RWTH Aachen University

Bachelor Thesis

Translation-Based Theorem Proving for First-Order Modal Logic

Translationsbasiertes Beweisen von Theoremen
für die Modallogik erster Stufe

Hannah Louise Wilk
Student ID: 443879

October 21, 2024

Communicated by Prof. Dr. Gerhard Lakemeyer

Submitted to: Knowledge-Based Systems Group
Chair for Computer Science 5 (Information Systems)
Department of Computer Science, RWTH Aachen University

First Supervisor: Prof. Dr. Gerhard Lakemeyer
Second Supervisor: Prof. Dr. Shakil M. Khan (University of Regina, Canada)

Advisor: Qihui Feng, M.Sc.

Contents

1 Introduction

Motivation

Modal logics are extensions of traditional logical systems that introduce operators representing modalities, such as necessity, possibility, knowledge, and belief. These logics enable the formalization and analysis of statements about what agents know or believe and what is possible or necessary within certain contexts. In the realm of epistemic modal logic, reasoning is centered around mental states - what agents know or believe about themselves, about the environment, and about other agents. This feature makes modal logics indispensable for understanding and modeling multi-agent systems where coordination, communication, and interaction play a critical role.

Over the years, epistemic reasoning has garnered substantial interest due to its widespread applicability. It is foundational to understanding distributed system protocols, where agents need to reason about the knowledge states of others to ensure the correctness of communication and computation (Halpern and Moses, 1990). Another key application is in epistemic planning, where the planning process involves reasoning about the beliefs and knowledge of multiple agents, such as in collaborative robotics or game-theoretic scenarios (Belle et al., 2023). Similarly, it is crucial in the development of knowledge-based systems, where logical reasoning guides decisionmaking (Levesque and Lakemeyer, 2001). These examples illustrate how epistemic modal logic provides a formal language to articulate and solve complex multi-agent problems.

Following Kripke's possible-world semantics, modal logics extend the propositional framework by incorporating multiple possible worlds, each representing a different perspective or scenario (Kripke, 1963). This semantics provides an intuitive graphical interpretation of different modal properties, which makes it particularly useful in representing and analyzing various aspects of human reasoning (Blackburn et al., 2001). For instance, modal systems like K45 for belief and S5 for knowledge can be precisely characterized by Kripke frames that satisfy particular relational properties, namely transitivity and Euclideanity for K45 and transitivity, Euclideanity, and reflexivity for S5 (Kramer and Sack, 2012).

Problem Statement

The complexity of reasoning tasks in modal logic, specifically the problem of satisfiability (SAT), presents significant challenges, particularly as we move from propositional to more expressive logical systems. Satisfiability checking in modal logic requires determining whether there is a model (a set of possible worlds and accessibility relations together with a valuation) that makes a given formula true. While the SAT problem for propositional first-order logic is well known to be NP-complete (Cook, 1971), modal logic extends beyond propositional reasoning by involving multiple possible worlds. The introduction of modalities increases the complexity of verification. When multiple agents are involved, each with their own knowledge and interacting modalities, the complexity increases even further (Halpern and

Moses, 1992). This increase in complexity is attributed to the need to evaluate nested and interdependent modalities, resulting in a potentially vast number of possible combinations of worlds and epistemic states.

Current State of Research

To address these challenges, researchers have explored multiple approaches to modal reasoning. There are approaches involving direct methods for (single-agent) modal logics, such as using tableaux calculi (Fitting, 1972) or modal resolution techniques (Enjalbert and Del Cerro, 1989). These methods aim to establish satisfiability by directly manipulating the modal formulas and constructing models that either verify or refute their validity. However, these direct methods can suffer from scalability issues when applied to larger, more complex multi-agent systems (Massacci, 1998).

An alternative methodology is to translate modal formulas into more well-studied frameworks, such as first-order logic (FOL), thereby enabling the utilisation of well-established automated theorem provers. First-order logic has been the subject of extensive study, and efficient SAT-solvers and theorem proving tools such as VAMPIRE (Kovács and Voronkov, 2013) have been developed over several decades.

The initial exploration of translating modal logic into first-order logic was conducted by van Benthem (1977) and Morgan (1976), who showed that it is possible to embed modal systems, such as K, into specific well-defined subsets of FOL. Among these subsets are the guarded fragment (GF) and the two-variable fragment (FO^2). These fragments are particularly appealing because their intersection possesses the complexity bounds of EXPTIME, which ensures that the satisfiability problem remains decidable (Ganzinger et al., 1999, Grädel, 1999). The translation approach aims to utilise existing tools in a way that extends their benefits to the domain of modal logics. However, this translation is not without its own set of challenges. Certain properties inherent to modal systems, such as transitivity and Euclideanity of accessibility relations can be difficult to represent within the confines of classical FOL, which is why embeddings into higher-order logic have also been explored (Benzmüller and Paulson, 2008). Furthermore, the increased formula size and complexity arising from these translations can also significantly impact the performance of automated reasoning tools.

Morgan (1976) distinguishes between semantical and syntactical translation approaches. The most renowned semantical translation is arguably the standard translation as proposed by van Benthem (1984), which, however, also has the problems described above. To overcome these limitations, a variety of modifications and enhancements to the standard translation have been suggested - Auffray and Enjalbert (1992), for example, propose translating modal logic into a typed first-order "Path Logic," allowing modal systems to be characterized by equations and theorem proving to be performed using resolution or rewriting techniques. Nonnengart (1993) aims to overcome the problem of exponential growth in formula size in relational translation approaches by proposing a hybrid functional and relational method. Ohlbach and Schmidt (1997) propose a functional translation using an accessibility function f, Demri and De Nivelle (2005) present a method for translating regular grammar logics with converse.

We are seeing that each of these approaches has a potential drawback: Auffray and Enjalbert (1992) lift the logic to a many-sorted one and do not give a concrete proof that a translation using their approach will remain in the decidable fragment. Nonnengart (1993) introduces a second-order quantification as part of their translation. A potential drawback of the approach of Ohlbach and Schmidt (1997) is that the introduction of nested functional terms and complex quantifier manipulation can lead to increased

computational complexity, particularly in systems with multi-modal or quantified modal logics. For the approach presented by Demri and De Nivelle (2005) it is not very clear how it can be applied for multi-agent systems.

Furthermore, we notice that most translation-based approaches focus primarily on the translation of single-agent languages. Moreover, it is notable that the majority of translation approaches aim to simulate and axiomatise Kripke structures. This methodology often encounters difficulties when attempting to encapsulate frame conditions like transitivity. It thus becomes pertinent to question whether translation approaches based on Kripke structures and Kripke's possible-world semantics should be regarded as the default means of translation.

Research Goals

Qihui Feng developed and proved the satisfiability preservation of a translation approach for single-agent systems that draws on the set-based possible-world semantic. The principal objective of this thesis is to develop a multi-agent extension of Feng's translation approach for single-agent systems and with this to provide an efficient approach to theorem proving in first-order multi-agent modal systems. Our research goal following this is to demonstrate that the translation function we propose preserves the satisfiability of the formula that it is applied to. The second objective of this thesis is to demonstrate that the translation function developed can be utilised in a practical setting.

Contributions

In pursuing our research objectives, we contribute to the field of translation-based approaches to theorem proving for modal logics in the following ways: We are providing an overview of the main translation-based and non-translation-based approaches for single-agent epistemic systems. Furthermore, we demonstrate that our translation function for multi-agent epistemic systems, which is based on Feng's translation approach for single-agent systems, can be used to translate first-order modal formulas into satisfiability-preserving formulas of first-order logic, which can then be processed by theorem provers. This allows us to reason in logics pertaining to epistemic sentences in multi-agent systems, which is particularly significant given that the majority of existing literature in this field of research focuses on single-agent systems. Thirdly, our proposed translation function can in fact be used in a practical setting. This will be demonstrated by means of a description of the methodology for implementing it and the presentation of pseudocode for an implementation.

Thesis Structure

The remainder of this thesis is structured as follows: Chapter 2 provides an overview of the related work, including modal logic, deductive reasoning, as well as previous approaches for modal reasoning, including both translation-based and non-translation-based techniques. Chapter 3 presents the first-order modal logic $\mathcal{KL}_n$. Chapter 4 introduces the translation function $\boldsymbol{R}[\cdot]$ for single-agent systems, along with its extension to multi-agent systems and our proof for the satisfiability preservation of the function. Chapter 5 evaluates the developed translation function and discusses its advantages. Chapter 6 concludes the thesis with a summary of the work presented and an outlook.

2 Related Work

This chapter provides an overview of the relevant research in modal logic and automated reasoning techniques, with a particular focus on the core components that underpin this thesis. We commence with an exposition of the essentials of modal logic, including the Kripke model for knowledge and the formal language, $\mathcal{L}_n$, which employs a possible-world semantics. This provides the basis for reasoning about knowledge and belief in multi-agent systems.

Subsequently, we examine deductive reasoning methods, including key approaches such as SAT and SMT solving, which are instrumental in determining the satisfiability of logical formulas. Furthermore, we examine tools such as SAT- and SMT-solvers and theorem provers, with a particular focus on the VAMPIRE theorem prover, which is widely employed in automated reasoning for first-order logic.

In a next step, we direct our attention to approaches that have been specifically designed for modal reasoning. We commence by analysing the complexity of reasoning tasks in the context of (modal)logic, which presents a considerable challenge in multi-agent scenarios. Subsequently, we examine translation-based methodologies, including both semantic and syntactical translations, which aim to convert modal formulas into first-order logic, making them accessible to well-established theorem proving techniques. Finally, we review non-translation-based approaches, such as tableau methods and resolution methods, which offer alternative strategies for modal reasoning that do not require translation.

This review sets the groundwork for the developments presented later in this thesis, particularly our proposed translation function for multi-agent systems.

2.1 Modal Logic

Modal logic extends propositional logic by introducing two additional connectives that allow for the expression of various truth modalities. These two new connectives are $\Box$ and $\Diamond$. Following Huth and Ryan (2008), the formulas of basic modal logic ϕ can be built using the following Backus-Naur form (BNF):

$$\phi ::= \top|\bot|(\neg\phi)|(\phi \wedge \phi)|(\phi \vee \phi)|(\phi \rightarrow \phi)|(\phi \leftrightarrow \phi)|(\Box\phi)|(\Diamond\phi)$$

For $\Box\phi$ to be true at some state x, ϕ must hold in all worlds that are accessible from x via a binary relation R. For $\Diamond$ to be true at x, there must be at least one state accessible from x where ϕ holds.

Modal logics have emerged as powerful tools for specifying and verifying multi-agent systems (MAS). These logics can capture temporal, epistemic, and strategic properties of agents (Jamroga and Penczek, 2010). Since we want to be able to translate the formulas of an epistemic modal language, denoted as $\mathcal{KL}_n$, for reasoning about multi-agent systems (MAS) into first-order logic, we need to find a framework for reasoning about knowledge. To this end, we employ possible-world semantics, as outlined in the

seminal works of Hintikka (1962) and Moore (1985) to model knowledge.

This section will present the standard Kripke model, which is based on a possible-world semantics. Furthermore, we will present the language $\mathcal{L}_n$, as proposed by Halpern and Moses (1992) for reasoning about knowledge.

2.1.1 Standard Kripke Model for Knowledge

Kripke's Possible-World Semantics

According to Kripke's idea of possible worlds semantics for modal logic, there is, in addition to the true state of affairs (the *actual world*) a set of alternative states (*worlds*) the agent can consider as possible (Fagin and Vardi, 1985). To illustrate, the precise weather conditions in Germany on a given day are uncertain. However, it is possible to eliminate certain regions as potential candidates based on established temperature thresholds. For instance, states with temperatures below -30°C or above +50°C can be excluded from consideration. The remaining states are those that remain plausible and are therefore still under consideration. To represent epistemic concepts, as we want to in this thesis, using Kripke models entails the adoption of a philosophical stance with respect to those concepts (Rendsvig et al., 2024). Knowledge can be interpreted in terms of informational indistinguishability between possible worlds (ibid.).

In considering two distinct atoms, designated as a and b, and focusing solely on these two atoms, it becomes evident that there are four potential outcomes for a possible world:

w_1: $a = true, b = true$;

w_2: $a = true, b = false$;

w_3: $a = false, b = true$;

w_4: $a = false, b = false$ (ibid.).

The set of possible worlds is thus $\mathbf{W} = \{w_1, w_2, w_3, w_4\}$. We can also identify the valuation function $\mathbf{V}$, that assigns each atom at each world a truth value: $\mathbf{V}(a) = \{w_1, w_2\}$, $\mathbf{V}(b) = \{w_1, w_3\}$. A world can be considered a representation of the actual world of a model. Additionally, there is an accessibility relation, denoted by $\mathbf{R}_a$, which illustrates that one possible world is indistinguishable from another possible world. $\mathbf{R}_a$ is a subset of $\{(w, w')|w, w' \in \mathbf{W}\}$, the set of ordered pairs of possible worlds, and if $(w, w') \in \mathbf{R}_a$ for some $w, w' \in \mathbf{W}$, this can be interpreted as world w pointing to world w' and thus w' being accessible or insdistinguishable from w, denoted as $\mathbf{R}_a ww'$ (ibid.).

An agent i is said to know a fact p, if, at all worlds, i considers possible, p is true. With this, we can define a Kripke model as follows:

Definition 1 (Kripke Model)**.** *A Kripke model for the basic modal language is a tuple* $\mathfrak{M} = (\mathbf{W}, \mathbf{R}, \mathbf{V})$ *such that*

- $\mathbf{W}$ *is a non-empty set.*
- $\mathbf{R}$ *is a binary relation on* $\mathbf{W}$.

- $\mathbf{V}$ *is a function assigning to each proposition letter* p *in* Φ *a subset* $\mathbf{V}(p)$ *of* $\mathbf{W}$*, i.e.,* $\mathbf{V}$*:* $\Phi \rightarrow \mathcal{P}(\mathbf{W})$*, where* $\mathcal{P}(\mathbf{W})$ *denotes the power set of* $\mathbf{W}$
  (Blackburn et al., 2001, p. 17).

Note that Φ is a set of primitive propositions.
The satisfiability conditions are defined as follows:

Definition 2. *Suppose* w *is a state in a model* $\mathfrak{M} = (\mathbf{W}, \mathbf{R}, \mathbf{V})$*. Then we inductively define the notion of a formula* Φ *being satisfied or true in* $\mathfrak{M}$ *at state* w *as follows:*

- $\mathfrak{M}, w \models p$ *iff* $w \in \mathbf{V}(p)$*, where* $p \in \Phi$*,*
- $\mathfrak{M}, w \models \bot$ *never,*
- $\mathfrak{M}, w \models \neg\Phi$ *iff not* $\mathfrak{M}, w \models \Phi$*,*
- $\mathfrak{M}, w \models \Phi \vee \Psi$ *iff* $\mathfrak{M}, w \models \Phi$ *or* $\mathfrak{M}, w \models \Psi$*,*
- $\mathfrak{M}, w \models \Diamond\Phi$ *iff for some* $v \in \mathbf{W}$ *with* $\mathbf{R}wv$ *we have* $\mathfrak{M}, v \models \Phi$*.*

(Blackburn et al., 2001, p.17f.).

Axioms and Modal Logics

As we have seen, the possible worlds semantics of modal logic provides a way of interpreting modal operators in terms of different possible worlds. The accessibility relation $\mathbf{R}$, which connects pairs of worlds, plays a crucial role in defining the behavior of modal operators. Various axioms can be added to modal logic, each corresponding to specific properties of the accessibility relation in a Kripke model, leading to different logics suited to different interpretations of necessity, knowledge, obligation, or time. These axioms ensure that modal logics can adequately represent the dynamics of reasoning in different contexts (Garson, 2024).

This section presents the axioms and corresponding frame conditions for a model $\mathfrak{M} = (\mathbf{W}, \mathbf{R}, \mathbf{V})$ of the modal logics we will use in this thesis, following the presentation of Garson (2024) of Chellas's 1980 original account.

- (**K**) ("basic modal logic" (ibid.)):
  The **K** axiom, $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$, is the most basic axiom of normal modal logic. It corresponds to no specific restriction of the accessibility relation $\mathbf{R}$, meaning that the relation can take any form. With the epistemic modal operator $\boldsymbol{K}$, this axiom can be formulated as $\boldsymbol{K}(A \rightarrow B) \rightarrow (\boldsymbol{K}A \rightarrow \boldsymbol{K}B)$. This axiom implies necessitation, i.e. $A \rightarrow \boldsymbol{K}A$, and distribution $(\boldsymbol{K}A \wedge \boldsymbol{K}(A \rightarrow B)) \rightarrow \boldsymbol{K}B$.
- (**4**) transitivity:
  The **4** axiom, $\Box A \rightarrow \Box\Box A$, adds transitivity to the accessibility relation, meaning that if a world $w_1 \in \mathbf{W}$is accessible from w_0, and w_2 is accessible from w_1 (for $w_0, w_2 \in \mathbf{W}$) then w_1 is also accessible from w_0. In epistemic modal logic, the axiom $\boldsymbol{K}A \rightarrow \boldsymbol{K}\boldsymbol{K}A$ introduces positive introspection.

- (**5**) Euclideanity:
  The **5** axiom, $\Diamond A \rightarrow \Box\Diamond A$, introduces the Euclidean property, meaning that if two worlds $w_1 \in \mathbf{W}$ and w_2 are accessible from a world $w_0 \in \mathbf{W}$, then w_1 and w_2 must be accessible from each other. This property is particularly relevant in S5 logic (which has the axiomatic specification $\{\mathbf{K}, \mathbf{T}, \mathbf{5}\}$), where accessibility is universal — meaning that all worlds are mutually accessible. This is often used in epistemic logic, where it implies negative introspection, i.e., that if an agent does not know A, then they know that they do not know A ($\neg \boldsymbol{K} A \rightarrow \boldsymbol{K} \neg \boldsymbol{K} A$).

- (**D**) seriality:
  The **D** axiom, $\Box A \rightarrow \Diamond A$, imposes seriality, ensuring that for every world w, there exists at least one world accessible from w. This means that seriality guarantees that every world has some "future" or "next" world. In epistemic modal logic, the axiom is $\boldsymbol{K} A \rightarrow\neq \boldsymbol{K} \neg A$, which makes sure that the knowledge of an agent is consistent.

- (**T**) reflexivity:
  The **T** axiom, $\Box A \rightarrow A$, introduces reflexivity to the accessibility relation. Reflexivity ensures that a world can access itself, which is important for epistemic logics, where if something is known (i.e., $\boldsymbol{K} A$ holds), it must also be true in the current world. Without reflexivity, modal logic could not ensure that knowledge corresponds to truth. Reflexivity enables modeling introspective reasoning.

These axioms and their corresponding properties allow modal logic to represent different forms of reasoning in a flexible but rigorous way. The relevant modal logics that we will use in this paper are K45, KD45 and S5.

2.1.2 The Language $\mathcal{L}_n$

Halpern and Moses (1992) have posited that reasoning about knowledge is of central importance when reasoning about protocols in distributed systems or systems with multiple agents. They argue that the use of possible-worlds semantics, as presented by Hintikka (1962), provides an effective formal tool for the customisation of a logic, as it allows to capture different sets of axioms through the implementation of minor changes in the semantics (Halpern and Moses, 1992). The language they propose for formal reasoning about knowledge relies on a S5 possible-worlds semantics. Here we present the language $\mathcal{L}_n$ as proposed in Halpern and Moses (1992).

Syntax

When considering a system with n agents $1, \ldots, n$, and a nonempty set Φ of primitive propositions we want to reason about, Halpern and Moses (1992) define$\mathcal{L}_n(\Phi)$ as the smallest set of formulas that contain Φ which is closed under negation, conjunction and the modal operators $K_1, \ldots, K_2$ (Halpern and Moses, 1992, p. 323). Thus, expressions of $\mathcal{L}_n$ are drawn from the alphabet $\Phi \cup \{\neg, \wedge, (,), K_1, \ldots, K_n\}$, though when building formulas of $\mathcal{L}_n$, parentheses that are not necessary for the readability of the formula can be omitted. As usual, $\phi \vee \psi$ can be used as an abbreviation for $\neg(\neg\phi \wedge \neg\phi)$ and $\phi \rightarrow \psi$ can be written instead of $\neg(\phi \wedge \neg\psi)$. Similarly, $true$ is used as an abbreviation for any valid formula (i.e. $p \vee \neg p$) and

$false$ is used as an abbreviation for $\neg true$. The depth of a formula ϕ $dep(\Phi)$ is used as a measure to capture the depth of nesting of K_i-operators in ϕ:

1. $dep(p) = 0$ for any primitive proposition $p \in \Phi$;
2. $dep(\neg\psi) = dep(\psi)$;
3. $dep(\phi \wedge \psi) = \max(dep(\phi), dep(\psi))$;
4. $dep(K_i\psi) = 1 + dep(\psi)$, for $1 \leq i \leq n$ (ibid.).

Halpern and Moses define a subformula ψ of ϕ as a formula that is a substring of ϕ.

Semantics

The semantics of the language $\mathcal{L}_n$ are based on Kripke structures (Kripke, 1963), the single-agent version of which we presented in Section 2.1.1. We will now take a closer look at the multi-agent version. A Kripke structure for n agents is a tuple $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where "S is a set of states or possible worlds, π is a truth assignment to the primitive propositions of Φ for each state $s \in S$ [...] and $\mathcal{K}_i$ is a binary relation on the states of S for $i = 1, \ldots, n$" (Halpern and Moses, 1992, p. 323). The number of states in S defines the size of M. The truth assignment π assigns true or false to each primitive proposition p in each state s, i.e., $\pi(s)(p) = true$ denotes the situation where π assigns p to be true in state s. A formula ϕ is considered true or false in a pair (M, s), where M is a structure as defined above and s is a state in S. The satisfaction relation is defined as follows:

- $(M, s) \models p$ (for $p \in \Phi$) iff $\pi(s)(p) = true$;
- $(M, s) \models \phi \wedge \psi$ iff $(M, s) \models \phi$ and $(M, s) \models \psi$;
- $(M, s) \models \neg\phi$ iff $(M, s) \not\models \phi$;
- $(M, s) \models K_i\phi$ iff $(M, t) \models \phi$ for all t satisfying $(s, t) \in \mathcal{K}_i$ (Halpern and Moses, 1992, p. 324).

Intuitively, the clause $(M, s) \models K_i\phi$ describes the situation that agent i knows ϕ in world s of structure M exactly when ϕ holds true in all worlds that the agent i still considers possible in s, i.e., all the worlds that are related to s by the relation $\mathcal{K}_i$ of agent i.

Halpern and Moses define a formula ϕ as *valid* if it is true in every state of every structure. Given a structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, they define ϕ to be *valid in M* (written $M \models \phi$) if for every state $s \in S$ $(M, s) \models \phi$ holds (ibid. p.325). A formula ϕ is defined to be *valid with respect to class $\mathcal{M}$ of structures* ($\mathcal{M} \models \phi$) if ϕ is valid in all structures in $\mathcal{M}$ (ibid.). ϕ is *satisfiable* in M if $(M, s) \models \phi$ holds for some $s \in S$ and *satisfiable with respect to $\mathcal{M}$* if there is some structure in $\mathcal{M}$ in which ϕ is satisfiable (ibid.).

As Halpern and Moses (1992) state, the core concept of possible-worlds semantics is that an agent's knowledge is linked to their ability to identify which world they are in. Given any world, we can assign to each agent a collection of worlds that, based on their knowledge, could potentially be the actual world. An agent is said to know ϕ if ϕ holds true in every world within this collection. Conversely, the agent does not know ϕ if there exists at least one world they consider possible in which ϕ is false.

Properties

For the proposed language $\mathcal{L}_n$ the following axioms hold:

- any axiomatisation for propositional logic;
- Necessitation: $\models \phi \Rightarrow \models K_i\phi$ for any formula ϕ of $\mathcal{L}_n$;
- Distribution: $(K_i\phi \wedge K_i(\phi \rightarrow \psi)) \rightarrow K_i\psi$ for formulas ϕ and ψ of $\mathcal{L}_n$;
- Truthfulness: $K_i\phi \rightarrow \phi$, provided that $\mathcal{K}_i$ is reflexive;
- Positive introspection: $K_i\phi \rightarrow K_iK_i\phi$, provided that $\mathcal{K}_i$ is transitive;
- Negative introspection: $\neg K_i\phi \rightarrow K_i\neg K_i\phi$, provided that $\mathcal{K}_i$ is euclidean (ibid.).

Note that the following relationship is true: A relation $\mathcal{K}_i$ is reflexive and euclidean iff $\mathcal{K}_i$ is reflexive, symmetric, and transitive (ibid. p.331). For the proof of these properties the reader is referred to Halpern and Moses (1992).

2.2 Deductive Reasoning

In order to ascertain the validity or satisfiability of a given formula, two fundamental strategies may be employed (Kroening and Strichman, 2016, p.3). The first of these is the model-theoretic approach, which involves enumerating the potential solutions from a finite set of candidates (cf. ibid.). The objective here is the verification of formulas in potential worlds within a Kripke model. The second strategy is the proof-theoretic approach, which utilises a deductive mechanism of reasoning based on an inference system (ibid.). Here it follows that automated deduction is the process of automating formal reasoning based on the principles of logic. The validity of formulas is checked with respect to a class of Kripke models. Given that this thesis concerns a translation function that translates $\mathcal{KL}_n$ formulas into formulas of first-order logic (FOL) with the aim of analysing them for satisfiability or validity, it is appropriate to limit the discussion to the proof-theoretic approach. We will initially examine the process of satisfiability checking and present the concept of SAT modulo theories, with a view to subsequently presenting the theorem proving based on this in the following step.

2.2.1 SAT Solving

The satisfiability problem for propositional logic is arguably the most well-known NP-complete problem (Cook, 1971). The problem is formulated as follows: Given a formula combining some atomic propositions using the Boolean operators $\wedge$, $\vee$ and $\neg$, decide whether we can substitute truth values for the propositions such that the formula evaluates to true. In order for a problem to be solved with the assistance of a tool developed to solve the satisfiability problem (SAT-solvers), it is necessary to model the problem within the framework of propositional logic. The SAT-solver then determines, for a given propositional formula, whether it is satisfiable, and if so, provides a satisfying assignment, as the algorithm used to

determine the satisfiability of a given formula can be used to identify a satisfying assignment. SAT-solvers are frequently employed in different research domains, e.g., for analysis or optimisation, as well as in industry for verification and digital circuit design (Kroening and Strichman, 2016). The process of SAT-solving is comprised of a combination of three distinct techniques, as illustrated in Kroening and Strichman (2016) (p. 33 ff.). The initial technique is enumeration or exploration, whereby the assignments are listed in an incremental manner, one after another. Subsequently, a variable is selected, and a value is assigned based on the chosen heuristic. The second technique is propagation. In the event that a clause within the formula is assigned the value false for all but one of its literals, it follows that the last remaining literal, which is unassigned, must be assigned the value true. Otherwise, the clause would be evaluated to false. The third technique employed in the domain of SAT-solving is resolution. In this technique, the objective is to identify the reason why a particular assignment is unsatisfactory. Once a source of conflict has been identified through resolution, it is possible to exclude this source rather than simply guessing the values of the variables once more. The classification of decision heuristics for assigning values to variables can be divided into two categories: static and dynamic heuristics. Static heuristics include the Jeroslow-Wang-Heuristic or the Naive-Decision-Heuristic, whereby a single computation is performed at the beginning of the process, which then determines the order of the allocation of truth values to variables. In contrast, when employing dynamic heuristics, such as the Dynamic Largest Individual Sum-heuristic (DLIS) or the Variable State Independent Decaying Sum-heuristic (VSIDS), the value on which the order is based must be recalculated in each iteration.

2.2.2 SMT Solving

Since propositional logic is sometimes insufficient for modelling, as it is the case with multi-agent systems (Rabbi et al., 2017), we need more expressive logics and decision procedures for them. To solve the problem for such logics, it is possible to utilise Satisfiability Modulo Theories (SMT) solving, which extends SAT-solving for propositional logic with theories. SMT-solving is a field that focuses on checking the satisfiability of first-order formulas with respect to specific logical theories (Tinelli, 2007). Two main approaches to SMT solving have been identified: eager and lazy. The eager approach involves encoding the problem into propositional logic for solving by SAT solvers, while the lazy approach separates Boolean reasoning from theory reasoning, using specialized procedures for the latter (Tinelli, 2007; Kroening and Strichman, 2016). These approaches are complementary, with each being more efficient for different types of problems (Hadarean et al., 2014).

The objective of this thesis is to present a translation function that converts a $\mathcal{KL}_n$-formula into a satisfiability-equivalent FOL formula. This formula can then be evaluated for satisfiability or validity using an SMT solver or theorem prover. Consequently, we will focus on the eager SMT-solving approach, as the details of the lazy approach are not essential to understanding the following chapters.

The eager SMT-solving approach entails the transformation of a logical formula, for instance, one based on bit-vector arithmetic or equality logic, into a satisfiability-equivalent propositional logic formula. In the case of finite-precision bit-vector arithmetic, this can be achieved by directly transforming the bit-vector arithmetic into a satisfiability-equivalent propositional logic formula (Kroening and Strichman, 2016, p. 142ff.). In order to achieve this, the propositional flattening (or Boolean skeleton of the initial

expression) is constructed, after which a Boolean variable is added for each bit of each term, together with constraints that define the meaning of each sub-expression (ibid.). Once this expression has been created, Tseitin's encoding can be employed to convert the propositional logic formula into a propositional logic formula in conjunctive normal form, after which SAT-solving can be used to check the formula for satisfiability (ibid.). The process of SMT-solving of formulas of equality logic is analogous. By employing Ackermann's reduction, we are able to transform a formula of equality logic with uninterpreted functions (that is, a formula that entails functions whose semantics are not defined and of which we only know that they are functions) into a formula of equality logic without uninterpreted functions (Kroening and Strichman, 2016, p. 142ff.). By employing Bryant's reduction or the sparse method, the formula can be transformed into a satisfiability-equivalent formula of propositional logic, which can then be subjected to SAT solving.

While SAT solving is traditionally used for propositional logic, it can be effectively applied to first-order logic problems through various approaches. One method involves generating SAT instances from first-order formulas when the domain is finite (Huang and Zhang, 2005). Another approach integrates SAT solvers into first-order reasoners, periodically grounding first-order clauses and converting them to propositional format (Schulz, 2017). Incremental translation to SAT during solution search has also proven effective, translating only relevant portions of first-order formulas (C. W. Barrett et al., 2002).

2.2.3 SAT- and SMT-Solvers

SAT- and SMT-solvers have become integral to a range of applications in the fields of artificial intelligence and formal verification (C. Barrett et al., 2021), especially since verification problems can often be reduced to the SAT problem. Significant advancements have been made in the field of SAT solvers over the past decade, driven by annual solver competitions (Balint et al., 2015). These solvers utilise a range of techniques, including both sequential and parallel approaches, to efficiently solve Boolean satisfiability problems (Doijade and Kulkarni, 2014). Modern parallel SAT-solvers aim to exploit multi-core processors, though challenges remain in fully utilizing their potential (Hölldobler et al., 2011). Sequential solvers have also been improved, with techniques enhancing performance over time (Doijade and Kulkarni, 2014). SMT solvers, which determine satisfiability with respect to specific background theories, present a powerful alternative to SAT solvers for certain tasks (C. Barrett et al., 2021). Recent research has integrated SMT solvers into tools like FeatureIDE for variability analyses, with comparisons made between their performance and that of SAT solvers (Sprey et al., 2020). Although SMT solvers demonstrate proficiency in generating explanations for unsatisfiable requests, SAT solvers continue to exhibit superior performance in other analyses (ibid.). These solvers are now widely used as core decision procedures for numerous combinatorial problems, demonstrating their importance in various applications, including software verification (e.g. Microsoft's SMT solver Z3, De Moura and Bjørner, 2008)and analysis (Balint et al., 2015).

2.2.4 Theorem Proving

Theorem proving is a core technique in formal verification, utilized for systematically proving or disproving the correctness of mathematical statements. Unlike SAT-solvers, which are designed to

address the satisfiability problem by determining whether a given propositional formula can be satisfied, automated theorem provers go a step further by operating within a proof system. They are tasked with verifying the truth of a goal formula based on a set of assumptions, using logical reasoning to derive a proof. Theorem proving is, therefore, an essential tool in formal verification, providing rigorous guarantees about the behavior of systems by either proving or disproving given properties (Bledsoe and Henschen, 1985).

Procedures employed by theorem provers can be categorized based on their approach to reasoning. One key distinction lies between methods that work backwards from the goal formula towards the axioms, and those that work forward from the axioms to derive the goal (Bonacina, 2001). Backward reasoning methods, such as tableau methods, connection methods, and matrix methods, trace logical dependencies back from the target theorem to known axioms. These approaches explore potential contradictions or prove consistency by dissecting the logical structure of the theorem (ibid.). On the other hand, forward reasoning methods, like classical resolution and the inverse method, systematically apply rules to the axioms in order to generate intermediate theorems, eventually reaching the goal formula (ibid.). Each approach has its advantages depending on the nature of the logical problem at hand.

The architecture of theorem provers often incorporates decision procedures tailored to individual theories. Two common architectures are Cooperating Decision Procedures, where individual decision procedures communicate discovered equalities to improve overall reasoning and SAT-based Theorem Provers, which leverage SAT solvers to decompose and simplify the proof task. Additionally, theorem provers can be linked to SMT solvers to create hybrid systems that benefit from the strengths of each. This combination improves efficiency in reasoning about equalities, quantified variables, and complex logical constructs, ultimately enhancing the overall capability of formal verification systems (Bonacina et al., n.d.).

While SMT solvers, for instance, specialize in determining the satisfiability of first-order formulas within specific theories, they often rely on highly optimized inference methods to improve efficiency (C. Barrett et al., 2010). Theorem proving, by contrast, can tackle more complex logical structures, handling a broader variety of logic types beyond satisfiability. However, SMT solvers can outperform theorem provers for specific problem classes, particularly when specialized theories are involved (Bonacina, 2001). The integration of theorem proving with model checking and SMT techniques has proven to be a promising avenue for enhancing formal verification systems. Such integrations allow for automatic counterexample generation and can simplify bug detection, addressing limitations inherent to each method when used independently (Kong et al., 2005).

Theorem proving plays a critical role in formal verification by providing a structured method for verifying the correctness of systems through logical proof. While it shares certain characteristics with other verification techniques, such as SAT and SMT, its capacity to handle a broader range of logical structures and integrate with other methods makes it a crucial element in the rigorous verification of complex systems.

2.2.5 Theorem Prover VAMPIRE

VAMPIRE is a high-performance automatic theorem prover for first-order classical logic, developed by Riazanov and Voronkov (2002). In Chapter 4 of this thesis, we will present a translation function $\boldsymbol{R}$ as well as a methodology for implementing it, to show that our translation approach could be used in practice. As the subsequent step after the translation of formulas would be to evaluate the translated formulas using a theorem prover, we will introduce VAMPIRE as an exemplar of a widely-used theorem prover to demonstrate the methodology. The current version (version 2.6, revision 1692) has been implemented in C++ by Andrei Voronkov and Krystof Hoder (Kovács and Voronkov, 2013). We chose this automated theorem prover for this thesis because it is exceptionally fast and has won several trophies in first-order theorem proving CASC, uses a limited-resource strategy, and understands sorts, which is very important for our use since we have a many-sorted logic as a result of our translation (as we will explain further in Chapters 4-6). To use VAMPIRE, we have to write our formula of FOL as a problem of the following form (as presented in Kovács and Voronkov (2013)):

Axioms (of $\mathcal{KL}_n$):	$\alpha \supset K_i\alpha$
	$(K_i\alpha \wedge K_i(\alpha \supset \beta)) \supset K_i\beta$
	$K_i\alpha \supset K_iK_i\alpha$
	$\neg K_i\alpha \supset K_i\neg K_i\alpha$
	$\forall x.K_i\alpha \leftrightarrow K_i\forall x.\alpha$
	$\exists x.K_i\alpha \supset K_i\exists x.\alpha$
Assumptions:	$K_i\alpha \wedge K_i(\alpha \supset \beta)$
Conjecture:	$K_i\beta$

Then we have to write this problem in a way that complies with the TPTP-syntax.

TPTP-Syntax

The TPTP (Thousands of Problems for Theorem Provers) language is a standardized syntax for representing problems and solutions in first-order and higher-order logic (Otten and Sutcliffe, 2010). It includes various formats, such as the untyped first-order form (FOF) and typed first-order form (TFF0) (Blanchette and Paskevich, 2013). Recent extensions include TFF1, which adds rank-1 polymorphism to TFF0, allowing easier processing by existing reasoning tools (ibid.). The TPTP syntax can be used to represent derivations in tableau, sequent, and connection calculi (Otten and Sutcliffe, 2010), but also for modern first-order theorem provers. Kovács and Voronkov, 2013 show the correspondence between first-order logic and TPTP notations as follows (Kovács and Voronkov, 2013, p.5):

- $\bot$ and $\top$ in FOL corresponds to `$false`, `$true` in TPTP notation;
- $\neg\alpha$ in FOL corresponds to `"F` in TPTP notation;
- $\alpha_1 \wedge \ldots \wedge \alpha_n$ in FOL corresponds to α`1 & ...& `α`n` in TPTP notation;
- $\alpha_1 \vee \ldots \vee \alpha_n$ in FOL corresponds to α`1 | ...| `α`n` in TPTP notation;

- $\alpha_1 \rightarrow \alpha_n$ in FOL corresponds to α`1 => `α`n` in TPTP notation;
- $\alpha_1 \leftrightarrow \alpha_n$ in FOL corresponds to α`1 <=> `α`n` in TPTP notation;
- $(\forall x_1) \ldots (\forall x_n)\alpha$ in FOL corresponds to `![x1, ..., xn]:`α in TPTP notation;
- $(\exists x_1) \ldots (\exists x_n)\alpha$ in FOL corresponds to `?[x1, ..., xn]:`α in TPTP notation.

Note that in the TPTP syntax we can use comments. For further details on representing FOL problems in TPTP the reader is referred to Kovács and Voronkov (2013).

Algorithm used by VAMPIRE

When given a problem in TPTP notation, VAMPIRE determines the proof-search options that should be used for the problem and preprocesses the problem (Kovács and Voronkov, 2013, p.9). After preprocessing, the problem is converted into conjunctive normal form and then a saturation algorithm is run on it (ibid.). The result of the calculation, which may include a refutation, is returned. A saturation algorithm uses resolution and superposition. The latter is used for reasoning in equational logic, while the former operates on clauses, which are disjunctions of literals, and aims to prove unsatisfiability by generating an empty clause (Meltzer, 1966). VAMPIRE implements ordered binary resolution with set-of-support strategy and ordered hyperresolution (Riazanov and Voronkov, 2001). Recent advancements in Vampire include the AVATAR architecture, which incorporates SAT or SMT solvers for making splitting decisions in first-order resolution (Reger et al., 2014). This approach builds upon earlier methods that relied on Herbrand's theorem and explicit instantiation of clauses (Meltzer, 1966). The resolution method iteratively applies a single operation on pairs of clauses containing complementary literals, generating new clauses called resolvents. This process continues until an empty clause is derived, proving the original set of clauses unsatisfiable and thus the theorem valid. We will not go further into details on superposition, as we will only formulate problems in FOL and thus will not be using equality logic.

2.3 Approaches for Modal Reasoning

In order to demonstrate the validity of a given formula, two possible approaches can be taken. The first is to transform the formula into a first-order formula using a translation method and then show that the translation is valid. The approach of translating modal logics into first-order logic, specifically aimed at mechanizing such logics, was first introduced by Morgan (1976), who distinguishes between semantical translation and syntactic translation. The second approach is to employ a more direct method and search for an interpretation that falsifies the formula. In the absence of a counterexample, it can be inferred that the formula is indeed valid.

This section will begin by presenting the complexity of formal verification for different logical systems. We will proceed by presenting a number of well-known approaches to semantic translation. We will continue by presenting the syntactic approach to translation, although our focus will beon semantic approaches, since the approach we will present in Chapter 4 is a semantic one. The presentation will then move on to consider two non-translation-based approaches.

2.3.1 Complexity

To present a translation function that enables the conversion of formulas from $\mathcal{KL}_n$ into formulas of first-order logic (FOL), thus allowing the application of automated theorem provers to solve problems involving such formulas, it is crucial to thoroughly understand the complexity of formal verification across different logical systems.

Complexity of SAT for FOL and its Fragments

The complexity of satisfiability (SAT) problems in first-order logic (FOL), as well as in its various fragments, varies considerably depending on the properties of each fragment. For the propositional part, the SAT problem is known to be NP-complete. This result, demonstrated by Cook, 1971, implies that if a polynomial-time algorithm for solving the satisfiability of propositional logic could be devised, it would extend to provide polynomial-time solutions for every other problem in the class NP. This implication illustrates the significance of SAT as a core computational challenge (Halpern and Moses, 1992). For first-order logic, on the other hand, the SAT problem is undecidable.

More specialized fragments of FOL have different complexity characteristics. For instance, guarded fragments of FOL, as studied by Grädel (1999), demonstrate that the satisfiability problem for both guarded and loosely guarded fragments is complete for deterministic double exponential time (2EXPTIME). This reflects the inherent computational complexity when dealing with logics that employ guards to restrict quantifier scopes. However, certain subfragments characterized by bounded variable counts or bounded arity offer a more favorable complexity, being EXPTIME-complete, which means that the computational effort required for satisfiability scales more manageably compared to the broader class of guarded fragments.

In addition to guarded fragments, the two-variable fragment of first-order logic (FO^2) has specific complexity properties worth noting. Over arbitrary structures, the satisfiability problem for FO^2 is NEXPTIME-complete, as established by Grädel et al. (1997). This complexity implies that the satisfiable formulas in FO^2 generally have models that are exponentially sized, reflecting the growth in computational requirements. When restricted to finite trees, the satisfiability of FO^2 is bounded by 2EXPTIME, as demonstrated by Benaim et al. (2013). Moreover, in contexts where the domains are ordered or well-ordered, FO^2 remains decidable for both general satisfiability and finite satisfiability, and the associated complexity remains similar to that of unconstrained FO^2. This was discussed by Otto (2001), highlighting that even with these structural constraints, the complexity does not significantly change.

Complexity of SAT for Modal Logics

In modal logics, the complexity of satisfiability also varies depending on the specific system under consideration. In traditional modal logics, ranging from K to S4, the problem of deciding satisfiability has been shown to be PSPACE-complete (Ladner, 1977). This classification indicates that while satisfiability for these modal systems remains challenging, it is still computationally feasible within polynomial space.

However, for the modal system S5, the satisfiability problem is NP-complete (ibid.). This means that, unlike other modal systems like S4, the complexity of SAT for S5 is not greater than that for propositional

logic or first-order logic.

In situations involving multi-agent modal logics, the complexity becomes more nuanced. As shown by Halpern and Moses (1992), when there are two or more agents involved, the complexity of the SAT problem in these multi-agent scenarios is PSPACE-complete across all cases. This result underlines the increased complexity inherent in modeling interactions between multiple agents, where each agent's knowledge or beliefs must be accounted for, thereby maintaining the need for polynomial space in the computational analysis.

2.3.2 Translation-Based Approaches: Semantical Translation

The concept of semantical translation involves converting the implicit semantic elements of modal logic into an explicit form within the predicate logic version of the corresponding modal logic formula. In approaches of this kind the accessibility relation between possible worlds is often explicitly represented using a binary relation symbol. Each world is associated with a unique element in the domain of discourse, and accessibility relations between these worlds are made explicit in the formula. While this method faithfully reflects the semantics of modal logic, it often results in exponential growth in the number and size of formulas (Nonnengart, 1993). This is because each modal operator introduces a new layer of relations, which increases the complexity of the formula exponentially. In this section we will introduce the standard translation as presented by van Benthem and Blackburn (1988) as a well-known relational approach. The standard translation translates into the GF^2-fragment of FOL and preserves K_n consistency. To deal with frame conditions like transitivity or Euclideanity, other approaches are used. These approaches include the equational approach of Auffray and Enjalbert (1992), the relational-functional approach proposed by Nonnengart (1993) and the optimised functional approach developed by Ohlbach (1991), which will be presented in this section. Finally, we will present the approach of Demri and De Nivelle (2005), which translates modal formulas into the GF^2-fragment of first-order logic, thereby preserving decidability.

Van Benthem and Blackburn's Standard Translation

This section presents the Standard Translation approach presented by van Benthem and Blackburn (1988), which employs Kripke models to facilitate the interpretation and reasoning about modal logics. We will follow the presentation of van Benthem and Blackburn (1988). A Kripke model is a mathematical structure that is used to interpret modal logic. It is defined by a tuple $(W, \{R^m\}_{m \in MOD}, V)$, where W is the domain of quantification, each R^m is an binary accessibility relation on W that captures the notion of one world being accessible from another for each $m \in MOD$ (in our case, we have $MOD = K$), and V is a valuation function that assigns to each propositional variable the set of worlds where it is true: $V : \ P \subset 2^W$ (ibid.). Van Benthem and Blackburn (1988) specify that "every basic modal formula [...]

corresponds to a first-order formula" (ibid.) when using the standard translation approach where:

$$
\begin{aligned}
&ST_x(p) = Px, \text{where } p \text{ is an atomic formula} \\
&ST_x(\bot) = \bot \\
&ST_x(\neg\varphi) = \neg ST_x(\varphi) \\
&ST_x(\varphi \wedge \psi) = ST_x(\varphi) \wedge ST_x(\psi) \\
&ST_x(K\varphi) = \forall y(R^m xy \supset ST_y(\varphi)).
\end{aligned}
$$

The symbol x is used here to indicate the current world that the formula is evaluated from. The symbol y is a newly introduced variable, not previously part of the translation. It is used to represent a world accessible from world x. The translation establishes a correspondence between propositions such as p and unary predicates, where the predicate $P(x)$ holds true if the proposition p is true in world x.

Auffray and Enjalbert's Equational Viewpoint

The translation of modal logic to path logic, proposed by Auffray and Enjalbert (1989), involves converting modal logic into a specially designed typed first-order logic called Path Logic, designed to maintain the semantic structure of modal formulas while enabling efficient automated theorem proving. Their method involves a carefully structured translation that introduces specific types to represent the core elements of modal systems. This approach allows classical modal systems to be characterized by sets of equations, transforming modal theorem proving into classical theorem proving in equational theories. This section will present a detailed explanation of the translation process, as outlined by Auffray and Enjalbert (1992).

The translation process proposed by Auffray and Enjalbert begins by categorizing the objects in the discourse domain, the worlds, and the operators acting on those worlds. To achieve this, they introduce three types of objects in their Path Logic:

- Type D: This type represents the objects in the discourse domain—essentially, the entities that exist in the possible worlds described by the modal system.
- Type W: These are the worlds themselves. In modal logic, possible worlds are implicit in the structure but are made explicit in the Path Logic translation.
- Type A: This type denotes the operators that act on the worlds of type W. These operators capture the accessibility relation between worlds, which is fundamental to modal logic.

In a typical modal logic system, the accessibility relation is expressed semantically as a relation between worlds. However, Auffray and Enjalbert adopt an alternative approach. Rather than introducing a binary relation to represent accessibility (as we have seen in Section 2.1.1), they employ operators from Type A. These operators transform one world into another, thereby encoding the accessibility relation.

The core of the translation process from modal logic to Path Logic involves converting modal operators into applications of these algebraic operators, ensuring that the accessibility relations between worlds are expressed in terms of the manipulation of worlds through operators.

For every modal logic system S, a corresponding equational theory $E(S)$ is defined in Path Logic. This theory consists of equations that reflect the properties of the accessibility relations (e.g., reflexivity, transitivity). An example for this would be the following: In system KT4 (transitive and reflexive), the operators satisfy additional constraints such as $a * 1 = a$ and $a * a^{-1} = 1$, reflecting the reflexivity and transitivity of the accessibility relation.

The translation function T converts any modal formula B into a formula in Path Logic. This is done recursively:

1. Atomic formulas (terms and predicates) in modal logic are mapped directly to their counterparts in Path Logic. For example, a modal predicate $p(x)$ in a world w is translated as a predicate on worlds in Path Logic: $p'(w, x)$.

2. Modal operators $\Box$ and $\Diamond$ are translated into quantifiers over the operators in type A. Specifically:
   - $\Box B$ is translated as $\forall a(w* \Rightarrow T(a))$, meaning that for every accessible world (expressed by applying an operator a to w), the formula B holds.
   - $\Diamond B$ is translated as $\exists a(w * a \Rightarrow T(a))$, meaning there exists an accesible world where B holds.

In Path Logic, worlds are not abstract entities but are explicitly represented by terms of type W, such as expressions like $e!a!b$, where e represents an initial world and a, b, etc., are operators applied successively. For example, the path from world e to a new world $e!a$ is represented as e being transformed by the operator a.

The accessibility relation R from modal logic is expressed as operator applications. For example, the accessibility relation $wR2'$ is represented in Path Logic as $w' = w * a$ for some operator a. This avoids the need for an explicit binary relation, improving both the structure and efficiency of reasoning about worlds. Each modal logic system is associated with specific equations that constrain the behavior of the operators in Path Logic. For instance, in S4, the accessibility relation is both reflexive and transitive. This is captured by defining the operator $*$as associative and ensuring that there is an identity operator 1 such that $w * 1 = w$ for any world w, capturing reflexivity.

An important part of the translation process is dealing with quantified variables in modal logic. Auffray and Enjalbert introduce a specialized version of skolemization, known as strong skolemization, which ensures that the translated formulas maintain the desired properties even in systems like S4, which involve complex operator constraints (e.g., associativity). Strong skolemization helps guarantee that unification procedures in theorem proving remain tractable, even when dealing with potentially infinite sets of unifiers.

Auffray and Enjalbert's translation method provides an efficient and semantically robust way to translate modal logic into Path Logic, a first-order classical logic system. Debart et al. (1992) extended this method to multi-modal logic programming, associating each modal subsystem with a specific sort and set of equations in Path Logic. Schmidt (1997) further developed the optimized functional translation, mapping modal logics to a lattice of clausal logics called path logics. This translation enables the use of resolution and condensing for deciding satisfiability in basic path logic and certain extensions.

Nonnengart's Functional and Relational Approach

Nonnengart proposes a unique translation approach that merges the advantages of relational and functional translations (Nonnengart, 1993). This hybrid method aims to overcome the primary weaknesses of each approach, namely the exponential growth in formula size in relational translation and the complexities of equality handling in functional translation. Nonnengart's approach is particularly effective in serial modal logics, where it simplifies the representation of accessibility relations and leads to more efficient theorem proving. We will present Nonnengarts approach following his work on first-order modal logic theorem proving and functional simulation (ibid.).

Nonnengart proposes a hybrid approach that combines elements of both relational and functional translations. The key idea is to split the accessibility relation into functional simulators for each component of the accessibility relation. This hybrid method has two main components:

The relational component of Nonnengart's approach involves expressing the accessibility relation R in a way that limits the formula's growth. Instead of introducing an entirely new relational symbol R for each modal operator, Nonnengart applies a relational translation that avoids exponential growth by treating accessibility as partially functional. Specifically, the accessibility relation is broken down into functional components, which are treated like partial functions. By arranging the pairs in the accessibility relation in a two-dimensional array, each row of this array corresponds to a subrelation of the accessibility relation that is partially functional. Instead of reasoning directly with the full accessibility relation, Nonnengart's method allows reasoning with these functional subcomponents. This drastically reduces the number of clauses and limits the size of the translated formula.

The second component of the hybrid method is the use of functional simulators. Each partial function that represents a component of the accessibility relation is treated as a functional simulator. A functional simulator is defined as a function that simulates the behavior of a relational accessibility condition, allowing reasoning to proceed as if the accessibility relation were purely functional. The functional simulator is represented by a new function symbol, which simulates the application of the accessibility relation as a function. Nonnengart defines two simulator axioms that govern the behavior of these simulators. These axioms ensure that the simulator behaves correctly with respect to the underlying accessibility relation. However, only one of these axioms is strictly necessary, as the second axiom can be eliminated through simplification. This further reduces the size and complexity of the translated formula.

Nonnengart emphasizes that the hybrid translation approach is particularly effective when applied to serial modal logics, such as the systems KD, KDB, KD4, and S4. In these logics, the accessibility relation is serial (i.e., every world has at least one accessible successor). The simplifications introduced by the hybrid method can be applied in the following ways: For serial logics like KD and KDB, a single unit clause suffices to represent the accessibility relation. In KD, the unit clause R(u,u:x) captures the seriality of the accessibility relation, while in KDB, an additional unit clause representing symmetry is introduced. In transitive logics like KD4 and S4, the hybrid approach simplifies the transitivity axiom by representing it as a two-literal clause. This allows the translated formula to remain compact, avoiding the exponential growth typically associated with relational translation.

In the case of logics with Euclidian properties like KD5 and logics with reflexivity and transitivity like S5, Nonnengart's method proves to be efficient as well. These logics can be represented by a single unit

clause, reflecting the fact that all worlds are accessible from one another. This mirrors the observation of Segerberg (1971) that systems like S5 are characterized by a universal accessibility relation, making the hybrid method particularly suitable for such logics.

The hybrid approach also addresses non-serial modal logics and logics with varying domains. In non-serial logics, the assumption that every world has a successor no longer holds. Therefore, Nonnengart introduces a normality predicate that indicates whether a world behaves "normally" with respect to the accessibility relation. This predicate ensures that only normal worlds are considered in the functional simulation, restoring the completeness of the translation in non-serial contexts.

For logics with varying domains, Nonnengart introduces an existence predicate $E(u, u : x)$ that tracks the presence of domain elements in different worlds. This predicate is functionally simulated in the same way as the accessibility relation, ensuring that the translation remains efficient even when dealing with changing domain structures.

Ohlbach and Schmidt's optimised Functional Translation

While Nonnengart (1993) combines relational and functional approaches, Ohlbach and Schmidt (1997) propose an optimized functional translation approach that aims to address the inefficiencies inherent in traditional relational translations of modal logic into first-order logic. This approach is (like part of Nonnegart's approach) rooted in the functional translation method, where accessibility relations between worlds are represented as functions mapping worlds to accessible worlds. By using functions instead of binary relations, Ohlbach and Schmidt (1997) provides a compact and expressive framework for handling modal logic, especially when dealing with complex modal systems that are not easily translated into first-order logic. In this section, we will present Ohlbach and Schmidt's approach following their 1997 work on functional translation and second-order frame properties of modal logics.

Ohlbach and Schmidt's approach arises from the observation that relational translations of modal logic, while semantically straightforward, often fail to capture the expressiveness needed for certain modal logics—especially those with second-order frame properties. The functional translation method replaces these binary relations with accessibility functions that map a world w to the set of worlds accessible from w. The key idea of Ohlbach and Schmidt's functional translation is the use of accessibility functions f, where $f(w)$ represents the set of worlds accessible from w. The relational translation of modal formulas is replaced by terms involving functional symbols. For instance, the formula $R(w, v)$ is replaced by $apply(f, w) = v$, where f is a function from worlds to accessible worlds.

This allows the relational formula $\Box p$ (stating that p is true in all accessible worlds) to be translated into a functional formula:

$$\Box p(w) \text{ becomes } \forall f(apply(f, w) \rightarrow p(f(w)))$$

Here, $f(w)$ represents the world v accessible from w under the accessibility function f.

Second-order frame properties, such as reflexivity, transitivity, or seriality, are often difficult to express in first-order logic. Ohlbach and Schmidt's translation addresses this problem by demonstrating that certain second-order frame properties can be reduced to first-order formulas using functional representations. For example, the axiom $\Box p \rightarrow p$, which is valid in a reflexive frame, translates relationally to

$\forall w(R(w,w) \rightarrow p(w))$. Using the functional translation, this becomes $\forall f(apply(f,w) = w \rightarrow p(w))$, where $apply(f,w) = w$ expresses the reflexivity of the accessibility function. Ohlbach (1991) state that this first-order functional translation is more compact and easier to handle within automated theorem proving systems because it eliminates the need for explicit second-order quantification over relations.

The translation approach optimizes the functional translation by ensuring that existential quantifiers in the original second-order formula can be moved outside over universal quantifiers. This quantifier exchange is crucial for reducing the complexity of the translated formula and improving the performance of theorem proving systems. In particular, Ohlbach and Schmidt prove that for certain types of modal logics (such as those defined by serial frames), this quantifier exchange preserves the equivalence of the translated formula. This optimization is made possible by the introduction of maximal functional frames, which guarantee that every world in a Kripke frame can be mapped to another accessible world via some function. This allows the translation to maintain soundness and completeness while simplifying the formula's structure. Ohlbach and Schmidt show this using the example $\Box\Diamond p \rightarrow \Diamond\Box p$, which can be translated into $\forall w \exists f \exists g(\forall v(apply(f,w) \rightarrow \exists u(apply(g,v) \rightarrow p(u))))$. The quantifier exchange rule allows for reordering these quantifiers, making the formula more efficient to process during theorem proving.

This functional translation has significant implications for the automation of modal theorem proving. By reducing the need for second-order reasoning, the functional approach makes it possible to use first-order theorem proving techniques on a broader range of modal logics, including those with graded modalities and temporal operators.

Additionally, this method can be extended to handle non-serial modal logics, where not every world has a successor. Ohlbach introduces a way to encode partial functions, ensuring that worlds without successors are handled within the functional framework.

The translation also applies to multi-modal logics, where multiple accessibility relations (or functions) are used to model different types of modalities. This flexibility makes Ohlbachs and Schmidt's approach highly versatile and applicable to a wide range of logical systems.

Demri and de Nivelle's Translation of Regular Grammar Logics with Converse

In their 2005 paper, Demri and De Nivelle present a method for translating regular grammar logics with converse into GF^2, a fragment of first-order logic known as the intersection of the guarded fragment (GF) and the two-variable fragment (FO^2). This approach is designed to handle modal logics where frame properties, such as transitivity and reflexivity, are implicitly maintained through the use of nondeterministic finite automata (NDFA) without needing to be explicitly encoded in the first-order logic itself (ibid.). This method provides a significant advantage by making it feasible to decide the satisfiability problem for such logics in a way that is computationally efficient and avoids undecidability. This chapter will present Demri and De Nivelle's approach following their 2005 paper on decideing regular grammar logics with converse through first-order logic.

Regular grammar logics extend standard modal logics by allowing the accessibility relations between possible worlds to be governed by production rules. These rules define how accessibility relations propagate, and can represent more complex frame conditions than those available in traditional modal

logic, such as transitivity, reflexivity, and symmetry. In addition, the converse of each accessibility relation is also included, meaning that for any accessibility relation R_a (denoted by the modal operator $[a]$), there is a corresponding converse relation $R_{\bar{a}}$ that allows movement backward along the same path. In the context of regular grammar logics, modal formulas must take into account these frame conditions, making a direct translation to first-order logic challenging. If such frame properties are encoded explicitly in the first-order logic, it can lead to undecidability. To avoid this, the translation method proposed by Demri and de Nivelle uses NDFAs to represent the accessibility relations, which allows frame conditions to be preserved implicitly.

The core goal of the translation is to map modal formulas from regular grammar logics with converse into the guarded fragment with two variables (GF^2), a decidable fragment of first-order logic. GF^2 restricts the use of variables and quantifiers in a way that ensures the satisfiability problem remains in EXPTIME, making the logic computationally feasible to handle. This translation is done by encoding the modal operators and their corresponding converse operators into first-order formulas, while using the NDFA to handle the frame conditions.

Translation of Modal Operators:

Consider a modal operator $[a]\phi$, which asserts that ϕ holds in all worlds accessible from a given world w via the relation R_a. In the translation to GF^2, this is encoded by a guarded formula that checks the accessibility of a world w_1 from w_0 through R_a. The translation ensures that if a world w_1 is accessible, then the formula $\phi(w_1)$ must hold. The guarded formula looks like this:

$$\forall w_0 w_1 (R_a(w_0, w_1) \rightarrow \phi(w_1))$$

This formula ensures that the modal operator $[a]\phi$ is faithfully represented in the first-order logic, preserving the same meaning as in the original modal system.

Converse Relations:

One of the essential features of regular grammar logics with converse is the existence of a backward relation for every forward accessibility relation. For every $R_a(x_0, X_1)$, there is a corresponding relation $R_{\bar{a}}$ that represents the converse. The translation accounts for this by adding a condition to ensure that the converse relation holds wherever the forward relation holds. The following first-order formula guarantees this symmetry:

$$\forall w_0 w_1 (R_a(w_0, w_1) \rightarrow R_{\bar{a}}(w_1, w_0))$$

Frame Properties:

A central challenge in translating regular grammar logics is maintaining the frame properties (such as transitivity) without explicitly encoding them in the first-order logic. Demri and de Nivelle solve this by using nondeterministic finite automata (NDFAs) to simulate the frame conditions. The NDFA is constructed to process the accessibility relations according to the production rules that define how accessibility relations can propagate through the modal structure. For example, if a modal logic includes the transitivity condition $a \rightarrow a \cdot a$ which means that a world accessible via a can be reached in two steps, this transitivity is handled by the NDFA, which simulates this behavior without the need for an explicit first-order axiom. The automaton processes the relations and ensures that if there is a path from one

world to another via the relation a, the transitivity property holds implicitly. This allows the translation to remain within GF^2 and avoid the complexities associated with second-order frame conditions.

To illustrate the translation, consider the modal formula $[a]p$, which asserts that for every world accessible via relation a, the predicate p holds. In regular grammar logics with converse, this formula must also account for the backward relation $R_{\bar{a}}$. The translation into GF^2 would proceed as follows:

- The modal formula $[a]p$ is translated into the guarded formula $\forall w_0 w_1 (R_a(w_0, w_1) \rightarrow p(w_1))$ (similarly to the example we introduced earlier).
- The corresponding condition ensuring the symmetry of accessibility would be $\forall w_0 w_1 (R_a(w_0, w_1) \rightarrow R_{\bar{a}}(w_1, w_0))$
- The production rules are handled by the NDFA. The automaton simulates how accessibility relations propagate, ensuring that if there is a path through multiple worlds via relation aa, it behaves as expected. This part of the translation is implicit and does not need to be expressed directly in the first-order logic but is instead encoded in the automaton's transitions.

The result is a decidable formula in the guarded fragment with two variables (GF^2), ensuring that the satisfiability problem remains computationally feasible.

2.3.3 Translation-Based Approaches: Syntactic Translation

Syntactic translation refers to a method of translating modal logics into first-order logic by encoding the syntax of the logic directly into first-order terms. This approach differs from semantic translations, such as relational or functional translations, which focus on converting the modal semantics. Instead, syntactic translation focuses on translating the logical rules and axioms of a modal system into a form that first-order theorem provers can handle (Schmidt and Hustadt, 2013). This section will introduce the concept of syntactic translation, as outlined by Schmidt and Hustadt (2013).

One of the key advantages of syntactic translation is that if a sound and complete axiomatization exists for a modal logic, the logic can usually be embedded straightforwardly into first-order logic. For example, in the case of the modal system S4, which is axiomatically defined with the propositional logic axioms and additional modal axioms such as K, T, and 4, a syntactic translation can represent the modal formulas as terms in first-order logic.

In a Hilbert-style axiomatization, each axiom of the modal system is translated into first-order logic using a special predicate symbol T, which represents the truth or entailment relation. The modal formulas are encoded as terms, and the predicate T is used to indicate when a formula holds. For example, a modal formula A is encoded as $\forall p T(\pi_t(A))$, where $\pi_t(A)$ is a term representing the modal formula, and T asserts that A is true in this translation.

The axioms of the modal logic are converted into universally quantified first-order formulas. Each modal axiom, such as K (the distribution of $\Box$ over implication), T (the reflexivity axiom $\Box p \rightarrow p$), and 4 (transitivity, $\Box p \rightarrow \Box\Box p$), is encoded as a term using function symbols to represent logical operators. For instance, the axiom 4 is encoded using designated function symbols, such as i to represent implication and

b to represent the box operator. Thus, the formula $\Box p \rightarrow \Box\Box p$ is translated to $\pi_t(4) = i(b(p), b(b(p)))$ where i denotes implication and b denotes the box operator. Similarly, the rules of inference, such as modus ponens and necessitation, are encoded as universally quantified first-order formulas. For example, modus ponens is translated as $\forall p_1 p_2((T(p_1) \wedge T(i(p_1, p_2))) \rightarrow T(p_2))$, which captures the rule that if $T(p_1)$ is true and $T(p_1 \rightarrow p_2)$ is true, then $T(p_2)$ must also be true.

Schmidt and Hustadt (2013) state that syntactic translation has been widely used in automated reasoning, particularly in investigations into small axiom bases for algebras and other mathematical problems. The syntactic translation approach is particularly useful in cases where the semantic-based approaches, such as relational or functional translations, are not feasible due to the lack of known first-order semantic characterizations. Since syntactic translation operates purely at the level of formulas and rules, it can handle logics that do not have a clear first-order semantics.

2.3.4 Non-Translation-Based Approaches

Two such direct approaches will be presented here: a tableau-based one, as proposed by Fitting (1972), and a resolution-based one, as presented by Enjalbert and Del Cerro (1989).

Fitting's Tableau-based Approach

Tableau methods for modal logics extend classical tableau techniques to handle modal operators, using branching structures to represent potential worlds or scenarios (Fitting, 1972). They operate directly within the Kripke semantics of modal logic and represent a systematic way of testing the satisfiability of formulas in modal logic by constructing tree-like structures known as L-tableaux (ibid.). These methods provide an alternative approach to translation-based theorem proving, working directly with modal logic formulas to evaluate their validity or find counterexamples. This section follows Fitting (1972) in showing how a tableau proof system for a language $\mathcal{L}$, where $\mathcal{L}$ is one of the logics S5, S4, B, T, DS4, DT, KS4 or KT, can be constructed from a tableau structure for $\mathcal{L}$.

A tableau is, in essence, a tree-like structure used to break down a logical formula in a step-by-step manner. In modal tableau systems, a prefixed formula of the form $P : X$ is employed, where P represents a possible world and X is a logical formula. The process entails positioning $P : X$ at the origin of the tableau and subsequently applying rules to extend branches of the tableau in a systematic manner until either all branches are closed or at least one branch remains open.

The construction of a tableau begins by assuming the negation of the formula you wish to prove. If a formula X is being tested, the tableau starts with $P : \neg X$. The aim is to extend branches by applying rules until a contradiction is found for every branch, which would indicate that the original formula X is valid. The rules for extending branches depend on the type of formula encountered, we can distinguish between α and β rules, rules for modal operators and branch closure.

α and β rules are used on α and β formulas. α formulas represent conjunctive expressions and lead to branch extension without splitting. For instance, if $P : (A \wedge B)$ is encountered, both $P : A$ and $P : B$ are added to the same branch. β formulas represent disjunctive expressions and cause the branch to split. For $P : (A \vee B)$, the tableau splits into two branches: one with $P : A$ and the other with $P : B$.

For modal operators, Fitting (1972) introduces the υ rule for necessity and the π rule for possibility. For a formula of the form $P : \Box X$, the υ rule dictates that for every related world Q, $Q : X$ must also hold. Thus, $Q : X$ is added to the tableau for all worlds Q accessible from P that have been used on the branch or are unrestricted. For a formula of the form $P : \Diamond X$, the π rule allows the introduction of a new world Q, adding $Q : X$ to the branch. This indicates that there must be at least one possible world accessible from P where X is true.

Branch closure is handled as follows: A branch is closed if it contains a contradiction, such as both $P : X$ and $P : \neg X$. A closed tableau is one where every branch contains such a contradiction, thus proving the validity of the original formula. If there is at least one open branch, it indicates a counterexample—an $\mathcal{L}$-Hintikka set—which provides an $\mathcal{L}$-realization showing that the formula is not valid in all models.

The tableau expansion follows a systematic procedure, applying the rules to each branch until all non-atomic formulas are used. Each occurrence of a prefixed formula is systematically extended until either the branch closes or all possibilities are exhausted. If all branches close, the original formula is confirmed to be an $\mathcal{L}$-theorem. If an open branch remains, it indicates that the formula is not valid across all models. Fitting (1972) shows that modal tableau systems are both sound and complete. This guarantees that the method does not produce false positives, and every derived formula is indeed valid in the given logical framework, as well as ensuring that all valid formulas can be derived using this method, making it a reliable approach for determining validity in modal logics.

Enjalbert and Del Cerro's Resolution-Based Approach

Modal resolution is a key strategy for automated deduction in modal logics, inspired by Robinson's (Robinson, 1965) classical resolution method (Enjalbert and Del Cerro, 1989). It has been applied to various modal systems, including K, Q, T, S4, and S5, with completeness proofs based on tableau methods (ibid.). We will present the resolution-based approach by Enjalbert and Del Cerro (1989) following their 1989 paper on modal resolution in clausal form.

Note that Nonnengart (1993) and Auffray and Enjalbert (1992) also use resolution in their approaches presented in Section 2.3.2.

In the 1993 work by Enjalbert and Farinas del Cerro, a resolution-based approach for modal logics is introduced, specifically focusing on extending classical resolution techniques to handle the intricacies of modal operators. The approach is designed primarily for normal modal logics, with the minimal normal system K. The proposed resolution system, called RK, extends classical resolution by adapting it to modal contexts. Classical resolution is a well-established method for proving the inconsistency of a set of formulas in clausal form by iteratively applying an inference rule to produce resolvents until either a contradiction (the empty clause) is derived or no new clauses can be generated. In the modal case, RK similarly aims to show inconsistency but must consider the additional complexities introduced by modal operators. The rules of computing resolvents in the RK system extend classical resolution to accommodate modal formulas. In first-order logic, when two clauses $D_1 = p \vee C$ and $D_2 = \neg p \vee C'$ are resolved, the resolvent is $C \vee C''$. In the context of modal logic, the resolvent must respect the scope of the modal context from which it originates. For instance, given two clauses involving the operator

$\Box$, such as $\Box(\neg p \vee q)$ and $\Box(\neg p)$, the resolution must occur within the modal context represented by $\Box$, producing a new resolvent like $\Box q$. This approach allows classical resolution rules to be effectively applied within the modal scope, maintaining consistency across different possible worlds.

To maintain efficiency and manage the complexity of the resulting clauses, the RK system includes simplification rules. These rules are applied to the computed resolvents to remove any redundancies. For instance, if a clause contains repeated literals or contains unnecessary parts that do not contribute to the inconsistency, these are eliminated. Simplifying the clauses ensures that the resolution process remains tractable and helps in limiting the search space, which can grow rapidly in modal logics. The RK system also includes specific rules for drawing inferences from the computed and simplified clauses. This recursive process of generating new clauses and simplifying them continues until a contradiction is found or no further clauses can be derived.

Clauses originating from different modal contexts cannot always be combined arbitrarily. The correct combination of modal contexts depends on the relationships between the modal operators involved. While $\Box$ and $\Box$ can be combined, as they represent universally true conditions across accessible worlds, $\Box$ and $\Diamond$ may not necessarily be combined without careful consideration of their semantics, as they represent different modalities. Enjalbert and Del Cerro (1989) have demonstrated that the RK system can be extended to more complex modal systems such as T, S4, and S5. They have also established that the RK system is both sound and complete.

3 First-Order Modal Logic $\mathcal{KL}_n$

This chapter introduces the logic $\mathcal{KL}_n$, a first-order modal logic for multi-agent systems, which extends the work of Levesque and Lakemeyer (2001). The logic $\mathcal{KL}_n$ provides a formal framework for reasoning about knowledge in systems where multiple agents interact and hold potentially different beliefs about the world and each other. It builds upon set-based possible-world semantics, offering a more refined approach to capturing the complexity of epistemic states in multi-agent environments.

We begin by discussing the set-based possible-world semantics, which form the foundation for modeling knowledge in $\mathcal{KL}_n$. This semantics defines a world as a set of ground atoms and introduces epistemic states that represent the worlds an agent considers possible. These concepts are extended to handle the nested knowledge typical in multi-agent settings, where agents may hold beliefs about the beliefs of others.

Next, we present the formal structure of $\mathcal{KL}_n$, including its syntax and semantics, detailing how truth values are assigned to formulas based on the worlds and epistemic states of agents.

Finally, we outline the key properties of $\mathcal{KL}_n$ such as the K45$_n$ axioms and the validity of the Barcan formulas, demonstrating that this logic is well-suited for reasoning in multi-agent systems. The chapter concludes with formal proofs of these properties, showing that $\mathcal{KL}_n$ maintains the expected logical behavior for knowledge operators in multi-agent scenarios.

3.1 Set-Based Possible-World Semantics

Levesque proposed a semantics for possible worlds based on sets (Levesque, 1981). Following his semantics, a world is a set of ground atoms. In the context of a single-agent system, a situation is identified by a pair (W, w) with w being a possible world, namely, the actual world, and W denotes the set of possible worlds that the agent still considers possible (Halpern and Lakemeyer, 2001). It should be noted that the set of worlds considered possible does not necessarily include w (ibid.). Additionally, the set of worlds W does not have to be nonempty (ibid.). As in the logic $\mathcal{L}_n$, which was presented in Section 2.1.2, we can describe a possible world in terms of the sentences that are true in that world. This implies that two worlds that satisfy the same sentences are to be regarded as equivalent (Levesque and Lakemeyer, 2001).

An agent is considered to know (at least) α if α holds true in all worlds considered possible by that agent (ibid.). Using the modality L, where $L\alpha$ can be understood as "knowing at least α", Halpern and Lakemeyer (2001) define the semantics as follows:

- $(W, w) \models p$ if $w \models p$ if p is a primitive proposition;
- $(W, w) \models \neg\alpha$ if $(W, w) \not\models \alpha$;

- $(W, w) \models \alpha \vee \beta$ if $(W, w) \models \alpha$ or $(W, w) \models \beta$;
- $(W, w) \models L\alpha$ if $(W, w') \models \alpha$ for all $w' \in W$.

The addition of knowledge to the field of set-based possible-world semantics allows us to posit that "what a knower cares about is the way the world is, that is, the possible world that is actual" (ibid., p. 49). Thus, when acquiring new information, the knower may be able to exclude certain possibilities regarding the actual world. At some point, the knower may have been able to eliminate all but a single possible world, which would then be regarded as the actual world.

Halpern and Lakemeyer (2001) introduce the notion of an epistemic state e that can be represented by the set of potential worlds that have not been excluded by the knower as the actual world. This can be denoted as a set of world states. If the knowledge of the agent is complete, then the epistemic state contains only one world and for every formula it holds that if it is not "known to be true[, it] is known to be false" (ibid., p. 50), though this does not mean that the knowledge is accurate, i.e., that the one world in the epistemic state corresponds to the real world state. An agent may also have knowledge about the epistemic state. To deal with this, Halpern and Lakemeyer (2001) use the notion of a possible universe that consists of a world state as well as an epistemic state. They stipulate that only one world state and one epistemic state can be actual at any given time (ibid.). This means that we extend the notion of epistemic states to involving "a set of possible universes, each consisting of both a world state and an epistemic state" (ibid., p. 51). Considering these notions, it is clear that any objective sentence, in addition to any subjective sentence, can be regarded as known if the sentence is true in all potential universes.

3.2 The Logic $\mathcal{KL}_n$

The logic $\mathcal{KL}_n$ defines a mathematical model of knowing. It can be regarded as a multi-agent extension of the logic $\mathcal{KL}$ as specified by Levesque and Lakemeyer (2001) or as a fragment of the logic $\mathcal{OL}_n$ (specified by Belle and Lakemeyer (2015)), not mentioning the only-knowing.

Syntax

The expressions of $\mathcal{KL}_n$ are constituted by sequences of symbols drawn from the following two sets (cf. Levesque and Lakemeyer, 2001): Firstly, the logical symbols, which consist of the logical connectives and quantifiers $\exists, \vee, \neg$, as well as punctuation and parentheses. Furthermore, it compromises a countably infinite supply of first-order variables, denoted by the set $\{x, y, z, \ldots\}$, a countably infinite supply of standard names, represented by the set $\{\#1, \#2, \ldots\}$, and the equality symbol =. The second set is that of the non-logical symbols. It includes predicate symbols of any arity $\{P, Q, R...\}$, which are intended to represent domain-specific properties and relations. Additionally, it contains function symbols of any arity, which are used to denote mappings from individuals to individuals. Furthermore, it comprises a finite set of agents $\mathsf{Ag} = \{1, \ldots, n\}$. Finally, it includes a modal operator for knowledge K_i with $i \in \mathsf{Ag}$.

The language of $\mathcal{KL}_n$ is composed of two distinct types of expressions: terms and well-formed formulas (wffs). The terms of the language are employed to describe individuals within the application domain. They constitute the smallest set of expressions such that

1. every first-order variable is a term;

2. every standard name is a term;

3. if t_1,...,t_j are terms and f is a function of arity j, then $f(t_1, ..., t_j)$ is a term.

A ground term is defined as a term with no variables, entailing only standard names. A primitive term is a ground term with only a single function symbol, thus not having any nested functions. In the following, the set of all ground terms, denoted by $\mathcal{G}$, will be used. Wffs are employed to delineate relations, properties, or conditions within the domain of application. An atomic formula is defined as a formula $P(t_1, ..., t_k)$ where P is a predicate and t_i are terms. A ground atom is defined as an atom that does not contain variables. A primitive atom is a ground atom where each of the terms t_i is a standard name. The set of well-formed formulas is the smallest set that includes

1. atomic formulas;

2. $(t_1 = t_2)$ with t_1 and t_2 being variables or standard names;

3. if α, β are formulas, then $(\alpha \vee \beta)$, $\neg\alpha$, $\exists x(\alpha(x))$;

4. if α is a formula, then $K_i(\alpha)$ is a formula.

Note that in this thesis we might omit the outermost parantheses when the meaning is clear. The symbol $K_i(\alpha)$ is read as "α is known by agent i". The abbreviations $\alpha \wedge \beta$, $\alpha \rightarrow \beta$, $\alpha \leftrightarrow \beta$, and $\forall x.\alpha$ are used for formulas that can be produced from the aforementioned operators. Note that the language $\mathcal{KL}_n$ is not a quantifier-free language, as opposed to the language $\mathcal{L}_n$ that we have presented in Section 2.1.2. An objective formula is a formula without any K-operators, while a subjective formula has all predicates within the scope of a K-operator. In addition, a formula with no free variables is referred to as a sentence.

We define an i-objective formula as a formula where every occurrence of an operator K_i is in the scope of some K_j, $j \neq i$, i.e., a formula with no introspections for agent i. A formula is called i-subjective if any predicates or K_j-operators where $j \neq i$ only appear within the scope of a K_i-operator.

In order to extend the notion of knowing to multi-agent systems, it is necessary to capture the nesting of K-operators of different agents. For this purpose, a depth function is defined. A formula of depth 1 should be understood as capturing beliefs or non-beliefs only concerning the agent's own beliefs or facts about the world (e.g. $K_i\alpha$). A formula of depth 2 should be regarded as capturing beliefs or non-beliefs concerning the agent's own beliefs, facts about the world, as well as beliefs held by other agents up to depth 1. This can be formally described as follows:

1. The depth D_i of a formula α $D_i(\alpha) = 0$ for atomic formulas α;

2. $D_i(\neg\alpha)$ = $D_i(\alpha)$;

3. $D_i(\alpha \vee \beta)$ = max{$D_i(\alpha)$, $D_i(\beta)$};

4. $D_i(\exists x.\alpha)$ = $D_i(\alpha)$;

5. $D_i(K_i\alpha)$ = max{$D_i(\alpha)$, $\max_{j\neq i}(D_j(\alpha) + 1)$};

6. $D_i(K_j\alpha) = 0$ for $j \neq i$.

Note that our definition of depth for $\mathcal{KL}_n$ differs from the one we presented for $\mathcal{L}_n$, as in our metric, introspection will not deepen the formula.

Using this definition of depth, we can adjust our specifications of wffs as follows: if α is a formula of depth d that entails no K-operator or where the outermost K-operator is K_j with $j \neq i$, then $K_i\alpha$ is a formula of depth $d + 1$.

For example, the formula K_1K_2P has an 1-depth of 2, as

$$\begin{aligned}
&D_1(K_1K_2P)\\
=&max\{D_1(K_2P), D_2(K_2P) + 1\}\\
=&max\{0, max\{D_2(P), D_1(P) + 1\} + 1\}\\
=&max\{0, max\{0, 1\} + 1\}\\
=&max\{0, 2\}\\
=&2
\end{aligned}$$

The formula $K_1(P \wedge K_2Q)$ has a 1-depth of 2 as well:

$$\begin{aligned}
&D_1(K_1(P \wedge K_2Q))\\
=&max\{D_1(P \wedge K_2Q), D_2(P \wedge K_2Q) + 1\}\\
=&max\{max\{D_1(P), D_2(K_2Q)\}, max\{D_2(P), D_2(K_2Q)\} + 1\}\\
=&max\{0, max\{0, max\{D_2(Q), D_1(Q) + 1\}\} + 1\}\\
=&max\{0, max\{0, max\{0, 1\}\} + 1\}\\
=&max\{0, 2\}\\
=&2
\end{aligned}$$

Semantics

Let $\mathcal{W}$ be the set of all possible worlds. A world $w \in \mathcal{W}$ is defined as a set of (ground) atoms that are considered true. An epistemic state e_i is defined as a set of possible worlds that have not been ruled out by agent i as being the actual world, where $i \in \mathsf{Ag}$. This implies, that an epistemic state is a subset of $\mathcal{W}$.

In order to distinguish between the worlds an agent believes in and the worlds the agent considers other agents to believe in, it is necessary to be able to represent the available knowledge up to a specific depth k. In order to achieve this, we utilise k-structures (based on the concept of Belle and Lakemeyer (2010)) to define k-epistemic states inductively for $k \geq 0$:

- $k = 0$: e^k is the epistemic state where everything is believed (Belle and Lakemeyer, 2015).
- $k = 1$: $e^k \subseteq \mathcal{W} \times \{\emptyset\}^{n-1}$. We denote the set of all 1-states as $\mathcal{E}^1$.
- $k > 1$: $e^k \subseteq \mathcal{W} \times (\mathcal{E}^{k-1})^{n-1}$.

Thus, e^1 is a set of tuples each containing a world $w \in \mathcal{W}$ that one of the agents could believe in, and the possible states for the other agents, where in this case, for $k = 1$, the possible states are the 0-states and therefore contain no information about any other agent's beliefs. This means, that 1-states only entail facts about a world plus, through introspection, knowledge about these facts. It should be noted that for one agent there is no state, but the world the agent believes in. A (nonempty) 1-state from agent i's point of view, denoted as e_i^1, could thus look like $\{\langle w, \{\}, ..., \{\}\rangle, ...\}$ and represents a set of worlds (Belle and Lakemeyer, 2010).

A 2-structure is defined as:

$e^2 \subseteq \mathcal{W} \times \underbrace{\mathcal{E}^1 \times ... \times \mathcal{E}^1}_{n-1}$

A 2-structure for agent 1, e_1^2, has the form: $\{\langle w, e_2^1, ..., e_n^1\rangle, \langle w', e_2'^1, ..., e_n'^1\rangle, ...\}$, which may be interpreted as meaning that at world $w \in \mathcal{W}$, agent 1 believes the other agents to consider worlds from $e_2^1, ..., e_n 1$, respectively, and at $w' \in \mathcal{W}$, agent 1 believes the other agents to consider worlds from $e_2'^1, ..., e_m'^1$, respectively. This allows us to capture facts about the world and an agent's knowledge about those facts, as well as the agent's belief about other of the agents' beliefs about the actual world. Using this, we are now also able to convey that an agent has only partial information about the other agents. Consequently, the agent's beliefs about what the other agents know differ at different worlds. With this inductive definition of k-structures, we are able to evaluate them recursively.

We say an element of a set of k-states, $s^k \in \mathcal{E}^k$, is a k-structure and denote the set of all k-structures as $\mathcal{S}^k$. When the context is clear, we omit the superscript k.

We can now establish rules that define the truth of sentences in $\mathcal{KL}_n$ based on the specification of Belle and Lakemeyer (2015). For the non-modal part of the specification, we use the specification of Levesque and Lakemeyer (2001). A sentence is considered to be true or false in a multi-agent universe consisting of both a world state and a vector of epistemic states $\vec{e} = (e_1, e_2, ..., e_n)$ for $1, 2, ..., n \in \mathsf{Ag}$. Thus, we define a model as a tuple $(w, e_1, \ldots e_n)$ (also written as $(w, \vec{e})$ for simplicity). For state e' and structure $s = (w, e_1, \ldots e_{i-1}, e_{i+1}, \ldots e_n)$ we use $s \cup_i e'$ to denote the model $(w, e_1, \ldots e_{i-1}, e', e_{i+1}, \ldots e_n)$. We say a model is *compatible* with a formula α if for any $i \in \mathsf{Ag}$, the i-depth of α is not deeper than the epistemic state for agent i. The satisfaction relation is defined as follows:

- $(w, \vec{e}) \models P(t_1, \ldots, t_k)$ iff $w[P(n_1, \ldots, n_k)] = 1$, where $n_i = w(t_i)$;
- $(w, \vec{e}) \models (t_1 = t_2)$ iff $w(t_1)$ is the same name as $w(t_2)$;
- $(w, \vec{e}) \models \neg\alpha$ iff it is not the case that $(w, \vec{e}) \models \alpha$;
- $(w, \vec{e}) \models \alpha \wedge \beta$ iff $(w, \vec{e}) \models \alpha$ and $(w, \vec{e}) \models \beta$;
- $(w, \vec{e}) \models \forall x.\alpha$ iff $(w, \vec{e}) \models \alpha_n^x$ f.a. $n \in \mathcal{N}$;
- $(w, \vec{e}) \models K_i\varphi$ iff f.a. $s \in e_i$, $s \cup_i e_i \models \varphi$.

We are now able to define the notions of entailment, validity and satisfiability within our language: For any sentence $\alpha \in \mathcal{KL}_n$ and set of sentences $\Gamma \subseteq \mathcal{KL}_n$, we define the relation of Γ logically entailing α

as follows:
$\Gamma \models \alpha$ iff for any model $(w, \vec{e})$ compatible with each formula in $\Gamma \cup \{\alpha\}$, iff $(w, \vec{e}) \models \varphi$ f.a. $\varphi \in \Gamma$, then $w, \vec{e} \models \alpha$.
A sentence α is valid (written as $\models \alpha$) iff $\{\} \models \alpha$.
A sentence α is satisfiable, if there exists a model $(w, \vec{e})$ such that $(w, \vec{e}) \models \alpha$.

Furthermore, it should be noted that for an objective sentence α, we write $w \models \alpha$ instead of $w, \vec{e} \models \alpha$. Similarly, for a subjective sentence α, we write $\vec{e} \models \alpha$ instead of $w, \vec{e} \models \alpha$. For an i-subjective sentence α, we write $e_i \models \alpha$.

Properties

The language $\mathcal{KL}_n$ satisfies the K45$_n$ properties, the Barcan formulas are valid as well. In the following, we will prove these properties.

Necessitation: $\models \alpha \Rightarrow \models K_i\alpha$

Proof. In order to demonstrate the validity of the proposition, it is sufficient to prove the following: if $w, \vec{e} \models \alpha$ holds for all $w, \vec{e}$ that are compatible with α, then $w, \vec{e} \models K_i\alpha$ holds for all $w, \vec{e}$ that are compatible with $K_i\alpha$. We assume the antecedent $\models \alpha$. By the semantics of $\mathcal{KL}_n$, $w, \vec{e} \models K_i\alpha$ holds iff for all $s \in e_i$, $s \cup_i e_i \models \varphi$. Since $\models \alpha$ is assumed, $s \cup_i e_i \models \alpha$ holds. Therefore $w, \vec{e} \models K_i\alpha$. □

Distribution: $\models (K_i\alpha \wedge K_i(\alpha \rightarrow \beta)) \rightarrow K_i\beta$

Proof. Again, it is sufficient to demonstrate that this property holds for all models $(w, \vec{e})$ that are compatible with it. If we assume $\neg K_i\alpha$, then the property holds trivially. Let us now assume $K_i\alpha \wedge K_i(\alpha \rightarrow \beta)$. Consequently, it follows for all structures $s \in e_i$ that $s \cup_i e_i \models \alpha$ and $s \cup_i e_i \models \alpha \rightarrow \beta$. It follows that $s \cup_i e_i \models \beta$. Since this holds for all such $s \in e_i$, we can conclude that $e_i \models K_i\beta$ or every e_i that the property is compatible with. It therefore holds that $\models (K_i\alpha \wedge K_i(\alpha \rightarrow \beta)) \rightarrow K_i\beta$. □

Positive Introspection: $\models K_i\alpha \rightarrow K_iK_i\alpha$

Proof. To show that this property holds, we will proceed to prove it for all epistemic states e_i that are compatible with it. Let us assume that $e_i \models \neg K_i\alpha$. In this case, the transitivity property holds vacuously. Conversely, we suppose that $e_i \models K_i\alpha$ holds. Thus, for any $s \in e_i$ it holds that $s \cup_i e_i \models K_i\alpha$ since the selection of s is irrelevant to the satisfaction of the i-subjective formula $K_i\alpha$. Thus, since f.a. $s \in e_i$ $s \cup_i e_i \models K_i\alpha$ holds, we can deduce that $e_i \models K_iK_i\alpha$ for every e_i initially considered. Therefore, $\models K_i\alpha \rightarrow K_iK_i\alpha$ holds. □

Negative Introspection: $\models \neg K_i\alpha \rightarrow K_i\neg K_i\alpha$

Proof. This can be shown in a way similar to the previous proof. It suffices to prove the property for all epistemic states e_i that it is compatible with. Assume $e_i \models K_i\alpha$. Then, trivially, it holds that $e_i \models \neg K_i\alpha \rightarrow K_i\neg K_i\alpha$ since agent i knows α and thus the antecedent is not satisfied. Suppose now $e_i \models \neg K_i\alpha$. Then for any $s \in e_i$ it holds that $s \cup_i e_i \models \neg K_i\alpha$ since, as specified in the previous proof,

the selection of s is irrelevant to the satisfaction of $\neg K_i\alpha$ because it only concerns the epistemic status e_i of agent i. It follows that $e_i \models K_i\neg K_i\alpha$. Consequently, we can deduce $\neg K_i\alpha \rightarrow K_i\neg K_i\alpha$. □

Barcan Formula - Universal: $\models \forall x.K_i\alpha \rightarrow K_i\forall x.\alpha$

Proof. Suppose that $w, e_i \models \forall x.K_i\alpha$. Then, by definition, we have $w, e_i \models K_i\alpha^x_n$ for all $n \in \mathcal{N}$. This means that for all $n \in \mathcal{N}$ and $s \in e_i$ that $s \cup_i e_i \models \alpha^x_n$. By definition $s \cup_i e_i \models \forall x.\alpha$ for all $s \in e_i$. It follows that $e_i \models K_i\forall x.\alpha$.

It is important to note that the converse, $\models K_i\forall x.\alpha \rightarrow \forall x.K_i\alpha$, holds as well:
Assume $w, e_i \models K_i\forall x.\alpha$. It follows from this that f.a. $s \in e_i$ $s \cup_i e_i \models \forall x.\alpha$. Thus we have for all $n \in \mathcal{N}$ and $s \in e_i$ that $s \cup_i e_i \models \alpha^x_n$. By the semantics of the K-operator, it follows for all $n \in \mathcal{N}$ that $e_i \models K_i\alpha^x_n$. This is equivalent to $\neg\exists n \in \mathcal{N}$ s.t. $e_i \models \neg K_i\alpha^x_n$. Using the semantic interpretation of the $\exists$-operator, we deduce $e_i \models \neg\exists x.\neg K_i\alpha$. Using the relation between $\neg\exists x$ and $\forall x$, we can conclude $e_i \models \forall x.K_i\alpha$. □

Barcan Formula - Existential: $\models \exists x.K_i\alpha \rightarrow K_i\exists x.\alpha$

Proof. The proof adheres to the same fundamental principles as the universal one. Suppose $e_i \models \neg\exists x.K_i\alpha$, then the property holds vacuously. Assume now that $e_i \models \exists x.K_i\alpha$. By definition it holds that $e_i \models K_i\alpha^x_n$ for some $n \in \mathcal{N}$. It holds by the semantics of K_i that for all $s \in e_i$ there exists a $n \in \mathcal{N}$ such that $s \cup_i e_i \models \alpha^x_n$. Thus we have, by using the semantic definition of the $\exists$-operator, that for all $s \in e_i$, $s \cup_i e_i \models \exists x.\alpha$. It follows that $e_i \models K_i\exists x.\alpha$. □

4 Translation Function R

This chapter introduces the translation functions $\boldsymbol{R}$, which maps modal logic formulas from $\mathcal{KL}$ (for the single-agent-system version of the function) and $\mathcal{KL}_n$ (for the multi-agent extension of $\boldsymbol{R}$) to first-order logic (FOL). This enables the use of FOL-based automated theorem provers. We commence by presenting the function $\boldsymbol{R}$ for the single-agent case. We proceed to present a concise overview of Feng's proof, which demonstrates that the translation preserves satisfiability in the single-agent case. We proceed to examine a practical implementation of the translation function, outlining a recursive method that translates each component of a formula and can be used with automated theorem provers like VAMPIRE, ensuring practical usability. We subsequently extend the function to multi-agent systems, translating formulas in $\mathcal{KL}_n$, which introduces additional complexity due to multiple agents' knowledge states. This extension allows us to handle nested knowledge and inter-agent beliefs. We will prove that the multi-agent extension of $\boldsymbol{R}$ is satisfiability-preserving, which will be the main focus of this chapter. We will then proceed to examine how the proposed implementation for the single-agent case can be extended to the multi-agent case, thereby demonstrating the practical applicability of our proposed extension of function $\boldsymbol{R}$.

4.1 Single-Agent Cases

This section presents a translation approach for single-agent systems developed and proved by Feng that draws on the set-based possible-world semantics we presented in Section 3.1. Note that this translation function is defined on formulas of $\mathcal{KL}$, which is the single-agent version of the language $\mathcal{KL}_n$ as specified in Section 3.2. The reader is referred to Levesque and Lakemeyer (2001) for a complete definition of the syntax and semantics of $\mathcal{KL}$.

Feng defines a translation function $\boldsymbol{R}[\cdot]$ inductively as follows:

1. $\boldsymbol{R}[\alpha] = (\boldsymbol{R}[\alpha]) \uparrow_{w_0}$ for objective formula α, where w_0 is a constant in FOL, $(\alpha) \uparrow_{w_0}$ means that for any sub-formula $P(t_1, ..., t_m)$ in α, we substitute $P_W(t_1, ..., t_m, w_0)$ (where the subscript W is used to distinguish a symbol in $\mathcal{KL}$ from its FOL counterpart), and for any term $F(t_1, ..., t_m)$ in α, we substitute $F_W(t_1, ..., t_m, w_0)$
2. $\boldsymbol{R}[\alpha_1 \wedge \alpha_2] = \boldsymbol{R}[\alpha_1] \wedge \boldsymbol{R}[\alpha_2]$
3. $\boldsymbol{R}[\neg\alpha] = \neg\boldsymbol{R}[\alpha]$
4. $\boldsymbol{R}[\forall x.\alpha] = \forall x.\boldsymbol{R}[\alpha]$
5. $\boldsymbol{R}[K\alpha] = \forall u.E(u) \rightarrow (\boldsymbol{R}[\alpha]) \uparrow_u$, where u is a new variable, E is a special predicate which is only used for the translation, and $(\Phi) \uparrow_u$ is employed as a substitute for every occurrence of w_0 with u

We show how formulas are translated by this translation function on several examples that showcase different structures of formulas of $\mathcal{KL}$:

Example 1. ***Basic Knowledge***
$\boldsymbol{R}[KP] = \forall u(E(u) \rightarrow P_W(u))$

Example 2. ***Nested Knowledge***
$\boldsymbol{R}[KKP] = \forall u(E(u) \rightarrow \forall v(E(v) \rightarrow P_W(v)))$

Example 3. ***Knowledge with implication***
$\boldsymbol{R}[K(P \rightarrow Q)] = \forall u(E(u) \rightarrow (P_W(u) \rightarrow Q_W(u)))$

Example 4. ***Implication with Knowledge***
$\boldsymbol{R}[P \rightarrow KQ] = P_W(w_0) \rightarrow \forall u(E(u) \rightarrow Q_W(u))$

Example 5. ***Knowledge of an Implication involving Knowledge***
$\boldsymbol{R}[K(P \rightarrow KQ)] = \forall u(E(u) \rightarrow (P_W(u) \rightarrow \forall v(E(v) \rightarrow Q_W(v))))$

Example 6. ***Combination of Knowledge Operators***
$\boldsymbol{R}[K(P \rightarrow KQ) \wedge KS] = \forall u(E(u) \rightarrow (P_W(u) \rightarrow \forall v(E(v) \rightarrow Q_W(v))) \wedge \forall u(E(u) \rightarrow S_W(u)))$

Feng shows that any formula $\alpha \in \mathcal{KL}$ is satisfiability-equivalent to its translation to FOL. We will only briefly sketch his proof here, as our focus is not the translation function for single-agent systems, but its multi-agent extension. The full proof of the satisfiability preservation of this function is provided in Appendix A.1.

In his proof, Feng first states 8 conditions that a FOL-model (M, μ) based on a $\mathcal{KL}$-model (e, w) has to satisfy. These conditions pertain, among other things, to the domain of discourse, the treatment of standard names in translation, and the translation of predicates and functions. Feng claims that a FOL-model (M, μ) satisfying those conditions can be constructed. He proves this by demonstrating that there is a variable map μ' for which it holds that $\mu' \sim_u \mu$ and $\mu'(u) = w'$ for $w' \in e$. This allows the construction of a satisfiability equivalent model (M, μ') from a model (e, w') for formulas capturing the identity between two ground terms or for atoms containing only ground terms. Feng proceeds to demonstrates that this result holds not only for formulas of this form but for any $\alpha \in \mathcal{KL}$. That is, $(e, w') \models \alpha$ iff $(M, \mu') \models (\boldsymbol{R}[\alpha]) \uparrow_u$. He continues by showing that the same holds without the variable mapping: this is captured in Appendix A.1 Theorem 3.

In the second part of the proof, Feng demonstrates that the converse holds as well, i.e., that if there is a FOL-model (M, μ) satisfying $\boldsymbol{R}[\alpha] \cup \Sigma_{\text{eq}} \cup \Sigma_{\text{una}}$, then α is satisfied in $\mathcal{KL}$ (Appendix A.1Theorem 4). He adds Σ_{eq} and Σ_{una} here, as in $\mathcal{KL}$ the unique name assumption is made regarding the domain of quantification, which is why we need the same for the translated FOL formulas. While Σ_{eq} requires the equality symbol '=' to be interpreted as usual, Σ_{una} requires the domain of discourse to be infinite:

- Σ_{eq} is the following infinite set of formulas:
  - $\forall x(x = x)$
  - $\forall x \forall y.(x = y) \supset (y = x)$

- $\forall x \forall y \forall z((x = y) \wedge (y = z)) \supset (x = z)$
- for any function symbol F:
  $\forall x_1 \ldots \forall x_k \forall y_1 \ldots \forall y_k((x_1 = y_1) \wedge \ldots \wedge (x_k = y_k)) \supset F(x_1, \ldots, x_k) = F(y_1, \ldots, y_k)$
- for any predicate symbol P:
  $\forall x_1 \ldots \forall x_k \forall y_1 \ldots \forall y_k((x_1 = y_1) \wedge \ldots \wedge (x_k = y_k)) \supset P(x_1, \ldots, x_k) \equiv P(y_1, \ldots, y_k)$

- $\Sigma_{una} = \{(n \neq n') \mid n, n' \text{ are distinct standard names in } \mathcal{N}\}$

From both conditionals captured in the theorems, Feng concludes the biconditional, showing that for $\alpha \in \mathcal{KL}$, α is satisfiable iff $\boldsymbol{R}[\alpha] \cup \Sigma_{\text{eq}} \cup \Sigma_{\text{una}}$ is FO-satisfiable (Appendix A.1 Theorem 5).

4.2 Single-Agent Implementation

To show that the proposed translation function is implementable and therefore also usable in practice, we will present a methodology for implementing the functionality of the translation in a way that it can be used as an input to theorem provers like VAMPIRE.

We use three methods for implementing the function: one method, `translateStart`, that takes the formula it is given as input (which would be a wff of $\mathcal{KL}$), adds the starting variable u and calls up another method which we call `translate` with the formula as well as the variable u. The method `translate` to which these arguments are passed then calls upon the third method, `partition`, which partitions the formula into its outermost operator and, in case of the operator being a binary connective, both connected parts, and in case of the operator only taking one argument, this argument.

Note that to improve the readability of the proposed pseudocode, we use the mathematical comparison operator ("==") on the relevant characters or character strings instead of the corresponding methods for character or string processing.

The second method, `translate`, then uses a conditional branching to distinguish between the operators. In case of the operator being in $\{\wedge, \vee, \rightarrow, \leftrightarrow, =\}$, the formula of FOL that is being returned is the result of `translate` called on the left argument of the connective, then a string representation of the connective, followed by `translate` called on the left argument of the connective (note that we will partition the parts of the intitial formula as well before continuing to translate them). That is, we use `translate` recursively on the parts of our initial formula.

Similarly, if by the partitioning method the outermost operator is found to be the $\neg$ operator, `translate` returns $\neg$ followed by the result of the call on `translate` of the other part of the formula. For $\forall$ as the outermost operator of the formula, `translate` returns $\forall$, the variable that is bound by the operator, as discovered by the partitioning method, a special predicate O of that variable, to represent the many-sortedness of $\mathcal{KL}$ in FOL, and as a consequence the `translate`-method called on the rest of the formula (i.e., the initial formula without the quantifier and the quantified variable). For $\exists$ as the outermost operator, `translate` returns a similar statement, where a $\exists$-quantifier is substituted for the $\forall$-quantifier and, instead of the conditional, a conjunction is introduced.

In the case of the K-operator being the outermost one, i.e., for formulas of form $K\alpha$ `translate`

returns the following:

$$(\forall var1(E(u_{varcounter}) \wedge W(u_{varcounter})) \rightarrow (translate(\alpha, u_{varcounter}) \wedge E(var1)))$$

where $var1$ is, at the first call of `translate`, the variable representing the actual world, and after the first translation of the K-operator the new variable, as mentioned in the description of the translation function. W is a predicate that we use to imply that we refer to sort world (because of the many-sortedness). $varcounter$ is just a counter that we increase with every call on `translate` with a K-operator as the outermost function, as, using this,that a new variable is being utilised in that part of the formula.

If none of these cases hold, the default case will be selected, which is exactly the case if `translate` is called on in the highest nesting level of the recursivity - and this is precisely when there is no outermost operator left, i.e., when `translate` is called on a formula of form $P(t_1, ..., t_n)$ or $F(t_1, ..., t_n)$. In this case, we `translate` returns $P_W(t_1, ..., t_n, var1)$ or $F(t_1, ..., t_n, var1)$, where $var1$ is the variable the method was called on with. The methods now recursively return the translated partial formulas and the initial method that was called upon with the initial formula returns the FOL-translation of the formula.

For a pseudocode representation of the proposed methodology for implementing the function, the reader is directed to Appendix A.2.

4.3 Multi-Agent Cases

This section presents the multi-agent extension of the translation function $\boldsymbol{R}$ that was introduced in the previous section. Let FOL be a first-order, many-sorted logic of sorts *object* and *state*. We reiterate that the many-sorted setting is only for simplicity in proving, and it is commonplace that many-sorted logic is easily convertible to classical first-order logic.

We define the translation function from $\mathcal{KL}_n$ formulas to FOL formulas $\boldsymbol{R}[\phi, \sigma, u, v]$ inductively as follows, note that ϕ is a modal formula, σ is a finite word from $\mathsf{Ag}*$, and u and v are two variables we use for the translation:

- For primitive formula α, $\boldsymbol{R}[\alpha, \sigma, u, v] := \alpha \uparrow^u_\sigma$, where:
  - For standard name n, $n \uparrow^u_\sigma := n$
  - For variable x, $x \uparrow^u_\sigma := x$
  - For term $F(t_1, \dots t_m)$,
    $F(t_1, \dots t_m) \uparrow^u_\sigma := F_\sigma(t_1 \uparrow^u_\sigma, \dots, t_m \uparrow^u_\sigma, u)$
  - For atom $P(t_1, \dots t_m)$, $P(t_1, \dots t_m) \uparrow^u_\sigma := P_\sigma(t_1 \uparrow^u_\sigma, \dots, t_m \uparrow^u_\sigma, u)$
  - For formula $t_1 = t_2$, $(t_1 = t_2) \uparrow^u_\sigma := (t_1 \uparrow^u_\sigma = t_2 \uparrow^u_\sigma)$
- $\boldsymbol{R}[\alpha_1 \wedge \alpha_2, \sigma, u, v] := \boldsymbol{R}[\alpha_1, \sigma, u, v] \wedge \boldsymbol{R}[\alpha_2, \sigma, u, v]$
- $\boldsymbol{R}[\neg\alpha, \sigma, u, v] := \neg\boldsymbol{R}[\alpha, \sigma, u, v]$
- $\boldsymbol{R}[\forall x.\alpha, \sigma, u, v] := \forall x.\boldsymbol{R}[\alpha, \sigma, u, v]$

- If $end(\sigma) = i$, $\boldsymbol{R}[\boldsymbol{K}_i\alpha, \sigma, u, v] := \forall u'.E_\sigma(u', v) \rightarrow (\boldsymbol{R}[\alpha, \sigma, u', v])$, where,
  - u' is a new variable, which does not occur in $\boldsymbol{R}[\alpha, \sigma, u', v]$
  - E_σ is a special predicate which is only used in the translation,
- If $end(\sigma) \neq i$, $\boldsymbol{R}[\boldsymbol{K}_i\alpha, \sigma, u, v] := \forall u'.E_{\sigma\cdot i}(u', u) \rightarrow (\boldsymbol{R}[\alpha, \sigma \cdot i, u', u])$

The following examples show how this function can be used to translate formulas and cover most of the relevant cases:

Example 7. ***Multi-Agent Nested Knowledge***
$\boldsymbol{R}[K_1K_2P, \epsilon, u, v] =$

$$\begin{aligned}
&\forall wE_1(w, u) \rightarrow (\boldsymbol{R}[K_2P, 1, w, u]) \\
=&\forall wE_1(w, u) \rightarrow (\forall xE_{12}(x, w) \rightarrow (\boldsymbol{R}[P, 12, x, w])) \\
=&\forall wE_1(w, u) \rightarrow (\forall xE_{12}(x, w) \rightarrow P_{12}(x))
\end{aligned}$$

Example 8. ***Multi-Agent Knowledge of an Implication involving Knowledge***
$\boldsymbol{R}[K_1(P \rightarrow K_2Q), \epsilon, u, v] =$

$$\begin{aligned}
&\forall wE_1(w, u) \rightarrow (\boldsymbol{R}[P \rightarrow K_2Q, 1, w, u]) \\
=&\forall wE_1(w, u) \rightarrow (\boldsymbol{R}[P, 1, w, u] \rightarrow \boldsymbol{R}[K_2Q, 1, w, u]) \\
=&\forall wE_1(w, u) \rightarrow (P_1(w) \rightarrow \boldsymbol{R}[K_2Q, 1, w, u]) \\
=&\forall wE_1(w, u) \rightarrow (P_1(w) \rightarrow (\forall xE_{12}(x, w) \rightarrow (\boldsymbol{R}[Q, 12, x, w]))) \\
=&\forall wE_1(w, u) \rightarrow (P_1(w) \rightarrow (\forall xE_{12}(x, w) \rightarrow Q_{12}(x)))
\end{aligned}$$

Example 9. ***Multi-Agent Combination of Knowledge Operators***
$\boldsymbol{R}[K_1(P \rightarrow K_2Q) \wedge K_1S, \epsilon, u, v] =$

$$\begin{aligned}
&\boldsymbol{R}[K_1(P \rightarrow K_2Q), \epsilon, u, v] \wedge \boldsymbol{R}[K_1S, \epsilon, u, v] \\
=&\forall wE_1(w, u) \rightarrow (P_1(w) \rightarrow (\forall xE_{12}(x, w) \rightarrow Q_{12}(x))) \wedge \forall wE_1(w, u) \rightarrow (\boldsymbol{R}[S, 1, w, u]) \\
=&\forall wE_1(w, u) \rightarrow (P_1(w) \rightarrow (\forall xE_{12}(x, w) \rightarrow Q_{12}(x))) \wedge \forall wE_1(w, u) \rightarrow (S_1(w))
\end{aligned}$$

Example 10. ***Multi-Agent Disjunction of Nested Knowledge***
$\boldsymbol{R}[K_1K_2P \vee \neg(K_1K_2P), \epsilon, u, v] =$

$$\begin{aligned}
&\boldsymbol{R}[K_1K_2P, \epsilon, u, v] \vee \boldsymbol{R}[\neg(K_1K_2P), \epsilon, u, v] \\
=&\forall wE_1(w, u) \rightarrow (\forall xE_{12}(x, w) \rightarrow P_{12}(x)) \vee \boldsymbol{R}[\neg(K_1K_2P), \epsilon, u, v] && \text{(Example 7)} \\
=&\forall wE_1(w, u) \rightarrow (\forall xE_{12}(x, w) \rightarrow P_{12}(x)) \vee \neg\boldsymbol{R}[K_1K_2P, \epsilon, u, v] \\
=&\forall wE_1(w, u) \rightarrow (\forall xE_{12}(x, w) \rightarrow P_{12}(x)) \vee \neg(\forall wE_1(w, u) \rightarrow (\boldsymbol{R}[K_2P, 1, w, u])) \\
=&\forall wE_1(w, u) \rightarrow (\forall xE_{12}(x, w) \rightarrow P_{12}(x)) \vee \neg(\forall wE_1(w, u) \rightarrow (\forall xE_{12}(x, w) \rightarrow (\boldsymbol{R}[P, 12, x, w]))) \\
=&\forall wE_1(w, u) \rightarrow (\forall xE_{12}(x, w) \rightarrow P_{12}(x)) \vee \neg(\forall wE_1(w, u) \rightarrow (\forall xE_{12}(x, w) \rightarrow (P_{12}(x))))
\end{aligned}$$

Example 11. ***Multi-Agent Conjunction of Nested Knowledge***
$\boldsymbol{R}[K_1K_2P \wedge \neg(K_2K_1P), \epsilon, u, v] =$

$$
\begin{aligned}
&\boldsymbol{R}[K_1K_2P, \epsilon, u, v] \wedge \boldsymbol{R}[\neg(K_2K_1P), \epsilon, u, v] \\
=&\forall w E_1(w,u) \rightarrow (\forall x E_{12}(x,w) \rightarrow P_{12}(x)) \wedge \boldsymbol{R}[\neg(K_2K_1P), \epsilon, u, v] && \text{(Example 7)} \\
=&\forall w E_1(w,u) \rightarrow (\forall x E_{12}(x,w) \rightarrow P_{12}(x)) \wedge \neg\boldsymbol{R}[K_2K_1P, \epsilon, u, v] \\
=&\forall w E_1(w,u) \rightarrow (\forall x E_{12}(x,w) \rightarrow P_{12}(x)) \wedge \neg(\forall w E_2(w,u) \rightarrow (\boldsymbol{R}[K_1P, 2, w, u])) \\
=&\forall w E_1(w,u) \rightarrow (\forall x E_{12}(x,w) \rightarrow P_{12}(x)) \wedge \neg(\forall w E_2(w,u) \rightarrow (\forall x E_{21}(x,w) \rightarrow (\boldsymbol{R}[P, 21, x, w]))) \\
=&\forall w E_1(w,u) \rightarrow (\forall x E_{12}(x,w) \rightarrow P_{12}(x)) \wedge \neg(\forall w E_2(w,u) \rightarrow (\forall x E_{21}(x,w) \rightarrow (P_{21}(x))))
\end{aligned}
$$

To prove that this function is satisfiability-preserving, we show that a canonical FOL-model can be constructed from a $\mathcal{KL}_n$-model. Such a canonical FOL model M^c should be constructed satisfy multiple conditions stating the following:

C1 Domain of discourse $\mathcal{D} = \mathcal{D}_o \cup \mathcal{D}_w$, where the domain of sort object $\mathcal{D}_o$ is isomorphic to the set of standard name $\mathcal{N}$, and the domain of sort state $\mathcal{D}_w$ is isomorphic to the set of $\mathcal{KL}_n$ models $\mathbb{M}$. We use $f_o \colon \mathcal{N} \rightarrow \mathcal{D}_o$ and $f_w \colon \mathbb{M} \rightarrow \mathcal{D}_w$ to denote the functions of isomorphism and use f_o^{-1} and f_w^{-1} for the inverse functions. In particular, $n^{M^c} = f_o(n)$ for any $n \in \mathcal{N}$.

From this it follows in particular that we define our model on a many-sorted FOL that contains elements of sort object (with the domain of sort object being denoted as $\mathcal{D}_o$) as well as elements of sort state (where the domain of sort state is $\mathcal{D}_w$). Secondly, the model should be constructed such that the equality symbol '=' is interpreted in the usual sense:

C2 Equality is interpreted in the usual sense, i.e. $d =^{M^c} d$ and $d \neq^{M^c} d'$ for any distinct elements d, d'.

We also use the following conditions to stipulate the connection between predicates and functions of $\mathcal{KL}_n$ and FOL-predicates and functions:

- $P_\sigma \subseteq \underbrace{\mathcal{D}_o \times \cdots, \times \mathcal{D}_o}_{j} \times \mathcal{D}_w$ and $F_\sigma : \underbrace{\mathcal{D}_o \times \cdots, \times \mathcal{D}_o}_{j} \times \mathcal{D}_w \rightarrow \mathcal{D}_o$

C3 f.a. $\vec{d} \in (\mathcal{D}_o)^j$ and $\omega \in \mathcal{D}_w$, $(\vec{d}, \omega) \in P_\sigma^{M^c}$ iff $\vec{n} \in w[P]$ where $(w, \vec{e'}) = f_w^{-1}(\omega)$ with some $\vec{e'}$ and $n_i = f_o^{-1}(d_i)$ for $i \in \{1, \ldots, j\}$

C4 f.a. $\vec{d} \in (\mathcal{D}_o)^j$ and $\omega \in \mathcal{D}_w$, $F_\sigma^{M^c}(\vec{d}, \omega) = d'$ iff $w[F](\vec{n}) = n'$ where $(w, \vec{e'}) = f_w^{-1}(\omega)$ with some $\vec{e'}$, $n_i = f_o^{-1}(d_i)$ and $n' = f_o^{-1}(d')$

With the first of these conditions we specify that all tuples of elements of sort object in FOL together with an element of sort state in FOL are an element of a predicate $P_\sigma^{M^c}$, where $P_\sigma \subseteq \mathcal{D}_o \times \ldots \times \mathcal{D}_o \times \mathcal{D}_w$, iff there is a predicate P in $\mathcal{KL}_n$ such that the standard names corresponding to the elements of sort object by f_o are in the value of the predicate P in the epistemic state that corresponds to the element of state state by f_w. We state a similar condition for the evaluation of functions.

Finally, we use the conditions to set the interpretation of binary predicates $E_\sigma(\omega, \omega')$ as corresponding to having moved in an agents epistemic state ω' to epistemic state ω:

C5 Let $end(\sigma) = i$. For $\omega, \omega' \in \mathcal{D}_w$ s.t. $f_w^{-1}(\omega) = (w, \vec{e})$ and $f_w^{-1}(\omega') = (w', \vec{e'})$, $(\omega, \omega') \in E_\sigma^{M^c}$ iff $e_i = e'_i$ and $(w, \vec{e}_{-i}) \in e'_i$

Our goal is to show that for a FOL-model M^c constructed to satisfy these condition, and for $\mathcal{KL}_n$ models $(w, \vec{e})$ and $(w', \vec{e'})$ such that $e_i = e'_i$ and $(w, \vec{e_{-i}}) \in e'_i$, i.e. such that we can move to $(w, \vec{e})$ in agent i's epistemic state e'_i in $(w', \vec{e'})$, the following holds for any sentence α in $\mathcal{KL}_n$: Firstly, the satisfiability is preserved for formulas where we do not have to consider the possible structures from an agent's epistemic state. For this we show that, given a mapping μ that maps the variable u to the element of sort state that corresponds to $(w, \vec{e})$ by f_w, the $\mathcal{KL}_n$ model $(w, \vec{e})$ satisfies α exactly if our canonical FOL-model M^c together with the mapping μ satisfies the translation $\boldsymbol{R}[\alpha, \varepsilon, u, v]$. Note that we have not mapped v to anything here. Secondly, we want to show that satisfiability is preserved by $\boldsymbol{R}[\cdot]$, even when we do consider subjective sentences, i.e., when we move from an agent i's epistemic state to the structures they consider possible. Given the same mapping of u by μ to $(w, \vec{e})$, but also $\mu(v) = f_w(w', \vec{e'})$ s.t. $(w, \vec{e} = s \cup_i e'_i)$ for some $s \in e'_i$, our goal is to show that for any string $\sigma \in \mathsf{Ag}^*$ the $\mathcal{KL}_n$ model $(w, \vec{e})$ satisfies α if and only if our canonical FOL-model M^c together with the function μ satisfies the translation $\boldsymbol{R}[\alpha, \sigma \cdot i, u, v]$, where i corresponds to the agent in whose epistemic state we are moving. This is stated in the following theorem:

Theorem 1. *[Goal] Let M^c be constructed as above, consider models $(w, \vec{e})$ and $(w', \vec{e'})$ such that $e_i = e'_i$ and $(w, \vec{e}_{-i}) \in e'_i$, then for any sentence $\alpha \in \mathcal{KL}_n$,*

- *Given μ s.t. $\mu(u) = f_w(w, \vec{e})$, then $w, \vec{e} \models \alpha$ iff $M^c, \mu \models_{FO} \boldsymbol{R}[\alpha, \epsilon, u, v]$*
- *Given μ s.t. $\mu(u) = f_w(w, \vec{e})$, $\mu(v) = f_w(w', \vec{e'})$, then for any $\sigma \in \mathsf{Ag}^*$,*

$$w, \vec{e} \models \alpha \text{ iff } M^c, \mu \models_{FO} \boldsymbol{R}[\alpha, \sigma \cdot i, u, v]$$

To prove this, we first show, that this holds for the equality of any ground terms, as well as for atomic formula of form $P(t_1, \ldots, t_n)$:

Lemma 1. *Given $\mathcal{KL}_n$ model $(w, \vec{e})$, let M^c be constructed as above. Considering σ, variables u, v and variable map μ s.t. $\mu(u) = f_w(w, \vec{e})$,*

1. *For any ground terms t_1, t_2,*

$$w, \vec{e} \models t_1 = t_2 \text{ iff } M^c, \mu \models_{FO} \boldsymbol{R}[t_1 = t_2, \sigma, u, v] \tag{4.1}$$

2. *For any atomic formula $P(t_1, \ldots, t_j)$,*

$$w, \vec{e} \models P(t_1, \ldots, t_j) \text{ iff } M^c, \mu \models_{FO} \boldsymbol{R}[P(t_1, \ldots, t_j), \sigma, u, v] \tag{4.2}$$

We use induction to prove this lemma, where in the base case we consider terms of form $t = F(n_1, \ldots, n_k)$, and in the inductive step we consider terms that are functions of terms, i.e., $t = F(t_1, \ldots, t_k)$. After we show that Lemma 1 holds, we know that Theorem 1 holds for base cases with primitive atoms. In the subsequent step, we show that it also holds for more complex formulas, which we prove by induction on the operators $\neg$, $\wedge$, and $\forall$. In these cases we do not move from any agent's epistemic state to a structure they consider possible. For the K-operator, on the other hand, we consider both variables that are part of the translation function, to show that $w, \vec{e} \models K_i\alpha \Leftrightarrow M^c, \mu \models \boldsymbol{R}[K_i\alpha, \sigma, u, v]$ for both cases ($end(\sigma) \neq i$ and $end(\sigma) = i$), where we show the equivalency by using the definition of the interpretation of the predicate E, as well as the function μ. Note that the first part of the theorem follows easily from this, as we move into the agent $i's$ epistemic state not from a different agent's epistemic state but from the state we are evaluating a formula in when we haven't considered any agent's epistemic state yet. With this, the proof concludes, since we have shown that for any $\mathcal{KL}_n$-sentence α the canonical model M^c can be constructed such that $w, \vec{e} \models \alpha$ if and only if $M^c, \mu \models \boldsymbol{R}[\alpha, \delta, u, v]$, where δ is the empty word ϵ if we just consider the current epistemic state, i.e., only $\mu(u)$ in cases where we have an objective formula, and $\delta = \sigma \cdot i$ for any $\sigma \in \mathsf{Ag}^*$, if we consider formulas where we have to move to a structure an agent still considers possible in their epistemic state, which is the case for subjective formulas, i.e., formulas containing the K_i-operator.

To prove the correctness of our proposed translation function, we prove that the converse holds as well, namely that given a translation $\boldsymbol{R}[\alpha, \epsilon, u, v] \cup \Sigma_{eq} \cup \Sigma_{una}$, where $\alpha \in \mathcal{KL}_n$ that is satisfiable in FOL, it holds that α is satisfiable in $\mathcal{KL}_n$:

Theorem 2. *Given $\alpha \in \mathcal{KL}_n$, if $\boldsymbol{R}[\alpha, \varepsilon, u, v] \cup \Sigma_{eq} \cup \Sigma_{una}$ is FO-satisfiable, then α is $\mathcal{KL}_n$-satisfiable.*

Here we use Σ_{eq} and Σ_{una} to be able to capture the unique name assumption of $\mathcal{KL}_n$ in the FOL formulas, as well as that equality should be interpreted in the usual sense. Note that Σ_{eq} and Σ_{una} are defined in the same way as Feng did in the single-agent case.

We define the sets $\mathcal{M}^{e,u}$ and $\mathcal{M}^{e,u,o}$ as a prerequisite for this part of the proof. While $\mathcal{M}^{e,u}$ denotes the set of all models which satisfy $\Sigma_{eq} \cup \Sigma_{una}$, $\mathcal{M}^{e,u,o}$ denotes the set of all models in $\mathcal{M}^{e,u}$ that share the same domain of object with the canonical model M^c and for which every element of sort object is named.

Our proof consists of three steps: Firstly, we show that, given a formula $\alpha \in \mathcal{KL}_n$, if $\boldsymbol{R}[\alpha, \epsilon, u, v] \cup \Sigma_{eq} \cup \Sigma_{una}$ is satisfiable in FOL, then $\boldsymbol{R}[\alpha, \epsilon, u, v]$ is satisfiable in $\mathcal{M}^{e,u}$. Because of the definition of $\mathcal{M}^{e,u}$, this enables us to drop the $\Sigma_{eq} \cup \Sigma_{una}$ part. The proof of this step is trivial.

Secondly, we show that given a formula $\alpha \in \mathcal{KL}_n$, if $\boldsymbol{R}[\alpha, \epsilon, u, v]$ is satisfiable in $\mathcal{M}^{e,u}$, then $\boldsymbol{R}[\alpha, \epsilon, u, v]$ is satisfiable in $\mathcal{M}^{e,u,o}$:

Lemma 2. *For $\alpha \in \mathcal{KL}_n$, given $M \in \mathcal{M}^{e,u}$ and an assignment μ s.t. $M, \mu \models \boldsymbol{R}[\alpha, \epsilon, u, v]$, there exists a model $M' \in \mathcal{M}^o$ and an assignment μ' s.t. $M', \mu' \models \boldsymbol{R}[\alpha, \epsilon, u, v]$*

Intuitively speaking, this means that we construct a model M' to respect the interpretation of objects in $\mathcal{M}^{e,u,o}$. We do this to show that we can move from $\mathcal{M}^{e,u}$ to its subset $\mathcal{M}^{e,u,o}$, so that, in the next step, we are able to move to the actual model M^c. Given a model $M \in \mathcal{M}^{e,u}$ and an assignment μ such that $M,\ u \models \boldsymbol{R}[\alpha, \epsilon, u, v]$, we prove the second step by using the Löwenheim-Skolem theorem

to demonstrate that the domain of sort and the domain of sort state both are countable and thus, to construct a bijection between the domain of sort object for any model M and the set of standard names. Using this, we can construct a model $M' \in \mathcal{M}^{e,u,o}$ and an assignment μ' based on M and μ such that $M', \mu' \models \boldsymbol{R}[\alpha, \epsilon, u, v]$.

In the third and final step of our proof, we show that if $\boldsymbol{R}[\alpha, \epsilon, u, v]$ is satisfiable in the set $\mathcal{M}^{e,u,o}$, it is also satisfiable in the canonical model M^c together with some variable mapping μ. We show this by construction a set of mappings between the domain of sort state of some model $M \in \mathcal{M}^{e,u,o}$ and the domain of sort state of the canonical model M^c, such that the mapping preserves the truth value for formulas up to a certain depth. We inductively construct mapping $\phi^{(\sigma)}$ and set S^{ω}_{σ} given $M \in \mathcal{M}^{e,u,o}$ and $k \in \mathbb{N}$, for any $\sigma \in \mathsf{Ag}^*$ such that $|\sigma| \leq k$ as follows:

1. When $|\sigma| = k$, f.a. $\omega \in \mathcal{D}_w$, we select a state $\omega^c \in \mathcal{D}^c_w$ such that ,

   1.i) $(\vec{d}, \omega) \in P^M_\sigma$ iff $(\vec{d}, \omega^c) \in P^{M^c}_\sigma$ for any P and $\vec{d}$;

   1.ii) $F^M_\sigma(\vec{d}, \omega) = F^{M^c}_\sigma(\vec{d}, \omega^c)$ for any F and $\vec{d}$.

   We use $\phi^{(\sigma)}$ to denote the mapping from ω to the selected state, i.e. $\phi^{(\sigma)}(\omega) = \omega^c$. Given ω, we define the set $S^{\omega}_{\sigma} = \{\omega' \,|\, (\omega', \omega) \in E^M_\sigma\}$. For simplicity, we write $\phi^{(\sigma)}(S^{\omega}_{\sigma})$ to mean $\{\phi^{(\sigma)}(\omega') \,|\, \omega' \in S^{\omega}_{\sigma}\}$.

2. When $|\sigma| < k$, f.a. $\omega \in \mathcal{D}_w$, we select a state $\omega^c \in \mathcal{D}^c_w$ such that

   2.i) $(\vec{d}, \omega) \in P^M_\sigma$ iff $(\vec{d}, \omega^c) \in P^{M^c}_\sigma$ for any P and $\vec{d}$;

   2.ii) $F^M_\sigma(\vec{d}, \omega) = F^{M^c}_\sigma(\vec{d}, \omega^c)$ for any F and $\vec{d}$;

   2.iii) F.a. $i \neq end(\sigma)$, $(\omega', \omega^c) \in E^{M^c}_{\sigma \cdot i}$ iff $\omega' \in \phi^{(\sigma \cdot i)}(S^{\omega}_{\sigma \cdot i})$

   $\phi^{(\sigma)}$ and S^{ω}_{σ} are defined identically as the first case.

   Note that for the mapping ϕ^{σ} with $\sigma = \sigma' \cdot j \cdot i$, i.e., j is the second last letter in the word, we always select ω^c such that $(w, \vec{e}) = f^{-1}_w(\omega^c)$, where the depth of e_j is higher by one than any $j' \neq j$, and $(w, \vec{e}_{-j}) \in e_j$.

Our definition of the mapping means that we map states of M to states of M^c if they, together with some objects, are elements of the predicates and functions in, respectively, M and M^c corresponding to the same predicate in $\mathcal{KL}_n$. Furthermore, this means that the set S^{ω}_{σ} contains all the states that we can reach from a state ω by entering an epistemic state. For $|\sigma| < k$, $\phi^{(\sigma)}$ and S^{ω}_{σ} are defined based on the definition of $\phi^{(\sigma \cdot j)}$ and $S^{\omega}_{\sigma \cdot j}$ with $j \neq end(\sigma)$.

We continue by showing that for any model M and finite k, such a set of mappings always exists, which we state in the following lemma:

Lemma 3. *Given $M \in \mathcal{M}^{e,u,o}$, let ϕ^{σ} and S^{ω}_{σ} be defined as above, for any sentence α and σ s.t. $dep(\boldsymbol{R}[\alpha, \sigma, u, v,]) \leq k$ (the longest subscript is at most k),*

$$M, \mu \models \boldsymbol{R}[\alpha, \sigma, u, v] \textit{ iff } M^c, \tilde{\mu} \models \boldsymbol{R}[\alpha, \sigma, u, v] \tag{4.3}$$

Where $\tilde{\mu}(u) = \phi^{(\sigma)}(\mu(u))$. When $|\sigma| > 0$, $\tilde{\mu}(v) = \phi^{(\sigma^-)}(\mu(v))$. [1]

[1] σ^- means σ without the last letter, e.g. $\sigma = 1 \cdot 3 \cdot 2$, then $\sigma^- = 1 \cdot 3$

For our proof of this lemma we use induction on the complexity of a formula α. We notice that the base case for this induction follows from the definition, the cases for negation and conjunction are easily provable by the semantics of FOL. For the case of universal quantification of sort object we use the fact that, by the defintion of $\mathcal{M}^{e,u,o}$, the sub-domain of sort object is countable and each element is named, therefore the quantification can be interpreted substitutionally. Using the definition of S^{ω}_{σ}, we can also prove that the lemma holds for formulas of form $K_i\alpha$.

With this, together with Theorem 1, we have shown the third and last step of this direction of the proof, which proves that given $\alpha \in \mathcal{KL}_n$, if $\boldsymbol{R}[\alpha, \varepsilon, u, v] \cup \Sigma_{eq} \cup \Sigma_{una}$ is FO-satisfiable, then α is $\mathcal{KL}_n$-satisfiable.

This concludes the proof of the satisfiability preservation of the proposed translation function $\boldsymbol{R}[\cdot]$ for $\mathcal{KL}_n$ formulas. The complete proof is provided in Appendix A.3.

4.4 Multi-Agent Implementation

The translation function for the multi-agent case can be implemented in a similar way as the translation function for the single-agent case: we use three methods, namely one method to partition the formula (or the currently considered part of the formula), `partitionMAS`, one that recursively translates the formula, `translateMAS`, and one method for starting the translation, which we call `translateStartMAS`.

In the starting method, we call on the function `translateMAS` with the formula, an empty string representing σ in the beginning of the translation, and two variables, u and v.

The method `translateMAS` operates in a manner analogous to its single-agent version, with the exception that in this instance, σ and the two variables representing the two most recent worlds are retained and passed to the recursive calls of the function. A second distinction can be identified in the manner in which the K-operator is handled. In the multi-agent version of the translation function, two cases must be distinguished depending on whether an operator K_i has a subscript that is equal to the end of σ, or not. This distinction must be reflected in the implementation. We also have to update the last two worlds according to the definition of the translation function as proposed in the previous section.

The pseudocode of the proposed methodology for implementing the function is presented in Appendix A.4.

5 Analysis

Church and Turing have proven that the satisfiability of sentences of FOL is not decidable (Turing, 1936, Church, 1936). But while FOL is undecidable, there are decidable syntactic fragments of FOL. One of these fragments is FO^2, the fragment of 2-variable first-order logic. Vardi (1996) has shown that propositional modal logic can be viewed as a fragment of FO^2 and is thus decidable, while Andréka et al. (1998) proved that propositional modal logics respond to a guarded fragment of FOL. For models in propositional modal logics, the standard translation from modal logic to FOL does not need more than two free variables (Vardi, 1996). Further, it is known that the modal logic K is a fragment of GF^2 (Grädel, 1999). But, for modal logics such as K45 or S5 (and their multi-agent versions), the translation is not in the decidable fragment because properties of frames like transitivity or euclideanity cannot be expressed in GF or FO^2.

We claim that, ideally, a translation approach translates formulas of modal logics into a decidable fragment of FOL in a way that (1) the resulting formula is in a decidable fragment of FOL, for example in GF^2, and (2) the complexity of evaluating the satisfiability of the resulting formula is not too high. Some of the translation approaches we presented in Chapter 3 fulfil the first requirement, some the second, but none of these approaches fulfil both.

We claim that our approach fulfils both requirements, at least for the propositional part, and that this is rooted in the way we defined $\mathcal{KL}_n$, its semantics, as well as the translation function capturing the semantics by the definition of the E-predicates and the use of σ. We start by analysing the other presented semantical translation approaches and identify possible drawbacks. We continue by showing that property (1) holds for our translation approach, for the propositional fragment of $\mathcal{KL}_n$. After that, we will argue that our approach also satisfies property (2).

5.1 Analysing Semantical Translation Approaches

Each of the discussed semantic translation-based approaches introduces unique techniques for handling modal logic by translating it into a form of first-order logic (FOL). However, each approach presents certain limitations or drawbacks that can affect its scalability, efficiency, or applicability in specific contexts such as multi-agent systems.

The translation proposed by Auffray and Enjalbert (1992) involves lifting the modal logic into a many-sorted logic, where a sub-domain of action is introduced to manage modal operators and their nesting. A significant drawback of this method is that the authors do not provide a concrete proof that the translation remains in a decidable fragment of logic. In many-sorted logics, the boundary between decidable and undecidable fragments is even less explored and understood, making it uncertain whether the translated formulas remain computationally manageable in all cases. Without guarantees on

decidability, this approach may result in formulas that, while expressively powerful, cannot be handled by existing automated theorem provers, especially in complex multi-agent systems.

Note that while the translation of $\mathcal{KL}_n$ using our translation approach also results in a many-sorted logic, only one sort is involved for translating propositional cases.

Nonnengart (1993) introduced a translation that combines both functional and relational approaches. One of the key issues with this method is its reliance on second-order quantification. Although this hybrid approach attempts to balance between two powerful methodologies, the introduction of second-order quantifiers poses a significant challenge because second-order logic is known to be undecidable. To use this translation effectively, it would be necessary to eliminate these second-order quantifiers, a process that often introduces significant complexity and computational overhead. The need for second-order quantification limits the approach's utility in scenarios that require efficient reasoning and satisfiability checking, as these operations may no longer be tractable with the increased logical expressiveness.

The functional translation approach proposed by Ohlbach and Schmidt (1997) introduces an accessibility function to manage modal operators, focusing on simplifying the translation by assigning function symbols to represent the accessibility relations between possible worlds. While this method effectively handles certain modal systems, it comes with its own challenges. One potential drawback of Ohlbach and Schmidt's 1997 functional translation approach is that it introduces nested functional terms as arguments to predicates, which can significantly complicate the formulas. Although the method allows for some optimization by using flat terms and functional notation, the handling of complex quantifier structures, particularly when dealing with second-order frame properties, may lead to inefficiencies in practical theorem proving. Furthermore, while the translation can map certain second-order frame properties to first-order formulas, the need for quantifier manipulation (such as moving existential quantifiers) can increase the computational complexity, especially in systems with multi-modal logics or quantified modal logics, where exchanging quantifiers becomes more intricate

In their approach to translating regular grammar logics with converse, Demri and De Nivelle (2005) face challenges in scalability when applied to multi-agent systems. Their translation depends on automata theory, and the size of the resulting formulas is closely tied to the complexity of the automata involved. In multi-agent systems, the number of agents increases the complexity of interactions, which in turn causes the size of the corresponding automata to scale up dramatically. This increased formula size can lead to computational inefficiency, making it difficult to apply their approach in practical multi-agent scenarios. Moreover, it remains unclear how this method would handle multi-agent systems in a way that maintains efficiency while addressing the additional complexity introduced by the interaction of multiple agents.

5.2 Decidability of the Translation (Property (1))

Translating Transitivity and Euclideanity:
We know that for modal logics that satisfy the K45 or S5 properties, such as $\mathcal{KL}_n$, the translation using the standard translation approach is out of the decidable fragment, because, using this translation, frame properties like transitivity or Euclideanity cannot be expressed in FO^2 or GF.

That they cannot be expressed in FO^2 can be shown, for example, by translating the formula that would then hold as a theorem on transitive frames: $\Box P \rightarrow \Box\Box P$. We know that $ST_x(\Box P \rightarrow \Box\Box P)$

$= \forall y(Rxy \rightarrow P(y)) \rightarrow \forall z(Rxz \rightarrow (Rzu \rightarrow P(u)))$. To obtain the global first-order correspondent, we would need to add $\forall x$, but this would mean that we need at least three different variables, so the formula is not in the FO^2-fragment. Further, to make the formula valid in a logic, i.e., on frames instead of on a specific model with a valuation, it would need to be valid for all predicates P, which we could achieve by quantifying over the predicates, but then the formula would be a formula of second-order logic. To achieve the same result without needing to use second-order logic, we can use the Sahlqvist-van Benthem algorithm (Blackburn et al., 2001) to make use of the Sahlqvist correspondence (ibid.) and thus obtain a satisfiability-equivalent first-order formula $\forall x \forall z \forall u((Rxz \wedge Rzu) \rightarrow Rxu)$. This formula makes it even more evident, that the modal formula capturing transitivity cannot be translated into a satisfiability-equivalent formula in FO^2 using the standard translation approach or approaches that similarly aim to capture the meaning of these properties as modelled in Kripke semantics.

We have proven that transitivity is a property of $\mathcal{KL}_n$ in Chapter 3. Thus, the corresponding formula, $K_iK_i\alpha \rightarrow K_i\alpha$ for $i \in \mathsf{Ag}$ holds for all α and all $i \in \mathsf{Ag}$. Because of the set-based semantics of our logic, we do not have the problem of having to quantify over predicates, since we do not make a distinction similar to that between frames and models in Kripke semantics.

In Kripke-semantics a formula is valid on a frame exactly if it holds for all valuations and at all states. This means, we have to consider that predicates could be true or false at a point depending on the valuation function, which is why we have to quantify over predicates to express the frame property transitivity, to make sure that it holds for every predicate. In the set-based semantics introduced in Chapter 3 on the other hand, a formula is valid exactly if, when it is objective, it gets evaluated to "1" in all worlds that are considered possible, or, when it is subjective, if the objective part of the formula is true in the worlds of all states that are still considered possible by the agent whose epistemic state we are considering, as implied by the index of the K-operators. So instead of having to quantify over the predicates, we know that the translation of formulas $K_iK_iP \rightarrow K_iP$ holds, since, when K_iK_iP for some $i \in \mathsf{Ag}$ and predicate P, and then we know that it holds in i's epistemic state, i.e., i is only considering all the worlds (and possible epistemic states of other agents) where P is evaluated to "1", but this is exactly when the corresponding first-order predicate P_W holds (by definition). Then, by the definition of our semantics K_iP holds as well and thus also the translation of it. When K_iK_iP doesn't hold, the conditional holds vacuously, this is also reflected in the translation. So, because our logic is based on sets and every objective formula can be evaluated in a world, we have an implicit valuation.

Translating into FO^2: Propositional Part

By the definition of our translation function, the translation of the propositional part of our logic can be translated into FO^2: For objective formulas, this is trivial. For subjective formulas with no nested operators, this is obvious as well. Thus, we have to show that this holds for nested formulas. If we have a nesting of form $K_iK_i\alpha$, where $i \in \mathsf{Ag}$ and α is objective, then we can translate this as follows:

$$\begin{aligned}
&\boldsymbol{R}[K_iK_iP, \epsilon, u, v] \\
=&\forall w E_i(w, u) \rightarrow (\forall s E_i(s, u) \rightarrow P_i(s)) \\
=&\forall w E_i(w, u) \rightarrow (\forall w E_i(w, u) \rightarrow P_i(w))
\end{aligned}$$

It is obvious that the resulting formula is in FO^2, but it is also equivalent to $\forall w E_i(w,u) \rightarrow P_i(w)$. Thus, $\boldsymbol{R}[K_iK_iP, \epsilon, u, v] \equiv \boldsymbol{R}[K_iP, \epsilon, u, v]$. It can easily be shown that this is also the case for higher nestings of K_i.

If, on the other hand, we have a nesting of form $K_iK_j\alpha$, where $i, j \in \mathsf{Ag}$, $i \neq j$ and α is objective, the translation would be as follows:

$$\begin{aligned}
&\boldsymbol{R}[K_iK_jP, \epsilon, u, v] \\
=&\forall w E_i(w,u) \rightarrow (\forall v E_{ij}(v,w) \rightarrow P_{ij}(v)) \\
=&\forall w E_i(w,u) \rightarrow (\forall u E_{ij}(u,w) \rightarrow P_{ij}(u))
\end{aligned}$$

Here, by reusing the variable u, which is possible because u is not a free variable in the resulting formula, the formula only entails two different variables and is thus in FO^2.

By reusing the variables it is possible to translate even formulas with higher levels of nesting into FO^2-formulas:

Example 12.

$$\begin{aligned}
&\boldsymbol{R}[K_1K_2K_3P, \epsilon, u, v] \\
=&\forall v E_1(v,u) \rightarrow (\boldsymbol{R}[K_2K_3P, 1, v, u]) \\
=&\forall v E_1(v,u) \rightarrow (\forall u E_{12}(u,v) \rightarrow (\boldsymbol{R}[K_3P, 1, u, v])) \\
=&\forall v E_1(v,u) \rightarrow (\forall u E_{12}(u,v) \rightarrow (\forall v E_{123}(v,u) \rightarrow P_{123}(v)))
\end{aligned}$$

This works because we use the E-predicates to keep track of the epistemic states we are entering (by the subscript), and also because the variables u and v are not free in the resulting formula.

Thus, we have shown that, at least for the propositional part, the formulas that are the result of a translation using our translation function $\boldsymbol{R}[\cdot]$ are in FO^2.

Translating into GF: Propositional Part

It is also fairly easy to prove that the translation of any formula α of the propositional part of $\mathcal{KL}_n$ is in GF. For this, we can use induction on the structure of α: In the base case, $\boldsymbol{R}[P, \sigma, u, v]$, which is translated to $P(u)$, which is guarded. By adding negation or conjunction, the guardedness is still preserved. Quantifiers can only occur when a α is of form $K_i\beta$, but then the translated formula has a guard E_σ containing all free variables.

We have thus shown that using our translation function, we are able to translate every formula of the propositional part of $\mathcal{KL}_n$ into a formula of GF^2, which is a decidable fragment of FOL. We argue that this is mainly because of the way we have defined our semantics, namely, on sets, and subsequently, because of the use of the E-predicate in the translation. Because of this, we avoid the problems described that other approaches, like the standard translation, encounter. We conclude that axiomatizing Kripke frames might not be the best way to express modal properties in first-order language.

Translating the Non-Propositional Part:
For the non-propositional part, we first take a closer look at objective formulas: it is clear that if we have only the monadic fragment of $\mathcal{KL}_n$, the translated formula will still be in GF^2, for formulas with up to two variables. In contrast, for formulas with three or more variables and binary predicates, the translations will unquestionably not be in FO^2 and thus not be in GF^2 and will also not be decidable.

For subjective formulas, if the objective part of the formula is in GF, and we have a formula of the form $K_i\varphi$ where φ is objective has only monadic predicates and at most two variables, then the translation can only be in the decidable fragment of FOL if φ is bounded, i.e., if there are no free variables in φ. This is because we can translate such a formula into the loosely guarded fragment (LGF) of FOL. This fragment is an extension of the guarded fragment (GF), allowing for more flexibility in how quantifiers are applied while still maintaining some of the beneficial properties of the original guarded fragment, particularly decidability. In LGF, we can use loosely guarded quantification, which means that a formula can be guarded by multiple guards. For example, the formula $\forall XXK_1q(x)$, where $q(x)$ is objective and bounded, as we can easily see, can be translated to the formula $\forall x(O(x) \rightarrow ((\forall vE_i(v,u) \wedge S(v)) \rightarrow Q_E(x,v)))$, where O is a predicate that guards the variables that correspond to some element of the domain of sort object, and S is a predicate that guards the variables that correspond to some element of the domain of sort state.

We have thus shown that, apart from the propositional part of our logic, our proposed translation function also translates some formulas of the non-propositional part of $\mathcal{KL}_n$ into a decidable fragment of FOL. While the other formulas are translated into some fragment of FOL as well and thus the satisfiability is preserved, as we have shown in the previous chapter, it is not sure whether they are decidable.

5.3 Complexity of the Translation (Property (2))

While we have argued in the previous section that our translation approach translates formulas of the propositional part of $\mathcal{KL}_n$ into a decidable fragment of FOL, this is not something that is unique to our approach: For approaches like the one presented by Demri and De Nivelle (2005), the translated formulas are in GF^2, though this has the price of scaling up the size of the formula immensely, as we have seen in Chapter 3. Using our translation approach, the resulting formula, while longer than the original formula, does not have this problem to that extent: our approach translates formulas of $\mathcal{KL}_n$ into FOL-formulas of linear size compared to the original $\mathcal{KL}_n$-formula.

This is obvious for objective formulas, and for subjective formulas, this is, again, due to our use of the E-predicates in the translation and the σ being used as a subscript to the translated predicates and functions. For objective formulas, we only add a variable corresponding to the state to at the end of the predicates or functions, as denoted in Chapter 4. The translation of quantifiers and the operators $\neg$, $\wedge$, $\vee$, $\rightarrow$ do not add any additional characters to the formula. For subjective formulas, there is an increase in size, which we cannot avoid for nested operators, as adding antecedents is part of our translation function. But our translation approach still seems more efficient than other approaches, as in our approach the SAT-problem of the resulting formula is decomposable. For example, the standard translation, for

formulas of form $K_1K_2P \wedge \neg K_2K_1P$, would be translated as follows:

$$\begin{aligned}&\boldsymbol{R}[K_1K_2P \wedge \neg K_2K_1P, \epsilon, u, v]\\ =&(\forall v E_1(v,u) \rightarrow (\forall u E_{12}(u,v) \rightarrow P_{12}(u))) \wedge \neg(\forall v E_2(v,u) \rightarrow (\forall u E_{21}(u,v) \rightarrow P_{21}(u)))\end{aligned}$$

Here, each conjunct can be evaluated for satisfiability or validity independently from the other. A similar formula would be translated using the standard translation to $(\forall y(R_1(x,y) \rightarrow (\forall z R_2(y,z)))) \wedge \neg(\forall y'(R_2(x,y') \rightarrow (\forall z' R_1(y',z') \rightarrow P(z'))))$. This formula cannot be evaluated by splitting it into its two conjuncts and using the evaluation of the conjuncts to evaluate the formula, because the satisfiability of both conjuncts cannot be checked independently. This is due to the fact that both conjuncts share the same underlying structure of the accessibility relation R and they consider the same set of possible worlds. Because of this, the predicates $P(z)$ and $P(z)$ cannot be evaluated independently.

Our translation does not have this problem, which is due to the use of the set-based semantics as opposed to a Kripke-style one. For formulas of a similar form but where the conjuncts do not contain K-operators for the same agents, it is apparent that the presented line of argumentation still holds.

Thus, we have shown that the presented translation approach using the translation function $\boldsymbol{R}[\cdot]$ overcomes the main problems that occur with the other approaches.

6 Conclusion

In this thesis, we set out to develop a multi-agent extension of Feng's translation approach for single-agent systems, aiming to provide a method for theorem proving in first-order multi-agent modal systems that is based on a set-based semantics. Our work has sought to address the inherent complexities involved in deductive reasoning of multi-agent epistemic logics, particularly those complexities that arise from the interdependencies and nested modalities that characterize such systems.

We began by discussing the theoretical underpinnings of modal logic, emphasizing the motivations for studying reasoning about knowledge in multi-agent contexts. This exploration highlighted the need for enhanced deductive reasoning methods that can cope with the increased complexity brought about by multiple interacting agents and their corresponding knowledge states.

The problem statement focused on the challenges associated with satisfiability checking in modal logics, which becomes noticeably more complex when modalities and multi-agent scenarios are involved. Given these challenges, we reviewed the current state of research — including both direct proving techniques for single-agent systems, such as tableau calculi and modal resolution, and the alternative approach of translating modal logic into first-order logic. While translation-based methods can use the powerful capabilities of first-order theorem provers, they come with their own set of challenges, such as increased formula size and the difficulty of representing specific frame conditions like transitivity and Euclideanity.

To address these challenges, we introduced our translation function $\boldsymbol{R}[\cdot]$, building on Feng's translation for single-agent systems and extending it to handle multi-agent epistemic logics. This function was shown to preserve the satisfiability of the original formulas of $\mathcal{KL}_n$, thus providing a sound and reliable way to translate epistemic statements into first-order logic. Our approach enables the use of automated theorem provers like VAMPIRE to perform efficient reasoning in multi-agent contexts.

We have posited two desiderata for a translation approach, namely that it translates into a decidable fragment of FOL, like GF^2, and that the resulting formula is of a size that can still be efficiently evaluated. We have subsequently shown that our translation approach is able to fulfil both desiderata (at least for the propositional part of our language $\mathcal{KL}_n$), which is especially relevant given that most translation approaches are not able to satisfy the first desideratum and those that do often fail to satisfy the second one. Given that the main difference between our translation approach and other approaches is that we base our semantics on sets, which eliminates the need for a valuation function, while all of the existing approaches are based on simulation and axiomatization of Kripke structures, we conclude that translation approaches like ours might be a superior option to expressing modal properties in first-order language.

With this, we have provided a satisfiability-preserving translation function for multi-agent modal languages like $\mathcal{KL}_n$. This is particularly noteworthy given that the majority of existing literature focuses predominantly on single-agent systems.

While this thesis has laid the foundation for an efficient translation approach to theorem proving in multi-agent epistemic modal systems, several open questions and opportunities for further research remain.

A possible area for future research is the extension of the translation approach to other modal systems, particularly to systems like KD45 and S5, as well as their multi-agent variants KD45$_n$ and S5$_n$. These systems are frequently used for the modeling of knowledge and belief, and exploring how the translation function can be adapted to accommodate these additional modal properties would be highly valuable for broadening the applicability of the method.

Another potential extension is to adapt the multi-agent translation approach for use in the context of only-knowing. This would enable reasoning not only about what agents know but also about the absence of knowledge, which is critical for decision-making scenarios in which agents must be aware of their limitations in knowledge.

Additionally, although we provided a methodology for implementing the translation function and presented pseudocode, the actual development of a functioning software implementation remains to be completed. This implementation would allow the translation to be practically utilized, enabling automated reasoning in real-world multi-agent scenarios.

7 Bibliography

Andréka, H., Németi, I., & van Benthem, J. (1998). Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, *27*(3), 217–274. https://doi.org/10.1023/a:1004275029985 (cit. on p. 44).

Auffray, Y., & Enjalbert, P. (1992). Modal theorem proving: An equational viewpoint. *Journal of Logic and Computation*, *2*(3), 247–295 (cit. on pp. 2, 16, 17, 25, 44).

Balint, A., Belov, A., Järvisalo, M., & Sinz, C. (2015). Overview and analysis of the sat challenge 2012 solver competition. *Artificial Intelligence*, *223*, 120–155 (cit. on p. 11).

Barrett, C., Sebastiani, R., Seshia, S. A., & Tinelli, C. (2021). Satisfiability modulo theories. In *Handbook of satisfiability* (pp. 1267–1329). IOS Press. (Cit. on p. 11).

Barrett, C., Stump, A., Tinelli, C., et al. (2010). The smt-lib standard: Version 2.0. *Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, UK)*, *13*, 14 (cit. on p. 12).

Barrett, C. W., Dill, D. L., & Stump, A. (2002). Checking satisfiability of first-order formulas by incremental translation to sat. *Computer Aided Verification: 14th International Conference, CAV 2002 Copenhagen, Denmark, July 27–31, 2002 Proceedings 14*, 236–249 (cit. on p. 11).

Belle, V., Bolander, T., Herzig, A., & Nebel, B. (2023). Epistemic planning: Perspectives on the special issue. (Cit. on p. 1).

Belle, V., & Lakemeyer, G. (2010). Multi-agent only-knowing revisited. *Twelfth International Conference on the Principles of Knowledge Representation and Reasoning* (cit. on pp. 30, 31).

Belle, V., & Lakemeyer, G. (2015). Semantical considerations on multiagent only knowing. *Artificial Intelligence*, *223*, 1–26 (cit. on pp. 28, 30, 31).

Benaim, S., Benedikt, M., Charatonik, W., Kieroński, E., Lenhardt, R., Mazowiecki, F., & Worrell, J. (2013). Complexity of two-variable logic on finite trees. *ACM Transactions on Computational Logic (TOCL)*, *17*(4), 1–38 (cit. on p. 15).

Benzmüller, C., & Paulson, L. (2008). Exploring properties of normal multimodal logics in simple type theory with leo-ii. *Festschrift in Honor of Peter B. Andrews on His 70th Birthday*, 386–406 (cit. on p. 2).

Blackburn, P., Rijke, M. d., & Venema, Y. (2001). *Modal logic*. Cambridge University Press. (Cit. on pp. 1, 6, 46).

Blanchette, J. C., & Paskevich, A. (2013). Tff1: The tptp typed first-order form with rank-1 polymorphism. *Automated Deduction–CADE-24: 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings 24*, 414–420 (cit. on p. 13).

Bledsoe, W. W., & Henschen, L. J. (1985). What is automated theorem proving? *Journal of Automated Reasoning*, *1*(1), 23–28 (cit. on p. 12).

Bonacina, M. P. (2001). A taxonomy of theorem-proving strategies. In *Artificial intelligence today: Recent trends and developments* (pp. 43–84). Springer. (Cit. on p. 12).

Chellas, B. F. (1980). *Modal logic: An introduction*. Cambridge university press. (Cit. on p. 6).

Church, A. (1936). An unsolvable problem of elementary number theory. american journal of mathematics, 58, 345-63. *Repr. in Davis*, *1965*, 88–107 (cit. on p. 44).

Cook, S. A. (1971). The complexity of theorem-proving procedures, 151–158 (cit. on pp. 1, 9, 15).

De Moura, L., & Bjørner, N. (2008). Z3: An efficient smt solver. *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, 337–340 (cit. on p. 11).

Debart, F., Enjalbert, P., & Lescot, M. (1992). Multimodal logic programming using equational and order-sorted logic. *Theoretical computer science*, *105*(1), 141–166 (cit. on p. 18).

Demri, S., & De Nivelle, H. (2005). Deciding regular grammar logics with converse through first-order logic. *Journal of Logic, Language and Information*, *14*, 289–329 (cit. on pp. 2, 3, 16, 21, 45, 48).

Doijade, M. M., & Kulkarni, D. B. (2014). Overview of sequential and parallel sat solvers. *International Conference on Information Communication and Embedded Systems (ICICES2014)*, 1–4 (cit. on p. 11).

Enjalbert, P., & Del Cerro, L. F. (1989). Modal resolution in clausal form. *Theoretical Computer Science*, *65*(1), 1–33 (cit. on pp. 2, 24–26).

Fagin, R., & Vardi, M. Y. (1985). An internal semantics for modal logic. *Proceedings of the Seventeenth Annual ACM symposium on Theory of Computing*, 305–315 (cit. on p. 5).

Fitting, M. (1972). Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, *13*(2), 237–247 (cit. on pp. 2, 24, 25).

Ganzinger, H., Meyer, C., & Veanes, M. (1999). The two-variable guarded fragment with transitive relations. *Proceedings. 14th Symposium on Logic in Computer Science (Cat. No. PR00158)*, 24–34 (cit. on p. 2).

Garson, J. (2024). Modal Logic. In E. N. Zalta & U. Nodelman (Eds.), *The Stanford encyclopedia of philosophy* (Spring 2024). Metaphysics Research Lab, Stanford University. (Cit. on p. 6).

Grädel, E. (1999). On the restraining power of guards. *The Journal of Symbolic Logic*, *64*(4), 1719–1742 (cit. on pp. 2, 15, 44).

Grädel, E., Kolaitis, P. G., & Vardi, M. Y. (1997). On the decision problem for two-variable first-order logic. *Bulletin of symbolic logic*, *3*(1), 53–69 (cit. on p. 15).

Hadarean, L., Bansal, K., Jovanović, D., Barrett, C., & Tinelli, C. (2014). A tale of two solvers: Eager and lazy approaches to bit-vectors. *International Conference on Computer Aided Verification*, 680–695 (cit. on p. 10).

Halpern, J. Y., & Lakemeyer, G. (2001). Multi-agent only knowing. *Journal of Logic and Computation*, *11*(1), 41–70 (cit. on pp. 27, 28).

Halpern, J. Y., & Moses, Y. (1990). Knowledge and common knowledge in a distributed environment. *Journal of the ACM (JACM)*, *37*(3), 549–587 (cit. on p. 1).

Halpern, J. Y., & Moses, Y. (1992). A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial intelligence*, *54*(3), 319–379 (cit. on pp. 1, 5, 7–9, 15, 16).

Hintikka, J. (1962). Knowledge and belief: An introduction to the logic of the two notions. (Cit. on pp. 5, 7).

Hölldobler, S., Manthey, N., Stecklina, J., Steinke, P., et al. (2011). A short overview on modern parallel sat-solvers. *2011 International Conference on Advanced Computer Science and Information Systems*, 201–206 (cit. on p. 11).

Huang, Z., & Zhang, J. (2005). Generating sat instances from first-order formulas. *Journal of Software*, *16*(3), 327–335 (cit. on p. 11).

Huth, M., & Ryan, M. (2008). Logic in computer science: Modelling and reasoning about systems. *Journal of Functional Programming*, *18*(3), 421–422 (cit. on p. 4).

Jamroga, W., & Penczek, W. (2010). Specification and verification of multi-agent systems. *European Summer School in Logic, Language and Information*, 210–263 (cit. on p. 4).

Kong, W., Ogata, K., Seino, T., & Futatsugi, K. (2005). A lightweight integration of theorem proving and model checking for system verification. *12th Asia-Pacific Software Engineering Conference (APSEC'05)*, 8–pp (cit. on p. 12).

Kovács, L., & Voronkov, A. (2013). First-order theorem proving and vampire. *International Conference on Computer Aided Verification*, 1–35 (cit. on pp. 2, 13, 14).

Kramer, S., & Sack, J. (2012). Parametric constructive kripke-semantics for standard multi-agent belief and knowledge (knowledge as unbiased belief). *arXiv preprint arXiv:1209.1885* (cit. on p. 1).

Kripke, S. A. (1963). Semantical analysis of modal logic i normal modal propositional calculi. *Mathematical Logic Quarterly*, *9*(5-6), 67–96 (cit. on pp. 1, 8).

Kroening, D., & Strichman, O. (2016). *Decision procedures*. Springer. (Cit. on pp. 9–11).

Ladner, R. E. (1977). The computational complexity of provability in systems of modal propositional logic. *SIAM journal on computing*, *6*(3), 467–480 (cit. on p. 15).

Levesque, H. J. (1981). The interaction with incomplete knowledge bases: A formal treatment. *IJCAI*, 240–245 (cit. on p. 27).

Levesque, H. J., & Lakemeyer, G. (2001). *The logic of knowledge bases*. Mit Press. (Cit. on pp. 1, 27, 28, 31, 34).

Massacci, F. (1998). Tableau methods for formal verification of multi-agent distributed systems. *Journal of Logic and Computation*, *8*(3), 373–400 (cit. on p. 2).

Meltzer, B. (1966). Theorem-proving for computers: Some results on resolution and renaming. *The Computer Journal*, *8*(4), 341–343 (cit. on p. 14).

Moore, R. C. (1985). A formal theory of knowledge and action. *Formal theories of the commonsense world*, 319–358 (cit. on p. 5).

Morgan. (1976). Methods for automated theorem proving in nonclassical logics. *IEEE Transactions on Computers*, *100*(8), 852–862 (cit. on pp. 2, 14).

Nonnengart, A. (1993). First-order modal logic theorem proving and functional simulation. *IJCAI*, *93*, 80–85 (cit. on pp. 2, 16, 19, 20, 25, 45).

Ohlbach, H. J. (1991). Semantics-based translation methods for modal logics. *Journal of Logic and Computation*, *1*(5), 691–746 (cit. on pp. 16, 21).

Ohlbach, H. J., & Schmidt, R. A. (1997). Functional translation and second-order frame properties of modal logics. *Journal of Logic and Computation*, *7*(5), 581–603 (cit. on pp. 2, 20, 45).

Otten, J., & Sutcliffe, G. (2010). Using the tptp language for representing derivations in tableau and connection calculi. *Proceedings of the Workshop on Practical Aspects of Automated Reasoning, 5th International Joint Conference on Automated Reasoning*, 90–100 (cit. on p. 13).

Otto, M. (2001). Two variable first-order logic over ordered domains. *The Journal of Symbolic Logic*, *66*(2), 685–702 (cit. on p. 15).

Rabbi, F., Lamo, Y., & Kristensen, L. M. (2017). An mde approach for modelling and reasoning about multi-agent systems. *Multi-Agent Systems and Agreement Technologies: 14th European Conference, EUMAS 2016, and 4th International Conference, AT 2016, Valencia, Spain, December 15-16, 2016, Revised Selected Papers*, 49–57 (cit. on p. 10).

Reger, G., Suda, M., & Voronkov, A. (2014). The challenges of evaluating a new feature in vampire. *Vampire Workshop*, 70–74 (cit. on p. 14).

Rendsvig, R., Symons, J., & Wang, Y. (2024). Epistemic Logic. In E. N. Zalta & U. Nodelman (Eds.), *The Stanford encyclopedia of philosophy* (Summer 2024). Metaphysics Research Lab, Stanford University. (Cit. on p. 5).

Riazanov, A., & Voronkov, A. (2001). Vampire 1.1. *Automated Reasoning: First International Joint Conference, IJCAR 2001 Siena, Italy, June 18–22, 2001 Proceedings 1*, 376–380 (cit. on p. 14).

Riazanov, A., & Voronkov, A. (2002). The design and implementation of vampire. *AI communications*, *15*(2-3), 91–110 (cit. on p. 13).

Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, *12*(1), 23–41 (cit. on p. 25).

Schmidt, R. A. (1997). Optimised modal translation and resolution (cit. on p. 18).

Schmidt, R. A., & Hustadt, U. (2013). First-order resolution methods for modal logics. In *Programming logics: Essays in memory of harald ganzinger* (pp. 345–391). Springer. (Cit. on pp. 23, 24).

Schulz, S. (2017). Light-weight integration of sat solving into first-order reasoners–first experiments. *Vampire Workshop*, 9–19 (cit. on p. 11).

Segerberg, K. (1971). An essay in classical modal logic (cit. on p. 20).

Sprey, J., Sundermann, C., Krieter, S., Nieke, M., Mauro, J., Thüm, T., & Schaefer, I. (2020). Smt-based variability analyses in featureide. *Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems*, 1–9 (cit. on p. 11).

Tinelli, C. (2007). An abstract framework for satisfiability modulo theories. *TABLEAUX*, *4548* (cit. on p. 10).

Turing, A. (1936). On computable numbers, with an application to the entscheidungs problem. *Proceedings of the London Mathematical Society Series/2 (42)*, 230–42 (cit. on p. 44).

van Benthem, J. (1977). *Modal correspondence theory* [Doctoral dissertation, University of Amsterdam]. (Cit. on p. 2).

van Benthem, J. (1984). Correspondence theory. In *Handbook of philosophical logic: Volume ii: Extensions of classical logic* (pp. 167–247). Springer. (Cit. on p. 2).

van Benthem, J., & Blackburn, P. (1988). Modal logic: A semantic perspective (cit. on p. 16).

Vardi, M. Y. (1996). Why is modal logic so robustly decidable? *Descriptive complexity and finite models*, *31*, 149–184 (cit. on p. 44).

A Appendix

A.1 Proof $R[\cdot]$ Single-Agent Case

A model (M, μ) in FOL that is constructed based on a $\mathcal{KL}$ model (e, w) must satisfy the following conditions:

C1 The domain of discourse is $\mathcal{D} = \mathcal{D}_o \cup \mathcal{D}_w$, where the domain of sort object $\mathcal{D}_o$ is isomorphic to the set of standard name $\mathcal{N}$ in $\mathcal{KL}$, and the domain of sort world $\mathcal{D}_w$ is isomorphic to $\mathcal{W}$;[1]

C2 Symbols of standard names are treated as object constants. For any $n \in \mathcal{N}$ there is a unique element $d \in \mathcal{D}_o$ s.t. $n^{\mathcal{M}} = d$, and for each element $d' \in \mathcal{D}_o$, there's a unique co-referring standard name $n' \in \mathcal{N}$ s.t. $n'^{\mathcal{M}} = d'$;

C3 Equality is interpreted in the usual sense, i.e. $d =^{\mathcal{M}} d$ and $d \neq^{\mathcal{M}} d'$ for any distinct elements $d, d' \in \mathcal{D}$.

Using this, we interpret a substitution P_W as $P_W : \underbrace{\mathcal{D}_o \times \cdots, \times \mathcal{D}_o}_{k} \times \mathcal{D}_w$ and $F_W : \underbrace{\mathcal{D}_o \times \cdots, \times \mathcal{D}_o}_{k} \times \mathcal{D}_w \to \mathcal{D}_o$.

C4 f.a. $n_1, \cdots, n_k \in \mathcal{N}$ $(n_1^{\mathcal{M}}, \cdots n_k^{\mathcal{M}}, w_0^{\mathcal{M}}) \in P_W^{\mathcal{M}}$ iff $(n_1, \cdots, n_k) \in w[P]$;

C5 f.a. $w' \in e$, $(n_1^{\mathcal{M}}, \cdots, n_k^{\mathcal{M}}, w') \in P_W^{\mathcal{M}}$ iff $(n_1, \cdots, n_k) \in w'[P]$

C6 f.a. $n_1, \cdots, n_k \in \mathcal{N}$, $F_W^{\mathcal{M}}(n_1^{\mathcal{M}}, \cdots n_k^{\mathcal{M}}, w_0^{\mathcal{M}}) = w[F(n_1, \cdots, n_k)]$

C7 f.a. $w' \in e$, $F_W^{\mathcal{M}}(n_1^{\mathcal{M}}, \cdots n_k^{\mathcal{M}}, w') = w'[F(n_1, \cdots, n_k)]$

C8 f.a. $w', w' \in E^{\mathcal{M}}$ iff $w' \in e$

Given a $\mathcal{KL}$ model (e, w) we can construct a corresponding FOL model $(\mathcal{M}, \mu)$ that satisfies the above mentioned conditions by using the translation function presented in the previous section.

Lemma 4. *Let μ' be a variable map s.t. $\mu' \sim_u \mu$, $\mu'(u) = w'$ and $w' \in e$, then*

1. *For any ground terms t, t',*

$$e, w' \models (t = t') \textit{ iff } \mathcal{M}, \mu' \models (\boldsymbol{R}[t = t]) \uparrow_u \tag{A.1}$$

[1] Note that we use the same labels $w, w' \ldots$ to denote elements of $\mathcal{D}_w$ or of $\mathcal{W}$, and do not distinguish them.

2. *For any atom of form* $P(t_1, ..., t_k)$*,*

$$e, w' \models P(t_1, ..., t_k) \textit{ iff } \mathcal{M}, \mu' \models (\boldsymbol{R}[P(t_1, ..., t_k)]) \uparrow_u \tag{A.2}$$

Proof. We will demonstrate an inductive proof only for the first case, the second case can then be proven in a similar manner.

Basis: Suppose that $t = F(n_1, \dots n_k)$ and $t' = F'(n'_1, \dots n'_{k'})$ (note that, when t or t' is a standard name, the proof is similar) we have

$$\begin{aligned}
& e, w' \models (t = t') && \text{(A.3)}\\
\Longleftrightarrow & w'[F(n_1, \dots, n_k)] = w'[F'(n'_1, \dots, n'_{k'})] && \text{(A.4)}\\
\Longleftrightarrow & {F_W}^{\mathcal{M}}(n_1^{\mathcal{M}}, \cdots n_k^{\mathcal{M}}, w') = {F'_W}^{\mathcal{M}}({n'_1}^{\mathcal{M}}, \cdots {n'_{k'}}^{\mathcal{M}}, w') && \text{(cond C7.)}\\
\Longleftrightarrow & \mathcal{M}, \mu' \models F_W(n_1, \dots n_k, u) = F_W(n'_1, \dots, n'_{k'}, u) && (\mu'(u) = w')\\
\Longleftrightarrow & \mathcal{M}, \mu' \models (\boldsymbol{R}[t = t']) \uparrow_u && \text{(Def. } \boldsymbol{R}[\cdot] \text{ and } \uparrow_u)
\end{aligned}$$

Induction: Given $t = F(t_1, \dots t_k)$ and $t' = F'(t'_1, \dots t'_{k'})$, suppose that

$$\begin{aligned}
& e, w' \models t_i = n_i \text{ iff } \mathcal{M}, \mu' \models (\boldsymbol{R}[t_i = n_i]) \uparrow_u \text{ for } i \in \{1, \dots, k\}\\
& e, w' \models t'_j = n'_j \text{ iff } \mathcal{M}, \mu' \models (\boldsymbol{R}[t'_j = n'_j]) \uparrow_u \text{ for } j \in \{1, \dots, k'\}
\end{aligned}$$

Let $\hat{t}_i$ be the term obtained by replacing any functional term $f(\hat{t}_1, \dots, \hat{t}_l)$ in t_i with $f_W(\hat{t}_1, \dots, \hat{t}_l, u)$ recursively (e.g. if $t_i = f(g(x), n)$, then $\hat{t}_i = f_W(g_W(x, u), y, u)$). Then $(\boldsymbol{R}[t_i = n_i]) \uparrow_u$ can be

re-written as $\hat{t}_i = n_i$ (analogously we build $\hat{t'_j}$ for each t'_j). Using this, we can now show that (1.) holds:

$$
\begin{aligned}
& e, w' \models (t = t') \\
\Longleftrightarrow & w'(F(t_1, \ldots, t_k)) = w'(F'(t'_1, \ldots, t'_{k'})) \\
\Longleftrightarrow & w'[F(n_1, \ldots, n_k)] = w'[F'(n'_1, \ldots, n'_{k'})] \\
& \text{where } n_i = w'(t_i) \text{ and } n'_j = w'(t'_j) \text{ f.a. } i, j \\
\Longleftrightarrow & w'[F(n_1, \ldots, n_k)] = w'[F'(n'_1, \ldots, n'_{k'})] \\
& \text{where } e, w' \models n_i = t_i \text{ and } e, w' \models n'_j = t'_j \text{ f.a. } i, j \\
\Longleftrightarrow & \mathcal{M}, \mu' \models F_W(n_1, \ldots n_k, u) = F_W(n'_1, \ldots, n'_{k'}, u) \\
& \text{where } \mathcal{M}, \mu' \models (\boldsymbol{R}[n_i = t_i]) \uparrow_u \text{ and } \mathcal{M}, \mu' \models (\boldsymbol{R}[n'_j = t'_j]) \uparrow_u \text{ f.a. } i, j \quad \text{(Induction hypothesis)} \\
\Longleftrightarrow & \mathcal{M}, \mu' \models F_W(n_1, \ldots n_k, u) = F_W(n'_1, \ldots, n'_{k'}, u) \\
& \text{where } \mathcal{M}, \mu' \models n_i = \hat{t}_i \text{ and } \mathcal{M}, \mu' \models n'_j = \hat{t'}_j \text{ f.a. } i, j \qquad \text{(Def. } \hat{t}_i, \hat{t'}_j\text{)} \\
\Longleftrightarrow & \mathcal{M}, \mu' \models F_W(n_1, \ldots n_k, u) = F_W(n'_1, \ldots, n'_{k'}, u) \\
& \text{where } {n_i}^{\mathcal{M}} = \hat{t}_i^{(\mathcal{M}, \mu')} \text{ and } {n'_j}^{\mathcal{M}} = \hat{t'}_j^{\,(\mathcal{M}, \mu')} \text{ f.a. } i, j \\
\Longleftrightarrow & \mathcal{M}, \mu' \models F_W(\hat{t_1}, \ldots \hat{t_k}, u) = F_W(\hat{t'_1}, \ldots, \hat{t'_{k'}}, u) \qquad \text{(FOL semantics)} \\
\Longleftrightarrow & \mathcal{M}, \mu' \models (\boldsymbol{R}[F(t_1, \ldots t_k) = F(t'_1, \ldots, t'_{k'})]) \uparrow_u \qquad \text{(Def. } \boldsymbol{R}[\cdot] \text{ and } \uparrow_u\text{)} \\
\Longleftrightarrow & \mathcal{M}, \mu' \models (\boldsymbol{R}[t = t']) \uparrow_u
\end{aligned}
$$

□

Lemma 5. *Given $\mathcal{KL}$ model (e, w), the corresponding FO model $(\mathcal{M}, \mu)$ is constructed as above. For any $w' \in e$, let μ' be a variable map s.t. $\mu' \sim_u \mu$ and $\mu'(u) = w'$, then for any $\alpha \in \mathcal{KL}$, $e, w' \models \alpha$ iff $\mathcal{M}, \mu' \models (\boldsymbol{R}[\alpha]) \uparrow_u$.*

Proof. We give an inductive proof for Lemma 5.

Basis: From Lemma 4 we know $e, w' \models \alpha$ holds iff $\mathcal{M}, \mu' \models (\boldsymbol{R}[\alpha]) \uparrow_u$ for any atom α.

Induction:

- Given α, α', suppose that f.a. $w'' \in e, \mu'' \sim_u \mu$ with $\mu''(u) = w''$,

$$
\begin{aligned}
e, w'' \models \alpha \text{ iff } \mathcal{M}, \mu'' \models (\boldsymbol{R}[\alpha]) \uparrow_u \\
e, w'' \models \alpha' \text{ iff } \mathcal{M}, \mu'' \models (\boldsymbol{R}[\alpha']) \uparrow_u
\end{aligned}
$$

  For $\neg\alpha$, $\alpha \wedge \alpha'$, the proof follows the definition of $\boldsymbol{R}[\cdot]$ and $\uparrow_u$.

- Suppose that that the lemma holds for α_n^x f.a. $n \in \mathcal{N}$,

$$\begin{aligned}
& e, w' \models \forall x.\alpha \\
\Longleftrightarrow & e, w' \models \alpha_n^x \text{ f.a. } n \in \mathcal{N} \\
\Longleftrightarrow & \mathcal{M}, \mu' \models ((\boldsymbol{R}[\alpha]) \uparrow_u)_n^x \text{ f.a. } n \in \mathcal{N} \\
\Longleftrightarrow & \mathcal{M}, \mu'' \models (\boldsymbol{R}[\alpha]) \uparrow_u \text{ f.a. } n \in \mathcal{N}, \mu'' \sim_x \mu', \mu''(x) = n^{\mathcal{M}} \\
\Longleftrightarrow & \mathcal{M}, \mu'' \models (\boldsymbol{R}[\alpha]) \uparrow_u \text{ f.a. } \mu'' \sim_x \mu' && \text{(cond. C2)} \\
\Longleftrightarrow & \mathcal{M}, \mu'' \models (\boldsymbol{R}[\forall x.\alpha]) \uparrow_u
\end{aligned}$$

- For $\boldsymbol{K}\alpha$ and $w' \in e$,

$$\begin{aligned}
& e, w' \models \boldsymbol{K}\alpha \\
\Longleftrightarrow & \text{ f.a. } w'' \in e, e, w'' \models \alpha \\
\Longleftrightarrow & \text{ f.a. } w'' \in e, \mu'' \sim_u \mu \text{ with } \mu''(u) = w'', \mathcal{M}, \mu'' \models (\boldsymbol{R}[\alpha]) \uparrow_u && \text{(Induction hypothesis)} \\
\Longleftrightarrow & \text{ f.a. } w'' \in \mathcal{W}, \mu'' \sim_u \mu \text{ with } \mu''(u) = w'', \mathcal{M}, \mu'' \models E(u) \supset (\boldsymbol{R}[\alpha]) \uparrow_u && \text{(cond. C8)} \\
\Longleftrightarrow & \mathcal{M}, \mu \models \forall u.E(u) \supset (\boldsymbol{R}[\alpha]) \uparrow_u \\
\Longleftrightarrow & \mathcal{M}, \mu' \models \forall u.E(u) \supset (\boldsymbol{R}[\alpha]) \uparrow_u && \text{(Def. } \mu') \\
\Longleftrightarrow & \mathcal{M}, \mu' \models \boldsymbol{R}[\boldsymbol{K}\alpha] && \text{(Def. } R[\cdot])
\end{aligned}$$

□

Corollary 1. *Let $e, w, \mathcal{M}, \mu$ be defined as above. For any $\alpha \in \mathcal{KL}$, $e, w \models \boldsymbol{K}\alpha$ iff $\mathcal{M}, \mu \models \boldsymbol{R}[\boldsymbol{K}\alpha]$*

Theorem 3. *Let $e, w, \mathcal{M}, \mu$ be defined as above. For any $\alpha \in \mathcal{KL}$, $e, w \models \alpha$ iff $\mathcal{M}, \mu \models \boldsymbol{R}[\alpha]$*

Proof. Again, we consider the induction on $\mathcal{KL}$ formulas:

Basis: For any atomic sentence α, the proof is similar to Lemma 4. The only difference is that we consider $\uparrow_{w_0}$ instead of $\uparrow_u$, and replace every application of condition C5/C7 by C4/C6.

Induction:

- Given α, α', suppose that

$$\begin{aligned}
e, w \models \alpha \text{ iff } \mathcal{M}, \mu \models \boldsymbol{R}[\alpha] \\
e, w \models \alpha' \text{ iff } \mathcal{M}, \mu \models \boldsymbol{R}[\alpha']
\end{aligned}$$

  For $\neg\alpha$ and $\alpha \wedge \alpha'$, the proof follows the definition of $\boldsymbol{R}[\cdot]$.

- $\forall x.\alpha$ bases on the induction hypothesis that the lemma holds for α_n^x f.a. $n \in \mathcal{N}$.

- For $\boldsymbol{K}\alpha$,

$$
\begin{aligned}
& e, w \models \boldsymbol{K}\alpha \\
\Longleftrightarrow & \text{f.a. } w' \in e, e, w' \models \alpha \\
\Longleftrightarrow & \text{f.a. } w' \in e, \mu' \sim_u \mu \text{ with } \mu'(u) = w', \mathcal{M}, \mu' \models (\boldsymbol{R}[\alpha]) \uparrow_u && \text{(Lemma 5)} \\
\Longleftrightarrow & \text{f.a. } w' \in \mathcal{W}, \mu' \sim_u \mu \text{ with } \mu'(u) = w', \mathcal{M}, \mu' \models E(u) \supset (\boldsymbol{R}[\alpha]) \uparrow_u && \text{(cond. C8)} \\
\Longleftrightarrow & \mathcal{M}, \mu \models \forall u.E(u) \supset (\boldsymbol{R}[\alpha]) \uparrow_u \\
\Longleftrightarrow & \mathcal{M}, \mu \models \boldsymbol{R}[\boldsymbol{K}\alpha] && \text{(Def. } R[\cdot]\text{)}
\end{aligned}
$$

□

Furthermore, in $\mathcal{KL}$ the unique name assumption is made regarding the domain of quantification, correspondingly, we need the same assumption when translating a $\mathcal{KL}$ formula into FO-formula:

- Σ_{eq} is the following infinite set of formulas:
  - $\forall x(x = x)$
  - $\forall x \forall y.(x = y) \supset (y = x)$
  - $\forall x \forall y \forall z((x = y) \wedge (y = z)) \supset (x = z)$
  - for any function symbol F:
    $\forall x_1 \ldots \forall x_k \forall y_1 \ldots \forall y_k((x_1 = y_1) \wedge \ldots \wedge (x_k = y_k)) \supset F(x_1, \ldots, x_k) = F(y_1, \ldots, y_k)$
  - for any predicate symbol P:
    $\forall x_1 \ldots \forall x_k \forall y_1 \ldots \forall y_k((x_1 = y_1) \wedge \ldots \wedge (x_k = y_k)) \supset P(x_1, \ldots, x_k) \equiv P(y_1, \ldots, y_k)$
- $\Sigma_{una} = \{(n \neq n') \mid n, n' \text{ are distinct standard names in } \mathcal{N}\}$

Intuitively, Σ_{eq} means that the symbol $=$, essentially as a binary predicate, is interpreted as its usual sense. Σ_{una} requires that the domain of discourse must be infinite.

Corollary 2. *For $\alpha \in \mathcal{KL}$, if α is satisfiable, then $\boldsymbol{R}[\alpha] \cup \Sigma_{una} \cup \Sigma_{eq}$ is FO-satisfiable.*

Proof. Theorem 1 shows that $(\mathcal{M}, \mu)$ is a model of $\boldsymbol{R}[\alpha]$, so we only need to prove that $(\mathcal{M}, \mu)$ also satisfies $\Sigma_{una} \cup \Sigma_{eq}$. This is guaranteed since we have specified the interpretation of "=" in $\mathcal{M}$. □

Inversely, we expect that the satisfiability of $\mathcal{KL}$ sentences can be inferred from the satisfiability of FOL sentences, this can be proven via the following lemma and theorem:

Lemma 6. *For $\alpha \in \mathcal{KL}$, if $\boldsymbol{R}[\alpha] \cup \Sigma_{eq} \cup \Sigma_{una}$ is FO-satisfiable, then it exists a model $(\mathcal{M}, \mu)$ s.t.*

C1' $(\mathcal{M}, \mu)$ satisfies $\boldsymbol{R}[\alpha] \cup \Sigma_{eq} \cup \Sigma_{una}$;

C2' The world domain $\mathcal{D}_w$ is countable;

C3' *Equality of objects is interpreted in the usual sense, i.e.* $d =^{\mathcal{M}} d$ *and* $d \neq^{\mathcal{M}} d'$ *for any distinct elements* $d, d' \in \mathcal{D}_o$.

C4' *The object domain* $\mathcal{D}_o$ *is isomorphic to* $\mathcal{N}$*, i.e. for any* $n \in \mathcal{N}$ *there is a unique element* $d \in \mathcal{D}_o$ *s.t.* $n^{\mathcal{M}} = d$*, and for each element* $d' \in \mathcal{D}_o$*, there's a unique co-referring standard name* $n' \in \mathcal{N}$ *s.t.* $n'^{\mathcal{M}} = d'$.

Proof. $\boldsymbol{R}[\alpha] \cup \Sigma_{una} \cup \Sigma_{eq}$ is satisfiable iff it's satisfied in a countable domain. Suppose that $(\mathcal{M}, \mu)$ is a model of $\boldsymbol{R}[\alpha] \cup \Sigma_{eq} \cup \Sigma_{una}$, where both $\mathcal{D}_o$ and $\mathcal{D}_w$ are countable. We first construct a new model $(\mathcal{M}', \mu')$ which satisfies C3'.

Suppose that d, d' are distinct elements in $\mathcal{D}_o$ s.t. $d =^{\mathcal{M}} d'$. W.l.o.g. d' does not have a co-referring standard name (If both d, d' are referred to standard names, then Σ_{una} is not satisfied). Let $(\mathcal{M}', \mu')$ be a model with object domain $\mathcal{D}'_o = \mathcal{D}_o \backslash \{d'\}$. For any variable x s.t. $\mu(x) = d'$, we require that $\mu'(x) = d$. For the rest, $(\mathcal{M}', \mu')$ is identical with $(\mathcal{M}, \mu)$. Through induction on sentence α we can prove that $(\mathcal{M}', \mu')$ is a model of $\boldsymbol{R}[\alpha] \cup \Sigma_{una} \cup \Sigma_{eq}$. Now we obtain a model $(\mathcal{M}', \mu')$ which satisfies $\boldsymbol{R}[\alpha] \cup \Sigma_{una} \cup \Sigma_{eq}$ and interprets the equality of object in the usual sense. Suppose that $n_1, \ldots, n_m$ are all the standard names used in α. From the definition of $\boldsymbol{R}[\cdot]$ we know $n_1, \ldots, n_m$ are also all constants in $\boldsymbol{R}[\alpha]$. Since $\mathcal{M}', \mu \models \Sigma_{una}$, we know that the domain of object $\mathcal{D}'_o$ is countably infinite. We re-number the elements in $(\mathcal{D}'_o \backslash \{n_1^{\mathcal{M}}, \ldots, n_m^{\mathcal{M}}\})$ as $\{d'_1, d'_2, \ldots\}$ and re-number $(\mathcal{N} \backslash \{n_1, \ldots n_m\})$ as $\{n'_1, n'_2, \ldots\}$, then we can assign each element in $\mathcal{D}_o$ a co-referring standard name by re-defining the interpretation of irrelevant constants: We define $\mathcal{M}''$ s.t.

- $n_i'^{\mathcal{M}''} = d'_i$ for $i \in \mathbb{N}$;
- $\mathcal{M}''$ interprets everything else as $\mathcal{M}'$.

Since $\mathcal{M}''$ and $\mathcal{M}'$ only differ in the interpretation on constants which do not occur in $\boldsymbol{R}[\alpha]$, we conclude that $(\mathcal{M}'', \mu')$ satisfies all the conditions in lemma. □

Theorem 4. *For* $\alpha \in \mathcal{KL}$*, if* $\boldsymbol{R}[\alpha] \cup \Sigma_{eq} \cup \Sigma_{una}$ *is FO-satisfiable, then* α *is satisfiable.*

Proof. Suppose that $\boldsymbol{R}[\alpha] \cup \Sigma_{eq} \cup \Sigma_{una}$. From the lemma above we have a model $(\mathcal{M}, \mu)$ which satisfies the sentences, has a countable domain and the object sub-domain $\mathcal{D}_o$ is isomorphic to $\mathcal{N}$.

We construct a world w which satisfies:

- f.a. $n_1, \cdots, n_k$ and P_W, $(n_1, \cdots, n_k) \in w[P]$ iff $(n_1^{\mathcal{M}}, \cdots, n_k^{\mathcal{M}}) \in P_W^{\mathcal{M}}$.
- f.a. $n_1, \cdots, n_k, n'$ and F_W,
  $w[F(n_1, \cdots, n_k)] = n'$ iff $F_W^{\mathcal{M}}(n_1^{\mathcal{M}}, \cdots, n_k^{\mathcal{M}}) = n'^{\mathcal{M}}$.

For $d \in \mathcal{D}_w$, we construct a world w_d which satisfies:

- f.a. $n_1, \cdots, n_k$ and P_W, $(n_1, \cdots, n_k) \in w_d[P]$ iff $(n_1^{\mathcal{M}}, \cdots, n_k^{\mathcal{M}}, d) \in P_W^{\mathcal{M}}$.
- f.a. $n_1, \cdots, n_k, n'$ and F_W,
  $w_d[F(n_1, \cdots, n_k)] = n'$ iff $F_W^{\mathcal{M}}(n_1^{\mathcal{M}}, \cdots, n_k^{\mathcal{M}}, d) = n'^{\mathcal{M}}$.

Let $e = \{w_d \mid d \in \mathcal{D}_w\}$, it can be proved that $(\mathcal{M}, \mu)$ is the corresponding model of (e, w) which satisfies the conditions C1-C7. From Theorem 1 we have $e, w \models \alpha$. □

Directly from the proof of Theorem 2, we notice an interesting property:

Corollary 3. *Any satisfiable $\alpha \in \mathcal{KL}$ can be satisfied by a model e, w where e is a countable set.*

From Corollary 2 and Theorem 2, we conclude that

Theorem 5. *For $\alpha \in \mathcal{KL}$, α is satisfiable iff $\Sigma_{una} \cup \Sigma_{eq} \cup \boldsymbol{R}[\alpha]$ is FO-satisfiable.*

Specifically, when equality and standard names are not involved, we can spare the infinite set of axioms:

Proposition 1. *For any $\alpha \in \mathcal{KL}$, suppose that α does not contain standard names or equality, then α is satisfiable iff $\boldsymbol{R}[\alpha]$ is satisfiable.*

A.2 Pseudocode $R[\cdot]$ Single-Agent Case

Algorithm 1 translateStart

Data: formula
Result: Translated FOL Formula
`/* Starts the recursive function translate */`
1 String result = *translate*(formula, "u")
return result

Algorithm 2 translate

Data: formula, var1
Result: Translated FOL Formula

```
/* A counter which ensures fresh variables                              */
int varCounter=0
    /* An array where the result of the partitioning-method is saved
    in                                                                   */
String Array formulaArray = partition(formula)
if formulaArray[0] == ∧:
    return "("translate(formulaArray[1], var1) " ∧ " translate(formulaArray[2], var1) ")"
elif formulaArray[0] == ∨:
    return "("translate(formulaArray[1], var1) " ∨ "translate(formulaArray[2], var1) ")"
elif formulaArray[0] == →:
    return "("translate(formulaArray[1], var1) " → "translate(formulaArray[2], var1) ")"
elif formulaArray[0] == ↔:
    return "("translate(formulaArray[1], var1) " ↔ "translate(formulaArray[2], var1) ")"
elif formulaArray[0] == =:
    return "("translate(formulaArray[1], var1) "=" translate(formulaArray[2], var1) ")"
elif formulaArray[0] == ∀:
    return "∀" formulaArray[1]"( O(" formulaArray[1]") →" translate(formulaArray[2], var1 "))"
elif formulaArray[0] == ∃:
    return "∃" formulaArray[1] "(O(" formulaArray[1]") ∧" translate(formulaArray[2], var1 "))"
elif formulaArray[0] == ¬:
    return "¬" translate(formulaArray[2], var1)
elif formulaArray[0] == K:
    varCounter++
     return "∀" var1 "(E("u_varCounter") ∧ W("u_varCounter")) → (" translate(formulaArray[1]
    ,u_varCounter) "∧E("var1"))"
/* If none of the conditions above hold, then formula is of form
    P(t_1,...,t_2) or F(t_1,...,t_2)                                     */
else:
    return formulaArray[0]"_W ("formulaArray[1]"," u_varCounter")"
```

Algorithm 3 partition

Data: formula

Result: Array containing: outermostOperator, Subformula1, (Subformula2)

```
/* Searches for operator at the highest level, returns array with
   the outermost operator in the first position, then the remaining
   formula(s).                                                      */
```

```
24 binaryOperators = [∧, ∨, →, ↔, =]
   unaryOperators = [K, ∀, ∃, ¬]
   String outermostOperator
   int operatorPosition

   /* If first character of formula is "(", then the outermost operator
      is binary                                                         */
25 if formula[0] == (:
26     stack = []
       for (i=1; i< formula.length; i++ ):
27         character = fomula[i]  if character == (:
28             stack.PUSH(i)
29         elif character == ):
30             stack.POP()
31         elif stack.isEmpty AND character in binaryOperators:
32             outermostOperator = character
               operatorPosition = i
               break
33     String leftSubformula = formula.substring(1, operatorPositions-1)
       String rightSubformula = formula.substring(operatorPositions+1, formula.length-1)
       return [outermostOperator, leftSubformula, rightSubformula]
   /* Outermost operator is unary/only predicate or function is left   */
34 else:
35     outermostOperator = formula[0]
          /* Case where only a function or predicate is left           */
36     if outermostOperator not in unaryOperators:
37         return[formula[0], formula.substring(2, formula.length-1)]
38     elif outermostOperator in [∀, ∃]:
39         return[formula[0], formula[1], formula.substring(2, formula.length)]
40     elif outermostOperator == ¬:
41         return[formula[0], formula.substring(1, formula.length)]
42     elif outermostOperator == K:
43         return[formula[0], formula.substring(1, formula.length)]
44     return error
```

A.3 Proof $R[\cdot]$ Multi-Agent Case

In order to prove the accuracy of the translation function $\boldsymbol{R}[\cdot]$, it s necessary to show that we can construct a satisfiability-equivalent model in FOL using a $\mathcal{KL}_n$ model as a starting point, and similarly, that a satisfiability-equivalent model in $\mathcal{KL}_n$ can be generated from a FOL model. We will show both directions and conclude that, using the translation function $\boldsymbol{R}$, we can translate any formula of $\mathcal{KL}_n$ into a satisfiability-equivalent FOL-formula.

A.3.1 Construction of Model in FOL ($\mathcal{KL}_n \Longrightarrow FOL$)

We construct a canonical FOL model M^c as follows:

C1 Domain of discourse $\mathcal{D} = \mathcal{D}_o \cup \mathcal{D}_w$, where the domain of sort object $\mathcal{D}_o$ is isomorphic to the set of standard name $\mathcal{N}$, and the domain of sort state $\mathcal{D}_w$ is isomorphic to the set of $\mathcal{KL}_n$ models $\mathbb{M}$. We use $f_o \colon \mathcal{N} \to \mathcal{D}_o$ and $f_w \colon \mathbb{M} \to \mathcal{D}_w$ to denote the functions of isomorphism and use f_o^{-1} and f_w^{-1} for the inverse functions. In particular, $n^{M^c} = f_o(n)$ for any $n \in \mathcal{N}$.

C2 Equality is interpreted in the usual sense, i.e. $d =^{M^c} d$ and $d \neq^{M^c} d'$ for any distinct elements d, d'.

We consider the interpretation of P_σ and F_σ in FOL as follows:

- $P_\sigma \subseteq \underbrace{\mathcal{D}_o \times \cdots, \times \mathcal{D}_o}_{j} \times \mathcal{D}_w$ and $F_\sigma : \underbrace{\mathcal{D}_o \times \cdots, \times \mathcal{D}_o}_{j} \times \mathcal{D}_w \to \mathcal{D}_o$

C3 f.a. $\vec{d} \in (\mathcal{D}_o)^j$ and $\omega \in \mathcal{D}_w$, $(\vec{d}, \omega) \in P_\sigma^{M^c}$ iff $\vec{n} \in w[P]$ where $(w, \vec{e'}) = f_w^{-1}(\omega)$ with some $\vec{e'}$ and $n_i = f_o^{-1}(d_i)$ for $i \in \{1, \ldots, j\}$

C4 f.a. $\vec{d} \in (\mathcal{D}_o)^j$ and $\omega \in \mathcal{D}_w$, $F_\sigma^{M^c}(\vec{d}, \omega) = d'$ iff $w[F](\vec{n}) = n'$ where $(w, \vec{e'}) = f_w^{-1}(\omega)$ with some $\vec{e'}$, $n_i = f_o^{-1}(d_i)$ and $n' = f_o^{-1}(d')$

Given $\sigma \in \mathsf{Ag}^+$, the interpretation of predicates E_σ is as follows [2]:

C5 Let $end(\sigma) = i$. For $\omega, \omega' \in \mathcal{D}_w$ s.t. $f_w^{-1}(\omega) = (w, \vec{e})$ and $f_w^{-1}(\omega') = (w', \vec{e'})$, $(\omega, \omega') \in E_\sigma^{M^c}$ iff $e_i = e_i'$ and $(w, \vec{e}_{-i}) \in e_i'$

Theorem 1. *[Goal] Let M^c be constructed as above, consider models $(w, \vec{e})$ and $(w', \vec{e'})$ such that $e_i = e_i'$ and $(w, \vec{e}_{-i}) \in e_i'$, then for any sentence $\alpha \in \mathcal{KL}_n$,*

- *Given μ s.t. $\mu(u) = f_w(w, \vec{e})$, then $w, \vec{e} \models \alpha$ iff $M^c, \mu \models_{FO} \boldsymbol{R}[\alpha, \epsilon, u, v]$*
- *Given μ s.t. $\mu(u) = f_w(w, \vec{e})$, $\mu(v) = f_w(w', \vec{e'})$, then for any $\sigma \in \mathsf{Ag}^*$,*

$$w, \vec{e} \models \alpha \text{ iff } M^c, \mu \models_{FO} \boldsymbol{R}[\alpha, \sigma \cdot i, u, v]$$

[2] In σ, the adjacent letters must be distinct, i.e. $\sigma \notin (\mathsf{Ag}^* \cdot i \cdot i \cdot \mathsf{Ag}^*)$ for any $i \in \mathsf{Ag}$

Lemma 1. *Given $\mathcal{KL}_n$ model $(w, \vec{e})$, let M^c be constructed as above. Considering σ, variables u, v and variable map μ s.t. $\mu(u) = f_w(w, \vec{e})$,*

1. *For any ground terms t_1, t_2,*

$$w, \vec{e} \models t_1 = t_2 \textit{ iff } M^c, \mu \models_{FO} \boldsymbol{R}[t_1 = t_2, \sigma, u, v] \tag{4.1}$$

2. *For any atomic formula $P(t_1, \ldots, t_j)$,*

$$w, \vec{e} \models P(t_1, \ldots, t_j) \textit{ iff } M^c, \mu \models_{FO} \boldsymbol{R}[P(t_1, \ldots, t_j), \sigma, u, v] \tag{4.2}$$

Proof. We show the proof of part 1 by induction, where part 2 can be proved similarly.
Base case: Suppose that $t = F(n_1, \ldots, n_k)$ and $t' = F'\left(n'_1, \ldots, n'_{k'}\right)$, we have

$$\begin{aligned}
& w, \vec{e} \models \left(t = t'\right) \\
\Longleftrightarrow & w\left[F\left(n_1, \ldots, n_k\right)\right] = w\left[F'\left(n'_1, \ldots, n'_{k'}\right)\right] && (\mathcal{KL}_n \text{ semantics}) \\
\Longleftrightarrow & F_\sigma^{M^c}\left(f_o(n_1), \ldots, f_o(n_k), \mu(u)\right) = {F'}_\sigma^{M^c}\left(f_o(n'_1), \ldots, f_o(n'_{k'}), \mu(u)\right) && (\text{C4}) \\
\Longleftrightarrow & M^c, \mu \models F'_\sigma\left(n_1, \ldots, n_k, u\right) = F'_\sigma\left(n'_1, \ldots, n'_{k'}, u\right) && (\text{FOL semantics}) \\
\Longleftrightarrow & M^c, \mu \models \left(t = t'\right) \uparrow_\sigma^u && (\text{Def. } \uparrow_\sigma^u)
\end{aligned}$$

Constants are considered function symbols without arguments. When t or t' is a standard name, the proof is similar but simpler.
Inductive step:

Given $t = F(t_1, \ldots, t_k)$ and $t' = F'(t'_1, \ldots t'_{k'})$, as induction hypotheses:

1 For $i \in \{1, \ldots, k\}$, $w, \vec{e} \models t_i = n_i$ iff $M^c, \mu \models (t_i = n_i) \uparrow_\sigma^u$;

2 For $j \in \{1, \ldots, k'\}$, $w, \vec{e} \models t'_j = n'_j$ iff $M^c, \mu' \models (t'_j = n'_j) \uparrow_\sigma^u$

3 For $\tilde{t} = F(n_1, \ldots, n_k)$ and $\tilde{t'} = F'(n'_1, \ldots, n'_{k'})$,
$w, \vec{e} \models \tilde{t} = \tilde{t'}$ iff $M^c, \mu \models (\tilde{t} = \tilde{t'}) \uparrow_\sigma^u$

Let $\hat{t_i}$ be the term on the LHS in $(t_i = n_i) \uparrow^u_\sigma$ and $\hat{t'_j}$ on the LHS in $(t'_j = n'_j) \uparrow^u_\sigma$,

$$
\begin{aligned}
& w, \vec{e} \models (t = t') \\
\Longleftrightarrow & w[F(t_1, \ldots, t_k)] = w[F'(t'_1, \ldots, t'_{k'})] && (\mathcal{KL}_n \text{ semantics}) \\
\Longleftrightarrow & w[F(n_1, \ldots, n_k)] = w[F(n'_1, \ldots, n'_{k'})] \text{ where} \\
& \quad n_i = w[t_i] \text{ and } n'_j = w[t'_j] \text{ f.a. } i, j && (\text{denotation}) \\
\Longleftrightarrow & w, \vec{e} \models F(n_1, \ldots, n_k) = F(n'_1, \ldots, n'_{k'}) \text{ where} \\
& \quad w, \vec{e'} \models n_i = t_i \text{ and } w, \vec{e'} \models n'_j = t'_j \text{ f.a. } i, j && (\text{semantics}) \\
\Longleftrightarrow & M^c, \mu \models F_\sigma(n_1, \ldots, n_k, u) = F'_\sigma(n'_1, \ldots, n'_{k'}, u) \text{ where} \\
& \quad w, \vec{e'} \models n_i = t_i \text{ and } w, \vec{e'} \models n'_j = t'_j \text{ f.a. } i, j && (\text{I.H. } 3) \\
\Longleftrightarrow & M^c, \mu \models F_\sigma(n_1, \ldots, n_k, u) = F'_\sigma(n'_1, \ldots, n'_{k'}, u) \text{ where} \\
& \quad M^c, \mu \models (t_i = n_i) \uparrow^u_\sigma \text{ and } M^c, \mu \models (t'_j = n'_j) \uparrow^u_\sigma \text{ f.a. } i, j && (\text{I.H. } 1,2) \\
\Longleftrightarrow & M^c, \mu \models F_\sigma(n_1, \ldots, n_k, u) = F'_\sigma(n'_1, \ldots, n'_{k'}, u) \text{ where} \\
& \quad M^c, \mu \models \hat{t_i} = n_i \text{ and } M^c, \mu \models \hat{t'_j} = n'_j \text{ f.a. } i, j && (\text{Def. } \hat{t_i}, \hat{t'_j}) \\
\Longleftrightarrow & M^c, \mu \models F_\sigma(\hat{t_1}, \ldots, \hat{t_k}, u) = F'_\sigma(\hat{t'_1}, \ldots, \hat{t'_{k'}}, u) && (\text{FOL semantics}) \\
\Longleftrightarrow & M^c, \mu \models (F(t_1, \ldots, t_k) = F'(t'_1, \ldots, t'_{k'})) \uparrow^u_\sigma && (\text{Def. } \uparrow^u_\sigma) \\
\Longleftrightarrow & M^c, \mu \models (t = t') \uparrow^u_\sigma \\
\Longleftrightarrow & M^c, \mu \models \boldsymbol{R}[t = t', \sigma, u, v]
\end{aligned}
$$

□

Now we complete the proof of Theorem 1 (our goal):

Proof. Given $(w, \vec{e})$ and $(w', \vec{e'})$ such that $e_i = e'_i$ and $(w, \vec{e}_{-i}) \in e'_i$, base cases with primitive atoms are proved in the previous lemma.
Induction on $\neg, \wedge, \forall$:

- $\alpha \stackrel{def}{=} \neg\beta$

$$
\begin{aligned}
& w, \vec{e} \models \neg\beta \text{ for } \beta \text{ compatible with } (w, \vec{e}) \\
\Longleftrightarrow & w, \vec{e} \not\models \beta \\
\Longleftrightarrow & M^c, \mu \not\models \boldsymbol{R}[\beta, \sigma, u, v] && (\text{I.H.}) \\
\Longleftrightarrow & M^c, \mu \models \neg\boldsymbol{R}[\beta, \sigma, u, v] \\
\Longleftrightarrow & M^c, \mu \models \boldsymbol{R}[\neg\beta, \sigma, u, v] && (\text{Def. } \boldsymbol{R}[\cdot])
\end{aligned}
$$

- $\alpha \stackrel{def}{=} \beta \wedge \gamma$

$$
\begin{aligned}
& w, \vec{e} \models \beta \wedge \gamma \\
\Longleftrightarrow & w, \vec{e} \models \beta \text{ and } w, \vec{e} \models \gamma && \text{(Semantics } \mathcal{KL}_n\text{)} \\
\Longleftrightarrow & M^c, \mu \models \boldsymbol{R}[\beta, \sigma, u, v]) \text{ and } M^c, \mu \models \boldsymbol{R}[\gamma, \sigma, u, v] && \text{(I.H.)} \\
\Longleftrightarrow & M^c, \mu) \models \boldsymbol{R}[\beta, \sigma, u, v] \wedge \boldsymbol{R}[\gamma, \sigma, u, v] \\
\Longleftrightarrow & M^c, \mu) \models \boldsymbol{R}[\beta \wedge \gamma, \sigma, u, u'] && \text{(Def. } \boldsymbol{R}[\cdot]\text{)}
\end{aligned}
$$

- $\alpha \stackrel{def}{=} \forall x.\beta$, where β has no free variable other than x,

$$
\begin{aligned}
& w, \vec{e} \models \forall x.\beta \\
\Longleftrightarrow & w, \vec{e} \models \beta^x_n \text{ f.a. } n \in \mathcal{N} && \text{(Semantics } \mathcal{KL}_n\text{)} \\
\Longleftrightarrow & M^c, \mu \models \boldsymbol{R}[\beta^x_n, \sigma, u, v] \text{ f.a. } n \in \mathcal{N} && \text{(I.H.)} \\
\Longleftrightarrow & M^c, \mu \models \boldsymbol{R}[\beta, \sigma, u, v]^x_n \text{ f.a. } n \in \mathcal{N} \\
\Longleftrightarrow & M^c, \mu' \models \boldsymbol{R}[\beta, \sigma, u, v] \text{ f.a. } n \in \mathcal{N}, \mu' \sim_x \mu, \mu'(x) = f_o(n) \\
\Longleftrightarrow & M^c, \mu' \models \forall x.\boldsymbol{R}[\beta, \sigma, u, u'] && \text{(FOL semantics)} \\
\Longleftrightarrow & M^c, \mu' \models \boldsymbol{R}[\forall x.\beta, \sigma, u, u'] && \text{(Def. } \boldsymbol{R}[\cdot]\text{)}
\end{aligned}
$$

For induction on $\boldsymbol{K}_i$ operator, as induction hypothesis we assume that for any $(w, \vec{e})$ and $(w', \vec{e'})$ such that $e_i = e'_i$ and $(w', \vec{e'}_{-i}) \in e_i$, for μ' such that $\mu'(u') = f_w(w', \vec{e'})$ and $\mu'(u) = f_w(w, \vec{e})$, let $\sigma \in \mathsf{Ag}^*$ and $end(\sigma) \neq i$.

$$
(w', \vec{e'}_{-i}) \cup_i e_i \models \alpha \text{ iff } M^c, \mu' \models \boldsymbol{R}[\alpha, \sigma \cdot i, u', u] \tag{A.5}
$$

Then

$$
\begin{aligned}
& w, \vec{e} \models \boldsymbol{K}_i\alpha \\
\Longleftrightarrow & \text{F.a. } (w', \vec{e'}_{-i}) \in e_i, (w', \vec{e'}_{-i}) \cup_i e_i \models \alpha && (\mathcal{KL}_n \text{ semantics}) \\
\Longleftrightarrow & \text{F.a. } (w', \vec{e'}_{-i}) \in e_i \text{ and } \mu' \text{ s.t. } \mu'(u') = f_w((w', \vec{e'}_{-i}) \cup_i e_i), \\
& \qquad \mu'(u) = f_w(w, \vec{e}), M^c, \mu' \models \boldsymbol{R}[\alpha, \sigma \cdot i, u', u] && \text{(I.H.)} \\
\Longleftrightarrow & \text{F.a. } \mu' \text{ s.t. } \mu'(u) = f_w(w, \vec{e}) \text{ and } M^c, \mu' \models E_{\sigma \cdot i}(u', u), \\
& \qquad M^c, \mu' \models \boldsymbol{R}[\alpha, \sigma \cdot i, u', u] && \text{(cond. C5)} \\
\Longleftrightarrow & M^c, \mu \models \forall u' E_{\sigma \cdot i}(u', u) \supset \boldsymbol{R}[\alpha, \sigma \cdot i, u', u] && \text{(FOL semantics)} \\
\Longleftrightarrow & M^c, \mu \models \boldsymbol{R}[\boldsymbol{K}_i\alpha, \sigma, u, v] && \text{(Def. } \boldsymbol{R}[\cdot]\text{)}
\end{aligned}
$$

Considering the case $\sigma = \epsilon$, then we proved for μ s.t. $\mu(u) = f_w(w, \vec{e})$,

$$
w, \vec{e} \models \boldsymbol{K}_i\alpha \text{ iff } M^c, \mu \models \boldsymbol{R}[\boldsymbol{K}_i\alpha, \epsilon, u, v] \tag{A.6}
$$

Which is the first case in Thm 1. To complete the induction on operator $\boldsymbol{K}_i$ of the second case, it remains to prove that for any $\sigma \in \mathsf{Ag}^*$,

$$w, \vec{e} \models \boldsymbol{K}_i\alpha \text{ iff } M^c, \mu \models_{FO} \boldsymbol{R}[\boldsymbol{K}_i\alpha, \sigma \cdot i, u, v] \tag{A.7}$$

$$\begin{aligned}
& M^c, \mu \models_{FO} \boldsymbol{R}[\boldsymbol{K}_i\alpha, \sigma \cdot i, u, v] \\
\Longleftrightarrow & M^c, \mu \models \forall u'.E_{\sigma\cdot i}(u', v) \supset \boldsymbol{R}[\alpha, \sigma \cdot i, u', v] && \text{(Def. } \boldsymbol{R}[\cdot]) \\
\Longleftrightarrow & M^c, \mu' \models E_{\sigma\cdot i}(u', v) \supset \boldsymbol{R}[\alpha, \sigma \cdot i, u', v] \text{ f.a. } \mu' \sim_{u'} \mu && \text{(FOL semantics)} \\
\Longleftrightarrow & M^c, \mu' \models \boldsymbol{R}[\alpha, \sigma \cdot i, u', v] \text{ f.a. } \mu' \sim_{u'} \mu, \mu'(u') = f_w(w'', \vec{e''}) \text{ s.t.} \\
& \qquad (w'', \vec{e''}_{-i}) \in e_i, e''_i = e_i && \text{(Cond. C5)} \\
\Longleftrightarrow & w'', \vec{e''} \models \alpha \text{ f.a. } (w'', \vec{e''}) \text{ s.t. } (w'', \vec{e''}_{-i}) \in e_i, e''_i = e_i && \text{(I.H. Eq. A.5)} \\
\Longleftrightarrow & w, \vec{e} \models \boldsymbol{K}_i\alpha && (\mathcal{KL}_n \text{ semantics})
\end{aligned}$$

□

A.3.2 Construction of Model in $\mathcal{KL}_n$ ($FOL \Longrightarrow \mathcal{KL}_n$)

In $\mathcal{KL}_n$ the unique name assumption is made regarding the domain of quantification, correspondingly, we need the same assumption when translating a $\mathcal{KL}_n$ formula into FO-formula:

- Σ_{eq} is the following infinite set of formulas:
  - $\forall x(x = x)$
  - $\forall x \forall y.(x = y) \supset (y = x)$
  - $\forall x \forall y \forall z((x = y) \wedge (y = z)) \supset (x = z)$
  - for any function symbol F:
    $\forall x_1 \ldots \forall x_k \forall y_1 \ldots \forall y_k((x_1 = y_1) \wedge \ldots \wedge (x_k = y_k)) \supset F(x_1, \ldots, x_k) = F(y_1, \ldots, y_k)$
  - for any predicate symbol P:
    $\forall x_1 \ldots \forall x_k \forall y_1 \ldots \forall y_k((x_1 = y_1) \wedge \ldots \wedge (x_k = y_k)) \supset P(x_1, \ldots, x_k) \equiv P(y_1, \ldots, y_k)$
- $\Sigma_{una} = \{(n \neq n') \mid n, n' \text{ are distinct standard names in } \mathcal{N}\}$

Intuitively, Σ_{eq} means that the symbol $=$, essentially as a binary predicate, is interpreted in its usual sense. Σ_{una} requires that the domain of discourse must be infinite.
In this direction, we aim to prove:

Theorem 2. *Given $\alpha \in \mathcal{KL}_n$, if $\boldsymbol{R}[\alpha, \varepsilon, u, v] \cup \Sigma_{eq} \cup \Sigma_{una}$ is FO-satisfiable, then α is $\mathcal{KL}_n$-satisfiable.*

The proof of the theorem consists of the following steps: Given $\alpha \in \mathcal{KL}_n$

1. If $\boldsymbol{R}[\alpha, \varepsilon, u, v] \cup \Sigma_{eq} \cup \Sigma_{una}$ is FO-satisfiable, then $\boldsymbol{R}[\alpha, \varepsilon, u, v]$ is satisfiable in $\mathcal{M}^{e,u}$
2. If $\boldsymbol{R}[\alpha, \varepsilon, u, v]$ is satisfiable in $\mathcal{M}^{e,u}$ then $\boldsymbol{R}[\alpha, \varepsilon, u, v]$ is satisfiable in $\mathcal{M}^{e,u,o}$

3. If $\boldsymbol{R}[\alpha, \varepsilon, u, v]$ is satisfiable in $\mathcal{M}^{e,u,o}$ then $M^c, \mu \models \boldsymbol{R}[\alpha, \varepsilon, u, v]$ for some μ

Here $\mathcal{M}^{e,u}$ denotes the set of all models which satisfy $\Sigma_{eq} \cup \Sigma_{una}$, $\mathcal{M}^{e,u,o}$ denotes the set of all models in $\mathcal{M}^{e,u}$ of form $M = \langle(\mathcal{D}_o^c, \mathcal{D}_w), I\rangle$ and for any $d \in \mathcal{D}_o^c$ it exists n s.t. $n^I = d$. Namely, it shares the same domain of object with the canonical model M^c, and every element of sort object is named. When items 1-3 are proven, then with Thm. 1 we prove Thm. 2.

Among the three steps above, the first one is trivial and the second is proved by applying the Löwenheim-Skolem theorem, renaming elements in $\mathcal{D}_o^M$ and define elements in $\mathcal{D}'_w$ such that they preserve the truth values.

Lemma 2. *For $\alpha \in \mathcal{KL}_n$, given $M \in \mathcal{M}^{e,u}$ and an assignment μ s.t. $M, \mu \models \boldsymbol{R}[\alpha, \epsilon, u, v]$, there exists a model $M' \in \mathcal{M}^o$ and an assignment μ' s.t. $M', \mu' \models \boldsymbol{R}[\alpha, \epsilon, u, v]$*

Proof. Supposing that $M, \mu \models \boldsymbol{R}[\alpha, \epsilon, u, v]$ for some $M \in \mathcal{M}^{e,u}$. Since $\mathcal{M}^{e,u}$ satisfies the unique name assumption Σ_{una}, the domain of discourse is infinite. By the Löwenheim-Skolem theorem, any first-order satisfiable theory with an infinite model also has a countable model. Therefore, we can assume that both $\mathcal{D}_w$ and $\mathcal{D}_o$ are countable. Thus a bijection $\pi : \mathcal{D}_o^M \to \mathcal{N}$ can be found. In addition, for finitely many standard names in α (i.e. in $\boldsymbol{R}[\alpha, \epsilon, u, v]$), w.l.o.g. we can require that $\pi(n_i^M) = n_i$.
Now we define $M' = \langle \mathcal{D}_o^c, \mathcal{D}'_w, \Phi\rangle$ as follows: For each $\omega \in \mathcal{D}_w$, we introduce a new element ω' in $\mathcal{D}'_w$ s.t. for each symbols P_σ or F_σ occurs in $\boldsymbol{R}[\alpha, \epsilon, u, v]$,

- $P_\sigma^{M'}(\vec{d'}, \omega') = 1$ iff $P_\sigma^M(\vec{d}, \omega) = 1$ where $d'_i = f_o(\pi(d_i))$
- $F_\sigma^{M'}(\vec{d'}, \omega') = d'_0$ iff $F_\sigma^M(\vec{d}, \omega) = d_0$ where $d'_i = f_o(\pi(d_i))$

Here f_o is the bijection used in defining the canonical model M^c, and ϕ : denote the bijection $\mathcal{D}_w \to \mathcal{D}'_w$. For each E_σ occurs in $\boldsymbol{R}[\alpha, \epsilon, u, v]$, we define the interpretation that f.a. $\omega'_1, \omega'_2 \in \mathcal{D}'_w$, $E_\sigma^{M'}(\omega'_1, \omega'_2) = 1$ iff $E_\sigma^M(\phi^{-1}(\omega'_1), \phi^{-1}(\omega'_2)) = 1$.
Let μ' satisfies $\mu'(u) = \phi(\mu(u))$ and $\mu'(v) = \phi(\mu(v))$, then it is straightforward that $M', \mu' \models \boldsymbol{R}[\alpha, \epsilon, u, v]$ and $\mathcal{M}' \in \mathcal{M}^o$. □

For the third item, the proof is achieved by constructing a set of mappings between $\mathcal{D}_w^M$ and $\mathcal{D}_w^c$ such that it preserves the truth value for formulas up to a certain depth: Given $M \in \mathcal{M}^{e,u,o}$ and $k \in \mathbb{N}$, for any $\sigma \in \mathsf{Ag}^*$ such that $|\sigma| \leq k$, we inductively construct mapping $\phi^{(\sigma)}$ and set S_σ^ω as follows:

1. When $|\sigma| = k$, f.a. $\omega \in \mathcal{D}_w$, we select a state $\omega^c \in \mathcal{D}_w^c$ such that ,

   1.i) $(\vec{d}, \omega) \in P_\sigma^M$ iff $(\vec{d}, \omega^c) \in P_\sigma^{M^c}$ for any P and $\vec{d}$;

   1.ii) $F_\sigma^M(\vec{d}, \omega) = F_\sigma^{M^c}(\vec{d}, \omega^c)$ for any F and $\vec{d}$.

   We use $\phi^{(\sigma)}$ to denote the mapping from ω to the selected state, i.e. $\phi^{(\sigma)}(\omega) = \omega^c$. Given ω, we define the set $S_\sigma^\omega = \{\omega' \mid (\omega', \omega) \in E_\sigma^M\}$. For simplicity, we write $\phi^{(\sigma)}(S_\sigma^\omega)$ to mean $\{\phi^{(\sigma)}(\omega') \mid \omega' \in S_\sigma^\omega\}$.

2. When $|\sigma| < k$, f.a. $\omega \in \mathcal{D}_w$, we select a state $\omega^c \in \mathcal{D}_w^c$ such that

   2.i) $(\vec{d}, \omega) \in P_\sigma^M$ iff $(\vec{d}, \omega^c) \in P_\sigma^{M^c}$ for any P and $\vec{d}$;

2.ii) $F_\sigma^M(\vec{d}, \omega) = F_\sigma^{M^c}(\vec{d}, \omega^c)$ for any F and $\vec{d}$;

2.iii) F.a. $i \neq end(\sigma)$, $(\omega', \omega^c) \in E_{\sigma\cdot i}^{M^c}$ iff $\omega' \in \phi^{(\sigma\cdot i)}(S_{\sigma\cdot i}^{\omega})$

$\phi^{(\sigma)}$ and S_σ^ω are defined identically as the first case.

Note that for the mapping ϕ^σ with $\sigma = \sigma' \cdot j \cdot i$, i.e., j is the second last letter in the word, we always select ω^c such that $(w, \vec{e}) = f_w^{-1}(\omega^c)$, where the depth of e_j is higher by one than any $j' \neq j$, and $(w, \vec{e}_{-j}) \in e_j$.

Apparently, for $|\sigma| < k$, $\phi^{(\sigma)}$ and S_σ^ω are defined based on the definition of $\phi^{(\sigma\cdot j)}$ and $S_{\sigma\cdot j}^\omega$ with $j \neq end(\sigma)$. Since the value assignments in 1.i) - 2.iii) are independent, we can prove that for any model M and finite k, such a set of mappings always exists.

Lemma 3. *Given $M \in \mathcal{M}^{e,u,o}$, let ϕ^σ and S_σ^ω be defined as above, for any sentence α and σ s.t. $dep(\boldsymbol{R}[\alpha, \sigma, u, v,]) \leq k$ (the longest subscript is at most k),*

$$M, \mu \models \boldsymbol{R}[\alpha, \sigma, u, v] \textit{ iff } M^c, \tilde{\mu} \models \boldsymbol{R}[\alpha, \sigma, u, v] \tag{4.3}$$

Where $\tilde{\mu}(u) = \phi^{(\sigma)}(\mu(u))$. When $|\sigma| > 0$, $\tilde{\mu}(v) = \phi^{(\sigma^-)}(\mu(v))$. [3]

Proof. We prove the statement by induction on the structure of formula α.

- Basis: For objective α, by 1.i), 1.ii), 2.i) and 2.ii) we prove that Eq. 4.3 holds f.a. σ and μ
- Induction:
  - Negation: Suppose that the lemma holds for α,

$$\begin{aligned}
& M, \mu \models \boldsymbol{R}[\neg\alpha, \sigma, u, v] \\
\Longleftrightarrow & M, \mu \models \neg\boldsymbol{R}[\alpha, \sigma, u, v] && \text{(Def. } \boldsymbol{R}[\cdot]) \\
\Longleftrightarrow & M, \mu \not\models \boldsymbol{R}[\alpha, \sigma, u, v] && \text{(FOL semantics)} \\
\Longleftrightarrow & M^c, \tilde{\mu} \not\models \boldsymbol{R}[\alpha, \sigma, u, v] && \text{(I.H.)} \\
\Longleftrightarrow & M^c, \tilde{\mu} \models \neg\boldsymbol{R}[\alpha, \sigma, u, v] && \text{(FOL semantics)} \\
\Longleftrightarrow & M^c, \tilde{\mu} \models [\neg\alpha, \sigma, u, v] && \text{(Def. } \boldsymbol{R}[\cdot])
\end{aligned}$$

[3] σ^- means σ without the last letter, e.g. $\sigma = 1 \cdot 3 \cdot 2$, then $\sigma^- = 1 \cdot 3$

– Conjunction: Suppose that the lemma holds for α and β,

$$
\begin{aligned}
& M, \mu \models \boldsymbol{R}[\alpha \wedge \beta, \sigma, u, v] \\
\Longleftrightarrow & M, \mu \models \boldsymbol{R}[\alpha, \sigma, u, v] \wedge \boldsymbol{R}[\beta, \sigma, u, v] && \text{(Def. } \boldsymbol{R}[\cdot]) \\
\Longleftrightarrow & M, \mu \models \boldsymbol{R}[\alpha, \sigma, u, v] \text{ and } M, \mu \models \boldsymbol{R}[\beta, \sigma, u, v] && \text{(FOL semantics)} \\
\Longleftrightarrow & M^c, \tilde{\mu} \models \boldsymbol{R}[\alpha, \sigma, u, v] \text{ and } M^c, \tilde{\mu} \models \boldsymbol{R}[\beta, \sigma, u, v] && \text{(I.H.)} \\
\Longleftrightarrow & M^c, \tilde{\mu} \models \boldsymbol{R}[\alpha, \sigma, u, v] \wedge \boldsymbol{R}[\beta, \sigma, u, v] && \text{(FOL semantics)} \\
\Longleftrightarrow & M^c, \tilde{\mu} \models \boldsymbol{R}[\alpha \wedge \beta, \sigma, u, v] && \text{(Def. } \boldsymbol{R}[\cdot])
\end{aligned}
$$

– Universal quantification of sort object: Suppose that the lemma holds for any α_n^x with $n \in \mathcal{D}_o^c$

$$
\begin{aligned}
& M, \mu \models \boldsymbol{R}[\forall x.\alpha, \sigma, u, v] \\
\Longleftrightarrow & M, \mu \models \forall x.\boldsymbol{R}[\alpha, \sigma, u, v] && \text{(Def. } \boldsymbol{R}[\cdot]) \\
\Longleftrightarrow & M, \mu \models \boldsymbol{R}[\alpha, \sigma, u, v]_n^x \text{ f.a. } n \in \mathcal{N} && \text{(\#)} \\
\Longleftrightarrow & M, \mu \models \boldsymbol{R}[\alpha_n^x, \sigma, u, v] \\
\Longleftrightarrow & M^c, \tilde{\mu} \models \boldsymbol{R}[\alpha_n^x, \sigma, u, v] \text{ f.a. } n \in \mathcal{N} && \text{(I.H.)} \\
\Longleftrightarrow & M^c, \tilde{\mu} \models \boldsymbol{R}[\alpha, \sigma, u, v]_n^x \text{ f.a. } n \in \mathcal{N} \\
\Longleftrightarrow & M^c, \tilde{\mu} \models \forall x.\boldsymbol{R}[\alpha, \sigma, u, v] && \text{(\#)} \\
\Longleftrightarrow & M^c, \tilde{\mu} \models \boldsymbol{R}[\forall x.\alpha, \sigma, u, v]
\end{aligned}
$$

To explain the steps labeled with (#): By the definition of $\mathcal{M}^{e,u,o}$, the sub-domain of sort object is countable and each element is named, therefore the quantification can be interpreted substitutionally.

– For sentence $\boldsymbol{K}_i\alpha$ s.t. $i \neq end(\sigma)$

$$
\begin{aligned}
& M, \mu \models \boldsymbol{R}[\boldsymbol{K}_i\alpha, \sigma, u, v] \\
\Longleftrightarrow & M, \mu \models \forall u'.E_{\sigma \cdot i}(u', u) \supset \boldsymbol{R}[\alpha, \sigma \cdot i, u', u] && \text{(Def. } \boldsymbol{R}[\cdot]) \\
\Longleftrightarrow & M, \mu' \models \boldsymbol{R}[\alpha, \sigma \cdot i, u', u] \text{ f.a. } \mu' \sim_{u'} \mu \text{ s.t. } (\mu'(u'), \mu'(u)) \in E_{\sigma \cdot i}^M && \text{(FOL semantics)} \\
\Longleftrightarrow & M, \mu' \models \boldsymbol{R}[\alpha, \sigma \cdot i, u', u] \text{ f.a. } \mu' \sim_{u'} \mu \text{ s.t. } \mu'(u') \in S_{\sigma \cdot i}^{\mu'(u)} && \text{(Def. } S_\sigma^\omega) \\
\Longleftrightarrow & M^c, \tilde{\mu}' \models \boldsymbol{R}[\alpha, \sigma \cdot i, u', u] \text{ f.a. } \tilde{\mu}' \text{ and } \mu' \sim_{u'} \mu \text{ s.t. } \mu'(u') \in S_{\sigma \cdot i}^{\mu'(u)}, \\
& \qquad \phi^{(\sigma \cdot i)}(\mu'(u')) = \tilde{\mu}'(u') \text{ and } \phi^{(\sigma)}(\mu'(u)) = \tilde{\mu}'(u) && \text{(I.H.)}
\end{aligned}
$$

Since $\mu' \sim_{u'} \mu$, by $\mu'(u') \in S_{(\sigma \cdot i)}^{\mu'(u)}$ and $\phi^{(\sigma \cdot i)}(\mu'(u')) = \tilde{\mu}'(u')$ we derive $(\tilde{\mu}'(u'), \tilde{\mu}'(u)) \in E_{\sigma \cdot i}^{M^c}$. Since $\boldsymbol{R}[\alpha, \sigma \cdot i, u', u]$ has no free variables other than u, u', $\tilde{\mu}'(u) = \phi^{(\sigma)}(\mu'(u)) =$

$\phi^{(\sigma)}(\mu(u)) = \tilde{\mu}(u)$, w.l.o.g. $\tilde{\mu}' \sim_{u'} \tilde{\mu}$.

$$\begin{aligned}
&\Longleftrightarrow M^c, \tilde{\mu}' \models \boldsymbol{R}[\alpha, \sigma \cdot i, u', u] \text{ f.a. } \tilde{\mu}' \sim_{u'} \tilde{\mu} \text{ s.t. } (\tilde{\mu}'(u'), \tilde{\mu}'(u)) \in E^{M^c}_{\sigma \cdot i} \\
&\Longleftrightarrow M^c, \tilde{\mu} \models \boldsymbol{R}[\boldsymbol{K}_i\alpha, \sigma, u, v] && \text{(Def. } \boldsymbol{R}[\cdot])
\end{aligned}$$

– For sentence $\boldsymbol{K}_i\alpha$ s.t. $i = end(\sigma)$

$$\begin{aligned}
&M, \mu \models \boldsymbol{R}[\boldsymbol{K}_i\alpha, \sigma, u, v] \\
\Longleftrightarrow &M, \mu \models \forall u'.E_\sigma(u', v) \supset \boldsymbol{R}[\alpha, \sigma, u', v] && \text{(Def. } \boldsymbol{R}[\cdot]) \\
\Longleftrightarrow &M, \mu' \models \boldsymbol{R}[\alpha, \sigma, u', v] \text{ f.a. } \mu' \sim_{u'} \mu \text{ s.t. } (\mu'(u'), \mu'(v)) \in E^M_\sigma && \text{(FOL semantics)} \\
\Longleftrightarrow &M, \mu' \models \boldsymbol{R}[\alpha, \sigma, u', v] \text{ f.a. } \mu' \sim_{u'} \mu \text{ s.t. } \mu'(u') \in S^{\mu'(v)}_\sigma && \text{(Def. } S^\omega_\sigma) \\
\Longleftrightarrow &M^c, \tilde{\mu}' \models \boldsymbol{R}[\alpha, \sigma, u', v] \text{ f.a. } \tilde{\mu}' \text{ and } \mu' \sim_{u'} \mu \text{ s.t. } \mu'(u') \in S^{\mu'(v)}_\sigma, \\
&\quad \phi^{(\sigma)}(\mu'(u')) = \tilde{\mu}'(u') \text{ and } \phi^{(\sigma^-)}(\mu'(v)) = \tilde{\mu}'(v) && \text{(I.H.)}
\end{aligned}$$

Since $\mu' \sim_{u'} \mu$, by $\mu'(u') \in S^{\mu'(v)}_{\sigma^-}$ and $\phi^{(\sigma)}(\mu'(u')) = \tilde{\mu}'(u')$ we derive $(\tilde{\mu}'(u'), \tilde{\mu}'(v)) \in E^{M^c}_\sigma$. Since $\boldsymbol{R}[\alpha, \sigma, u', v]$ has no free variables other than v, u' and $\tilde{\mu}'(v) = \phi^{(\sigma^-)}(\mu'(v)) = \phi^{(\sigma^-)}(\mu(v)) = \tilde{\mu}(v)$, w.l.o.g. $\tilde{\mu}' \sim_{u'} \tilde{\mu}$.

$$\begin{aligned}
&\Longleftrightarrow M^c, \tilde{\mu}' \models \boldsymbol{R}[\alpha, \sigma, u', v] \text{ f.a. } \tilde{\mu}' \sim_{u'} \tilde{\mu} \text{ s.t. } (\tilde{\mu}'(u'), \tilde{\mu}'(v)) \in E^{M^c}_\sigma \\
&\Longleftrightarrow M^c, \tilde{\mu} \models \boldsymbol{R}[\boldsymbol{K}_i\alpha, \sigma, u, v] && \text{(Def. } \boldsymbol{R}[\cdot])
\end{aligned}$$

□

A.4 Pseudocode $R[\cdot]$ Multi-Agent Case

Algorithm 4 translateStartMAS

Data: formula
Result: result
`/* Starts the recursive function translate */`
String result = *translateMAS*(formula, " ", "u", "v")
return result

Algorithm 5 translateMAS

Data: formula, String sigma, String var1, String var2
Result: Translated FOL Formula

```
/* A counter which ensures fresh variables                          */
int varCounter=0
    /* An array where the result of the partitioning-method is saved
    in                                                               */
String Array formulaArray = partitionMAS(formula)
if formulaArray[0] == ∧:
    return "("translateMAS(formulaArray[1], sigma, var1, var2) " ∧ " translateMAS(formulaArray[2],
     sigma, var1, var2) ")"
elif formulaArray[0] == ∨:
    return "("translateMAS(formulaArray[1], sigma, var1, var2) "∨" translateMAS(formulaArray[2],
     sigma, var1, var2) ")"
elif formulaArray[0] == →:
    return "("translateMAS(formulaArray[1], sigma, var1, var2) "→ " translateMAS(formulaArray[2],
     sigma, var1, var2) ")"
elif formulaArray[0] == ↔:
    return "("translateMAS(formulaArray[1], sigma, var1, var2) "↔" translateMAS(formulaArray[2],
     sigma, var1, var2) ")"
elif formulaArray[0] == =:
    return "("translateMAS(formulaArray[1], sigma, var1, var2) "=" translateMAS(formulaArray[2],
     sigma, var1, var2) ")"
elif formulaArray[0] == ∀:
    return "∀" formulaArray[1]"( O(" formulaArray[1]") →" translateMAS(formulaArray[2], sigma, var1,
     var2 "))"
elif formulaArray[0] == ∃:
    return "∃" formulaArray[1] "(O(" formulaArray[1]") ∧" translateMAS(formulaArray[2], sigma, var1,
     var2 "))"
elif formulaArray[0] == ¬:
    return "¬" translateMAS(formulaArray[1], sigma, var1, var2 ")"
elif formulaArray[0] == K:
    if formulaArray[1] ≠ sigma.lastChar:
        varCounter++
         return "∀ u"varCounter "(E_sigma·i (" var1", u"varCounter") ∧ W(u"varCounter")) →"
         translateMAS(formulaArray[2], sigma · i ,u_varCounter, var1)
    else:
        varCounter++
         return "∀ u"varCounter "(E_sigma (" var2", u"varCounter") ∧ W(u"varCounter")) →" trans-
         lateMAS(formulaArray[2], sigma ,u_varCounter, var1)
/* If none of the conditions above hold, then formula is of form
   P(t_1,...,t_2) or F(t_1,...,t_2)                                 */
else:
    return formulaArray[0]_sigma "("formulaArray[1]"," var1")"
```

Algorithm 6 partitionMAS

Data: formula
Result: Array containing: outermostOperator, Subformula1, (Subformula2)

```
/* Searches for operator at the highest level, returns array with
   the outermost operator in the first position, then the remaining
   formula(s).                                                      */
binaryOperators = [∧, ∨, →, ↔]
 unaryOperators = [K, ∀, ∃, ¬]
 String outermostOperator
 int operatorPosition

/* If first character of formula is "(", then the outermost operator
   is binary                                                        */
if formula[0] == (:
    stack = []
     for (i=1; i< formula.length; i++ ):
        character = fomula[i]  if character == (:
            stack.PUSH(i)
        elif character == ):
            stack.POP()
        elif stack.isEmpty AND character in binaryOperators:
            outermostOperator = character
             operatorPosition = i
             break
    String leftSubformula = formula.substring(1, operatorPositions-1)
     String rightSubformula = formula.substring(operatorPositions+1, formula.length-1)
     return [outermostOperator, leftSubformula, rightSubformula]
/* Outermost operator is unary/only predicate or function is left  */
else:
    outermostOperator = formula[0]
        /* Case where only a function or predicate is left          */
    if outermostOperator not in unaryOperators:
        return[formula[0], formula.substring(2, formula.length-1)]
    elif outermostOperator in [∀, ∃]:
        return[formula[0], formula[1], formula.substring(2, formula.length)]
    elif outermostOperator == ¬:
        return[formula[0], formula.substring(1, formula.length)]
    elif outermostOperator == K:
        return[formula[0], formula[1], formula.substring(2, formula.length)]
    return error
```