# The Complexity of the Validity Problem and Justification Logics

Master's Thesis

Antonis Achilleos
$\mu\prod\lambda\forall$,

Master's Program in Logic and Algorithms

February 12, 2009

**Thank You**

First, I want to thank my advisor, Stathis Zachos, who, simply said, has helped and encouraged me in ways I had never imagined before. Secondly, I thank the other two members of my committee, George Koletsos and Costas Dimitrakopoulos, who had been always very supportive during the time I spent in MPLA. Finally, I thank Sergei Artemov for introducing me to the world of Justification Logic and all it contains. Others must also be thanked for their support and patience, but this will be done in person.

# Contents

# Chapter 1

# Introduction

One of the most well known problems in theoretical computer science is the satisfiability problem for classical propositional logic (SAT). In fact, SAT is the problem that first comes to mind when NP-completeness is discussed. Furthermore, for every logic defined, the question of how easy or hard it is to decide for a formula whether it is satisfiable, or provable for this logic is one of the most compelling and the first to ask. Thus, it is natural to ask this question and try to classify the satisfiability problem for intuitionistic and modal logic, as well as for the much younger justification logic.
Steve Cook was the first to study the complexity of the satisfiability problem, in [10], in 1971, resulting in the well-known Cook's theorem, showing that it is NP-complete, with profound consequences for the field of computational complexity. For the various modal logics, the problem was studied by Ladner in [30] (1977), while Statman in [38] (1979) showed that intuitionistic propositional logic is PSPACE-complete.
The logic of proofs (LP) was first presented by S.Artemov in [3], providing an answer to questions posed for a long time in logic. Since then, it has further developed and evolved into a plethora of justification logics, as with the paradigm of modal logics, a collection of logics, of which the original, LP, is a member. This system of logics has been shown to carry significant properties and its study has led to fresh and naturally ambitious directions. It is thus the obvious next step to consider the provability problems and dual, the satisfiability problem, for justification logic. Kuznets, Krupski and Milnikel have moved towards this end and provided many results that demonstrate in their own way that this system of justification logics is non-trivial in that it behaves in very different ways than known logics, while still being connected in a robust way to the well-known modal logics.
As the field is still young, however, there are still plenty of unanswered questions and open problems concerning the complexity of satisfiability. Further-

more, due to the complexity of the whole structure, these questions had been many, to begin with.
This thesis attempts to present three different topics. The first is a summary of historical results concerning the complexity of the provability (validity) and the satisfiability problem for propositional intuitionistic and modal logics. The second is an overview of the system of logics, now known as justification logic. The focus lies, however, on the presentation of the main results concerning the complexity of the validity problem for justification logic. Not all justification logics are presented here and not all concepts are discussed.

# Chapter 2

# Needed Concepts

## 2.1 Logic

### 2.1.1 Classical and Intuitionistic Propositional Logic

#### Syntax

The syntax of Intuitionistic and Classical Logic is the same. Propositional (or sentence) variables are included, $p, q, r, \ldots$, the bottom symbol $\bot$ and the propositional connectives $\wedge, \vee, \rightarrow, \neg$.
Formally, the set of propositional variables will be $Slet = \{p_1, \ldots, p_n, \ldots\}$.

**Definition 2.1.1.** If $A, B$ are formulas of propositional logic, then so are:

$\bot, p, q, \ldots$, where $p, q, \ldots$ are propositional variables.

$(A \wedge B)$

$(A \vee B)$

$(A \rightarrow B)$

$\neg A$

Many times it is convenient to consider $\neg A$ to be short for $A \rightarrow \bot$, and especially when dealing with intuitionistic logic. On the other hand, while it is customary when dealing with classical logic to include only two symbols to the language (usually $\neg$ and one more, or $\bot$ and $\rightarrow$) and consider the rest of the connectives definable from those, this is not possible in the case of intuitionistic logic.

As usual, parentheses will be omitted when not necessary. $\neg$ will bind more strongly than $\wedge$, which will in turn bind more strongly than $\vee$, and $\vee$ will in turn bind more strongly than $\rightarrow$. $\rightarrow$ will be right-associative.

### Axiomatizations

A Hilbert-style axiomatization of Intuitionistic and Classical propositional Logic follows. This has been taken from Melvin Fitting's notes on classical propositional logic. The axioms are

$A \to B \to A$

$(A \to B \to C) \to (A \to B) \to A \to C$

$A \wedge B \to A$

$A \wedge B \to B$

$A \to B \to A \wedge B$

$A \to A \vee B$

$B \to A \vee B$

$(A \to C) \to (B \to C) \to A \vee B \to C$

$\bot \to A$

$\neg\neg A \to A$ - double negation rule : only in Classical Logic,

where $A, B, C$ are formulas of the language. And one rule, Modus Ponens:

$$\frac{A \to B \qquad A}{B}$$

**Definition 2.1.2.** A proof of formula $A$ in $C$ (in $int$) is a finite sequence of formulas, $A_1, \ldots, A_n$, where for all $i$, $A_i$ is either an axiom of classical (intuitionistic) propositional logic, or occurs by application of modus ponens on two formulas appearing previously in the proof, and $A_n$ is $A$.
We write $C \vdash A$, or $\vdash_C A$ ($int \vdash A$, or $\vdash_{int} A$), if there exists a proof of $A$ in $C$ ($int$).

### Semantics

For classical propositional logic, defining semantics is simple:

**Definition 2.1.3.** A valuation is a function $v : Slet \longrightarrow \{True, False\}$.

**Definition 2.1.4.** A valuation $v$ is said to make a formula $\phi$ of classical propositional logic true ($\phi^v = true$), if

- $\phi = p$, a propositional variable and $v(p) = True$,
- $\phi = \psi_1 \wedge \psi_2$ and $v$ makes both $\psi_1$ and $\psi_2$ true,
- $\phi = \psi_1 \vee \psi_2$ and $v$ makes at least one of $\psi_1$ and $\psi_2$ true,
- $\phi = \psi_1 \rightarrow \psi_2$ and $v$ makes $\psi_2$ true, or $\psi_1$ false, or
- $\phi = \neg\psi$ and $v$ makes $\psi$ false.

For Intuitionistic Logic, a valuation function is not sufficient. The semantics presented here are Kripke semantics, and although they are not the only ones defined for this logic, they are the most convenient for the material of this thesis.

**Definition 2.1.5.** A *Kripke structure* $K = (W, \leq, D)$ *for intuitionistic logic* is a partially ordered set $W$ of states and a function $D$ from states to sets of propositional variables.

$a \leq b \Longrightarrow D(a) \subseteq D(b)$

The truth relation in $K$ is defined:

- $K, a \not\models \bot$
- $K, a \models p$ iff $p \in D(a)$
- $K, a \models A \wedge B \Longleftrightarrow a \models A$ and $a \models B$
- $K, a \models A \vee B \Longleftrightarrow a \models A$ or $a \models B$
- $K, a \models A \rightarrow B \Longleftrightarrow \forall b \geq a \ (b \models A \Rightarrow b \models B)$

**Definition 2.1.6.** A propositional formula is called classically (intuitionistically) satisfiable if there is a valuation (Intuitionistic Kripke model) that makes the formula true (at some state of the model).

**Definition 2.1.7.** A propositional formula is called classically (intuitionistically) valid if there for all valuations (Intuitionistic Kripke models) make the formula true (at all states).

*Observation* 2.1.1. For classical logic, the notions of validity and satisfiability are dual: a formula $\phi$ is satisfiable if and only if its negation is not valid, and vice-versa. This is not true for intuitionistic logic, but it is true for all other logics considered in this thesis.

**Theorem 2.1.1** (Completeness)**.** A formula is classically (intuitionistically) valid if and only if it is classically (intuitionistically) valid.

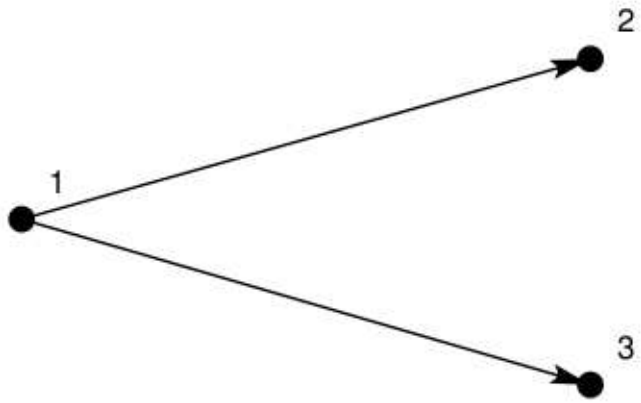


Figure 2.1: The model $K$. $D(1) = D(3) = \emptyset, D(2) = \{p\}$.

**Corollary 2.1.1.** Therefore, the Validity problem is equivalent to the Provability problem. "provable" and "valid" will be used interchangeably.

**Example 2.1.1.** *Consider the Kripke model* $K(W, \leq, D)$*, where* $W = \{1, 2, 3\}$ *and* $\leq = \{(1,1), (2,2), (3,3), (1,2), (1,3)\}$ *and* $D(1) = D(3) = \emptyset, D(2) = \{p\}$*. In* $K$*,* $K, s_2 \models p$*,* $K, 3 \models \neg p$*, but* $K, 1 \not\models p$ *and* $K, 1 \not\models \neg p$*, therefore* $K, 1 \not\models p \vee \neg p$*, thus* $p \vee \neg p$*, a classical tautology is not intuitionistically valid.*

### 2.1.2 Normal Modal Logics

Here, the syntax and semantics of some well-known modal logics will be briefly presented, along with tableau rules for these logics.

#### Syntax

**Definition 2.1.8.** The formulas of modal logic are defined in the following way:

- Propositional variables are modal formulas : $p_1, p_2, p_3, \ldots$.
- If $A, B$ are formulas $\Longrightarrow$ so are $A \wedge B, A \vee B, A \rightarrow B, \bot, \Box A$.
- Nothing else is a formula.

Again, we can think of $\neg A$ as short for $A \rightarrow \bot$. The set of modal formulas is $\mathcal{ML}$

### Axiomatizations

Like in the case of classical and intuitionistic propositional logic, axioms will be provided for Normal Modal Logics. Of course, not all normal modal logics share the same axioms.
Axioms:

**A1** Tautologies of propositional logic

**A2** $\Box\phi \wedge \Box(\phi \rightarrow \psi) \rightarrow \Box\psi$ (**K**)

**A3** $\Box\phi \rightarrow \phi$ (**T**)

**A4** $\Box\phi \rightarrow \Box\Box\phi$ (**4**)

**A5** $\neg\Box\phi \rightarrow \Box\neg\Box\phi$ (**5**)

**A6** $\neg\Box\bot$ (**D**)

**R1** Modus Ponens

**R2** Knowledge Generalization:

$$\frac{\phi}{\Box\phi}$$

**Definition 2.1.9.** The following table defines a number of normal modal logics, according to the axioms it uses.

| Logic | **K** | **T** | **4** | **5** | **D** |
|---|---|---|---|---|---|
| K | ✓ | | ✓ | | |
| D | ✓ | | | | ✓ |
| T | ✓ | ✓ | | | |
| S4 | ✓ | ✓ | ✓ | | |
| S5 | ✓ | ✓ | ✓ | ✓ | |
| KD45 | ✓ | | ✓ | ✓ | ✓ |

In general, considering that $K$ is included in all normal modal logics, and with the exception of $S4, S5$, the name of a modal logic directly implies the axioms it contains.

**Definition 2.1.10.** A Normal Modal Logic is a modal logic that contains at least the propositional tautologies, axiom scheme $K$ and is closed under Modus Ponens and Knowledge Generalization.

*Observation* 2.1.2. Apparently, the smallest (in terms of provable formulas) normal modal logic is $K$.

**Semantics**

Similarly to the semantics of intuitionistic logic, the semantics for modal logic will include Kripke structures. Of course, the semantics for Modal Logic preceded the ones for Intuitionistic logic. In fact, the Kripke models for Intuitionistic logic are just a special case of the ones that will be presented below.

**Definition 2.1.11.**

- A Kripke frame is a pair $(W, R)$, where $R \subseteq W^2$ (accessibility relation)
- A Kripke structure is a triple $\mathcal{M} = (W, R, v)$, where $v : \mathcal{ML} \to 2^W$ For $s \in W$, the truth relation $\models$ is defined between worlds and modal formulas.
  - For any propositional variable $p$, $(\mathcal{M}, s) \models p \iff s \in v(p)$.
  - $(\mathcal{M}, s) \models \phi \wedge \psi \iff (\mathcal{M}, s) \models \phi$ and $(\mathcal{M}, s) \models \psi$.
  - $(\mathcal{M}, s) \models \phi \vee \psi \iff (\mathcal{M}, s) \models \phi$ or $(\mathcal{M}, s) \models \psi$.
  - $(\mathcal{M}, s) \models \phi \to \psi \iff (\mathcal{M}, s) \not\models \phi$ or $(\mathcal{M}, s) \models \psi$.
  - $(\mathcal{M}, s) \models \neg\phi \iff (\mathcal{M}, s) \not\models \phi$
  - $(\mathcal{M}, s) \models \Box\phi \iff$ for all $t$, accessible from $s$, $(\mathcal{M}, t) \models \phi$

**Definition 2.1.12.** Now, the Kripke models for each Modal Logic will be specified, by demanding certain conditions from the accessibility relation of the frames.

- For $K$, no conditions are needed. All Kripke models defined above are models for $K$.
- For $D$, we demand that the accessibility relation is serial (For each world $w$ there is a world $u$ s.t. $R(w, u)$).
- $T$, we demand that the accessibility relation is reflexive(For each world $w$, $R(w, w)$).
- $K4$, we demand that the accessibility relation is transitive. (For every $x, y, w$ worlds, $R(x, y)$ and $R(y, w)$ imply that $R(x, w)$)
- $S4$, we demand that the accessibility relation is reflexive and transitive.
- $S5$, we demand that the accessibility relation is reflexive, symmetric and transitive (it is an equivalence relation).

**Theorem 2.1.2** (Completeness)**.** Each Modal Logic introduced above is sound and complete with respect to its corresponding Kripke models.

### Tableaux Rules for Modal Logic

Tableau Rules will be presented for Modal Logic. These will be particularly useful when dealing with the complexity of Validity for Modal Logic.

**Definition 2.1.13.** A prefixed modal formula is a formula of the form $T\phi$, or $F\phi$, where $\phi$ is a modal formula.

We will also use state prefixes for the formulas in the tableaux. The formulas that will be actually used are of the form $\sigma\ V\phi$, where $\sigma$ is a state prefix, $V \in \{T, F\}$ and $\phi$ is a modal formula.
The following are the tableau rules for the modal logics introduced.

$$\frac{\sigma T\ X \wedge Y}{\begin{array}{c}\sigma T\ X\\ \sigma T\ Y\end{array}} \quad \frac{\sigma F\ X \vee Y}{\begin{array}{c}\sigma F\ X\\ \sigma F\ Y\end{array}} \quad \frac{\sigma F\ X \to Y}{\begin{array}{c}\sigma T\ X\\ \sigma F\ Y\end{array}}$$

$$\frac{\sigma T\ X \vee Y}{\sigma T\ X \mid \sigma T\ Y} \quad \frac{\sigma F\ X \wedge Y}{\sigma F\ X \mid \sigma F\ Y} \quad \frac{\sigma T\ X \to Y}{\sigma F\ X \mid \sigma T\ Y}$$

*Remark* 2.1.1. Up until now, by ignoring the following rules and the state prefixes, we have a sound and complete system for classical propositional logic. There are also tableu systems for intuitionistic logic, but they will not be used in the following.

$$\frac{\sigma F \neg X}{\sigma T\ X} \quad \frac{\sigma T \Diamond X}{\sigma.nTX} \quad \frac{\sigma F \Box X}{\sigma.nFX},$$

for $\sigma.n$ new.

$$\frac{\sigma T \Box X}{\sigma.nTX} \quad \frac{\sigma F \Diamond X}{\sigma.nFX},$$

for $\sigma.n$ that already occurs in the branch.

If the logic contains **T**: $\dfrac{\sigma T\ \Box X}{\sigma T\ X} \quad \dfrac{\sigma F\ \Diamond X}{\sigma F\ X}$

If the logic contains **D**: $\dfrac{\sigma T\ \Box X}{\sigma T\ \Diamond X} \quad \dfrac{\sigma F\ \Diamond X}{\sigma F\ \Box X}$

If the logic contains **4**: $\dfrac{\sigma T\ \Box X}{\sigma.nT\ \Box X} \quad \dfrac{\sigma F\ \Diamond X}{\sigma.nF\ \Diamond X}$

If the logic contains **5**: $\dfrac{\sigma.nT\ \Box X}{\sigma T\ \Box X}$ $\dfrac{\sigma.nF\ \Diamond X}{\sigma F\ \Diamond X}$,

for $\sigma, \sigma.n$ already occurring in the tableau branch.

A rule of the form

$$\frac{A}{\begin{array}{c}B\\C\end{array}}$$

or

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

describes the following action: from $A$, produce formulas $B$, $C$. For the first rule, insert both to the tableau, while for the second, split it to two branches each containing one of the formulas.

**Definition 2.1.14.** A tableau branch is closed if it contains both $\sigma\ T\phi$ and $\sigma\ F\phi$, for some $\sigma$ and $\phi$.

**Definition 2.1.15.** A formula $\phi$ is provable if there is a tableu that has 1 $F\phi$ at its root, and by applying tableau rules, all its branches eventually become closed.

**Theorem 2.1.3.** The above tableau proof systems are sound and complete w.r.t. their corresponding logics.

### 2.1.3 Justification Logics

LP was the first justification logic to appear, in [2]. Since then, justification logic has developed significantly, with a multitude of logics and variations, many of which are presented in the following.

#### Syntax

Here, the common language of the justification logics that this thesis will study will be defined. The language will include justification constants $c_i$, $i \in \mathbb{N}$, justification variables: $x_i$, $i \in \mathbb{N}$, justification terms, usually denoted $t, s, \ldots$ and also called evidence terms, proof terms, proof polynomials, and in some cases in this text and when the meaning is clear from the context, just terms. These are defined:

**Definition 2.1.16** (Justification Terms )**.**

- Constants and variables are terms
- If $t_1, t_2$ are terms, so are

$$(t_1 \cdot t_2),\ (t_1 + t_2),\ (!t_1)$$

$\cdot$ is called application, $+$ is called sum and ! proof checker. The set of justification terms will be called $Tm$.

Also, propositional variables will be used in the language: $p_i,\ i \in \mathbb{N}$. The set of propositional variables will be called $SLet$. The formulas of the language are

**Definition 2.1.17.**

- All propositional variables are formulas
- If $p$ is a propositional variable, $t$ is a term and $F_1, F_2$ are formulas, then so are

$$p,\ \bot,\ (F_1 \to F_2),\ (t : F_1)$$

The language formed like that will be called $\mathcal{JL}$.

$\neg F$ is seen as short for $F \to \bot$, and the rest of the connectives can be defined in the usual way from these. Also as usual, parentheses will be omitted using standard conventions, and naturally, $!s : s : F$ will be read as $(!s : (s : F))$.

### Axiomatizations

Some axioms of justification logic are the following.

**A1** Finitely many schemes of classical propositional logic

**A2** $s : (F \to G) \to (t : F \to s \cdot t : G)$ - Application Axiom

**A3**

$$\begin{array}{ll} s : F \to s + t : F & \\ s : F \to t + s : F & \text{- Monotonicity Axiom} \end{array}$$

**A4** $t : F \to F$ - Factivity Axiom

**A5** $t : F \to !t : t : F$ - Positive introspection

**A6** $t : \bot \to \bot$ - Consistency Axiom
and the rules:

**MP** Modus Ponens Rule :

$$\frac{F \to G \qquad F}{G}$$

**R4**

$$\frac{}{c : A},$$

**R4**$^!$

$$\frac{}{! \cdots !!c : \cdots !!c :!c : c : A},$$

**R4**$_{\mathcal{CS}}$

$$\frac{}{c : A},$$

where $c : A \in \mathcal{CS}$

**R4**$^!_{\mathcal{CS}}$

$$\frac{}{! \cdots !!c : \cdots !!c :!c : c : A},$$

where $c : A \in \mathcal{CS}$,

where in the above, $F$ and $G$ are formulas in $\mathcal{JL}$, $c$ a justification constant, $A$ an axiom of the logic and $\mathcal{CS}$ a constant specification for the logic in question (defined below).

*Remark* 2.1.2. R4$^!_{\mathcal{CS}}$ is admissible in the presence of Positive Introspection rule and R4$_{\mathcal{CS}}$. Similarly, A6 is an instance of A4. Therefore, these will not be used together in the same logic.

**Definition 2.1.18.** A constant specification for a justification logic $JL$ is any set

$$\mathcal{CS} \subset \{c : A \mid c \text{ is a constant, } A \text{ an axiom of } JL\}$$

A c.s. is:

- axiomatically appropriate if each axiom is justified by at least one constant.
- injective if every constant justifies at most one axiom.
- schematic if every constant justifies a certain number of axiom schemes (0 or more).

- schematically injective if it is schematic and every constant justifies at most one scheme
- finite if it is a finite set
- almost schematic if it is the union of a schematic and a finite c.s.

The empty constant specification, $\emptyset$, will also be considered, as well as the total constant specification, $TCS_{JL} = \{c : A \mid c \text{ is a constant, } A \text{ an axiom of } JL\}$.

*Observation* 2.1.3. The total constant specification is schematic and axiomatically appropriate, but not schematically injective.

Below, a table appears that provides a guide for the justification logics this thesis deals with.

| Logic | A1 | A2 | A3 | A4 | A5 | A6 | MP | $\mathrm{R4}_{\mathcal{CS}}$ | $\mathrm{R4}^!_{\mathcal{CS}}$ |
|---|---|---|---|---|---|---|---|---|---|
| J | ✓ | ✓ | ✓ | | | | ✓ | | ✓ |
| JD | ✓ | ✓ | ✓ | | | ✓ | ✓ | | ✓ |
| JT | ✓ | ✓ | ✓ | ✓ | | | ✓ | | ✓ |
| J4 | ✓ | ✓ | ✓ | | ✓ | | ✓ | ✓ | |
| JD4 | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | |
| LP | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | |

**Definition 2.1.19.** The justification logics $J, JD, JT, J4, JD4, LP$ in the language $\mathcal{JL}$ are defined by the axioms and rules indicated by thr above table. The original justification logic, $LP$, originally the Logic of Proofs, according to the uniform notation used for all other logics could also have been named $JT4$. As $LP$ is chronologically the first justification logic defined, it keeps its name to avoid confusion.

*Remark* 2.1.3. Other justification logics have also been defined that are not mentioned, in extended languages. This thesis does not deal with those.

**Definition 2.1.20.** If JL is a justification logic and $\mathcal{CS}$ a constant specification for this logic, then $JL_{\mathcal{CS}}$ is $JL$ with $R4$ (or $R4!$, respectively) replaced by $R4_{\mathcal{CS}}$ ($R4^!_{\mathcal{CS}}$, respectively). $JL_0 = JL_\emptyset$ and $JL = JL_{TCS_{JL}}$.

### Properties of Justification Logics and Definitions

**Definition 2.1.21.** The *forgetful projection* is a function $^\circ : \mathcal{JL} \longrightarrow \mathcal{ML}$ that converts justification formulas into modal formulas. It is defined by induction:

- $p^\circ = p$
- $\bot^\circ = \bot$
- $(F \to G)^\circ = (F^\circ \to G^\circ)$, and finally,
- $(t : F)^\circ = \Box(F^\circ)$,

where $p$ is a propositional variable, $F, G$ are justification formulas and $t$ is a justification term.

**Theorem 2.1.4** (Realization Theorem [2, 9, 37, 6] )**.** The following hold:

1. $J^\circ = K$
2. $JD^\circ = D$
3. $JT^\circ = T$
4. $J4^\circ = K4$
5. $JD4^\circ = D4$
6. $LP^\circ = S4$

*Remark* 2.1.4. From now and on, except when made explicit, whenever it is said that $L$ is a justification logic, it will be meant that $L$ is one of the logics $J, JD, JT, J4, JD4, LP$.

**Corollary 2.1.2.** For any constant specification $\mathcal{CS}$ and justification logic $L$ of those defined above, $L_{\mathcal{CS}}$ is consistent.

**Proof.** It suffices to show that the full $L$ is consistent. But if $L$ is inconsistent, $L \vdash \bot$, which leads to $L^\circ \vdash \bot$. This is a contradiction to the established consistency of the modal logics established above. □

*Lemma* 2.1.1 (Internalization Property [4, 6] ). For any justification logic $L$ and $\mathcal{CS}$ axiomatically appropriate constant specification for $L$, if

$$F_1, \ldots, F_n \vdash_{L_{\mathcal{CS}}} B,$$

then there exists a term $t(x_1, \ldots, x_n)$ for some fresh justification variables $x_1, \ldots, x_n$, such that

$$x_1 : F_1, \ldots, x_n : F_n \vdash_{L_{\mathcal{CS}}} t(x_1, \ldots, x_n) : B$$

*Remark* 2.1.5. The requirement that $\mathcal{CS}$ is axiomatically appropriate cannot be dropped. According to the theorem, all axioms are justified. For this, an axiomatically appropriate $\mathcal{CS}$ is needed.

**Corollary 2.1.3** (Constructive Necessitation)**.** For any justification logic $L$ and $\mathcal{CS}$ axiomatically appropriate constant specification for $L$, if

$$L_{\mathcal{CS}} \vdash B,$$

then there exists a ground term $t$ s.t.

$$L_{\mathcal{CS}} \vdash t : B.$$

*Lemma* 2.1.2 (Lifting Lemma [4, 6]). For any justification logic $L$ with positive introspection $(J4, JD4, LP)$, $\mathcal{CS}$ axiomatically appropriate constant specification for $L$, if

$$F_1, \ldots, F_n,\ q_1 : G_1, \ldots q_m : G_m \vdash_{L_{\mathcal{CS}}} B,$$

for some justification terms $q_1, \ldots, q_m$, then there exists a term

$$t(x_1, \ldots, x_n, y_1, \ldots, y_m)$$

for $x_i, y_i$ fresh variables, s.t.

$$x_1 : F_1, \ldots, x_n : F_n,\ q_1 : G_1, \ldots q_m : G_m \vdash_{L_{\mathcal{CS}}} t(x_1, \ldots, x_n, y_1, \ldots, y_m) : B$$

*Lemma* 2.1.3 (Deduction Theorem [4, 6]). For any justification logic $L$, constant specification $\mathcal{CS}$, if

$$\Gamma, F \vdash_{L_{\mathcal{CS}}} G,$$

then

$$\Gamma \vdash_{L_{\mathcal{CS}}} F \to G.$$

The following property has certain requirements from the constant specification.

*Lemma* 2.1.4 (Substitution Property[4, 6]). For any justification logic $L$ and schematic constant specification for $L$, $\mathcal{CS}$, if

$$\Gamma \vdash_{L_{\mathcal{CS}}} F,$$

then

$$\Gamma[s\backslash x, G\backslash p] \vdash_{L_{\mathcal{CS}}} F[s\backslash x, G\backslash p],$$

where $[s\backslash x, G\backslash p]$ means substituting justification term $s$ for justification variable $x$ and/or formula $G$ for propositional variable $p$.

*Remark* 2.1.6. Again, the requirement for a schematic constant specification cannot be dropped. However, a different version of the property exists, with different requirements of the constant specification.

*Lemma* 2.1.5 (Substitution Property with renaming of constants [16]). For any justification logic $L$ and axiomatically appropriate constant specification for $L$, $\mathcal{CS}$, if

$$\Gamma \vdash_{L_{\mathcal{CS}}} F,$$

then

$$\Gamma[s\backslash x, G\backslash p] \vdash_{L_{\mathcal{CS}}} \bar{F}[s\backslash x, G\backslash p],$$

where $[s\backslash x, G\backslash p]$ means substituting justification term $s$ for justification variable $x$ and/or formula $G$ for propositional variable $p$, and formula $\bar{F}$ is obtained from $F$ by possibly substituting some justification constants for others.

**Definition 2.1.22.** For any justification logic $L$ and constant specification $\mathcal{CS}$, the reflected fragment of $L_{\mathcal{CS}}$ is

$$rL_{\mathcal{CS}} = \{t : F \mid L_{\mathcal{CS}} \vdash t : F\}.$$

**Definition 2.1.23** ($*$-calculi)**.** $*\mathcal{CS}$ Axioms: $*(c : A)$, where $c : A \in \mathcal{CS}$

$*\mathcal{CS}^!$ Axioms: $*(!\cdots!!c :!c : c : A)$, where $t : A \in \mathcal{CS}$

$*A2$

$$\frac{*(s, F \to G) \qquad *(t, F)}{*(s \cdot t, G)},$$

$*A3$

$$\frac{*(t, F)}{*(s + t, t : F)} \quad \frac{*(s, F)}{*(s + t, t : F)}$$

$*A4$

$$\frac{*(t, F)}{*(!t, t : F)},$$

**The calculi:** $*_{\mathcal{CS}}$-calculus includes $*\mathcal{CS}^!$, $A2$, $A3$, and $*!_{\mathcal{CS}}$-calculus includes $*\mathcal{CS}$, $A2$, $A3$ and $A5$

*Remark* 2.1.7. The $*$-calculi provide an independent axiomatization of the reflected fragments of the justification logics.

**Theorem 2.1.5** ([25, 28])**.** For any justification logic $L$ and constant specification $\mathcal{CS}$,

$$L_{\mathcal{CS}} \vdash t : F \Longleftrightarrow rL_{\mathcal{CS}} \vdash t : F \Longleftrightarrow \vdash_{*_{\mathcal{CS}}} *(t, F),$$

if $L$ is one of $J, JD, JT$, and

$$L_{\mathcal{CS}} \vdash t : F \Longleftrightarrow rL_{\mathcal{CS}} \vdash t : F \Longleftrightarrow \vdash_{*!_{\mathcal{CS}}} *(t, F),$$

otherwise.

### Semantics

Semantics for the justification logics that were introduced above will be provided. In this thesis, the semantics that are mostly needed are via M-models, as these are very useful for studying complexity properties due to their compact description. However, there is another way to provide semantics for the justification logics. These are the F-models that provide Kripke-style semantics. These are useful for other reasons. They provide semantics for logics not discussed here, for which there are no known semantics via M-models, they are useful for studying the interaction between the justification logics and the classical modal logics, and even define Hybrid Justification Logics that combine both the justification formalism and the classical modality. However, for the logics studied here, the two approaches are equivalent.

**Definition 2.1.24.** An *M-model for a justification logic* $L_{\mathcal{CS}}$ in language $\mathcal{JL}$, where $\mathcal{CS}$ is a constant specification for $L$ is a pair

$$\mathcal{M} = (V, \mathcal{A}),$$

where propositional valuation

$$V : SLet \longrightarrow \{True, False\}$$

assigns a truth value to each propositional variable and

$$\mathcal{A} : Tm \times \mathcal{JL} \longrightarrow True, False$$

is an *admissible evidence function.* Informally, $\mathcal{A}(t, F)$ specifies whether term $t$ is considered admissible evidence for formula $F$. $\mathcal{A}(t, F)$ will be used as an abbreviation for $\mathcal{A}(t, F) = True$ and $\neg\mathcal{A}(t, F)$ as an abbreviation for $\mathcal{A}(t, F) = False$.
The admissible evidence function must satisfy certain closure conditions that depend on the axioms and rules of $L_{\mathcal{CS}}$:

**Application Closure:** If $\mathcal{A}(s, F \to G)$ and $\mathcal{A}(t, F)$ then $\mathcal{A}(s \cdot t, G)$.

**Sum Closure:** If $\mathcal{A}(s, F)$ then $\mathcal{A}(s + t, F)$.
If $\mathcal{A}(t, F)$ then $\mathcal{A}(s + t, F)$.

**$\mathcal{CS}$ Closure:** If $c : A \in \mathcal{CS}$, then $\mathcal{A}(c, A)$ and

$$\mathcal{A}(\underbrace{!!\cdots!}_{n}c, \underbrace{!!\cdots!}_{n-1}c : \cdots : !c : c : A), \text{ for any } n \geq 1.$$

**Positive Introspection Closure (only if A5 is an axiom of $L$)** If $\mathcal{A}(t, F)$ then $\mathcal{A}(!t, t : F)$

**Consistent Evidence Condition (only if A6 is an axiom of $L$)** $\mathcal{A}(t, \bot) = False$,

For any formulas $F, G$, any terms $s, t$, any $c : A \in \mathcal{CS}$, and any integer $n \geq 1$. The truth relation $\mathcal{M} \models H$ is defined as follows:

- $\mathcal{M} \models p$ iff $V(p) = True$
- $\mathcal{M} \not\models \bot$
- $\mathcal{M} \models F \to G$ iff $\mathcal{M} \not\models F$ or $\mathcal{M} \models G$
- $\mathcal{M} \models t : F$ iff $\mathcal{M} \models F$ and $\mathcal{A}(t : F)$ (if A4 is an axiom of $L$) -definition 1
- $\mathcal{M} \models t : F$ iff $\mathcal{A}(t : F)$ (if A4 is not an axiom of $L$) -definition 2

for any formulas $F, G$, any term $t$ and any propositional variable $p$.

**Definition 2.1.25** (Simplified $\mathcal{CS}$ Closure)**.** Another condition is the following: If $c : A \in \mathcal{CS}$, then $\mathcal{A}(c, S)$.

**Proposition 2.1.1.** Let $\mathcal{A} : Tm \times \mathcal{JL} \longrightarrow True, False$ satisfy both the Positive Introspection Closure and the Simplified $\mathcal{CS}$ Closure. Then $\mathcal{A}$ also satisfies the full $\mathcal{CS}$ Closure condition.

The following table summarizes which closure conditions and which definition of truth for formulas $t : F$ should be used for various justification logics. The $\mathcal{CS}$ Closure condition is replaced by its simplified version whenever possible.

| Logic | Appl. Closure | Sum Closure | $\mathcal{CS}$ Closure | Simpl.$\mathcal{CS}$ Closure | Pos.Intr. Closure | Cons.Evid. Condition | def. 1 | def.2 |
|---|---|---|---|---|---|---|---|---|
| J | ✓ | ✓ | ✓ | | | | | ✓ |
| JD | ✓ | ✓ | ✓ | | | ✓ | | ✓ |
| JT | ✓ | ✓ | ✓ | | | | ✓ | |
| J4 | ✓ | ✓ | | ✓ | ✓ | | | ✓ |
| JD4 | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ |
| LP | ✓ | ✓ | | ✓ | ✓ | | ✓ | |

**Theorem 2.1.6** (Completeness Theorem for M-models [34, 26])**.** Each justification logic from $J_{\mathcal{CS}}$, $JD_{\mathcal{CS}}$, $JT_{\mathcal{CS}}$, $J4_{\mathcal{CS}}$, $JD4_{\mathcal{CS}}$, $LP_{\mathcal{CS}}$, where $\mathcal{CS}$ is a constant specification for that logic is sound and complete with respect to its M-models.

In what follows, possible evidence functions are introduced and some of their properties are presented.

**Definition 2.1.26.** An M-type possible evidence function is any function

$$\mathcal{B} : Tm \times Fm \longrightarrow \{True, False\}.$$

A possible evidence function is essentially an admissible evidence function with no conditions imposed on it.

**Definition 2.1.27.** We say that an M-type possible evidence function $\mathcal{B}_2$ is based on an M-type possible evidence function $\mathcal{B}_1$ and write

$$\mathcal{B}_1 \subseteq \mathcal{B}_2,$$

if for all terms $t$ and formulas $F$,

$$\mathcal{B}_1(t, F) \Longrightarrow \mathcal{B}_2(t, F).$$

**Definition 2.1.28.** Let $\mathcal{EF}$ be a class of M-type possible evidence functions. A possible evidence function $\mathcal{B} \in \mathcal{EF}$ is called the minimal evidence function in $\mathcal{EF}$ if for all $\mathcal{B}' \in \mathcal{EF}$,

$$\mathcal{B}' \subseteq \mathcal{B}.$$

**Proposition 2.1.2.** If the minimal evidence function in a class exists, it is unique.

**Theorem 2.1.7** ([34])**.** For any $L$ justification logic of the ones considered here, $\mathcal{CS}$ constant specification for $L$ and any possible evidence function $B$, the class of M-type admissible evidence functions for $L_{\mathcal{CS}}$ based on $B$ is symbolized $\mathcal{AEF}_{\mathcal{B}}(L_{\mathcal{CS}})$, is nonempty and has a unique minimal element.

## 2.2 Computational Complexity

Here, the concepts from Complexity Theory needed for later will be presented. The model of computation assumed is the Turing Machine, although others could be easily used with no, or with minimal effect.

**Definition 2.2.1.** $TIME(t(n))$ (or $DTIME(t(n))$): problems that can be solved by a *deterministic* TM in time $t(n)$.

**Definition 2.2.2.** $NTIME(t(n))$: problems that can be solved by a *non deterministic* TM in time $t(n)$.

**Definition 2.2.3.** $SPACE(s(n))$ (or $DSPACE(s(n))$): problems that can be solved by a *deterministic* TM by using additional working space $s(n)$.

**Definition 2.2.4.** $NSPACE(s(n))$: problems that can be solved by a *non deterministic* TM by using additional working space $s(n)$.

Based on the above, we define:

- $P = PTIME = \bigcup_{i\geq 1} DTIME(n^i)$
- $NP = NPTIME = \bigcup_{i\geq 1} NTIME(n^i)$
- $PSPACE = \bigcup_{i\geq 1} DSPACE(n^i)$
- $NPSPACE = \bigcup_{i\geq 1} NPSPACE(n^i)$
- $L = DSPACE(\log n)$
- $NL = NSPACE(\log n)$
- $DEXP = \bigcup_{i\geq 1} DTIME(2^{n^i})$
- $DEXPSPACE = \bigcup_{i\geq 1} DSPACE(2^{n^i})$
- $f$ is *constructible*: there is a TM which, for all inputs $x$ with $|x| = n$, accepts the input in time $O(n + f(n))$ (time- constructible) or working space $O(f(n))$ (space- constructible).

**Proposition 2.2.1.** If *If $f$ is constructible, then*:

- $DSPACE(f(n)) \subseteq NSPACE(f(n))$
- $DTIME(f(n)) \subseteq NTIME(f(n))$

- $DTIME(f(n)) \subseteq DSPACE(f(n))$
- $NTIME(f(n)) \subseteq DSPACE(f(n))$
- If $f(n) > \log n$, then
  - $NTIME(f(n)) \subseteq DTIME(c^{f(n)})$
  - $DSPACE(f(n)) \subseteq DTIME(c^{f(n)})$
- $NSPACE(f(n)) \subseteq DTIME(k^{\log n + f(n)})$.

**Theorem 2.2.1** (Savitch's Theorem)**.** If $f(n) \geq \log n$, then

$$NSPACE(f(n)) \subseteq DSPACE(f^2(n))$$

An immediate consequence: $PSPACE = NPSPACE$.

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE$$

We know that $L \neq PSPACEandNL \neq PSPACE$.
The rest of the relationships between these classes are open problems.

**Definition 2.2.5.** $FP$ is the class of functions that can be computed by a deterministic TM in polynomial time.

**Definition 2.2.6.** $FL$ is the class of functions that can be computed by a deterministic TM in logarithmic space.

**Definition 2.2.7. Complement of a language**

$$\overline{L} = \{x | x \notin L\}$$

**Complement class:**

$$coC = \{\overline{L} | L \in C\}$$

For example, $\overline{SAT} \in coNP$, and $P$ is closed under complement.

**Theorem 2.2.2** (Immerman- Szelepcsényi)**.** $NSPACE(s(n))$ is closed under complement.

**Definition 2.2.8** (Karp Reduction)**.** We say that a language $A$ is polynomially (or Karp) reducible to $B$,

$$A \leq_m^P B \text{ iff } \exists f \in FP, \forall x (x \in A \iff f(x) \in B)$$

**Definition 2.2.9** (Log-space reduction)**.** We say that a language $A$ is Log-space reducible to $B$,

$$A \leq_m^L B \text{ iff } \exists f \in FL, \forall x(x \in A \iff f(x) \in B)$$

We have $A \leq_m^L B \implies A \leq_m^P B$, but not the converse.

**Definition 2.2.10.** We say that class C is closed under reduction $\leq$ if

$$A \leq B \wedge B \in C \implies A \in C.$$

**Definition 2.2.11** (Hardness)**.** A is C-hard, under *leq*, if:

$$\forall B \in C : B \leq A$$

**Definition 2.2.12** (Completeness)**.** A is $C$- complete, under $\leq$, if:

$$\text{A is } C \text{ - hard under } \leq \wedge A \in C \ .$$

**Definition 2.2.13.** The problem SAT (satisfiability) is defined:

**Given:** A boolean formula in CNF.

**Question:** Is there an assignment that satisfies the formula ?

**Theorem 2.2.3.** (Cook).The SAT problem in NP- complete.

Computation with an oracle: An algorithm uses an oracle for problem $\Pi$, if the algorithm can pose (during the computation) a query to the oracle for some instance $\chi$ of problem $\Pi$, whether $\chi \in \Pi$, and the oracle answers 'yes' or 'no'. The algorithm does not spend any additional resources when it asks a question to the oracle (it gets the correct answer for 'free' even though problem $\Pi$ can be very hard).

**Definition 2.2.14.**

- $\mathcal{C}^{\Pi}$ : class of problems solvable by an algorithm in class $\mathcal{C}$ that uses an oracle for problem $\Pi$
- $$\mathcal{C}^{\mathcal{C}_l} = \bigcup_{\Pi \in \mathcal{C}_o} \mathcal{C}^{\Pi}$$

E.g.: $P^{SAT}$: Problems solvable by a deterministic polynomial algorithm using an oracle for the SAT problem. Another equivalent description: $P^{NP}$ (because SAT is NP- complete).

**Definition 2.2.15.** $(k \geq 0)$

- ${\Sigma^p}_0 = \Pi^p_0 = \Delta^p_0 = P$
- $\Sigma^p_{k+1} = NP^{\Sigma 6 p_k}, \Pi^p_{k+1} = co\Sigma^p_{k+1}, \Delta^p_{k+1} = P^{\Sigma^p_k}, \Delta\Sigma^p_k = \Sigma^p_k \cup \Pi^p_k$
- Polynomial hierarchy: $PH = \cap_{k\in\mathbb{N}}\Sigma^p_k$

$\Sigma^p_1 = NP$, $\Pi^p_1 = coNP$ and $\forall k \geq 0 : \Sigma^p_k \subseteq \Sigma^p_{k+1}$ and $\Pi^p_k \subseteq \Sigma^p_{k+1}$.

Although there is no proof of the strictness of the above inclusions (like in the arithmetic hierarchy), it is believed that the polynomial hierarchy is strict. If PH is not strict, then $\exists k : PH = \Sigma^p_k$, i.e., the $PH$ collapses at the $k$-th level.

A different way to define $PH$: using quantifier alteration ($\exists and \forall$). In all cases, quantifiers quantify over strings whose size is bounded by a polynomial $p$ w.r.t input size.

**Proposition 2.2.2.** $L \in \Sigma^p_k$ iff $\exists$ predicate $R$ computable in polynomial time and a polynomial $p$ that bounds the quantified variables, such that:

$$x \in L \Longleftrightarrow \exists y_1 \forall y_2 \ldots Q_k y_k R(x, y_1, y_2, \ldots, y_k),$$

where

$$Q_k = \begin{cases} \exists \text{ , if k is odd} \\ \forall \text{, if k is even} \end{cases}$$

Similarly for $\Pi^p_k$, but the quantifiers start from $\forall$:

**Proposition 2.2.3.** $L \in \Pi^p_k iff \exists$ predicated R computable in polynomial time and a polynomial p that bounds the quantified variables, such that:

$$x \in L \Longleftrightarrow \forall y_1 \exists y_2 \ldots Q y_k R(x, y_1, y2, \ldots, y_k),$$

where

$$Q_k = \begin{cases} \forall, \text{ if k is odd} \\ \exists, \text{ if k is even} \end{cases}$$

**Definition 2.2.16.** An *alternating Turing machine* is a nondeterministic Turing machine in which the set of states is partitioned into two sets, $K_{AND}, K_{OR}$. (The corresponding configurations are usually refered to as type $\wedge$ and type $\vee$ nodes in the computation tree)
Let $x$ be an input to the machine. The set of eventually accepting configurations of the machine is defined as follows:

- All halting configurations are eventually accepting iff they are accepting.
- A configuration with its state in $K_{AND}$ is eventually accepting iff all its permissible succesor configurations are accepting, and
- A configuration with its state in $K_{OR}$ is eventually accepting iff any of its permissible succesor configurations is accepting.

The TM accepts the input $x$ iff the initial configuration is an eventually accepting configuration.
An alteration in the computation of the machine is a transition, during the computation, from a state in in $K_{AND}$ to a state in in $K_{OR}$, or vice-versa.

**Definition 2.2.17.** Similarly to the definitions of $DTIME(f(n))$, $NTIME(f(n))$, $P$, $NP$, $DSPACE(f(n))$, $NSPACE(f(n))$, $L$, $NL$, $PSPACE$, and $NPSPACE$, the classes $ATIME(f(n))$, $AP$, $ASPACE(f(n))$, $AL$ and $APSPACE$ are defined.

**Proposition 2.2.4.**
- $AL = P$
- $AP = PSPACE$

When describing an alternating algorithm later on, it might be said that a choice is universal, or existential, meaning that the current state when the choice is made is in $K_{AND}$, or in $K_{OR}$, respectively.

It can be shown that the polynomial hierarchy is exactly the class of languages accepted by TMs with a bounded number of alternations.

- $L \in \Sigma_k^p$ iff $L$ is acceptable by a TM with at most $k$ alternations, starting with type $\vee$ (at the root);
- $L \in \Pi_k^p$ iff $L$ is acceptable by a TM with at most k alterations, starting with type $\wedge$.

Finally, the following problems will be used later in reductions:

**Definition 2.2.18.** The problem $\forall\exists$3SAT is defined:

**Input:** A formula of the form $\forall\vec{p}\,\exists\vec{q}\,\phi(\vec{p},\vec{q})$, where $\phi$ is a propositional formula in $3CNF$ and $\vec{p},\vec{q}$ are short for $p_1,\ldots,p_n,q_1,\ldots,q_m$, respectively.

**Question:** Is it true that $\forall\vec{p}\,\exists\vec{q}\,\phi(\vec{p},\vec{q})$?

This problem is known to be $\Pi_2^p$-complete.

**Definition 2.2.19.** The problem QBF is defined:

**Input:** A closed formula of the form $\phi = Q_1 p_1 \cdots Q_k p_k$, where $\psi$ is a propositional formula in $3CNF$ and $p_1, \ldots, p_k$, are propositional letters.

**Question:** Is it true that $\phi$?

This problem is known to be $PSPACE$-complete.

# Chapter 3

# The Complexity of the Validity Problem

In this chapter, the complexity of the Validity problem is discussed, for Intuitionistic, Modal and Justification Logic. Of course, the problem was first studied for the case of Classical Propositional Logic, by S.Cook, in [10].

## 3.1 Intuitionistic Logic and Validity

The Complexity of the Validity Problem for the case of Intuitionistic Logic was first studied by Statman in [38] and was proven to be $PSPACE-$*complete*. An alternative proof was given by Svejdar in [39].

**Theorem 3.1.1.** [38] The problem of determining if an arbitrary formula is intuitionistically valid is $PSPACE-hard$.

**Proof.** [39] The proof is by reduction from $QBF$.
Let $A = Q_m p_m \cdots Q_1 p_1 B$ be a $QBF$ formula, where $B$ contains no quantifiers and $Q_i \in \{\exists, \forall\}$. An effort will be made to keep connectives others than $\rightarrow$ and $\wedge$ out of the formula that will be formed. For this, new variables, namely $q_1, \ldots, q_m$ and $s_1, \ldots, s_m$. The formulas $A_i^*$ will be defined recursively.
Let $A_0^*$ be $Bp_1, \ldots, p_m$. If $i > 0$ and $Q_i$ is $\exists$, then $A_i^*$ is

$$(A_{i-1}^* \rightarrow q_i) \wedge ((p_i \rightarrow q_i) \rightarrow s_i) \wedge ((\neg p_i \rightarrow q_i) \rightarrow s_i) \rightarrow s_j$$

and if $i > 0$ and $Q_i$ is $\forall$, then $A_i^*$ is

$$(A_{i-1}^* \rightarrow q_i) \wedge ((p_i \rightarrow q_i) \wedge (\neg p_i \rightarrow q_i) \rightarrow q_i) \rightarrow q_j$$

Let $A^*$ be $A_m^*$. By induction on $i$, it will be shown that for $0 \leq i \leq m$, and any $e$, evaluation of the variables $p_{i+1}, \ldots, p_m$, $e \nvDash Q_i p_i \cdots Q_1 p_1 B(p_1, \ldots, p_m)$

$\iff A_j^*$ has a counterexample in which each variable among $p_{i+1}, \ldots, p_m$ is evaluated according to $e$ in all states of the structure.
So, let $e$ be an evaluation of $p_{i+1} \cdots p_m$.

**For $i = 0$ (base case)** If $e$ makes $B$ false, then the structure with one state that evaluates $p_1, \ldots, p_m$ according to $e$ is a counterexample to $A_0^* = B$. On the other hand, if $K$ is a counterexample for $A_0^*$, in which each $p_j$ is evaluated according to $e$ in all states, then it is straightforward to see that in all states, $v$, $e(p_j) = true \implies v \models p_j$ and $e(p_j) = false \implies v \models \neg p_j$. Thus, immediately, $e \not\models B$.

**For $i > 0$** Let $Q_i = \exists$, and $e \not\models \exists p_i Q_{i-1} p_{i-1} \cdots Q_1 p_1 B$. Also, let $e_t$ be an evaluation of $p_{i-1}, \ldots, p_m$ s.t. $e_t(p_i) = true$ and for all $j > i$, $e_t(p_j) = e(p_j)$, and let $e_f$ be an evaluation of $p_{i-1}, \ldots, p_m$ s.t. $e_f(p_i) = false$ and for all $j > i$, $e_f(p_j) = e(p_j)$. Neither of $e_t$ and $e_f$ satisfy $Q_{i-1} p_{i-1} \cdots Q_1 p_1 B$. Thus, by the induction hypothesis, there exist counterexamples, $K_0, K_1$, for $A_{i-1}^*$, s.t. $p_{i-1}, \ldots, p_m$ are evaluated in all states according to $e_f$ and $e_t$, respectively. Let $a_0, a_1$ be the respective roots of these structures. Let $K$ be the structure constructed by combining $K_0$ and $K_1$, together with new state $a$, s.t. $a \leq a_0, a_1 \not\leq a$, $\leq$ on $K_0, K_1$ and $\leq$ behaves in the obvious manner for all other pairs. Now, the truth values of the variables, that have not yet been defined, will be given for the states of $K$.

- In $a$, $p_{i+1}, \ldots, p_m$ are assigned truth values according to $e$.
- $p_1, \ldots, p_i, q_1, \ldots, q_{i-1}, s_1, \ldots, s_{i-1}$ are all negative in $a$.
- $q_j$ has everywhere the same truth value as $A_{i-1}^*$
- $s_i$ has everywhere the same truth value as $(p_i \to q_i) \vee (\neg p_i \to q_i)$

Now, it can be checked that $K$ is a counterexample for $A_i^*$.
If $Q_i = \exists$, and $K$ is a counterexample for $A_i^*$, in which each $p_j$, $j > i$ is evaluated according to $e$ in all states, then in $K$ there must be a state $a$ s.t.

$$a \models (A_{i-1}^* \to q_i) \wedge ((p_i \to q_i) \to s_i) \wedge ((\neg p_i \to q_i) \to s_i)$$

and $a \not\models s_j$. So, $a \models (\neg p_i \to q_i) \to s_i$, therefore $a \not\models p_i \to q_i$. From that, there exists a $a_0 \geq a$ in $K$ s.t. $a_0 \models \neg p_i$, $a_0 \not\models q_i$. And because $a \leq a_0$, $a_0 \models A_{i-1}^* \to q_i$. In conclusion, $a_0 \not\models A_{i-1}^*$. The substructure generated by $a_0$ is a counterexample to $A_{i-1}^*$, where $p_i$ is everywhere negative. Similarly, by taking $a_1 \models p_i$, we have a counterexample for $A_{i-1}^*$, where

$p_i$ is everywhere true. Let $e_t, e_f$ be defined as above. By the induction hypothesis, $e_t \not\models Q_{i-1}p_{i-1}\cdots Q_1p_1B$ and $e_f \not\models Q_{i-1}p_{i-1}\cdots Q_1p_1B$. Therefore, $e \not\models \exists p_i Q_{i-1}p_{i-1}\cdots Q_1p_1B$.
The case for $Q_i = \forall$ is similar.

□

*Remark* 3.1.1. Later, for the case of various modal logics, a reduction is given to the satisfiability problem for these logics, thus establishing the $PSPACE$-completeness for it. Then, we conclude that since $PSPACE = coPSPACE$, then the provability problem is also $PSPACE$-complete. This cannot be done here, as the set of intuitionistically provable and the set of intuitionistically satisfiable formulas are not complements of each other, as is the case for all other logics considered in this thesis. Moreover, the set of intuitionistically satisfiable formulas is the same as the set of classically satisfiable propositional formulas, which is known to be $NP$-complete. Thus, the satisfiability problem for intuitionistic logic is widely believed not to be $PSPACE$-hard.

*Observation* 3.1.1. We can also reach a conclusion about the implicational fragment of intuitionistic logic, that is the set of formulas in intuitionistic logic that contain only the symbol $\rightarrow$. Remember that $\neg A$ is short for $A \rightarrow \bot$. That is the reason that an effort has been made to avoid connectives other than $\rightarrow$ and $\wedge$ in the construction of the proof above. To show hardness of implicational formulas, the following changes can be made to the construction. First, replace $B$ with a classically equivalent implicational formula. Then, after the construction, use the fact that $A \wedge B \wedge C \wedge \cdots \wedge D \rightarrow E$ is intuitionistically equivalent to $A \rightarrow (B \rightarrow (C \rightarrow (\cdots \rightarrow (D \rightarrow E)\cdots)))$. Notice that conjunctions appear only in this context. In fact, we can even avoid the symbol $\bot$, but this will not be covered here.

So far, all at has been proven is the $PSPACE-hardness$ of intuitionistic logic. Proof of $PSPACE-completeness$ also needs a $PSPACE$ algorithm deciding whether a given formula is intuitionistically valid. This will be accomplished by giving a translation $t$ of every propositional formula $A$ into $S4$, s.t.

$$Int \vdash A \Longleftrightarrow S4 \vdash t(A)$$

and a $PSPACE$ algorithm for the validity problem for $S4$. This translation can be described simply: "Prefix all subformulas of $A$ with $\Box$".

**Proposition 3.1.1.** $Int \vdash A \Longleftrightarrow S4 \vdash t(A)$

**Theorem 3.1.2.** The problem of determining if an arbitrary formula is intuitionistically valid is $PSPACE-complete$.

The theorem is proven by combining 3.1.1, the above proposition and theorem 3.2.2.

## 3.2 Modal Logic and Validity

*Observation* 3.2.1. It is immediately apparent that the satisfiability problem for all modal logics mentioned here is $NP-hard$, as any propositional formula is classically satisfiable iff it is satisfiable for any modal logic and, furthermore, the satisfiability problem for classical propositional logic is known to be $NP-complete$. Therefore, to establish $NP-completeness$ for the satisfiability problem on a modal logic, all that is needed is to show that the problem is in $NP$. Also, according to the previous section, to establish $PSPACE-completeness$ for the satisfiability problem for $S4$, all that is required is to show that the problem is in $PSPACE$.

**Proposition 3.2.1.** Given a structure $M$, and a (modal) formula $\phi$, there is an algorithm for checking if $\phi$ is satisfied in $M$, that runs in time $O(\|M\| \cdot \phi)$.

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

**Proposition 3.2.2.** An $S5$ (or $KD45$) formula $\phi$ is satisfiable iff it is satisfiable in a structure in $\mathcal{M}^{rst}$ ( in $\mathcal{M}^{elt}$) with at most $|\phi|$ states.

**Proof.** First, the proof for $S5$.
All that is needed to show is that if $\phi$ is satisfiable in a structure $M$, then it is satisfiable in a structure $M'$, with at most $|\phi|$ states.
Suppose $(M, s) \models \phi$. We can assume, w.l.o.g. that $M = (S, \pi, \mathcal{R})$, where $\mathcal{R} = S^2$. Otherwise, we can consider $M$ to be the induced substructure on the equivalence class (defined by the relation $\mathcal{R}$) that includes $s$. Let $F$ be the set of subformulas of $\phi$ that have the form $\Box\psi$ that are false at $s$. For each $\Box\psi \in F$, let $s_\psi$ be one of the states in $S$, s.t. $(M, s_\psi) \models \neg\psi$. Let $M' = (S', \pi', \mathcal{R}')$ be the induced substructure of $M$ on $S' = \{s\} \cup \{s_\psi \mid K\psi \in F\}$. Since $|F| < |Sub(\phi)| \leq |\phi|$, $|S'| \leq |\phi|$. It will be shown by induction on the structure of $\psi$, for any $\psi$, subformula of $\phi$, and any $s' \in S'$, $(M, s') \models \psi \iff (M', s') \models \psi$.
The cases of propositional variables or connectives, are trivial. If the formula is $\Box\psi$, then

$$(M, s') \models \Box\psi \Rightarrow$$

$$(M, s'') \models \psi, \text{ for all } s'' \in S \Rightarrow$$

$$(M, s'') \models \psi, \text{ for all } s'' \in S' \Rightarrow$$

$$(M', s'') \models \psi, \text{ for all } s'' \in S' \text{ (Ind.Hyp.)}$$

$$\Rightarrow (M, s') \models \Box\psi$$

Conversely,

$$(M, s') \models \neg\Box\psi \Rightarrow$$

$$(M, s'') \models \neg\psi, \text{ for some } s'' \in S \Rightarrow$$

$$(M, s) \models \neg\Box\psi, \text{ as } (s, s'') \in \mathcal{R} \Rightarrow$$

$$\Box\psi \in F \Rightarrow$$

$$\exists s_\psi \in S' \ (M, s_\psi) \models \neg\psi \Rightarrow$$

$$(M', s_p si) \models \neg\psi \text{ (Ind.Hyp.)} \ \Rightarrow$$

$$(M, s') \models \neg\Box\psi.$$

This completes the proof for $S5$.
For $KD45$, the proof is similar, with the following changes. We don't assume $\mathcal{R} = S^2$, but instead, that for all $s' \in S$, that $(s, s') \in \mathcal{R}$. It can be easily checked that this doesn't affect the satisfiability of the formula. Also, $F$ will be the set of all subformulas of the form $\Box\psi$, and $s_\psi$ will be a state in $S$ s.t. $(M, s_\psi) \models \psi \Leftrightarrow (M, s) \models \Box\psi$. This is to make $R'$ serial. If $F$ is empty, though, then $M'$ is a single state reflexive structure, and $\phi$ is satisfied in that, because it is propositional, and we are done. Finally, keeping in mind that for any $s', s'' \neq s$, $(s', s'') \in \mathcal{R}$, as $R$ is euclidean, we can complete the proof. □

From the above, the following corollary follows immediately. To decide whether a formula $\phi$ is satisfiable, all we have to do is guess nondeterministically a structure $M$ of size at most $|\phi|$ and then check in time $O(|\phi|^2)$ if $\phi$ is satisfied in $M$.

**Corollary 3.2.1.** The satisfiability problem for $S5$ and $KD45$ is $NP-complete$. Therefore, the validity problem for these logics is $coNP-complete$.

*Remark* 3.2.1. The above results cannot be generalized for $K, T$ and $S4$. Indeed, an infinite set of counterexamples of the previous proposition can be constructed for any of these logics.

**Theorem 3.2.1.** The satisfiability problem for $K, T$ and $S4$ is $PSPACE-hard$.

**Proof.** The proof is by reducing the problem $QBF$ (quantified boolean formula) to the satisfiability problem for these logics. One way to check if a $QBF$ formula is true or not is the following. If the formula has no quantifiers, i.e. all occurrences of variables are replaced by truth values, just evaluate its truth value. If the formula is of the form $\exists pB$, replace it with $B_1 \vee B_2$, and if it is of the form $\forall pB$, with $B_1 \wedge B_2$, where $B_1$ is $B$, with $p$ replaced by $True$, and $B_2$ is $B$, with $p$ replaced by $False$. Then, check if the new formula is true. The correctness of this procedure is immediate and follows from the definition of the problem.
To construct the reduction from $QBF$ to $S4$, a formula that will be formed, that will describe the above procedure. Suppose the given formula, $A$, is $Q_1 p_1 \cdots Q_m p_m B$, where $B$ contains no quantifiers and $Q_i \in \{\exists, \forall\}$. A formula $\phi_A^{S4}$. The following propositional variables will be used: $p_1, \ldots, p_m, d_0, \ldots, d_{m+1}$. First, the following formulas are defined.

$$depth = \bigwedge_{i=1}^{m+1} (d_i \rightarrow d_{i-1})$$

$$determined = \bigwedge_{i=1}^{m} (d_i \rightarrow ((p_i \rightarrow K(d_i \rightarrow p_i)) \wedge (\neg p_i \rightarrow \Box(d_i \rightarrow \neg p_i))))$$

$$branching =$$

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

$$\wedge$$

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

and finally,

$$\phi_A^{S4} = d_0 \wedge \neg d_1 \wedge \Box(depth \wedge determined \wedge branching \wedge (d_m \rightarrow B)).$$

The role of the formula *depth* is to make the truth values of the variables $d_1, \ldots, d_{m+1}$ characterize the depth of the decision tree for the procedure. *determined* expresses the fact that once a truth value is assigned to a variable in the decision tree, that value does not change when we go deeper in the tree. Finally, *branching* describes if the current node of the decision tree is universal, or existential, and how the tree continues. The size of $\phi_A^{S4}$ is linear w.r.t. the size of $A$, and it is easily constructed from $A$. As mentioned before, $\phi_A^{S4}$ mimics the previous procedure for determining the truth of $QBF$ formulas, on an $\mathcal{M}^{rt}$ structure. Therefore it is easy to see that $A$ is true if

and only if $\phi_A^{S4}$ is satisfiable by a $\mathcal{M}^{rt}$ structure.
Respectively, the formulas constructed for $T$ and $K$ are $\phi_A^T$ and $\phi_A^K$. They are the same as $\phi_A^{S4}$, except that to deal with the fact that structures for $T$ are not transitive, and structures for $K$ are not even reflexive,

$$\phi_A^T = d_0 \wedge \neg d_1 \wedge \Box^m(depth \wedge determined \wedge branching \wedge (d_m \rightarrow B)),$$

and

$$\phi_A^K = d_0 \wedge \neg d_1 \wedge \bigwedge_{i=0}^{m} \Box^i(depth \wedge determined \wedge branching \wedge (d_m \rightarrow B)).$$

□

Below, a $PSPACE$ procedure is given, that decides whether a ($K, T$, or $S4$) formula is satisfiable.

**Theorem 3.2.2.** The satisfiability problem for $K, T$ and $S4$ is in $PSPACE$.

**Proof.** An alternating algorithm that runs in polynomial time will be given below. This proves that the satisfiability problem for these logics is in $APTIME = PSPACE$.
The algorithm constructs a prefixed tableau branch for the given formula, $\phi$. At each step, a tableau rule is applied to add one or two new formulas to the constructed branch, which we will call $B$. At first, $B$ only contains $1.\phi$, which is not marked. $B$ is a set, i.e. it contains each element at most once. The algorithm also uses a variable, called $w$. At each step, the algorithm does the following.

1. Pick an unmarked formula, $\psi$ in $B$ and mark it. This is a universal choice.

2. If a propositional rule is applicable for the formula, apply it. If this produces two new formulas in the same branch, add them to $B$. If the rule branches, existentially choose one of the formulas it produces, add it to $B$. If now two formulas of the form $\sigma.\psi'$ and $\sigma\neg\psi'$ are contained in $B$, reject.

3. If the formula is of the form $w.\Diamond X$, add to $B$ $(w+1)X$ and $w := w+1$. If the formula is of the form $w.\neg\Box X$, add to $B$ $(w+1)\neg X$ and $w := w+1$. If now two formulas of the form $\sigma\psi'$ and $\sigma\neg\psi'$ are contained in $B$, reject.

4. If $\psi$ is of the form $\sigma.\Box X$, let $\psi'$ be $X$, and if it is of the form $\sigma\neg\Diamond X$, let $\psi'$ be $\neg X$. Also, let $\psi$ be $\sigma.F$. In these cases, do:
If $\sigma \geq w$, accept. Otherwise,

- For $K$, add $(\sigma+1).\psi'$ to $B$.
- For $T$, add $(\sigma+1).\psi'$ and $\sigma.\psi$ to $B$.
- For $S4$, add $(\sigma+1).\psi'$, $\sigma.\psi$ and $(\sigma+1)F$ to $B$.

If now two formulas of the form $\sigma'\psi'$ and $\sigma'\neg\psi'$ are contained in $B$, reject.

Since to increase the value of $w$, the complexity of the formula used in the rule decreases, $w \leq |\phi|$. The number of different prefixed formulas with at most the same complexity as $\phi$ is therefore at most $|\phi|^2$. Every step of the algorithm applies a rule to a different unmarked prefixed formula and marks it. So, the algorithm runs in at most $|\phi|^2$ steps. Each step takes at most $O(|\phi|^2)$ time, so the algorithm uses polynomial time.
The correctness of the algorithm is immediate, as it simply follows the tableau construction rules. □

**Corollary 3.2.2.** The satisfiability problem for $K, T$ and $S4$ is $PSPACE-$*complete*. Therefore, the validity problem for these logics is $PSPACE$-complete.

## 3.3 Justification Logic and Validity

Now, the case of Justification Logic is considered. Upper and lower bounds will be provided for the complexity of the validity problem for justification logics that have been defined here. The following theorem provides information about the case of the reflected fragments of justification logics.

**Theorem 3.3.1** ([23, 28])**.** Let $\mathcal{CS}$ be a schematic constant specification decidable in polynomial time. Then,

1. There exists a non-deterministic algorithm that runs in polynomial time and determines, given a finite set $S$ of $*$-expressions, a formula $F$ and a term $t$, whether
$$S \vdash_{*\mathcal{CS}} *(t, F)$$
2. There exists a non-deterministic algorithm that runs in polynomial time and determines, given a finite set $S$ of $*$-expressions, a formula $F$ and a term $t$, whether
$$S \vdash_{*!\mathcal{CS}} *(t, F)$$

**Proof.** Two nondeterministic algorithms will be provided, $*_{\mathcal{CS}}$-DERIVE and $*!_{\mathcal{CS}}$-DERIVE for the respective calculi. Variables $P, Q, R, \ldots$ will be used over formulas. $X, Y, \ldots$ will denote schemes. An axiom scheme can be written as a formula in this extended language. $F \in X$ will denote that formula $F$ is an instance of the scheme $X$. Also, the empty scheme will be considered, $\emptyset$.

*procedure $*_{\mathcal{CS}}$-DERIVE(S,*(t,F)) and procedure $*!_{\mathcal{CS}}$-DERIVE(S,*(t,F)):*

**1.** For each occurrence of a subterm $s$ in $t$, where $*(s, G) \in S$ for some $G$, choose, nondeterministically, to mark it with the symbol '$S$' or not. Make sure that if $s$ was thus marked, then no proper suboccurrence of this occurrence is assigned anything, or marked with anything. Furthermore, all such proper suboccurrences will not be taken into account for the following.

**2.** For each occurrence of operator + (outside terms marked with $S$), nondeterministically mark it with 'l' or 'r'.

**3.** For each occurrence of $r = \underbrace{!!!\cdots!}_{n} c$ in $t$, for a constant $c$ and an integer $n$, nondeterministically choose an axiom scheme $X$. If $c : X \subseteq \mathcal{CS}$, make the assignment:

$$\underbrace{!!\cdots!}_{n} c \leftarrow \underbrace{!\cdots!}_{n-1} c : \cdots : !c : c : X, \text{ for } n \geq 1, \text{ or}$$

$$c \leftarrow X, \text{ for } n = 0.$$

If $c : X \not\subseteq \mathcal{CS}$, then reject the input.
For $*!_{\mathcal{CS}}$-DERIVE(S,*(t,F)), this step is:

**3!.** For each occurrence of a constant $c$ in $t$, nondeterministically choose an axiom scheme $X$. If $c : X \subseteq \mathcal{CS}$, make the assignment:

$$c \leftarrow X.$$

If $c : X \not\subseteq \mathcal{CS}$, then reject the input.

**4.** For each occurrence of a justification variable $x$, assign:

$$x \leftarrow \emptyset.$$

**5.** For each occurrence of $s$ marked with $S$ in step 1, nondeterministically choose a formula $G$ s.t. $*(s, G) \in S$. and make the following assignment:

$$s \leftarrow G.$$

Repeat steps $6-8$ until an assignment is made to $t$.

**6.** Nondeterministically, choose an occurrence of a subterm $s_1+s_2$ s.t. $s_1, s_2$ have been assigned with $X_1, X_2$, respectively, but $s_1+s_2$ has not been assigned yet (if such a subterm exists). If this appearance of $+$ has been marked with $l$, let $i=1$. If it was marked with $r$, then let $i=2$. Assign:

$$s_1+s_2 \leftarrow X_i.$$

**7.** Nondeterministically, choose an occurrence of a subterm $!s$ s.t. $s$ has been assigned with $X$, but $!s$ has not been assigned yet (if such a subterm exists). Assign

$$!s \leftarrow \emptyset.$$

For $*!_{\mathcal{CS}}$-DERIVE(S,*(t,F)), this step is:

**7!.** Nondeterministically, choose an occurrence of a subterm $!s$ s.t. $s$ has been assigned with $X$, but $!s$ has not been assigned yet (if such a subterm exists). If $X \neq \emptyset$, then assign

$$!s \leftarrow s : X;$$

else, assign

$$!s \leftarrow \emptyset.$$

**8.** Nondeterministically, choose an occurrence of a subterm $s_1 \cdot s_2$ s.t. $s_1, s_2$ have been assigned with $Z_1, X_2$, respectively, but $s_1 \cdot s_2$ has not been assigned yet (if such a subterm exists). If $Z_1 = X_1 \rightarrow Y_1$ and $X_2$ is unifiable with $X_1$, then assign

$$s_1 \cdot s_2 \leftarrow Y^*,$$

where $Y^*$ is the form $Y$ takes after the unification of $X_1$ and $X_2$. If $Z_1 = P$ and $X_2 \neq \emptyset$, then assign

$$s_1 \cdot s_2 \leftarrow Q,$$

where $Q$ is fresh. Otherwise, assign

$$s_1 \cdot s_2 \leftarrow \emptyset.$$

**9.** Let $X$ be the scheme assigned to $t$. If $X$ is unifiable with $F$, then accept. Otherwise, reject.

To complete the proof, two conditions must be shown to be satisfied. First, the above algorithms must run in polynomial time. But this is easily seen, as no step can be executed more times than the number of occurrences of subterms of $t$ in $t$. Secondly, it must be shown that the algorithms accept if and only if $S \vdash_{*\mathcal{CS}} *(t, F)$ ($S \vdash_{*!\mathcal{CS}} *(t, F)$). For the only if part, it can be established that $s \leftarrow X \implies$ for all $G \in X$, $S \vdash_* *(s, G)$. This can be proven by induction on the assignments made by the algorithm. For the opposite direction, notice that in a $*$-calculus derivation, each rule application increases the complexity of the terms and conserves these terms as distinct occurrences. Term $t$ is constructed by applying rules of the respective $*$-calculus. Therefore, we can associate each subterm appearance with a formula that was associated with that occurrence of the subterm during its construction by a rule of the $*$-calculus. This is what the algorithm attempts to accomplish. Therefore, if $(t, F)$ is indeed derivable, there exists a series of choices of the algorithm that lead to an accepting configuration. □

**Corollary 3.3.1.** There exists a deterministic algorithm that runs in polynomial time and determines, given a finite set $S$ of $*$-expressions, a formula $G$ and a term $t$, whether

$$S \vdash_{*!\emptyset} *(t, G)$$

**Proof.** The algorithm for this proof is similar to the algorithm of the previous proof. Only, it assigns formulas to terms in a different way (deterministically). Call $S_s$ the set of formulas $F$ s.t. $s : F \in S$. By recursion on the structure of the term $s$, assign to an occurrence of it a set of formulas, which will be called $SF(s)$:

- If $s$ is a constant, then $SF(s) := S_s$. Notice that $|SF(s)| \leq |S_s|$.
- If $s = s_1 + s_2$ then $SF(s) := S_s \cup SF(s_1) \cup SF(s_2)$. Notice that $|SF(s)| \leq |S_s| + |SF(s_1)| + |SF(s_2)|$.
- If $s = s_1 \cdot s_2$ then $SF(s) := S_s \cup \{F \mid \exists H \in SF(s_2)$ s.t. $H \to F \in SF(s_1)\}$. Notice that $|SF(s)| \leq |S_s| + |SF(s_1)|$.
- If $s =!s_1$ then $SF(s) := S_s \cup \{s_1 : F \mid F \in SF(s_1)\}$. Notice that $|SF(s)| \leq |S_s| + |SF(s_1)|$.

If $G \in SF(t)$, then accept. Otherwise, reject.
The above algorithm is correct with similar, yet simpler reasoning as the one used in the previous proof. For the complexity of the algorithm, all that is needed is to establish that each term is assigned with a set that contains few enough formulas. Let $For(n) = \max_{|s| \leq n} |FS(s)|$. Then,

$$For(n) \leq |S| + For(n - m) + For(m), \text{ for some } m < n$$

and

$$For(1) \leq |S|.$$

Therefore, $For(n) \leq |S|^2$.

**Corollary 3.3.2.** Let $\mathcal{CS}$ be an almost schematic constant specification decidable in polynomial time. Then,

1. There exists a non-deterministic algorithm that runs in polynomial time and determines, given a finite set $S$ of $*$-expressions, a formula $F$ and a term $t$, whether
$$S \vdash_{*\mathcal{CS}} *(t, F)$$
2. There exists a non-deterministic algorithm that runs in polynomial time and determines, given a finite set $S$ of $*$-expressions, a formula $F$ and a term $t$, whether
$$S \vdash_{*!\mathcal{CS}} *(t, F)$$

**Proof.** The proof for this is simple. Since $\mathcal{CS}$ is almost schematic, it is the union of a finite $X$ and a schematic $\mathcal{CS}'$. To decide for a given $S$ and $*(t : F)$ if $S \vdash_{*\mathcal{CS}} *(t, F)$, simply run the nondeterministic polynomial-time algorithm from above for determining if $S \cup X \vdash_{*\mathcal{CS}} *(t, F)$. Since $X$'s size is constant, the running time of the algorithm remains a polynomial of the size of the input. □

**Corollary 3.3.3.** Let $\mathcal{CS}$ be a finite, or injective constant specification decidable in polynomial time. Then, there exists a deterministic algorithm that runs in polynomial time and determines, given a finite set $S$ of $*$-expressions, a formula $F$ and a term $t$, whether

$$S \vdash_{*\mathcal{CS}} *(t, F)$$

In the following, by "decidable", it will be meant "decidable in polynomial time".

**Theorem 3.3.2** ([26])**.** $J_{\mathcal{CS}}, JT_{\mathcal{CS}}, J4_{\mathcal{CS}}, LP_{\mathcal{CS}}$ with a decidable almost schematic $\mathcal{CS}$ are in $\Pi_2^p$.

**Proof.** The algorithm for deciding these logics consists of two parts. Let $\Phi$ be the given formula. The first part constructs a tableau branch, starting from $F\ \Phi$, and applying all propositional rules possible. In addition, the following rules are also used. The first two are for logics $J_{\mathcal{CS}}$ and $J4_{\mathcal{CS}}$. The following two are used for $JT_{\mathcal{CS}}$ and $LP_{\mathcal{CS}}$.

$$\frac{T\ s : G}{T\ *(s : G)} \qquad \frac{F\ s : G}{F\ *(s : G)}$$

$$\frac{T\ s:G}{\begin{array}{c}T\ G\\ T\ *(s:G)\end{array}} \qquad \frac{F\ s:G}{F\ *(s:G)\mid F\ G}$$

This part of the algorithm is similar to the algorithm used to prove theorem 3.2.2. The $*$-expressions are not analyzed further for the moment. If no more rules can be applied, the branch is called completed, and if it contains both $T\ G$ and $F\ G$, it is called closed. Every application of a rule reduces the complexity of the formulas, and, furthermore, the sum of the lengths of the resulting formulas is at most as much as the length of the initial formula. Therefore, a closed, or a completed tableau branch is reached in time linear with respect to the length of the initial formula. All choices made during this part are universal.
The second stage of the algorithm when a completed, or closed branch is formed. If the branch is closed, accept. If not, then decide in nondeterministic polynomial time if for some $*$-expression produced by the first part, of the form $F\ *(t:R)$, $X \vdash_{*\mathcal{CS}} *(t,R)$, where $X$ is the set of all positively prefixed $*$-expressions in the branch. If the answer is "yes", then accept. Otherwise, reject.
From the above description, the following lemma should follow clearly.

*Lemma* 3.3.1. The algorithm above runs in polynomial time, is a $coNP$ algorithm and uses an $NP$ oracle.

Therefore, to show that the problems in question are in $\Pi_2^p$, all that is needed is the following lemma:

*Lemma* 3.3.2. A formula $G$ is derivable in $J_{\mathcal{CS}}, JT_{\mathcal{CS}}, J4_{\mathcal{CS}}$, or $LP_{\mathcal{CS}}$, if and only if the tableau that is produced from $F\ G$ has in every not closed branch a $*$-expression of the form $F\ *(t:R)$ s.t. $*(t:R)$ is derivable from $X$ by the corresponding $*$-calculus.

**Proof.** First, suppose that $G$ is not derivable. Then, by the Completeness Theorem, there exists an $M$-model $\mathcal{M}=(V,A)$ s.t. $\mathcal{M}\models\neg G$. A not closed, completed branch will be constructed that contains only $*$-expressions of the form $F\ *(t:G)$ s.t. $*(t:G)$ is not derivable from $X$ by the $*$-calculus considered. More specifically, all formulas that will appear in the branch will be satisfied by the model and for all $*$-expressions $T/F\ *(t:G)$ will be satisfied by $\mathcal{A}$, i.e. $\mathcal{A}(t,G)$ will hold iff we are in the case of $T\ *(t:G)$. This branch will be constructed as follows:

- Of course, $F\ \Phi$ is satisfied in the model.

- For any propositional rule, at least one of the possible choices that can be made produces formulas satisfied in the model. Otherwise, the hypothesis of the rule would not be satisfied in the first place.

- For the new rules introduced here, for

$$\frac{T\ s:G}{T\ *(s:G)} \qquad \frac{F\ s:G}{F\ *(s:G)},$$

the formula $s:G$ is satisfied in the model iff $\mathcal{A}(t,G)$ holds. On the other hand, for

$$\frac{T\ s:G}{\begin{array}{c}T\ G\\ T\ *(s:G)\end{array}} \qquad \frac{F\ s:G}{F\ *(s:G)\mid F\ G},$$

from $T\ s:G$ the formula $T\ G$ is produced. This formula must be satisfied in the model, by the definition of $\models$, otherwise, neither would $T\ s:G$. The same holds for $\mathcal{A}(s,G)$. From $F\ s:G$, either the formula $F\ G$ must be satisfied by the model, or $\mathcal{A}(t,G)$ must not hold.

From the above, the model satisfies all elements of the branch. Therefore, the branch cannot be closed. Furthermore, for no negatively prefixed $*$-expression $F\ *(s,G)$, it is the case that $X \vdash_{*\mathcal{CS}} *(s,G)$, where $X$ is as defined above. If this was true, then since $\mathcal{A}(t,R)$ holds for all $*(t,R) \in X$, then it should be that also $\mathcal{A}(s,G)$ holds. But this cannot be, because $F\ *(s,G)$ is in the branch.
For the opposite direction, given an open branch, where for no expression $F\ *(t:G)$, $X \vdash_{*\mathcal{CS}} *(t,G)$, a model can be constructed, that satisfies all elements of the branch. Simply, take $\mathcal{M} \models p$ iff $p$ is in the branch, and define $B$ to be the set of all positively prefixed expressions in the branch. From this, define $\mathcal{A}$ to be the minimal evidence function based on $B$. It can be seen from the properties of the $*$-calculi and the evidence functions that $\mathcal{M}$ is indeed a model. And it satisfies all formulas of the branch (by induction on the structure of the formula), therefore also $\neg G$. The only nontrivial case is of formulas of the form $F\ t:G$. Since the algorithm accepts, $B \nvdash_{*\mathcal{CS}} *(t,G)$, therefore, since $\mathcal{A}$ is the minimal evidence function on $B$, $\mathcal{A}(t,G)$ does not hold, so $\mathcal{M} \models \neg t:G$. □

This completes the proof of the theorem. □

**Corollary 3.3.4.** $LP_{\mathcal{CS}}$ with a finite or decidable injective $\mathcal{CS}$ is in $coNP$.

**Proof.** The proof is the same as above, only now deciding if $X \vdash_{*!\mathcal{CS}} t : G$ is in $P$. And, of course, there is no need to check nondeterministically a formula $F\ t : G$ in the branch, but instead this can be done deterministically, checking all such formulas, that are only polynomially many. □

**Theorem 3.3.3** ([28, 29])**.** $JD_{\mathcal{CS}}$ with a decidable, almost schematic and axiomatically appropriate $\mathcal{CS}$ is in $\Pi_2^p$.

**Proof.** A somewhat different approach is needed for the case of $JD_{\mathcal{CS}}$. The fact that $\mathcal{CS}$ is axiomatically appropriate is needed for the proof. The difference in this case is that admissible evidence functions for $JD_{\mathcal{CS}}$ models must satisfy the Consistent Evidence function condition. That is, for no admissible evidence function $\mathcal{A}$ and term $t$, should it be true that $\mathcal{A}(t : \bot)$. To deal with this, the algorithm is modified in the following ways:

- The formulas in the tableau are also prefixed with an integer. The formulas are of the form $n\ V\ G$, where $n$ is a natural number, $V \in \{T, F\}$, $G$ is a formula of the logic. Instead of the tableau rules introduced above, the following ones are used:

$$\frac{n\ T\ s : G}{\begin{array}{c} n\ T\ *(s : G) \\ n+1\ T\ G \end{array}} \qquad \frac{n\ F\ s : G}{n\ F\ *(s : G)}$$

  The integer prefixes are not used to denote possible worlds in a model, as is the usual case with $S5$, but different M-models. They are to ensure that the Consistent Evidence function condition is maintained: from $t_1 : F_1,\ \ldots, t_k : F_k$ we cannot infer $t : \bot$.

- Instead of having $X$ be the set of positive $*$ expressions in the branch, $X_n$ are the sets of positive $*$ expressions in the branch that are prefixed with $n$. Then, for the algorithm to accept, it must be the case that for some $n$, $F\ *(t, G)$ that $X_n \vdash_{*\mathcal{CS}} *(t, G)$.

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

*Lemma* 3.3.3. A formula $G$ is derivable in $JD_{\mathcal{CS}}$, if and only if the tableau that is produced from $1\ F\ G$ has in every not closed branch a $*$-expression of the form $n\ F\ *(t : R)$ s.t. $*(t : R)$ is derivable from $X_n$ by the corresponding $*$-calculus.

**Proof.** The proof is similar to the proof of the previous lemma and proceeds by proving that a formula $G$ is not derivable in $JD_{\mathcal{CS}}$, if and only if the tableau that is produced from 1 $F$ $G$ has a not closed branch with all $*$-expressions of the form $n$ $F$ $*(t : R)$ satisfy that $*(t : R)$ is not derivable from $X_n$ by the corresponding $*$-calculus.. What is needed is to make sure that the Consistent Evidence function condition is satisfied by the admissible evidence function of the model considered when a model is constructed and that when a branch is constructed, then for each $n$, there exists a model satisfying all $n$-prefixed elements of the branch. Therefore, instead of a single model, a sequence of models will be considered, $\mathcal{M}_1, \ldots, \mathcal{M}_N$, where $N$ is sufficiently large. The proof proceeds by recursively constructing the models, starting from the given $\mathcal{M}_1 = \mathcal{M} \models \neg G$, thus ensuring for all $n$ that all $n$-prefixed element introduced in the branch will be satisfied in $\mathcal{M}_n$. And similarly, for the other direction of the proof, the induction is on $N - n$, showing that a model exists that satisfies all $n$-prefixed elements. □

The proof of the theorem is thus complete. □

A trivial lower bound holds for all logics considered:

**Proposition 3.3.1.** For any $\mathcal{CS}$, any of $J_{\mathcal{CS}}$, $JD_{\mathcal{CS}}$, $JT_{\mathcal{CS}}$, $J4_{\mathcal{CS}}$, $JD4_{\mathcal{CS}}$, $LP_{\mathcal{CS}}$, $J5_{\mathcal{CS}}$, $J45_{\mathcal{CS}}$, $JD45_{\mathcal{CS}}$, $JT45_{\mathcal{CS}}$ is $coNP$-hard.

This holds because any propositional formula is valid for classical propositional logic iff it is valid for any of these logics.
However, other less trivial lower bounds are known.

**Theorem 3.3.4** ([33])**.**
1. $J4_{\mathcal{CS}}$ with a decidable schematic $\mathcal{CS}$ is $\Pi_2^p$-hard.
2. $LP_{\mathcal{CS}}$ with a decidable schematically injective axiomatically appropriate $\mathcal{CS}$ is $\Pi_2^p$-hard.

**Proof.** The proof for both parts of the theorem will be by reducing the problem $\forall\exists 3SAT$ and requires the following lemma. In the following, if no logic is specified, it is meant any of the two logics considered in this theorem.

*Lemma* 3.3.4. Let $\psi(p_1, \ldots, p_n)$ be a propositional formula in $3CNF$, built up from propositional variables $p_1, \ldots, p_n$, and an axiomatically appropriate, schematic constant specification $\mathcal{CS}$. Then, there is a ground proof polynomial $q_\psi$ that can be constructed in time polynomial to the size of $\psi$, s.t. for all assignments of $p_j^e$'s to $p$ or $\neg p$,

$$p_1^e \to p_2^e \to \cdots \to p_n^e \to \psi \text{ is a tautology}$$

$$\Longleftrightarrow$$

$$\vdash_{\mathcal{CS}} g_\psi : (p_1^e \to p_2^e \to \cdots \to p_n^e \to \psi).$$

Of course, the size of $g_\psi$ will also be a polynomial to the size of $\psi$.

**Proof.** First, a term $g_i$ will be constructed for every clause $C_i$, such that $\vdash_{\mathcal{CS}} g_i : (p_1^e \to p_2^e \to \cdots \to p_n^e \to C_i)$ for all assignments of $p_j$, or $\neg p_j$ to $p_j^e$ that makes $p_1^e \to p_2^e \to \cdots \to p_n^e \to C_i$ valid.
Here it is worth noticing that

a. Due to the substitution property, we can think of each $C_i$ to be of the form $(p \vee q \vee r)$

b. The substitutions of $p_i^e$'s that make $p_1^e \to p_2^e \to \cdots \to p_n^e \to (p^e \vee q^e \vee r^e)$ valid are exactly those that substitute $p^e$ with $p$, $q^e$ with $q$, or $r^e$ with $r$.

c. If terms $g_i^p, g_i^q, g_i^r$ are constructed, s.t. $\vdash_{\mathcal{CS}} g_i^x : (p_1^e \to p_2^e \to \cdots \to p_n^e \to (p^e \vee q^e \vee r^e))$ for all substitutions of $x^e$ with $x$, where $x \in \{p, q, r\}$, then the term $g_i^p + g_i^q + g_i^r$ will be $g_i$.

The formula $p \to (p \vee q \vee r)$ is (propositionally) valid. Thus, due to constructive necessitation, there exists a term $b$, s.t. $\vdash_{\mathcal{CS}} b : (p \to (p \vee q \vee r))$. There also exist terms $a_1, a_2$ s.t. $\vdash_{\mathcal{CS}} a_1 : ((A \to B) \to (A \to C \to B))$ and $\vdash_{\mathcal{CS}} a_2 : (A \to B \to A)$. Thus, if $C_i$ has in the place of $p$ a (positive or negative) appearance of $p_j$, then $g_i^p$ will be

$$\underbrace{a_2 \cdot (a_2 \cdots (a_2}_{j-1} \cdot (\underbrace{a_1 \cdot (a_1 \cdots (a_1 \cdot (a_1}_{n-j} \cdot b)) \cdots ))) \cdots ).$$

Now, the $g_i$'s are constructed and they are of size linear to $n$. Let $d, a_3$ be such that

$$\vdash_{\mathcal{CS}} d : ((A \to (B \to C)) \to (D \to A) \to (D \to B) \to (D \to C))$$

and

$$\vdash_{\mathcal{CS}} a_3 : (A \to B \to (A \wedge B)).$$

Then,

$$\vdash_{\mathcal{CS}} d \cdot a_3 : ((D \to A) \to (D \to B) \to (D \to (A \wedge B))),$$

and if $b^* = \underbrace{d \cdots (d}_{n} \cdot a_5) \cdots )$, then

$$\vdash_{\mathcal{CS}} b^* : (p_1^e \to \cdots \to p_n^e \to C_1) \to (p_1^e \to \cdots \to p_n^e \to C_2) \to$$
$$\to (p_1^e \to \cdots \to p_n^e \to C_1 \wedge C_2),$$

and this implies that $g_\psi = (b^* \cdots ((b^* \cdot g_1) \cdot g_2) \cdots g_{m-1}) \cdot g_m$.
To complete the proof, what needs to be shown is that

$$p_1^e \to p_2^e \to \cdots \to p_n^e \to \psi \text{ is a tautology}$$
$$\Longleftrightarrow$$
$$\vdash_{\mathcal{CS}} g_\psi : (p_1^e \to p_2^e \to \cdots \to p_n^e \to \psi).$$

The "$\Longrightarrow$" part occurs from the above construction. For the reverse, a simple observation is needed. In $LP_{\mathcal{CS}}$,

$$\vdash_{\mathcal{CS}} g_\psi : (p_1^e \to p_2^e \to \cdots \to p_n^e \to \psi)$$

implies

$$\vdash_{\mathcal{CS}} (p_1^e \to p_2^e \to \cdots \to p_n^e \to \psi)$$

due to factivity. So,

$$g_\psi : (p_1^e \to p_2^e \to \cdots \to p_n^e \to \psi)$$

is derivable in $LP_{\mathcal{CS}}$ only if

$$(p_1^e \to p_2^e \to \cdots \to p_n^e \to \psi)$$

is a tautology. Since $LP_{\mathcal{CS}}$ can derive more formulas than $J4_{\mathcal{CS}}$, this also holds for $J4_{\mathcal{CS}}$, thus completing the proof.

□

Now, to prove the first part of the theorem:
By reduction from $\forall\exists 3SAT$. Let $\forall p_1, \ldots, p_n \exists q_1, \ldots, q_m \psi(p_1, \ldots, p_n, q_1, \ldots, q_m)$ be a quantified boolean formula, where $p_1, \ldots, p_n, q_1, \ldots, q_m$ are propositional variables and $\psi$ is a quantifier-free propositional formula using variables from $p_1, \ldots, p_n, q_1, \ldots, q_m$. Using the lemma, we have constructed the term $g_\psi$. The produced formula, $F$, is

$$[(x_1 : p_1 \vee x_1 : \neg p_1) \wedge \cdots \wedge (x_n : p_n \vee x_n : \neg p_n)$$
$$\wedge (y_1 : q_1 \wedge z_1 : \neg q_1) \wedge \cdots \wedge (y_m : q_m \wedge z_m : \neg q_m)] \to$$
$$\to (g_\psi \cdot x_1 \cdots x_n \cdot (y_1 + z_1) \cdots (y_m + z_m)) : \psi.$$

This formula can easily be seen that it can be constructed in time polynomial with respect to $m+n$. What must be proven is that $F$ is provable in $J4_{\mathcal{CS}}$ if and only if $\forall p_1, \ldots, p_n \exists q_1, \ldots, q_m \psi$ holds.
By the deduction theorem, the above formula is provable in $J4_{\mathcal{CS}}$ iff

$$(x_1 : p_1 \vee x_1 : \neg p_1), \cdots, (x_n : p_n \vee x_n : \neg p_n),$$

$$(y_1 : q_1), (z_1 : \neg q_1), \cdots, (y_m : q_m), (z_m : \neg q_m)$$

$$\vdash_{\mathcal{CS}} (g_\psi \cdot x_1 \cdots x_n \cdot (y_1 + z_1) \cdots (y_m + z_m)) : \psi,$$

which in turn holds if and only if for every possible assignment of the $p_i^e$'s to $p_i$ or $\neg p_i$,

$$(x_1 : p_1^e), \cdots, (x_n : p_n^e), (y_1 : q_1), (z_1 : \neg q_1), \cdots, (y_m : q_m), (z_m : \neg q_m)$$

$$\vdash_{\mathcal{CS}} (g_\psi \cdot x_1 \cdots x_n \cdot (y_1 + z_1) \cdots (y_m + z_m)) : \psi.$$

Of course, this derivation is possible iff it is possible in the $*!_{\mathcal{CS}}$-calculus; and since the derived formula is not in the premises, then it follows from a formula of the form $(y_m + z_m) : H$ and

$$(g_\psi \cdot x_1 \cdots x_n \cdot (y_1 + z_1) \cdots (y_{m-1} + z_{m-1})) : H \to \psi.$$

Similarly, $(y_m + z_m) : H$ could only occur from $y_m : H$, or $z_m : H$, thus $H$ is either $q_m$, or $\neg q_m$, call it $q_m^e$. Now, we have that

$$(x_1 : p_1^e), \cdots, (x_n : p_n^e), (y_1 : q_1), (z_1 : \neg q_1), \cdots, (y_m : q_m), (z_m : \neg q_m)$$

$$\vdash_{\mathcal{CS}} (g_\psi \cdot x_1 \cdots x_n) : q_1^e \to \cdots \to q_m^e \to \psi.$$

As before, a formula $H$ exists, s.t.

$$(g_\psi \cdot x_1 \cdots x_n) : q_1^e \to \cdots \to q_m^e \to \psi$$

occurs from $x_n : H$ and

$$(g_\psi \cdot x_1 \cdots x_{n-1}) : H \to q_1^e \to \cdots \to q_m^e \to \psi.$$

Of course, $H$ now is $p_n^e$. Proceeding similarly for all $p_i^e$, it is shown that the initial formula is derivable in $J4_{\mathcal{CS}}$ if and only if for every assignment of the $p_i^e$'s to $p^e$ or $\neg p^e$, there is an assignment of the $q_i^e$'s to $q_i$ or $\neg q_i$, s.t.

$$(x_1 : p_1^e), \ldots, (x_n : p_n^e), (y_1 : q_1), (z_1 : \neg q_1), \ldots, (y_m : q_m), (z_m : \neg q_m)$$

$$\vdash_{\mathcal{CS}} g_\psi : p_1^e \to \cdots \to p_n^e \to q_1^e \to \cdots \to q_m^e \to \psi.$$

And since the premises contain no constants and $g_\psi$ is a ground term, because when a $*!_{\mathcal{CS}}$-calculus rule is used the terms of the hypotheses are subterms of the term in the conclusion, in the above proof, the premises are not needed. Therefore, the initial formula is derivable in $J4_{\mathcal{CS}}$ if and only if for every assignment of the $p_i^e$'s to $p^e$ or $\neg p^e$, there is an assignment of the $q_i^e$'s to $q_i$ or $\neg q_i$, s.t.

$$\vdash_{\mathcal{CS}} g_\psi : p_1^e \to \cdots \to p_n^e \to q_1^e \to \cdots \to q_m^e \to \psi.$$

By lemma 3.3.4, the proof for the first part is complete.

Now, the proof for the second part of the theorem (for $LP_{\mathcal{CS}}$) will be given. Again, this will be done by reduction from $\forall\exists 3SAT$, and again, let $\forall p_1, \ldots, p_n \exists q_1, \ldots, q_m \psi(p_1, \ldots, p_n, q_1, \ldots, q_m)$ be a quantified boolean formula, where $p_1, \ldots, p_n, q_1, \ldots, q_m$ are propositional variables and $\psi$ is a quantifier-free propositional formula using variables from $p_1, \ldots, p_n, q_1, \ldots, q_m$. The fact that $\mathcal{CS}$ is axiomatically appropriate and schematically injective will play a crucial role in the following. Let $c_2, c_L, c_R$ be justification constants s.t. $c_2 : A \in \mathcal{CS}$ iff $A$ is an instance of the application axiom, $c_L : A \in \mathcal{CS}$ iff $A$ is an instance of $s : F \to s + t : F$ and $c_R : A \in \mathcal{CS}$ iff $A$ is an instance of $s : F \to t + s : F$. Also, let $g_\psi$ be the one provided by lemma 3.3.4.
Then, $k_0, k_1, \ldots, k_n$ will be defined inductively: $k_0 =!g_\psi$ and $k_{i+1} = c_2 \cdot k_i \cdot !x_{i+1}$. By its construction, the length of $k_n$ is a polynomial of the size of $\psi$. From lemma 3.3.4, it is known that:

$$p_1^e \to \cdots \to p_n^e \to q_1^e \to \cdots \to q_m^e \to \psi \text{ is valid}$$

$$\Updownarrow$$

$$\vdash_{\mathcal{CS}} g_\psi : (p_1^e \to \cdots \to p_n^e \to q_1^e \to \cdots \to q_m^e \to \psi)$$

$$\Updownarrow$$

$$x_1 : p_1^e, \ldots, x_n : p_n^e \vdash_{\mathcal{CS}} (g_\psi \cdot x_1 \cdot x_2 \cdots x_n) : (q_1^e \to \cdots \to q_m^e \to \psi).$$

By lemma 2.1.1, the last part is equivalent to

$$x_1 : p_1^e, \ldots, x_n : p_n^e \vdash_{\mathcal{CS}} k : ((g_\psi \cdot x_1 \cdot x_2 \cdots x_n) : (q_1^e \to \cdots \to q_m^e \to \psi)),$$

for some term $k$. In fact, for $\Gamma_n$ being $\{x_1 : p_1^e, \ldots, x_n : p_n^e\}$,

*Lemma* 3.3.5. For $i \leq n$, $\Gamma_n \vdash_{\mathcal{CS}} k_i : H$ if and only if $H$ is of the form $(g_\psi \cdot x_1 \cdot x_2 \cdots x_i) : G$ and $\Gamma_n \vdash_{\mathcal{CS}} g_\psi : p_1^e \to \cdots \to p_i^e \to G$

**Proof. For $i = 0$,** the proposition simply states that $\Gamma_n \vdash !g_\psi : H$ if and only if $H$ is of the form $g_\psi : G$ and $\Gamma_n \vdash g_\psi : G$, which according to the rules of the $*!_{\mathcal{CS}}-$calculus is true.

**Assuming the lemma holds for $i$, it will be shown for $i+1$.** $\Gamma_n \vdash_{\mathcal{CS}} c_2 \cdot k_i \cdot !x_{i+1} : H$ if and only if there is some $F$, s.t. $\Gamma_n \vdash_{\mathcal{CS}} !x_{i+1} : F$ and $\Gamma_n \vdash_{\mathcal{CS}} c_2 \cdot k_i : F \to H$. It is evident that $F$ is $x_{i+1} : p^e_{i+1}$. Therefore, $\Gamma_n \vdash_{\mathcal{CS}} c_2 \cdot k_i \cdot !x_{i+1} : H$ if and only if $\Gamma_n \vdash_{\mathcal{CS}} c_2 \cdot k_i : (x_{i+1} : p^e_{i+1}) \to H$. Again, this holds iff there is some formula $F'$ s.t. $\Gamma_n \vdash_{\mathcal{CS}} k_i F'$ and $\Gamma_n \vdash_{\mathcal{CS}} c_2 : F' \to (x_{i+1} : p^e_{i+1}) \to H$. By the induction hypothesis, this is equivalent to

$$\Gamma_n \vdash_{\mathcal{CS}} g_\psi : p^e_1 \to \cdots \to p^e_i \to G$$

and

$$\Gamma_n \vdash_{\mathcal{CS}} c_2 : ((g_\psi \cdot x_1 \cdot x_2 \cdots x_i) : G) \to (x_{i+1} : p^e_{i+1}) \to H$$

for some $G$. By the way $c_2$ was defined before, $((g_\psi \cdot x_1 \cdot x_2 \cdots x_i) : G) \to (x_{i+1} : p^e_{i+1}) \to H$ is an instance of the application axiom. Therefore, $G$ is of the form $p^e_{i+1} \to G'$, so $H$ is of the form $x_1 \cdots x_i \cdot x_{i+1} : G'$ and $\Gamma_n \vdash_{\mathcal{CS}} g_\psi : p^e_1 \to \cdots \to p^e_{i+1} \to G'$, thus completing the induction. □

What would be desirable is to be able to use the formula of the previous part and show that

$$(x_1 : p^e_1), \ldots, (x_n : p^e_n), (y_1 : q_1 \wedge z_1 : \neg q_1), \ldots, (y_m : q_m \wedge z_m : \neg q_m)$$
$$\vdash_{\mathcal{CS}} g_\psi \cdot x_1 \cdots x_n \cdot (y_1 + z_1) \cdots (y_m + z_m) : \psi$$

for any assignment of $p^e_i$'s, if and only if $\phi$ is true. However, this time, due to reflexion, $(y_1 : q_1 \wedge z_1 : \neg q_1), \ldots, (y_m : q_m \wedge z_m : \neg q_m)$ is inconsistent, thus proving anything, whether $\psi$ is satisfiable or not. On the other hand, from the previous part, we know that there is a proof that does not use reflexion for the above formula for all assignments of $p^e_i$'s, if and only if $\phi$ is true. This fact will be used and the restriction of not using reflexion will be encoded using a justification term $t$, such that

$$x_1 : p^e_1, \ldots, x_n : p^e_n \vdash_{\mathcal{CS}} t : [(y_1 : q_1 \wedge z_1 : \neg q_1) \wedge \cdots \wedge (y_m : q_m \wedge z_m : \neg q_m)] \to$$
$$\to (g_\psi \cdot x_1 \cdots x_n \cdot (y_1 + z_1) \cdots (y_m + z_m)) : \psi.$$

Specifically, that $t$ will be $t_\psi$; and let $t_\psi$ be

$$(g_0 \cdot \underbrace{(c_L + c_R) \cdot c_2 \cdot (c_L + c_R) \cdot c_2 \cdots (c_L + c_R) \cdot c_2}_{m}) \cdot k_n,$$

where $g_0$ is a ground term that contains only constants justifying propositional axioms and constructed along the lines of the proof of lemma 3.3.4, is

independent of the assignments of the $p_i^e$'s that make $\psi$ true and its length is a polynomial of the length of $\psi$. Let $F$ be:

$$[(x_1 : p_1 \vee x_1 : \neg p_1) \wedge \cdots \wedge (x_n : p_n \vee x_n : \neg p_n)]$$
$$\rightarrow t_\psi : [(y_1 : q_1 \wedge z_1 : \neg q_1) \wedge \cdots \wedge (y_m : q_m \wedge z_m : \neg q_m)] \rightarrow$$
$$\rightarrow (g_\psi \cdot x_1 \cdots x_n \cdot (y_1 + z_1) \cdots (y_m + z_m)) : \psi.$$

Of course, $\vdash_{\mathcal{CS}} F$ if and only if for each possible assignment of $p_i^e$'s to $p_i$ or $\neg p_i$,

$$x_1 : p_1^e, \ldots x_n : p_n^e \vdash_{\mathcal{CS}} t_\psi : [(y_1 : q_1 \wedge z_1 : \neg q_1) \wedge \cdots \wedge (y_m : q_m \wedge z_m : \neg q_m) \rightarrow$$
$$\rightarrow (g_\psi \cdot x_1 \cdots x_n \cdot (y_1 + z_1) \cdots (y_m + z_m)) : \psi].$$

Let $\Gamma_n$ be as before. Then,

*Lemma* 3.3.6. Let $\mathcal{CS}, c_2, c_L, c_R t_\psi, k_n$ be defined as above, and let $t$ be some justification term. Then,

$$\Gamma_n \vdash_{\mathcal{CS}} t_\psi : [(y_1 : q_1 \wedge z_1 : \neg q_1) \wedge \cdots \wedge (y_m : q_m \wedge z_m : \neg q_m) \rightarrow$$
$$\rightarrow (t \cdot x_1 \cdots x_n \cdot (y_1 + z_1) \cdots (y_m + z_m)) : \psi]$$

if and only if

$$\Gamma_n \vdash_{\mathcal{CS}} k_n : t : (q_1^e \rightarrow \cdots \rightarrow q_m^e \rightarrow \psi),$$

for some assignment of the $q_i^e$'s to $q_i^e$ or $\neg q_i^e$.

**Proof.** The lemma will be proven for $m = 2$. The proof generalizes easily for the general case. Thus, it will be proven that

$$\Gamma_n \vdash_{\mathcal{CS}} (g_0 \cdot (c_L + c_R) \cdot c_2 \cdot (c_L + c_R) \cdot c_2) \cdot k_n :$$
$$[((y_1 : q_1 \wedge z_1 : \neg q_1) \wedge (y_2 : q_2 \wedge z_2 : \neg q_2)) \rightarrow (t \cdot (y_1 + z_1) \cdot (y_2 + z_2)) : \psi]$$

implies that $\Gamma_n \vdash_{\mathcal{CS}} k_n : t : (q_2^e \rightarrow q_1^e \rightarrow \psi)$, for some assignment of the $q_i^e$'s. All the implications will also hold for the reverse, but for convenience the proof will focus only on one direction.
Starting from

$$\Gamma_n \vdash_{\mathcal{CS}} (g_0 \cdot (c_L + c_R) \cdot c_2 \cdot (c_L + c_R) \cdot c_2) \cdot k_n :$$
$$[((y_1 : q_1 \wedge z_1 : \neg q_1) \wedge (y_2 : q_2 \wedge z_2 : \neg q_2)) \rightarrow (t \cdot (y_1 + z_1) \cdot (y_2 + z_2)) : \psi],$$

there must be a formula $F_n$ s.t.

$$Gamma_n \vdash_{\mathcal{CS}} K_n : F_n$$

and

$$\Gamma_n \vdash_{\mathcal{CS}} (g_0 \cdot (c_L + c_R) \cdot c_2 \cdot (c_L + c_R) \cdot c_2) :$$

$$(F_n \to [((y_1 : q_1 \wedge z_1 : \neg q_1) \wedge (y_2 : q_2 \wedge z_2 : \neg q_2)) \to (t \cdot (y_1 + z_1) \cdot (y_2 + z_2)) : \psi]).$$

Again, similarly, and because of the rules of $*!_{\mathcal{CS}}$-calculus, there must be formulas $A_1, A_2, A_3, A_4$ such that

$$\Gamma_n \vdash_{\mathcal{CS}} c_2 : A_1$$

$$\Gamma_n \vdash_{\mathcal{CS}} (c_L + c_R) : A_2$$

$$\Gamma_n \vdash_{\mathcal{CS}} c_2 : A_3$$

$$\Gamma_n \vdash_{\mathcal{CS}} (c_L + c_R) : A_4$$

and

$$\Gamma_n \vdash_{\mathcal{CS}} g_0 : (A_4 \to A_3 \to A_2 \to A_1 \to F_n \to$$

$$\to [((y_1 : q_1 \wedge z_1 : \neg q_1) \wedge (y_2 : q_2 \wedge z_2 : \neg q_2)) \to (t \cdot (y_1 + z_1) \cdot (y_2 + z_2)) : \psi]).$$

It is evident that $A_1$ is of the form $t_1 : (F_1 \to G_1) \to (s_1 : F_1 \to (t_1 \cdot s_1) : G_1)$, $A_2$ is of the form $u_1 : H_1 \to (u_1 + v_1) : H_1$, or $v_1 : H_1 \to (u_1 + v_1) : H_1$, $A_3$ is of the form $t_2 : (F_2 \to G_2) \to (s_2 : F_2 \to (t_2 \cdot s_2) : G_2)$ and $A_4$ is of the form $u_2 : H_2 \to (u_2 + v_2) : H_2$, or $v_2 : H_2 \to (u_2 + v_2) : H_2$. Lets suppose $A_2$ is of the form $u_1 : H_1 \to (u_1 + v_1) : H_1$ and $A_4$ is of the form $v_2 : H_2 \to (u_2 + v_2) : H_2$. Therefore, we have that $\Gamma_n \vdash_{\mathcal{CS}} k_n : F_n$ and

$$\Gamma_n \vdash_{\mathcal{CS}} g_0((v_2 : H_2 \to (u_2 + v_2) : H_2)$$

$$\to (t_2 : (F_2 \to G_2) \to (s_2 : F_2 \to (t_2 \cdot s_2) : G_2))$$

$$\to (u_1 : H_1 \to (u_1 + v_1) : H_1)$$

$$\to (t_1 : (F_1 \to G_1) \to (s_1 : F_1 \to (t_1 \cdot s_1) : G_1)) \to F_n$$

$$\to [((y_1 : q_1 \wedge z_1 : \neg q_1) \wedge (y_2 : q_2 \wedge z_2 : \neg q_2)) \to (t \cdot (y_1 + z_1) \cdot (y_2 + z_2)) : \psi]),$$

for some $F_1, F_2, G_1, G_2, H_1, H_2, F_n, s_1, s_2, t_1, t_2, u_1, u_2, v_1, v_2$.
As $g_0$ is a ground term that contains only constants specifying propositional axioms, it follows that the above formula, without $g_0$ is a propositional tautology (seeing prefixed terms as atoms). What remains is to unify the formulas

- $y_1 : q_1$
- $z_1 : \neg q_1$
- $y_2 : q_2$

- $z_2 : \neg q_2$
- $t_1 : (F_1 \to G_1) \to (s_1 : F_1 \to (t_1 \cdot s_1) : G_1)$
- $t_2 : (F_2 \to G_2) \to (s_2 : F_2 \to (t_2 \cdot s_2) : G_2)$
- $u_1 : H_1 \to (u_1 + v_1) : H_1$
- $v_2 : H_2 \to (u_2 + v_2) : H_2$
- $F_n$

in such a way, that the conclusion $(t \cdot (y_1 + z_1) \cdot (y_2 + z_2)) : \psi$ is implied. We cannot use $F_n$, as, according to lemma 3.3.5, it cannot unify with any of the first four formulas, or the desired conclusion at this point. Keeping in mind that $y_i$'s, $z_i$'s and $q_i$'s are elementary, this can happen, resulting in $F_n$ unifying with $t : q_1^e \to q_2^e \to \psi$, for some assignment of $q_i^e$'s. Therefore, $\Gamma_n \vdash_{\mathcal{CS}} k_n : t : q_1^e \to q_2^e \to \psi$, which is the intended conclusion for the case of $m = 2$. As it was said before, this proof generalizes naturally to the general case for $m$. □

Now, if for all $i$, $p_i^e$ is any of $p_i$ or $\neg p_i$, and for those $p_i^e$'s the $q_i^e$'s are the ones guaranteed by lemma 3.3.6, then what needs to be shown is simply that

$$(x_1 : p_1^e), \ldots, (x_n : p_n^e) \vdash_{\mathcal{CS}} (g_\psi \cdot x_1 \cdots x_n) : q_1^e \to \cdots q_m^e \to \psi$$

if and only if $\psi$ is true under the valuation implied by the choices of the $p_i^e$'s and $q_i^e$'s. We know that

$$\begin{gathered}(x_1 : p_1^e), \ldots, (x_n : p_n^e) \vdash_{\mathcal{CS}} (g_\psi \cdot x_1 \cdots x_n) : q_1^e \to \cdots q_m^e \to \psi \\ \Updownarrow \text{ by lemma 3.3.5} \\ \vdash_{\mathcal{CS}} g_\psi : p_1^e \to \cdots \to p_n^e \to q_1^e \to \cdots \to q_m^e \to \psi.\end{gathered}$$

And the last statement is true if and only if $p_1^e \to \cdots \to p_n^e \to q_1^e \to \cdots \to q_m^e \to \psi$ is a tautology. This completes the proof of the second and last part of the theorem. □

**Corollary 3.3.5.** 1. $J4_{\mathcal{CS}}$ with a decidable schematic $\mathcal{CS}$ is $\Pi_2^p$-complete.

2. $LP_{\mathcal{CS}}$ with a decidable schematically injective axiomatically appropriate $\mathcal{CS}$ is $\Pi_2^p$-complete.

**Corollary 3.3.6.** $J4$ is $\Pi_2^p$-complete.

# Chapter 4

# Conclusions and Open Questions

Modal Logic has been studied extensively in the past and it is reasonable that the complexity of Validity for the normal modal logics presented here has been determined. On the other hand, Justification Logic is a relatively new field, presenting much variety and although important efforts have been made to determine the complexity of Validity in this case, many questions remain open. Below is a table that summarizes the complexity results presented in this thesis.

| Logic | Upper Bound | Lower Bound |
|---|---|---|
| Classical Logic | $NP$ | $NP$-hard |
| Intuitionistic Logic | $PSPACE$ | $PSPACE$-hard |
| $K$ | $PSPACE$ | $PSPACE$-hard |
| $D$ | $PSPACE$ | $PSPACE$-hard |
| $T$ | $PSPACE$ | $PSPACE$-hard |
| $S4$ | $PSPACE$ | $PSPACE$-hard |
| $S5$ | $NP$ | $NP$-hard |
| $KD45$ | $NP$ | $NP$-hard |
| $J_{\mathcal{CS}}$ | $\Pi_2^p$ | $NP$-hard |
| $JD_{\mathcal{CS}}$ | $\Pi_2^p$ | $NP$-hard |
| $JT_{\mathcal{CS}}$ | $\Pi_2^p$ | $NP$-hard |
| $J4_{\mathcal{CS}}$ | $\Pi_2^p$ | $\Pi_2^p$-hard |
| $JD4_{\mathcal{CS}}$ | Unknown | $NP$-hard |
| $LP_{\mathcal{CS}}$ | $\Pi_2^p$ | $\Pi_2^p$-hard |

The reader can see that no conditions are mentioned for the constant specification. It is assumed that the most strict restrictions mentioned so far

apply for each logic (except from $\mathcal{CS}$ being finite), for clarity. Even with this simplification, it is evident from looking at the table that many currently known upper and lower bounds do not match. Furthermore, for $JD4_{\mathcal{CS}}$, no reasonable upper bound is known and there are other Justification Logics, not defined here, $J5, J45, JD45, JT45$; these include the extra axiom $\neg t : F \rightarrow ?t : \neg t : F$ that corresponds to negative introspection and of course also use the extra symbol ? in their syntax. No $M$ models are known for these, or non-trivial complexity bounds.

Other topics could also be covered in this thesis. Only monomodal logics were considered. The purpose was to contrast these logics to their justification counterparts. Indeed, it is apparent that the explicitness of the justification terms causes the complexity of determining the validity of a formula dramatically (assuming $PH \neq PSPACE$). This contrast is magnified even more when the reflected fragments of the above justification logics are considered; there, the justification term plays an essential role.

In this thesis, no $F$-, or $Fk$-models were introduced. This is because, for the purposes of this thesis, $M$-models were sufficient. Also, it seems that $M$-models are more apporpriate for studying complexity issues, whenever available. $F$- (and $Fk$-) models are Kripke models with an admissible evidence function and were first introduced for $LP$ in [15], by Mel Fitting. Because of their simmilarity to Kripke models, they are more appropriate for epistemic settings, for comparisons to Modal Logic, or for considering hybrid logics that include both justification terms and normal modalities.

An obvious research direction from here would be to try to answer all these questions that have been left ananswered, considering justification logic. Another direction, both for modal and for justification logic would be to consider the problem of validity from a parameterized point of view. Of course, as the field of computational complexity theory is diverse, there may be even more meaningful ways to study satisfiability that have not been thought of before.

# Bibliography


[1] Sergei [N.] Artëmov. Logic of proofs. *Annals of Pure and Applied Logic*, 67(1–3):29–59, May 1994.

[2] Sergei N. Artemov. Operational modal logic. Technical Report MSI 95–29, Cornell University, December 1995.

[3] Sergei N. Artemov. Logic of Proofs: a unified semantics for modality and $\lambda$-terms. Technical Report CFIS 98–06, Cornell University, March 1998.

[4] Sergei N. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, March 2001.

[5] S[ergei] N. Artemov. Kolmogorov and Gödel's approach to intuitionistic logic: current developments. *Russian Mathematical Surveys*, 59(2):203–229, 2004. Originally published in Russian.

[6] Sergei [N.] Artemov. Justification logic. Technical Report TR–2007019, CUNY Ph.D. Program in Computer Science, October 2007. Later version can be found as [7].

[7] Sergei [N.] Artemov. The logic of justification. Technical Report TR–2008010, CUNY Ph.D. Program in Computer Science, September 2008.

[8] Sergei [N.] Artemov. Why do we need Justification Logic? Technical Report TR–2008014, CUNY Ph.D. Program in Computer Science, September 2008.

[9] Vladimir N. Brezhnev. On explicit counterparts of modal logics. Technical Report CFIS 2000–05, Cornell University, 2000.

[10] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM.

[11] Jacques Corbin and Michel Bidoit. A rehabilitation of robinson's unification algorithm. In *IFIP Congress*, pages 909–914, 1983.

[12] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.

[13] Melvin Fitting. Tableau methods of proof for modal logics. 1972.

[14] Melvin Fitting. A semantic proof of the realizability of modal logic in the Logic of Proofs. Technical Report TR–2003010, CUNY Ph.D. Program in Computer Science, September 2003.

[15] Melvin Fitting. A semantics for the Logic of Proofs. Technical Report TR–2003012, CUNY Ph.D. Program in Computer Science, September 2003.

[16] Melvin Fitting. The logic of proofs, semantically. *Annals of Pure and Applied Logic*, 132(1):1–25, February 2005.

[17] Melvin Fitting. A replacement theorem for LP. Technical Report TR–2006002, CUNY Ph.D. Program in Computer Science, March 2006.

[18] Melvin Fitting. Realizations and LP. In Sergei N. Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science, International Symposium, LFCS 2007, New York, NY, USA, June 4–7, 2007, Proceedings*, volume 4514 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2007.

[19] Melvin Fitting. Realizing substitution instances of modal theorems. Technical Report TR–2007006, CUNY Ph.D. Program in Computer Science, March 2007.

[20] Melvin Fitting. *Notes on Classical Propositional Logic*. 2008. Notes.

[21] Melvin Fitting and Richard L. Mendelsohn. *First-order modal logic*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[22] Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell.*, 54(3):319–379, 1992.

[23] Nikolai V. Krupski. On the complexity of the reflected logic of proofs. Technical Report TR–2003007, CUNY Ph.D. Program in Computer Science, May 2003. Later version published as [25].

[24] Nikolai V. Krupski. *On Certain Algorithmic Problems for Formal Systems with Internalization Property*. PhD thesis, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, April 2006. In Russian.

[25] Nikolai V. Krupski. On the complexity of the reflected logic of proofs. *Theoretical Computer Science*, 357(1–3):136–142, July 2006.

[26] Roman Kuznets. On the complexity of explicit modal logics. In *Proceedings of the 14th Annual Conference of the EACSL on Computer Science Logic*, pages 371–383, London, UK, 2000. Springer-Verlag.

[27] Roman Kuznets. On decidability of the logic of proofs with arbitrary constant specifications. In *2004 Annual Meeting of the Association for Symbolic Logic, Carnegie Mellon University, Pittsburgh, PA, May 19–23, 2004*, volume 11(1) of *Bulletin of Symbolic Logic*, page 111. Association for Symbolic Logic, March 2005. Abstract.

[28] Roman Kuznets. *Complexity Issues in Justification Logic*. PhD thesis, CUNY Graduate Center, May 2008.

[29] Roman Kuznets. Complexity through tableaux in justification logic. In *2008 European Summer Meeting of the Association for Symbolic Logic, Logic Colloquium '08, Bern, Switzerland, July 3–July 8, 2008*, volume 15(1) of *Bulletin of Symbolic Logic*, page 121. Association for Symbolic Logic, March 2009. Abstract.

[30] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

[31] Fabio Massacci. Strongly analytic tableaux for normal modal logics. In Alan Bundy, editor, *CADE*, volume 814 of *Lecture Notes in Computer Science*, pages 723–737. Springer, 1994.

[32] Fabio Massacci. Single step tableaux for modal logics. *J. Autom. Reasoning*, 24(3):319–364, 2000.

[33] Robert [S.] Milnikel. Derivability in certain subsystems of the Logic of Proofs is $\Pi_2^p$-complete. *Annals of Pure and Applied Logic*, 145(3):223–239, March 2007.

[34] Alexey Mkrtychev. Models for the logic of proofs. In Sergei Adian and Anil Nerode, editors, *Logical Foundations of Computer Science, 4th*

*International Symposium, LFCS'97, Yaroslavl, Russia, July 6–12, 1997, Proceedings*, volume 1234 of *Lecture Notes in Computer Science*, pages 266–275. Springer, 1997.

[35] Eric Pacuit. A note on some explicit modal logics. In *Proceedings of the 5th Panhellenic Logic Symposium*, pages 117–125, Athens, Greece, July 25–28, 2005. University of Athens.

[36] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[37] Natalia [M.] Rubtsova. Evidence reconstruction of epistemic modal logic S5. In Dima Grigoriev, John Harrison, and Edward A. Hirsch, editors, *Computer Science — Theory and Applications, First International Computer Science Symposium in Russia, CSR 2006, St. Petersburg, Russia, June 8–12, 2006, Proceedings*, volume 3967 of *Lecture Notes in Computer Science*, pages 313–321. Springer, 2006.

[38] Richard Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, 9:67–72, 1979.

[39] Vitezslav Svejdar. On the polynomial-space completeness of intuitionistic propositional logic. *Archive for Mathematical Logic*, 42(7):711–716, 2003.

[40] Stathis Zachos. *Computability and Complexity*. NTUA, 2006.