom \( T \), then \( R \) is reflexive. For this, we just need that if \( \Gamma \in \mathcal{W} \),
then \( \Gamma^\# \subseteq \Gamma \). If \( \phi \in \Gamma^\# \), then \( \Box \phi \in \Gamma \). Because of the Factivity axiom,
\( \neg \phi \notin \Gamma \), since \( \{ t : \phi, \neg \phi \} \) is inconsistent. Therefore, as \( \Gamma \) is maximal consistent, \( \phi \in \Gamma \).
If $M$ has axiom $D$, then $R$ is serial. To establish this, we just need to show that $\Gamma^\#$ is consistent. If it is not, then there are formulas $\phi_1, \ldots, \phi_k \in \Gamma^\#$ s.t. $\phi_1, \ldots, \phi_k \vdash \bot$. This means that $\Box \phi_1, \ldots, \Box \phi_k \in \Gamma$, and $\Box \phi_1, \ldots, \Box \phi_k \vdash \Box \bot$ (by induction on the derivation, using axiom $K$), which is a contradiction.

If $M$ has axiom 4 and $\Gamma R \Delta R E$, then $\Gamma R E$. If $\Box \phi \in \Gamma$, then $\Box \Box \phi \in \Gamma$.

Therefore, $\Box \phi \in \Gamma^\#$. So, if $\Gamma R \Delta$, then $\Box \phi \in \Delta$. So, $\Gamma^\# \subseteq \Delta^\#$ and if $\Delta R E$, then $\Gamma R E$.

If $M$ has axiom 5 and $\Gamma R \Delta, E$, then $\Delta R E$. If $\phi \notin \Gamma^\#$, then $\Box \phi \notin \Gamma$, so $\neg \Box \phi \in \Gamma$, so by axiom 5, $\Box \neg \Box \phi \in \Gamma$, meaning that $\neg \Box \phi \in \Gamma^\#$.

Therefore, if $\Gamma R \Delta$, $\neg \Box \phi \in \Delta$, so $\Box \phi \notin \Delta$, meaning $\phi \notin \Delta^\#$. This concludes that $\Delta^\# \subseteq \Gamma^\#$, which means that if $\Gamma R E$, then $\Delta R E$.

Thus, if $\vdash_M \phi$, then $\phi$ is valid, while if $\not\vdash_M \phi$, then $\{\neg \phi\}$ is consistent, so it can be expanded to a maximally consistent set; therefore, $\neg \phi$ is satisfiable in the canonical model.

Tableaux Rules for Modal Logic

There are several proof systems for Modal Logic. We already presented Hilbert-style axiomatizations and proofs, while there are several others, including sequent calculi and tableaux. We consider prefixed tableaux to be
ideal candidates to use in order to produce decision procedures for Modal Logic (and Justification Logic as well). They are natural and directly correspond to frame conditions, while the upper bounds we extract are often evident from the tableau we give for a logic. In fact, we prove most upper bounds using some kind of a tableau. Thus, we present Tableau procedures for Modal Logic; the origin of the prefixed tableaux we present here can be found in [Fit72] and [Mas94]. For more on tableaux see [DGHP99].

A tableau operates on formulas that have been prefixed using two kinds of prefixes (also called a prefix and a sign). One is a truth prefix and the other is a state prefix. Specifically, such a prefixed formula is of the form $\sigma S \phi$, where $\phi$ a modal formula, $\sigma \in \mathbb{N}^*$ is a world- (state-)prefix, and $S \in \{ T, F \}$ is a truth-prefix. In the rules, $|$ denotes a nondeterministic choice. A tableau branch for $\phi$ is a set of formulas which can be generated by successive applications of the tableau rules for some set of non-deterministic choices. A branch is propositionally closed (or rejecting) if for some state-prefix $\sigma$ and formula $\psi$, both $\sigma T \psi$ and $\sigma F \psi$ are in the branch. A branch is accepting if it is complete (closed under the rules) and not rejecting. The tableau procedure accepts $\phi$ if there is an accepting branch for $\phi$.

A tableau procedure can be viewed either as a satisfiability-testing procedure – which is the way we view and use tableaux – or as a proof system –
by starting from $0 F \phi$ in this case and requiring that the procedure rejects
by our definitions. The latter is probably the usual way to view a tableau,
but for the logics we handle in this thesis, satisfiability and provability are
dual and therefore computationally equivalent: $\phi$ is provable iff $\neg \phi$ is not
satisfiable. A rule of the form

$$
\frac{A}{B} \\
\frac{}{C}
$$

describes the following action: from $A$, produce formulas $B, C$ and insert
both to the tableau branch. A rule of the form

$$
\frac{A}{B \mid C}
$$

describes the following action: from $A$, nondeterministically choose between
formulas $B$ and $C$ and produce one of the two and insert it to the branch
– equivalently, produce formulas $B, C$, and split the current branch to two
branches each containing one of the formulas.

Table 2.2 gives some of the tableau rules. These are the rules that apply
to propositional cases (to prefixed formulas for which the formula part is of
the form $\neg \phi$ or $\phi \circ \psi$, where $\circ$ is a propositional connective). Of course,
depending on whether we consider certain connectives as built from others,
we may not need all of these. Table 2.3 gives the remaining tableau rules,
depending on the axioms of each modal logic.
The rules from Table 2.2 give a sound and complete system for propositional logic, if we ignore the state-prefixes (or not, as they do not affect anything):

**Theorem 2.1.3.** The rules from Table 2.2 give a sound and complete proof system for propositional logic.

**Proof.** Let $b$ be an accepting branch for $\phi$. We define truth-valuation

$$V : \text{Prop} \rightarrow \{\text{true, false}\}$$

such that $V(p) = \text{true}$ if and only if $T \ p \in b$. Then, by induction on $\psi$ it is not hard to see that for every subformula $\psi$ of $\phi$, $\psi$ is true under $V$ if $T \ \psi \in b$ and $\psi$ is false under $V$ if $F \ \psi \in b$: if $\psi$ is a propositional variable, by definition of $V$; if not, it is clear from the tableau rules. We just look at
the case of $\psi = \psi_1 \rightarrow \psi_2$. If $T \psi \in b$, then by the rules, either $F \psi_1 \in b$, or $T \psi_2 \in b$ (depending on the nondeterministic choice that was made to construct $b$). Therefore, either $\psi_1$ is false or $\psi_2$ is true under $V$ (by inductive hypothesis), so $\psi$ must be true under $V$. On the other hand, if $F \psi \in b$, then by the rules, both $T \psi_1, F \psi_2 \in b$. Therefore, $\psi_1$ is true and $\psi_2$ is false under $V$ (by inductive hypothesis), so $\psi$ must be false under $V$.

On the other hand, if there is a satisfying truth-assignment $V$ for $\phi$, we can construct a complete branch $b$ for $\phi$, by making appropriate nondeterministic choices that ensure that for every subformula $\psi$ of $\phi$, if $T \psi \in b$, then $\phi$ is true under $V$ and if $F \psi \in b$, then $\phi$ is false under $V$. Therefore, $b$ cannot have both $T \psi$ and $F \psi$, as there is no truth-valuation $V$ that makes the same formula both true and false.

\[ \square \]

**Theorem 2.1.4.** The tableau systems as described by Tables 2.2 and 2.3 are sound and complete w.r.t. their corresponding modal logics.

**Proof.** Let $b$ be an accepting branch for $\phi$, given the tableau system for a certain fixed modal logic $ML$. We construct a model (for the fixed modal logic) $\mathcal{M} = (W, R, V)$ that satisfies $\phi$. For axiom $A$, we use $ML(A)$ to denote that the logic has $A$ as an axiom.

- $W$ is the set of all world-prefixes in $b$;
CHAPTER 2. DEFINITIONS

Axioms:

<table>
<thead>
<tr>
<th>Common Rules for all modal logics:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma T \diamond \phi$</td>
</tr>
<tr>
<td>$\sigma.n T \phi$</td>
</tr>
<tr>
<td>for some $\sigma.n$ that</td>
</tr>
<tr>
<td>has not already</td>
</tr>
<tr>
<td>appeared in the</td>
</tr>
<tr>
<td>branch.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding Tableau Rules:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma F \square \phi$</td>
</tr>
<tr>
<td>$\sigma.n F \phi$</td>
</tr>
<tr>
<td>for some $\sigma.n$ that</td>
</tr>
<tr>
<td>has not already</td>
</tr>
<tr>
<td>appeared in the</td>
</tr>
<tr>
<td>branch.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If the logic contains axiom $T$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma T \square \phi$</td>
</tr>
<tr>
<td>$\sigma T \phi$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If the logic contains $D$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma T \square \phi$</td>
</tr>
<tr>
<td>$\sigma T \diamond \phi$</td>
</tr>
<tr>
<td>for $\sigma.n$ that has already</td>
</tr>
<tr>
<td>occurred in the</td>
</tr>
<tr>
<td>branch.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding Tableau Rules:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma F \diamond \phi$</td>
</tr>
<tr>
<td>$\sigma F \phi$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If the logic contains $4$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma T \square \phi$</td>
</tr>
<tr>
<td>$\sigma.n T \phi$</td>
</tr>
<tr>
<td>for $\sigma.n$ that has already</td>
</tr>
<tr>
<td>occurred in the</td>
</tr>
<tr>
<td>branch.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding Tableau Rules:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma F \diamond \phi$</td>
</tr>
<tr>
<td>$\sigma.n F \phi$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If the logic contains $5$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma.n T \phi$</td>
</tr>
<tr>
<td>$\sigma T \square \phi$</td>
</tr>
<tr>
<td>for $\sigma$ and $\sigma.n$</td>
</tr>
<tr>
<td>that have already</td>
</tr>
<tr>
<td>occurred in the</td>
</tr>
<tr>
<td>tableau branch.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding Tableau Rules:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma.n F \phi$</td>
</tr>
<tr>
<td>$\sigma F \diamond \phi$</td>
</tr>
<tr>
<td>for $\sigma$ and $\sigma.n$</td>
</tr>
<tr>
<td>that have already</td>
</tr>
<tr>
<td>occurred in the</td>
</tr>
<tr>
<td>tableau branch.</td>
</tr>
</tbody>
</table>

Table 2.3: Modal Tableau rules
• $R$ is the closure under the closure conditions of the accessibility relations in the frames that are associated to our fixed logic (if $R$ must be serial, then for every $a$ for which there is no $aRb$, we can introduce $(a, a)$ into $R$) of $R_0 = \{ (\sigma, \sigma.n) \in W \times W \}$;

• for every propositional variable $p$, $V(p) = \{ a \in W \mid a \vdash p \in b \}$.

Then, by induction on $\psi$ we can see that for every subformula $\psi$ of $\phi$ and state $\sigma, M, \sigma \models \psi$ if $\sigma \vdash \psi \in b$ and $M, \sigma \not\models \psi$ if $\sigma \models \psi \in b$: if $\psi$ is a propositional variable, by definition of $V$; propositional cases are similar to the proof of 2.1.3. For the cases of $\sigma \vdash \chi$ and $\sigma \models \chi$, it suffices to verify that there is some $\sigma' \in W$, such that $\sigma R \sigma'$ and $\sigma' \vdash \chi \in b$ or $\sigma' \models \chi \in b$, respectively. But this is easy to see by the corresponding rules and for $\sigma' = \sigma.n$. Finally, the remaining cases are $\sigma \vdash \chi$ and $\sigma \models \chi$, which are equivalent – so we only examine the first one. It suffices to prove that if $\sigma \vdash \chi \in b$ and $\sigma R \sigma'$, then $\sigma' \vdash \chi \in b$.

Let $R_0 \subseteq R_1 \subseteq R_2 \subseteq \cdots$ be defined in the following way: as we defined above, $R_0 = \{ (\sigma, \sigma.n) \in W \times W \}$; for $i > 0$,

$$R_i = R_{i-1} \cup \{ (a, a) \in W \times W \mid ML(T) \} \cup$$

$$\cup \{ (a, a) \in W \times W \mid ML(D), \overline{a}(a, b) \in R_{i-1} \}$$

$$\cup \{ (a, b) \in W \times W \mid ML(4), \exists(a, c), (c, b) \in R_{i-1} \}$$
\[ \bigcup \{(a, b) \in W \times W \mid ML(5), \exists (c, a), (c, b) \in R_{i-1}\}. \]

Then, \( \bigcup_{i \in \mathbb{N}} R_i = R \). We actually prove by induction on \( i \) that for every \( i \in \mathbb{N} \),

1. if \( \sigma T \square \chi \in b \) and \( \sigma R_i \sigma' \), then \( \sigma' T \chi \in b \);

2. if \( ML(4), \sigma T \square \chi \in b \) and \( \sigma R_i \sigma' \), then \( \sigma' T \square \chi \in b \);

3. if \( ML(5), \sigma T \square \chi \in b \) and \( \sigma' R_i \sigma \), then \( \sigma' T \square \chi \in b \).

For \( i = 0, 1, 2, \) and \( 3 \) are easy to verify by the tableau rules. For \( i > 0 \), we take cases, depending on how a new pair of \( R_i \) was introduced and whether we prove 1, 2, or 3 for \( i \):

- if \( \sigma = \sigma' \) and \( ML(T) \), then 1 is easy to confirm by the corresponding tableau rule;

- if \( \sigma = \sigma' \), \( ML(D) \), and there is no \( (\sigma, \sigma'') \in R_{i-1} \), then by the corresponding tableau rule, \( \sigma T \Box \chi \notin b \);

- if \( ML(4) \) and \( \exists (\sigma, c), (c, \sigma') \in R_{i-1} \), then we get 1 by the inductive hypothesis for 2;

- if \( ML(5) \) and \( \exists (a, \sigma), (a, \sigma') \in R_{i-1} \), then we get 1 by the inductive hypothesis for 3;
• if $\sigma = \sigma'$, then we immediately have both 2 and 3;

• if $ML(4)$ and $\exists(\sigma, c), (c, \sigma') \in R_{i-1}$, then if $\sigma T \Box \chi \in b$, then $c T \Box \chi \in b$
(by I.H., 2), so $\sigma' T \Box \chi \in b$ (again, by I.H., 2), while if $ML(5)$, then
by I.H., 2, if $\sigma' T \Box \chi \in b$, then $c T \Box \chi \in b$, and $\sigma T \Box \chi \in b$;

• if $ML(5), ML(4)$ and $\exists(a, \sigma), (a, \sigma') \in R_{i-1}$, then if $\sigma T \Box \chi \in b$, then
$a T \Box \chi \in b$ (by I.H., 3), so $\sigma' T \Box \chi \in b$ (by I.H., 2), while by I.H., 3,
and then 2, if $\sigma' T \Box \chi \in b$, then $a T \Box \chi \in b$, and $\sigma T \Box \chi \in b$.

On the other hand, if there is a model and a state $\mathcal{M}, a \models \phi$, then we can
construct a complete branch $b$ for $\phi$, by making appropriate nondeterministic
choices that preserve the mapping condition: we map each state-prefix $\sigma$ to
a state $b$, such that 0 is mapped to $a$ and for every subformula $\psi$ of $\phi$,
if $\sigma T \psi \in b$, then $\mathcal{M}, b \models \psi$ and if $\sigma F \psi \in b$, then $\mathcal{M}, b \not\models \psi$, for
some $b$ that is mapped to $\sigma$. Therefore, $b$ cannot have both $\sigma T \psi$ and
$\sigma F \psi$. The part of the nondeterministic choices is similar to the proof of
the propositional version of this theorem. To preserve a mapping as new
state-prefixes are introduced from rule $\frac{\sigma T \psi}{\sigma n T \psi}$, notice that since $\sigma$ is mapped
to some $b$, $b \models \Diamond \psi$, so there is some $c$ accessible from $b$, such that $c \models \psi$.
We map $\sigma n$ to $c$. Now it just remains to check that the remaining tableau
rules give formulas that preserve the mapping condition. □
CHAPTER 2. DEFINITIONS

Notice in the proof above that we needed to assume that if $ML(5)$, then it must also be the case that $ML(4)$. Indeed, the subset of modal logics we presented is not as extensive as it could be: we do not take all combinations of the axioms, but instead we assume that if the logic has 5, then it also has 4.\(^3\) This choice has been made mostly for clarity; the tableau rules of Table 2.3 are not completely modular, so they do not work, for example, for logic $K5$ (the reader can confirm that $\Diamond \Box p \rightarrow \Box \Diamond p$ is derivable in $K5$, but not in the corresponding tableau). There are tableau rules to address this, but we do not use them. When we deal with modal satisfiability for logics with negative introspection, we use different methods.

A note on tableau termination and the order of application for the rules. We have defined a complete branch as a branch closed under the tableau rules. This definition is general enough to include infinite branches, which definitely exist – consider running the $D4$-tableau on $\Box p$. However, if we want to use the tableau as an actual procedure to decide satisfiability, we need it to actually terminate at some point. For the $K$-tableau (or for any other logic without axiom 4), we can see that the maximum modal depth of the formulas prefixed by $\sigma$ is greater than the maximum modal depth

\(^3\)We also assume that $ML(D)$ and $ML(T)$ are mutually exclusive, but this makes sense, since axiom $D$ is a special case of $T$. 

of the formulas prefixed by \( \sigma.n \), therefore it always terminates. When the logic has axiom 4, then we can see that boxed formulas (of the form \( \Box \phi \) or \( \Diamond \phi \)) “accumulate” as the prefixes grow. For other cases (including ones that appear in following chapters), we will argue that we do not need to examine all the prefixes, but we can cut off the procedure after a certain length of prefixes has been reached. Furthermore, we often need to run the tableau using a restricted amount of space, thus keeping in memory a certain number of prefixes at a time. In that case it makes sense to give lower priority to rules that produce new prefixes and highest to the ones that move formulas from one prefix to the other (for example the box rules from Table 2.3); this way we keep all necessary information for a specific prefix at a time, even if we delete certain prefixes.

2.2 Justification Logic

In this section we present the (single-agent) justification logics \( J_{CS}, JD_{CS}, J4_{CS}, JD4_{CS}, JT_{CS}, LP_{CS} \) – \( CS \) is a parameter of each logic called a constant specification, to be explained later. Other justification logics have also been defined, but we focus on these because they are the single-agent logics with known complexity and they are the ones we need to present the full multi-agent systems in the following chapters.
CHAPTER 2. DEFINITIONS

2.2.1 Syntax

The syntax of Justification Logic is similar to the one of Modal Logic, only instead of using □, we use justification terms. These are constructed from justification variables \(x_1, x_2, \ldots\), justification constants \(c_1, c_2, \ldots\), and justification operators !, ·, +: justification constants and variables are justification terms and if \(t, s\) are terms, then so are \([t + s], [t \cdot s], !t\). In short,

\[
t ::= x \mid c \mid [t + t] \mid [t \cdot t] \mid !t.
\]

The set of justification terms is called \(Tm\) and in fact is the set of terms we will use for all the justification logics we present in this thesis. The common language \(L_J\) of the justification logics \(J, JD, J4, JD4, JT, LP\) has propositional variables \(p_1, p_2, \ldots\) like \(L_M\) and formulas are defined:

\[
\phi ::= \bot \mid p \mid \neg \phi \mid (\phi \rightarrow \phi) \mid (\phi \land \phi) \mid (\phi \lor \phi) \mid t : \phi,
\]

but depending on convenience we will treat some connectives as constructed from others, while we often omit parentheses. If we omit parentheses, : and \(\neg\) bind more strongly than \(\land\), which binds more strongly than \(\lor\), which in turn binds more strongly than \(\rightarrow\).

Intuitively, · applies a justification for a statement \(\phi \rightarrow \psi\) to a justification for \(\phi\) and gives a justification for \(\psi\). Using + we can combine two justifications.
and have a justification for anything that can be justified by any of the two initial terms – much like the concatenation of two proofs. Finally, ! is a unary operator called the proof checker. Given a justification \( t \) for \( \phi \), \(!t\) justifies the fact that \( t \) is a justification for \( \phi \).

### 2.2.2 Axiomatizations

Let’s assume a complete axiomatization of classical propositional logic which uses finitely many axiomatic schemes. We expand on these axioms. All justification logics have Modus Ponens and they come with the following axioms:

- Finitely many axiomatic schemes of classical propositional logic;
- Application Axiom: \( s : (\phi \to \psi) \to (t : \phi \to [s \cdot t] : \psi) \);
- Concatenation Axiom: \( s : \phi \to [s + t] : \phi \) and \( s : \phi \to [t + s] : \phi \),

where in the above, \( \phi, \psi \) are formulas in \( L_J \), \( t, s \) justification terms.

We also include a set of axioms from the following ones; which particular combination we use depends upon the logic. Specifically, if the logic is \( JT_{CS} \) or \( LP_{CS} \), we include Factivity; if the logic is \( J4_{CS} \) or \( LP_{CS} \), we include Positive Introspection; if the logic is \( JD_{CS} \) or \( JD4_{CS} \), we include the Consistency axiom.

- Factivity Axiom: \( t : \phi \to \phi \);
CHAPTER 2. DEFINITIONS

Positive introspection: $t : \phi \to !t : \phi$;

Consistency Axiom: $t : \bot \to \bot$ (or just $\neg t : \bot$),

where in the above, $\phi$ is a formula in $L_J$, $t$ a justification term.

To complete the description of a justification logic, a *constant specification* $\mathcal{CS}$ is needed: a constant specification for a justification logic is a set of formulas of the form $c : A$, where $c$ a justification constant and $A$ an axiom of the logic from the ones above. We say that axiom $A$ is justified by a constant $c$ when $c : A \in \mathcal{CS}$. Then we can introduce our final axiom,

Axiom Necessitation: $t : \phi$, where either $t : \phi \in \mathcal{CS}$ or $\phi = s : \psi$ an instance of Axiom Necessitation and $t =!s$.

Axiom Necessitation will be called AN for short. Therefore, $J_{\mathcal{CS}}$ is the version of logic $J$ equipped with constant specification $\mathcal{CS}$.

A constant specification is:

- axiomatically appropriate if each axiom is justified by at least one constant;

- schematic if every constant justifies a certain number of the axiom schemes of the logic except for AN (0 or more) – in this case, a constant
may justify only a finite number of schemes, but either 0 or infinite 

axioms;

• schematically injective if it is schematic and every constant justifies at 

most one scheme;

• finite if it is a finite set.

\( J_\emptyset, JD_\emptyset, J4_\emptyset, JD4_\emptyset, JT_\emptyset, LP_\emptyset \) are the respective justification logics under 

the empty constant specification, \( \emptyset \). \( J, JD, J4, JD4, JT \) and \( LP \) are \( JTCS_J, JD_{\text{TCS}_J}, J4_{\text{TCS}_J}, JD4_{\text{TCS}_J}, JT_{TCS_J} \) and \( LP_{\text{TCS}_J} \) respectively, where for 

some justification logic \( J \), \( TCS_J \) is the total constant specification,

\[
TCS_J = \{ c:A \mid c \text{ is a constant, } A \text{ an axiom of } J \text{ except for AN} \}.
\]

The total constant specification is schematic and axiomatically appropriate, 

but not schematically injective.

Proposition 2.2.1 is a very characteristic property of Justification Logic. 

It is the counterpart of the Necessitation Rule from Modal Logic, only for 

Justification Logic it is a derived property and not a rule. It shows that 

Justification Logic can internalize the proofs of its own theorems – as long 

as we have an axiomatically appropriate constant specification.
Proposition 2.2.1. For an axiomatically appropriate constant specification, if \( \phi_1, \ldots, \phi_k \vdash \phi \), then for terms \( t_1, \ldots, t_k \), there is some term \( t \) such that \( t_1 : \phi_1, \ldots, t_k : \phi_k \vdash t : \phi \).

Proof. By induction on the proof of \( \phi \): If \( \phi \) is an axiom, then by AN, the theorem holds and it obviously holds for any \( \phi_i \). This covers the base cases. Using the application axiom, if \( \phi \) is the result of \( \psi, \chi \) and modus ponens, since the theorem holds for some \( r : (\psi \rightarrow \phi) \) and \( s : \psi \), the theorem holds for \( \phi \) and \( t = [r \cdot s] \).

\[ \Box \]

2.2.3 Semantics

We present Fitting-models (often called F-models), which were first introduced by Fitting in [Fit05] and later for more logics in [Pac05, Kuz08b] and Mkrtchyan-models (often called M-models), introduced by Mkrtchyan in [Mkr97], but later given for more logics in [Kuz00]. Traditionally ([Kuz00, Kuz09, Kuz08a]), Mkrtchyan models were used to prove complexity upper bounds for Justification Logic, but in more recent work ([Ach14b, Ach14c, Ach15b]), there was a need to work with Fitting models – especially when the Consistency axiom is present.

Fitting models are essentially Kripke-models, but with an additional mechanism to handle justification terms, which we call an admissible evi-
dence function. On the other hand, Mkrtchyan models can mostly (but not entirely) be thought of as Fitting models but of only one state. Thus we can drop the multiple-world part of the semantics and just keep an admissible evidence function and a truth valuation function – which we do. We start by giving the definition for Fitting models, then for Mkrtchyan models, and then we prove a soundness and completeness theorem for both.

**Definition 2.2.1.** Let \( J \) be one of the logics \( J, JD, J4, JD4, JT, \) or \( LP \). A Fitting model \( M \) for logic \( J_{CS} \) is a quadruple \((W, R, E, V)\), where \( W \neq \emptyset \) is the set of worlds (or states) of the model, \( R \) is a binary relation on \( W \), \( V \) assigns a subset of \( W \) to each propositional variable, and \( E \) assigns a subset of \( W \) to each pair of a justification term and a formula:

\[
V : Prop \rightarrow 2^W
\]

and

\[
E : Tm \times L_J \rightarrow 2^W.
\]

\( E \) is called an admissible evidence function and must satisfy the following closure conditions:

**Application closure:** for any formulas \( \phi, \psi \) and justification terms \( t, s \),

\[
E(s, \phi \rightarrow \psi) \cap E(t, \phi) \subseteq E(s \cdot t, \psi);
\]
CHAPTER 2. DEFINITIONS

Sum closure: for any formula $\phi$ and justification terms $t, s$,

$$\mathcal{E}(t, \phi) \cup \mathcal{E}(s, \phi) \subseteq \mathcal{E}(t + s, \phi);$$

CS-closure: for any instance $t : \phi$ of $AN$,

$$\mathcal{E}(t, \phi) = W.$$

The following closure conditions are in effect only when $J$ has Positive Introspection (i.e. $J$ is $J4$, $JD4$, or $LP$):

Positive Introspection closure: for any formula $\phi$ and justification term $t$,

$$\mathcal{E}(t, \phi) \subseteq \mathcal{E}(!t, t: \phi);$$

Distribution: for any formula $\phi$, justification term $t$ and $a, b \in W$, if $aRb$

and $a \in \mathcal{E}(t, \phi)$, then $b \in \mathcal{E}(t, \phi);$ 

Just like in the definition for Kripke models for Modal Logic, the accessibility relation $R$ must satisfy the following conditions:

- If $J = JT$ or $J = LP$, then $R$ must be reflexive.

- If $J = JD$ or $J = JD4$, then $R$ must be serial.

- If $J = J4$, $J = JD4$, or $J = LP$, then $R$ must be transitive.
Truth in the model – as described by relation \( \models \) – is defined in the following way, given a state \( a \):

- \( M, a \not\models \bot \).

- If \( p \) is a propositional variable, then \( M, a \models p \) iff \( a \in \mathcal{V}(p) \).

- If \( \phi, \psi \) are formulas, then \( M, a \models \phi \rightarrow \psi \) if and only if \( M, a \models \psi \), or \( M, a \not\models \phi \).

- If \( \phi \) is a formula and \( t \) a term, then \( M, a \models t : \phi \) if and only if \( a \in \mathcal{E}(t, \phi) \) and for all \( b \in W \), if \( aRb \), then \( M, b \models \phi \).

As we mention above, Mkrtchyan models can mostly be considered Fitting models of only one world; this view works out, if one carefully compares the two definitions. The only exception comes up in the presence of the Consistency axiom – when the logic in question is either JD or JD4. In that case, the seriality condition for Fitting models enforces the existence of more than one world. Thus it is replaced by the consistent evidence condition, which turns out to be more general, in that it gives completeness for more logics, but seems rather arbitrary – especially when dealing with complexity issues.

**Definition 2.2.2.** Let \( J \) be one of the logics J, JD, J4, JD4, JT, or LP. A
Mkrtychev model for $J_{CS}$, where $CS$ is a constant specification for $J$, is a pair $\mathcal{M} = (A, \mathcal{V})$, where propositional valuation

$$\mathcal{V} : \text{Prop} \rightarrow \{\text{true}, \text{false}\}$$

assigns a truth value to each propositional variable and

$$\mathcal{E} : \text{Tm} \times L_J \rightarrow \{\text{true}, \text{false}\}.$$  

Many times $\mathcal{E}(t, \phi)$ will be used as an abbreviation for $\mathcal{E}(t, \phi) = \text{true}$ and $\neg \mathcal{E}(t, \phi)$ as an abbreviation for $\mathcal{E}(t, \phi) = \text{false}$.

Like in the definition of Fitting models, the admissible evidence function must satisfy certain conditions that depend on the axioms and rules of $J_{CS}$:

**Application Closure:** if $\mathcal{E}(s, \phi \rightarrow \psi)$ and $\mathcal{E}(t, \phi)$, then $\mathcal{E}(s \cdot t, \psi)$;

**Sum Closure:** if $\mathcal{E}(s, \phi)$ or $\mathcal{E}(t, \phi)$, then $\mathcal{E}(s + t, \phi)$;

**CS Closure:** if $t : \phi$ is an instance of Axiom Necessitation, then $\mathcal{E}(t, \phi)$;

**Positive Introspection Closure:** if $\mathcal{E}(t, \phi)$ then $\mathcal{E}(!t, t : \phi)$ — when $J$ has Positive Introspection as an axiom;

**Consistent Evidence Condition:** $\mathcal{E}(t, \bot) = \text{false}$ — when $J$ has Consistency as an axiom.

The truth relation $\mathcal{M} \models \phi$ is defined as follows:
• $M \not\models \bot$;

• $M \models p$ iff $\forall (p) = \text{true}$ for propositional variable $p$;

• $M \models \phi \rightarrow \psi$ iff $M \not\models \phi$ or $M \models \psi$;

• $M \models t : \phi$ iff $M \models \phi$ and $E(t : \phi) = \text{true}$ — if factivity is an axiom of $J$;

• $M \models t : \phi$ iff $E(t : \phi) = \text{true}$ — if factivity is not an axiom of $J$.

So, the accessibility relation is not considered in the case of the Mkrtychev models. The way factivity is ensured is through the definition of truth in Mkrtychev models, which varies from logic to logic. We say that a Mkrtychev model (respectively Fitting model) $M$ with admissible evidence function $E$ has the Strong Evidence Property when for every $t : \phi \in L_J$ (resp. and state $a$), $M \models \phi$ if and only if $E(t, \phi) = \text{true}$ (resp. $M, a \models \phi$ if and only if $a \in E(t, \phi)$).

**Theorem 2.2.2** (Completeness Theorem for Mkrtychev models ). Each justification logic from $J_{CS}$, $JD_{CS}$, $JT_{CS}$, $J4_{CS}$, $JD4_{CS}$, $LP_{CS}$, where $CS$ is a constant specification for that logic is sound and complete with respect to its Mkrtychev models; each such logic is also sound and complete with respect to its Mkrtychev models that have the Strong Evidence Property.
Proof. To prove soundness, we just need to prove that every Mkrtchyan model satisfies all axioms, since it is not hard to see that Modus Ponens preserves truth. We do not deal with the axioms of Propositional Logic.

Application: if $M \models s : (\phi \rightarrow \psi)$ and $M \models t : \phi$, then $E(s, \phi \rightarrow \psi) = E(t, \phi) = true$, which in turn gives $E(s \cdot t, \psi) = true$, by the application closure; if $J = JT$ or $LP$, then also $M \models \phi \rightarrow \psi, \phi$, so $M \models \psi$; in any case (whether $J \in \{JT, LP\}$ or not), then, we can conclude that $M \models [s \cdot t] : \psi$.

Concatenation: by the sum closure condition, whether $J$ has factivity or not.

Factivity: trivial, by the definition of $\models$.

Consistency: trivial, by the Consistent Evidence condition.

Positive Introspection: if $M \models t : \phi$, then $E(t, \phi) = true$, so by the

Positive Introspection closure, $E(!t, t : \phi) = true$ and thus $M \models !t : t : \phi$.

To prove completeness, let $\not\vdash_J \phi$. Then, $\neg \phi$ is consistent. Let $\Gamma$ be a maximally consistent set of formulas such that $\neg \phi \in \Gamma$. Notice that if $\Gamma \vdash \psi$, then $\Gamma \cup \{\psi\}$ must be consistent (since $\Gamma$ is consistent) and since $\Gamma$ is maximally consistent, $\psi \in \Gamma$. Let $M = (E, V)$ be such that $E(t, \psi) = true$.
iff \( t : \psi \in \Gamma \) and \( \forall p(t) = true \) iff \( p \in \Gamma \). It is not hard to prove that \( \mathcal{E} \) satisfies all the conditions that are required from an admissible evidence function:

**Application Closure:** If \( \mathcal{E}(s, \phi \rightarrow \psi) \) and \( \mathcal{E}(t, \phi) \), then \( s : (\phi \rightarrow \psi), t : \phi \in \Gamma \), therefore \( \Gamma \vdash [s \cdot t] : \psi \), so \( [s \cdot t] : \psi \in \Gamma \) and thus \( \mathcal{E}(s \cdot t, \psi) \);

**Sum Closure:** If \( \mathcal{E}(s, \phi) \) or \( \mathcal{E}(t, \phi) \), then \( s : \phi \) or \( t : \phi \in \Gamma \), so \( \Gamma \vdash [s + t] : \phi \) and therefore \( \mathcal{E}(s + t, \phi) \);

**CS Closure:** If \( t : \phi \) is an instance of Axiom Necessitation, \( \Gamma \vdash t : \phi \), so \( t : \phi \in \Gamma \) and thus \( \mathcal{E}(t, \phi) = true \);

**Positive Introspection Closure:** If \( \mathcal{E}(t, \phi) \) then \( t : \phi \in \Gamma \), because of Positive Introspection \( \Gamma \vdash !s : s : \phi \) and therefore \( \mathcal{E}(!t, t : \phi) \);

**Consistent Evidence Condition:** because of Consistency, if \( t : \bot \in \Gamma \), then \( \Gamma \vdash \bot \), but \( \Gamma \) is consistent, so \( \mathcal{E}(t, \bot) = false \).

Therefore, \( \mathcal{M} \) is indeed a Mkrtchyan model for \( J \). It remains to prove that \( \mathcal{M} \models \neg \phi \). For this we prove the following lemma:

**Lemma 2.2.3** (Truth Lemma). For every \( \psi \in L_J \), \( \mathcal{M} \models \psi \) if and only if \( \psi \in \Gamma \).

*Proof.* We prove this by induction on \( \psi \). Propositional cases are easy. If
\( \psi = t : \chi \), then \( \psi \in \Gamma \) if and only if \( E(t, \chi) = \text{true} \); furthermore, if \( J \) has Factivity and \( \psi \in \Gamma \), then \( \chi \in \Gamma \), so \( \mathcal{M} \models \chi \). Therefore, \( \mathcal{M} \models \psi \). \( \square \)

Therefore, since \( \neg \psi \in \Gamma \), it is also the case that \( \mathcal{M} \models \neg \psi \). For the last part of the theorem, notice that \( \mathcal{M} \) has the Strong Evidence Property by definition: using the Truth Lemma,

\[
E(t, \psi) = \text{true} \iff t : \psi \in \Gamma \iff \Gamma \models s : \psi.
\]

\( \square \)

Now we prove the same theorem for Fitting models. The reader may notice that completeness for Fitting models is less general than completeness for Mkrtchyan models: we need to assume an axiomatically appropriate constant specification when the logic in question has the Consistency axiom (i.e. it is either \( JD \) or \( JD4 \)). However, we consider that the assumption that a constant specification is axiomatically appropriate is a very natural one, which also naturally corresponds to the Necessitation rule from Modal Logic.

**Proposition 2.2.4.** Let \( J \) be one of \( J, JT, J4, LP \). \( J_{CS} \) is sound and complete with respect to its Fitting models. Furthermore, \( J_{CS} \) is sound and complete with respect to its Fitting models that have the Strong Evidence Property. If \( J \) is either \( JD \) or \( JD4 \), then \( J_{CS} \) is sound and complete with respect to its
Fitting models as long as $CS$ is axiomatically appropriate. In that case, $J_{CS}$ is also sound and complete with respect to its Fitting models that have the Strong Evidence Property.

Proof. Soundness is very similar to the case of Mkrtchyan models from Theorem 2.2.2. Completeness will be proven using a canonical model construction. Let $W$ be the set of all maximal consistent subsets of $L_J$. We know that $W$ is not empty, because $J$ is consistent. For every $\Gamma \in W$, let $\Gamma^\# = \{\phi \in L_J \mid \exists t \in \text{TM} \ t : \phi \in \Gamma\}$. $R$ is a binary relation on $W$, such that $\Gamma R \Delta$ if and only if $\Gamma^\# \subseteq \Delta$. Let $E(t, \phi) = \{\Gamma \in W \mid t : \phi \in \Gamma\}$. Finally, $V : \text{Prop} \rightarrow 2^W$ is such that $V(p) = \{\Gamma \in W \mid p \in \Gamma\}$. The canonical model is $\mathcal{M} = (W, R, E, V)$.

Define the relation between worlds of the canonical models and formulas of $L_J$, $\models$, as in the definition of models.

Lemma 2.2.5 (Truth Lemma). For all $\Gamma \in W$, $\phi \in L_J$, $\mathcal{M}, \Gamma \models \phi$ if and only if $\phi \in \Gamma$.

Proof. By induction on the structure of $\phi$. The cases for $\phi = p$, a propositional variable, $\bot$, or $\psi_1 \rightarrow \psi_2$, are immediate from the definition of $V$ and $\models$. We examine the case when $\phi = t : \psi$. If $\mathcal{M}, \Gamma \models t : \psi$, then $\Gamma \in E(t, \psi)$ and therefore $t : \psi \in \Gamma$. For the other direction, if $t : \psi \in \Gamma$, then $\Gamma \in E(t, \psi)$.
and for every $\Delta \in W$ such that $\Gamma R \Delta$, $\psi \in \Delta$, so $\Delta \models \psi$ – which means that $\mathcal{M}, \Gamma \models \psi$ and completes the proof.

The canonical model is, indeed, a model for $J$. To establish this, we must show that the conditions expected from $R$ and $E$ are satisfied. First, the admissible evidence function closure conditions:

**Application closure:** If $\Gamma \in E(s, \phi \rightarrow \psi) \cap E(t, \phi)$, then $s : (\phi \rightarrow \psi), t : \phi \in \Gamma$. Because of the application axiom, $[s \cdot t] : \psi \in \Gamma$, so $\Gamma \in E_i(s \cdot t, \psi)$.

**Sum closure:** If $\Gamma \in E(t, \phi)$, then $t : \phi \in \Gamma$, so, by the Concatenation axiom, $[s + t] : \phi, [t + s] : \phi \in \Gamma$, therefore, $\Gamma \in E(t + s, \phi) \cap E(s + t, \phi)$.

**Positive Introspection closure:** If $J$ has Positive Introspection and $\Gamma \in E(t, \phi)$, then $t : \phi \in \Gamma$ and because of Positive Introspection, $!t : t : \phi \in \Gamma$, therefore, $\Gamma \in E(!t, t : \phi)$.

**CS closure:** Any $\Gamma \in W$ includes all instances of AN, so this is satisfied.

**Distribution:** If $J$ has Positive Introspection, $\Gamma R \Delta$, and $\Gamma \in E(t, \phi)$, then $t : \phi \in \Gamma$ and by Positive Introspection, $!t : t : \phi \in \Gamma$, thus $t : \phi \in \Gamma \# \subseteq \Delta$, concluding that $\Delta \in E(t, \phi)$.

To complete the proof, we prove that $R$ satisfies the required conditions:
If $F(i) = JT$ or $LP$, then $R$ is reflexive. For this, we just need that if

$\Gamma \in W$, then $\Gamma^# \subseteq \Gamma$. If $\phi \in \Gamma^#$, then there is some justification term, $t$, for which $t : \phi \in \Gamma$. Because of the Factivity axiom, $\neg \phi \notin \Gamma$, since

$\{t : \phi, \neg \phi\}$ is inconsistent. Therefore, as $\Gamma$ is maximal consistent, $\phi \in \Gamma$.

If $J = JD$ or $JD4$, then $R$ is serial. To establish this, we just need to show that $\Gamma^#$ is consistent. If it is not, then there are formulas $\phi_1, \ldots, \phi_k \in \Gamma^#$ s.t. $\phi_1, \ldots, \phi_k \vdash \bot$. This means that there are $t_1 : \phi_1, \ldots t_k : \phi_k \in \Gamma$, s.t. $t_1 : \phi_1, \ldots t_k : \phi_k \vdash t : \bot$ (by Proposition 2.2.1, which requires an axiomatically appropriate constant specification), which is a contradiction.

If $J$ has Positive Introspection and $\Gamma R \Delta RE$, then $\Gamma RE$. If $t : \phi \in \Gamma$, then !$t : t : \phi \in \Gamma$. Therefore, $t : \phi \in \Gamma^#$. So, if $\Gamma R \Delta$, then $t : \phi \in \Delta$. So, $\Gamma^# \subseteq \Delta^#$ and if $\Delta RE$, then $\Gamma RE$.

Finally, notice that the canonical model has the Strong Evidence Property: if $\Gamma \in E(t, \phi)$ then $t : \phi \in \Gamma$ and by the Truth Lemma, $\Gamma \models t : \phi$.  

The Strong Evidence Property is a very useful property; it is used in tableaux to remove nondeterminism and in small-model results. It is not hard from a model $\mathcal{M} = (W, R, E, V)$ to construct a model $\mathcal{M}' = (W, R, E', V)$
which has the Strong Evidence Property and for every \( w \in W \) and \( \phi \in L_J \),
\[ M, w \models \phi \text{ iff } M', w \models \phi \; : \text{ just define} \]
\[ \mathcal{E}'(t, \phi) = \{ w \in W \mid M, w \models t : \phi \}. \]

### 2.2.4 Explicit Modal Logic

As the reader may have noticed from the axioms of Justification Logic and the naming of each logic, there is a correspondence between Justification Logic and Modal Logic. For each justification logic there is a corresponding modal logic. For instance, \( J \) corresponds to \( K \), \( JD \) to \( D \), \( JT \) to \( T \), \( J4 \) to \( K4 \), \( JD4 \) to \( D4 \), and \( LP \) to \( S4 \). The Realization Theorem makes this correspondence formal, further justifying Justification Logic’s alternative characterization as Explicit Modal Logic.

The Realization Theorem says that there is a straightforward translation from the language of Justification Logic to the language of Modal Logic, called forgetful projection. This translation simply turns justification terms into boxes and translates theorems of a justification logic into theorems of the corresponding modal logic. However, the most important property of this translation that the Realization Theorem establishes is that for each theorem \( \phi \) of this corresponding modal logic, there is a theorem (perhaps more than one) of the justification logic, such that \( \phi \) is its translation.
Definition 2.2.3. The forgetful projection is a function $^\circ : L_J \rightarrow L_M$ that converts justification formulas into modal formulas. It is defined recursively:

\[ p^\circ = p, \ \bot^\circ = \bot, \ (\phi \rightarrow \psi)^\circ = (\phi^\circ \rightarrow \psi^\circ), \text{ and } (t; \phi)^\circ = \Box(\phi^\circ), \]

where $p$ is a propositional variable, $\phi, \psi$ are justification formulas and $t$ is a justification term.

The forgetful projection can be generalized on sets of formulas: if $A \subseteq L_J$, then $A^\circ = \{ \phi^\circ \in J_M \mid \phi \in A \}$. By identifying a logic with the set of its theorems, we can apply the forgetful projection on a logic and get the set of the projections of the logic’s theorems.

Theorem 2.2.6 (Realization Theorem [Art95, Bre00, Art08]). $J^\circ = K$; $JD^\circ = D$; $JT^\circ = T$; $J4^\circ = K4$; $JD4^\circ = D4$; $LP^\circ = S4$.

In other words, the set of forgetful projections of the theorems of each justification logic is exactly the set of the theorems of the corresponding modal logic. It is not hard to see that the projection operator respects Modus Ponens and that each axiom of a justification logic $J$ is projected to a tautology of the corresponding modal logic $M$. Therefore, immediately $J^\circ \subseteq M$. The other direction is the harder one to prove and requires a realization procedure, which maps each modal box of a modal theorem to a

\footnote{See also [Fit03, Fit05, Fit06, Fit07b, Fit07a].}
certain justification term. The topic of realization procedures and theorems is an extensive and important one. We do not cover it as it is beyond the scope of this thesis. As such, in the following chapters we informally call a modal logic the corresponding modal logic of a justification logic (and vice-versa) if their axioms match. The reader can read [Fit15] for a recent investigation in the matter, and [Yav08] for realization results for Multi-Agent Justification Logic.

2.2.5 Two versions of the $*$-calculus and admissible evidence

We present the $*$-calculus for Justification Logic. The $*$-calculus is an invaluable tool in the study of the complexity of Justification Logic. We present two variations of the calculus, depending on the semantics we are using. Each of those operates on $*$-expressions, which are expressions of the form $*(t, \phi)$, where $t : \phi \in L_J$. The difference is that while one of the two types, which we simply call a $*$-calculus, operates on $*$-expressions as they are, the other type, which we call a $*$-calculus on frames (or just $*$-calculus if it is clear from the context which one we are using), has several instances, each based on a fixed frame and operates on $*$-expressions prefixed by states from that frame. In fact, we first present the $*$-calculus on frames and explain how we can retrieve the plain $*$-calculus by simply ignoring all parts that refer to a frame.
A $*$-calculus was first introduced in [Kru06], but its origins can be found in [Mkr97]. The first time it explicitly appeared as fully frame-based was in [Ach14c].\footnote{[Kuz08a] shows a way to use the calculus with a frame in a meaningful way without it being based on it.} The form on which the ones in this section are based is mainly from [Kuz08a]; this subsection’s results are from [Mkr97, Kru06, Kuz08a].

If $t$ is a term and $\phi$ is a formula then $*(t, \phi)$ is a $*$-expression. Given a frame $\mathcal{F} = (W, R)$ for $J$, the $*F$-calculus for $J$ on the frame $\mathcal{F}$ is a calculus on $*$-expressions prefixed by worlds from $W$ ($*F$-expressions from now on) with the axioms and rules that are shown in Table 2.4.

If $\mathcal{E}$ is an admissible evidence function of $\mathcal{M}$, we define $\mathcal{M}, w \models *(t, \phi)$ iff $\mathcal{E} \models w * (t, \phi)$ iff $w \in \mathcal{E}(t, \phi)$. Notice that the calculus rules correspond to the closure conditions of the admissible evidence functions. In fact, because of this, given a frame $\mathcal{F} = (W, R)$ and a set $S$ of $*F$-expressions, the function $\mathcal{E}$ such that $\mathcal{E} \models e \iff S \vdash_{*F} e$ is an admissible evidence function. Furthermore, $\mathcal{E}$ is minimal and unique: if some admissible evidence function $\mathcal{E}'$ is such that for every $e \in S$, $\mathcal{E}' \models e$, then for every $*F$-expression $e$, $\mathcal{E} \models e \Rightarrow \mathcal{E}' \models e$.

Therefore, given a frame $\mathcal{F} = (W, R)$ and two sets $X, Y$ of $*F$-expressions, there is an admissible evidence function $\mathcal{E}$ on $\mathcal{F}$ such that for every $e \in X$, $\mathcal{E} \models e$ and for every $e \in Y$, $\mathcal{E} \not\models e$, if and only if there is no $e \in Y$ such that
CHAPTER 2. DEFINITIONS

*CS(\mathcal{F})** Axioms: \( w * (t, \phi) \), where \( t : \phi \) an instance of AN

*App(\mathcal{F}):\]
\[
\frac{w * (s, \phi \rightarrow \psi) \quad w * (t, \phi)}{w * (s \cdot t, \psi)}
\]

*Sum(\mathcal{F}):\]
\[
\frac{w * (t, \phi) \quad w * (s, \phi)}{w * (s + t, \phi) \quad w * (s + t, \phi)}
\]

If \( J \) has Positive Introspection:

*PI(\mathcal{F}):\]
\[
\frac{w * (t, \phi)}{w * (!t, t : \phi)}
\]

and

*Dis(\mathcal{F}): For any \( (a, b) \in R \),
\[
\frac{a * (t, \phi)}{b * (t, \phi)}
\]

Table 2.4: The *\mathcal{F}-calculus for \( J \): where \( \mathcal{F} = (W, R) \)

\( X \vdash_\ast e \). These statements are made exact by Theorem 2.2.7.

**Theorem 2.2.7.** For any \( J \) justification logic of the ones considered here, \( CS \)
constant specification for \( J \), frame \( \mathcal{F} = (W, R) \) and set \( S \) of *\mathcal{F}-expressions,

The function \( \mathcal{E} \), such that
\[
\mathcal{E}(t, \phi) = \{ w \in W \mid S \vdash_\ast w * (t, \phi) \}
\]
is the unique minimal admissible evidence function such that $E \models S$ – i.e. if there is an admissible evidence function $E' \models S$, then for every $*F$-expression $e$, if $E \models e$ then $E' \models e$.

Proof. Simply notice that $E$ is the closure of $S$ under the admissible evidence (closure) conditions.

Another version of the $*$-calculus – the one that actually appears in [Mkr97, Kru06, Kuz08a] – is not based on a frame. In fact, it is identical to the one we presented if we ignore all frame-related parts (and therefore it does not have rule $*\text{Dis}(F)$). It appears in Table 2.5. This version of the calculus is more appropriate when we discuss Mkrtchyan models and admissible evidence functions on Mkrtchyan models.

An important difference from the $*$-calculus on frames is that we do not know that this one will give us an admissible evidence function. M-models have the additional consistent evidence condition for their admissible evidence functions. Therefore, even if the calculus gives us the closure of a set of $*$-expressions under the other conditions, we are not guaranteed to get an admissible evidence function. We call a set $S$ of $*$-expressions consistent if there is no term $t$ so that $S \vdash_* *(t, \bot)$.

**Corollary 2.2.8.** For any $J$ justification logic of the ones considered here,
*CS Axioms: *(t, φ), where t : φ an instance of AN

*App:

\[
\frac{* (s, φ \rightarrow ψ) \quad * (t, φ) }{ * (s \cdot t, ψ) }
\]

*Sum:

\[
\frac{* (t, φ) \quad * (s, φ) }{ * (s + t, φ) \quad * (s + t, φ) }
\]

If J has Positive Introspection:

*PI:

\[
\frac{* (t, φ) }{ * (!t, t : φ) }
\]

Table 2.5: The *-calculus for J

**CS** constant specification for J, and set S of *-expressions, if S is consistent or J does not have the consistency axiom, then the function \( \mathcal{E} \), such that

\[
\mathcal{E}(t, φ) = \begin{cases} 
\text{true} , & \text{if } S \vdash_\ast *(t, φ) \\
\text{false} , & \text{otherwise.}
\end{cases}
\]

is the unique minimal (M-type) admissible evidence function such that \( \mathcal{E} \models S \) — i.e. if there is an admissible evidence function \( \mathcal{E}' \models S \), then for every *-expression \( e \), if \( \mathcal{E} \models e \) then \( \mathcal{E}' \models e \).

**Definition 2.2.4.** For any justification logic J and constant specification
CHAPTER 2. DEFINITIONS

CS, the reflected fragment of $J_{CS}$ is

$$rJ_{CS} = \{ t : \phi \mid J_{CS} \vdash t : \phi \}.$$  

The $*$-calculus, in addition to providing a way to produce minimal admissible evidence functions, also provides an independent axiomatization of the reflected fragments of the justification logics:

Theorem 2.2.9. For any justification logic $J$ and constant specification $CS$,

$$J_{CS} \vdash t : \phi \iff \vdash \_ {\star}_{CS} (* (t, \phi)).$$

**Proof.** $J_{CS} \vdash t : \phi$ if and only if (Theorem 2.2.2) for every Mkrtchyan model $M$, $M \models t : \phi$, which is true if and only if for every Mkrtchyan model $M = (\mathcal{E}, \mathcal{V}), \mathcal{E} \models * (t, \phi)$, which is true if and only if $\mathcal{E}m \models * (t, \phi)$, where $\mathcal{E}m$ the minimal admissible evidence function (such that $\mathcal{E}m \models \emptyset$), which in turn is true if and only if $\vdash \_ {\star}_{CS} * (t, \phi)$ (Theorem 2.2.8).  

2.3 Computational Complexity

This thesis is mainly focused on Complexity results for Justification Logic. Therefore it is natural to give a little background on Computational Complexity Theory, especially as it relates to Modal Logic and Justification Logic.
CHAPTER 2. DEFINITIONS

2.3.1 Short Background on Complexity Theory

The model of computation assumed is the Turing Machine, although others could be easily used with no, or with minimal, effect. We consider only classes of decision problems, also called languages, often literally considered languages over the alphabet \(\{0, 1\}\). For an overview of Complexity Theory, the reader can see [Pap94].

**Definition 2.3.1.** We define the following time and space-complexity classes:

\[ \text{TIME}(t(n)) \text{ (or DTIME}(t(n))) \text{ is the class of problems that can be decided by a deterministic Turing Machine in time } t(n). \]

\[ \text{NTIME}(t(n)) \text{ is the class of problems that can be decided by a nondeterministic Turing Machine in time } t(n). \]

\[ \text{SPACE}(s(n)) \text{ (or DSPACE}(s(n))) \text{ is the class of problems that can be decided by a deterministic Turing Machine by using additional working space } s(n). \]

\[ \text{NSPACE}(s(n)) \text{ is the class of problems that can be solved by a nondeterministic Turing Machine by using additional working space } s(n). \]

For a complexity class \(C\), \(\text{co}C\) is the class of problems with their complement in \(C\), i.e. \(\text{co}C = \{A \subseteq \{0, 1\}^* \mid \overline{A} \in C\}. \)
The complexity classes that interest us in this thesis are the following:

- $P = \bigcup_{k \geq 1} \text{DTIME}(n^k)$;
- $NP = \bigcup_{k \geq 1} \text{NTIME}(n^k)$;
- $\text{PSPACE} = \bigcup_{k \geq 1} \text{DSpace}(n^k)$;
- $\text{NPSPACE} = \bigcup_{k \geq 1} \text{NPSPACE}(n^k)$;
- $\text{EXP} = \bigcup_{k \geq 1} \text{DTIME}(2^{n^k})$;
- $\text{NEXP} = \bigcup_{k \geq 1} \text{NTIME}(2^{n^k})$;
- $\text{NP}^C$ is the class of problems that can be decided by a nondeterministic Turing Machine that uses time $p(n)$, where $p$ a polynomial, and an oracle from a problem in $C$;
- the Polynomial Hierarchy,

$$\text{PH} = \bigcup_{k \in \mathbb{N}} \Sigma^p_k = \bigcup_{k \in \mathbb{N}} \Pi^p_k,$$

where

$$\Sigma^p_0 = \Pi^p_0 = P$$

and for $k \in \mathbb{N}$,

$$\Sigma^p_{k+1} = \text{NP}^{\Sigma^p_k} \quad \text{and} \quad \Pi^p_{k+1} = \text{coNP}^{\Sigma^p_k}.$$
It is easy to see that
\[ P = \text{coP} \subseteq \text{NP} \subseteq \text{PH} \subseteq \text{PSPACE} = \text{coPSPACE} \subseteq \text{EXP} = \text{coEXP} \subseteq \text{NEXP}. \]

As for inequalities, we know that
\[ P \neq \text{EXP} \quad \text{and} \quad \text{NP} \neq \text{NEXP}, \]
(by the Time Hierarchy Theorems [HS65, Coo73]) but not much more; we also know that \( \text{PSPACE} = \text{NPSPACE} \) (Savitch’s theorem [Sav70]). It is fairly standard to assume that the remaining inclusions of these complexity classes are proper, though.

**Alternating Classes** Often we use alternating complexity classes – or rather alternating characterizations of certain classes. These are complexity classes characterized relative to an *alternating Turing Machine*. An alternating Turing Machine is a nondeterministic Turing Machine, only it has universal and existential states. Like with a nondeterministic TM, a configuration with an existential (respectively universal) state can reach an accepting configuration if at least one (resp. all) of its subsequent configurations can reach an accepting state. Therefore, a usual nondeterministic TM is an alternating TM with only existential states.

\( \text{ATIME}(t(n)) \) is the class of problems that can be decided by an *alternating*
Turing Machine in time \( t(n) \), i.e. a TM which can make both existential and universal nondeterministic choices.

\( \text{ASPACE}(s(n)) \) is the class of problems that can be solved by an *alternating* Turing Machine by using additional working space \( s(n) \).

Then, \( \text{AP} = \bigcup_{k \geq 1} \text{ATIME}(n^k) \) and \( \text{APSPACE} = \bigcup_{k \geq 1} \text{ASPACE}(n^k) \). These are not really new complexity classes, but very useful alternative characterizations of ones we have presented before:

**Theorem 2.3.1.** \( \text{AP} = \text{PSPACE} \ (\text{[SM73]}); \text{APSPACE} = \text{EXP} \ (\text{[CKS81]}). \)

![Computation Tree](image)

Each node represents a configuration, with the root being the starting configuration. An \( \exists \) configuration eventually accepts if and only if there is a next configuration that eventually accepts; a \( \forall \) configuration eventually accepts if and only if all of its next configurations eventually accept.

Figure 2.1: The computation tree of an alternating TM (of exactly two choices a each step).
Reductions, Hard, and Complete Problems  When we prove that a problem is in a complexity class $C$, we establish an upper bound for its complexity: we know that the problem can be solved using the resources specified by $C$. On the other hand, the problem may be even easier to solve, belonging into a lower complexity class. If we want to establish a lower bound for the complexity of a problem, we must prove that the problem is one of the hardest problems in $C$. For this, we need to be able to compare problems with respect to their hardness.

We say that a language (decision problem) $A$ is polynomially (or Karp) reducible to $B$, $A \leq_p B$, if there is a polynomial time algorithm $M$ which transforms instances of $A$ to instances of $B$ and for every instance $x$ of $A$, $x \in A$ if and only if $M(x) \in B$. Then we say that $A$ is $C$-hard if for every $B \in C$, $B \leq_p A$ and that $A$ is $C$-complete if $A$ is $C$-hard and $A \in C$.

To establish a lower bound for a problem – i.e. to establish that it is $C$-complete for a certain class $C$ – we usually give a reduction to that problem from a known $C$-complete problem. Probably, the most well-known $\text{NP}$-complete problem is $\text{SAT}$ [Coo71], or Propositional Satisfiability. These are not exactly the same problem, but they are close. Furthermore we discuss several versions of satisfiability (of Propositional, First-order, Modal, and Justification Logic) in this thesis, so it makes sense to consider $\text{SAT}$ to be
the propositional version of satisfiability.

In our reductions we use the following problems:

\textbf{QBF}_2: \quad \text{Given a Quantified Boolean Formula,}

\[ \phi = \exists x_1 \exists x_2 \cdots \exists x_k \forall y_1 \forall y_2 \cdots \forall y_{k'} \psi, \]

where \( \psi \) is a propositional formula on propositional variables \( x_1, \ldots, x_k, y_1, \ldots, y_{k'} \), is \( \phi \) true? That is, are there truth-values for \( x_1, \ldots, x_k \), such that for all truth-values for \( y_1, \ldots, y_{k'} \), a truth-assignment that gives these values makes \( \psi \) true?

\( \text{QBF}_2 \) is known to be \( \Sigma^p_2 \)-complete [Wra76].

\textbf{QBF}: \quad \text{Given a Quantified Boolean Formula,}

\[ \phi = Q_1 x_1 \cdots Q_k x_k, \]

where \( \psi \) is a propositional formula on propositional variables \( x_1, \ldots, x_k \) and \( Q_1, \ldots, Q_k \in \{ \exists, \forall \} \), is \( \phi \) true?

\( \text{QBF} \) is \( \text{PSPACE} \)-complete [SM73].

\textbf{SCHÖNFINKEL-BERNAYS SAT:}

Given a first-order formula \( \phi \) of the form

\[ \exists x_1 \cdots \exists x_k \forall y_1 \cdots \forall y_{k'} \psi, \]
where $\psi$ contains no quantifiers or function symbols, is $\phi$ satisfiable?

SCHÖNFINKEL-BERNAYS SAT is NEXP-complete [Lew80].

**BINARY SCHÖNFINKEL-BERNAYS SAT:**

Given a first-order formula $\phi$ of the form

$$\exists x_1 \cdots \exists x_k \forall y_1 \cdots \forall y_k \psi,$$

where $\psi$ contains no quantifiers or function symbols, is $\phi$ satisfiable by a first-order model of exactly two elements?

Lemma 2.3.2 was given in [Ach15a] and demonstrates that although BINARY SCHÖNFINKEL-BERNAYS SAT (BINARY S-B SAT from now on) is a special case of SCHÖNFINKEL-BERNAYS SAT (S-B SAT from now on), it retains all the hardness of the original problem – i.e. it is NEXP-complete.

**Lemma 2.3.2.** BINARY S-B SAT is NEXP-complete.

*Proof.* We use the following notation: for a non-negative integer $x \in \mathbb{N}$, let $\text{bin}(x) = \text{bin}_0(g), \ldots, \text{bin}_{\log g}(g)$ be its binary representation. Let $\phi$ be a first-order formula of the form

$$\exists x_1 \cdots \exists x_k \forall y_1 \cdots \forall y_k \psi,$$
where \( \psi \) contains no quantifiers or function symbols. Furthermore, we assume that \( \psi \) contains no constants. We replace each \( x_a \) by \( \vec{x}_a = x_{a1}, x_{a2}, \ldots, x_{a^{\lceil \log k \rceil}} \) and each \( y_b \) by \( \vec{y}_b = y_{b1}, y_{b2}, \ldots, y_{b^{\lceil \log k \rceil}} \) in the quantifiers and wherever they appear in a relation. Therefore \( \exists x_a \) is replaced by \( \exists x_{a1} \exists x_{a2} \cdots \exists x_{a^{\lceil \log k \rceil}} \) (\( \exists \vec{x}_a \) for short) and \( \forall x_a \) is replaced by \( \forall x_{a1} \forall x_{a2} \cdots \forall x_{a^{\lceil \log k \rceil}} \) (\( \forall \vec{y}_a \) for short) and \( R(z_1, \ldots, z_r) \) is replaced by \( R(\vec{z}_1, \ldots, \vec{z}_r) \). Furthermore, every expression \( z = z' \) where \( z, z' \) are variables, is replaced by \( \bigwedge_{1 \leq a \leq \lceil \log k \rceil} z^a = z'^a \) (\( \vec{z} = \vec{z}' \) for short). The result of all these replacements in \( \psi \) is called \( \psi' \). The new formula is:

\[
\phi' = \exists \vec{x}_1 \cdots \exists \vec{x}_k \forall \vec{y}_1 \cdots \forall \vec{y}_{k'} \left( \bigwedge_{b=1}^{k'} \bigvee_{a=1}^{k} \vec{x}_a = \vec{y}_b \rightarrow \psi' \right)
\]

We can also define a corresponding transformation of first-order models: assume that the universe of model \( M \) for \( \phi \) is a set of at most \( k \) natural numbers (each of which is at most \( k - 1 \) and an interpretation for some \( x_a \)); then \( M' \) is the model with \( \{0, 1\} \) as its universe, where for every relation \( R \) (on tuples of naturals) of \( M \) there is some \( R' \), which is essentially the same relation, but on the binary representations of the elements of \( M \). That is,

\[
R' = \{(\text{bin}(a_1), \ldots, \text{bin}(a_r)) \in \{0, 1\}^* \mid (a_1, \ldots, a_r) \in R\}
\]

It is not hard to see that if \( M \) satisfies the original formula, then \( M' \) satisfies the new one: each \( \vec{x}_a \) can be interpreted as the binary representation of
the interpretation of $x_a$ in $\mathcal{M}$ and notice that the added equality assertions effectively limit the $\vec{y}$'s to range over the interpretations of the $\vec{x}$'s, which are then exactly the image of the elements of $\mathcal{M}$.

On the other hand, if $\phi'$ is satisfied by a model with $\{0, 1\}$ as its universe, then $\phi$ is satisfied by the model which has the $\lceil \log k \rceil$-tuples of $\{0, 1\}$ that are the interpretations of $\vec{x}_1, \ldots, \vec{x}_k$ as elements and as relations the restrictions of the two-element model’s relations on these tuples.

Another way to prove completeness of a problem $A$ for a class $C$ is by a reduction from a generic problem in $C$. To be able to do this we need a normal form of the generic problem in $C$. But this normal form is readily provided in the form of a Turing Machine that satisfies the restrictions of $C$. Therefore, a PSPACE problem can be represented by a deterministic/nondeterministic TM which uses polynomial space, or an alternating TM which uses polynomial time; an EXP problem can be represented by a deterministic TM which uses exponential time, or an alternating TM which uses polynomial space; and so on. Therefore we may use a reduction from a Turing Machine to show completeness with respect to a certain complexity class.
Figure 2.2: A hierarchy of complexity classes
2.3.2 The Complexity of Modal Logic

It is immediately apparent that the satisfiability problem for all logics mentioned here is \(\text{NP}\)-hard, as any propositional formula is classically satisfiable iff it is satisfiable for any modal or justification logic and, furthermore, the satisfiability problem for classical propositional logic is known to be \(\text{NP}\)-complete. Therefore, to establish \(\text{NP}\)-completeness for the satisfiability problem on a modal logic, all that is needed is to show that the problem is in \(\text{NP}\).

Most modal logics that interest us are \(\text{PSPACE}\)-complete.

Ladner proved in [Lad77] that satisfiability for certain modal logics (\(\text{K}, \text{D}, \text{T}, \text{K4}, \text{D4}, \text{S4}\)) is \(\text{PSPACE}\)-complete, while for \(\text{S5}\) and \(\text{KD45}\) it is in \(\text{NP}\). Halpern and Rêgo in [HR07] characterized the gap in the complexity of Modal Logic, determining that negative introspection is the cause of the drop in complexity for \(\text{S5}\) and \(\text{KD45}\), since for all these logics, satisfiability drops to \(\text{NP}\) if we add negative introspection. The claims of this subsection are originally from [Lad77], but they were established in their full generality (which we present here) in [HR07].

Proposition 2.3.3. Given a structure \(M\), and a modal formula \(\phi\), there is an algorithm for checking if \(\phi\) is satisfied in \(M\), that runs in time \(O(|M| \cdot |\phi|)\).

Proof. In time \(O(|\phi|)\), an increasing (in terms of length) sequence of all
subformulas of $\phi$ can be produced. Taking each formula in turn, we can fill a $|\phi| \times |M|$ matrix that keeps the information of which formula is true in which state. By the definition of truth, this can be done in the required time. Then, all we have to do is check if $\phi$ is satisfied in any state.

**Theorem 2.3.4.** If $M$ is one of $K$, $T$, $D$, $S4$ (or any other combination of axioms we have given for Modal Logic), then a $M+5$-satisfiable formula can be satisfied by an $M+5$-model of at most $O(|\phi|)$ states.\(^6\)

*Proof.* Let $\mathcal{M} = (W, R, V)$ be an $M+5$-model and $s \in W$, such that $\mathcal{M}, s \models \phi$. Let $W_R = \{w \in W \mid \exists w' Rw\}$; we assume that there are no worlds in $W$ to which there is no path from $s$, therefore $W = W_R \cup \{s\}$. For convenience we also assume $\phi$ is in negation normal form: all negations are pushed to the level of the propositional connectives. Since $\mathcal{M}$ is an $M+5$-model, $R$ is euclidean (if $aRb, c$, then $bRc$). Therefore, the restriction of $R$ on $W_R$ is reflexive (for all $a \in W_R$, $aRa$). This in turn means $R$ is symmetric in $W_R$: if $a, b \in W_R$ and $aRb$, since $aRa$, we also have $bRa$. Finally, $R$ is transitive in $W_R$: if $aRbRc$ and $a, b, c \in W_R$, then $bRa$, so $aRc$. Therefore $R$ is an equivalence relation when restricted on $W_R$.

If $W_R = \emptyset$, then we are done. Otherwise let $s_R \in W_R$. Notice that if for

\(^6\)Given a logic $M$ and axiom $A$, $M + A$ is the logic that results from adding axiom $A$ to $M$.\]
some \( s' \in W \), \( \mathcal{M}, s' \models \diamond \psi \), then \( \mathcal{M}, s_R \models \diamond \psi \).

Let

\[
F = \{ \psi \in \text{sub}(\phi) \mid \mathcal{M}, s_R \models \diamond \psi \} \cup \{ \neg \psi \in L_M \mid \psi \in \text{sub}(\phi), \mathcal{M}, s_R \models \diamond \neg \psi \}.
\]

For every \( \psi \in F \), there is some \( s_\psi \in W_R \), such that \( \mathcal{M}, s_\psi \models \psi \); let

\[
W' = \{ s_\psi \in W_R \mid \psi \in F \} \cup \{ s \},
\]

\( R' \) the restriction of \( R \) on \( W' \), and \( V' \) the restriction of \( V \) on \( W' \) – i.e. \( V'(p) = V(p) \cap W' \). By induction on \( \psi \), where \( \psi \) a subformula of \( \phi \), we can show that for every \( s' \in W' \), \( \mathcal{M}, s' \models \psi \) iff \( \mathcal{M}', s' \models \psi \): propositional variables and connectives are trivial; if \( \psi = \square \psi' \) and \( \mathcal{M}, s' \models \square \psi' \), then for every \( s'' \in W_R \), \( \mathcal{M}, s'' \models \psi' \), then for every \( s'' \in W_R \cap W' \), \( \mathcal{M}, s'' \models \psi' \), then (by the inductive hypothesis) for every \( s'' \in W_R \cap W' \), \( \mathcal{M}', s'' \models \psi' \), then \( \mathcal{M}, s' \models \square \psi' \); if \( \psi = \square \psi' \) and \( \mathcal{M}, s' \not\models \square \psi' \), then \( \mathcal{M}, s' \models \diamond \neg \psi' \), then \( \neg \psi' \in F \), so there is some \( s_{\neg \psi'} \in W' \cap W_R \), so \( \mathcal{M}, s_{\neg \psi'} \models \neg \psi' \), so (by I.H.) \( \mathcal{M}', s_{\neg \psi'} \models \neg \psi' \), so \( \mathcal{M}', s_{\neg \psi'} \not\models \diamond \psi' \); the cases for \( \diamond \) are symmetrical.

That is, an \( M + 5 \)-frame can be considered to be either one single equivalence class or an equivalence class with the addition of a node \( s \).

From the above, the following corollary follows immediately. To decide whether a formula \( \phi \) is satisfiable, all we have to do is guess nondeterministically a Kripke model \( \mathcal{M} \) of size at most \( |\phi| \) and then check in time \( O(|\phi|^2) \) if \( \phi \) is satisfied in \( \mathcal{M} \).
Corollary 2.3.5. The satisfiability problem for $M+5$ is $\text{NP}$-complete. Therefore, the validity problem for these logics is $\text{coNP}$-complete.

Theorem 2.3.6. The satisfiability problem for $K$, $T$, $D$, $K4$, $D4$, and $S4$ is $\text{PSPACE}$-hard.

Proof. The proof is by reducing the problem $QBF$ (quantified boolean formula) to the satisfiability problem for these logics. One way to check if a $QBF$ formula is true or not is the following. If the formula has no quantifiers, i.e. all occurrences of variables are replaced by truth values, just evaluate its truth value. If the formula is of the form $\exists p\psi$, replace it with $\psi[\top/p] \lor \psi[\bot/p]$\footnote{$\psi[\chi/p]$ is the result of substituting $p$ by $\chi$ in $\chi$.} and if it is of the form $\forall p\psi$, with $\psi[\top/p] \land \psi[\bot/p]$. Then, check if the new formula is true. The correctness of this procedure is immediate and follows from the definition of the problem.

To construct the reduction from $QBF$ to $S4$, a formula will be constructed, that will describe the above procedure. Suppose the given formula, $\phi$, is $Q_1p_1 \cdots Q_mp_m\psi$, where $\psi$ contains no quantifiers and $Q_i \in \{\exists, \forall\}$. We construct formula $\phi^{S4}$. The following propositional variables will be used: $p_1, \ldots, p_m, d_0, \ldots, d_{m+1}$. First, the following formulas are defined.

$$\text{depth} = \bigwedge_{i=1}^{m+1} (d_i \to d_{i-1})$$
determined = $\bigwedge_{i=1}^{m} (d_i \rightarrow ((p_i \rightarrow \Box(d_i \rightarrow p_i)) \land (\neg p_i \rightarrow \Box(d_i \rightarrow \neg p_i))))$

branching =

$\bigwedge_{Q_{i+1}=\forall} (d_i \land \neg d_{i+1}) \rightarrow (\Diamond(d_{i+1} \land \neg d_{i+2} \land p_i) \land \Diamond(d_{i+1} \land \neg d_{i+2} \land \neg p_i))$

$\land$

$\bigwedge_{Q_{i+1}=\exists} (d_i \land \neg d_{i+1}) \rightarrow (\Diamond(d_{i+1} \land \neg d_{i+2} \land p_i) \lor \Diamond(d_{i+1} \land \neg d_{i+2} \land \neg p_i))$

and finally,

$\phi_{S4}^A = d_0 \land \neg d_1 \land (\text{depth} \land \text{determined} \land \text{branching} \land (d_m \rightarrow B))$.

The role of the formula depth is to make the truth values of the variables $d_1, \ldots, d_{m+1}$ characterize the depth of the decision tree for the procedure. determined expresses the fact that once a truth value is assigned to a variable in the decision tree, that value does not change when we go deeper in the tree. Finally, branching describes if the current node of the decision tree is universal, or existential, and how the tree continues. The size of $\phi_{S4}^A$ is linear w.r.t. the size of $\phi$, and it is easily constructed from $\phi$. As mentioned before, $\phi_{S4}^A$ mimics the previous procedure for determining the truth of QBF formulas, on an reflexive and transitive structure. Therefore it is easy to see that $\phi$ is true if and only if $\phi_{S4}^A$ is satisfiable by a reflexive and transitive structure.
Respectively, the formulas constructed for $T$ and $K$ are $\phi^T$ and $\phi^K$. They are the same as $\phi^{S4}$, except that to deal with the fact that structures for $T$ are not transitive, and structures for $K$ are not even reflexive,

$$\phi^T = d_0 \land \neg d_1 \land \Box^m (\text{depth} \land \text{determined} \land \text{branching} \land (d_m \rightarrow \psi)),$$

and

$$\phi^K = d_0 \land \neg d_1 \land \bigwedge_{i=0}^{m} \Box^i (\text{depth} \land \text{determined} \land \text{branching} \land (d_m \rightarrow \psi)),$$

We leave the remaining cases to the reader.

Below, a PSPACE procedure is given, that decides whether a formula is satisfiable.

**Theorem 2.3.7.** If $M \in \{K, D, T, K4, D4, S4\}$, then the satisfiability problem for $M$ is in PSPACE.

**Proof.** An alternating algorithm that runs in polynomial time will be given below. This proves that the satisfiability problem for these logics is in APTIME=PSPACE.

The algorithm constructs a prefixed tableau branch for the given formula, $\phi$. At each step, a tableau rule is applied to add one or two new formulas to the constructed branch, which we will call $b$. At first, $b$ only contains 1.$\phi$, 
which is not marked. $b$ is a set, i.e. it contains each element at most once.

The algorithm does the following. We call a rule of the tableau a local rule for prefix $a$ if it is of the form $\frac{a \alpha}{a \beta}$ (so the prefix does not change). Thus, propositional rules are local and so are rules for axioms $T$ and $D$. We keep a “current prefix” $w$ in memory. Initially $w = 1$.

1. Keep applying local rules for $w$ as long as such a rule can be applied; if a rule branches the tableau, nondeterministically (existential choice) choose which branch you keep.

2. If the branch has both $w T \psi$ and $w F \psi$ in it, reject.

3. Universally choose a rule which produces a new prefix, called $w'$; if no such rule exists, accept.

4. Apply all tableau rules that can be applied to formulas prefixed by $w$ to give formulas prefixed by $w'$. These new formulas are called the initial formulas for $w'$.

5. If the initial formulas for $w'$ are the same as the initial formulas for $w$, (terminate and) accept.

6. Delete all formulas prefixed by $w$.

7. $w := w'$. 
8. Go back to 1.

If 4 is an axiom of $M$, then since to increase the value of $|w|$, the complexity of the formula used in the rule decreases, $|w| \leq |\phi|$. If it is not, then since by the tableau rules for 4, the initial formulas of a prefix can only be more than those of its previous one and we have an upper bound of $|\phi|$ for those, $|w| \leq |\phi|$. The number of different prefixed formulas with at most the same complexity as $\phi$ is therefore at most $|\phi|^2$. Every step of the algorithm applies a rule to a different unmarked prefixed formula and marks it. So, the algorithm runs in at most $|\phi|^2$ steps. Each step takes at most $O(|\phi|^2)$ time, so the algorithm uses polynomial time.

The correctness of the algorithm is immediate, as it simply follows the tableau construction rules. Specifically, if $\phi$ is satisfiable, the algorithm can accept by simply exploring certain parts (chosen universally) of an accepting branch (chosen existentially). On the other hand, if the algorithm accepts, let $b'$ be the union over universal choices of the sets of prefixed formulas the algorithm produces. If the algorithm accepts because of step 5 at prefixes $w, w'$, then $b'$ can easily be expanded to a complete accepting branch $b$, so that $b$ at $w'$ recursively simulates $b$ at $w$. \hfill $\square$

\footnote{Depending on how you count the upper bound may be $|\phi| + 2$, but this difference is not important.}
Corollary 2.3.8. The satisfiability problem for \( K, D, T, K4, D4, \) and \( S4 \) is PSPACE-complete. Therefore, the validity problem for these logics is PSPACE-complete.

Figure 2.3 demonstrates the placement of these logics in the complexity hierarchy.
Figure 2.3: The complexity of Unimodal Logic: in the figure, unless otherwise specified, modal satisfiability (or validity, provability) refers to Unimodal Logic (one modality).
Chapter 3

The Complexity of Single-agent Justification Logic

In this chapter we examine the complexity of satisfiability for single-agent Justification Logic — we mostly phrase the problem we study in terms of satisfiability instead of validity/derivability, as it is not hard to translate from one to the other: $\phi$ is satisfiable if and only if $\neg\phi$ is not valid. What distinguishes Justification Logic from Modal Logic, if one compares their semantics or axiomatizations, is the presence of justification terms. Therefore, in Section 3.1 we present upper bounds for the complexity of the $\ast$-calculus, which gives us a way to deal with justification terms in a satisfiability-testing procedure. Then we present tableaux (that result in upper bounds) based on M-models for $J$, $J4$, $JT$, and $LP$ and tableaux based on F-models for $JD$ and $JD4$. Finally, we give a general lower bound for all these logics, which shows that the upper bounds of the chapter are tight.
3.1 Reflected Fragments and the *-calculus

As we see in this chapter (and all the following ones as well), the *-calculus is an integral part of any algorithm that has been developed to test satisfiability for Justification Logic. Therefore we first examine the derivability-from-assumptions problem for the *-calculus. This section’s results are mainly due to Krupski [Kru06] and Kuznets [Kuz08a]; the results concerning +-free terms are from [AK06, AK09, AK13]. We first present a nondeterministic polynomial-time algorithm for derivability in the *-calculus.

**Proposition 3.1.1.** If $\mathcal{CS}$ is schematic and in $P$, then the following problem is in NP:

Given a finite set $S$ of *-expressions and a *-expression $e$, is it the case that

$$S \vdash_{\ast_{\mathcal{CS}}} e ?$$

**Proof.** The shape of a *-calculus derivation is mostly given away by the term $t$. So, we can use $t$ to extract the general shape of the derivation – the term keeps track of the applications of all rules. We can then plug in to the leaves of the derivation either axioms of the calculus or members of $S$ and unify ($\mathcal{CS}$ is schematic, so the derivation may include schemes) trying to reach the root.
The algorithm to decide the derivability of $*(t, \phi)$ from $S$ is the following:

- Nondeterministically construct a rooted tree with subterms of $t$, as nodes, such that $t$ is the root and the following conditions are met. Node $s$ can be the parent of $s_1$ or of both $s_1, s_2$ as long as there is a rule $\frac{*(s_1, \phi_1)}{*(s, \phi_3)}$ or $\frac{*(s_1, \phi_1) \cdot *(s_2, \phi_2)}{*(s, \phi_3)}$, respectively, of the $*$-calculus and $s_1, s_2$ are proper subterms of $s$. This results in a subtree of term $t$ (when $t$ is seen as a tree) and the nondeterministic choices correspond to three cases: wherever $+$ appears in $t$ (and we have to choose a version of the $*$Sum rule); whenever we encounter a term $s$, for which there is some $s : \psi \in S$, so we can choose to not break down $s$ any further; when we encounter some subterm $s = ! \cdots ! c$, where $c$ a constant, as $s$ can be part of an axiom of the calculus, or a result of consecutive applications of $*$PI on an axiom, but this is a choice without consequences.

- Nondeterministically assign to each leaf, $l$, either
  - some formula $\psi$ such that there is some $*(l, \psi) \in S$ or
  - as long as $l$ is of the form $! \cdots ! c$, where $c$ a constant, $k \geq 0$, then we can also assign some $! \underbrace{\cdots ! c : \cdots ! c : c : A}_{k-1}$, where $A$ an axiom scheme and $c : A \in CS$.

- If for some node $s$, all its children, say $s_1, s_2$ (resp. $s_1$) have been
assigned some scheme or formula $P_1, P_2$ (resp. $P_1$), assign to $s$ some scheme or formula $P$, such that $P_1, P_2$ can be unified to $P'_1, P'_2$ so that $\frac{s(s_1, P'_1)}{s(s, P)}$ is a rule (resp. $\frac{s(s_1, P'_1)}{s(s, P)}$ is a rule) in the $*_{CS}$-calculus. Apply this step until the root of the tree has been assigned some scheme or formula.

- Unify $\phi$ with the formula assigned to $t$.

If some step is impossible, the algorithm rejects. Otherwise, it accepts.

Using efficient representations of schemes using DAGs and Robinson’s unification algorithm, the algorithm runs in polynomial time (with respect to $|S| + |t| + |\phi|$). We can see that as the tree is constructed, if $s$ is assigned scheme $P$, then the construction effectively describes a valid derivation of any expression of the form $*(s, \psi)$, where $\psi$ any instance of $P$. Therefore, if the algorithm accepts, there exists a valid $*-calculus derivation of $*(t, \phi)$. On the other hand if there is some $*-calculus derivation for $*(t, \phi)$ from $S$, then the algorithm in the first two steps can essentially describe this derivation by producing the derivation tree and the formulas/schemes by which the derivation starts. Therefore, the algorithm accepts if and only if there is a $*-calculus derivation for $*(t, \phi)$ from $S$. 

\[\square\]

**Corollary 3.1.2.** Let $J \in \{J, JD, JT, J4, JD4, LP\}$ and $CS$ be a schematic
constant specification for $J$, such that $\mathcal{CS} \in \mathbb{P}$. Then, $rJ_{\mathcal{CS}} \in \text{NP}$.

By adjusting the algorithm from the proof of Proposition 3.1.1, we can prove Proposition 3.1.3, which was proven in [AK09] - see also [AK06].

**Proposition 3.1.3.** Assume $J \in \{J, JD, JT, J4, JD4, LP\}$ and that $\mathcal{CS}$ is a schematically injective constant specification for $J$, such that $\mathcal{CS} \in \mathbb{P}$. Then, there exists a deterministic algorithm that runs in polynomial time and determines, given a formula $\phi$ and a term $t$ with no appearances of $+$, whether $\vdash_{\mathcal{CS}} \ast(t, \phi)$.

**Proof.** Simply notice that if $\mathcal{CS}$ is schematically injective, $+$ does not occur in $t$, and there are no assumptions for the derivation, we have eliminated all nondeterministic choices from the algorithm of the proof of Proposition 3.1.1. $\square$

**Definition 3.1.1.** Let $J \in \{J, JD, JT, J4, JD4, LP\}$ and $\mathcal{CS}$ be a constant specification for this logic. Then,

\[ rJ_{\mathcal{CS}} = \{t : \phi \in rJ_{\mathcal{CS}} | + \text{ does not appear in } t \}. \]

**Corollary 3.1.4.** Let $J \in \{J, JD, JT, J4, JD4, LP\}$ and $\mathcal{CS}$ be a schematically injective constant specification for $J$, such that $\mathcal{CS} \in \mathbb{P}$. Then, $rJ_{\mathcal{CS}} \in \mathbb{P}$. 
Proposition 3.1.3 and corollary 3.1.4 were given by Artemov and Kuznets in [AK09] (see also [AK06, AK13]). Their purpose in [AK06, AK09, AK13] was to provide suggestions on how to treat Logical Omniscience, a widely known, important problem of epistemology. Their suggestion was that Logical Omniscience is an inherently computational complexity problem. Two tests for logical omniscience in that context were presented, LOT and SLOT. An epistemic system passes LOT if and only if for each valid knowledge assertion of $F$, there is a proof of $F$ of size bounded by a polynomial of $|F|$. If furthermore this proof can be retrieved in polynomial time, the epistemic system passes SLOT. By corollary 3.1.4, the epistemic system that occurs from $r_{J CS}$ passes SLOT (and LOT), therefore being a non-logically omniscient system in a strong sense\footnote{Note that we are talking about a generalized notion of proof here.}. On the other hand, $r_{JC S}$ passes LOT, thus being a non-logically omniscient system in the weaker, LOT sense.

Notice that we can tweak the algorithm from the proof of Proposition 3.1.1 so that it works for a $\ast$-calculus based on a frame $\mathcal{F} = (W, R)$ by keeping track of the states for which the derivation succeeds; that is, as we assign scheme $P$ to the node that corresponds to term $s$, we keep track of a set $V$ of all states $v$ such that we can derive $v \ast (s, P)$. If we assign an axiom to a leaf, then we also assign $W$ to that leaf; if we assign a formula $\psi$ such
that $w * (s, \psi) \in S$, then we also assign to it some $V \subseteq W$ such that (if the logic has Positive Introspection, then) $V$ is (the closure under $R$ of) the set
\[ \{ v \in W \mid v * (s, \psi) \in S \}. \]
If the children of $s$ have been assigned with $V_1$ and $V_1$, then we assign $V = V_1 \cap V_2$ to $s$. Then it is not hard to verify that if $V$ and $P$ are assigned to the node corresponding to $t$, then the calculus can derive $v * (t, \phi)$ if and only if $\phi$ is an instance of $P$ and $v \in V$. We conclude with the following proposition.

**Proposition 3.1.5.** If $CS$ is schematic and in $P$, then the following problem is in $NP$:

Given a finite frame $F$, a finite set $S$ of $*F$-expressions and a
$*F$-expression $e$, is it the case that

\[ S \vdash_{*F} e \]

### 3.2 Tableaux through Mkrtychev Models for One Agent

The following theorem, due to Kuznets [Kuz00], is based on Mkrtychev model semantics as well as on the properties of the $*$-calculus. Most of the upper bounds presented here have been achieved through a variation of this tableau procedure.
Theorem 3.2.1. Satisfiability for $J_{CS}$, $JD_{CS}$, $J4_{CS}$, $LP_{CS}$ with a schematic
$CS \in P$ is in $\Sigma_{2}^{p}$.

Proof. The algorithm for deciding these logics consists of two parts. Let $\phi$
be the given formula. The first part constructs a tableau branch, starting
from $T \phi$, and applying all the rules in Table 3.1. The first two are for logics
$J_{CS}$ and $J4_{CS}$. The following two are used for $JT_{CS}$ and $LP_{CS}$. What makes
the difference is Factivity.

For $J$ and $J4$:

- \[
\begin{array}{ll}
T s; \psi & F s; \psi \\
\hline
T * (s; \psi) & F * (s; \psi)
\end{array}
\]

For $JT$ and $LP$:

- \[
\begin{array}{ll}
T s; \psi & F s; \psi \\
\hline
T \psi & F * (s; \psi) | F \psi \\
T * (s; \psi)
\end{array}
\]

Table 3.1: Tableau rules for $J$, $JT$, $J4$, and $LP$.

We do not explicitly provide them, but, of course, we need a set of rules
to cover propositional cases as well. In particular we can use the same propo-
sitional rules we used for Modal Logic (see Table 2.2 and ignore the state-
prefixes).

Like before, separator | indicates a nondeterministic choice between the
two options it separates. The *-expressions are not analyzed any further
for the moment; intuitively they can be interpreted as requirements of an
admissible evidence function of a Mkrtychev model - \( T^* (t, \psi) \) corresponds to \( \mathcal{E}(t, \psi) = \text{true} \). If no more rules can be applied, the branch is called complete, and if it contains either \( T \perp \) or both \( T \psi \) and \( F \psi \), it is called closed. Every application of a rule reduces the complexity of the formulas, and, furthermore, the sum of the lengths of the resulting formulas is at most as much as the length of the initial formula. Therefore, a closed, or a completed tableau branch is reached in time linear with respect to the length of the initial formula.

The second stage of the algorithm starts when a complete, or closed branch is formed. If the branch is closed, reject. If not, then we know we can decide in nondeterministic polynomial time if for some \( * \)-expression \( e \), such that \( F e \) was produced by the first part, \( X \vdash_{*CS} e \), where \( X \) is the set of all positively prefixed \( * \)-expressions in the branch. If the answer is “no”, then accept (and the branch is called accepting). Otherwise, reject.

The algorithm above runs in polynomial time, uses nondeterministic choices and uses an \( \text{NP} \) oracle. Furthermore it is correct, that is it accepts \( \phi \) if and only if \( \phi \) is satisfiable.

If there is an accepting complete branch for \( T \phi \), there is a model for \( \phi \): we can construct a Mkrtychev model from an accepting branch by defining

\[
\mathcal{V}(p) = \text{true} \text{ iff } T \ p \text{ appears and using the minimal admissible evidence}
\]
function based on $X$. Then, we can see by induction on $\psi$ that the model satisfies $\psi$ if $T \psi$ in the branch and not if $F \psi$ is in the branch.

If $\phi$ is satisfiable, then there is an accepting complete branch for $T \phi$: we can construct an accepting branch from a Mkrtchyan model which satisfies $\phi$ by inductively (on the applications of the rules) ensuring that all nondeterministic choices produce formulas or $*$-expressions satisfied in the model.

Therefore, satisfiability for these logics is in $NP = \Sigma_2^p$.

3.3 Consistency and Fitting Models

In the case of JD, the same method cannot be applied as is, since the $*$-calculus is not enough to provide with certainty a legitimate admissible evidence function. In JD and JD4 Mkrtchyan models, an additional condition must be met, that is the consistency condition of the admissible evidence function. For these logics we base our tableau rules on Fitting model semantics. The tableau for JD is due to Kuznets [Kuz08a, Kuz09], while the one for JD4 is from [Ach14b]. The most important observation of this section is that for certain cases we have to (and that we also can) use F-models to base our tableaux instead of using M-models.
\[ \frac{n T \ s \ \psi}{n T \ast (s, \psi)} \quad \frac{n F \ s \ \psi}{n F \ast (s, \psi) \mid n + 1 F \ \psi} \]

Table 3.2: Tableau for JD.

**Theorem 3.3.1.** JD\(\mathcal{CS}\)-satisfiability, where \(\mathcal{CS} \in \mathbb{P}\) and is schematic and axiomatically appropriate, is in \(\Sigma^p_2\).

**Proof.** For this theorem we simply provide the tableau rules (propositional rules omitted). The difference in this case is that admissible evidence functions for JD\(\mathcal{CS}\) models must satisfy the Consistent Evidence function condition. That is, for no admissible evidence function \(\mathcal{E}\) and term \(t\), should it be true that \(\mathcal{E}(t, \bot)\). To deal with this, the algorithm is modified in the following ways:

- The formulas in the tableau are also prefixed with an integer. The formulas are of the form \(n \ V \ \psi\), where \(n\) is a natural number, \(V \in \{T, F\}\), \(\psi\) is a formula of the logic. Instead of the tableau rules introduced for J, JT, J4, and LP, we use the rules from Table 3.2.

- The \(\ast\)-calculus we run now is the one based on frames of Fitting models. The frame we base the calculus on is \(\mathcal{F} = (W, R)\), where \(W\) is the collection of all world-prefixes that appear in the branch and

\[ R = \{ (i, i + 1) \in W^2 \} \cup \{ (i, i) \mid i = \max(W) \}. \]
• Instead of having $X$ be the set of positive $\ast$ expressions in the branch, it is the set of positive $\ast^F$-expressions in the branch.

As before, the algorithm runs in polynomial time and correctness is what is left to show.

**Lemma 3.3.2.** A formula $\phi$ is $\text{JD}_{\text{CS}}$-satisfiable if and only if the tableau that is produced from $1^T\phi$ has a complete, not closed branch, where for every $\ast$-expression $e$, where $nF e$ appears in the branch, $X \not\vdash_{\text{CS}} e$.

**Proof.** The proof is similar to the proof of Theorem 3.2.1 and proceeds by proving that a formula $\phi$ is satisfiable in $\text{JD}_{\text{CS}}$, if and only if the tableau that is produced from $1^T\phi$ has a complete accepting branch.

First, let’s assume that $1^T\phi$ has a complete accepting branch. Let $\mathcal{V}(p) = \{n \in W \mid n T p \text{ appears in the branch}\}$ and $\mathcal{M} = (W, R, \mathcal{V})$. $\mathcal{M}$ is, of course, a $\text{JD}_{\text{CS}}$-model by induction on $\psi$, we can see that for every $n T \psi$ in the branch, $\mathcal{M}, n \models \psi$, while for every $n T \psi$ in the branch, $\mathcal{M}, n \not\models \psi$: propositional variables are trivial by the definition of $\mathcal{V}$; the case where $\psi = e$, a $\ast^F$-expression, is a consequence of Theorem 2.2.7; the remaining cases are direct consequences of the tableau rules and the inductive hypothesis.

If there are $\mathcal{M}', a \models \phi$, then let $N - 1$ be the nesting depth of justifications in $\phi$ and let $a_1a_2\cdots a_N$ be any path from $a$ in $M'$ (so $a_1 = a$). Then we can
ensure that whenever we apply a tableau rule and we produce \( n T \psi \), that 
\( \mathcal{M}', a_n \models \psi \), while if we produce \( n T \psi \), that 
\( \mathcal{M}', a_n \not\models \psi \). Therefore, the 
resulting branch is accepting: it can obviously not be closed and if \( X \vdash_{s,e,F} n e \), then 
\( \mathcal{M}', a_n \models e \) (again, by Theorem 2.2.7), so it cannot be the case 
that \( n F e \) has appeared in the branch. 

The proof of the theorem is thus complete. 

The upper bound for \( \text{JD}4 \)-satisfiability is based on Proposition 3.3.3 and 
the Fitting models we base our tableau are the ones described in that proposition.

**Proposition 3.3.3.** A formula \( \phi \) is \( \text{JD}4_{CS} \)-satisfiable if and only if it is 
satisfiable by a Fitting model \( \mathcal{M} = (W, R, E, V) \) for \( \text{JD}4_{CS} \) that additionally 
has the following properties:

- \( W \) has exactly two elements, \( a, b \).

- \( R = \{(a, b), (b, b)\} \).

**Proof.** Let \( \phi \) be a formula that is \( \text{JD}4_{CS} \)-satisfiable and let \( \mathcal{M}^* = (W, R, E, V) \) 
be a model and \( a \in W \) a world of the model that satisfies \( \phi \). Assume that 
\( \mathcal{M}^* \) satisfies the Strong Evidence Property. For this proof, for a state \( a \in W \), 
let \( E_a = \{*(t, \phi) \mid a \in E(t, \phi)\} \).
We know that \( R \) is serial and transitive and that \( E \) satisfies the distribution closure condition. From this, we know that there is an infinite sequence of elements of \( W \), \( \alpha = (a_i)_{i \in \mathbb{N}} \), such that \( a_0 = a \), \( i < j \Rightarrow a_i Ra_j \) \& \( E_{a_i} \subseteq E_{a_j} \).

For any \( t: \psi \), there is at most one \( j \in \mathbb{N} \), \( M^*, a_j \not| t: \psi \rightarrow \psi \). Otherwise, there are \( i < j \) s.t. \( M^*, a_i, a_j \not| t: \psi \rightarrow \psi \). Since \( M^*, a_i \not| t: \psi \rightarrow \psi \), we have \( M^*, a_i \models t: \psi \). From this, it follows that \( M^*, a_j \models \psi \), so \( M^*, a_j \models t: \psi \rightarrow \psi \) - a contradiction.

Therefore, for any finite set of term-prefixed formulas, there is an \( i \), after which for all terms \( c \) of sequence \( \alpha \) that set is factive at \( c \). More specifically, let \( \Phi \) be the set of term-prefixed subformulas of \( \phi \) and let \( b \) be a term of sequence \( \alpha \), where \( \Phi \) is factive.

Define \( M \) to be the model \((\{a, b\}, \{(a, b), (b, b)\}, E', \mathcal{V}')\), such that \( \mathcal{V}', E' \) agree with \( \mathcal{V}, E \) on \( a, b \). That is, for any \( w \in \{a, b\} \), \( t \) term, \( \psi \) formula, \( p \) propositional variable, \( w \in \mathcal{V}(p) \) if and only if \( w \in \mathcal{V}'(p) \), and \( w \in E(t, \psi) \) if and only if \( w \in E'(t, \psi) \). It is easy to see that the new model satisfies the conditions required of Fitting models for JD4_{CS}^2.\(^2\)

By induction on the structure of \( \chi \), we can show that for any \( \chi \), subformula of \( \phi \), \( M^*, b \models \chi \) iff \( M, b \models \chi \). The propositional cases are trivial; if

\(^2\)Of course, we assume here that \( a \neq b \), but this is a legitimate assumption. If we need to make this explicit, we could simply have \( W = \{(a, 0), (b, 1)\} \) and the accessibility relation, \( \mathcal{V}', A' \) behave in the same way.
\( \chi = t : \omega \), then \( M^*, b \models \chi \) iff \( M^*, b \models t : \omega \) iff \( M^*, b \models \omega \) and \( b \in \mathcal{E}(t, \omega) \) (Strong Evidence) iff \( M, b \models \omega \) and \( b \in \mathcal{E}'(t, \omega) \) iff \( M, b \models \chi \).

To prove that \( M, a \models \phi \), we will first prove that

\[ M^*, a \models \psi \iff M, a \models \psi, \]

for any \( \psi \) subformula of \( \phi \), by induction on the structure of \( \psi \). If \( \psi \) is a propositional variable, and for the propositional cases, again, this is obvious and the only interesting case is when \( \psi = t : \chi \). In this case, \( M^*, a \models t : \chi \) iff \( a \in \mathcal{E}(t, \chi) \) and \( M^*, b \models \chi \) iff \( a \in \mathcal{E}'(t, \chi) \) and \( M, b \models \chi \) iff \( M, a \models t : \chi \).

Note that we can now replace the admissible evidence function with another, say \( \mathcal{E}^m \), such that \( w \in \mathcal{E}^m(t, \psi) \) iff \( M, w \models t : \psi \). This new function will satisfy the necessary conditions to be an admissible evidence function and the new model will satisfy the same formulas as the old one in the same worlds. Therefore, from this observation and the proof of proposition 3.3.3, we can claim the following corollary, which is, in fact, what we will be using.

**Corollary 3.3.4.** A formula \( \phi \) is JD4 CS-satisfiable if and only if there is a Fitting model \( M = (W, R, \mathcal{E}, V) \) for JD4 CS that has the following properties:

- \( W \) has exactly two elements, \( a, b \);
- \( R = \{(a, b), (b, b)\} \);
• $\mathcal{M}, a \models \phi$;

• $a \in \mathcal{E}(t, \psi)$ if and only if $\mathcal{M}, a \models t : \psi$ for all $t : \psi$. (Strong Evidence Condition)

At this point it is interesting to compare and contrast the models of this section to Mkrtchyan models for JD4. The models presented in this section are as minimal as possible, which makes them similar to Mkrtchyan models: instead of focusing only on one world, use two worlds. This keeps the simple character of Mkrtchyan models, but we also have the advantage of not needing the consistent evidence condition, which is convenient when designing an algorithm to test satisfiability.

While these new models allow us to disregard the consistent evidence condition, they also give us a different view of JD4: they reveal that a set $\{t_1 : \phi_1, \ldots, t_k : \phi_k\}$ is JD4-satisfiable if and only if $\{\phi_1, \ldots, \phi_k\}$ is LP-satisfiable; a JD4 agent believes they are an LP agent and this is characteristic of the logic and separates it from JD.

**Theorem 3.3.5.** When $\mathcal{CS}$ is in P and is axiomatically appropriate and schematic, then JD4-satisfiability is in $\Sigma_2^p$.

**Proof.** The algorithm that determines JD4$_{\mathcal{CS}}$-satisfiability is similar to the ones already used to establish the same upper bound for the satisfiability
for J, J4, JT, LP, and JD, except for certain differences that stem from the specific kind of models it is based on.

Prefixed expressions are used and there will be two types of prefixes and two types of expressions. The first will be the usual $T$ or $F$ prefix and the other will be the prefix that will intuitively denote the world we are referring to; these are $a$ and $b$. So, the prefixed formulas will be of the form $w \ P \ e$, where $w \in \{a, b\}, P \in \{T, F\}$ and $e$ is either a formula of the language or a $*$-expression. $w$ will be called the world prefix and $P$ the truth prefix. If a formula (or $*$-expression) is prefixed by $w \ F$, then it will be called a negative or negatively prefixed formula (or $*$-expression) in $w$, or that the formula appears negatively in the branch for world prefix $w$.

As was mentioned previously, the algorithm will be based on a tableau construction. The propositional tableau rules will be the usual ones and they are not mentioned here. The non-propositional cases are covered by the rules of Table 3.3.

The algorithm runs in two phases, similar to the one we described for JD.

If $\phi$ is the formula which must be checked for JD$_{CS}$-satisfiability, then during the first phase, the algorithm will construct a tableau branch, starting from just $a \ T \ \phi$ and using the tableau rules to generate more prefixed formulas in a non-deterministic way. After all possible tableau derivations have been
applied, there are two possibilities for the constructed branch. It can either be propositionally closed, or it can be complete, that is, the branch is not propositionally closed and no application of a tableau rule gives a new prefixed formula. If it is propositionally closed, the input is rejected, otherwise, the second phase of the algorithm begins. Let $F = (\{a, b\}, \{(a, b), (b, b)\})$ and let $X$ be the set of $*F$-expressions $x e$, such that $x T e$ appears in the branch (equivalently, let $X_a$ be the set of star expressions prefixed with $a T$ and $X_b$ the set of star expressions prefixed with $b T$ in the branch). Confirm that no $x F e$ appears in the branch such that $x e$ a $*F$-expression and $X \vdash_{sF} x e$ (equivalently that no $*$-expression $e$ where $w F e$ appears in the branch can be derived from $X_w$). If this is indeed the case, the algorithm accepts, otherwise, it rejects.

For a fixed, complete branch, let $F$, $W$, $R$, and $X$ be as above. The proof of the correctness of the algorithm follows.
Supposing formula $\phi$ is satisfiable by $\mathcal{M} = (\{a,b\},\{(a,b),(b,b)\},\mathcal{E},\mathcal{V})$ such as the ones described in corollary 3.3.4 and starting the procedure with $a T \phi$, it is easy to see that there is a way to perform the tableau rules while producing $a$-prefixed expressions satisfied at world $a$ and $b$-prefixed expressions satisfied at world $b$. Now, if $X$ derives $w \ast (t,\psi)$, then, because of Theorem 2.2.8, immediately $w \in \mathcal{E}(t,\psi)$. Therefore, if $w F \ast (t,\psi)$ appears in the branch, $w \ast (t,\psi)$ will not be derivable from $X$. In conclusion, the algorithm accepts.

On the other hand, suppose the algorithm accepts. Consider a complete branch of the tableau that is constructed in an accepting branch of the computation tree. A model will be constructed to satisfy $\phi$. This will be $\mathcal{M} = (W,\{(a,b),(b,b)\},\mathcal{E},\mathcal{V})$. $a \in \mathcal{V}(p)$ iff $a T p$ appears in the tableau branch and similarly for $b$. $\mathcal{E} \models e$ iff $X \vdash_x e$; we know $\mathcal{E}$ is an admissible evidence function, because of theorem 2.2.8. If for $\mathcal{E}$ so defined, $\mathcal{E} \models e$, where $e$ a $\ast$-expression and $F e$ has appeared in the branch, the second phase of the algorithm would have rejected the input and the computation branch would not be accepting.

The model satisfies at $a$ all $a$-prefixed expressions and at world $b$ all $b$-prefixed expressions. This can be proven by induction on the structure of the expressions. By the above argument, this is automatically true for starred
expressions. Also, by definition of \( V \), this is true for propositional variables. Propositional cases are easy, so it remains to show this for formulas of the form \( t: \psi \).

First, for \( b \)-prefixed formulas. If \( b T t: \psi \) is in the branch, there must also be \( b T^*(t, \psi) \) and \( b T \psi \). By the I.H., these are already satisfied, so \( b \in \mathcal{E}(t, \psi) \) and \( \mathcal{M}, b \models \psi \). Therefore, \( \mathcal{M}, b \models t: \psi \). If \( b F t: \psi \) is in the branch, then the branch must also include either \( b F \psi \) or \( b F^*(t, \psi) \). In either case, the conclusion is \( \mathcal{M}, b \not\models t: \psi \).

Finally, the case of \( a \)-prefixed formulas. If \( a T t: \psi \) is in the branch, there must also be \( a T^*(t, \psi) \) and \( b T \psi \) (and \( b T t: \psi \) too, but it is not relevant here). By I.H., these are already satisfied, so \( a \in \mathcal{E}(t, \psi) \) and \( \mathcal{M}, b \models \psi \). Therefore, \( \mathcal{M}, a \models t: \psi \). If \( a F t: \psi \) is in the branch, then the branch must also include either \( b F \psi \) or \( a F^*(t, \psi) \). In either case, the conclusion is \( \mathcal{M}, b \not\models t: \psi \).

This completes the correctness proof of the algorithm.

The first phase of the algorithm runs in nondeterministic polynomial time, while the second verifies a condition known to be in \text{coNP} (Theorem 3.1.1). Therefore, the algorithm establishes that JD4-satisfiability is in \( \Sigma^p_2 \). \( \Box \)
3.4 A Lower Complexity Bound

In this chapter so far we have seen that for (single-agent) justification logics $J$, $JT$, $J4$, and $LP$, the satisfiability problem is in $\Sigma^p_2$ for a schematic constant specification ([Kuz00]) and for $JD$, $JD4$, the satisfiability problem is in $\Sigma^p_2$ for an axiomatically appropriate and schematic constant specification ([Kuz08a, Ach14b]). It is normal to ask at this point whether we can do better. As this section demonstrates, the answer is “no”. We present lower bounds for the complexity of single-agent Justification Logic.

Milnikel has proven ([Mil07]) that $J4$-satisfiability is $\Sigma^p_2$-hard for an axiomatically appropriate and schematic constant specification and that $LP$-satisfiability is $\Sigma^p_2$-hard for an axiomatically appropriate, (schematic) and schematically injective constant specification. Then, Buss and Kuznets gave a general lower bound in [BK12], proving that for all the above logics, satisfiability is $\Sigma^p_2$-hard for an axiomatically appropriate, (schematic,) and schematically injective constant specification.

We present a general lower bound, which applies to all the single-agent justification logics we have presented, for an axiomatically appropriate and schematic constant specification. Furthermore, it holds for all multi-agent justification logics we will present in the following chapters as well. The
proof we give is from [Ach15a].

Of course, a trivial lower bound holds for all logics considered:

**Proposition 3.4.1.** For any $\mathcal{CS}$, satisfiability for any of $J_{\mathcal{CS}}$, $JD_{\mathcal{CS}}$, $JT_{\mathcal{CS}}$, $J_4_{\mathcal{CS}}$, $JD_4_{\mathcal{CS}}$, $LP_{\mathcal{CS}}$ is NP-hard.

This holds because any propositional formula is valid for classical propositional logic iff it is valid for any of these logics.

The main result of this section can be found in Theorem 3.4.2. We give the theorem first and then its proof.

**Theorem 3.4.2.** If $J$ has an axiomatically appropriate and schematic constant specification, then $J$-satisfiability is $\Sigma^p_2$-hard.

The idea behind the reduction we use to prove Theorem 3.4.2 is very similar to Milnikel’s proof of $\Pi^p_2$-completeness for $J_4$-provability [Mil07] (which also worked for $J$-provability). Both Milnikel’s and this reduction are from $QBF_2$. The main difference has to do with the way each reduction transforms (or not) the $QBF$ formula. Milnikel uses the propositional part of the $QBF$ formula as it is and he introduces existential nondeterministic choices on a satisfiability-testing procedure (think of Kuznets’ algorithm as described above) using formulas of the form $x : p \lor y : \neg p$ and universal nondeterministic choices using formulas of the form $x : p \land y : \neg p$ and term $[x + y]$ in the
final term, forcing a universal choice between $x$ and $y$ during the $\ast$-calculus testing.

This approach works well for $J$ and $J_4$, but it fails in the presence of the Consistency or Factivity axiom, as $x : p \land y : \neg p$ becomes inconsistent. For the case of $LP$, he used a different approach and made use of his assumption of a schematically injective constant specification (i.e. that all constants justify at most one scheme) to construct a term $t$ to specify an intended proof of a formula of the form $\bigwedge_i(x_i : p_i \land y_i : \neg p_i) \rightarrow s : \psi$ – which is always provable, since the left part of the implication is inconsistent. In this paper we bypass the problem of the inconsistency of $x : p \land y : \neg p$ by replacing each propositional formula by two corresponding propositional variables, $[\chi]^\top$ and $[\chi]^{\perp}$ to correspond to “$\chi$ is true” and to “$\chi$ is false” respectively. Therefore, we use $x : [p]^\top \land y : [p]^{\perp}$ instead of $x : p \land y : \neg p$ and we have no inconsistent formulas. As a side-effect we need to use several extra formulas to encode the behavior of the formulas with respect to a truth-assignment – for instance, $[p]^\top \rightarrow [p \lor q]^\top$ is not a tautology, so we need a formula to assert its truth (see the definitions of $Eval_j$ below).

Buss and Kuznets in [BK12] use the same assumption as Milnikel on the constant specification to give a general lower bound by a reduction from Vertex Cover and a $\Sigma^p_2$-complete generalization of that problem. Their con-
construction has the advantage that it additionally proves an \textbf{NP}-hardness result for the reflected fragment of the logics they study, while ours does not. On the other hand we do not require a schematically injective constant specification, as, much like Milnikel’s construction for \textit{J4}, we do not need to limit a *-calculus derivation.

Lemma 3.4.3 is a simple observation on the resources (number of assumptions) used by a *-calculus derivation: if there is a derivation of *\((t, \phi)\) and \(t\) only has one appearance of term \(s\), then the derivation uses at most one premise of the form *\((s, \psi)\). In fact, this observation can be generalized to \(k\) appearances of \(s\) using at most \(k\) premises, but this is not important for the proof of Theorem 3.4.2.

**Lemma 3.4.3.** Let \(\phi\) a justification formula, \(t\) a justification term in which \(! does not appear, and \(s\) a subterm of \(t\) which appears at most once in \(t\). Let \(S_s = \{s : \phi_1, \ldots, s : \phi_k\}\) and \(S \subset rL_J\), such that \(S \cup S_s\) is consistent. Then, \(S \cup S_s \vdash t : \phi\) if and only if there is some \(1 \leq a \leq k\) such that \(S \cup \{s : \phi_a\} \vdash t : \phi\).

**Proof.** Easy, by induction on the *-calculus derivation (on \(t\)).

The proof of Theorem 3.4.2 is by reduction from \(QBF_2\).

As mentioned above, for every \(\psi_a \in \Psi\), let \([\psi_a]^\top, [\psi_a]^\bot\) be new proposi-
tional variables. As we argued earlier, we need formulas to help us evaluate the truth of variables under a certain valuation in a way that matches the truth of the original formula, \( \psi - [\psi] \rightarrow [\neg \psi] \) for instance. These kinds of formulas (prefixed by a corresponding justification term) are gathered into \( S(\phi) \). \( T^J(\phi) \) is constructed in such a way that under the formulas of \( S(\phi) \) and given a valuation \( v \)

\[
\bigwedge_{v(p_a) = \text{true}} x_a : [p_a] \land \bigwedge_{v(p_a) = \text{false}} x_a : [p_a] \land S(\phi) \vdash T^J(\phi) : [\phi]
\]

if and only if \( v \) makes \( \phi \) true. In other words, \( T^J(\phi) \) encodes the method we would use to evaluate the truth value of \( \phi \).

To construct \( T^J(\phi) \), we first need certain justification terms to encode needed operations to manipulate formulas. We will often need to work on long conjuncts like \( (\phi_1 \land \cdots \land \phi_r) \), which we can view as a string of formulas. Therefore we need operations like projections \( (proj^r_x) \), appending a formula \( (\text{append}) \), appending a formula to a hypothesis \( (\text{hypappend}) \), appending the conclusions of two implications \( (\text{appendconc}) \), and so on. We start by providing these terms.

We define terms \( proj^r_x \) (for \( x \leq r \)), \( \text{append} \), \( \text{hypappend} \), and \( \text{appendconc} \),
to be such that

\[ t : (\phi_1 \land \phi_2 \land \cdots \land \phi_r) \vdash [proj^r_x \cdot t] : \phi_x, \]

\[ t : \phi_1, s : \phi_2 \vdash [append \cdot t \cdot s] : (\phi_1 \land \phi_2), \]

\[ t : (\phi_1 \rightarrow \phi_2) \vdash [hypappend \cdot t] : (\phi_1 \rightarrow \phi_1 \land \phi_2), \text{ and} \]

\[ t : (\phi_1 \rightarrow \phi_2), s : (\phi_1 \rightarrow \phi_3) \vdash [appendconc \cdot t \cdot s] : (\phi_1 \rightarrow \phi_2 \land \phi_3), \]

\text{append, hypappend, and appendconc can simply be any terms such that}

\[ \vdash append : (\phi_1 \rightarrow (\phi_2 \rightarrow \phi_1 \land \phi_2)), \]

\[ \vdash hypappend : ((\phi_1 \rightarrow \phi_2) \rightarrow (\phi_1 \rightarrow \phi_1 \land \phi_2)), \text{ and} \]

\[ \vdash appendconc : ((\phi_1 \rightarrow \phi_2) \rightarrow ((\phi_1 \rightarrow \phi_3) \rightarrow (\phi_1 \rightarrow \phi_2 \land \phi_3))). \]

Such terms exist, because they justify propositional tautologies and the constant specification is schematic and axiomatically appropriate (see Lemma 2.2.1). To define \( proj^r_x \), we need terms \( left, right, id, tran \), so that

\[ \vdash left : (\phi_1 \land \phi_2 \rightarrow \phi_1), \quad \vdash right : (\phi_1 \land \phi_2 \rightarrow \phi_2), \]

\[ \vdash id : (\phi_1 \rightarrow \phi_1), \text{ and} \]

\[ \vdash tran : ((\phi_1 \rightarrow \phi_2) \rightarrow ((\phi_2 \rightarrow \phi_3) \rightarrow (\phi_1 \rightarrow \phi_3))). \]

Again, such terms exist, because they justify propositional tautologies. Then, \( proj^1_1 = id \); for \( r > 1 \), \( proj^r_r = right \); and for \( l < r \), \( proj^{r+1}_l = [trans \cdot left \cdot proj^r_l] \).
Now we provide the formulas that will help us with evaluating the truth of the propositional part of the QBF formula under a valuation. These were axioms provided by the constant specification in Milnikeł’s proof [Mil07], but as we argued before, we need the following formulas in our case. Let Ψ = {ψ₁, ..., ψₗ} be an ordering of all subformulas of ψ, such that if a < b, then |ψₐ| ≤ |ψ₉|³. Furthermore, ρ = |{χ ∈ Ψ | |χ| = 1}| and for every 1 ≤ j ≤ l,

if ψₗ = ¬γ, then \(Eval_j = truth_j : ([γ]ᵀ → [ψ_j]⁺) ∧ truth_j : ([γ]⁻ → [ψ_j]ᵀ)\);

if ψₗ = γ ∨ δ, then

\[
\]

if ψₗ = γ ∧ δ, then

\[
\]

if ψₗ = γ → δ, then

\[
\]

\(³\) assume a |·|, such that |pⱼ| = 1 and if γ is a proper subformula of δ, then |γ| < |δ|
We now construct term $T^J(\phi)$. To do this we first construct terms $T^a$, where $1 \leq a \leq l$. Given a valuation $v$ in the form $x_1 : [p_1]^{v_1}, \ldots, x_k : [p_k]^{v_k}$, $T^1$ through $T^k$ simply gather these formulas in one large conjunct (or string). Then for $k + 1 \leq a \leq l$, $T^a$ evaluates the truth of $\psi_a$, resulting in either $[\psi]^\top$ or $[\psi]^\bot$ and appending the result at the end of the conjunct.

Let $T^1 = x_1$ and for every $1 \leq a \leq k$, $T^a = [\text{append } \cdot T^{a-1} \cdot x_a]$. It is not hard to see that for $v_1, \ldots, v_k \in \{\top, \bot\}$,

$$x_1 : [p_1]^{v_1}, \ldots, x_k : [p_k]^{v_k} \vdash T^k : ([p_1]^{v_1} \land \cdots \land [p_k]^{v_k}). \quad (3.1)$$

If $\psi_a = \neg \psi_b$, then

$$T^a = \text{hypappend } \cdot [\text{trans } \cdot \text{proj}^{a-1}_b \cdot \text{truth}_a] \cdot T^{a-1}$$

and if $\psi_a = \psi_b \circ \psi_c$, then

$$T^a = \text{hypappend } \cdot [\text{trans } \cdot [\text{appendconc } \cdot \text{proj}^{a-1}_b \cdot \text{proj}^{a-1}_c \cdot \text{truth}_a] \cdot T^{a-1}.$$ 

Let

$$S(\phi) = \bigwedge_{\rho<j\leq l} \text{Eval}_j$$

and given a truth valuation $v$, let

$$S^v(\phi) = \bigwedge_{v(p_j)=\text{true}} x_j : [p_j]^\top \land \bigwedge_{v(p_j)=\text{false}} x_j : [p_j]^\bot \land \bigwedge_{\rho<j\leq l} \text{Eval}_j.$$
By induction on \(a\), for every truth assignment \(v\),

\[
S^v(\phi) \vdash T^a : ([\psi_1]^{v_1} \land \cdots \land [\psi_a]^{v_a}),
\]

where if \(\psi_b\) is true under \(v\), then \(v_b = \top\) and \(v_b = \bot\) otherwise. The cases where \(a \leq k\) are easy to see from (3.1). For the remaining cases it is enough to demonstrate that

if \(\psi_a = \neg \psi_j\), then \(S(\phi) \vdash [\text{trans} \cdot \text{proj}_{j-1} \cdot \text{truth}_a \cdot T^{a-1}] : [\psi_a]^{v_a}\) and

if \(\psi_a = \psi_b \circ \psi_c\), then

\[
S(\phi) \vdash [\text{trans} \cdot [\text{appendconc} \cdot \text{proj}_{b-1} \cdot \text{proj}_{c-1}] \cdot \text{truth}_a \cdot T^{a-1}] : [\psi_a]^{v_a},
\]

which is not hard to see by the way we designed each term.

Finally, let \(T^J(\phi) = [\text{right} \cdot T]\). We can now prove Lemma 3.4.4:

**Lemma 3.4.4.** \(T^J(\phi), S(\phi)\) are computable in polynomial time with respect to \(|\phi|\). \(\phi\) is true under truth assignment \(v\) if and only if

\[
\bigwedge_{v(p_a) = \text{true}} x_a : [p_a]^\top \land \bigwedge_{v(p_a) = \text{false}} x_a : [p_a]^\bot \land S(\phi) \vdash T^J(\phi) : [\phi]^\top.
\]

**Proof.** From the above construction we can see that if \(\phi\) is true under \(v\) then \(S^v(\phi) \vdash T^J(\phi) : [\phi]^\top\). On the other hand, if \(S^v(\phi) \vdash T^J(\phi) : [\phi]^\top\), then \(S^v(\phi) \vdash \bullet \bullet ([\text{right} \cdot T], [\phi]^\top)\), which in turn gives \((S^v(\phi))^{\#} \vdash [\phi]^\top\) (the terms do not include the operator \(!\) and thus the right side of a \(\bullet\)-derivation is a
derivation in propositional logic). If \( \phi \) is not true under \( v \), then let \( v' \) be the valuation, such that \( v'([\psi]^\top) = \text{true} \) iff \( \psi \) is true under \( v \) and \( v'([\psi]_\perp) = \text{true} \) iff \( \psi \) is false under \( v \). Then all of \( (S_v^\phi)^\# \) is true under \( v' \) and \( [\phi]^\top \) is not, therefore \( (S_v^\phi)^\# \not\vdash [\phi]^\top \), so \( S_v^\phi \not\models T^J(\phi) : [\phi]^\top \).

**Corollary 3.4.5.** The QBF formula \( \exists p_1, \ldots, p_k \forall p_{k+1}, \ldots, p_{k+l} \phi \) is true if and only if the following formula is \( J \)-satisfiable:

\[
\bigwedge_{j=1}^k (x_j : [p_j]^\top \lor x_j : [p_j]_\perp) \land \bigwedge_{j=k+1}^l (x_j : [p_j]^\top \land x_j : [p_j]_\perp) \land S(\neg \phi) \land \neg T^J(\neg \phi) : [\neg \phi]^\top.
\]

**Proof.** If

\[
\bigwedge_{j=1}^k (x_j : [p_j]^\top \lor x_j : [p_j]_\perp) \land \bigwedge_{j=k+1}^l (x_j : [p_j]^\top \land x_j : [p_j]_\perp) \land S(\neg \phi) \land \neg T^J(\neg \phi) : [\neg \phi]^\top
\]

is not satisfiable, then

\[
\bigwedge_{j=1}^k (x_j : [p_j]^\top \lor x_j : [p_j]_\perp) \land \bigwedge_{j=k+1}^l (x_j : [p_j]^\top \land x_j : [p_j]_\perp) \land S(\neg \phi) \vdash T^J(\neg \phi) : [\neg \phi]^\top,
\]

and then for every choice \( c_1 : \{1, \ldots, k\} \rightarrow \{\top, \perp\} \),

\[
\bigwedge_{j=1}^k (x_j : [p_j]^{c_1(j)}) \land \bigwedge_{j=k+1}^l (x_j : [p_j]^\top \land x_j : [p_j]_\perp) \land S(\neg \phi) \vdash T^J(\neg \phi) : [\neg \phi]^\top,
\]

and then since every variable from \( x_1, \ldots, x_{k+l} \) appears at most once in \( T^J \) and \( T^J \) does not include \(!\), by Lemma 3.4.3 there is some choice \( c_2 : \)
\[ \{1, \ldots , l\} \longrightarrow \{\top , \bot\} \] such that
\[ \bigwedge_{j=1}^{k} (x_j : [p_j]^{c_1(j)}) \land \bigwedge_{j=k+1}^{l} (x_j : [p_j]^{c_2(j)}) \land S(\neg \phi) \vdash T^J(\neg \phi)[\neg \phi]^\top. \]

Therefore, for every assignment of truth-values on \( p_1, \ldots , p_k \) there truth-values for \( p_{k+1}, \ldots , p_{l+k} \) that make \( \phi \) false.

On the other hand, if
\[ \bigwedge_{j=1}^{k} (x_j : [p_j]^\top \lor x_j : [p_j]^\bot) \land \bigwedge_{j=k+1}^{l} (x_j : [p_j]^\top \land x_j : [p_j]^\bot) \land S(\neg \phi) \land \neg T^J(\neg \phi)[\neg \phi]^\top \]
is satisfiable, then there is some choice \( c_1 : \{1, \ldots , k\} \longrightarrow \{\top , \bot\} \), such that
\[ \bigwedge_{j=1}^{k} (x_j : [p_j]^{c_1(j)}) \land \bigwedge_{j=k+1}^{l} (x_j : [p_j]^\top \land x_j : [p_j]^\bot) \land S(\neg \phi) \land \neg T^J(\neg \phi)[\neg \phi]^\top \]
is satisfiable, and then since every variable from \( x_1, \ldots , x_{k+l} \) appears at most once in \( T^J \), for every choice \( c_2 : \{1, \ldots , l\} \longrightarrow \{\top , \bot\} \),
\[ \bigwedge_{j=1}^{k} (x_j : [p_j]^{c_1(j)}) \land \bigwedge_{j=k+1}^{l} (x_j : [p_j]^{c_2(j)}) \land S(\neg \phi) \not\vdash T^J(\neg \phi)[\neg \phi]^\top. \]

Therefore, there is some truth assignment on \( p_1, \ldots , p_k \) such that every truth assignment on \( p_{k+1}, \ldots , p_{l+k} \) makes \( \phi \) true. \( \square \)

Theorem 3.4.2 is then a direct corollary of the above.
Figure 3.1: The complexity of single-agent Justification and Modal Logic
Chapter 4

Justification Logic and Modal Logic with Multiple Agents

We present the multi-agent justification and modal logics we examine in this thesis. Then we give certain immediate upper bounds for the complexity of Multi-Agent Justification Logic. For these we can either use known techniques, or expected ones. We first reiterate the $*$-calculus results in the multi-agent context. Then we handle the cases where there is no agent with the Consistency Axiom. Finally, we prove a general, probably expected, upper bound based on a small model theorem.

4.1 Multi-Agent Justification Logic

The first multi-agent justification logics were defined in [Yav08] by Yavorskaya, who presented two-agent variations of $\text{LP}$. These logics feature interactions between the two agents' justifications: for $\text{LP}_\uparrow$, for instance, every justifica-
tion for agent 1 can be converted to a justification for agent 2 for the same fact, while in LP$_1$ agent 2 is aware of agent 1’s justifications. In this section we present the general family of multi-agent justification logics with interactions that we study.

Other multi-agent justification logics have been introduced as well (for example, see [BKS11, Ren11]). They present a different approach. Our analysis is static: we concern ourselves with interactions among the justifications of the agents and not with actual agent interactions.

For every $n \in \mathbb{N}$, the justification terms of the language $L^n_J$ include constants $c_1, c_2, c_3, \ldots$ and variables $x_1, x_2, x_3, \ldots$ and if $t_1$ and $t_2$ are terms, then the following are also terms: $[t_1 + t_2], [t_1 \cdot t_2], !t_1$. The set of terms will be referred to as $Tm$ – notice that we use one set of terms for all agents. We use a set $Prop$ of propositional variables, which will usually be $p_1, p_2, \ldots$.

Formulas of the language $L^n_J$ include all propositional variables and if $\phi, \psi$ are formulas, $1 \leq i \leq n$ (i.e. $i$ is an agent) and $t$ is a term, then the following are also formulas of $L^n_J$: $\bot, \neg a, a \lor b, a \land b, \phi \rightarrow \psi$, and $t : i \phi$. We may consider certain propositional connectives as constructed from the others as needed.

For example: $\neg a := a \rightarrow \bot$, $a \lor b := \neg a \rightarrow b$, and $a \land b := \neg(\neg a \lor \neg b)$. The operators $\cdot, +$ and $!$ are explained by the following axioms. Intuitively, $\cdot$ applies a justification for a statement $A \rightarrow B$ to a justification for $A$ and gives
a justification for $B$. Using $+$ we can combine two justifications and have a justification for anything that can be justified by any of the two initial terms – much like the concatenation of two proofs. Finally, $!$ is a unary operator called the proof checker. Given a justification $t$ for $\phi$, $!t$ justifies the fact that $t$ is a justification for $\phi$. A multi-agent justification logic is denoted by the quadruple $J = (N, \subset, \sqsubseteq, F)_{CS}$, where $N \neq \emptyset$, so that $N = \{1, 2 \ldots, n\}$ is the set of agents (therefore $n = |N|$), $\subset, \sqsubseteq$ are binary relations on $N$, and for every agent $i$, $F(i)$ is a (single-agent) justification logic. In this paper we assume that $F : N \rightarrow \{J, JD, JT, J4, JD4, LP\}$. $CS$ is called a constant specification and is explained later in this section. We also define $i \supset j$ iff $j \subset i$ and $i \shortrightarrow j$ iff $j \sqsubseteq i$.

For an agent $i$, $F(i)$ specifies the logic agent $i$ is based on – and as we observe below, this mainly affects the reliability of the agent’s justifications. As for the interactions, for agents $i, j$, if $i \supset j$, then the justifications of $i$ are also accepted as such by agent $j$. If $i \shortrightarrow j$, then agent $j$ is aware of agent $i$’s justifications – but awareness does not necessarily imply acceptance. In the latter case, we also say that $j$ can verify the justifications of $i$. In the original logic, $LP$, where justifications were proofs, if $t$ is a proof of $\phi$, then the proof of that fact comes from verifying $t$ for $\phi$ and is denoted as $!t$. In the current system we expect that since $j$ is aware that $t \vdash_1 \phi$, $j$ should have
a justification for the fact and this justification simply comes from verifying that $t$ is a justification of $\phi$ for $i$.

$J$ uses modus ponens and all agents share the following common axioms.

**Propositional Axioms:** Finitely many schemes of classical propositional logic;

**Application:** $s : (\phi \rightarrow \psi) \rightarrow (t : \phi \rightarrow (s \cdot t) : \psi)$;

**Concatenation:** $s : \phi \rightarrow (s + t) : \phi$, $s : \phi \rightarrow (t + s) : \phi$.

For every agent $i$, we also include a set of axioms that depend upon the logic $i$ is based on (i.e. $F(i)$). So, if $F(i)$ has Factivity, then we include the Factivity axiom for $i$, if $F(i)$ has Consistency, then we include Consistency for $i$, while if $F(i)$ has Positive Introspection, then we include Positive Introspection for agent $i$.

**Factivity:** for every agent $i$, such that $F(i) \in \{JT, LP\}$, $t : i \phi \rightarrow \phi$;

**Consistency:** for every agent $i$, such that $F(i) \in \{JD, JD4\}$, $t : i \bot \rightarrow \bot$;

**Positive Introspection:** for every $i$, such that $F(i) \in \{J4, LP\}$, $t : i \phi \rightarrow !t : i$.

The following, interaction axioms depend upon the binary relations $\subset$ and $\subseteq$. 
CHAPTER 4. MULTIPLE AGENTS

Conversion: for every $i \supset j$, $t : i \phi \rightarrow t : j \phi$;

Verification: for every $i \models j$, $t : i \phi \rightarrow !t : j t : i \phi$.

In this context Positive introspection is a special case of Verification. From now on we assume that for every agent $i$, $F(i) \in \{J, JD, JT\}$, so agent $i$ has positive introspection iff $i \models i$.

To complete the description of justification logic $(N, \subset, \models, F)_{CS}$, a constant specification $CS$ is needed: A constant specification for $(N, \subset, \models, F)$ is a set of formulas of the form $c : i A$, where $c$ a justification constant, $i$ an agent, and $A$ an axiom of the logic from the ones above. We say that axiom $A$ is justified by a constant $c$ for agent $i$ when $c : i A \in CS$. Then we can introduce our final axiom,

Axiom Necessitation: $t : i \phi$, where either $t : i \phi \in CS$ or $\phi = s : j \psi$ an instance of Axiom Necessitation and $t = !s$.

Axiom Necessitation will be called AN for short. In this paper we will be making the assumption that the constant specifications are axiomatically appropriate: each axiom is justified by at least one constant; and schematic: every constant justifies only a certain number of schemes from the ones above (as a result, if $c$ justifies $A$ for $i$ and $B$ results from $A$ and substitution, then $c$ justifies $B$ for $i$).
Proposition 2.2.1’s version for this multi-agent family can be stated for any agent and using the same proof: for an axiomatically appropriate constant specification $\mathcal{CS}$, if $\phi_1, \ldots, \phi_k \vdash \phi$, then for any $1 \leq i \leq n$ and terms $t_1, \ldots, t_k$, there is some term $t$ such that $t_1 : i \phi_1, \ldots, t_k : i \phi_k \vdash t : i \phi$.

We fix a certain logic $J = (N, \subset, F)_{\mathcal{CS}}$ and we make certain (reasonable) assumptions. We assume that $\subset$ is transitive and $(N, \subset)$ has no cycles. This is reasonable, because if $i \subset j \subset k$, then $t : k \phi \rightarrow t : i \phi$ is a theorem and if $i \subset j \subset i$, then agent $i$ and $j$ have exactly the same justifications for the same formulas, since $t : i \phi \leftrightarrow t : j \phi$ is a theorem and thus the agents are indistinguishable – there may be some effect of these $\subset$-paths and cycles on the logic, depending on the constant specification, but not in any way that interests us here. We also assume that if $F(i) = \text{JD}$ (resp. $F(i) = \text{JT}$) and $i \subset j$, then $F(j) = \text{JD}$ (resp. $F(j) = \text{JT}$), that if $i \subset j$ and $k \subset i$, then $j \subset k$, and that if $j \subset i$ and $F(i) = \text{JT}$, then $i \subset j$ – notice that if $j \subset i$, $F(i) = \text{JT}$, and $c : j (t : i \phi \rightarrow \phi) \in \mathcal{CS}$, then $t : i \phi \rightarrow (c : t) : j \phi$ is a theorem of $J$. Making these assumptions simplifies the system and often the notation, as they make the behavior and interactions among the agents clearer, while it is not hard to adjust the analysis in their absence. However, the assumptions we will be making about $\mathcal{CS}$ (that it is schematic, axiomatically appropriate, in $P$, or a combination of these) are mostly required.
4.1.1 Semantics

We introduce Fitting models for this multi-agent context. We use similar notation as with the single-agent logics; the main difference comes up due to the agent interactions.

Definition 4.1.1. A Fitting model $\mathcal{M}$ for $J = (N, C, \subset, F)_{CS}$ is a quadruple $(W, (R_i)_{i=1}^n, (\mathcal{E}_i)_{i=1}^n, \mathcal{V})$, where $W \neq \emptyset$ is a set, for every $1 \leq i \leq n$, $R_i \subseteq W^2$ is a binary relation on $W$,

$$\mathcal{V} : Prop \rightarrow 2^W$$

and for every $1 \leq i \leq n$,

$$\mathcal{E}_i : (Tm \times L^n_J) \rightarrow 2^W.$$

$W$ is called the universe of $\mathcal{M}$ and its elements are the worlds or states of the model. $\mathcal{V}$ assigns a subset of $W$ to each propositional variable, $p$, and $\mathcal{E}_i$ assigns a subset of $W$ to each pair of a justification term and a formula. $(\mathcal{E}_i)_{i=1}^n$ will often be seen and referred to as

$$\mathcal{E} : N \times Tm \times L^n_J \rightarrow 2^W$$

and $\mathcal{E}$ is called an admissible evidence function. Additionally, $\mathcal{E}$ and $(R_i)_{i=1}^n$ must satisfy the following conditions:
Application closure: for any $1 \leq i \leq n$, formulas $\phi, \psi$, and justification terms $t, s$,

$$\mathcal{E}_i(s, \phi \rightarrow \psi) \cap \mathcal{E}_i(t, \phi) \subseteq \mathcal{E}_i(s \cdot t, \psi).$$

Sum closure: for any $1 \leq i \leq n$, formula $\phi$, and justification terms $t, s$,

$$\mathcal{E}_i(t, \phi) \cup \mathcal{E}_i(s, \phi) \subseteq \mathcal{E}_i(t + s, \phi).$$

AN-closure: for any instance of AN, $t :_i \phi$, $\mathcal{E}_i(t, \phi) = W$.

Verification Closure: If $i \sqsubseteq j$, then $\mathcal{E}_j(t, \phi) \subseteq \mathcal{E}_i(t, t :_j \phi)$

Conversion Closure: If $i \subset j$, then $\mathcal{E}_j(t, \phi) \subseteq \mathcal{E}_i(t, \phi)$

Distribution: for any formula $\phi$, justification term $t$, $j \sqsubseteq i$ and $a, b \in W$,

if $a R_i b$ and $a \in \mathcal{E}_i(t, \phi)$, then $b \in \mathcal{E}_i(t, \phi)$.

- If $F(i) = JT$, then $R_i$ must be reflexive.

- If $F(i) = JD$, then $R_i$ must be serial ($\forall a \in W \exists b \in W a R_i b$).

- If $i \sqsubseteq j$, then for any $a, b, c \in W$, if $a R_i b R_j c$, we also have $a R_j c$.\(^1\)

- For any $i \subset j$, $R_i \subseteq R_j$.

Truth in the model is defined in the following way, given a state $a$:

\(^1\)Thus, if $i$ has positive introspection (i.e. $i \sqsubseteq i$), then $R_i$ is transitive.
• $\mathcal{M}, a \not\models \bot$ and if $p$ is a propositional variable, then $\mathcal{M}, a \models p$ iff $a \in \mathcal{V}(p)$.

• $\mathcal{M}, a \models \phi \rightarrow \psi$ if and only if $\mathcal{M}, a \models \psi$, or $\mathcal{M}, a \not\models \phi$.

• $\mathcal{M}, a \models t \cdot i \phi$ if and only if $a \in E_i(t, \phi)$ and $\mathcal{M}, b \models \phi$ for all $aR_ib$.

Like before, a formula $\phi$ is called satisfiable if there are some $\mathcal{M}, a \models \phi$; we then say that $\mathcal{M}$ satisfies $\phi$ in $a$. A pair $(W, (R_i)_{i=1}^n)$ as above is a frame for $(N, \subset, \prec, F)_{CS}$. We say that $\mathcal{M}$ has the Strong Evidence Property when $\mathcal{M}, a \models t \cdot i \phi$ iff $a \in E_i(t, \phi)$.

Like for the single-agent cases we now define Mkrtychev models for $(N, \subset, \prec, F)_{CS}$.

**Definition 4.1.2.** A Mkrtychev model $\mathcal{M}$ for $J = (N, \subset, \prec, F)_{CS}$ is a pair $((E_i)_{i=1}^n, \mathcal{V})$, where

$$
\mathcal{V} : \text{Prop} \rightarrow \{\text{true, false}\}
$$

and for every $i \in N$,

$$
E_i : (Tm \times L^n_J) \rightarrow \{\text{true, false}\}.
$$

$\mathcal{V}$ assigns a truth value to each propositional variable, $p$, and $E_i$ characterizes the set of admissible justification terms for a given formula. $(E_i)_{i=1}^n$ will often
be seen and referred to as

$$E : N \times Tm \times L^n_j \rightarrow \{true, false\}$$

and $E$ is called an admissible evidence function. Additionally, $E$ must satisfy the following conditions:

Application closure: for any $i \in N$, formulas $\phi, \psi$, and terms $t, s$,

$$if E_i(s, \phi \rightarrow \psi) = true and E_i(t, \phi), then E_i(s \cdot t, \psi);$$

Sum closure: for any $1 \leq i \leq n$, formula $\phi$, and justification terms $t, s$,

$$if E_i(t, \phi) = true or E_i(s, \phi) = true, then E_i(t + s, \phi) = true;$$

AN-closure: for any instance of AN, $t : i \phi$, $E_i(t, \phi) = true$;

Verification Closure: If $i \subset j$ and $E_j(t, \phi) = true$, then $E_i(|t, t : j \phi) = true$;

Conversion Closure: If $i \subset j$ and $E_j(t, \phi) = true$, then $E_i(t, \phi) = true$.

Truth in the model is defined in the following way:

- $M \not\models \bot$;

- if $p$ is a propositional variable, then $M \models p$ iff $a \in V(p)$;

- $M \models \phi \rightarrow \psi$ if and only if $M \models \psi$, or $M \not\models \phi$;
• $\mathcal{M} \models t : i \phi$ if and only if $a \in \mathcal{E}_i(t, \phi)$ and $\mathcal{M} \models \phi$ in case $F(i) \in \{\text{JT}, \text{LP}\}$ – i.e. in case $F(i)$ has Factivity;

• $\mathcal{M} \models t : i \phi$ if and only if $a \in \mathcal{E}_i(t, \phi)$ in case $F(i) \not\in \{\text{JT}, \text{LP}\}$.

Proposition 4.1.1. $J$ with an axiomatically appropriate constant specification is sound and complete with respect to its Fitting models; it is also complete with respect to Fitting models with the Strong Evidence property.

Proof. Soundness is left to the reader. Completeness will be proven using a canonical model construction. Let $W$ be the set of all maximal consistent subsets of $L^n_J$. We know that $W$ is not empty, because $J$ is consistent. For $\Gamma \in W$ and $1 \leq i \leq n$, let $\Gamma^{\#i} = \{\phi \in L^n_J \mid \exists t \in Tm \ t : i \phi \in \Gamma\}$. For any $1 \leq i \leq n$, $R_i$ is a binary relation on $W$, such that $\Gamma R_i \Delta$ if and only if $\Gamma^{\#i} \subseteq \Delta$. Also, for $1 \leq i \leq n$, let $\mathcal{E}_i(t, \phi) = \{\Gamma \in W \mid t : i \phi \in \Gamma\}$. Finally, $\mathcal{V} : \text{Prop} \rightarrow \mathcal{P}(W)$ is such that $\mathcal{V}(p) = \{\Gamma \in W \mid p \in \Gamma\}$. The canonical model is $\mathcal{M} = (W, (R_i)_{i=1}^n, (\mathcal{E}_i)_{i=1}^n, \mathcal{V})$.

Define the relation between worlds of the canonical models and formulas of $L^n_J$, $\models$, as in the definition of models.

Lemma 4.1.2 (Truth Lemma). For all $\Gamma \in W$, $\phi \in L^n_J$, $\mathcal{M}, \Gamma \models \phi \iff \phi \in \Gamma$. 

CHAPTER 4. MULTIPLE AGENTS

Proof. By induction on the structure of $\phi$. The cases for $\phi = p$, a propositional variable, $\bot$, or $\psi_1 \rightarrow \psi_2$, are immediate from the definition of $V$ and $\models$.

If $\phi = t;_i \psi$, then $M, \Gamma \models t;_i \psi \Rightarrow \Gamma \in \mathcal{E}_i(t, \psi) \Leftrightarrow t;_i \psi \in \Gamma$;

$t;_i \psi \in \Gamma \Rightarrow \forall \Delta \in W \ (\Gamma R_i \Delta \rightarrow \psi \in \Delta) \Rightarrow \forall \Delta \in W \ (\Gamma R_i \Delta \rightarrow \Delta \models \psi)$;

finally, $\Gamma \in \mathcal{E}_i(t, \psi)$ and $\forall \Delta \in W \ (\Gamma R_i \Delta \rightarrow \Delta \models \psi) \Rightarrow M, \Gamma \models t;_i \psi$, which completes the proof. \qed

The canonical model is, indeed, a model for $J$. To establish this, we must show that the conditions expected from $R_1, R_2$ and $\mathcal{E}_1, \mathcal{E}_2$ are satisfied. First, the admissible evidence function conditions:

Application closure: If $\Gamma \in \mathcal{E}_i(s, \phi \rightarrow \psi) \cap \mathcal{E}_i(t, \phi)$, then $s;_i (\phi \rightarrow \psi), t;_i \phi \in \Gamma$. Because of the application axiom, $[s \cdot t] ;_i \psi \in \Gamma$, so $\Gamma \in \mathcal{E}_i(s \cdot t, \psi)$.

Sum closure: If $\Gamma \in \mathcal{E}_i(t, \phi)$, then $t;_i \phi \in \Gamma$, so, by the Concatenation axiom, $[s + t] ;_i \phi, [t + s] ;_i \phi \in \Gamma$, therefore, $\Gamma \in \mathcal{E}_i(t + s, \phi) \cap \mathcal{E}_i(s + t, \phi)$.

CS closure: Any $\Gamma \in W$ includes all instances of AN, so this is satisfied.

Verification closure: If $\Gamma \in \mathcal{E}_i(t, \phi)$, then $t;_i \phi \in \Gamma$. If $i \rightarrow j$ then $!t;_j t;_i \phi \in \Gamma$, therefore, $\Gamma \in \mathcal{E}_j(!t, t;_i \phi)$. 
CHAPTER 4. MULTIPLE AGENTS

Conversion closure: If $\Gamma \in E_i(t, \phi)$, then $t :_i \phi \in \Gamma$. If $i \supset j$ then $t :_j \phi \in \Gamma$, therefore, $\Gamma \in E_j(t, \phi)$.

Distribution: If $\Gamma R_j \Delta$ and $\Gamma \in E_i(t, \phi)$, then $t :_i \phi \in \Gamma$. If $i \rightarrow j$ then $!t : _j \phi \in \Gamma$, thus $t :_i \phi \in \Gamma \#^j \subseteq \Delta$, concluding that $\Delta \in E_i(t, \phi)$.

To complete the proof, we prove that $(R_i)_{i=1}^n$ satisfy the required conditions:

If $F(i) = JT$, then $R_i$ is reflexive. For this, we just need that if $\Gamma \in W$, then $\Gamma \#^i \subseteq \Gamma$. If $\phi \in \Gamma \#^i$, then there is some justification term, $t$, for which $t :_i \phi \in \Gamma$. Because of the Fitting Factivity axiom, $\neg \phi \not\in \Gamma$, since $\{t :_i \phi, \neg \phi\}$ is inconsistent. Therefore, as $\Gamma$ is maximal consistent, $\phi \in \Gamma$.

If $F(i) = JD$, then $R_i$ is serial. To establish this, we just need to show that $\Gamma \#^i$ is consistent. If it is not, then there are formulas $\phi_1, \ldots, \phi_k \in \Gamma \#^i$ s.t. $\phi_1, \ldots, \phi_k \vdash \bot$. This means that there are $t_1 :_i \phi_1, \ldots t_k :_i \phi_k \in \Gamma$, s.t. $t_1 :_i \phi_1, \ldots t_k :_i \phi_k \vdash t :_i \bot$ (by Proposition 2.2.1), which is a contradiction.

If $i \leftarrow j$ and $\Gamma R_j \Delta R_i E$, then $\Gamma R_i E$. If $t :_i \phi \in \Gamma$ then $!t :_j t :_i \phi \in \Gamma$, so $t :_i \phi \in \Gamma \#^j$. If $\Gamma R_j \Delta$, then $t :_i \phi \in \Delta$. So, $\Gamma \#^i \subseteq \Delta \#^i$ and if $\Delta R_i E$, 


then $\Gamma R_i E$.

If $i \subset j$, then $R_i \subseteq R_j$. If $i \subset j$ then for any $\Gamma \in W$, $\Gamma^{\#i} \subseteq \Gamma^{\#j}$, i.e. $R_j \subseteq R_i$.

Finally, notice that the canonical model has the Strong Evidence Property:

if $\Gamma \in \mathcal{E}_i(t, \phi)$ then $t :_i \phi \in \Gamma$ and by the Truth Lemma, $\Gamma \models t :_i \phi$.

$\Box$

**Proposition 4.1.3.** $J$ is sound and complete with respect to its Mkrtchyan models.

Notice that completeness for M-models does not have any requirements of the constant specification. We skip the proof of soundness and completeness for M-models since it is very similar to the single-agent case and the F-model case – for the F-models we had slightly different conditions because of the agent interactions.

### 4.1.2 Some Examples and Graphical Representations

We present some examples of logics, each of interest for different reasons.

In the following, for $i \in [16]$, let $J_i = (N_i, \subseteq_i, \models_i, F_i)_{i \in \mathcal{S}_i}$, where $N_i =$

\[2^{2}\text{In fact, it is not hard to demonstrate how to construct from a model } M = (W_i, (R_i)_{i=1}^n, \mathcal{E}, \mathcal{V}) \text{ a model } M' = (W_i, (R_i)_{i=1}^n, \mathcal{E'}, \mathcal{V}) \text{ which has the Strong Evidence Property and for every } w \in W \text{ and } \phi \in \mathcal{L}_i, \ M, w \models \phi \text{ iff } M', w \models \phi: \text{ just define } \mathcal{E}_i(t, \phi) = \{w \in W \mid M, w \models t :_i \phi\}.\]
\{1,2,\ldots,n_i\}$ and $\mathcal{CS}_i$ is an axiomatically appropriate constant specification for logic $\mathcal{J}_i$. It is convenient to also assume that $\mathcal{CS}_i \in \mathcal{P}$. For some of these logics we provide graphical representations where each agent is represented by a node marked with a letter from a, j, t, and d. A generic agent may be marked with a; if $F(i) = J$, then the node representing $i$ is marked with j; if $F(i) = JD$, then the node representing $i$ is marked with d; and if $F(i) = JT$, then the node representing $i$ is marked with t. The interactions are represented by arrows: if $i \subset j$, then there is an arrow from $i$ to $j$ marked with a c; if $i \prec j$, then there is an arrow from $i$ to $j$ marked with a v. Since the constant specification, as long as it is schematic, axiomatically appropriate and in $\mathcal{P}$ (which we always assume), does not affect the complexity of the logic, it is not represented in the graphical representation.

$\mathcal{J}_1$ through $\mathcal{J}_5$: The five logics defined in [Yav08]\(^3\); $n_1 = n_2 = n_3 = n_4 = n_5 = 2$, for $i = 1,2$, $F_1(i) = F_2(i) = F_3(i) = F_4(i) = F_5(i) = \mathcal{LP}$; for $\mathcal{J}_1 = \mathcal{LP}^2$, $\rightarrow_1 = \{(1,1),(2,2)\}$ and $\supset_1 = \emptyset$; for $\mathcal{J}_2 = \mathcal{LP}^2_\uparrow$, $\rightarrow_2 = \{(1,1),(2,2)\}$ and $\supset_2 = \{(1,2)\}$; for $\mathcal{J}_3 = \mathcal{LP}^2_\downarrow$, $\rightarrow_3 = \{(1,1),(2,2),(1,2)\}$, $\supset_3 = \emptyset$; for $\mathcal{J}_4 = \mathcal{LP}^2_{\uparrow\downarrow}$, $\rightarrow_4 = \{(1,1),(2,2)\}$ and

\(^3\)The reader may notice some differences between the logics defined in [Yav08] and the ones defined here. We took a somewhat different approach when defining the logics in this paper. With this in mind, it is easy to see that logics $\mathcal{J}_1$ through $\mathcal{J}_5$ correspond to Yavorskaya’s logics and for simplicity we treat them as the same.
\(\succeq_4 = \{(1, 2), (2, 1)\}\); for \(\mathcal{J}_5 = \mathbb{LP}^2\), \(\succeq_5 = \{(1, 1), (2, 2), (1, 2), (2, 1)\}\) and \(\succeq_5 = \emptyset\).

\(\mathcal{J}_6\): Let \(\mathcal{J}_6\) be some logic, where \(\succeq_6, \succeq_6 \subseteq \{(i, i) \in N_6^2\}\). That is, there are no interactions between the agents. In a way, \(\mathcal{J}_6\) is the generic justification logic of multiple agents when we have no interactions.

\(\mathcal{J}_7\): Let \(n_7 = 3, F_7(2) = F_7(3) = \text{JD}, F_7(1) = \text{JT}, \succeq_7 = \{(1, 2), (2, 3)\}\) and \(\succeq_7 = \{(3, 1)\}\). Instead of interpreting 1, 2, 3 as three different agents, we can think of them as three different degrees of belief of some agent. 1 can be thought the agent’s knowledge, so if the agent knows \(\phi\) with justification \(t\) (i.e. \(t :_1 \phi\) is true), then it may be considered reasonable to assume that the agent also believes \(\phi\) with degree 2 and thus with degree 3. On the other hand, if we want the agent to have some sort of positive introspection and be aware of their own beliefs, it may be reasonable to assume that that awareness is knowledge and thus \(t :_3 \phi \rightarrow !t :_1 t :_3 \phi\) is true. An interesting phenomenon here is that if the agent knows a fact, then the agent knows they believe the fact, but does not necessarily know this belief is knowledge, i.e. \(t :_1 \phi \rightarrow !t :_1 t :_1 \phi\) is valid, but \(t :_1 \phi \rightarrow !t :_1 t :_1 \phi\) is not. A countermodel for \(t :_1 \phi \rightarrow !t :_1 t :_1 \phi\) is: \((W, (R_i)_{i \in [3]}, \mathcal{E}, \mathcal{V})\) is the following: \(W = \)
\{a, b, c\}, \ R_2, R_3 = \{(a, b), (b, b), (c, b)\}, \ R_1 = R_2 \cup \{(a, a), (b, c), (c, c)\}, \\
\mathcal{V}(p) = \{a, b\} and assume that if \(p \neq q\), \(\mathcal{V}(q) = \emptyset\), while \(\mathcal{E}(s, \psi) = W\) always.

Figure 4.1: The logic \(J_7\) graphically.

\(J_8\) and \(J_9\): For \(J_8\) and \(J_9\), let \(n_8 = n_9 = 2\), \(F_8(1) = F_8(2) = F_9(1) = \\
F_9(2) = JD, \mathcal{D}_{8=9} = \mathcal{D}_{8} = \emptyset\) and \(\mathcal{D}_{8} = \{(1, 2), (2, 1)\}\).

Figure 4.2: The logic \(J_8\) or \(J_9\), depending on how we mark the edges.

\(J_{10}\): Let \(J_{10}\) be any logic, such that \(D_{10} = \emptyset\). For any such logic, we can simply use Mkrtchyan models and, as we will see, it makes it easier to build an algorithm to solve satisfiability.

\(J_{11}\): Let \(n_{11} = 3\), for \(i \in N\), \(F_{11}(i) = JT\), and \(\mathcal{D}_{11} \subseteq \{(1, 2), (1, 3)\}\).

\(J_{12}\): Let \(n_{12} = 2\), \(F_{12}(1) = F_{12}(2) = JD\), \(\mathcal{D}_{12} = \{(1, 1)\}\), and \(\mathcal{D}_{12} = \{(1, 2)\}\).
$J_{12}$: Let $n_{12} = 2$, for $i = 1, 2$, $F_{12}(i) = JD$, $\gamma_{12} = \{(1, 2)\}$, and $\cup_{12} = \{(1, 2)\}$.

$J_{13}$: Let $n_{13} = 2$, for $i = 1, 2$, $F_{13}(i) = JD$, $\gamma_{13} = \{(1, 2)\}$, and $\cup_{13} = \{(1, 2)\}$.

$J_{14}$: Let $n_{14} = 3$, for $i = 1, 2, 3$, $F_{14}(i) = JD$, $\gamma_{14} = \emptyset$, and $\cup_{14} = \{(1, 2), (1, 3)\}$.

$J_{15}$: Let $n_{15} = 3$, for $i = 1, 2, 3$, $F_{15}(i) = JD$ and $\gamma_{15} \cup \cup_{15} = \{(1, 2), (1, 3)\}$.
CHAPTER 4. MULTIPLE AGENTS

Figure 4.6: The logic $J_{15}$ can be any of these two, or $J_{14}$.

$J_{16}$: Let $n_{16} = 3$, for $i = 1, 2, 3$, $F_{16}(i) = JD$, $\rightarrow_{16} = \{(1, 1)\}$, and $\supset_{16} = \{(1, 2), (1, 3)\}$.

Figure 4.7: The logic $J_{16}$ graphically.

For $J_{11}$, $J_{14}$ and $J_{15}$, the beliefs of one agent affect the beliefs of the other two. We will see how this affects the complexity of each of these logics. In fact, we will present some result about the complexity of each of these logics.

4.1.3 The $*$-calculus.

We present the $*$-calculus for $(N, \subset, \subset, F)_{CS}$. As we have seen in previous chapters, the $*$-calculi for the single-agent justification logics are an invalu-
able tool in the study of the complexity of these logics. This concept and results were adapted to the two-agent setting in [Ach14c] and to the general multi-agent setting in [Ach15b]. Although the calculi have significant similarities to the ones of the single-agent justification logics, there are differences that mainly have to do with the interactions between the agents. Their overall principle and form remain the same, though.

If \( t \) is a term, \( \phi \) is a formula, and \( 1 \leq i \leq n \), then \( *_i(t, \phi) \) is a \(*\)-expression.

Given a frame \( F = (W, (R_i)_{i=1}^n) \) for \( J \), the \(*_F\)-calculus for \( J \) on the frame \( F \) is a calculus on \(*\)-expressions prefixed by worlds from \( W \) (\(*_F\)-expressions from now on) with the axioms and rules that are shown in Table 4.1.

We can repeat similar arguments and definitions as for the single-agent \(*\)-calculus: if \( E \) is an admissible evidence function of \( M \), we define \( M, w \models *_i(t, \phi) \) iff \( E \models w *_i(t, \phi) \) iff \( w \in E_i(t, \phi) \). Notice that the calculus rules correspond to the closure conditions of the admissible evidence functions. In fact, because of this, given a frame \( F = (W, (R_i)_{i=1}^n) \) and a set \( S \) of \(*_F\)-expressions, the function \( E \) such that \( E \models e \iff S \vdash *_F e \) is an admissible evidence function. Furthermore, \( E \) is minimal and unique: if some admissible evidence function \( E' \) is such that for every \( e \in S \), \( E' \models e \), then for every \(*_F\)-expression \( e \), \( E \models e \Rightarrow E' \models e \). Therefore, given a frame \( F = (W, (R_i)_{i=1}^n) \) and two set \( X, Y \) of \(*_F\)-expression there is an admissible evidence function
\*CS(\mathcal{F}) \textbf{Axioms:} w \ast_i (t, \phi), \text{ where } t \vdash_i \phi \text{ an instance of AN}

\*\text{App}(\mathcal{F}): \quad \frac{w \ast_i (s, \phi \rightarrow \psi) \quad w \ast_i (t, \phi)}{w \ast_i (s \cdot t, \psi)}

\*\text{Sum}(\mathcal{F}): \quad \frac{w \ast_i (t, \phi) \quad w \ast_i (s, \phi)}{w \ast_i (s + t, \phi)}

\ast \subseteq (\mathcal{F}): \text{ For any } i \rightarrow j, \quad \frac{w \ast_i (t, \phi)}{w \ast_j (t \cdot t, \phi)}

\ast \supset (\mathcal{F}): \text{ For any } i \supset j, \quad \frac{w \ast_i (t, \phi)}{w \ast_j (t, \phi)}

\ast \subseteq \text{ Dis}(\mathcal{F}): \text{ For any } i \rightarrow j \text{ and } (a, b) \in R_j, \quad \frac{a \ast_i (t, \phi)}{b \ast_i (t, \phi)}

where \mathcal{F} = (W, (R_i)_{i=1}^n) \text{ and for every } 1 \leq i \leq n.

Table 4.1: The \*\mathcal{F}-calculus for \mathcal{J}.

\mathcal{E} \text{ on } \mathcal{F} \text{ such that for every } e \in X, \mathcal{E} \models e \text{ and for every } e \in Y, \mathcal{E} \not\models e, \text{ if and only if there is no } e \in Y \text{ such that } X \vdash e. \text{ When } X = \emptyset, \text{ as for the single-agent logics, this yields the following proposition:}

\textbf{Proposition 4.1.4.} For any frame } \mathcal{F}, \text{ state } w, \mathcal{J} \vdash t :_i \phi \iff \vdash \mathcal{F} w \ast_i (t, \phi).
4.2 The Complexity of the $\ast$-calculus on Multiple Agents

Proposition 4.2.1. If $\mathcal{CS}$ is schematic and in $P$, then the following problem is in $NP$:

Given a finite frame $\mathcal{F} = (W, (R_i)_{i=1}^n)$, a finite set $S$ of $\ast$-expressions prefixed by worlds from $W$, a formula $t : i \phi$, and a $w \in W$, is it the case that $S \vdash_{\ast, \mathcal{F}} w \ast_i (t, \phi)$?

Proof. The proof of this proposition is very similar to the one for Proposition 3.1.1, which can be found in [Kuz08a]. The shape of a $\ast$-calculus derivation is mostly given away by the term $t$. So, we can use $t$ to extract the general shape of the derivation – the term keeps track of the applications of all rules besides $\ast \subseteq$ and $\ast \sqsubseteq$ Dis. We can then plug in to the leaves of the derivation either axioms of the calculus or members of $S$ and unify ($\mathcal{CS}$ is schematic, so the derivation may include schemes) trying to reach the root – at the same time we need to keep track of the states of the frames and the agents for which each step of the derivation is possible (i.e. for step $(s, \psi)$ we keep track for which $(j, v)$ we can derive $v \ast_j (s, \psi)$). What is different here is the additional assignment of an agent and state set $R(s) \subseteq N \times W$ to each node $s$ of the derivation tree, which does not change things a lot. $R(s)$ will satisfy the
following closure conditions: if \((j, v) \in R(s)\) and \(j \supset j'\), then \((j', v) \in R(s)\),
while if \((j, v) \in R(s), j \subset j',\) and \(vR_j v'\) then \((j, v') \in R(s)\). Then for every
node \(s\), \(R(s)\) will be the smallest set such that these closure conditions are
satisfied, as well as the ones described in the following algorithm.

The algorithm to decide the derivability of \(w \ast_i (t, \phi)\) from \(S\) is the following:

- Nondeterministically construct a rooted tree with subterms of \(t\), as
  nodes, such that \(t\) is the root and the following conditions are met.
  Node \(s\) can be the parent of \(s_1\) or of both \(s_1, s_2\) as long as there is a
  rule \(\frac{v \ast_j (s_1, \phi_1)}{v \ast_j (s, \phi_3)}\) or \(\frac{v \ast_j (s_1, \phi_1)}{v \ast_j (s, \phi_3)}\), respectively, of the \(\ast\)-calculus
  and \(s_1, s_2\) are proper subterms of \(s\). This results in a subtree of term \(t\)
  (when \(t\) is seen as a tree) and the nondeterministic choices correspond
to two cases: wherever \(+\) appears in \(t\) (and we have to choose a version
  of the \(\ast\)Sum rule) and whenever we encounter a term \(s\), for which there
  is some \(s \vdash_j \psi \in S\), so we can choose to not break down \(s\) any further –
  there is a third possibility, when we encounter some subterm \(s =! \cdots \! c,\)
  where \(c\) a constant; \(s\) can be part of an axiom of the calculus, or a
  result of consecutive applications of \(\ast \subset\) on an axiom, but this is a
  choice without consequences.

- Nondeterministically assign to each leaf, \(l\), either
– some formula $\psi$ such that there is some $v \ast_j (l, \psi) \in S$ – in which case $(j, v) \in R(l)$ – or

– as long as $l$ is of the form $\underbrace{\cdot \cdot \cdot c}_k$, where $c$ a constant, $k \geq 0$, then we can also assign some $\underbrace{\cdot \cdot \cdot c}_{k-1} : i_1 \cdot \cdot \cdot c : i_2 \cdot \cdot \cdot A$, where $A$ an axiom scheme and $c : i_1 A \in \mathcal{CS}$ – in which case if $k = 0$, then for every $c : i_1 A \in \mathcal{CS}$ and $v \in W$, $(j, v) \in R(l)$ and if $k > 0$, then $R(l) = N \times W$.

• If for some node $s$, all its children, say $s_1, s_2$ (resp. $s_1$) have been assigned some scheme or formula $P_1, P_2$ (resp. $P_1$), assign to $s$ some scheme or formula $P$, such that $P_1, P_2$ can be unified to $P'_1, P'_2$ so that $\frac{v_{s_1}(s_1, P'_1)}{v_{s_2}(s_2, P'_2)}$ is a rule (resp. $\frac{v_{s_1}(s_1, P'_1)}{v_{s_2}(s, P)}$ is a rule) in the $\ast_{\mathcal{CS}}(V, C)$-calculus (but not $\ast \subset$ or $\ast \subseteq$ Dis). Then, if the rule was $\ast$App or $\ast$Sum, then $R(s) = R(s_1) \cap R(s_2)$; if the rule was $\ast \subseteq$, then for every $(j, v) \in R(s_1)$ and $j \subseteq j'$, $(j', v) \in R(s)$. Apply this step until the root of the tree has been assigned some scheme or formula.

• Unify $\phi$ with the formula assigned to $t$ and verify that $(i, w) \in R(t)$

If some step is impossible, the algorithm rejects. Otherwise, it accepts. Using efficient representations of schemes using DAGs and Robinson's unification algorithm, the algorithm runs in polynomial time (with respect to
We can see that as the tree is constructed, if \( s \) is assigned scheme \( P \) and set \( R(s) \ni (j,v) \), then the construction effectively describes a valid derivation of any expression of the form \( v \ast_j (s,\psi) \), where \( \psi \) an instance of \( P \). Therefore, if the algorithm accepts, there exists a valid \( \ast^E \)-calculus derivation of \( w \ast_i (t,\phi) \). On the other hand if there is some \( \ast^E \)-calculus derivation for \( w \ast_i (t,\phi) \) from \( S \), then the algorithm in the first two steps can essentially describe this derivation by producing the derivation tree and the formulas/schemes by which the derivation starts. Therefore, the algorithm accepts if and only if there is a \( \ast^E \)-calculus derivation for \( w \ast_i (t,\phi) \) from \( S \). See [Kru06] and [Kuz08a] for a more detailed analysis.

### 4.3 Tableaux through Mkrtychev Models

If we do not have agents with consistent beliefs (there is no \( i \in N \) such that \( F(i) \in \{JD, JD4\} \)), then the study of the complexity of Multi-Agent Justification Logic with interactions becomes trivial. We can simply use the same methods as Kuznets in [Kuz00] and proofs and give tableaux through Mkrtychev models, which work nicely in this context as well. This is exactly what we do in this section.

**Theorem 4.3.1.** Satisfiability for \((N,\subset, \vdash, F)_{CS}\), where \( JD, JD4 \notin F[N] \), with an efficiently decidable, schematic \( CS \) is in \( \Sigma^p_2 \).
Proof. We just use the same tableaux and proofs as for Theorem 3.2.1 – see Table 4.2. Of course, what is not immediately obvious is that we use a different \(*\)-calculus which accounts for the interactions and the multiple agents.

If \(F(i) = J\) or \(J4\):
\[
\begin{array}{c}
T t \vdash_i \psi \\
T \ast_i (t, \psi)
\end{array}
\quad
\begin{array}{c}
F t \vdash_i \psi \\
F \ast_i (t, \psi)
\end{array}
\]

If \(F(i) = JT\) or \(LP\):
\[
\begin{array}{c}
T \psi \\
T \ast_i (t, \psi)
\end{array}
\quad
\begin{array}{c}
F t \vdash_i \psi \\
F \ast_i (t, \psi) \mid F \psi
\end{array}
\]

Table 4.2: Tableau rules for \((N, \subset, \rightarrow, F)_{CS}\) when no agent has the Consistency axiom.

4.4 A Small Model Theorem

In this section we prove a small model theorem, that is, that a satisfiable formula is satisfiable in a (Fitting) model of at most an exponential number of states. We then use it to prove a general upper bound for Multi-Agent Justification Logic. We start with a lemma.

Lemma 4.4.1. If \(\phi \in L_n\) is consistent and \(\Phi\) is a maximally consistent set of subformulas of \(\phi\), then \(\Phi \cup \{\neg t :_i \phi \in L_n^i \mid \Phi \not\vdash t :_i \phi\}\) is consistent.

Proof. We define \(\mathcal{M} = (E, V)\), where \(E(t, \phi) = true\) and \(V(p) = true\) iff for every M-model \(\mathcal{M'} = (E', V') \models \Phi\) with the strong evidence property, \(E'(t, \phi) = true\) and \(V'(p) = true\). It is not hard to see that \(E\) satisfies all
closure conditions, so it is an admissible evidence function and for every $\Phi \not\vdash t : i \psi$, there is some $\mathcal{M}' \models \Phi, \neg t : i \psi$, so $\mathcal{M} \not\models t : i \psi$. What remains is to prove that $M \models \Phi$, by proving that for every $\psi$ subformula of $\phi$, $M \models \psi$ iff $\psi \in \Phi$. By induction on $\psi$: this is obvious for propositional variables and connectives, while we have already argued about the case of $t : i \psi$. \qed

**Corollary 4.4.2.** If $\phi$ is $J$-satisfiable, then $\phi$ is satisfiable by a Fitting model for $J$ of at most $2^{|\phi|}$ states which has the strong evidence property.

**Proof.** Let $\mathcal{M} = (W, (R_i)_{i=1}^n, (E_i)_{i=1}^n, \mathcal{V})$ be the canonical model from the proof of Proposition 4.1.1 and $\mathcal{M}_f = (W^f, (R_i^f)_{i=1}^n, (E_i^f)_{i=1}^n, \mathcal{V}^f)$, where $W^f$ is the set of all maximally consistent sets of subformulas of $\phi$ and for all $1 \leq i \leq n$, $X, Y \in W_f$,

- $XR_i^f Y$ iff there is some $X' \in W$ such that $X \subseteq X'$ and for every $Y' \in W$, if $Y \subseteq Y'$ then $X'R_iY'$;

- $X \in E_i^f(t, \psi)$ iff for every $X' \in W$ such that $X \subseteq X' \in W$, $X' \in E_i(t, \psi)$ and

- $X \in V^f(p)$ iff for every $X' \in W$ such that $X \subseteq X' \in W$, $X' \in V(p)$.

Then, define $\mathcal{M}_f, X \models \psi$ in the usual way as for models. Notice that since the elements of $W_f$ are maximally consistent w.r.t. subformulas of $\phi$, for
every set $\Psi$ of subformulas of $\phi$, $\Psi \subseteq X \in W_f \iff$ for every $\Gamma \in W$ s.t. $X \subseteq \Gamma \in W$, $\Psi \subseteq \Gamma \iff$ there is some $\Gamma \in W$ s.t. $X \subseteq \Gamma \in W$ and $\Psi \subseteq \Gamma$. Also, for every $X \in W_f$, let $\overline{X} = \{\psi \in L^n_j | X \vdash \psi\}$. Then, for every $X, Y \in W_f$, propositional variable $p$ and $t : i \psi$ subformula of $\phi$, $X \in \mathcal{E}_f^i(t, \psi)$ iff $t : i \psi \in X$, and $X \in \mathcal{V}_f^i(p)$ iff $p \in X$.

Furthermore, $XR_i^fY$ iff $X^#_{-i} \subseteq Y$: if $XR_i^fY$, then there is some $X' \in W$ such that $X \subseteq X'$ and for every $Y' \in W$ for which $Y \subseteq Y'$, $X'R_iY'$. Then, for every $Y' \in W$ for which $Y \subseteq Y'$, $X^#_{-i} \subseteq (X')^#_{-i} \subseteq Y'$, so $X^#_{-i} \subseteq Y$. On the other hand, if $X^#_{-i} \subseteq Y$, then let $G = \overline{X} \cup \{\neg t : j \psi \in L^n_j | t : j \psi / \in \overline{X}\}$. Then $G$ is consistent by Lemma 4.4.1 and can be expanded to a maximally consistent $X' \in W$. $(X')^#_{-i} = \overline{X^#_{-i}} \subseteq \overline{Y} \subseteq Y'$ for every $Y' \in W$ for which $Y \subseteq Y'$. Thus, $XR_i^fY$.

It is not hard to follow the proof of Lemma 4.1.2 to prove that for every subformula $\psi$ of $\phi$ and $X \in W_f$, $X \models \psi$ iff $\psi \in X$ and then continue by following the proof of Proposition 4.1.1 to complete this one.

\[ \square \]

The number of nondeterministic choices made by the algorithm in the proof of proposition 4.2.1 is bounded by $|t| + |S'|$, where $S' = \{s'(s, \psi) | \exists w \ s_j(s, \psi) \in S\}$. Therefore, if there is some formula $\psi$ such that $t : i \phi$ is a
subformula of ψ and for every \(*_j(s, ψ') ∈ S'\), \(s \vdash_j ψ'\) is a subformula of ψ, then
\[ 2|ψ| ≥ |t| + |S'| \]
and therefore we can simulate all nondeterministic choices in time \(2^{O(|ψ|)}\). Thus the algorithm can be turned into a deterministic one running in time \(2^{O(|φ|)} \cdot O(|W|^2)\). This observation, the fact that a satisfiable \(φ\) can be satisfied by a model of at most \(2^{O(φ)}\) states (Corollary 4.4.2) and Propositions 4.1.4 and 4.2.1 give Corollary 4.4.3.

**Corollary 4.4.3.** 1. If \(CS\) is schematic and in \(P\), then deciding for \(t \vdash_i φ\) that \(J \vdash t \vdash_i φ\) is in \(NP\).

2. If \(CS\) is axiomatically appropriate, in \(P\), and axiomatically appropriate, then the satisfiability problem for \(J\) is in \(NEXP\).

**Proof.** 1 is a direct consequence of Propositions 4.1.4 and 4.2.1. For 2, to decide satisfiability of \(φ\), nondeterministically guess a frame \(F = (W, (R_i)_{i=1}^n)\) of at most \(2^{O(φ)}\) states, state \(s\), a propositional valuation \(V\) restricted on the propositional variables that appear in \(φ\) and set of \(*_F\)-expressions \(X\), where if \(w \quad *_i (t, ψ) \in X\) then \(w \in W\) and \(t \vdash_i ψ\) a subformula of \(φ\). Then confirm that for any admissible evidence function \(E\) such that \(e \in X\) iff \(E \models e\), \((W, (R_i)_{i=1}^n, E, V), s \models φ\) (in time at most \(2^{O(φ)}\)) and that there is actually such an \(E\) (the nondeterministic choices in the algorithm for Proposition 4.4.3 come from \(t\), so they can be simulated in at most time \(2^{O(φ)}\)). □
Figure 4.8: First results for multi-agent Justification Logic
Chapter 5

The Complexity of Interacting Agents: Conversion

5.1 Multi-Modal Logic

The language of Multi-Modal Logic for \( n \) modalities (agents), \( L^n_M \), is defined in the following way, where \( 1 \leq i \leq n \):

\[
\phi ::= p \mid \neg \phi \mid \bot \mid \neg \bot \mid \phi \land \phi \mid \phi \lor \phi \mid \lozenge_i \phi \mid \Box_i \phi.
\]

If we only consider formulas in negation normal form (NNF), we push all negations to the propositional level, so instead of \( \neg \phi \) we would have \( \neg p \) above. \( p \) and \( \neg p \) are called literals. When we consider only formulas in \( L^1_M \), \( \Box_1 \) is often just called \( \Box \), so we identify \( L_M \) with \( L^1_M \).

We describe each logic with a triple \( (N, \subset, F) \), where \( N = \{1, 2, \ldots, |N|\} \) is nonempty, \( \subset \) a binary relation on \( N \), and for every \( i \in N \), \( F(i) \) is a modal
logic; a frame for $(N, \subseteq, F)$ is a $(W, (R_i)_{i \in N})$, where for every $i \in N$, $(W, R_i)$ a frame for $F(i)$ and for every $i \subset j$, $R_i \subset R_j$. It is reasonable to assume that $(N, \subseteq)$ has no cycles – otherwise we can collapse all modalities in the cycle to just one – and that $\subseteq$ is transitive. Furthermore, we also assume that all $F(i)$’s have frames with serial accessibility relations – otherwise there is either some $j \subseteq i$ for which $F(j)$’s frames have serial accessibility relations and $R(i)$ would inherit seriality from $R_j$, or $\Box_i \psi$ can always be true by default, which makes the situation not very interesting. Thus, we assume that $F(i) \in \{D, T, D4, S5\}$.\(^{234}\)

Specifically, a Kripke model for a multimodal logic (a logic based on language $L^n_M$) is a tuple $\mathcal{M} = (W, (R_i)_{i=1}^n, V)$, where for $1 \leq i \leq n$, $R_i \subseteq W \times W$ and for every propositional variable $p$, $\forall(p) \subseteq W$. Then, $(W, (R_i)_{i=1}^n)$ is called a frame and the $R_i$ are called accessibility relations. We define the

---

\(^{1}\)In the general case, we describe a modal logic with a quadruple $(N, \subseteq, \prec, F)$, but when $\prec = \emptyset$ as in this chapter, we omit the symbol altogether from the description; thus, $(N, \subseteq, F)$ is the same as $(N, \subseteq, \emptyset, F)$. The remaining definitions for syntax and semantics are the same, but we also need the condition that if $i \prec j$ and in frame $(W, (R_a)_{a \in N})$, $xR_i yR_j z$, then also $xR_j z$ – just like for Justification Logic.

\(^{2}\)We can consider more logics as well, but these ones are enough to make the points we need. Besides, it is not hard to extend the reasoning of this section to other logics (ex. B, S4), especially since the absence of diamonds makes the situation simpler.

\(^{3}\)Frames for D have serial accessibility relations; frames for T have reflexive accessibility relations; frames for D4 have serial and transitive accessibility relations; frames for S5 have accessibility relations that are equivalence relations (reflexive, symmetric, transitive).

\(^{4}\)Note that we do not define modal logics with verification, although it is not hard to go that extra step. The reason is that we will concern ourselves with Modal Logic mainly when the only interaction is Conversion.
truth relation $\models$ between models, worlds (elements of $W$, also called states) and formulas in the following recursive way, similarly as for unimodal logics:

$$\mathcal{M}, a \not\models \bot \text{ and } \mathcal{M}, a \models p \text{ iff } a \in \mathcal{V}(p);$$

$$\mathcal{M}, a \models \neg \phi \text{ iff } \mathcal{M}, a \not\models \phi;$$

$$\mathcal{M}, a \models \phi \land \psi \text{ iff both } \mathcal{M}, a \models \phi \text{ and } \mathcal{M}, a \models \psi;$$

$$\mathcal{M}, a \models \phi \lor \psi \text{ iff } \mathcal{M}, a \models \phi \text{ or } \mathcal{M}, a \models \psi;$$

$$\mathcal{M}, a \models \Diamond_i \phi \text{ iff there is some } b \in W \text{ such that } aR_ib \text{ and } \mathcal{M}, b \models \phi;$$

$$\mathcal{M}, a \models \Box_i \phi \text{ iff for all } b \in W \text{ such that } aR_ib \text{ it is the case that } \mathcal{M}, b \models \phi.$$

### 5.1.1 Relating Diamond-free Modal Logic and Justification Logic

We first examine the complexity of satisfiability for the diamond-free fragments of several modal logics. Diamond-free fragments of Modal Logic may seem like an odd topic, especially in a thesis focused on the complexity of Justification Logic. In this section we explain why it is not. In fact, when trying to determine the complexity of a justification logic, it makes a lot of sense to closely examine the diamond-free fragment of the corresponding modal logic.\textsuperscript{5}

\textsuperscript{5}The observations in this section first appeared in [Ach14a].
If one examines the tableaux for various justification logics, there is a certain similarity with tableaux for Modal Logic – a similarity more evident when a tableau is based on Fitting models instead of Mkrtychev models, like the ones for JD and JD4. The differences are two: when the tableau encounters $a T t : \phi$, it additionally produces $a T \ast (t, \phi)$ to account for the admissible evidence function; and when it encounters $a F t : \phi$, it may either produce $a F \ast (t, \phi)$, or $a.n F \phi$. So, unlike the modal case, where $a F \Box \phi$ produces a new prefix, the Justification Logic tableau may choose not to, instead giving a $\ast$-calculus condition. In fact, if we base our tableaux on models with the Strong Evidence Condition (see the definition of F-models), which is what we will be doing from now on, the tableau does not even need to make a choice: $a F t : \phi$ always results in $a F \ast (t, \phi)$.\footnote{This is, in fact, a standard rule, which we use in all our tableaux, preceding and following.}

Therefore, it seems that although $t : \phi$ may behave much like the modal $\Box \phi$, there is no construct corresponding to $\Diamond \phi$ in Justification Logic. If we ignore the $\ast$-calculus of the tableaux, with respect to satisfiability testing, Justification Logic seems to be behaving a lot like diamond-free Modal Logic; therefore, one would expect these to have very similar complexity with respect to their satisfiability problem. Lemma 5.1.1 makes this observation
more explicit in a general setting. It also gives a tool for automatically transferring lower bounds from (diamond-free) Modal Logic to Justification Logic.

**Lemma 5.1.1.** Let $J$ be a justification logic whose Fitting models are characterized by constraints on their frames and their admissible evidence functions, so that the total admissible evidence function is an appropriate evidence function for $J$;

- let $M$ be the modal logic characterized by the class of frames for $J$;
- let $C$ be a complexity class closed under polynomial-time reductions.

If satisfiability for the diamond-free fragment of $M$ is $C$-hard, then so is $J$-satisfiability. If $J$-satisfiability is in $C$, then so is satisfiability for the diamond-free fragment of $M$.

**Proof.** For every diamond-free modal formula $\phi$, let $\phi'$ be the result of substituting each $\Box_i$ by $x : \tau$. If $\phi$ is satisfied by $(W, (R_i)_{i \in N}, \mathcal{V})$, then $\phi'$ is satisfied by $(W, (R_i)_{i \in N}, \mathcal{E}_{tot}, \mathcal{V})$, where $\mathcal{E}_{tot}(t, \psi) = W$ for every $t, \psi$ – the total admissible evidence function. On the other hand, if $\phi'$ is satisfied by some $(W, (R_i)_{i \in N}, \mathcal{E}, \mathcal{V})$, then $\phi$ is satisfied by $(W, (R_i)_{i \in N}, \mathcal{V})$. 

Notice that Lemma 5.1.1 is general and relates to all justification logics we encounter in this thesis and possibly many more.
CHAPTER 5. THE COMPLEXITY OF CONVERSION

5.1.2 Work on Dependent Modal Logic

The complexity of the satisfiability problem for Modal Logic, and thus of its
dual, modal provability/validity, has been extensively studied. Whether one
is interested in areas of application of Modal Logic, or in the properties of
Modal Logic itself, the complexity of modal satisfiability plays an important
role.

As we have shown, Ladner has established most of what are now con-
sidered classical results on the matter ([Lad77]), determining that most of
the usual modal logics, especially ones with more than one modality are
PSPACE-hard. Therefore, it makes sense to try to find fragments of these
logics that have an easier satisfiability problem by restricting the modal el-
ements of a formula – or prove that satisfiability remains hard even in frag-
ments that seem trivial (ex. [Hal95, CR02]). In this section we present mostly
hardness results for this direction and for certain cases of multimodal logics
with modalities that affect each other. Relevant syntactic restrictions and
their effects on the complexity of various modal logics have been examined
in [Hem01] and [HSS10]. For more on modal logic and its complexity, see
[HM92, FHMV95, Spa93].

We analyze the complexity of the satisfiability problem for modal formulas
in negation normal form that have no diamonds. When testing a modal formula for satisfiability (for example, trying to construct a model for the formula by using a tableau procedure), a clear source of complexity is the occurrence of diamonds in the formula. When we try to satisfy $\Diamond \phi$, we need to assume the existence of an extra world where $\phi$ is satisfied. Furthermore, when trying to satisfy $\Diamond p_1 \land \Diamond p_2 \land \Box \phi$, we require two new worlds where $p_1 \land \phi$ and $p_2 \land \phi$ are respectively satisfied, which can potentially cause an exponential explosion to the size of the constructed model (we just doubled the number of worlds by adding constant length to the formula). There are several modal logics, but it is usually the case that in the process of satisfiability testing, as long as there are no diamonds in the formula, we are not required to add more than one world to the constructed model, which makes identifying the existence of diamonds as an important source of complexity a natural conclusion. On the other hand, when the modal logic is $D$, its models are required to have a serial accessibility relation (no sinks in the graph). Thus, when we test $\Box \phi$ for $D$-satisfiability, we require a world where $\phi$ is satisfied. In such a unimodal setting and in the absence of diamonds, we avoid an exponential explosion in the number of worlds and we can consider models with only a polynomial number of worlds.

Several authors have examined the complexity of combinations of modal
logic (ex. [MV97, Gab03, Kur07]). The most relevant to this paper work on the complexity of combinations of modal logic is by Spaan in [Spa93] and Demri in [Dem00]. In particular, Demri studied $L_1 \oplus L_2$, which is $L_1 \oplus L_2$ (see [Spa93]) with the additional axiom $\Box_2 \phi \rightarrow \Box_1 \phi$ and where $L_1, L_2$ are among $K, T, B, S4$, and $S5$. For when $L_1$ is among $K, T, B$ and $L_2$ among $S4, S5$, he establishes EXP-hardness for $L_1 \oplus L_2$-satisfiability. We consider $L_1 \oplus L_2$, where $L_1$ is a unimodal or bimodal logic (usually $D$, or $D4$). When $L_1$ is bimodal, $L_1 \oplus L_2$ is $L_1 \oplus L_2$ with the extra axioms $\Box_3 \phi \rightarrow \Box_1 \phi$ and $\Box_3 \phi \rightarrow \Box_2 \phi$.

We examine the effect on the complexity of modal satisfiability testing of restricting our input to diamond-free formulas under the requirement of seriality and in a multimodal setting with connected modalities. In particular, we initially examine four examples: $D_2 \oplus K, D_2 \oplus K4, D \oplus K4$, and $D4_2 \oplus K4$. For these logics we look at their diamond-free fragment and establish that they are PSPACE-hard and in the case of $D_2 \oplus K4$, EXP-hard. Furthermore, $D_2 \oplus K, D \oplus K4$, and $D4_2 \oplus K4$ are PSPACE-hard and $D_2 \oplus K4$ is EXP-hard even for their 1-variable fragments. Of course these results can be naturally extended to more modal logics, but we treat what we consider simple characteristic cases. For example, it is not hard to see that nothing changes when in the above multimodal logics we replace $K$ by
\( D \), or \( K4 \) by \( D4 \), as the extra axiom \( \square_3 \phi \to \Diamond_3 \phi \) (\( \square_2 \phi \to \Diamond_2 \phi \) for \( D \oplus \subseteq K4 \)) is a derived one. It is also the case that in these logics we can replace \( K4 \) by other logics with positive introspection (ex. \( S4 \), \( S5 \)) without changing much in our reasoning.

Then, we examine a general setting of a multimodal logic where we include axioms \( \Box_i \phi \to \Box_j \phi \) for some pairs \( i, j \). For this setting we determine exactly the complexity of satisfiability for the diamond-free (and 1-variable) fragment of the logic and we are able to make some interesting observations. The study of this general setting is of interest, because determining exactly when the complexity drops to tractable levels for the diamond-free fragments illuminates possibly appropriate candidates for parameterization: if the complexity of the diamond-free, 1-variable fragment of a logic drops to \( P \), then we may be able to develop algorithms for the satisfiability problem of the logic that are efficient for variables of few diamonds and propositional variables; if the complexity of that fragment does not drop, then the development of such algorithms seems unlikely (we may be able to parameterize with respect to some other parameter, though). Another argument for the interest of these fragments is the hardness results of this paper. The fact that the complexity of the diamond-free, 1-variable fragment of a logic remains high means that this logic is likely a very expressive one, even when deprived of a significant
part of its syntax.

A very relevant approach is presented in [Hem01, HSS10]. In [Hem01], Hemaspaandra determines the complexity of Modal Logic when we restrict the syntax of the formulas to use only a certain set of operators. In [HSS10], Hemaspaandra et al. consider multimodal logics and all Boolean functions. In fact, some of the cases we consider have already been studied in [HSS10]. Unlike [HSS10], we focus on multimodal logics where the modalities are not completely independent – they affect each other through axioms of the form $\Box_i \phi \rightarrow \Box_j \phi$. Furthermore in this setting we only consider diamond-free formulas, while at the same time we examine the cases where we allow only one propositional variable. As far as the results of this section are concerned, it is interesting to note that in [Hem01, HSS10] when we consider frames with serial accessibility relations, the complexity of the logics under study tends to drop, while in this paper we see that serial accessibility relations (in contrast to arbitrary, and sometimes reflexive, accessibility relations) contribute substantially to the complexity of satisfiability.

As far as this thesis is concerned, of course, the motivation we have is the relation between the diamond-free fragments of Modal Logic with Justification Logic. As we are interested in the complexity of systems of multimodal and multi-Agent justification logics, we are additionally interested in these
5.1.3 Hardness Without Diamonds

In general, if $M_1, M_2$ are single-agent modal logics, we can define $M_1^k \oplus \subseteq M_2$ as being the logic $(N, \subseteq, \rightarrow, F)$, where $N = \{1, 2, \ldots, k + 1\}$, $\subseteq = N \times \{k + 1\}$, $F(k + 1) = M_2$, if $i \leq k$ then $F(i) = M_1$, and $\rightarrow = \emptyset$ (but as we have seen, in this chapter we will omit $\rightarrow$, as it will always be empty).

In this subsection we deal with five logics: $K$, $D_2 \oplus \subseteq K$, $D_2 \oplus \subseteq K4$, $D \oplus \subseteq K4$, and $D4_2 \oplus \subseteq K4$. All except for $K$ and $D \oplus \subseteq K4$ are trimodal logics, based on language $L^3_M$, $K$ is the unimodal logic we defined in Chapter 2 (the simplest normal modal logic) based on $L^1_M$, and $D \oplus \subseteq K4$ is a bimodal logic based on $L^2_M$. Each modal logic $M$ is associated with a class of frames $C$, where $C$ is the class of frames so that $M$ is sound and complete with respect to the derived class of models is. Therefore:

$K$

is the logic associated with the class of all frames;

$D_2 \oplus \subseteq K$

is the logic associated with the class of frames $\mathcal{F} = (W, R_1, R_2, R_3)$ for which $R_1, R_2$ are serial (for every $a$ there are $b, c$ such that $aR_1b$, $aR_2c$) and $R_1 \cup R_2 \subseteq R_3$;
\( D_2 \oplus \subseteq K_4 \)

is the logic associated with the class of frames \( \mathcal{F} = (W, R_1, R_2, R_3) \) for which \( R_1, R_2 \) are serial, \( R_1 \cup R_2 \subseteq R_3 \), and \( R_3 \) is transitive;

\( D \oplus \subseteq K_4 \)

is the logic associated with the class of frames \( \mathcal{F} = (W, R_1, R_2) \) for which \( R_1 \) is serial, \( R_1 \subseteq R_2 \), and \( R_2 \) is transitive;

\( D_4_2 \oplus \subseteq K_4 \)

is the logic associated with the class of frames \( \mathcal{F} = (W, R_1, R_2, R_3) \) for which \( R_1, R_2 \) are serial, \( R_1 \cup R_2 \subseteq R_3 \) and \( R_1, R_2, R_3 \) are transitive.

**Tableau**

We present tableau rules for \( K_7 \) and for the diamond-free fragments of \( D_2 \oplus \subseteq K \) and then for the remaining three logics. The reason we present these rules is because they are useful for later proofs and because they help to give intuition regarding the way we can test for satisfiability. The ones for \( K \) are classical and were given in Chapter 2; we restate them here. Formulas used in the tableau are given a prefix, which intuitively corresponds to a state in a model we attempt to construct and is a string of natural numbers, with \( \cdot \) representing concatenation. The tableau procedure for a formula \( \phi \) starts

\^[7] These were already presented in Chapter 2, but we give another version here for clarity.
Table 5.1: Tableau rules for K. Notice that we do not use a truth prefix.

From 0 $\phi$ and applies the rules it can to produce new formulas and add them to the set of formulas we construct, called a branch. As usual, a rule of the form $\frac{a}{b\mid c}$ means that the procedure nondeterministically chooses between $a$ and $b$ to produce, i.e. a branch is closed under that application of that rule as long as it includes $b$ or $c$. If the branch has $\sigma \bot$, or both $\sigma p$ and $\sigma \neg p$, then it is called propositionally closed and the procedure rejects its input. Otherwise, if the branch contains 0 $\phi$, is closed under the rules, and is not propositionally closed, it is an accepting branch for $\phi$; the procedure accepts $\phi$ exactly when there is an accepting branch for $\phi$. The rules for K are in Table 5.1.

For the remaining logics, we are only concerned with their diamond-free fragments, so we only present rules for those to makes things simpler. The rules for $D_2 \oplus \subseteq K$ are in Table 5.2.

We sketch a proof that these rules are correct, that is, there is a model for $\phi$ iff there is an accepting branch for $\phi$. From an accepting branch for $\phi$
we construct a model for $\phi$: let $W$ be all the prefixes that have appeared in the branch,

$$R_1 = \{(w, w.1) \in W^2\} \cup \{(w, w) \in W^2 \mid w.1 \notin W\},$$

$$R_2 = \{(w, w.2) \in W^2\} \cup \{(w, w) \in W^2 \mid w.2 \notin W\},$$

$R_3 = R_1 \cup R_2$, and $V(p) = \{w \in W \mid w \ p \ \text{appears in the branch}\}$. Then, it is not hard to see that $(W, R_1, R_2, R_3)$ is indeed a frame for $D_2 \oplus \sqsubseteq K$ ($R_1, R_2 \subseteq R_3$ and they are all serial), and that for $\mathcal{M} = (W, R_1, R_2, R_3, V)$, $\mathcal{M}, 0 \models \phi$ – by proving through a straightforward induction on $\psi$ that for every $w \psi$ in the branch, $\mathcal{M}, w \models \psi$.

On the other hand, given some $\mathcal{M}, a \models \phi$, we can construct an accepting branch for $\phi$ in the following way. We map 0 to $a$ and for every $w.i$, where $i = 1, 2$, if $w$ is mapped to state $b$ of the model, then $w.i$ is mapped to some state $c$, where $bR_i c$. Then we can make sure we make appropriate nondeterministic choices when applying a rule to ensure that whenever $w \psi$ is produced and $w$ is mapped to $a$, then $\mathcal{M}, a \models \psi$: if $\psi = \phi$, then this is trivially correct; if

$$\frac{\sigma \phi \lor \psi}{\sigma \phi \mid \sigma \psi} \quad \frac{\sigma \phi \land \psi}{\sigma \phi} \quad \frac{\sigma \Box_1 \phi}{\sigma.1 \phi} \quad \frac{\sigma \Box_2 \phi}{\sigma.2 \phi} \quad \frac{\sigma \Box_3 \phi}{\sigma.1 \phi \quad \sigma.2 \phi}$$

Table 5.2: The rules for $D_2 \oplus \sqsubseteq K$
we apply the first rule on \(w \psi_1 \lor \psi_2\), then since \(\mathcal{M}, a \models \psi_1 \lor \psi_2\), it is the case that \(\mathcal{M}, a \models \psi_1\) or \(\mathcal{M}, a \models \psi_2\) and we can choose the appropriate formula to introduce to the branch; the remaining rules are trivial. Therefore, the branch can never be propositionally closed.

To come up with tableau rules for the other three logics, we can modify the above rules. The first two rules that cover the propositional cases are always the same, so we give the remaining rules for each case without proof. In the following, notice that the resulting branch may be infinite. However we can simulate such an infinite branch by a finite one: we can limit the size of the prefixes, as after a certain size it is guaranteed that there will be two prefixes that prefix the exact same set of formulas. Thus, we can either assume the procedure terminates or that it generates a full branch, depending on our needs.

The rules for the diamond-free fragment of \(D_2 \oplus \subseteq K4\) are in Table 5.3; the rules for the diamond-free fragment of \(D \oplus \subseteq K4\) are in Table 5.4; and the

\[
\begin{align*}
\frac{\sigma \Box_1 \phi}{\sigma.1 \phi} & \quad \frac{\sigma \Box_2 \phi}{\sigma.2 \phi} & \quad \frac{\sigma \Box_3 \phi}{\sigma.1 \phi} \\
& \quad \frac{\sigma.2 \phi}{\sigma.1 \Box_3 \phi} & \quad \sigma.2 \phi \\
& \quad \sigma.1 \phi & \quad \sigma.2 \phi
\end{align*}
\]

Table 5.3: Tableau rules for the diamond-free fragment of \(D_2 \oplus \subseteq K4\)
\[
\begin{align*}
\frac{\sigma \Box_1 \phi}{\sigma.1 \phi} & \quad \frac{\sigma \Box_2 \phi}{\sigma.1 \phi} \\
& \quad \frac{}{\sigma.1 \Box_2 \phi}
\end{align*}
\]

Table 5.4: Tableau rules for the diamond-free fragment of \(D \oplus_{\subseteq} K_4\)

where \(n_i(\sigma) = \sigma\) if \(\sigma = \sigma'.i\) for some \(\sigma'\) and \(n_i(\sigma) = \sigma.i\) otherwise.

Table 5.5: Tableau rules for the diamond-free fragment of \(D_{4_2} \oplus_{\subseteq} K_4\)

rules for the diamond-free fragment of \(D_{4_2} \oplus_{\subseteq} K_4\) can be found in Table 5.5.

We skip any proof of correctness for these cases, as they are similar to the previous case. The exception is the tableau procedure for \(D_{4_2} \oplus_{\subseteq} K_4\), which is a little different and for which we must give some adjustments in the constructions of the model from the accepting branch and of the accepting branch from a model. The construction of the model is similar as for the case of \(D_2 \oplus_{\subseteq} K\), only this time for \(i = \{1, 2\}\) \(R_i = \{(\sigma, n_i(\sigma)) \in W^2\} \cup \{(\sigma, \sigma) \in W^2 \mid n_i(\sigma) \notin W\}\) (notice they are transitive) and \(R_3\) the transitive closure of \(R_1 \cup R_2\). On the other hand, when constructing an accepting branch, we need to make sure that if we map \(\sigma\) to \(b\), then we map \(\sigma.i\) to some \(c\) such
that for every $\square_i \psi$, subformula of $\phi$, $c \models \square_i \psi \rightarrow \psi$. We can find such a $c$ by considering a sequence $bR_i c_1 R_i c_2 R_i \cdots$; if some $c_j \not\models \square_i \psi \rightarrow \psi$, then $c_j \models \square_i \psi$, so for every $j' > j$, $c_j \models \square_i \psi \rightarrow \psi$. Since the subformulas of $\phi$ are finite in number, we can find some large enough $j \in \mathbb{N}$ and $c = c_j$.

**Proposition 5.1.2.** The satisfiability problem for the diamond-free fragments of $D_2 \oplus \subseteq K$, of $D \oplus \subseteq K_4$, and of $D_4 \oplus \subseteq K_4$ is in PSPACE; the satisfiability problem for the diamond-free fragment of $D_2 \oplus \subseteq K_4$ is in EXP.

*Proof.* We can use the rules to prove that satisfiability of the diamond-free fragment of $D_2 \oplus \subseteq K$ is in PSPACE. In fact, we can use an alternating polynomial-time algorithm to simulate the tableau procedure and given a formula $\phi$ to construct an accepting branch for $\phi$. The algorithm uses an existential non-deterministic choice when we apply the first rule to choose which of the resulting prefixed formulas to add to the branch; it also uses a universal choice to choose between $\sigma.1$ and $\sigma.2$ for every $\sigma$ it has produced. Other than that, it applies all the tableau rules it can, until there are none left. It is not hard to construct an accepting tableau branch from an accepting run of the algorithm and vice-versa. The fact that the algorithm runs in polynomial time can be established by observing that only up to $|\phi|$ formulas can be prefixed by a specific prefix, while the nesting depth of the boxes
in the formulas (also called modal depth) strictly decreases as the length of their prefix increases.

To establish upper complexity bounds for the diamond-free fragments of the remaining logics, we can use a similar procedure, only this time it is an alternating polynomial space algorithm to simulate the tableau procedure – we do not have the same bounds on the length of the prefixes as above, but we can just keep formulas prefixed by a single prefix in memory and as we argued before this is at most $|\phi|$ formulas – of course this means we give priority to propositional rules. Furthermore we do not even need to keep the current prefix in memory, but we can just use a counter of polynomial size for the length of the prefix (an important point, because the length of a prefix can be exponential); when the counter becomes larger than $2^{|\phi|}$, then of course we can terminate. This gives an ($\text{APSPACE} = \text{EXP}$)-upper bound for the complexity of satisfiability for the diamond-free fragment of $D_2 \oplus \subseteq K_4$; to get a $\text{PSPACE}$-upper bound for the other two logics, notice that the tableau for $D \oplus \subseteq K_4$ uses only prefixes of the form $0.1^x$ and the tableau for $D_4^2 \oplus \subseteq K_4$ only subprefixes of $0.(1.2)^\omega$ and $0.(2.1)^\omega$, therefore making universal choices unnecessary.

The cases of $D \oplus \subseteq K_4$ and $D_4^2 \oplus \subseteq K_4$ are especially interesting. In [Dem00],
Demri established that $D \oplus_{\subseteq} K4$-satisfiability (and because of the following section’s results also $D4_2 \oplus_{\subseteq} K4$-satisfiability) is $\text{EXP}$-complete. Here, though, we see that the complexity of these two logics’ diamond-free (and one-variable) fragments are $\text{PSPACE}$-complete (in this subsection we establish the $\text{PSPACE}$ upper bounds, while in the next one the lower bounds), which is a drop in complexity (assuming $\text{PSPACE} \neq \text{EXP}$), but not one that makes the problem tractable (assuming $\text{P} \neq \text{PSPACE}$).

**Lower Complexity Bounds for Diamond-free Fragments**

We give hardness results for the logics presented in the previous subsection – except for $K$. In [CR02], the authors prove that the variable-free fragment of $K$ remains $\text{PSPACE}$-hard. We make use of that result here and prove the same for the diamond-free, 1-variable fragment of $D_2 \oplus_{\subseteq} K$. Then we prove $\text{EXP}$-hardness for the diamond-free fragment of $D_2 \oplus_{\subseteq} K4$ and $\text{PSPACE}$-hardness for the diamond-free fragments of $D \oplus_{\subseteq} K4$ and of $D4_2 \oplus_{\subseteq} K4$, which we later improve to the same result for the diamond-free, 1-variable fragments of these logics.

**Proposition 5.1.3.** The diamond-free, 1-variable fragment of $D_2 \oplus_{\subseteq} K$ is $\text{PSPACE}$-complete.

*Proof.* The upper bound was given by Proposition 5.1.2. We give a trans-
lation from unimodal formulas to formulas of three modalities such that $\phi$ is $K$-satisfiable if and only if $\phi^{tr}$ (the result of the translation) is $D_2 \oplus \subseteq K$-satisfiable. The translation uses an extra propositional variable (not appearing in $\phi$), $q$. It is defined in the following way.

We want the tableau for $\phi^{tr}$ to simulate the tableau for $\phi$. However, $\phi$ may have diamonds, which are not allowed in $\phi^{tr}$. When the tableau for $K$ encounters a diamond, then it generates a unique prefix. Therefore, we must replace a diamond with something which will generate a unique prefix in the tableau for $D_2 \oplus \subseteq K$. This unique prefix can be generated by a unique sequence of boxes, which is provided by $dseq$:

For a formula $\phi$, let $\theta_1, \ldots, \theta_k$ be an enumeration of its subformulas which we view as distinct from each other (we can mark them if needed) and in increasing order with respect to their size (to ensure that if $\eta_1$ is a subformula of $\eta_2$, then $\eta_1$ appears first). Also, let\(^8\)

$$dseq : \{1, 2, \ldots, k\} \longrightarrow \{\Box_1, \Box_2\}^{\lceil \log k \rceil}$$

be some one-to-one mapping from those subformulas to a unique sequence of boxes. The actual mapping is not important, but an easy choice would be $dseq(x) = \Box_{x_1+1} \Box_{x_2+1} \cdots \Box_{x_{\lceil \log k \rceil}+1}$, where $bin(x) := x_1 x_2 \cdots x_{\lceil \log k \rceil}$ is the

\(^8\)Notice that if there is at least one diamond in $\phi$, then $\phi$ has at least two subformulas, thus if there are diamonds, then $\log k \geq 1$; if $k = 1$, then this discussion is meaningless: $\phi^{tr} = \phi$. 
binary representation of $x$ – so this is the one we assume. We can recursively on $i$ define $i^{tr}$:

- if $\theta_i$ is a literal, $\top$, or $\bot$, then $i^{tr} = \theta_i$;
- if $\theta_i = \theta_j \circ \theta_l$, where $\circ$ is either $\land$ or $\lor$, then $i^{tr} = j^{tr} \circ l^{tr}$;
- if $\theta_i = \Box \theta_j$, then $i^{tr} = \Box^{\lceil \log k \rceil} (j^{tr} \lor \lnot q)$;
- finally, if $\theta_i = \Diamond \theta_j$, then $i^{tr} = dseq(i) (j^{tr} \land q)$.

Then, $\phi^{tr} = k^{tr} \land q$ (as $\theta_k$ is actually $\phi$). The extra variable, $q$, is used to mark which prefixes in the $D_2 \oplus \subseteq K$-tableau correspond to prefixes in the $K$-tableau that have appeared.

For convenience assume that in the $K$-tableau for $\phi$, $\sigma \theta_i$, where $\theta_i = \Diamond \eta$ produces $\sigma.i \eta$ – which is reasonable, since for each $\sigma$ each $\theta_i$ appears at most once. Assume a complete accepting $K$-branch $b$ for $\phi$. Let $m(0) = 0$ and $m(\sigma.i) = m(\sigma).bin(i)$. Then, $b'$ is constructed in a recursive way, so that for every $\sigma' \eta, \sigma' q \in b'$, where $\eta \neq q, \lnot q$, there is some $\sigma \theta_i \in b$ such that $\sigma' = m(\sigma)$ and $\eta = i^{tr}$. When we apply the $\Box_1$- or $\Box_2$-rule from the ones we presented in Table 5.2, that is in the course of generating a prefix $m(\sigma)$ – so, from $m(\sigma) i^{tr}$, where $\theta_i = \Diamond \theta_j$, we eventually generate $m(\sigma.i) j^{tr}$ and $m(\sigma.i) q$ (and some auxiliary boxed formulas in-between); when we apply
the $\Box_3$-rule, then this started from some $m(\sigma) i^{tr}$, where $\theta_i = \Box \theta_j$, so for every $\sigma.l \theta_j \in b$, we produce $m(\sigma.l) j^{tr}$, while for $\sigma.l \theta_j \in b$ (where $l \leq k$), we produce $m(\sigma.l) \neg q$ (and auxiliary boxed formulas in-between); when we apply a propositional rule on $m(\sigma) i^{tr} \circ j^{tr}$, we just need to make the same nondeterministic choice that was made for $b$ (if applicable). Then, naturally, if $b'$ is rejecting, then that is because $m(\sigma) p, m(\sigma) \neg p \in b'$, or $m(\sigma) \bot \in b'$; but then either $\sigma p, \sigma \neg p \in b$, or $\sigma \bot \in b$, respectively.

On the other hand it is easier to give a complete accepting $K$-branch $b$ for $\phi$ given a complete accepting $D_2 \oplus \subseteq K$-branch $b'$ for $\phi^{tr}$: $b = \{\sigma \theta_i \mid m(\sigma) i^{tr} \in b'\}$. We leave the reader to verify this claim.

Notice that $\chi^{tr}$ has no diamonds and the number of propositional variables in $\chi^{tr}$ is one more than in $\chi$. Since we can assume $\chi$ is variable-free (see [CR02]), the proposition follows.

For the remaining logics we first present a reduction to show hardness for their diamond-free fragments and then we provide translations to their 1- or 2-variable fragments. We first treat the case of $D_2 \oplus \subseteq K4$.

**Lemma 5.1.4.** The diamond-free fragment of $D_2 \oplus \subseteq K4$ is EXP-complete, while the diamond-free fragments of $D \oplus \subseteq K4$ and of $D4_2 \oplus \subseteq K4$ are PSPACE-complete.
Proof. The upper bounds were given by Proposition 5.1.2. The proof resembles the one in [FL79] and is by reduction from a generic \textsc{APSpace} problem given as the alternating Turing machine of two tapes (input and working tape) which uses polynomial space to decide it. Let the machine be $(Q, \Sigma, \delta, s)$, where $Q$ the set of states, $\Sigma$ the alphabet, $\delta$ the transition relation and $s$ the initial state. Let $Q = U \cup E$, where $E$ and $U$ are distinct, $E$ the set of existential and $U$ the set of universal states and assume that the machine only has two choices at every step of the computation, provided by two transition functions, $\delta_1, \delta_2$: when the transition functions are given state $q \in Q$, and symbols $a, b \in \Sigma$ for tape 1 and 2 respectively, for $i = 1, 2$, $\delta_i(q, a, b) = (q', c, j_1, j_2) \in Q \times \Sigma \times \{0, -1, 1\}^2$, where $q'$ the new state, $c$ the symbol to replace $b$ in tape 2, and $j_1, j_2$ the respective moves for each tape, where 0 indicates no move, $-1$ a move to the left, and 1 a move to the right. Furthermore, let $x = x_1 x_2 \cdots x_{|x|}$ be the input, where for every $i \in \{1, 2, \ldots, |x|\}$, $x_i \in \Sigma$. Since the Turing machine uses polynomial space, there is a polynomial $p$, such that the working tape only uses cells 1 to $p(|x|)$ for an input $x$. For the input tape, we only need cells 0 through $|x| + 1$ (we may assume additional symbols to indicate the beginning and end of the input), because the head does not go any further and an output tape is not needed, since we are interested only in decision problems. There-
fore, there are $Y, N \in Q$, the accepting and rejecting states respectively. Let
$r_1 = \{0, 1, 2, \ldots, |x| + 1\}$ and $r_2 = \{1, 2, \ldots, p(|x|)\}$. A configuration $c$ of the
Turing machine is called accepting if the computation of the machine that
starts from $c$ is an accepting computation.

For this reduction, a formula will be constructed that will enforce that any
model satisfying it must describe a computation by the Turing machine. Each
propositional variable will correspond to some fact about a configuration of
the machine and the following propositional variables will be used:

- $t_1[i], t_2[j]$, for every $i \in r_1, j \in r_2$; $t_1[i]$ will correspond to the head for
  the first tape pointing at cell $i$ and similarly for $t_2[j]$,

- $\sigma_1[a, i], \sigma_2[a, j]$, for every $a \in \Sigma, i \in r_1, j \in r_2$; $\sigma_1[a, i]$ will correspond
to cell $i$ in the first tape having the symbol $a$ and similarly for $\sigma_2[a, j]$
  and the second tape,

- $q[e]$, for every $e \in Q$; $q[e]$ means the machine is currently in state $e$.

For each configuration $c$ of the Turing machine there is a formula that
describes it. This formula is the conjunction of the following and from now on
it will be denoted as $\phi_c$: $q[e]$, if $e$ is the state of the machine in $c$; $t_1[i]$ and
$t_2[j]$, if the first tape’s head is on cell $i$ and the second tape’s head is on cell
$j; \sigma_1[i_1, i_1], \sigma_2[i_2, i_2]$, if $i_1 \in r_1, i_2 \in r_2$ and $a_1$ is the current symbol in cell $i_1$ of the first tape and $a_2$ is the current symbol in cell $i_2$ of the second tape.

We need the following formulas. Intuitively, a world in a model for $\phi$ corresponds to a configuration of our Turing machine. $q$ ensures there is exactly one state at every configuration; $\sigma$ that there is exactly one symbol at every position of every tape; $t$ that for each tape the head is located at exactly one position; $\sigma'$ ensures that the only symbols that can change from one configuration to the next are the ones located in a position the head points at; $ac$ ensures we never reach a rejecting state (therefore the machine accepts); $st$ starts the computation at the starting configuration of the machine; finally, $d_E, d_U$ ensure for each configuration that the next one is given by the transition relation (functions). Then, if $com = q \land \sigma \land t \land \sigma' \land ac \land d_E \land d_U$ we define $\phi = st \land com \land \Box_3 com$.

$$q = \bigvee_{e \in Q} q[e] \land \bigwedge_{e, f \in Q, \ e \neq f} \neg (q[e] \land q[f]);$$

$$\sigma = \bigwedge_{j \in \{1, 2\}, i \in r_j} \left[ \bigvee_{a \in \Sigma} \sigma_j[a, i] \land \bigwedge_{a, b \in \Sigma, a \neq b} \neg (\sigma_j[a] \land \sigma_j[b]) \right];$$

$$t = \bigwedge_{j \in \{1, 2\}, i \in r_j} \left[ \bigvee_{i, k \in r_j, i \neq k} \sigma_j[i] \land \bigwedge_{i, j \in r_j} \neg (\sigma_j[i] \land \sigma_j[k]) \right];$$

$$\sigma' = \bigwedge_{j \in \{1, 2\}, i, i' \in r_j, i \neq i', a \in \Sigma} \left[ (\sigma_j[i] \land \sigma_j[a, i']) \rightarrow \Box_1 \sigma_j[a, i'] \land \Box_2 \sigma_j[a, i'] \right];$$
CHAPTER 5. THE COMPLEXITY OF CONVERSION

\[ ac = \neg q[N] \]

\[ st = \phi_{c_0}, \text{ where } c_0 \text{ is the initial configuration of the machine;} \]

let \( \text{locconf}(e, i_1, i_2, j_1, j_2) = q[e] \land \sigma_1[i_1, j_1] \land \sigma_2[i_2, j_2] \land t_1[j_1] \land t_2[j_2] \) and \( D(e, k, l_1, l_2, m_1, m_2) = q[e] \land \sigma_2[k, l_2] \land t_1[l_1 + m_1] \land t_2[l_2 + m_2] \); then,

\[ d_E = \bigwedge_{(e, i_1, i_2) \in E \times \Sigma \times \Sigma, j_1 \in r_1, j_2 \in r_2} \left[ \text{locconf}(e, i_1, i_2, j_1, j_2) \rightarrow \Box_1 D(e_1, k_1, j_1, j_2, m_1^1, m_2^1) \lor \Box_1 D(e_2, k_2, j_1, j_2, m_1^2, m_2^2) \right] \]

where \( (e_1, k_1, m_1^1, m_2^1) = \delta_1(e, i_1, i_2), (e_2, k_2, m_1^2, m_2^2) = \delta_2(e, i_1, i_2) \);

\[ d_U = \bigwedge_{(e, i_1, i_2) \in U \times \Sigma \times \Sigma, j_1 \in r_1, j_2 \in r_2} \left[ \text{locconf}(e, i_1, i_2, j_1, j_2) \rightarrow \Box_1 D(e_1, k_1, j_1, j_2, m_1^1, m_2^1) \land \Box_2 D(e_2, k_2, j_1, j_2, m_1^2, m_2^2) \right] \]

where \( (e_1, k_1, m_1^1, m_2^1) = \delta_1(e, i_1, i_2), (e_2, k_2, m_1^2, m_2^2) = \delta_2(e, i_1, i_2) \).

The few implications that appear above are of the form \( a \land b \land \cdots \land c \rightarrow \psi \) (where \( a, b, \ldots, c \) are propositional variables) and can thus be rewritten in negation normal form: \( \neg a \lor \neg b \lor \cdots \lor \neg c \lor \psi \). The correctness of the reduction follows from the following two claims.

Claim: If for some model \( M, w \models \phi \) and for some \( u \), such that \( (u = w \text{ or } w R_3 u) \), \( u \models \phi_c \) and \( c_1, c_2 \) are the next configurations from \( c \), then if \( c \) a universal configuration, there are \( w R_3 u_1 \) and \( w R_3 u_2 \), such that \( u_1 \models \phi_{c_1} \), \( u_2 \models \phi_{c_2} \) and if \( c \) an existential configuration, there is some \( w R_3 u_1 \), such that either \( u_1 \models \phi_{c_1} \) or \( u_1 \models \phi_{c_2} \). From this claim, it immediately follows that if \( \phi \) is satisfiable, then the Turing machine accepts its input (since it never
rejects it). We prove the claim for the case of the universal configuration. Because of formulas $q, \sigma, t$, in every world $v$, such that $wR_3v$, there is exactly one $\phi_c$ satisfied. There are worlds $u_1, u_2$, (because of seriality of $R_1, R_2$) such that $wR_1u_1$ and $wR_2u_2$ and if $u_1 \models \phi_{c_3}$, $u_2 \models \phi_{c_4}$, then because of $d_U$, $c_3$ will differ from $c$ in all respects $\delta_1$ demands; furthermore, because of $\sigma'$, $c_3$ differs only in the ways $\delta_1$ (or $\delta_2$) demands and we can reason the same way for $c_4$. Therefore, $\{c_3, c_4\} = \{c_1, c_2\}$.

Claim: If the Turing machine accepts $x$, then $\phi$ is satisfiable. Given the machine’s computation tree for $x$, we can construct model $(W, R_1, R_2, R_3, V)$ for $\phi$. $W$ is the set of configurations in the computation tree; let $R_1, R_2$ be minimal such that if $u$ is a universal configuration and $v, w$ its next configurations, then $uR_1v$ and $uR_2w$ (or $uR_2v$ and $uR_1w$), while if $u$ an existential configuration and $v$ its next accepting configuration, then $uR_1v$ and $uR_2v$; let $R_3$ be the transitive closure of $R_1 \cup R_2$. $V$ is defined to be such that if $M = (W, R_1, R_2, R_3, V)$, then $M, u \models \phi_u$. Then, it is not hard to see that $M, c_0 \models \phi$.

For the case of $D \oplus \subseteq K4$, notice that if the machine is deterministic, we can eliminate $d_U$, half of $d_E$ and the subformulas beginning with $\Box_2$ from $\sigma'$ and rename the remaining modalities from $\Box_1, \Box_3$ to $\Box_1, \Box_2$. For the case of $D4_2 \oplus \subseteq K4$, we can define a translation from the language of $D \oplus \subseteq K4$ to
the language of $\mathbf{D_4}_2 \oplus \leq \mathbf{K_4}$: given a formula $\phi$ with $\Box_1, \Box_2$ as modalities, simply replace $\Box_2$ by $\Box_1 \Box_3 \Box_2$ and $\Box_1$ by $\Box_1 \Box_2$. The remaining argument is similar for the one for the case of $\mathbf{D}_2 \oplus \leq \mathbf{K}$ – the iteration of $\Box_1$ and $\Box_2$ helps cut off the propagation of boxes in the tableau, which does not happen for $\mathbf{D} \oplus \leq \mathbf{K_4}$.

We can use Lemma 5.1.4 to prove the following proposition.

**Proposition 5.1.5.** The 1-variable, diamond-free fragment of $\mathbf{D}_2 \oplus \leq \mathbf{K_4}$ is $\text{EXP}$-complete; the 1-variable, diamond-free fragments of $\mathbf{D} \oplus \leq \mathbf{K_4}$ and of $\mathbf{D_4}_2 \oplus \leq \mathbf{K_4}$ are $\text{PSPACE}$-complete.

**Proof.** We present a method to translate a formula $\phi$ in negation normal form into a 1-variable formula $\phi'$ such that $\phi$ is $\mathbf{D}_2 \oplus \leq \mathbf{K_4}$-satisfiable iff $\phi'$ is $\mathbf{D}_2 \oplus \leq \mathbf{K_4}$-satisfiable. Let $p_1, \ldots, p_k$ be all the propositional variables that appear in $\phi$ and assume $q$ is not one of them. Then, $p_i^v = \Box_1 \Box_2 q$ and $(\neg p_i)^v = \Box_1 \Box_2 \neg q$. $\phi'$ results from $\phi$ by replacing each literal $l$ by $l^v$. Notice that in a model $\mathcal{M}$ and state $u$, only one of $p_i^v$ and $(\neg p_i)^v$ can be true. Let $\mathcal{M} = (W, R_1, R_2, R_3, V)$, where $(W, R_1 \cup R_2)$ is an infinite rooted tree ($aR_1 b$ iff $b$ the left child of $a$ and $aR_2 b$ iff $b$ the right child of $a$), $u \in W$, the root, and $\mathcal{M}, u \models \phi$ (it is not hard to see how to construct such a model from any other). Then, for every $x \in W$, if there are some $y \in W$ and some positive
$j \in \mathbb{N}$, such that $yR_1R_2^jx$ ($R_2^j$ is defined: $R_2^1 = R_2$ and $aR_2^{j+1}b$ iff there is some $c$ s.t. $aR_2cR_2^jb$), then $y, j$ are unique. Thus, if $V'(q) = \{x \in W \mid \exists yR_1R_2^jx$ s.t. $y \in V(p_j)\}$, it is the case that for $\mathcal{M}' = (W, R_1, R_2, R_3, V')$, $\mathcal{M}', u \models \phi'$. On the other hand given a model $\mathcal{M}', u \models \phi'$, we can just define $V(p_i) = \{x \in W \mid \mathcal{M}', x \models \Box_1\Box_2q\}$, thus $\phi$ is satisfiable iff $\phi'$ is. If $\phi$ is diamond-free, then $\phi'$ is diamond-free.

Notice that the method above does not work for $D \oplus_{\subseteq} K4$. Thus we use another method: we translate a formula $\phi$ to a formula $\phi^1$ such that $\phi$ is $D \oplus_{\subseteq} K4$-satisfiable iff $\phi^1$ is $D \oplus_{\subseteq} K4$-satisfiable and $\phi^1$ only uses one variable.

Let $p_1, \ldots, p_k$ be the propositional variables that appear in $\phi$ and let $q$ be a new variable (not among $p_1, \ldots, p_k$). Let $s = q \land \Box_1q \land \bigwedge_{i=1}^k \Box^{2i+1} \neg q$. Then, we recursively define: $(p_i)^1 = \Box_1^{2i} q$; $(-p_i)^1 = \Box_1^{2i} \neg q$; $\bot^1 = \bot$; $(-\bot)^1 = \neg \bot$; $(\psi_1 \land \psi_2)^1 = \psi_1^1 \land \psi_2^1$; $(\psi_1 \lor \psi_2)^1 = \psi_1^1 \lor \psi_2^1$; $(\Box_1\psi)^1 = \Box_1^{2k+2}(\psi^1 \land s)$ ($\Box_1^x$ is $x$ iterations of $\Box_1$); finally, $(\Box_2\psi)^1 = \Box_2((\psi^1 \land q \land \Box_1q) \lor (\neg q \lor \Box q))$. Formula $s$ gives a “mold” to a model. We can assume that the frames for $D \oplus_{\subseteq} K4$ are of the form $(\mathbb{N}, +1, \leq)$. Furthermore, if we restrict ourselves to formulas of the form $\psi^1$, then we can assume that for every $n \in \mathbb{N}$, $n(2k+2), n(2k+2)+1 \models q$ and for $1 \leq i \leq k$, $n(2k + 2) + 2i + 1 \models \neg q$. Then, $n(2k + 2) \models (\Box_1\psi)^1$ if and only if $(n+1)(2k+2) \models \psi^1$, while $q \land \Box_1q$ is true only at multiples of $2k + 2$. So, $n(2k + 2) \models (\Box_2\psi)^1$ exactly when $(n+1)(2k+2) \models \psi^1$. 
Therefore, by induction on $\phi$, we can see that $\phi$ is $D \oplus_\subseteq K4$-satisfiable iff $\phi^1$ is $D \oplus_\subseteq K4$-satisfiable.

We can end this argument like the one for the case of $D_2 \oplus_\subseteq K$: the tableau run for $\phi$ can simulate the run for $\phi^1$ and vice-versa. Just map every prefix $\sigma$ from the first tableau to $\sigma'$ of the second one, such that $0' = 0$ and $(\sigma.1)' = \sigma'.1^{k+2}$. Then $\sigma \psi$ appears in a branch of the first procedure iff $\sigma' \psi^1$ appears in a branch which results from the “same” nondeterministic choices in the second procedure. Furthermore, it is not hard to see that $\sigma' (p_i)^1$ and $\sigma' (\neg p_j)^1$ result in a closed branch iff $i = j$.

One may wonder whether we can say the same for the variable-free fragment of these logics. The answer however is that we cannot. The models for these logics have accessibility relations that are all serial. This means that any two models are bisimilar when we do not use any propositional variables, thus any satisfiable formula is satisfied everywhere in any model, thus we only need one prefix for our tableau and we can solve satisfiability recursively on $\phi$ in polynomial time.

Notice that for the proofs above, the requirement that the respective accessibility relations are serial was central. Indeed, otherwise there was no way to achieve these results, as we would not be able to force extra worlds
in a constructed model. Then we would have to rely on the complexity contributed by propositional reasoning and at best we would get an \( \text{NP} \)-hardness result – as long as we allowed enough variables in our formula.

Then what about \( D_4 \oplus \subseteq K_4 \)? Maybe we could attain similar hardness results for this logic as for \( D_{4_2} \oplus \subseteq K_4 \). Again, the answer is no. As frames for \( D_4 \) come with a serial and transitive accessibility relation, frames for \( D_4 \oplus \subseteq K_4 \) are of the form \((W, R_1, R_2)\), where \( R_1 \subseteq R_2 \) and \( R_1, R_2 \) are serial and transitive. It is not hard to come up with tableau rule(s) for the diamond-free fragment, by adjusting the ones we gave for \( D_{4_2} \oplus \subseteq K_4 \) to simply produce \( 0.1 \phi \) from every \( \sigma \sqcap_i \phi \). This drops the complexity of satisfiability for the diamond-free fragment of \( D_4 \oplus \subseteq K_4 \) to \( \text{NP} \) (and of the diamond-free, 1-variable fragment to \( \text{P} \)), as we can only generate two prefixes during the tableau procedure.

### 5.1.4 A General Characterization

In this section we examine a more general setting and we conclude by establishing tight conditions that determine the complexity of satisfiability of the diamond-free (and 1-variable) fragments of such multimodal logics. As we have mentioned before, in this chapter we omit the \( \preceq \) relation (interaction) from the description; thus, a multimodal logic is a triple \((N, \preceq, F)\).
CHAPTER 5. THE COMPLEXITY OF CONVERSION

The cases for which $\subset=\emptyset$ have already had the complexity of their diamond-free (and other) fragments determined in [HSS10]. For the general case, we already have an EXP upper bound from [DDN05]. We leave to the reader to verify that $(N, \subset, F)$ is, indeed, a (fragment of a) regular grammar modal logic with converse. For example, $D_2 \oplus \subset D_4$ can easily be reduced to $K_2 \oplus \subset K_4$ by mapping $\phi$ to $\Diamond_1 T \land \Diamond_2 T \land \Box_3 (\Diamond_1 T \land \Diamond_2 T) \land \phi$ to impose seriality, for which the corresponding regular languages would be $\Box_1$, $\Box_2$, and $(\Box_1 + \Box_2 + \Box_3)^*$ (see [DDN05] for more details).

For every $i \in N$, let $\min(i) = \{j \in N \mid j \subset i \text{ or } j = i, \text{ and } \not\exists j' \subset j\}$ and $\min(N) = \bigcup_{i \in N} \min(i)$. We can now give tableau rules for $(N, \subset, F)$. Let

- $n_i(\sigma) = \sigma$, if either
  - the accessibility relations of the frames for $F(i)$ are reflexive, or
  - $\sigma = \sigma'.i$ for some $\sigma'$ and the accessibility relations of the frames for $F(i)$ are transitive;

- $n_i(\sigma) = \sigma.i$, otherwise.

The tableau rules appear in Table 5.6.

From these tableau rules we can reestablish EXP-upper bounds for all of these cases (see the previous sections). To establish correctness, we only show
CHAPTER 5. THE COMPLEXITY OF CONVERSION

\[
\begin{array}{cccc}
\sigma \boxdot_i \phi & \sigma \boxdot_j \phi & \sigma \boxdot_i \phi & \sigma \boxdot_i \phi \\
\sigma \boxdot_j \phi & \sigma \boxdot_i \phi & \sigma \phi & \sigma_n(\sigma) \phi \\
where \ j \subset i & where \ & where \ the \ frames \ of \ F(i) \ & where \ j \in \min(i) \\
i \in \min(N) & i \in \min(N) & \text{have \ reflexive \ accessibility \ relations} & \text{and} \ F(i)'s \ frames \ \text{have \ transitive \ accessibility \ relations}
\end{array}
\]

Table 5.6: Tableau rules for the diamond-free fragment of \((N, \subset, F)\)

how to construct a model from an accepting branch for \(\phi\), as the opposite direction is easier. Let \(W\) be the set of all the prefixes that have appeared in the branch. The accessibility relations are defined in the following (recursive) way: if \(i \in \min(N)\), then \(R_i = \{ (\sigma, n_i(\sigma)) \in W^2 \} \cup \{ (\sigma, \sigma) \in W^2 \mid n_i(\sigma) \notin W \)
or \(F(i)\) has reflexive frames\}; if \(i \notin \min(N)\) and the frames of \(F(i)\) do not have transitive or reflexive accessibility relations, then \(R_i = \bigcup_{j \subseteq i} R_j\); if \(i \notin \min(N)\) and the frames of \(F(i)\) do have transitive (resp. reflexive, resp. transitive and reflexive) accessibility relations, then \(R_i\) is the transitive (resp. reflexive, resp. transitive and reflexive) closure of \(\bigcup_{j \subseteq i} R_j\). Finally, (as usual) \(V(p) = \{ w \in W \mid w \ p \ appears \ in \ the \ branch \}\). Again, to show that the constructed model satisfies \(\phi\), we use a straightforward induction.

By taking a careful look at the tableau rules above, we can already make some simple observations about the complexity of the diamond-free fragments of these logics. Modalities in \(\min(N)\) have an important role when
determining the complexity of a diamond-free fragment. In fact, the prefixes that can be produced by the tableau depend directly on \( \text{min}(N) \).

**Lemma 5.1.6.** If for every \( i \in \text{min}(N) \), \( F(i) \) has frames with reflexive accessibility relations (\( F(i) \in \{T,S5\} \)), then the satisfiability problem for the diamond-free fragment of \((N, \subset, F)\) is \( \text{NP} \)-complete and the satisfiability problem for the diamond-free, 1-variable fragment of \((N, \subset, F)\) is in \( \text{P} \).

**Proof.** Notice that in this case, for every \( i \in \text{min}(N) \), \( n_i(0) = 0 \), so there is no way to generate any other prefix besides 0. \( \text{NP} \)-hardness is the result of the \( \text{NP} \)-hardness of propositional satisfiability. By the above we can restrict ourselves to 1-world models; when we use only one variable, they can all be generated in constant time. \( \square \)

Taking this reasoning one step further:

**Corollary 5.1.7.** If \( \text{min}(N) \subseteq \{i\} \cup A \) and \( F(i) \) has frames with transitive accessibility relations (\( F(i) \in \{D4,S5\} \)) and for every \( j \in A \), \( F(j) \) has frames with reflexive accessibility relations, then the satisfiability problem for the diamond-free fragment of \((N, \subset, F)\) is \( \text{NP} \)-complete and the satisfiability problem for the diamond-free, 1-variable fragment of \((N, \subset, F)\) is in \( \text{P} \).

**Proof.** Like above, notice that we can only generate two prefixes: 0 and \( 0.i \). \( \square \)
In [Dem00], Demri shows that satisfiability for $L_1 \oplus \subseteq L_2 \oplus \cdots \oplus \subseteq L_n$ is \textsc{exp}-complete, as long as there are $i < j \leq n$ for which $L_i \oplus \subseteq L_j$ is \textsc{exp}-hard. On the other hand, Corollary 5.1.7 shows that for all these logics, their diamond-free fragment is in \textsc{np}, as long as $L_1$ has frames with transitive (or reflexive) accessibility relations.

Finally, we can establish a general result about the complexity of the diamond-free fragments of these logics.

\textbf{Theorem 5.1.8.} 1. If there is some $i \in N$ and some $A \subseteq \min(i)$ for which

- there is no $j \in A$ where $F(j)$ has frames with reflexive accessibility relations (i.e. $F[A] \cap \{T, S5\} = \emptyset$),
- or
- $|A| = 2$ and for some $j \in A$, $F(j)$ has frames with accessibility relations that are not transitive, or
- $|A| = 3$ and
- $F(i)$ has frames with transitive accessibility relations (i.e. $F(i) \in \{D4, S5\}$),

then the satisfiability problem for the diamond-free, 1-variable fragment of $(N, \subset, F)$ is \textsc{exp}-complete;
2. otherwise, if there is some \( i \in N \) and some \( A \subseteq \min(i) \) for which

- there is no \( j \in A \) where \( F(j) \) has frames with reflexive accessibility relations (i.e. \( F[A] \cap \{T, S5\} = \emptyset \) and
- either
  - \( |A| = 2 \) and for some \( j \in \min(N) \), \( F(j) \) has frames with accessibility relations that are neither reflexive nor transitive, or
  - \( |A| = 3 \),

then the satisfiability problem for the diamond-free, 1-variable fragment of \((N, \subset, F)\) is \( \text{PSPACE} \)-complete;

3. otherwise, if there is some \( i \in N \) and some \( A \subseteq \min(i) \) for which

- there is no \( j \in A \) where \( F(j) \) has frames with reflexive accessibility relations (i.e. \( F[A] \cap \{T, S5\} = \emptyset \)),
- either
  - \( |A| = 1 \) and for \( j \in A \), \( F(j) \) has frames with accessibility relations that are not transitive, or
  - \( |A| = 2 \), and
• \( F(i) \) has frames with transitive accessibility relations (i.e. \( F(i) \in \{ D4, S5 \} \)),

then the satisfiability problem for the diamond-free (1-variable) fragment of \((N, \subset, F)\) is \text{PSPACE-complete};

4. otherwise, the satisfiability problem for the diamond-free (resp. and 1-variable) fragment of \((N, \subset, F)\) is \text{NP-complete} (resp. in \text{P}).

Proof. All the lower bounds (except for the one in 4, of course) are established by providing suitable translations. Notice that by definition, if \(|\min(i)| > 1\), then \( i \notin \min(i) \) (and thus, \( i \notin \min(N) \)). In every case, assume that \( \phi \) is the formula that is given.

We first prove 1. This is done by a translation from \( D_2 \oplus \subseteq K4 \). \( \phi \) is translated to \( \phi^m \), such that \( \phi \) is \( D_2 \oplus \subseteq K4 \)-satisfiable if and only if \( \phi^m \) is \((N, \subset, F)\)-satisfiable. If \( A = \{x, y\} \) and \( F(x) \) has frames with accessibility relations that are not transitive, then let \( \Box_{(1)} = \Box_x \Box_x \Box_x \Box_y \Box_y \Box_x \Box_x \Box_x \) and \( \Box_{(2)} = \Box_y \Box_x \Box_y \Box_x \Box_y \Box_x \Box_x \Box_x \), and \( \Box_{(3)} = \Box_i \Box_y \Box_x \Box_y \Box_x \Box_x \). Then, \( \phi^m \) results from \( \phi \) by replacing \( \Box_k \) by \( \Box_{(k)} \), where \( k = 1, 2, 3 \). We can see that the tableau for \( \phi^m \) follows the one for \( \phi \) – as long as we map (say \( w \) is mapped to \( w^m \)) 0 to 0 and \( \sigma.1 \) to \( \sigma^m.x.x.x.y.x.y.x.x \) and \( \sigma.2 \) to \( \sigma^m.y.x.x.y.x.y.x.x \). The important observation here is that if \( \sigma^m \Box_3 \psi \) eventually produces \( \alpha \psi \), then \( \alpha \) is either
some \((\sigma.\tau)^m\), or it is not an initial segment of any such \((\sigma.\tau)^m\). Therefore, by restricting the branches produced by the tableau for \(\phi^m\) to prefixes of the form \((\sigma.\tau)^m\), we have a simulation of the corresponding branch for \(\phi\), while there are some other prefixes, but we can see that each of those (say \(\pi.a\)) prefixes a set of formulas, which is a subset of a set of formulas prefixed by another prefix, which ends at \(a\) and is mapped from a prefix of the first tableau. This means that if \(\phi\) is satisfiable, than \(\phi^m\) is satisfiable. The other direction is easier.

When \(|A| = 3\), we can use a more straightforward translation, which resembles than one given to translate from \(D \oplus_{\subseteq} K4\) to \(D_{42} \oplus_{\subseteq} K4\). For \(\{a, b, c\} = \{x, y, z\}\), \(\phi^{ma}\) is defined recursively: 
\[
(p)^{ma} = p; \quad (\neg p)^{ma} = \neg p;
\]
\[
\bot^{ma} = \bot; \quad (\neg \bot)^{ma} = \neg \bot; \quad (\psi_1 \land \psi_2)^{ma} = \psi_1^{ma} \land \psi_2^{ma}; \quad (\psi_1 \lor \psi_2)^{ma} = \psi_1^{ma} \lor \psi_2^{ma};
\]
\[
(\Box_1 \psi)^{ma} = \Box_b \psi^{mb}, \text{ where if } a = x \text{ (resp. } y, z), \text{ then } b = y \text{ (resp. } z, x); \quad (\Box_2 \psi)^{ma} = \Box_c \psi^{mc}, \text{ where if } a = x \text{ (resp. } y, z), \text{ then } c = z \text{ (resp. } x, y); \quad (\Box_3 \psi)^{ma} = \Box_i \psi. \quad \text{As the main translation we can pick any of } \phi^{mx}, \phi^{my}, \phi^{mz} \text{ and as in the previous cases, we can simulate one tableau by the other.}
\]

To establish the stated lower bound for 2 when \(|A| = 3\), we can use the exact same translation as above (from \(D_2 \oplus_{\subseteq} K\)). When \(|A| = 2\) and \(A = \{x, y\}\), define \(\Box_{(1)} = \Box_x \Box_j, \Box_{(2)} = \Box_y \Box_j, \text{ and } \Box_{(3)} = \Box_i \Box_j\). The
translation happens just by replacing $\square_a$ by $\square_{(a)}$, for all $a = 1, 2, 3$. The translations to prove the lower bound for (iii) are just simplified versions of the above.

To establish the stated upper bounds, we give bounds for the number of prefixes that a tableau run can produce. For this, assume that all subformulas of $\phi$ are distinct.

If for some $\sigma \square_i \psi$, the branch produces both $\sigma.x \psi$ and $\sigma.y \psi$ (and $x \neq y$), then $x, y \in \min(i)$ and $F(x), F(y)$ have frames with accessibility relations that are not reflexive, which means we are either in case (i) or case (ii). On the other hand, if for $\sigma \square_i \psi$, the branch produces $\sigma.x \square_i \psi$, then $x \in \min(i)$ and $F(x)$ has frames with accessibility relations that are not reflexive, while $F(i)$ has frames with accessibility relations that are transitive. If $F(x)$ has frames with accessibility relations that are not transitive, or there is also some $y \in \min(i)$ such that $F(y)$ has frames with accessibility relations that are not reflexive, then we are either in case (i), or in case (iii). Otherwise, $\sigma$ and $\sigma.x$ are the only prefixes for $\square_i \psi$ and thus the only possible prefixes for $\psi$ - if $\sigma.x'$ or $\sigma.x.x'$ is another prefix for $\psi$, then $x' \in \min(i)$, which is a contradiction, because of the above. This establishes 4, because every subformula of $\phi$ can only have up to 3 prefixes.

Assume we are not in case 1. We give a non-deterministic algorithm which
uses polynomial space to solve satisfiability. The algorithm runs the tableau procedure and uses non-determinism exactly for the non-deterministic propositional rule. Since it uses only polynomial space, it (possibly) cannot hold the whole branch in memory, so it explores the prefixes in a certain order. This order is what enables the algorithm to use only polynomial space. Every time the algorithm visits prefix \( \sigma \), it applies all the rules that have as premise a formula prefixed by \( \sigma = \sigma'.y \). This possibly results in new sets of formulas that are prefixed by new prefixes, \( \sigma.x_1, \ldots, \sigma.x_k \). If there is some \( x_a \) such that there is some \( i \in N \) for which \( x_a, y \in \min(i) \) and \( F(i) \) has frames with transitive accessibility relations, then the algorithm visits \( \sigma.x_a \) last (there is at most one) and \( \sigma.x_a \) is called a last prefix.

If \( \sigma.x_b \) is not a last prefix, then the maximum modal depth\(^9\) of the formulas prefixed by \( \sigma.x_b \) is one less than the maximum modal depth of the formulas prefixed by \( \sigma' \). This bounds the number of prefixes that are not last prefixes and that are initial segments of a current prefix by at most \( 2|\phi| \). The space the algorithm uses at any time is the number of prefixes it has scheduled to visit (and has not done so yet), times some quantity which is linear with respect to \( |\phi| \) (the formulas prefixed by those prefixes). This number of prefixes is at most \( |N| \) times the number of initial segments of the current

\(^9\)The nesting depth of the boxes in a formula.
prefix that are not last prefixes. But we argued above that these prefixes are at most $|\phi|$.

Case 4 is given by Corollary 5.1.7.

\[ \square \]

### 5.2 The Complexity of Multi-Agent Justification Logic with Conversion

Using the results on Diamond-free Modal Logic we can finally conclude with a characterization of Multi-Agent Justification Logic with Conversion. As the following proposition suggests (especially if contrasted to Lemma 5.1.1), the complexity of Diamond-free Modal Logic and of Justification Logic are very close. The observations of this section first appeared in [Ach14a].

**Lemma 5.2.1.** Let $\mathcal{F}$ be a finite frame and $S \cup \{e\}$ be a set of $\ast_{\mathcal{F}}$-expressions. Let $\subseteq \{(i, i) \in \mathbb{N}^2\}$ (if we allow positive introspection as in this chapter, $\subseteq = \emptyset$). If $S \vdash_{\ast_{\mathcal{F}}} e$, then $e$ can also be derived by a derivation for the $\ast_{\mathcal{F}}$-calculus where $\ast \subset (\mathcal{F})$ is applied some times first, then $\ast \subset \text{Dis}(\mathcal{F})$, and then these rules are not applied any more.

**Proof.** By induction on the derivation length we can prove the lemma just for $\ast \subset (\mathcal{F})$; then, separately for $\ast \subset \text{Dis}(\mathcal{F})$ in a way that does not introduce any applications of $\ast \subset (\mathcal{F})$ if none are already present; then we can assume $S$ is closed under $\ast \subset (\mathcal{F})$ and the full lemma follows. \[ \square \]
Figure 5.1: The complexity of Diamond-free Modal Logic
\[
\frac{\sigma t :_i \phi}{\sigma t :_j \phi}
\]

where \( j \subset i \)

\[
\frac{\sigma t :_i \phi}{n_i(\sigma) \phi}
\]

where \( i \in \text{min}(N) \)

\[
\frac{\sigma t :_i \phi}{\sigma \ast_i (t, \phi)}
\]

where the frames of \( F(i) \) have reflexive accessibility relations

\[
\frac{\sigma t :_i \phi}{n_j(\sigma) t :_i \phi}
\]

where \( j \in \text{min}(i) \) and \( F(i) \)'s frames have transitive accessibility relations

\[
\frac{\sigma \neg t :_i \phi}{\sigma \neg \ast_i (t, \phi)}
\]

Table 5.7: Tableau rules for the diamond-free fragment of \((N, \subset, F)\)

**Proposition 5.2.2.** If \((N, \subset, F_M)\) corresponds to \((N, \subset, F_J)_{CS}, CS \in P,\) is schematic and axiomatically appropriate, and satisfiability for the diamond-free fragment of \((N, \subset, F_M)\) is in complexity class \( C \in \{\text{PSPACE}, \text{EXP}\} \) (resp. in \( \text{NP} \)), then \((N, \subset, F_J)\)-satisfiability is in \( C \) (resp. in \( \Sigma^p_2 \)).

**Proof.** For \((N, \subset, F_J)_{CS}\)-satisfiability we can use practically the same tableau as for \((N, \subset, F_M)\)-satisfiability (see Table 5.7). Only now, for a branch \( b \) to be accepting it additionally requires that if \( X = \{v \ast_i (t, \psi) \in b\} \), then there is no \( w \neg \ast_i (t, \psi) \in b \) such that \( X \models \ast_i (t, \psi) \). Soundness,
CHAPTER 5. THE COMPLEXITY OF CONVERSION

completeness, and complexity arguments are essentially the same. The dif-
ference is the $*$-calculus procedure, which is easy to see it ensures soundness
and completeness (see also the tableaux for JD and JD4). The $*$-calculus
derivation can be performed on each prefix separately (by Lemma 5.2.1, the
rules $* \subset \mathcal{F}$ and $* \subset \text{Dis}(\mathcal{F})$ are handled by the tableau); this can be
performed using an oracle to NP.

Corollary 5.2.3. Let $(N, \subset, F)_{\mathcal{CS}}$, such that $F[N] \subseteq \{J, JT, J4, JD, JD4, LP\}$
and $\mathcal{CS}$ is axiomatically appropriate, schematic, and in P.

1. If there is some $i \in N$ and some $A \subseteq \min(i)$ for which

   • there is no $j \in A$ where $F(j)$ has frames with reflexive accessibility
     relations (i.e. $F[A] \cap \{JT, S4\} = \emptyset$),

   • either
     
     $|A| = 2$ and for some $j \in A$, $F(j)$ has frames with accessibility relations that are not transitive, or
     
     $|A| = 3$

     and

   • $F(i)$ has frames with transitive accessibility relations (i.e. $F(i) \in \{JD4, LP\}$),
then the satisfiability problem for $\langle N, \subset, F \rangle_{CS}$ is EXP-complete;

2. otherwise, if there is some $i \in N$ and some $A \subseteq \min(i)$ for which

- there is no $j \in A$ where $F(j)$ has frames with reflexive accessibility relations (i.e. $F[A] \cap \{JT, LP\} = \emptyset$) and
- either
  - $|A| = 2$ and for some $j \in \min(N)$, $F(j)$ has frames with accessibility relations that are neither reflexive nor transitive, or
  - $|A| = 3$,

then the satisfiability problem for $\langle N, \subset, F \rangle_{CS}$ is PSPACE-complete;

3. otherwise, if there is some $i \in N$ and some $A \subseteq \min(i)$ for which

- there is no $j \in A$ where $F(j)$ has frames with reflexive accessibility relations (i.e. $F[A] \cap \{JT, LP\} = \emptyset$),
- either
  - $|A| = 1$ and for $j \in A$, $F(j)$ has frames with accessibility relations that are not transitive, or
  - $|A| = 2$, and
• $F(i)$ has frames with transitive accessibility relations (i.e. $F(i) \in \{JD4, LP\}$),

then the satisfiability problem for $(N, \subset, F)$ is PSPACE-complete;

4. otherwise, the satisfiability problem for $(N, \subset, F)_{CS}$ is $\Sigma^p_2$-complete.

Proof. This corollary is the immediate result of Lemma 5.1.1 and Propositions 4.2.1 and 5.2.2.

A, perhaps unexpected, consequence of this section is that when justification logic $(N, \subset, F)_{CS}$ is PSPACE-complete (or EXP-complete), so is its fragment that uses only one propositional, one justification variable, and no other kinds of justification terms.
CHAPTER 5. THE COMPLEXITY OF CONVERSION

Figure 5.2: The complexity of Diamond-free Modal Logic and Multi-Agent Justification Logic with Conversion
Chapter 6

The Complexity of Interacting Agents: Conversion and Verification

Now we present complexity results for Multi-Agent Justification Logic when we also have Verification as an interaction (and not just Conversion). We first examine what happens when we only have two agents; then we identify a class of logics for which the satisfiability problem remains in the second level of the Polynomial Hierarchy; finally, we demonstrate that with Verification we can achieve a higher complexity than with just Conversion. In fact, in the final section of this chapter we present the first known justification logic with a higher complexity than its corresponding modal logic.
6.1 Two-agent Justification Logic

The original systems by Yavorskaya were two-agent versions of LP [Yav08]. In this section we examine what happens in terms of the complexity of Satisfiability when we have exactly two agents. The results of this section come from [Ach14c]. In these cases we can use special notation. Logic $J = (J_1 \times_o J_2)_{CS}$ is $((\{1, 2\}, \subset, \leftarrow, F)_{CS}$, where

- $F(1) = J_1$, $F(2) = J_2$;
- if $\times_o$ is $\times$, then $\subset = \leftarrow = \emptyset$;
- if $\times_o$ is $\times_C$, then $\subset = \{(1, 2)\}$ and $\leftarrow = \emptyset$;
- if $\times_o$ is $\times_{CC}$, then $\subset = \{(1, 2), (2, 1)\}$ and $\leftarrow = \emptyset$;
- if $\times_o$ is $\times_t$, then $\subset = \emptyset$ and $\leftarrow = \{(1, 2)\}$;
- if $\times_o$ is $\times_{!!}$, then $\subset = \emptyset$ and $\leftarrow = \{(1, 2), (2, 1)\}$.

6.1.1 Tableaux and Satisfiability - the Method and the Tools

To test the satisfiability of $\phi$, we use a tableau procedure, which starts from $0 T \phi$ and we apply tableau rules to gradually decompose the initial formula and produce more formulas. Formulas used in the tableau are of the form
Table 6.1: Tableau rules for \((\text{JD} \times \text{JD})_{\text{CS}}\)

\[
\begin{align*}
  w T \phi & \rightarrow \psi \\
  w F \phi & \rightarrow \psi \\
  w T t :_i \phi & \\
  w F t :_i (t, \phi) & \\
  w F \phi & \mid w T \psi \\
  w T \phi & \mid w F \psi \\
  w T *_i (t, \phi) & \\
  w F *_i (t, \phi) & \\
\end{align*}
\]

As an example we give tableau rules for \((\text{JD} \times \text{JD})_{\text{CS}}\) in Table 6.1.

For these rules,

\[
R_i = \{(w, w.i) \in W^2\} \cup \{(w, w) \in W^2 \mid w.i \notin W\}.
\]

Then, \(\mathcal{F} = (W, R_1, R_2)\) and \(\mathcal{V}(p) = \{w \in W \mid w T p \text{ appears in the branch}\}\).

Let \(S = \{w *_i (t, \psi) \mid w T *_i (t, \psi) \text{ appears in the branch}\}\) and \(\mathcal{E}\) be the
admissible evidence function such that

\[ E_i(t, \phi) = \{ w \in W \mid S \vdash_{\mathcal{E}_i(J)} w \ *_i (t, \phi) \} \].

\( \mathcal{M} = (W, R_1, R_2, \mathcal{E}_1, \mathcal{E}_2, \mathcal{V}) \) is a model, as \( R_1, R_2 \) are serial and \( \mathcal{E} \) is an admissible evidence function. It is not hard to see by induction on the structure of formulas \( \psi, \psi' \) that for every \( w \ T \psi \) and \( w \ F \psi' \) in the branch, \( \mathcal{M}, w \models \psi \) and \( \mathcal{M}, w \not\models \psi' \), as long as there is no prefixed \( \ast \)-expression \( w \ F \ast_i (t, \nu) \) appearing in the branch that \( w \in E_i(t, \nu) \). Thus we say the branch is accepting exactly when it is not propositionally closed and there is no prefixed \( \ast \)-expression \( w \ F e \) in the branch such that \( S \vdash \ast w e \).

If there is an accepting branch, then from the above we see that \( \phi \) is satisfiable. On the other hand it is not hard to see how to construct an accepting branch for \( \phi \) given a Fitting model for \( \phi \) that satisfies the Strong Evidence property: we map each world prefix \( w \) to a world \( w^M \) of the model such that \( 0^M \) is a world satisfying \( \phi \) and \( w.i \) is mapped to a world accessible through \( R_i \) from \( w^M \). Then we ensure that we only produce formulas \( w \ T \psi \) such that the world \( \mathcal{M}, w^M \models \psi \) and formulas \( w \ F \psi \) such that \( \mathcal{M}, w^M \not\models \psi \). Thus we ensure the branch is accepting. That the number of formulas in the branch is polynomially bounded results from the observation that the formulas prefixed by distinct world-prefixes are distinct – assuming all
subformulas of $\phi$ are distinct. This means that $(JD \times JD)_{CS}$-satisfiability is in $\Sigma^p_2$.

For the cases that follow we use a similar tableau procedure and arguments for its correctness and complexity. We will only explain what changes for each case as needed. The propositional rules (the first two) and the rule for $w \text{ F } t ;_i \phi$ (the last one) remain exactly the same for all logics – we only consider models with the strong evidence property. Notice that to prove the $\Sigma^p_2$ upper bound for a two-agent logic, it is enough to have a polynomial as a bound on the number of the world-prefixes in a branch of its corresponding tableau. This in turn gives a polynomial as an upper bound for the total size of a branch and thus the time it takes to (nondeterministically) apply all the tableau rules, as each rule increases the size of the branch. Since to decide satisfiability for a formula we need to do just that (nondeterministically run the tableau) and at the end for every prefixed $*$-expression $w \text{ F } e$ in the branch we need to determine whether $S \vdash _* w e$, we already have a nondeterministic algorithm that runs in polynomial time and uses an oracle from NP.

For several cases it is important to know Lemmata 6.1.1 and 6.1.2, as well as the finite frame property of these logics, as established by Corollary 4.4.2. Lemma 6.1.1 describes the situation when the logic is a pair of logics
with serial accessibility relations and has both versions of the Verification axiom. On the other hand, Lemma 6.1.2 is more general and, perhaps, more surprising. A very similar result is Proposition 3.3.3.

**Lemma 6.1.1.** If $\phi$ is $(\mathcal{J}_1 \times_{!!} \mathcal{J}_2)_{CS}$-satisfiable and $\mathcal{J}_1, \mathcal{J}_2 \in \{JD, JD4\}$, then there is some $(\mathcal{J}_1 \times_{!!} \mathcal{J}_2)_{CS}$-model $\mathcal{M} = (W, R_1, R_2, \mathcal{E}_1, \mathcal{E}_2, \mathcal{V})$, where $W = \{u, a_1, a_2\}$, $\mathcal{M}, u \models \phi$, and for $i \in \{1, 2\}$ $R_i = \{(x, a_i) \in W^2\}$.

**Proof.** Consider a Fitting model $\mathcal{M} = (W, R_1, R_2, \mathcal{E}_1, \mathcal{E}_2, \mathcal{V})$ which has the strong evidence property and such that $W$ is finite (see Corollary 4.4.2) and some $u \in W$ such that $\mathcal{M}, u \models \phi$. Let $a_0, b_0 \in W$ such that $uR_1a_0$ and $a_0R_2b_0$. Then, for $k \in \mathbb{N}$, let $a_{k+1}, b_{k+1} \in W$ be such that $b_kR_1a_{k+1}R_2b_{k+1}$. Then, for every $l, k \in \mathbb{N}$ such that $l < k$, $u, a_l, b_lR_1a_k$ and $u, a_l, b_lR_2b_k$. Since $W$ is finite, there are some $k, k' \in \mathbb{N}$ such that $k < k'$ and $a_k = a_{k'}$ (and thus, $a_k, b_kR_1a_k$ and $a_k, b_kR_2b_k$).

Let $W' = \{u, a_k, b_k\}$, $R'_1 = \{(a, a_k) \mid a \in W'\}$, $R'_2 = \{(a, b_k) \mid a \in W'\}$, and $\mathcal{V}'(p) = \mathcal{V}(p) \cap W'$. $\mathcal{E}'(t, \psi) = \mathcal{E}(t, \psi) \cap W'$ and $\mathcal{E}'$ is then an admissible evidence function. If not, it should violate one of its closure conditions, but it is not hard to see that they are all satisfied.

Then, $\mathcal{M}' = (W', R'_1, R'_2, \mathcal{E}', \mathcal{V}')$ is a model and we can determine in a straightforward way that $\mathcal{M}', u \models \phi$. \qed
Lemma 6.1.2. Let $J = (\mathcal{J}_1 \times_o \mathcal{J}_2)$ and for some $i \in \{1, 2\}$, $\mathcal{J}_i = JD4$.

If $\mathcal{M} = (W, R_1, R_2, \mathcal{E}_1, \mathcal{E}_2, V)$ is a $J$-model and $W$ is finite, then for every $a \in W$, there is some $b \in W$, such that $aR_ibR_ib$ and for every $c \in W$, if there is some $b' \in W$ for which $c,bR_ib'$, then $cR_ib$.

Proof. Consider such a Fitting model for $J$, $\mathcal{M} = (W, R_1, R_2, \mathcal{E}_1, \mathcal{E}_2, V)$.

For that $a \in W$ let $a_0 \in W$ such that $aR_ia_0$. Let $S_k = \{x \in W \mid \exists y, a_kR_ix \text{ and not } yR_ia_k\}$ and let $a_{k+1} \in S_k$, if $S_k \neq \emptyset$, and $a_{k+1} = a_k$ otherwise. Then, for every $l, k \in \mathbb{N}$ such that $l < k$, $a, a_lR_ia_k$ and thus $S_k \subset S_l$.

But since $W$ is finite, there must be some $k \in \mathbb{N}$ such that $S_k = \emptyset$. \hfill \Box

6.1.2 Complexity Results

The results of this section are summed up by Theorem 6.1.3:

Theorem 6.1.3. Let $J = (\mathcal{J}_1 \times_o \mathcal{J}_2)_{CS}$, where

$\mathcal{J}_1, \mathcal{J}_2 \in \{J, JD, JT, J4, JD4, LP\}$

and

$x_o \in \{x, x!, x!!, x_C, x_{CC}\}$.

If $\mathcal{J}_2 = JD$, $\mathcal{J}_1 \in \{J4, JD4, LP\}$ and $x_o = x_C$, or $\mathcal{J}_2 = JD$, $\mathcal{J}_1 \in \{JT, LP\}$ and $x_o = x_!$, then $J$-satisfiability is $\text{PSPACE}$-complete. In every other case, $J$-satisfiability is in $\Sigma_2^n$. 
\[
\frac{w \cdot T \cdot t : i \phi}{w \cdot T \cdot \ast_i (t, \phi)} \quad J
\]

\[
\frac{w \cdot T \cdot t : i \phi}{w \cdot i \cdot T \cdot \phi} \quad JD
\]

\[
\frac{w \cdot T \cdot t : i \phi}{w \cdot T \cdot \ast_i (t, \phi)} \quad JT
\]

\[
\frac{w \cdot T \cdot t : i \phi}{w \cdot T \cdot \ast_i (t, \phi)} \quad J4
\]

\[
\frac{w \cdot T \cdot t : i \phi}{v \cdot i \cdot T \cdot \phi} \quad JD4
\]

\[
\frac{w \cdot T \cdot t : i \phi}{w \cdot T \cdot \phi} \quad LP
\]

where if \( w \) of the form \( w' \cdot i \), then \( v = w' \) and otherwise \( v = w \).

Table 6.2: Tableau rules for logics without interactions. We use two of these rules: the ones that correspond to \( J_1 \) and \( J_2 \).

We do not examine \( (J_1 \times_{CC} J_2)_{CS} \), as it is essentially a single-agent logic: it is not hard to see that \( t : 1 \phi \leftrightarrow t : 2 \phi \) is a theorem of the logic. Furthermore, the cases where \( \times_o \in \{\times_C, \times_{CC}\} \) are handled by Corollary 5.2.3.

**No interactions:** \( \times_o = \times \).

Since there are no interactions we simply use the usual rule for \( w \cdot F \cdot t : i \psi \) that gives \( w \cdot F \cdot \ast_i (t, \psi) \) and one rule for each agent \( i \) depending on what \( J_i \) is. These are given in Table 6.2 together with the corresponding \( J_i \). The reasoning follows the one for the case of \( (J_D \times JD)_{CS} \), except for the case when \( J_i = JD4 \), where when we construct an accepting branch from a model (of finite states and with the strong evidence property), we can use Lemma 6.1.2 and thus if we map \( w \) to \( a \) we map \( w \cdot i \) to some \( b \) such that \( aR_i b \).
Of course, when we construct a model from an accepting branch we need to provide a different accessibility relation to account for the different logics. In particular, if $J_i \in \{J, J4\}$, then $R_i = \emptyset$; if $J_i \in \{JT, LP\}$, then $R_i = \{(w, w) \in W^2\}$; if $J_i = JD$, as in the case of $(JD \times JD)_{CS}$,

$$R_i = \{(w, w.i) \in W^2\} \cup \{(w, w) \in W^2 | w.i \notin W\};$$

finally, if $J_i = JD4$, then

$$R_i = \{(w, w.i) \in W^2\} \cup \{(w.i, w.i) \in W^2\} \cup \{(w, w) \in W^2 | w.i \notin W\}.$$  

**Verification:** $\times_\circ = \times_{\Downarrow}$.

When $\times_\circ = \times_{\Downarrow}$ and $J_1, J_2$ are among JD, JT, JD4, and LP, Lemma 6.1.1 applies, so we can base our rules on the three-world models it describes.\(^1\)

Then, the remaining argument remains the same as before, except when we define the accessibility relations, depending on the logics, if $J_i \in \{JT, LP\}$, then $R_i = \{(w, w) \in W^2\}$ and otherwise, $R_i = \{(w, 0.i) \in W^2\}$. The rules are in Table 6.3.

On the other hand if one of the two agents is based on J or J4, then we can use the same rules and reasoning as for the case when $\times_\circ = \times$.

\(^1\)In fact if one of $J_1, J_2$ is JT or LP, then only up to two worlds are required in the model, as these logics require reflexivity and not seriality of their accessibility relation.
\[
\frac{w \top t \vdash_1 \phi}{0.i \top \phi} \quad \text{JD,JD4}
\]
\[
\frac{w \top \ast_i (t, \phi)}{w \top \ast_i (t, \phi)}
\]
\[
\frac{w \top t \vdash_1 \phi}{w \top \phi} \quad \text{JT,LP}
\]

Table 6.3: Tableau rules for when \(\times_\circ = \times_!!\).

\[
\frac{w \top t \vdash_1 \phi}{w.s.1 \top \phi} \quad \text{JD}
\]
\[
\frac{w \top \ast_1 (t, \phi)}{w \top \ast_1 (t, \phi)}
\]
\[
\frac{w \top t \vdash_1 \phi}{w.s \top \phi} \quad \text{JT,LP}
\]
\[
\frac{w \top \ast_1 (t, \phi)}{w \top \ast_1 (t, \phi)}
\]
\[
\frac{w \top t \vdash_1 \phi}{v.1 \top \phi} \quad \text{JD4}
\]
\[
\frac{w \top \ast_1 (t, \phi)}{w \top \ast_1 (t, \phi)}
\]

where for the first two rules, \(s \in 2^*\) and \(w.s\) has already appeared and for the third one, either \(w\) of the form \(0.s\), where \(s \in 2^*\) and \(w.s\) has already appeared, or \(w\) of the form \(0.w_1.1.w_2\) (and \(w_1, w_2 \in 2^*\)) and \(v = 0.w_1\). \((2^*\) is the set of strings that only use 2 as a symbol. If \(A\) is a binary relation, then \(A^*\) is its reflexive transitive closure. When \(A\) is a set (of symbols) and not a binary relation, then \(A^*\) is the set of strings that use \(A\) as their alphabet. \(0.2^* = \{0.a \mid a \in 2^*\}\).

Table 6.4: Tableau rules for \(\times_\circ = \times_1\).

\textbf{Verification:} \(\times_\circ = \times_1\).

Like before, if one of the two agents is based on \(J\) or \(J4\), then we can use the same rules and reasoning as for the case when \(\times_\circ = \times\). Thus we only examine the cases when \(J_1, J_2 \in \{\text{JD, JT, JD4, LP}\}\). For these cases we can use the same rules as in the case where \(\times_\circ = \times\) for \(J_2\) as well as one of the following two rules for \(J_1\) (Table 6.4). The first should be used if \(J_1 = \text{JD}\), the second one if \(J_1 \in \{\text{JT, LP}\}\) and the third one should be used if \(J_1 = \text{JD4}\).

The argument for this case is similar to the ones that have already been covered. Notice in all these cases that if in a frame, \(aR_2bR_1c\), then \(aR_1c\). To
justify the third rule, which is different from the ones we have encountered, we gave Lemma 6.1.2. Then, when constructing a branch from a model, if \( w \) is mapped to \( u \), then we map \( w.1 \) to such a \( b \in W \) as indicated by Lemma 2, such that for every \( c \in W \), if there is some \( b' \in W \) such that \( c, bR_1 b' \), then \( cR_1 b \). When we construct a model from an accepting branch, we can define \( W \) to be the set of all world prefixes that appear in the branch as well as \( 0.s.1 \) for every \( s \in 2^* \) such that no world prefix \( 0.s.i \) where \( i \in \{1, 2\} \) appears. Then,

\[
R_1 = \{(w, w.u.1) \in W^2\} \cup \{(w.1.u, w.1) \in W^2\}.
\]

For the first rule we can impose two extra restrictions (without affecting the argument for correctness): we give this rule the lowest priority – it can only be applied when there are other rules to apply – and when it introduces \( w.s.1 \), then there must be no \( w.s.s' \) already in the branch, where \( s' \) not empty (i.e. \( w.s \) must be maximal). Thus we ensure that for every \( w T t : i \phi \) that appears in the branch, the rule produces only one formula of the form \( w' T \phi \); this condition gives an upper bound of \( |\phi| \) (where \( \phi \) the initial formula) for the number of world-prefixes. When we use the third rule \( (J_1 = JD4) \), then notice that \( 1 \) can only appear once in a prefix. Then, the number of prefixes of the form \( 0.2^* \) is at most \( |\phi| \) and so is for any given \( w.1 \) the number of prefixes of the form \( w.1.2^* \); this gives an upper bound of \( O(|\phi|^2) \) on the total
number of world-prefixes. The exception is when $\mathcal{J}_1 \in \{\text{JT}, \text{LP}\}$; in that case, if $\mathcal{J}_2 \in \{\text{JT}, \text{LP}, \text{JD4}\}$, we still have at most two prefixes, but $(\mathcal{J}_1 \times_1 \text{JD})_{CS}$-satisfiability is PSPACE-complete:

**Proposition 6.1.4.** $(\text{JT} \times_1 \text{JD})_{CS}$-satisfiability and $(\text{LP} \times_1 \text{JD})_{CS}$-satisfiability are PSPACE-complete.

**Proof.** To prove the stated upper bound we use the tableau from Table 6.5. Correctness is proven as usual. Notice that Lemma 5.2.1 can also be proven for these logics, as rule $\ast \phi$ cannot be applied before rule $\ast \phi \text{ Dis}(\mathcal{F})$ in the $\ast \mathcal{F}$-calculus, so the $\ast \mathcal{F}$ can be applied locally at each prefix. Therefore we can simulate the tableau using nondeterministic (for the propositional rules) polynomial space (keeping up to two prefixes – represented by their length in binary – in memory and terminating after $2^{\lvert \phi \rvert + 1}$ steps).

For the lower bound, simply notice that these logics have exactly the same corresponding modal logic as $(\text{LP} \times_1 \text{JD})_{CS}$. □
6.2 Tableau Procedures for Multi-Agent Justification Logic

In this section we give a general tableau procedure for every logic which varies according to each logic’s parameters. We can then use the tableau for a particular logic and make observations on its complexity.

If $A_1, \ldots, A_k$ are binary relations on the same set, then $A_1 \cdots A_k$ is the binary relation on the same set, such that $xA_1 \cdots A_k y$ if and only if there are $x_1, \ldots, x_k$ in the set, such that $x = x_1 A_1 x_2 A_2 \cdots A_{k-1} x_k A_k y$. If $A$ is a binary relation, then $A^*$ is the reflexive, transitive closure of $A$; if $A$ is a set (but not a set of pairs), then $A^*$ is the set of strings from $A$. We also use the following relation on strings: $a \sqsubseteq b$ iff there is some string $c$ such that $ac = b$.

We define $D = \{i \in N \mid F(i) = JD\}$, $\min(D) = \{i \in D \mid \nexists j \in D \text{ s.t. } j \subset i\}$ and for every $1 \leq i \leq n \min(i) = \{j \in \min(D) \mid j \subset i \text{ or } i = j\}$ (notice that if $F(i) \neq JD$, then $\min(i) = \emptyset$). These are important because we can use them to identify the agents that need to contribute new states we need to consider as we construct a model during the tableau. For example, consider a situation of three agents $F(1) = F(2) = F(3) = JD$, where $1 \subset 2 \subset 3$ and a state $w$ in a model. Then, since the accessibility relations $(R_1, R_2, R_3)$ are
serial, given formulas $t_1 : \phi_1$, $t_2 : \phi_2$, and $t_3 : \phi_3$, we need to consider some states $v_1 \models \phi_1$, $v_2 \models \phi_2$, and $v_3 \models \phi_3$. However, we also know that $R_1 \subseteq R_2 \subseteq R_3$, so $v_1$ is enough, as $v_1 \models \phi_1, \phi_2, \phi_3$. We will use this observation during the tableau.

The formulas used in the tableau will have the form $0.\sigma s \beta \psi$, where $\psi \in L^*_M$ or is a *-expression, $\sigma \in D^*$, $\beta$ is (either the empty string or) of the form $\Box_i \Box_j \cdots \Box_k$, $i, j, \ldots, k \in D$, and $s \in \{T, F\}$. Furthermore, $0.\sigma$ will be called a world-prefix or state-prefix, $s$ a truth-prefix and world prefixes will be denoted as $0.s_1.s_2 \cdots s_k$, instead of $0.s_1s_2 \cdots s_k$, where for all $1 \leq x \leq k$, $s_x \in D$.

Prefixes of the form $0.\sigma$, where $\sigma \in D^*$ represent states in the constructed model ($\mathcal{M} = (W, (R_i)^n_{i=1}, (E_i)^n_{i=1}, V)$ for this paragraph). The intuitive meaning of $\sigma T \psi$ is that $\mathcal{M}, \sigma \models \psi$ and of course, $\sigma F \psi$ declares that $\mathcal{M}, \sigma \not\models \psi$. Then, $\sigma T *_i (t, \psi)$ declares that $E \models \sigma T *_i (t, \psi)$ and $\sigma F *_i (t, \psi)$ declares that $E \not\models \sigma T *_i (t, \psi)$. As one may expect, the meaning of $\sigma T \Box_i \psi$ is that $\mathcal{M}, \sigma' \models \psi$ for every $\sigma R_i \sigma'$. Finally, $\sigma.i$ is some state in $W$ such that $\sigma R_i \sigma.i$.

A tableau branch is a set of formulas of the form $\sigma s \beta \psi$, as above. A branch is complete if it is closed under the tableau rules (they follow). It is propositionally closed if $\sigma T \beta \psi$ and $\sigma F \beta \psi$ are both in the branch, or if $\sigma T \bot$ is in the branch. We say that a tableau branch is constructed by
the tableau rules from \( \phi \), if it is a closure of \( \{0, T, \phi\} \) under the rules. The tableau rules for \( J \) can be found in Table 6.6, but before that we need some extra definitions.

For every \( i, j \in N \), \( i \in Nex(j) \) if \( i, j \in D \) and there is some \( j \subseteq i' \) such that \( i \in \text{min}(i') \). We can say that \( i \) is a “next” agent from \( j \) – if we can reach some state in the constructed model through \( R_i \), then we must consider a state that we can access through \( R_j \) from there. This is the essence of rule S (Table 6.6) and it is needed to prove correctness for the tableau (see the proof of Proposition 6.2.2).

We define the equivalence relation \( i \equiv j \) if \( i \subset^* j \subset^* j \subset^* i \) (notice that \( i \subset^* i' \) iff \( i \subset i' \) or \( i = i' \)). As an equivalence relation, it gives equivalence classes on \( D^+ = D \cup \{i \in N \mid F(i) = JT\} \); let the set of these classes be \( P \). Furthermore, notice that for any \( L \in P \), \( \exists x, y \in L \) s.t. \( x \preceq y \), or \( |L| = 1 \). In the first case, \( L \) is called a \( V \)-class of agents. For each agent \( i \in D^+ \), \( P(i) \) is the equivalence class which contains \( i \). The tableau we use for \( J \)-satisfiability makes use of the following lemma, which in many cases allows us to save on the number of states that are produced in the constructed frame.

**Lemma 6.2.1.** Let \( \mathcal{M} = (W, (R_i)_{i=1}^n, (E_i)_{i=1}^n, V) \) be a Fitting model on a finite frame, \( L \in P \) a \( V \)-class, and \( u \in W \). Then, there are states of \( W \),
(a_i)_{i \in L}, such that

1. For any \( i \in L \), \( uR_i a_i \).

2. For any \( i, j \in L, v, b \in W \), if \( a_i bR_j v \), then \( bR_j a_j \).

\((a_i)_{i \in L}\) will be called an \( L \)-cluster for \( u \).

Proof. For this proof we need to define the following. Let \( 1 \leq i \leq n, w, v \in W \). An \( E_V \)-path ending at \( i \) (and starting at \( i' \)) from \( w \) to \( v \) is a finite sequence \( v_1, \ldots, v_{k+1} \), such that for some \( j_1, \ldots, j_k \in N, E_1, \ldots, E_{k-1} \in \{\subset, \subseteq\} \),

where for some \( j \in [k-1] E_j = \subseteq \) and \( j_k = i \) (and \( j_1 = i' \)), for every \( a \in [k-1], j_a E_a j_{a+1} \) and if \( E_a = \subset \), then \( v_{a+1} = v_{a+2} \), while if \( E_a = \subseteq \), then \( v_{a+1} R_{j_{a+1}} v_{a+2} \) and \( v_1 = w, v_k = v, v_1 R_1 v_2 \). The \( E_V \)-path covers a set \( s \subseteq N \) if \( \{j_1, \ldots, j_k\} = s \). For this path and \( a \in [k], v_{a+1} \) is a \( j_a \)-state. Notice that if there is an \( E_V \) path ending at \( i \) from \( w \) to \( v \) and some \( j \in s \) and \( z \in W \) such that the path covers \( s \) and \( zR_j w \), it must also be the case that \( w, zR_i v \).

Let \( p : [m] \rightarrow L \) be such that \( m \in N, p([m]) = L \) and for every \( i+1 \in [m] \), either \( p(i+1) \subset p(i) \) or \( p(i+1) V p(i) \) and there is some \( i+1 \in [m] \) such that \( p(i+1) V p(i) \). For any \( s \in W, x \in N \) let \( b_0(s), b_1(s), b_2(s), \ldots, b_m(s) \) be the following: \( b_0(s) = s, \) for all \( k \in [m], b_1(s) \) will be such that there is an \( E_V \) path ending at \( p(1) \) from \( s \) to \( b_1(s) \) and covering \( P_a \) and if \( k > 1, b_k(s) \) is such that \( b_0(s), b_1(s), b_2(s), \ldots, b_k(s) \) is an \( E_V \) path ending at \( p(k) \). Let
(b^x_i)_{i \in [m], x \in \mathbb{N}}, (a^x_i)_{i \in [m], x \in \mathbb{N}} be defined in the following way. For every \( i \in [m] \), 
\( b^0_i = b_i(u) \) and for every \( x \in \mathbb{N} \), 
\( a^x_i = b_i(b^x_i) \). Finally, for \( 0 < x \in \mathbb{N} \), 
\( (b^x_i)_{i \in [m]} \) is defined in the following way. If there are some \( b_x, v \in W, i, j \in L \), such that
\( b_x R_j v, a^{x-1}_i R_j v \) and not \( b_x R_j a^{x-1}_j \), then for all \( i \in L \), 
\( b^x_i = b_i(v) \). Otherwise, 
\( (b^x_i)_{i \in L} = (a_i^x)_{i \in L} \). By induction on \( x \), we can see that for every \( x, y \in \mathbb{N} \), 
\( i \in L \), if \( y \geq x \), then \( b_x R_j b_y^x, a^y_i \). Since the model has a finite number of states, 
there is some \( x \in \mathbb{N} \) such that for every \( y \geq x \), 
\( (b^y_i)_{i \in L} = (a^y_i)_{i \in L} \). Therefore, 
we can pick appropriate \( (a_i)_{i \in L} \) among \( (a^k_i)_{i \in L} \) that satisfy conditions 1, 2.

We recursively define relation \( \Rightarrow \) on \( (D^+)^* \): if \( i \subset j \) then \( i \Rightarrow j \); if \( i \subseteq j \), 
then \( ij \Rightarrow j \); if \( \beta \Rightarrow \delta \), then \( \alpha \beta \gamma \Rightarrow \alpha \delta \gamma \). \( \Rightarrow^* \) is the reflexive, transitive closure 
of \( \Rightarrow \). We can see that \( \Rightarrow^* \) captures the closure conditions on the accessibility 
relations of a frame, so if for some frame \((W, (R_i)_{i=1}^n)\), \( aR_{i_1}R_{i_2} \cdots R_{i_k} b \) and 
\( i_1i_2 \cdots i_k \Rightarrow^* j_1j_2 \cdots j_l \), then \( aR_{j_1}R_{j_2} \cdots R_{j_l} b \). Furthermore, if, in addition, 
\( l = k \), then for every \( r \in [k], j_r \subset i_r \). For every agent \( F(i) = JD \), we introduce 
a new agent, \( \bar{i} \) and we extend \( \Rightarrow^* \), so that \( \bar{i} \Rightarrow^* i.\chi \) for every \( \chi \in D^* \) such 
that \( i.\chi \Rightarrow^* i \) (notice that \( \chi \) is not the empty string only if \( P(i) \) a V-class). 
Furthermore, if \( xy \in D^* \), then \( \overline{xy} = \overline{x} \overline{y} \). This extended definition of \( \Rightarrow^* \) 
tries to capture the closure of the conditions on the accessibility relations of 
a frame like the ones that will result from a tableau procedure as defined in
CHAPTER 6. CONVERSION AND VERIFICATION

the following.

Let $L \in P$ and $\sigma \in D^*$. Then, $L$ is visible from $0.\sigma$ if and only if there is some $i \in L$ and some $\chi, \alpha \in D^*$ such that $\sigma = \tau. i. \chi$ and $\overline{\chi} \alpha \Rightarrow^* i; \tau. \chi(i)$ is then called the $L$-view from $\sigma$. Notice that there is a similarity between this definition and the statement of Lemma 6.2.1 – this will be made explicit later on.

This discussion above the rules should explain rules TrB, SB, FB, C, and V, as well as S. Rule TrD merely says that when we encounter $\sigma T t :: i \psi$, we need to consider the states $\sigma'$ where $\sigma R_i \sigma'$ (see also the discussion on $\min(i)$ above). We do not need to produce $\sigma.j T \psi$, as this is handled by the following successive applications of the rules: TrB, C, SB. Rule Fa may seem strange, as in a model there may be two reasons for which $\sigma \not\models t :: i \psi$: either because of the admissible evidence function or because of the accessibility relation. Therefore, one would expect a nondeterministic choice for this rule (see for example [Kuz08a]); we use Fitting models with the Strong Evidence property, though, and in these models we know that $\sigma \not\models t :: i \psi$ because of the admissible evidence function.

If $b$ is a tableau branch, then Let $(R_i)_{i=1}^n$ be such that for every $1 \leq i \leq n$,

$$R_i = \{(\sigma, \sigma.i) \in (W(b))^2 \} \cup \{(w, w) \in (W(b))^2 \mid F(i) = JT\} \cup$$
CHAPTER 6. CONVERSION AND VERIFICATION

\[
\frac{\sigma T t : i \psi}{\sigma T : i (t, \psi)} \quad \text{TrB}
\]

\[
\frac{\sigma T : i \psi}{\sigma T \square_i \psi} \quad \text{SB}
\]

\[
\frac{\sigma T \square_i \psi}{\sigma. i T \psi} \quad \text{SB}
\]

\[
\frac{\sigma T : i \psi}{\sigma. j F \perp} \quad \text{TrD}
\]

if \( j \in \min(i) \), \( P(j) \) not a \( V \)-class visible from \( \sigma \);

\[
\frac{\sigma j \perp}{\sigma. j \perp} \quad \text{S}
\]

if \( i \in \text{Nex}(j) \), \( \sigma.i \) has appeared, and \( P(i) \) is not a \( V \)-class visible from \( \sigma.j \);

\[
\frac{\sigma T \square_i \psi}{\sigma T \tau \psi} \quad \text{C}
\]

if \( i \supset j \);

\[
\frac{\sigma T \square_i \psi}{\sigma T \square_j \psi} \quad \text{V}
\]

if \( i \supset j \).

\[
\frac{\sigma T \square_{i \psi}}{\tau. i T \psi} \quad \text{SVB}
\]

if \( P(i) \) a \( V \)-class visible from \( \sigma \),
\( \tau.j \) is the \( P(i) \)-view from \( \sigma \) and
\( \tau.j \) has already appeared in the tableau.

Table 6.6: The tableau rules for \( J \)
∪\{(\sigma, \tau.i) \in (W(b))^2 \mid P(i) a V\text{-class, } \tau.j the P(i)\text{-view from } \sigma\} then \( \mathcal{F}(b) = (W(b), (R'_i)_{i=1}^n) \), where \((R'_i)_{i=1}^n\) is the closure of \((R_i)_{i=1}^n\) under the conditions of frames for the accessibility relations, except for seriality.

Finally, let \( X(b) = \{ \sigma \ast_i (t, \psi) \mid \sigma T \ast_i (t, \psi) \text{ appears in } b \} \) and \( Y(b) = \{ \sigma \ast_i (t, \psi) \mid \sigma F \ast_i (t, \psi) \text{ appears in } b \} \). Branch \( b \) of the tableau is rejecting when it is propositionally closed or there is some \( f \in Y(b) \) such that \( X(b) \vdash \ast F(b) f \). Otherwise it is an accepting branch.

**Proposition 6.2.2.** Let \( \phi \in L^n_M \). \( \phi \) is \( J \)-satisfiable if and only if there is a complete accepting tableau branch \( b \) that is produced from \( 0 T \phi \).

**Proof.** We first prove the “if” direction. By induction on the construction of \( \mathcal{F}(b) \), it is not hard to see that for every \( (\sigma, \tau.j) \in R_i \), it must be the case that \( i \subset j \) or that \( F(i) = JT \) and \( \sigma = \tau.j \) and that if \( \sigma T \Box_i \phi \) appears in \( b \) and \( \sigma R_i \tau \), then \( \tau T \phi \) appears in \( b \). Let \( \mathcal{M} = (W, (R_i)_{i=1}^n, \mathcal{E}, \mathcal{V}) \), where \( (W, (R_i)_{i=1}^n) = \mathcal{F}(b) \), \( \mathcal{V}(p) = \{ w \in W \mid w T p \in b \} \), and \( \mathcal{E}_i(t, \psi) = \{ w \in W \mid \ast_T (b) \vdash \ast F.C \} \).

Let \( \mathcal{M}' = (W, (R'_i)_{i=1}^n, (\mathcal{E}_i)_{i=1}^n, \mathcal{V}) \), where for every \( 1 \leq i \leq n \), if \( F(i) = JD \), then \( R'_i = R_i \cup \{(a, a) \in W^2 \mid \exists j \in \min(i) \bar{\mathcal{E}}(a, b) \in R_j \} \) and \( R'_i = R_i \), otherwise. \( \mathcal{M}' \) is a Fitting model for \( J \): \( (\mathcal{E}_i)_{i=1}^n \) easily satisfy the appropriate conditions, as the extra pairs of the accessibility relations do not affect the
\textbf{CHAPTER 6. CONVERSION AND VERIFICATION} \hfill 205

\textasteriskcalculus derivation, and we can prove the same for \((R'_i)_{i=1}^n\). If \(aR'_ibR'_jc\) and \(i\preceq j\), if \((a,b)\in R'_i\setminus R_i\), then \(a=b\) and thus \(aR'_ic\). If \((a,b)\in R_i\), then, from rule S, there must be some \((b,c')\in R_j\), so \((b,c)\in R_j\) and thus, \((a,c)\in R_j\). If \((a,b)\in R'_i\) and \(i\subset j\), then, trivially, whether \((a,b)\in R_i\) or not, \((a,b)\in R'_j\).

By induction on \(\chi\), we prove that for every formula \(\chi\) and \(a\in W\), if \(a\top\chi\in b\) then \(\mathcal{M}',a\models_\chi\) and if \(a\bot\chi\in b\), then \(\mathcal{M}',a\not\models_\chi\). Propositional cases are easy. If \(\chi=t:i\psi\) and \(a\bot\chi\in b\), then \(a\not\in E_i(t,\psi)\), so \(\mathcal{M}',a\not\models_\chi\).

On the other hand, if \(a\top t:i\psi\in b\), then \(a\in E_i(t,\psi)\) and by rule TrD, for every \(j\in \min(i)\), there is some \((a,b)\in R_j\). Therefore, for every \((a,b)\in R'_j\), it is the case that \((a,b)\in R_j\), so by rule TrB and the inductive hypothesis, for every \((a,b)\in R_i\), \(\mathcal{M}',b\models_\psi\) and therefore, \(\mathcal{M}',a\models t:i\psi\).

We now prove the “only if” direction. Let \(\mathcal{M} = (W,(R_i)_{i=1}^n,(E_i)_{i=1}^n,V)\) that has the strong evidence property and a state \(s\in W\) such that \(\mathcal{M},s\models_\phi\). For every \(a\in W\) and V-class \(L\) fix some \(L\)-cluster for \(a\) (Lemma 6.2.1). For \(x\in D^*\), we define \((0.x)^\mathcal{M}\) to be such that \(0^\mathcal{M} = s\) and \((0.\sigma.i)^\mathcal{M}\) is some element of \(W\) s.t. \(((0.\sigma)^\mathcal{M},(0.\sigma.i)^\mathcal{M})\in R_i\); particularly, if \(P(i)\) a V-class and \((a_j)_{j\in L}\) is the fixed \(L\)-cluster for \(\sigma^\mathcal{M}\), then \((\sigma.i)^\mathcal{M} = a_i\).

Let \(L^\square_j = \{\square_{i_1}\cdots\square_{i_k}\phi \mid \phi \in L^n_j, k \in \mathbb{N}, i_1,\ldots,i_k \in \mathbb{N}\}\). Given a state \(a\) of the model, and \(\square_i\psi \in \text{sub}_\square(\phi)\), \(\mathcal{M},a\models_\square_i\psi\) has the usual, modal interpretation, \(\mathcal{M},a\models_\square_i\psi\) iff for every \((a,b)\in R_i\), \(\mathcal{M},b\models_\psi\).
CHAPTER 6. CONVERSION AND VERIFICATION

Notice that if $P(i)$ a $V$-class visible from $\sigma$ and $\tau.j$ is the $P(i)$-view from $\sigma$, then in model $\mathcal{M}$ there is some $v$ such that $\sigma^\mathcal{M}, (\tau.j)^\mathcal{M} R_i v$, which by the definition of clusters in turn means that $\sigma^\mathcal{M} R_j (\tau.j)^\mathcal{M}$. It is then straightforward to see by induction on the tableau derivation that there is a branch, such that if $\sigma T \psi$ appears in the branch and $\psi \in L^n_\Box$ or is a $*$-expression, then $\mathcal{M}, \sigma^\mathcal{M} \models \psi$ and if $\sigma F \psi$ appears, then $\mathcal{M}, \sigma^\mathcal{M} \not\models \psi$. The proposition follows.

6.2.1 Inside the Second Level of the Polynomial Hierarchy

In the tableau we presented, if for all appearing world-prefixes $\sigma.i$, $i$ in the same $V$-class $L$, then all prefixes are either of the form $0.j$, where $j \in L$. In this case we can simplify the box rules and in particular just ignore rule $V$ and end up with the following result.

Corollary 6.2.3. When $\min(D) = \emptyset$ or there is some $V$-class $L \in P$ such that $\min(D) \subseteq L$, then $J$-satisfiability is in $\Sigma^p_2$.

Corollary 6.2.3 may seem rather specific, but it settles that $J$-satisfiability is in $\Sigma^p_2$ for several cases. In particular, its assumptions are satisfied when $J$ is any multi-agent version of $\text{LP}$ ($F(i) = JT, i \subseteq i$ for all $i$), or even any combination of single-agent justification logics from $J$, $J4$, $JT$, $\text{LP}$ – not $JD$,
JD4 ($F(i) \neq JD$). Other interesting cases are logics with agents that form a single V-class – we consider these logics a way of generalizing JD4. For example, consider all agents in $D$ such that $i \subset j$ iff $i < j$ and $\subset = \{(n, i) \mid i \in N\}$. Then, we can think of the agent $i$’s justifications as increasing in reliability as $i$ increases and thus can model a situation where the agents are degrees of the belief of some other agent. Thus if the agent believes something with degree $n$, then the agent is aware of their belief with degree 1. On the other hand, if they believe something with degree 1, then they may not be aware of the fact that their belief is so reliable, so they are aware only of their least reliable belief. It would also make sense that for some $i_T < n$, for every $i < i_T$, $F(i) = JT$ instead of JD, so the most reliable beliefs could actually be knowledge. Furthermore, even if we have $\subset = \{(n, 1)\}$, the complexity of satisfiability remains in $\Sigma^p_2$.

### 6.2.2 More Hardness with Verification

In Chapter 5, we identified certain PSPACE-complete and EXP-complete logics. We demonstrate that in certain cases we can substitute a Conversion interaction or two by a Verification interaction. Specifically, let:\footnote{These logics have appeared again under different names in Chapter 4.}
\[J_1 = (\{1, 2\}, \subset_1, \leftarrow_1, F_1)_{CS} \text{ be such that for } i = 1, 2, \ F_1(i) = J D, \]
\[\rightarrow_1 = \supset_1 = \{(1, 2)\} \text{ (see Figure 6.1);} \]

\[\bullet \ J_2 = (\{1, 2, 3\}, \subset_2, \leftarrow_2, F_2)_{CS} \text{ be such that for } i = 1, 2, 3, \ F_1(i) = J D, \]
\[\subset = \emptyset, \text{ and } \rightarrow_2 = \{(1, 2), (1, 3)\}. \]

Figure 6.1: The logic \(J_1\) graphically.

Figure 6.2: The logic \(J_2\).

**Proposition 6.2.4.** For the logics \(J_1, J_2\) as defined above, under an axiomatically appropriate, schematic constant specification in \(P\), \(J_1\)-satisfiability and \(J_2\)-satisfiability are \(PSPACE\)-complete.

**Proof.** First we give tableau rules for each logic. These can be found in Table 6.7. To prove that for these logics the tableau for \(\phi\) accepts iff \(\phi\) is satisfiable,
Tableau rules for $J_1$:

\[
\begin{align*}
&\sigma T t : \psi \\
&\sigma T^* (t, \psi) \\
&\sigma.2 T \psi
\end{align*}
\]

\[
\begin{align*}
&\sigma T t : \psi \\
&\sigma.2 T t : \psi \\
&\sigma T^* (t, \psi) \\
&\sigma.2 T \psi
\end{align*}
\]

\[
\begin{align*}
&\sigma F t : \psi \\
&\sigma F^* (t, \psi)
\end{align*}
\]

Tableau rules for $J_2$:

\[
\begin{align*}
&\sigma T t : \psi \\
&\sigma T^*_i (t, \psi) \\
&\sigma.i T \psi
\end{align*}
\]

\[
\begin{align*}
&\sigma T t : \psi \\
&\sigma.i T t : \psi \\
&\sigma F t : \psi \\
&\sigma F^*_i (t, \psi)
\end{align*}
\]

if $i \in \{1, 2\}$ and

$\sigma.i$ has appeared;

we follow the usual procedure. Let $M, u \models \phi$. If the tableau produces prefixes of the form $0.\alpha$, where $\alpha \in A^*$, $A$ some set of agents, then for each $\alpha \in A^*$ let $0.\alpha$ be assigned to a state of $M$, where $0$ is assigned to $u$ and if $\alpha$ is assigned to $w$, then $\alpha.i$ is assigned to some $v$ such that $wR_i v$. Then it is not hard to verify that we can make sure that if the tableau produces $\sigma T g$ and $\sigma$ is assigned to $w$, then $M, w \models g$ and if the tableau produces $\sigma F g$ and $\sigma$ is assigned to $w$, then $M, w \not\models g$, so the branch accepts. On the other hand, from an accepting branch we can follow the standard construction and construct a model for $\phi$ (see the very similar proof of Proposition 6.1.4).

Notice that the tableau rules for $J_1$ are exactly the resulting tableau rules for $J_4 \times_C JD$ from Chapter 5, so we can easily argue these have the same
That $J_2$-satisfiability is in $\text{PSPACE}$ is also simple and very similar to the proof for Proposition 6.1.4. To prove that it is $\text{PSPACE}$-hard, we can show that the corresponding modal logic is $\text{PSPACE}$-hard and the proposition follows from Lemma 5.1.1. That is $M_2 = (\{1, 2, 3\}, \subset_2, \subset_2', F_2')$, where for $i = 1, 2, 3, F_1(i) = D$ and $\subset_2' = \{(1, 3), (2, 3)\}$ (we essentially renamed the agents for convenience). To prove $\text{PSPACE}$-hardness for $M_2$-satisfiability, we use a translation from $D_2 \oplus \subseteq D$. To translate a formula $\phi$, we replace $2_1$ by $2_1 2_3$, $2_2$ by $2_2 2_3$, and $2_3$ by $2_3$; the result is $\phi'$.

It is easy to construct a model $(W, R_1, R_2, R_3, V)$ for $\phi$ from a model $(W, R'_1, R'_2, R'_3, V)$ for $\phi'$, by $R_1 := R'_1 R'_3$, $R_2 := R'_2 R'_3$, and $R_3 := R'_3$. For the other direction, given a model $(W, R_1, R_2, R_3, V)$ for $\phi$, let $W' = W \cup R_1 \cup R_2 \cup R_3$, $R'_1 = \{(a, (a, b)) \in W \times R_1\}$, $R'_2 = \{(a, (a, b)) \in W \times R_2\}$, and $R'_3 = R_3 \cup \{((a, b), b) \in R_3 \times W\}$. Then, for $a, b \in W$, $a R'_1 R'_3 b$ iff $a R_1 b$ and $a R_3 b$; similarly, $a R'_2 R'_3 b$ iff $a R_2 b$; $a R'_3 b$ iff $a R_3 b$ by definition. Then, it is not hard to demonstrate by induction on $\psi$ that for every subformula $\psi$ of $\phi$ and $a \in W$, $M, a \models \psi$ if and only if $M', a \models \psi'$. \qed
Figure 6.3: Complexity characterizations for Multi-Agent Justification Logic
6.3 A NEXP-complete Justification Logic

So far we have seen modal and justification logics that are in at most \( \text{EXP} \)-complete. By now the reader may wonder whether the \( \text{NEXP} \) upper bound of Proposition 4.4.3 can be improved. The answer, as this section demonstrates, is that it cannot (assuming \( \text{EXP} \neq \text{NEXP} \)). The results of this section are from [Ach15a].

The justification logic we prove to have a \( \text{NEXP} \)-complete satisfiability problem is \( J_H = (\{1, 2, 3, 4\}, \subset, \sqsubseteq, F)_{CS} \), where

- \( \subset = \{(3, 4)\} \),
- \( \sqsubseteq = \{(1, 2), (2, 3), (4, 4)\} \),
- \( F(1) = F(2) = J, F(3) = F(4) = JD \), and\(^4\)
- \( CS \) is any axiomatically appropriate and schematic constant specification.

\( J_H \) is a four-agent logic. Its agents are based on justification logics \( J \) and \( JD \) – and essentially \( JD4 \), as agent 4 has Positive Introspection. Agent 3 has a significant variety of justifications. Since \( 1 \sqsupset 2 \sqsupset 3, 3 \) is aware

\(^4\)Notice that for the first time when we prove a hardness result, we do not require that all agents have serial accessibility relations. Nonetheless, agents 1 and 2 play important roles in the following reduction.
of the justifications of 2, who in turn is aware of the justifications of 1. Therefore, 3 can simulate the reasoning of 2 who can simulate the reasoning of 1. Additionally, 3 accepts two types of justifications: the ones 3 receives from 4, which come with Positive Introspection and the other ones 3 accepts, which do not. As Theorem 6.3.1 demonstrates, this complex interaction among agent 3’s justifications results in the significant hardness of $J_H$-satisfiability. Figure 6.4 gives a graphical representation of $J_H$.

**Theorem 6.3.1.** $J_H$-satisfiability is NEXP-hard.

The reduction we use is from the **BINARY SCHÖNFINKEL-BERNAYS SAT** problem, which is NEXP-complete (see Lemma 2.3.2). The reduction for Theorem 6.3.1 is essentially an extended version of the reduction we used to prove Theorem 3.4.2. Like then, consider a construction of a satisfying model, only this time it is a Fitting model with several states and accessibility relations for agents. Another difference is, of course, that now the original formula is from the first-order language. However, in the **BINARY**
SCHÖNFINKEL-BERNAYS SAT formulation, each (first-order) variable is quantified over two possible values (the elements of the two-element model), so they are essentially propositional variables. Since this is satisfiability we must existentially quantify each relation symbol over all $2^{r+1}$ r-ary relations. We can encode such a nondeterministic choice by forcing the existence of an exponential number of states, each representing one r-tuple $v = v_1, \ldots, v_r$ of the two possible values 0 and 1 (as mentioned above, we can do this using agents 3 and 4) by having \( \text{var} :_1 [p_a]^{v_a} \) being true and then at each such state enforce the choice between \( \text{rel} :_1 [R]^\top \) and \( \text{rel} :_1 [R]^\bot \), meaning that $v \in R$ or $v \notin R$ respectively – where $R$ an actual relation. In such a state conjunctions of the form \( \text{gather} :_1 ([p_1]^{v_1} \land \cdots \land [p_r]^{v_r} \land [R]^{\triangle}) \) (where $\triangle = \top$ or $\bot$) encode this choice. Due to the particular interaction among the agents and the logics they are based on, in the constructed model \( \text{gather} :_1 ([p_1]^{v_1} \land \cdots \land [p_r]^{v_r} \land [R]^{\triangle}) \) is true in a state if and only if that state represents $v$ and $\triangle = \top$ iff $v \in R$. Already this $J_H$-model encodes a first-order model. The trick now is to be able to gather in one state all these formulas that encode the relations through the admissible evidence function closure conditions (i.e. through the $\ast$-calculus), but making sure that individual conjuncts (i.e. something of the form \( \text{var} :_1 [p]^{\triangle} \) or \( \text{rel} :_1 [R]^{\triangle} \)) cannot be also transferred to that state through the calculus – in that case we would
be able to construct \[ \text{gather} :_1 ([p_1]^{v_1} \land \cdots \land [p_r]^{v_r} \land [R]^{\Delta}) \] for additional, invalid combinations of \((v, \Delta)\). This is achieved by considering formulas of the form \(!\text{gather} :_2 \text{gather} :_1 ([p_1]^{v_1} \land \cdots \land [p_r]^{v_r} \land [R]^{\Delta})\). The constructed model has empty accessibility relations for agents 1 and 2, thus such formulas can move freely through the accessibility relation of agent 3 (since \(2 \rightarrow 3\) and because of Distribution), but this is not the case for anything of the form \(t :_1 \chi\) (since \(1 \not\rightarrow 3, 4\)). Using certain additional formulas we can make sure that \(!\text{gather} :_2 \text{gather} :_1 ([p_1]^{v_1} \land \cdots \land [p_r]^{v_r} \land [R]^{\Delta}) \rightarrow [R(x_1, \ldots, x_r)]^{\Delta}\) becomes true if and only if \(x_1, \ldots, x_r\) are interpreted as \(v_1, \ldots, v_r\). The remaining of the formulas and methods we use are very similar to the ones we use for Theorem 3.4.2.

By combining Corollary 4.4.3 and Theorem 6.3.1, we can claim the following:

**Corollary 6.3.2.** \(J_H\)-satisfiability is \(\text{NEXP}\)-complete.

### 6.3.1 Proof of Theorem 6.3.1

Given a first-order formula \(\phi\) as above, we construct a justification formula, \(\phi^J\), in polynomial time, such that \(\phi\) is satisfiable by a two-element model if and only if \(\phi\) is satisfiable by a \(J\)-model. The reader will notice several similarities to the proof of Theorem 3.4.2. In fact, we use some of the definitions
from Section 3.4.

Let

\[
\phi = \exists x_1 \cdots \exists x_k \forall y_1 \cdots \forall y_{k'} \psi
\]

be such a formula, where \( \psi \) contains no quantifiers or function symbols. Let \( R_1, \ldots, R_m \) be the relation symbols appearing in \( \psi \), \( a_1, \ldots, a_m \) their respective arities. Then, let \( \alpha = \{ i \in \mathbb{N} \mid \exists r \leq m \text{ s.t. } i \leq a_r \} \); then, \( |\alpha| = \max\{a_1, \ldots, a_m \} \). We also define: \( X = \{x_1, \ldots, x_k\}; Y = \{y_1, \ldots, y_{k'}\}; Z = X \cup Y; \rho_0 = k + k' \).

For this reduction, in addition to the terms introduced in Section 3.4, we define the following justification terms. If we expect a term to justify a tautological scheme of fixed length, then we can just assume the term exists and has some constant size. Otherwise we construct the term in a way that gives it size polynomial with respect to the formula it (provably) justifies. Again we need certain terms to encode manipulations of long conjunctions (which we can see as strings) and we start with these.

**addhyp** is such that \( \vdash \text{addhyp}_1 (\phi \to (\psi \to \phi)) \);

**replaceleft** is such that \( \vdash \text{replaceleft}_1 ((\phi \to \phi') \to ((\phi \land \psi) \to (\phi' \land \psi))) \),

while
replaceright is such that \( \vdash \) replaceright : \( ((\psi \rightarrow \psi') \rightarrow ((\phi \land \psi) \rightarrow (\phi \land \psi')))));

We define replace\(^k\) in the following way:

\[
\text{replace}\_k = \text{replaceright},
\]

while for \( l < k \),

\[
\text{replace}\_k = \text{trans} \cdot \text{replace}\_k\_1 \cdot \text{replaced}.\]

Then it is not hard to see by induction on \( k - l \) that

\[
\vdash \text{replace}\_k : ((\phi_1 \rightarrow \phi_1') \rightarrow ((\phi_1 \land \cdots \land \phi_1 \land \cdots \land \phi_k) \rightarrow (\phi_1 \land \cdots \land \phi_1' \land \cdots \land \phi_k))).
\]

We define mphypothesis to be such that

\[
\vdash \text{mphypothesis} : (\phi \rightarrow \psi) \rightarrow ((\phi \rightarrow (\psi \rightarrow \chi)) \rightarrow (\phi \rightarrow \chi))).
\]

We use justification variables \( \text{var}_1, \ldots, \text{var}_{a_r}, \text{rel}_r \) for every \( r \in [m] \).

For \( 1 \leq r \leq m \) we define gather\(_r\) in the following way:

\[
gather\_r = [\text{append} \cdot [\text{append} \cdots [\text{append} \cdot \text{var}_1] \cdots \text{var}_{a_r}] \cdot \text{rel}_r],
\]
For every $1 \leq j \leq a_r + 1$, let $v_j, v'_j \in \{\top, \bot\}$. Then, for propositional variables $p_1, \ldots, p_{a_r}$,

$$\bigwedge_{j=1}^{a_r} \mathit{var}_j :_1 [p_j]^{v_j} \land \mathit{rel}_r :_1 [R_r]^{v_{ar+1}} \vdash$$

$$\mathit{gather}_r :_1 ([p_1]^{v_1} \land \cdots \land [p_{a_r}]^{v_{ar}} \land [R_r]^{v'_{ar+1}})$$

if and only if for every $1 \leq j \leq a_r + 1$, $v_j = v'_j$ (see the proof of Lemma 3.4.4). In fact it is not hard to see that if

$$\bigwedge_{j=1}^{a_r} \mathit{var}_j :_1 [p_j]^{v_j} \land \mathit{rel}_r :_1 [R_r]^{v_{ar+1}} \vdash \mathit{gather}_r :_1 \chi,$$

then

$$\bigwedge_{j=1}^{a_r} [p_j]^{v_j} \land [R_r]^{v_{ar+1}} \vdash \chi :$$

operator $!$ does not appear in $\mathit{gather}_r$, so the right-hand side of a corresponding $\ast$-calculus derivation for $\ast_1(\mathit{gather}_r, \chi)$ is a propositional derivation of $\chi$ from $[p_1]^{v_1}, \ldots, [p_{a_r}]^{v_{ar}}, [R_r]^{v_{ar+1}}$ and some propositional tautologies.

To give some intuition, conjunction $\bigwedge_{j=1}^{a_r} \mathit{var}_j :_1 [p_j]^{v_j} \land \mathit{rel}_r :_1 [R_r]^{v_{ar+1}}$ means that $(v_1, \ldots, v_{a_r}) \in R_r$ in a corresponding first-order model.

**We use justification variables** $\mathit{value}_z$ and $\mathit{match}(z, p_l)$ for every $z \in Z$, $l \in \alpha$. For every $z \in X$, we define $V_z = \mathit{value}_z :_1 [z]^T \lor \mathit{value}_z :_1 [z]^\perp$; for every $z \in Y$, $V_z = \mathit{value}_z :_1 [z]^T \land \mathit{value}_z :_1 [z]^\perp$. 
We also define

\[
\text{Match} = \bigwedge_{\substack{i \in \alpha \\
z \in \mathbb{Z} \\
\Delta \in \{\top, \bot\}}}
\text{match}(z, p_i) \cdot \bigl([z]^{\top} \to ([p_i]^{\top} \to \text{ok}_i)\bigr)
\]

For every \(R_r(\vec{z})\) which appears in \(\psi\) and \(0 \leq b \leq a_r\), we define term \(\text{match}^{R_r(\vec{z})}_b\) in the following way: \(\text{match}^{R_r(\vec{z})}_0 = \text{addhyp} \cdot \text{gather}_r\) and if \(b > 0\) and \(z_b = x_l\) or \(z_b = y_{l-k}\), then \(\text{match}^{R_r(\vec{z})}_b\) is defined to be the term

\[
[\text{mphypoth} \cdot \text{match}^{R_r(\vec{z})}_{b-1} \cdot \text{tran} \cdot \text{tran} \cdot \text{project}^{\rho_i} \cdot \text{match}(z, b)] \cdot \text{replace}^{a_r+1}_b.
\]

We can see by induction on \(b\) that for every \(0 \leq b \leq a_r\),

\[
\text{Match}, \text{gather}_r \cdot \bigl([p_1]^{v_1} \land \cdots \land [p_{a_r}]^{v_{a_r}} \land [R_r]^{v_{a_r+1}}\bigr) \vdash
\]

\[
\vdash \text{match}^{R_r(\vec{z})}_b(z_1, \ldots, z_{a_r}) \cdot \bigl([x_1]^{v_1} \land \cdots \land [x_k]^{v_k}, [y_1]^{v_{k+1}} \land \cdots \land [y_{k'}]^{v_{k'+k}} \rightarrow
\]

\[
\rightarrow (\text{ok}_1 \land \cdots \land \text{ok}_b \land [p_{b+1}]^{v_{b+1}} \land \cdots \land [p_{a_r}]^{v_{a_r}} \land [R_r]^{v_{a_r+1}})\bigr)
\]

if and only if for every \(j \in [a_r]\) and \(j' \in [k+k']\), if \(z_j = x_{j'}\) or \(z_j = y_{j'-k}\), then \(v'_j = v_{j'}\).

\(\text{Match}\) and term \(\text{match}^{R_r(\vec{z})}_b\) are used to confirm that given an assignment \(v\) for variables \(x_1, \ldots, x_k, y_1, \ldots, y_{k'}\), a tuple \(\vec{z} \in \mathbb{Z}^{a_r}\), and a tuple \((v'_1, \ldots, v'_{a_r+1}) \in \{\top, \bot\}^{a_r+1}\), that \((v(z_1), \ldots, v(z_{a_r})) = (v'_1, \ldots, v'_{a_r})\),
since this is a crucial condition to assert that $[R_r(z)]^{v_{a_r+1}}$ must be true (i.e. $R_r(z)$ is true iff $v_{a_r+1} = \top$).

$T^l(\text{match}_b^{R_r(z)})$ is defined in the following way:

$$T^l(\text{match}_0^{R_r(z)}) = c \cdot \text{!addhyp} \cdot \text{!gather}_r \text{ and for } b > 0 \text{ and } z_b = y_{l-k},$$

$$T^l(\text{match}_b^{R_r(z)}) =$$

$$\cdot [\text{!tran} \cdot [\text{!tran} \cdot \text{project}_{i}^{p_{1}} \cdot \text{match}(y_{l}, b) \cdot \text{replace}_{b}^{a_{r+1}}]$$

We can see by induction on $b$ that for every $0 \leq b \leq a_r$,

$$\vdash T^l(\text{match}_b^{R_r(z)}) :_2 \text{match}_b^{R_r(z)} :_1 ([p_{1}]^{x_{1}} \land \cdots \land [p_{a_{r}}]^{x_{a_{r}}} \land [R_{r}]^{v_{a_{r}+1}}) \vdash$$

$$\vdash T^l(\text{match}_b^{R_r(z)}) :_2 \text{match}_b^{R_r(z)} :_1 (\text{!addhyp} \cdot \text{!gather}_r \text{ and for } b > 0 \text{ and } z_b = y_{l-k},$$

if and only if

$$\vdash \text{match}_b^{R_r(z)} :_1 ([p_{1}]^{x_{1}} \land \cdots \land [p_{a_{r}}]^{x_{a_{r}}} \land [R_{r}]^{v_{a_{r}+1}}) \vdash$$

$$\vdash \text{match}_b^{R_r(z)} :_1 ([p_{1}]^{x_{1}} \land \cdots \land [p_{a_{r}}]^{x_{a_{r}}} \land [R_{r}]^{v_{a_{r}+1}}) \vdash$$

which in turn, as we have seen above, is true if and only if for every $j \in [a_r]$ and $j' \in [k + k']$, if $z_j = x_{j'}$ or $z_j = y_{j' - k}$, then $v_{j'} = v_{j'}'$. 

Using the terms (and formulas) we have defined above, we can construct terms $T^a$, where $0 < a \leq \rho_1$ and eventually $t^\phi$:

Let $\Psi = \{\psi_1, \ldots, \psi_l\}$ be an ordering of all subformulas of $\psi$ and of variables $x_1, \ldots, x_k, y_1, \ldots, y_{k'}$, which extends the ordering $x_1, \ldots, x_k, y_1, \ldots, y_{k'}$, such that if $a < b$, then $|\psi_a| \leq |\psi_b|$.\footnote{Assume $|\cdot|$, such that $|x_j| = |y_j| = 0, |R_j(v_1, \ldots, v_{a_j})| = 1$ and if $\gamma$ is a proper subformula of $\delta$, then $|\gamma| < |\delta|$} Furthermore, $\rho_0 = |\{a \in [l] \mid |\psi_a| = 0\}|$ ($= k + k'$) and $\rho_1 = |\{a \in [l] \mid |\psi_a| = 1\}|$.

Let $T^1 = value_{z_1}$ and for every $1 < a \leq \rho_0$, $T^a = [append \cdot T^{a-1} \cdot value_{z_a}]$.

It is not hard to see that for $v_1, \ldots, v_k \in \{\top, \bot\}$,

\[ value_{z_1} : 1[z_1]^{v_1}, \ldots, value_{z_k} : 1[z_k]^{v_k} \vdash T^{\rho_0} : 1([z_1]^{v_1} \land \cdots \land [z_k]^{v_k}). \quad (6.1) \]

For every $a \in [l]$,

if $\psi_a = R_r(z^a_1, \ldots, z^a_{a_r})$, then

\[ Eval_a = \text{truth}_a : 2([match_{a_r}^{\psi_a} : T^{\rho_0}] : 1(ok_1 \land \cdots \land ok_{a_r} \land [R_r]^T) \rightarrow [\psi_a]^T) \land \text{truth}_a : 2([match_{a_r}^{\psi_a} : T^{\rho_0}] : 1(ok_1 \land \cdots \land ok_{a_r} \land [R_r]^\perp) \rightarrow [\psi_a]^\perp); \]

if $\psi_a = \neg \gamma$, then

\[ Eval_a = \text{truth}_a : 2([\gamma]^T \rightarrow [\psi_a]^\perp) \land \text{truth}_a : 2([\gamma]^\perp \rightarrow [\psi_a]^T); \]
if $\psi_a = \gamma \lor \delta$, then


$$\land truth_a : 2 ([\gamma]^T \land [\delta]^T \rightarrow [\psi_a]^T) \land truth_a : 2 ([\gamma]^T \land [\delta]^T \rightarrow [\psi_a]^T);$$

if $\psi_a = \gamma \land \delta$, then


$$\land truth_a : 2 ([\gamma]^T \land [\delta]^T \rightarrow [\psi_a]^T) \land truth_a : 2 ([\gamma]^T \land [\delta]^T \rightarrow [\psi_a]^T);$$

if $\psi_a = \gamma \rightarrow \delta$, then


Let $Eval = \bigwedge_{a=\rho_0+1}^\rho Eval_a$.

For $\rho_0 < a \leq \rho_1$, we define $gathrel_a$ in the following way:

$$gathrel_{\rho_0+1} = c \cdot T^l(match_{a\in_a}^{\psi_a})$$

and for $\rho_0 + 1 < a \leq \rho_1$,

$$gathrel_{\rho_0+1} = appendconc \cdot gathrel_{a-1} \cdot [c \cdot T^l(match_{a\in_a}^{\psi_a})].$$
Then,

\[ T^{\rho_0 + 1} = \text{replace}_{\rho_1 - \rho_0}^{\rho_1} \cdot \text{truth}_{\rho_0 + 1} \cdot [\text{gathrel}_{\rho_1} \cdot !T^{\rho_0}] \]

and for \( \rho_0 + 1 < a \leq \rho_1 \),

\[ T^a = \text{replace}_{a}^{\rho_1 - \rho_0} \cdot \text{truth}_{\rho_0 + 1} \cdot T^{a - 1} \cdot \]

if \( \psi_a = \neg \psi_2 \), then

\[ T^a = \text{hypappend} \cdot [\text{trans} \cdot \text{proj}_{j}^{\rho_1 - \rho_0 - 1} \cdot \text{truth}_a] \cdot T^{a - 1} \]

and

if \( \psi_a = \psi_b \circ \psi_c \), then

\[ T^a = \text{hypappend} \cdot [\text{trans} \cdot [\text{appendconc} \cdot \text{proj}_{j}^{\rho_1 - \rho_0 - 1} \cdot \text{proj}_{c}^{\rho_1 - 1}] \cdot \text{truth}_a] \cdot T^{a - 1} \]

We then define \( t^\phi = [\text{right} \cdot T^i] \).

**Lemma 6.3.3.** For every \( b \in [\rho_1] \), \( j \in [a_{rb}] \), let \( \vec{l}^b = (l^b_1, \ldots, l^b_{a_{rb}}) \in \{p_j, \neg p_j\}^{a_{rb}} \) and \( v^b \in \{\top, \bot\} \). Assume that for every \( b_1, b_2 \in [\rho_1] \), if \( r_{b_1} = r_{b_2} \) and \( l^{b_1} = l^{b_2} \), then it must also be the case that \( v^{b_1} = v^{b_2} \). Then,\(^6\)

\[ \bigwedge_{b=1}^{\rho_1} \text{gather}_{r_b} \vdash \text{gather}_{r_b} : 1 \left( \vec{l}^b \wedge [R_{r_b}]^b \right) \wedge \text{Match} \wedge \text{Eval} \wedge \bigwedge_{z \in Z} \text{val}_z : 1 \left[ z \right]^{v_z} \vdash t^\phi : 2 [\phi]^T \]

if and only if \( M \models \phi \) for every model \( M \) with universe \( \{\top, \bot\} \) and interpretation \( I \) such that

\(^6\)For convenience and to keep the notation tidy, we identify \( \vec{l}^b \) with \( l^b_1 \wedge \cdots \wedge l^b_{a_{rb}} \) and \( \vec{ok} \) with \( ok_1 \wedge \cdots \wedge ok_{a_{rb}} \).
• for every $z \in \mathbb{Z}$, $v_z = I(z)$,

• for every $b \in [\rho_1]$, $\mathcal{M} \models R_{rb}(f(l^b_1), \ldots, f(l^b_{ar_b}))$ iff $v^b = \top$,

where for all $j \in \alpha$, $f(p_j) = \top$ and $f(\neg p_j) = \bot$.

Proof. The if direction is not hard to see by (induction on) the construction of the terms $T^a, t^\phi$. For the other direction, notice that a $*$-calculus derivation for

$$\bigwedge_{b \in \rho_1} !gather_{rb} \cdot_2 \text{gather}_{rb} \cdot_1 \left(\bar{l}^b \land [R_{rb}]^{\psi_b}\right),$$

\[\text{Match, Eval, } \bigwedge_{z \in \mathbb{Z}} \text{val}_z :_1 [z]^{v_z} \vdash t^\phi :_2 [\phi]^\top\]
gives on the right hand side a derivation of

$$\bigwedge_{b \in \rho_1} \text{gather}_{rb} \cdot_1 \left(\bar{l}^b \land [R_{rb}]^{\psi_b}\right), \text{Match, Eval}^\#_2, \bigwedge_{z \in \mathbb{Z}} \text{val}_z :_1 [z]^{v_z} \vdash [\phi]^\top$$

Some $\chi = [R_r(\bar{z}^r)]^{\Delta}$, where $R_r(\bar{z}^r) = \psi_a$, a subformula of $\phi$, can be derived from the assumptions above only if $[\text{match}_{ar_a}^\psi \cdot T^{po}] :_1 (\bar{o}^k \land [R_{ra}]^{\Delta})$ can be derived as well – notice that the assumptions cannot be inconsistent and we can easily adjust a model that does not satisfy $[\text{match}_{ar_a}^\psi \cdot T^{po}] :_1 (\bar{o}^k \land [R_{ra}]^{\Delta})$ so that it does not satisfy $\chi$ either, by simply changing the truth value of $\chi$.

The derivation of $\text{match}_{ar_a}^\psi :_1 (\bar{o}^k \land [R_{ra}]^{\Delta})$ is not affected by $\text{Eval}^\#_2$: if there is a model that satisfies all assumptions except for $\text{Eval}^\#_2$ and not $\text{match}_{ar_a}^\psi :_1 (\bar{o}^k \land [R_{ra}]^{\Delta})$, we can assume the strong evidence property and
change the truth-values of every \([\psi_b]_{\Delta'}\) to true, so the new model satisfies all the assumptions and not \([\text{match}^\psi_{\Delta'} \cdot T_{\rho_0}] :_1 (\vec{o} k \land [R_{\tau_a}]_{\Delta})\).

Therefore we have a *-calculus derivation of \([\text{match}^\psi_{\Delta'} \cdot T_{\rho_0}] :_1 (\vec{o} k \land [R_{\tau_a}]_{\Delta})\) and since \(\text{gather}_r\) only appears once in \(\text{match}^\psi_{\Delta'}\), there is some \(b \in [\rho_1]\) such that (see Lemma 3.4.3)

\[
gather_{r_b} :_1 \left(\vec{i}_b \land [R_{\tau_b}]_{\nu_b}\right), \text{Match}, \bigwedge_{z \in Z} \text{val}_z :_1 [z]_{\nu_z} \vdash \vdash [\text{match}^\psi_{\Delta'} \cdot T_{\rho_0}] :_1 (\vec{o} k \land [R_{\tau_a}]_{\Delta})
\]

Similarly, we can remove the terms from this derivation, so

\[
\vec{i}_b, [R_{\tau_b}]_{\nu_b}, \text{Match}^\#, \bigwedge_{z \in Z} [z]_{\nu_z} \vdash \vec{o} k \land [R_{\tau_a}]_{\Delta}
\]

From which it is not hard to see that for all \(z \in Z\), \(\nu_b = \Delta\), so every first-order model as described in the Lemma satisfies \(\chi\). Then it is not hard to see by induction that all such models satisfy all \([\psi_a]_{\Delta}\) derivable from these same assumptions.

\(\square\)

Now to construct the actual formula the reduction gives. For this let \(\rho\) be a fixed justification variable. We define the following formulas.

\[
\text{start} = \neg [\text{active}] \land \rho :_3 \left( [\text{active}] \land \bigwedge_{a \in [\alpha]} \text{var}_a :_1 \neg p_a \right)
\]
Then, \( \phi_{FO}^J \), the formula constructed by the reduction is the conjunction of these formulas above:

\[
\text{start} \land \text{forward}_A \land \text{forward}_B \land \text{forward}_C \land
\]
Theorem 6.3.4. $\phi_{FO}^J$ is $J$-satisfiable if and only if $\phi$ is satisfiable by a two-element first-order model.

Proof. First, assume $\phi$ is satisfiable by two-element first-order model, say $\mathcal{M}$ with interpretation $\mathcal{I}$, and assume that for every $1 \leq a \leq k$, $\mathcal{I}(x_a)$ is such that $\mathcal{M} \models \forall y_1, \ldots, \forall y_k \psi$. For a $m \in \mathbb{N}$, let $[m] = \{1, 2, \ldots, m\}$. We construct a $J$-model for $\phi_{FO}^J$:

$$\mathcal{M}_J = (W, R_1, R_2, R_3, R_4, \mathcal{E}, \mathcal{V}),$$

where:

- $W = \{\sigma \in \mathbb{N} \mid \sigma + 2 \in [2^a + 2]\}$ (i.e. $\sigma \in \{-1, 0, 1, 2, \ldots, 2^a\}$);
- $R_1 = R_2 = \emptyset$, $R_3 = \{(\sigma, \sigma + 1) \mid \sigma < 2^a\} \cup \{(2^a, 2^a)\}$, and $R_4 = \{(\sigma, \sigma') \mid \sigma < \sigma'\} \cup \{(2^a, 2^a)\}$;
- $\mathcal{E}$ is minimal such that
  - $\mathcal{E}_3(\rho, \chi) = \mathcal{E}_4(\rho, \chi) = W$ for any formula $\chi$,
  - $\mathcal{E}_1(var_a, p_a) = \{\sigma \in W \mid \sigma + 1 \in [2^a] \text{ and } bin_a(\sigma) = 1\}$,
  - $\mathcal{E}_1(var_a, \neg p_a) = \{\sigma \in W \mid \sigma + 1 \in [2^a] \text{ and } bin_a(\sigma) = 0\}$,
  - $\mathcal{E}_1(rel_r, [R_r]^\top) = \{\sigma \in W \mid \sigma + 1 \in [2^a] \text{ and }$ $\mathcal{M} \models R_r(bin_0(\sigma), \ldots, bin_{a_r}(\sigma))\}$.  


\(- \mathcal{E}_1(\text{rel}, [R_a]^{\perp}) = \{ \sigma \in W | \sigma + 1 \in [2^\alpha] \text{ and } \}
\neg \mathcal{M} \models R_r(bin_0(\sigma), \ldots, bin_a(\sigma)) \},
\)

- for every \(a \in [k]\), \(\mathcal{E}_1(\text{value}_{x_a}, [x_a]^\top) = \{2^\alpha\}\), if \(\mathcal{I}(x_a) = \top\) and \(\emptyset\) 
  otherwise,

- for every \(a \in [k]\), \(\mathcal{E}_1(\text{value}_{x_a}, [x_a]^\perp) = \{2^\alpha\}\), if \(\mathcal{I}(x_a) = \bot\) and \(\emptyset\) 
  otherwise,

- for every \(a \in [k']\), \(\mathcal{E}_1(\text{value}_{y_a}, [y_a]^\top) = \mathcal{E}_1(\text{value}_{y_a}, [y_a]^\perp) = \{2^\alpha\}\),
  and

- \(\mathcal{M}_J, 2^\alpha \models \text{Match, Eval};\)

\(\bullet \ V([\text{active}]) = \{ \sigma \in W | \sigma + 1 \in [2^\alpha] \}\) and for any other propositional 
variable \(q\), \(V(q) = \emptyset\).

It is not hard to verify that \(\mathcal{M}_J, -1 \models \phi_{FO}^J\), as long as we establish that 
\(\mathcal{M}_J, 2^\alpha \not\models t^\phi_2 [\neg \phi]^T\), for which it is enough that \(2^\alpha \not\in \mathcal{E}_j(t^\phi, [\neg \phi]^T)\).

The definition of \(\mathcal{E}\) is equivalent to \(\sigma \in \mathcal{E}_g(s, \chi) \iff S \vdash \sigma \ast_g (s, \chi)\), where 
\(S = \)

\[\{ w \ast_3 (\rho, F) | w \in W, F \text{ a formula} \} \cup \{ w \ast_4 (\rho, F) | w \in W, F \text{ a formula} \} \cup \]

\[\{ w \ast_1 (\text{var}_a, p_a) | w + 1 \in [2^\alpha] \text{ and } \text{bin}_a(w) = 1 \} \cup \]

\[\{ w \ast_1 (\text{var}_a, \neg p_a) | w + 1 \in [2^\alpha] \text{ and } \text{bin}_a(w) = 0 \} \cup \]
\( \{ w \ast_1 (rel_r, [R_r]^T) \mid w + 1 \in [2^\alpha] \text{ and } \mathcal{M} \models R_r(bin_0(w), \ldots, bin_a(w)) \} \cup \\
\{ w \ast_1 (rel_r, [R_r]^\perp) \mid w + 1 \in [2^\alpha] \text{ and } \mathcal{M} \not\models R_r(bin_0(w), \ldots, bin_a(w)) \} \cup \\
\{ 2^\alpha \ast_1 (value_{x_a}, [x_a]^T) \mid a \in [k], \mathcal{I}(x_a) = \top \} \cup \\
\{ 2^\alpha \ast_1 (value_{x_a}, [x_a]^\perp) \mid a \in [k], \mathcal{I}(x_a) = \bot \} \cup \\
\{ 2^\alpha \ast_1 (value_{y_a}, [y_a]^T) \mid a \in [k'] \} \cup \{ 2^\alpha \ast_1 (value_{y_a}, [y_a]^\perp) \mid a \in [k'] \} \cup \\
\{ 2^\alpha e \mid e \in \ast\text{Eval} \cup \ast\text{Match} \} \)

Then, \( 2^\alpha \in \mathcal{E}_2(t^\phi, [\neg \phi]^T) \) iff \( S \vdash \ast_2 2^\alpha \ast_2 (t^\phi, [\neg \phi]^T) \). Notice the following: since \( t^\phi \) does not have \( \rho \) as a subterm, the \( \ast \)-expressions in

\( \{ w \ast_3 (\rho, F) \mid w \in W, F \text{ a formula} \} \cup \{ w \ast_4 (\rho, F) \mid w \in W, F \text{ a formula} \} \)

cannot be a part of a derivation for \( S \vdash \ast_2 2^\alpha \ast_2 (t^\phi, [\neg \phi]^T) \).

Since 1 \( \rightarrow \) 2 \( \rightarrow \) 3 and 1, 2 do not interact with any agents in any other way, for any term \( s \) with no \( ! \), if for some \( a \) or \( r \), \( \text{var}_a \) or \( \text{rel}_r \) are subterms of \( s \), if \( S \vdash w \; s :_{a, \chi} \), then \( a = 1, 0 \leq w < 2^\alpha \), and \( \{ w \; e \in S \} \vdash w \; s :_{1, \chi} \). \( t^\phi \) includes exactly one \( !\text{gather}_{r_b} \) for every \( b \) and one of \( \text{value}_z \) for every \( z \in Z \).

Therefore, if \( S \vdash \ast_2 2^\alpha \ast_2 (t^\phi, [\neg \phi]^T) \), then there are

\[ \bigwedge_{b \in \{ \rho \}} !\text{gather}_{r_b} :_{2} \text{gather}_{r_b} :_{1} \Phi \land \text{Match} \land \text{Eval} \land \bigwedge_{z \in Z} \text{val}_z :_{1} [z]^{wz} \vdash t^\phi :_{2} [\neg \phi]^T \]

and by Lemma 6.3.3, \( \mathcal{M} \models \neg \phi \), a contradiction.
On the other hand, let there be some $\mathcal{M}'_J$ where $\phi'$ is satisfied. Then, we name $-1$ a state where $\mathcal{M}'_J, -1 \models \phi'$ and let $-1R_00R_31R_3 \cdots R_32^n$. Then,

- $E_1(var_a, p_a) \subseteq \{ \sigma \in W \mid \sigma + 1 \in [2^n] \text{ and } \text{bin}_a(\sigma) = 1 \}$,
- $E_1(var_a, \neg p_a) \subseteq \{ \sigma \in W \mid \sigma + 1 \in [2^n] \text{ and } \text{bin}_a(\sigma) = 0 \}$,
- $\mathcal{M}_J, 2^n \models \text{Match, Eval}$ and for every $a \in [k']$,
  
  $\mathcal{M}_J, 2^n \models \text{value}_{y_a} : \top[y_a], \text{value}_{y_a} : \bot[y_a]$

as we can see by induction on $\sigma$ - the conditions on $A_1(var_a, p_a), A_1(var_a, \neg p_a)$ as imposed by $\text{forward}_B$, $\text{forward}_C$, $\text{forward}_D$ are positive. Notice here that if for some $0 \leq w < 2^n - 1$, $w \in \bigcap_{a \in a} E_1(var_a, p_a)$, then we have a contradiction: $w + 1 \models \neg[\text{active}]$ and if $w$ is minimal for this to happen, then $w \models [\text{active}]$, so since there is some $a$ s.t. $w \in E_1(var_a, \neg p_a)$, $w + 1 \models [\text{active}]$ (by $\text{forward}_A$).

Then, $\{ w \mid w + 1 \in [2^n] \} \subseteq E_1(rel_r, [R_r]^\top) \cup E_1(rel_r, [R_r]^\bot)$ and then we can define a first-order model $\mathcal{M}$ such that:

- $E_1(rel_r, [R_r]^\top) \subseteq \{ \sigma \in W \mid \sigma + 1 \in [2^n] \text{ and } \mathcal{M} \models R_r(bin_0(\sigma), \ldots, bin_{a_r}(\sigma)) \}$,
- $E_1(rel_r, [R_r]^\bot) \subseteq \{ \sigma \in W \mid \sigma + 1 \in [2^n] \text{ and } \mathcal{M} \not\models R_r(bin_0(\sigma), \ldots, bin_{a_r}(\sigma)) \}$,
• for every $a \in [k]$, $\mathcal{E}_1(\text{value}_{x_a}, [x_a]^\top) \subseteq \{2^\alpha\}$, if $\mathcal{I}(x_a) = \top$ and $\emptyset$ otherwise,

• for every $a \in [k]$, $\mathcal{E}_1(\text{value}_{x_a}, [x_a]^\bot) \subseteq \{2^\alpha\}$, if $\mathcal{I}(x_a) = \bot$ and $\emptyset$ otherwise,

Since it must be the case that $\mathcal{M}, 2^\alpha \not\models t^\phi \vdash_2 [\neg \phi]$, it cannot be the case that

$$\bigwedge_{b \in [\rho_1]} \text{gather}_{r_b} : 2 \text{gather}_{r_b} : 1 \Phi \land \text{Match} \land \text{Eval} \land \bigwedge_{z \in \mathcal{Z}} \text{val}_{z} : 1 [z]^{\text{ex}} \vdash t^\phi \vdash_2 [\neg \phi]^\top$$

and since $\mathcal{M}$ satisfies the conditions from Lemma 6.3.3, $\mathcal{M} \not\models \neg \phi$.

$\square$

Theorem 6.3.1 is then a direct consequence.

### 6.3.2 Justification Logic is harder than Modal Logic

The $\text{NEXP}$-hardness result we presented in this chapter makes $J_H$ the first justification logic with known complexity having a harder satisfiability problem (assuming $\text{EXP} \neq \text{NEXP}$) than its corresponding modal logic. In fact, as Proposition 6.3.5 demonstrates, if $M_H$ is the modal logic which corresponds to $J_H$ (the modal logic with the same frame restrictions as $J_H$), then $M_H$-satisfiability is in $\text{EXP}$: we can simulate the tableau procedure from Table 6.8 using an exponential time algorithm – an alternating polynomial
space one actually, where we use nondeterministic existential choices to apply the tableau rules and universal choices to select exactly one prefix $\sigma.(g, i)$ from $\sigma$ to explore. While Modal Satisfiability has been studied extensively, we are not aware of anyone investigating specifically the complexity of $M_H$-satisfiability; furthermore, there are additional conclusions we can reach from the following analysis; therefore we provide a brief proof.

**Proposition 6.3.5.** Let $M_H$ be the four-modalities modal logic associated with the class of frames $(W, R_1, R_2, R_3, R_4)$ where $R_3, R_4$ are serial, $R_3 \subseteq R_4$, and for $(i, j) \in \{(1, 2), (2, 3), (4, 4)\}$, if $aR_j bR_i c$, then $aR_i c$. Then, $M_H$-satisfiability is in EXP.

**Proof.** We first prove that the tableau procedure from Table 6.8 is sound and complete. From an accepting branch for $\phi$ we can construct a model for $\phi$: let $W$ be the set of prefixes that have appeared in the branch; let $a \in V(p)$ iff $a T p$ has appeared in the branch, let for $i = 1, 2, 3, 4$, $r_i = \{(a, a.(g, i)) \in W \times W\}$, for $i = 1, 2$, $R_i = r_i$, $R_3$ is the transitive closure of $r_3$, and $R_4$ is the transitive closure of $r_3 \cup r_4$. It is not hard to verify that model $\mathcal{M} = (W, R_1, R_2, R_3, R_4)$ satisfies all necessary conditions and that $\mathcal{M}, (0, 0) \models \phi$ – by inductively proving that if $a T \psi$ in the branch then $\mathcal{M}, a \models \psi$ and if $a F \psi$ in the branch then $\mathcal{M}, a \not\models \psi$. 
To test $\phi$ for $M_H$-satisfiability, start from a branch which only contains $(0, 0) \ T \phi$ and keep expanding according to the rules above. A branch with $\sigma T \psi$ and $\sigma F \psi$ is propositionally closed. A (possibly infinite) branch which is not propositionally closed, but is closed under the rules is an accepting branch.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma T \Diamond_i \psi \quad \frac{\sigma T}{\sigma(g,i) \ T \psi}$</td>
<td>$\sigma T \Diamond_i \psi$ when $(g,i)$ is new; where $(g,i)$ has already appeared and $i &lt; 4$;</td>
</tr>
<tr>
<td>$\sigma F \Box_i \psi \quad \frac{\sigma F}{\sigma(g,i) \ F \psi}$</td>
<td>$\sigma T \Box_i \psi$ when $(g,i)$ is new; where $i \in {3, 4}$;</td>
</tr>
<tr>
<td>$\sigma T \Box_i \psi \quad \frac{\sigma T}{\sigma T \Box_j \psi}$</td>
<td>$\sigma T \Box_i \psi$ where $0 &lt; i &lt; j &lt; 4$;</td>
</tr>
<tr>
<td>$\sigma F \Box_i \psi \quad \frac{\sigma F}{\sigma(g,4) \ F \psi}$</td>
<td>$\sigma F \Box_i \psi$ where $(g,i)$ has already appeared and $i \in {3, 4}$;</td>
</tr>
<tr>
<td>$\sigma T \Box_i \psi \quad \frac{\sigma T}{\sigma(g,i) \ T \psi}$</td>
<td>$\sigma T \Box_i \psi$ where $(g,i)$ has already appeared and $i \in {3, 4}$;</td>
</tr>
<tr>
<td>$\sigma T \Diamond_i \psi \quad \frac{\sigma T}{\sigma(g,i) \ T \psi}$</td>
<td>$\sigma T \Diamond_i \psi$ where $(g,i)$ has already appeared and $i \in {3, 4}$;</td>
</tr>
</tbody>
</table>

Table 6.8: Tableau rules for $M_H$.

On the other hand, from a model $\mathcal{M} = (W, R_1, R_2, R_3, R_4)$ for $\phi$ we can make appropriate nondeterministic choices to construct an accepting branch for $\phi$. We map $(0, 0)$ to a state $w^{(0,0)}$ such that $\mathcal{M}, w^{(0,0)} \models \phi$; then, when $\sigma(g,i)$ appears first, it must be because of a formula of the form $\sigma T \Diamond_i \psi$ (or $\sigma F \Box_i \psi$, but it is essentially the same case). If $\mathcal{M}, w^\sigma \models \Diamond_i \psi$, then there must be some state $w^\sigma R_3 w$, such that $\mathcal{M} \models \psi$ and thus we name
It is not hard to see that we can make such choices when applying the rules, so that if \( a T \psi \) in the branch then \( \mathcal{M}, w^a \models \psi \) if \( a F \psi \) in the branch then \( \mathcal{M}, w^a \not\models \psi \). In fact the rules of Table 6.8 preserve this condition right away; we just need to make sure that the same thing happens with the propositional rules – for instance, rule \( \frac{\sigma T \psi \lor \chi}{\sigma T \psi | \sigma T \chi} \) can make an appropriate choice depending on whether \( \mathcal{M}, w^\sigma \models \psi \) or \( \mathcal{M}, w^\sigma \models \chi \). Thus the constructed branch cannot be propositionally closed.

What remains is to show that this tableau procedure can be simulated by an alternating algorithm which uses polynomial space – thus \( M_{H \text{-satisfiability}} \) is in \( \text{APSPACE} = \text{EXP} \). This can be done by applying the following method: always keep the formulas prefixed by a certain prefix \( \sigma \) in memory (at first \( \sigma = (0, 0) \)). First apply all the tableau rules you can on the formulas prefixed by \( \sigma \) – possibly use existential nondeterministic choices for this. Then, using a universal choice, pick one of the prefixes \( \sigma' = \sigma.(g, i) \) that were just constructed and replace the formulas you have in memory by the ones prefixed by \( \sigma' \). Repeat these steps until we either have \( \sigma T \psi \) and \( \sigma F \psi \) in memory or we see “enough” prefixes. In this case, “enough” would mean “more than \( 2^{6|\phi|} \)”, as \( \phi \) has up to \( |\phi| \) subformulas, so in a branch there can only be up to \( 6|\phi| \) formulas prefixed by some fixed \( \sigma \) – thus the algorithm only needs to use \( O(|\phi|) \) memory and if it goes through \( 6|\phi| + 1 \) prefixes, then two of these
have prefixed exactly the same set of formulas. If the algorithm accepts $\phi$, then we can easily reconstruct an accepting branch by just taking the union of the constructed formulas, while if there is an accepting branch, then the algorithm can explore only parts of that branch.

As for the diamond-free fragment of $M_H$, we observe the following. For a formula $\phi \in L^4_M$ in NNF and without any $\lozenge$, let the translation to a $\phi' \in L^2_M$, be such that in $\phi'$, every (maximal subformula of the form) $\Box_1\psi$ or $\Box_2\psi$ is replaced by $\top$, $\Box_3$ by $\Box_1$, and $\Box_4$ by $\Box_2$. Then, it is not hard to see that $\phi$ is $M_H$-satisfiable if and only if $\phi'$ is $D^{\oplus \subseteq} \Diamond 4$-satisfiable: if $(W, R_1, R_2, R_3, R_4, V)$ is a model for $\phi$, then $(W, R_3, R_4, V)$ is a model for $\phi'$ (by induction on $\phi$); if $(W, R_1, R_2, V)$ is a model for $\phi'$, then $(W, \emptyset, \emptyset, R_1, R_2)$ is a model for $\phi'$. This brings us to the following proposition.

**Proposition 6.3.6.** The diamond-free fragment of $M_H$ is in PSPACE.

Notice that Proposition 6.3.6 is a counterexample for the generalization of Proposition 5.2.2 for logics with Verification: diamond-free $M_H$ is in PSPACE, but $J_H$ is NEXP-complete – a even more remarkable gap in complexity.
Chapter 7

Ending Remarks

**What we did:** By extending and generalizing Yavorskaya’s two-agent LP [Yav08], we introduced a family of multi-agent justification logics with two types of interactions among the agents. Our purpose was to provide a general framework capable of modeling situations of multiple agents of different cognitive abilities and interdependencies, in a setting where we are also interested in the agents’ justifications.

We examined the complexity of the resulting logics; we discovered that when no agent is based on a logic of consistent beliefs (JD and JD4), there is not much difference in the complexity of the logic (or the methods to determine it). On the other hand, for the remaining cases we discovered that we can achieve diverse levels of complexity, a surprising phenomenon, as all known single-agent logics are $\Sigma^p_2$-complete.

We were able to completely determine the complexity of the logics for
the case of exactly two agents and for the case where we only consider one interaction, Conversion. We gave a general upper bound by proving that all logics in the system are in \( \text{NEXP} \) and we proved it tight by providing a \( \text{NEXP} \)-complete justification logic – perhaps the most surprising result of this thesis.

We observed a close connection between these logics and the diamond-free fragments of their corresponding modal logics. On the other hand, the \( \text{NEXP} \)-complete justification logic has a higher complexity than any modal logic that uses these interactions (and many more).

At the same time we made observations about the complexity of single-agent Justification Logic. We were able to determine the complexity of \( JD4 \) and to improve on the required conditions for the general lower bound of \( \Sigma_2^p \)-hardness by Milnikel [Mil07] and Buss and Kuznets [BK12]. At the same time we observed the need of tableau procedures based on Fitting semantics instead of Mkrtchian semantics when we have agents of consistent beliefs.

Figure 7.1 presents the whole picture.

**What must be done:** The class of logics we have identified to remain in the second level of the Polynomial Hierarchy is wide, but by no means complete. It would be good to have a characterization in the form of eas-
ily identifiable properties or an efficient algorithm to help us identify the complexity of a justification logic.

Another important open question is the complexity of justification logics with Negative Introspection (see [Art08, Pac05, Rub06a, Rub06b, Rub06d, Rub06c]); the Justification Logic version of Negative Introspection uses an extra operator (?) and is $\neg t : \phi \rightarrow ? t : t : \phi$. As Chapter 2 demonstrates, Negative Introspection plays an important role for Modal Logic, characterizing its computationally more tractable cases. On the other hand, in Justification Logic we do not know anything about the satisfiability problem of logics with Negative Introspection, with the exception of Studer’s decidability result for $J_5$, $J_45$, $JT_5$, and $JT_{45}$ under a finite constant specification [Stu13]. The problem here lies with the justifications and not the constructed frames as in many cases\(^1\) we have seen. In short, the $\ast$-calculus does not help us as much as for the other logics, as the closure conditions for the admissible evidence functions are nondeterministic; i.e. they force a choice between $t : \phi$ and $? t : \neg t : \phi$.

\(^1\)It is interesting to note that Positive and Negative Introspection are equivalent in Modal Logic with respect to their diamond-free fragments: simply compare the class of frames for the diamond-free fragments of each logic.
What can be done: It would be plausible to consider logics with other kinds of interactions as well. For example, a generalization of negative introspection would be interesting from a technical point of view. As Modal Logic is a vast field, Justification Logic has many options as to how to expand. It would be good to identify those areas that can benefit from Justification Logic and especially from the study of its complexity.
Figure 7.1: The complexity of Multi-Agent Justification Logic.
Bibliography


