A Formal Proof of Modal Completeness for Provability Logic

Marco Maggesi
University of Florence, Italy

Cosimo Perini Brogi
University of Genoa, Italy

Abstract

This work presents a formalized proof of modal completeness for Gödel-Löb provability logic (GL) in the HOL Light theorem prover. We describe the code we developed, and discuss some details of our implementation. In particular, we show how we adapted the proof in the Boolos' monograph according to the formal language and tools at hand. The strategy we develop here overcomes the technical difficulty due to the non-compactness of GL, and simplify the implementation. Moreover, it can be applied to other normal modal systems with minimal changes.

2012 ACM Subject Classification Theory of computation → Higher order logic; Theory of computation → Modal and temporal logics; Theory of computation → Automated reasoning

Keywords and phrases Provability Logic, Higher-Order Logic, Mechanized Mathematics, HOL Light Theorem Prover

Digital Object Identifier 10.4230/LIPIcs.ITP.2021.26

Related Version *Previous Version*: `https://arxiv.org/abs/2102.05945`

Supplementary Material The implementation described in this paper is hosted in the subdirectory `GL` of the distribution of HOL Light theorem prover.
Software (HOL Light source code, January 2021): `https://github.com/jrh13/hol-light/`
archived at `swh:1:dir:288139a46b5250ca678d928a823173f4f69c8594`

Funding *Marco Maggesi*: Supported by the Italian Ministry of University and Research and by INdAM-GNSAGA.

1 Introduction

In this paper we wish to report on our results and general experience in using HOL Light theorem prover to formally verify some properties of Gödel-Löb provability logic (GL).

Our work starts with a deep embedding of the syntax of propositional modal logic together with the corresponding relational semantics. Next, we introduce the traditional axiomatic calculus $\mathbb{GL}$ and prove the validity of the system w.r.t. irreflexive transitive finite frames.

After doing that, the most interesting part of our work begins with the proof of a number of lemmas *in* $\mathbb{GL}$ that are necessary for our main goal, namely the development of a formal proof of modal completeness for the system.

In order to achieve that, we had to formally verify a series of preliminary lemmas and constructions involving the behaviour of syntactical objects used in the standard proof of the completeness theorem. These unavoidable steps are very often only proof-sketched in wide-adopted textbooks in logic, for they mainly involve "standard" reasoning *within* the proof system we are dealing with. But when we are working in a formal setting like we did with HOL Light, experience reveals that it is generally more convenient to adopt a different line of reasoning and to make a smart use of our formal tools, so that we can succeed in developing alternative (or simpler) proofs, still totally verified by the computer.

In other terms: in order to give a formal proof of the completeness theorem for $\mathbb{GL}$, that is

▶ **Theorem 1.** *For any formula A, $\mathbb{GL} \vdash A$ iff A is true in any irreflexive transitive finite frame.*

we needed to split down the main goal into several subgoals – dealing with both the object- and the meta-level – which might seem trivial in informal reasoning. However, in order to carefully check them it has been more convenient to apply different proof-strategies where HOL Light infrastructure played a more active role.

We can briefly summarise the present paper as follows:

- In Section 2, we introduce the basic ingredients of our development, namely the formal counterparts of the syntax and relational semantics for provability logic, along with some lemmas and general definitions which are useful to handle the implementation of these objects in a uniform way, i.e. without the restriction to the specific modal system we are interested in. The formalization constitutes large part of the file `modal.ml`;
- In Section 3, we formally define the axiomatic calculus $\mathbb{GL}$, and prove in a neat way the validity lemma for this system. Moreover, we give formal proofs of several lemmas *in* $\mathbb{GL}$ ($\mathbb{GL}$-lemmas, for short), whose majority is in fact common to all normal modal logics, so that our proofs might be re-used in subsequent implementations of different systems. This corresponds to contents of our code in `gl.ml`;
- Finally, in Section 4 we give our formal proof of modal completeness of $\mathbb{GL}$, starting with the definition of maximal consistent *lists* of formulae. In order to prove their syntactic properties – and, in particular, the extension lemma for consistent lists of formulae to maximal consistent lists – we use the $\mathbb{GL}$-lemmas and, at the same time, we adapt an already known general proof-strategy to maximise the gain from the formal tools provided by HOL Light – or, informally, from higher-order reasoning. Therefore, the proof we are proposing in that section follows the standard lines presented in e.g. [7] – from extension lemma to modal completeness via truth lemma – but it adopts tools, results, and techniques which are specific to the theorem prover we used, in line with a clear philosophical attitude in computer-aided proof development.

  At the end of the Section, we give the formal definition of bisimilarity for our setup and we prove the associated bisimulation theorem [19, Ch. 11]. Our notion of bisimilarity is polymorphic, in the sense that it can relate classes of frames sitting on different types. With this tool at hand, we can correctly state our completeness theorem in its natural generality (`COMPLETENESS_THEOREM_GEN`) – i.e. for irreflexive, transitive finite frames over any (infinite) type. These results, together with a simple decision procedure for $\mathbb{GL}$, are gathered in the file `completeness.ml`.

Our code is integrated into the current HOL Light distribution, and it is freely available from there.[1] We care to stress that our formalization does not tweak any original HOL Light tools, and it is therefore "foundationally safe". Moreover, since we only used that original formal infrastructure, our results can be easily translated into another theorem prover belonging to the HOL family – or, more generally, endowed with the same automation toolbox.

Before presenting our results, in the forthcoming subsections of this introduction we provide the reader with some background material both on provability logic, and on formal theorem proving in HOL Light: further information about modalities and HOL Light can be found in [12] and [15], respectively.

[1] See "Supplementary Material" on the first page of this paper.

1.1 Developments of Provability Logic

The origin of provability logic dates back to a short paper by Gödel [11] where propositions about provability are formalized by means of a unary operator B with the aim of giving a classical reading of intuitionistic logic.

The resulting system corresponds to the logic $\mathbb{S}4$, and the proposition Bp is interpreted as "p is *informally* provable" – as claimed by Gödel himself. This implies that $\mathbb{S}4$ can be considered a provability logic lacking an appropriate semantics.

At the same time, that work opened the question of finding an adequate modal calculus for the formal properties of the provability predicate used in Gödel's incompleteness theorems. That problem has been settled since 1970s for many formal systems of arithmetic by means of Gödel-Löb logic GL.

The corresponding axiomatic calculus $\mathbb{GL}$ consists of the axiomatic system for classical propositional logic, extended by the distributivity axiom schema K, the necessitation rule NR, and the axiom schema GL (see Section 3).

The schema GL is precisely a formal version of Löb's theorem, which holds for a wide class of arithmetical theories satisfying the so-called Hilbert-Bernays-Löb (HBL) provability conditions.[2]

The semantic counterpart of this calculus is - through the Kripke formalism - the logic of irreflexive transitive finite frames. Moreover, the calculus can be interpreted arithmetically in a sound and complete way. In other terms, $\mathbb{GL}$ solves the problem raised in Gödel's paper by identifying a propositional formal system for provability in all arithmetical theories that satisfies the previously mentioned HBL conditions.

Published in 1976, Solovay's arithmetical completeness theorem [21] is in this sense a milestone result in the fields of proof theory and modal logic. As $\mathbb{GL}$ is arithmetically complete, it is capable of capturing and identifying *all relevant properties of formal provability for arithmetic* in a very simple system, which is decidable and neatly characterised.

Such a deep result, however, uses in an essential way the *modal* completeness of $\mathbb{GL}$: Solovay's technique basically consists of an arithmetization of a relational countermodel for a given formula that is not a theorem of $\mathbb{GL}$, from which it is possible to define an appropriate arithmetical formula that is not a theorem of the mathematical system.

In contemporary research, this is still the main strategy to prove arithmetical completeness for other modalities for provability and related concepts, in particular for interpretability logic. In spite of this, for many theories of arithmetic – including Heyting Arithmetic – this technique cannot be applied, and no alternatives are known.

Therefore, on the one hand, completeness of formal systems w.r.t. the relevant relational semantics is still an unavoidable step in achieving the more substantial result of arithmetical completeness; on the other hand, however, the area of provability logic keeps flourishing and suggesting old and new open problems, closely related to the field of proof theory, but in fact bounded also to seeking a uniform proof-strategy to establish adequate semantics in formal theories of arithmetic having different strengths and flavours.[3]

[2] These properties of the formal predicate for arithmetical provability were isolated first in [16].

[3] The reader is referred to [3] for a survey of open problems in provability logics. As an instance of relevant applications of this kind of formal systems to traditional investigations in proof theory see e.g. [1].

1.2 HOL Light Notation

The HOL Light proof assistant [14] is based on *classical* higher-order logic with polymorphic type variables and where equality is the only primitive notion. From a logical viewpoint, the formal engine defined by the *term-conversions* and *inference rules* underlying HOL Light is the same as that described in [17], extended by an infinity axiom and the classical characterization of Hilbert's choice operator. From a practical perspective, it is a theorem prover privileging a procedural proof style development – i.e. when using it, we have to solve goals by applying *tactics* that reduce them to (eventually) simpler subgoals, so that the interactive aspect of proving is highlighted. Proof-terms can then be constructed by means of *tacticals* that compact the proof into few lines of code evaluated by the machine.

Logical operators – defined in terms of equality – and λ-abstraction are denoted by specific symbols in ASCII: for the reader's sake, we give a partial glossary in the next table. In the third column of the table, we also report the notation used for the object logic GL (introduced at the beginning of Section 2.1).

Informal notation	HOL notation	GL notation	Description
$\bot$	`F`	`False`	Falsity
$\top$	`T`	`True`	Truth
$\neg p$	`~ p`	`Not p`	Negation
$p \wedge q$	`/\`	`&&`	Conjunction
$p \vee q$	`\/`	`||`	Disjunction
$p \Longrightarrow q$	`==>`	`-->`	Implication
$p \Longleftrightarrow q$	`<=>`	`<->`	Biconditional
$\Box p$		`Box p`	Modal Operator
$p_1, \ldots p_N \vdash p$	`p1 ... pN |- p`		HOL theorem
$\vdash p$		`|-- p`	Derivability in $\mathbb{GL}$
$\forall x.\, P(x)$	`!x. P(x)`		Universal quantification
$\exists x.\, P(x)$	`?x. P(x)`		Existential quantification
$\lambda x.\, M(x)$	`\x. M(x)`		Lambda abstraction
$x \in s$	`x IN s`		Set membership

We recall that a Boolean function `s : α -> bool` is also called a *set on* α in the HOL parlance. The notation `x IN s` is equivalent to `s x` and must not be confused with a type annotation `x : α`.

In the following sections, we will directly state our results as theorems and definitions in the HOL Light syntax. Note that theorems are prefixed by the turnstile symbol, as in `|- 2 + 2 = 4`. We often report a theorem with its associated name, that is, the name of its associated OCaml constant, e.g.

```
ADD_SYM
  |- !m n. m + n = n + m
```

As expository style, we omit formal proofs at all, but the meaning of definitions, lemmas, and theorems in natural language is hopefully clear after the table we have just given.

We warn the reader that the HOL Light printing mechanism omits type information completely. However in this paper we manually add type annotations when they might be useful, or even indispensable, in order to avoid ambiguity – including the case of our main results, `COMPLETENESS_THEOREM` and `COMPLETENESS_THEOREM_GEN`.

As already told in the introduction, our contribution is now part of the HOL Light distribution. The reader interested in performing these results on her machine – and perhaps build further formalization on top of it – can run our code with the command

```
loadt "GL/make.ml";;
```

at the HOL Light prompt.

2 Basics of Modal Logic

As we stated previously, we deal with a logic that extends classical propositional reasoning by means of a single modal operator which is intended to capture the abstract properties of the provability predicate for arithmetic.

To reason about and within this logic, we have to "teach" HOL Light – our meta-language – how to identify it, starting with its syntax – the object-language – and semantics – the interpretation of this very object-language.

We want to keep everything neat and clean from a foundational perspective, therefore we will define *both* the object-language and its interpretation with no relation to the HOL Light environment. In other terms: our formulae and operators are *real* syntactic objects which we keep distinct from their semantic counterpart – and from the logical operators of the theorem prover too.

2.1 Language and Semantics Defined

Let us start by fixing the propositional modal language we will use throughout the present work. We consider *all* classical propositional operators – conjunction, disjunction, implication, equivalence, negation, along with the 0-ary symbols $\top$ and $\bot$ – and we add a modal unary connective $\Box$. The starting point is, as usual, a denumerable infinite set of propositional atoms $a_0, a_1, \cdots$. Accordingly, formulae of this language will have one of the following form

$$a \mid A \wedge B \mid A \vee B \mid A \to B \mid A \leftrightarrow B \mid \neg A \mid \top \mid \bot \mid \Box A \ .$$

The following code extends the HOL system with an the **inductive type of formulae** up to the **atoms** – which we identify with the denumerable type of strings – by using the above **connectives**:

```
let form_INDUCT,form_RECURSION = define_type
  "form = False
        | True
        | Atom string
        | Not form
        | && form form
        | || form form
        | --> form form
        | <-> form form
        | Box form";;
```

Next, we turn to the semantics for our modal language. We use **relational models** – aka Kripke models[4]. Formally, a **Kripke frame** is made of a non-empty set "of possible worlds" `W`, together with a binary relation `R` on `W`. To this, we add a valuation function `V` which assigns to each atom of our language and each world `w` in `W` a Boolean value. This is extended to a **forcing relation** `holds`, defined recursively on the structure of the input formula `p`, that computes the truth-value of `p` in a specific world `w`:

```
let holds = new_recursive_definition form_RECURSION
  `(holds f V False (w:W) <=> F) /\
   (holds f V True w <=> T) /\
   (holds f V (Atom s) w <=> V s w) /\
```

[4] See [9] for the historical development of this notion.

```
  (holds f V (Not p) w <=> ~(holds f V p w)) /\
  (holds f V (p && q) w <=> holds f V p w /\ holds f V q w) /\
  (holds f V (p || q) w <=> holds f V p w \/ holds f V q w) /\
  (holds f V (p --> q) w <=> holds f V p w ==> holds f V q w) /\
  (holds f V (p <-> q) w <=> holds f V p w <=> holds f V q w) /\
  (holds f V (Box p) w <=> !u. u IN FST f /\ SND f w u ==> holds f V p u)‘;;
```

In the previous lines of code, `f` stands for a generic Kripke frame – i.e., a pair `(W,R)` of a set of worlds and an accessibility relation – and `V` is an evaluation of propositional variables. Then, the **validity** of a formula `p` with respect to a frame `(W,R)`, and a class of frames `L`, denoted respectively `holds_in (W,R) p` and `L |= p`, are

```
let holds_in = new_definition
  ‘holds_in (W,R) p <=> !V w. w IN W ==> holds (W,R) V p w‘;;

let valid = new_definition
  ‘L |= p <=> !f. L f ==> holds_in f p‘;;
```

The above formalization is essentially the one presented in Harrison’s HOL Light Tutorial [15, § 20]. Notice that the usual notion of Kripke frame requires that the set of possible worlds is non-empty: that condition could be imposed by adapting the `valid` relation. We have preferred to stick to Harrison’s original definitions in our code, but in the next section, when we define the classes of frames we are dealing with, the requirement on `W` is correctly integrated in the corresponding types.

2.2 Frames for GL

For carrying out our formalization, we are interested in the logic of the (non-empty) frames whose underlying relation R is **transitive** and conversely well-founded – aka **Noetherian** – on the corresponding set of possible worlds; in other terms, we want to study the modal tautologies in models based on an accessibility relation R on W such that

- if xRy and yRz, then xRz; and
- for no $X \subseteq W$ there are infinite R-chains $x_0Rx_1Rx_2\cdots$.

In HOL Light, `WF R` states that `R` is a well-founded relation, so that we express the latter condition as `WF(\x y. R y x)`. Here we see a recurrent motif in logic: defining a system from the semantic perspective requires non-trivial tools from the foundational point of view, for, to express the second condition, a first-order language is not enough. However, that is not an issue here, since our underlying system is natively higher order:

```
let TRANSNT = new_definition
  ‘TRANSNT (W:W->bool,R:W->W->bool) <=>
   ~(W = {}) /\
   (!x y:W. R x y ==> x IN W /\ y IN W) /\
   (!x y z:W. x IN W /\ y IN W /\ z IN W /\ R x y /\ R y z ==> R x z) /\
   WF(\x y. R y x)‘;;
```

We warn the reader that in the previous statement there occur two interrelated mathematical objects both denoted `W` (for convenience): one is the type `W` and the other is the set `W` on the former (in the sense explained in the introduction about the HOL syntax). From a theoretical point of view, moreover, the question has no deep consequences as we can characterize this class of frames by using a *propositional* language extended by a modal operator $\Box$ that satisfies the *Gödel-Löb axiom schema* (GL) : $\Box(\Box A \to A) \to \Box A$. Here is the formal version of our claim:

```
TRANSNT_EQ_LOB
  |- !W:W->bool R:W->W->bool.
       (!x y:W. R x y ==> x IN W /\ y IN W)
       ==> ((!x y z. x IN W /\ y IN W /\ z IN W /\ R x y /\ R y z ==> R x z) /\
            WF (\x y. R y x) <=>
            (!p. holds_in (W,R) (Box(Box p --> p) --> Box p)))
```

The informal proof of the above result is standard and can be found in [7, Theorem 10] and in [19, Theorem 5.7]. The computer implementation of the proof is made easy thanks to Harrison's tactic `MODAL_SCHEMA_TAC` for semantic reasoning in modal logic, documented in [15, § 20.3].

By using this preliminary result, we could say that the frame property of being transitive and Noetherian can be captured by Gödel-Löb modal axiom, without recurring to a higher-order language. Nevertheless, that class of frames is not particularly informative from a logical point of view: a frame in `TRANSNT` can be too huge to be used in practice – for instance, for checking whether a formula is indeed a theorem of our logic. In particular, when aiming at a completeness theorem, one wants to consider models that are useful for further investigations on the properties of the very logic under consideration – in the present case, decidability of GL, which, as for any other normal modal logic, is an easy corollary of the *finite model property* [19, Ch. 13].

To this aim we note that by definition of Noetherianness, our R cannot be reflexive – otherwise $xRxRx\cdots$ would give us an infinite R-chain. This is not enough: following our main reference [7], the frames we want to investigate are precisely those whose W is **finite**, and whose R is both **irreflexive** and **transitive**:

```
let ITF = new_definition
  `ITF (W:W->bool,R:W->W->bool) <=>
   ~(W = {}) /\
   (!x y:W. R x y ==> x IN W /\ y IN W) /\
   FINITE W /\
   (!x. x IN W ==> ~R x x) /\
   (!x y z. x IN W /\ y IN W /\ z IN W /\ R x y /\ R y z ==> R x z)`;;
```

Now it is easy to see that `ITF` is a subclass of `TRANSNT`:

```
ITF_NT
  |- !W R:W->W->bool. ITF(W,R) ==> TRANSNT(W,R)
```

That will be the class of frames whose logic we are now going to define syntactically.

3 Axiomatizing GL

We want to identify the logical system generating all the modal tautologies for transitive Noetherian frames; more precisely, we want to isolate the *generators* of the modal tautologies in the subclass of transitive Noetherian frames which are finite, transitive, and irreflexive. Notice that the lemma `ITF_NT` allows us to derive the former result as a corollary of the latter.

When dealing with the very notion of tautology – or *theoremhood*, discarding the complexity or structural aspects of *derivability* in a formal system – it is convenient to focus on axiomatic calculi. The calculus we are dealing with here is usually denoted by $\mathbb{GL}$.

It is clear from the definition of the forcing relation that for classical operators any axiomatization of propositional classical logic will do the job. Here, we adopt a basic system in which only $\to$ and $\bot$ are primitive – from the axiomatic perspective – and all the remaining

classical connectives are defined by axiom schemas and by the inference rule of Modus Ponens imposing their standard behaviour.[5]

As anticipated in the Introduction, to this classical engine we add

- the axiom schema K: $\Box(A \to B) \to \Box A \to \Box B$;
- the axiom schema GL: $\Box(\Box A \to A) \to \Box A$;
- the necessitation rule NR: $\dfrac{A}{\Box A}$ NR ,

where A, B are generic formulae (not simply atoms). Then, here is the complete definition of the **axiom system** $\mathbb{GL}$. The set of axioms is encoded via the inductive predicate `GLaxiom`:

```
let GLaxiom_RULES,GLaxiom_INDUCT,GLaxiom_CASES = new_inductive_definition
  `(!p q. GLaxiom (p --> (q --> p))) /\
   (!p q r. GLaxiom ((p --> q --> r) --> (p --> q) --> (p --> r))) /\
   (!p. GLaxiom (((p --> False) --> False) --> p)) /\
   (!p q. GLaxiom ((p <-> q) --> p --> q)) /\
   (!p q. GLaxiom ((p <-> q) --> q --> p)) /\
   (!p q. GLaxiom ((p --> q) --> (q --> p) --> (p <-> q))) /\
   GLaxiom (True <-> False --> False) /\
   (!p. GLaxiom (Not p <-> p --> False)) /\
   (!p q. GLaxiom (p && q <-> (p --> q --> False) --> False)) /\
   (!p q. GLaxiom (p || q <-> Not(Not p && Not q))) /\
   (!p q. GLaxiom (Box (p --> q) --> Box p --> Box q)) /\
   (!p. GLaxiom (Box (Box p --> p) --> Box p))`;;
```

The judgment $\mathbb{GL} \vdash A$, denoted `|-- A` in the machine code (not to be confused with the symbol for HOL theorems `|-`), is also inductively defined in the expected way:

```
let GLproves_RULES,GLproves_INDUCT,GLproves_CASES = new_inductive_definition
  `(!p. GLaxiom p ==> |-- p) /\
   (!p q. |-- (p --> q) /\ |-- p ==> |-- q) /\
   (!p. |-- p ==> |-- (Box p))`;;
```

3.1 $\mathbb{GL}$-lemmas

As usual, $\mathbb{GL} \vdash A$ denotes the existence of a derivation of A from the axioms of $\mathbb{GL}$; we could also define a notion of derivability from a set of assumptions just by tweaking the previous definitions in order to handle the specific limitations on NR – so that the deduction theorem would hold [13] – but this would be inessential to our intents.

Proving some lemmas in the axiomatic calculus $\mathbb{GL}$ is a technical interlude necessary for obtaining the completeness result.

In accordance with this aim, we denoted the classical axioms and rules of the system as the propositional schemas used by Harrison in the file `Arithmetic/derived.ml` of the HOL Light standard distribution [14] – where, in fact, many of our lemmas relying only on the propositional calculus are already proven there w.r.t. an axiomatic system for first-order classical logic; our further lemmas involving modal reasoning are denoted by names that are commonly used in informal presentations.

Therefore, the code in `gl.ml` mainly consists of the formalized proofs of those lemmas *in* $\mathbb{GL}$ that are useful for the formalized results we present in the next section. This file might be thought of as a "kernel" for further experiments in reasoning about axiomatic calculi by using HOL Light. The lemmas we proved are, indeed, standard tautologies of classical

[5] This is essentially the calculus introduced by Church in [8]

propositional logic, along with specific theorems of minimal modal logic and its extension for transitive frames – i.e. of the systems $\mathbb{K}$ and $\mathbb{K}4$ [19] –, so that by applying minor changes in basic definitions, they are – so to speak – take-away proof-terms for extensions of that very minimal system within the realm of normal modal logics.

More precisely, we have given, whenever it was useful, a "sequent-style natural deduction" characterization of classical operators both in terms of an implicit (or internal) deduction – and in that case we named the lemma with the suffix `_th` –, such as

```
GL_modusponens_th
  |- !p q. |-- ((p --> q) && p --> q)
```

and as a derived rule of the axiomatic system mimicking the behaviour of the connective in Gentzen's formalism, e.g.,

```
GL_and_elim
  |- !p q r. |-- (r --> p && q) ==> |-- (r --> q)  /\ |-- (r --> p)
```

We had to prove about 120 such results of varying degree of difficulty. We believe that this file is well worth the effort of its development, for two main reasons to be considered – along with the just mentioned fact that they provide a (not so) minimal set of internal lemmas which can be moved to different axiomatic calculi at, basically, no cost.

Indeed, on the one hand, these lemmas simplify the subsequent formal proofs involving consistent lists of formulae since they let us work formally within the scope of $\vdash$, so that we can rearrange subgoals according to their most useful equivalent form by applying the appropriate $\mathbb{GL}$-lemma(s).

On the other hand, the endeavour of giving formal proofs of these lemmas of the calculus $\mathbb{GL}$ has been important for checking how much our proof-assistant is "friendly" and efficient in performing this specific task.

As it is known, any axiomatic system fits very well an investigation involving the notion of *theoremhood* for a specific logic, but its lack of naturalness w.r.t. the practice of developing informal proofs makes it an unsatisfactory model for the notion of *deducibility*. In more practical terms: developing a formal proof of a theorem in an axiomatic system *by pencil and paper* can be a dull and uninformative task.

We therefore left the proof-search to the HOL Light toolbox as much as possible. Unfortunately, we have to express mixed feelings on the general experience. In most cases, relying on the automation tools of this specific proof assistant did indeed save our time and resources when trying to give a formal proof in $\mathbb{GL}$. Nevertheless, there has been a number of $\mathbb{GL}$-lemmas for proving which those automation tools did not revealed useful at all. In those cases, actually, we had to perform a tentative search of the specific instances of axioms from which deriving the lemmas,[6] so that interactive proving them had advantages as well as traditional instruments of everyday mathematicians.

Just to stress the general point: it is clearly possible – and actually useful in general – to rely on the resources of HOL Light to develop formal proofs both *about* and *within* an axiomatic calculus for a specific logic, in particular when the lemmas of the object system have relevance or practical utility for mechanizing (meta-)results on it; however, these very resources – and, as far as we can see, the tools of any other general proof assistant – do not look peculiarly satisfactory for pursuing investigations on derivability within axiomatic systems.

[6] The HOL Light tactics for first-order reasoning `MESON` and `METIS` were unable, for example, to instantiate autonomously the obvious middle formula for the transitivity of an implication, or even the specific formulae of a schema to apply to the goal in order to rewrite it.

3.2 Soundness Lemma

At this point, we can prove that $\mathbb{GL}$ is **sound** – i.e. every formula derivable in the calculus is a tautology in the class of irreflexive transitive finite frames. This is obtained by simply unfolding the relevant definitions and applying theorems `TRANSNT_EQ_LOB` and `ITF_NT` of Section 2.2:

```
GL_TRANSNT_VALID
  |- !p. (|-- p) ==> TRANSNT:(W->bool)#(W->W->bool)->bool |= p

GL_ITF_VALID
  |- !p. |-- p ==> ITF:(W->bool)#(W->W->bool)->bool |= p
```

From this, we get a model-theoretic proof of **consistency for the calculus**

```
GL_consistent
  ~ |-- False
```

Having exhausted the contents of file `gl.ml`, we shall move to consider the most interesting part of our effort, namely the mechanized proof of completeness for the calculus w.r.t. this very same class of frames. That constitutes the remaining contents of our implementation, beside the auxiliary code in `misc.ml` furnishing some general results about lists of items with same type we needed to handle the subsequent constructions (but have a more general utility).

4 Completeness and Decidability

When dealing with normal modal logics, it is common to develop a proof of completeness w.r.t. relational semantics by using the so-called "canonical model method". This can be summarized as a standard construction of countermodels made of maximal consistent sets of formulae and an appropriate accessibility relation [19].

For $\mathbb{GL}$, we cannot pursue this strategy, since the logic is not compact: maximal consistent sets are (in general) infinite objects, though the notion of derivability involves only a finite set of formulae. We cannot therefore reduce the semantic notion of (in)coherent set of formulae to the syntactic one of (in)consistent set of formulae: when extending a consistent set of formulae to a maximal consistent one, we might end up with a *syntactically* consistent set that nevertheless cannot be *semantically* satisfied.

In spite of this, it is possible to achieve a completeness result by

1. identifying the relevant properties of maximal consistent sets of formulae; and
2. tweaking the definitions so that those properties hold for specific consistent sets of formulae related to the formula we want to find a countermodel to.

That is, basically, the key idea behind the proof given in [7, Ch. 5]. In that monograph, however, the construction of a maximal consistent set from a simply consistent one is only proof-sketched and relies on a syntactic manipulation of formulae. By using HOL Light we do succeed in giving a detailed proof of completeness as direct as that by Boolos. But, as a matter of fact, we can claim something more: we can do that by carrying out in a *very natural way* a tweaked Lindenbaum construction to extend consistent *lists* to maximal consistent ones. This way, we succeed in preserving the standard Henkin-style completeness proofs; and, at the same time, we avoid the symbolic subtleties sketched in [7] that have no real relevance for the argument, but have the unpleasant consequence of making the formalized proof unnecessarily long, so that the implementation would sound rather pedantic – or even dull.

Furthermore, the proof of the main lemma `EXTEND_MAXIMAL_CONSISTENT` is rather general and does not rely on any specific property of $\mathbb{GL}$: our strategy suits all the others normal (mono)modal logics – we only need to modify the subsequent definition of `STANDARD_RELATION` according to the specific system under consideration. Thus, we provide a way for establishing completeness à la Henkin *and* the finite model property without recurring to filtrations [19] of canonical models for those systems.

4.1 Maximal Consistent Lists

Following the standard practice, we need to consider consistent finite sets of formulae for our proof of completeness. In principle, we can employ general sets of formulae in the formalization, but, from the practical view-point, lists without repetitions are better suited, since they are automatically finite and we can easily manipulate them by structural recursion. We define first the operation of finite conjunction of formulae in a list:[7]

```
let CONJLIST = new_recursive_definition list_RECURSION
  `CONJLIST [] = True /\
   (!p X. CONJLIST (CONS p X) = if X = [] then p else p && CONJLIST X)`;;
```

We proceed by proving some properties on lists of formulae and some $\mathbb{GL}$-lemmas involving `CONJLIST`. In particular, since $\mathbb{GL}$ is a normal modal logic – i.e. its modal operator distributes over implication and preserves theoremhood – we have that our $\Box$ distributes over the conjunction of X so that we have `CONJLIST_MAP_BOX`:

$$\mathbb{GL} \vdash \Box \bigwedge X \leftrightarrow \bigwedge \Box X,$$

where $\Box X$ is an abuse of notation for the list obtained by "boxing" each formula in X.

We are now able to define the notion of **consistent list of formulae** and prove the main properties of this kind of objects:

```
let CONSISTENT = new_definition
  `CONSISTENT (l:form list) <=> ~ (|-- (Not (CONJLIST l)))`;;
```

In particular, we prove that:

- a consistent list cannot contain both A and $\neg A$ for any formula A, nor $\bot$ (see theorems `CONSISTENT_LEMMA`, `CONSISTENT_NC`, and `FALSE_IMP_NOT_CONSISTENT`, respectively);
- for any consistent list X and formula A, either $X + A$ is consistent, or $X + \neg A$ is consistent (`CONSISTENT_EM`), where $+$ denotes the usual operation of appending an element to a list.

Our **maximal consistent lists** w.r.t. a given formula A will be consistent lists that do not contain repetitions and that contain, for any subformula of A, that very subformula or its negation:[8]

```
let MAXIMAL_CONSISTENT = new_definition
  `MAXIMAL_CONSISTENT p X <=>
   CONSISTENT X /\ NOREPETITION X /\
   (!q. q SUBFORMULA p ==> MEM q X \/ MEM (Not q) X)`;;
```

[7] Notice that in the this definition we perform a case analysis where the singleton list is treated separately (i.e., we have `CONJLIST [p] = p`). This is slightly uncomfortable in certain formal proof steps: in retrospect, we might have used a simpler version of this function. However, since this is a minor detail, we preferred not to change our code.

[8] Here we define the set of subformulae of A as the reflexive transitive closure of the set of formulae on which the main connective of A operates: this way, the definition is simplified and it is easier to establish standard properties of the set of subformulae by means of general higher-order lemmas in HOL Light for the closure of a given relation.

where X is a list of formulae and MEM q X is the membership relation for lists. We then establish the main closure property of maximal consistent lists:

```
MAXIMAL_CONSISTENT_LEMMA
  |- !p X A b. MAXIMAL_CONSISTENT p X /\
               (!q. MEM q A ==> MEM q X) /\
               b SUBFORMULA p /\
               |-- (CONJLIST A --> b)
               ==> MEM b X
```

After proving some further lemmas with practical utility – in particular, the fact that any maximal consistent list behaves like a restricted bivalent evaluation for classical connectives (MAXIMAL_CONSISTENT_MEM_NOT and MAXIMAL_CONSISTENT_MEM_CASES) – we can finally define the ideal (type of counter)model we are interested in. The type STANDARD_MODEL consists, for a given formula p, of:

1. the set of maximal consistent lists w.r.t. p made of *subsentences* of p – i.e. its subformulae or their negations – as possible worlds;
2. an irreflexive transitive accessibility relation R such that for any subformula Box q of p and any world w, Box q is in w iff, for any x R-accessible from w, q is in x;
3. an atomic valuation that gives value T (true) to a in w iff a is a subformula of p.

After defining the relation of *subsentences* as

```
let SUBSENTENCE = new_definition
  `!p q. q SUBSENTENCE p <=>
         q SUBFORMULA p \/ (?q'. q = Not q' /\ q' SUBFORMULA p)`;;
```

we can introduce the corresponding code:

```
let GL_STANDARD_FRAME = new_definition
  `GL_STANDARD_FRAME p (W,R) <=>
   W = {w | MAXIMAL_CONSISTENT p w /\ (!q. MEM q w ==> q SUBSENTENCE p)} /\
   ITF (W,R) /\
   (!q w. Box q SUBFORMULA p /\ w IN W
          ==> (MEM (Box q) w <=> !x. R w x ==> MEM q x))`;;

let GL_STANDARD_MODEL = new_definition
  `GL_STANDARD_MODEL p (W,R) V <=>
   GL_STANDARD_FRAME p (W,R) /\
   (!a w. w IN W ==> (V a w <=> MEM (Atom a) w /\ Atom a SUBFORMULA p))`;;
```

4.2 Maximal Extensions

What we have to do now is to show that the type GL_STANDARD_MODEL is non-empty. We achieve this by constructing suitable maximal consistent lists of formulae from specific consistent ones.

Our original strategy differs from the presentation given in e.g. [7] for being closer to the standard Lindenbaum construction commonly used in proving completeness results. By doing so, we have been able to circumvent both the pure technicalities in formalizing the combinatorial argument sketched in [7, p.79] *and* the problem – apparently inherent to the Lindenbaum extension – due to the non-compactness of the system, as we mentioned before.

The main lemma states then that, from any consistent list X of subsentences of a formula A, we can construct a maximal consistent list of subsentences of A by extending (if necessary) X:

```
EXTEND_MAXIMAL_CONSISTENT
  |- !p X.
       CONSISTENT X /\
       (!q. MEM q X ==> q SUBSENTENCE p)
       ==> ?M. MAXIMAL_CONSISTENT p M /\
               (!q. MEM q M ==> q SUBSENTENCE p) /\
               X SUBLIST M
```

The proof-sketch is as follows: given a formula A, we proceed in a step-by-step construction by iterating over the subformulae B of A not contained in X. At each step, we append to the list X the subformula B – if the resulting list is consistent – or its negation $\neg B$ – otherwise.

This way, we are in the pleasant condition of carrying out the construction by using the HOL Light device efficiently, and, at the same time, we do not have to worry about the non-compactness of $\mathbb{GL}$, since we are working with finite objects – the type `list` – from the very beginning.

Henceforth, we see that – under the assumption that A is not a $\mathbb{GL}$-lemma – the set of possible worlds in `STANDARD_FRAME` w.r.t. A is non-empty, as required by the definition of relational structures:

```
NONEMPTY_MAXIMAL_CONSISTENT
  |- !p. ~ |-- p
         ==> ?M. MAXIMAL_CONSISTENT p M /\
                 MEM (Not p) M /\
                 (!q. MEM q M ==> q SUBSENTENCE p)
```

Next, we have to define an R satisfying the condition 2 for a `STANDARD_FRAME`; the following does the job:

```
let GL_STANDARD_REL = new_definition
  `GL_STANDARD_REL p w x <=>
   MAXIMAL_CONSISTENT p w /\
   (!q. MEM q w ==> q SUBSENTENCE p) /\
   MAXIMAL_CONSISTENT p x /\
   (!q. MEM q x ==> q SUBSENTENCE p) /\
   (!B. MEM (Box B) w ==> MEM (Box B) x /\ MEM B x) /\
   (?E. MEM (Box E) x /\ MEM (Not (Box E)) w)`;;
```

Such an accessibility relation, together with the set of the specific maximal consistent lists we are dealing with, defines a structure in `ITF` with the required properties:

```
ITF_MAXIMAL_CONSISTENT
  |- !p. ~ |-- p
         ==> ITF ({M | MAXIMAL_CONSISTENT p M /\
                       (!q. MEM q M ==> q SUBSENTENCE p)},
                  GL_STANDARD_REL p),

ACCESSIBILITY_LEMMA
  |- !p M w q.
       ~ |-- p /\
       MAXIMAL_CONSISTENT p M /\
       (!q. MEM q M ==> q SUBSENTENCE p) /\
       MAXIMAL_CONSISTENT p w /\
       (!q. MEM q w ==> q SUBSENTENCE p) /\
       MEM (Not p) M /\
       Box q SUBFORMULA p /\
       (!x. GL_STANDARD_REL p w x ==> MEM q x)
       ==> MEM (Box q) w,
```

4.3 Truth Lemma and Completeness

For our ideal model, it remains to reduce the semantic relation of forcing to the more tractable one of membership to the specific world. More formally, we prove – by induction on the complexity of the subformula B of A – that if $\mathbb{GL} \nvdash A$, then for any world w of the standard model, B holds in w iff B is member of w:

```
GL_truth_lemma
  |- !W R p V q.
       ~ |-- p /\
       GL_STANDARD_MODEL p (W,R) V /\
       q SUBFORMULA p
       ==> !w. w IN W ==> (MEM q w <=> holds (W,R) V q w),
```

Finally, we are able to prove the main result: if $\mathbb{GL} \nvdash A$, then the list $[\neg A]$ is consistent, and by applying `EXTEND_MAXIMAL_CONSISTENT`, we obtain a maximal consistent list X w.r.t. A that extends it, so that, by applying `GL_truth_lemma`, we have that $X \nVdash A$ in our standard model. The corresponding formal proof reduces to the application of those previous results and the appropriate instantiations:

```
COMPLETENESS_THEOREM
  |- !p. ITF:(form list->bool)#(form list->form list->bool)->bool |= p
         ==> |-- p,
```

Notice that the family of frames `ITF` is polymorphic, but, at this stage, our result holds only for frames on the domain `form list`, as indicated by the type annotation. This is not an intrinsic limitation: the next section is devoted indeed to generalize this theorem to frames on an arbitrary domain.

By using our `EXTEND_MAXIMAL_CONSISTENT` lemma, we succeeded in giving a rather quick proof of both completeness and the finite model property for $\mathbb{GL}$.[9]

Indeed, as an immediate corollary, we have that the system $\mathbb{GL}$ is decidable and, in principle, we could implement a decision procedure for it in OCaml. This is a not-so-easy task – especially if one seeks efficiency and completeness – and it is out of the scope of the present work. Nevertheless, we feel like offering a very rough approximation.

We define the tactic `GL_TAC` and its associated rule `GL_RULE` that perform the following steps: (1) apply the completeness theorem, (2) unfold some definitions, and (3) try to solve the resulting *semantic* problem using first-order reasoning.

```
let GL_TAC : tactic =
  MATCH_MP_TAC COMPLETENESS_THEOREM THEN
  REWRITE_TAC[valid; FORALL_PAIR_THM; holds_in; holds;
              ITF; GSYM MEMBER_NOT_EMPTY] THEN
  MESON_TAC[];;

let GL_RULE tm = prove(tm, REPEAT GEN_TAC THEN GL_TAC);;
```

The above naive strategy is able to prove automatically some lemmas which are common to normal modal logic, but require some effort when derived in an axiomatic system. As an example consider the following $\mathbb{GL}$-lemma:

[9] For the completeness w.r.t. transitive Noetherian frames, it is common – see [7, 19] – to reason on irreflexive transitive structures and derive the result as a corollary of completeness w.r.t. the latter class.

```
GL_box_iff_th
  |- !p q. |-- (Box (p <-> q) --> (Box p <-> Box q))
```

When developing a proof of it within the axiomatic calculus, we need to "help" HOL Light by instantiating several further $\mathbb{GL}$-lemmas, so that the resulting proof-term consists of ten lines of code. On the contrary, our rule is able to check it in few steps:

```
# GL_RULE `!p q. |-- (Box (p <-> q) --> (Box p <-> Box q))`;;
  0..0..1..6..11..19..32..solved at 39
  0..0..1..6..11..19..32..solved at 39
val it : thm = |- !p q. |-- (Box (p <-> q) --> (Box p <-> Box q))
```

In spite of this, the automation offered by `MESON` tactic is not enough when the generated semantic problem involves in an essential way the fact that our frames are finite (or Noetherian). So, for instance, our procedure is not able to prove the Gödel-Löb axiom

$$\mathbb{GL} \vdash \Box(\Box A \longrightarrow A) \longrightarrow \Box A\,.$$

This suggests the need for improving `GL_TAC` to handle more general contexts: in the long run, it is a likely-looking outcome of what we reached so far.

4.4 Generalizing via Bisimulation

As we stated before, our theorem `COMPLETENESS_THEOREM` provides the modal completeness for $\mathbb{GL}$ with respect to a semantics defined using models built on the type `:form list`. It is obvious that the same result must hold whenever we consider models built on any infinite type. To obtain a formal proof of this fact, we need to establish a *correspondence* between models built on different types. It is well-known that a good way to make rigorous such a correspondence is by means of the notion of *bisimulation* [5].

In our context, given two models `(W1,R1)` and `(W2,R2)` sitting respectively on types `:A` and `:B`, each with a valuation function `V1` and `V2`, a **bisimulation** is a binary relation `Z:A->B->bool` that relates two worlds `w1:A` and `w2:B` when they can *simulate* each other. The formal definition is as follows:

```
BISIMIMULATION
  |- BISIMIMULATION (W1,R1,V1) (W2,R2,V2) Z <=>
     (!w1:A w2:B.
        Z w1 w2
        ==> w1 IN W1 /\ w2 IN W2 /\
            (!a:string. V1 a w1 <=> V2 a w2) /\
            (!w1'. R1 w1 w1'  ==> ?w2'. w2' IN W2 /\ Z w1' w2' /\ R2 w2 w2') /\
            (!w2'. R2 w2 w2' ==> ?w1'. w1' IN W1 /\ Z w1' w2' /\ R1 w1 w1'))
```

Then, we say that two worlds are *bisimilar* if there exists a bisimulation between them:

```
let BISIMILAR = new_definition
  `BISIMILAR (W1,R1,V1) (W2,R2,V2) (w1:A) (w2:B) <=>
   ?Z. BISIMIMULATION (W1,R1,V1) (W2,R2,V2) Z /\ Z w1 w2`;;
```

The key fact is that the semantic predicate `holds` respects bisimilarity:

```
BISIMILAR_HOLDS
  |- !W1 R1 V1 W2 R2 V2 w1:A w2:B.
       BISIMILAR (W1,R1,V1) (W2,R2,V2) w1 w2
       ==> (!p. holds (W1,R1) V1 p w1 <=> holds (W2,R2) V2 p w2)
```

From this, we can prove that validity is preserved by bisimilarity. The precise statements are the following:

```
BISIMILAR_HOLDS_IN
  |- !W1 R1 W2 R2.
       (!V1 w1:A. ?V2 w2:B. BISIMILAR (W1,R1,V1) (W2,R2,V2) w1 w2)
       ==> (!p. holds_in (W2,R2) p ==> holds_in (W1,R1) p)

BISIMILAR_VALID
  |- !L1 L2.
       (!W1 R1 V1 w1:A.
          L1 (W1,R1) /\ w1 IN W1
          ==> ?W2 R2 V2 w2:B. L2 (W2,R2) /\
                              BISIMILAR (W1,R1,V1) (W2,R2,V2) w1 w2)
       ==> (!p. L2 |= p ==> L1 |= p)
```

In the last theorem, recall that the statement `L(W,R)` means that `(W R)` is a frame in the class of frames `L`.

Finally, we can explicitly define a bisimulation between `ITF`-models on the type `:form list` and on any infinite type `:A`. From this, it follows at once the desired generalization of completeness for $\mathbb{GL}$:

```
COMPLETENESS_THEOREM_GEN
  |- !p. INFINITE (:A) /\ ITF:(A->bool)#(A->A->bool)->bool |= p ==> |-- p
```

5 Related Work

Our formalization gives a mechanical proof of completeness for $\mathbb{GL}$ in HOL Light which sticks to the original Henkin's method for classical logic. In its standard version, its nature is synthetic and intrinsically semantic [10], and, as we stated before, it is the core of the canonical model construction for most of normal modal logic.

That very approach does not work for $\mathbb{GL}$. Nevertheless, the modified extension lemma we proved in our mechanization introduces an analytic flavour to the strategy – for building maximal consistent lists in terms of components of a given non-provable formula in the calculus – and shows that Henkin's idea can be applied to $\mathbb{GL}$ too modulo appropriate changes.

As far as we know, *no other mechanized proof of modal completeness for* $\mathbb{GL}$ has been given before, despite there exist formalizations of similar results for several other logics, manly propositional and first-order classical and intuitionistic logic.

Formalizations of completeness for *classical logic* define an established trend in interactive theorem proving since [20], where a Hintikka-style strategy is used to define a theoremhood checker for formulae built up by negation and disjunction only.

In fact, a very general treatment of systems for classical propositional logic is given in [18]. There, an axiomatic calculus is investigated along with natural deduction, sequent calculus, and resolution system in Isabelle/HOL, and completeness is proven by Hintikka-style method for sequent calculus first, to be lifted then to the other formalisms by means of translations of each system into the others. Their formalization is more ambitious than ours, but, at the same time, it is focused on a very different aim. A similar overview of meta-theoretical results for several calculi formalized in Isabelle/HOL is given in [6], where, again, a more general investigation – unrelated to modal logics – is provided.

Concerning the area of intuitionistic modalities, [2] gives a constructive proof of completeness of IS4 w.r.t. a specific relational semantics verified in Agda, but it uses natural deduction and apply modal completeness to obtain a normalization result for the terms of the associated λ-calculus.

A Henkin-style completeness proof for $\mathbb{S}5$ is formalised in Lean [4]. That work applies the standard method of canonical models – since $\mathbb{S}5$ is compact – but does not prove the finite model property for the logic.

More recently, Xu and Norrish in [22] have used the HOL4 theorem prover for a general treatment of model theory of modal systems. As a future work, it might be interesting to make use of the formalization therein along with the main lines of our implementation of axiomatic calculi to merge the two presentations – syntactic and semantic – in an exhaustive way.

References

1 Juan Aguilera and David Fernández-Duque. Strong completeness of provability logic for ordinal spaces. *The Journal of Symbolic Logic*, 82:608–628, November 2017. doi:10.1017/jsl.2017.3.

2 Miëtek Bak. Introspective Kripke models and normalisation by evaluation for the $\lambda^{\Box}$-calculus. 7th Workshop on Intuitionistic Modal Logic and Applications (IMLA 2017). https://github.com/mietek/imla2017/blob/master/doc/imla2017.pdf, 2017.

3 Lev Beklemishev and Albert Visser. Problems in the logic of provability. In *Mathematical Problems from Applied Logic I*, pages 77–136. Springer, 2006.

4 Bruno Bentzen. A Henkin-style completeness proof for the modal logic S5. *CoRR*, abs/1910.01697, 2019. arXiv:1910.01697.

5 Patrick Blackburn and Johan Van Benthem. Modal logic: a semantic perspective. In *Handbook of modal logic*, volume 3, pages 1–84. Elsevier, 2007.

6 Jasmin Christian Blanchette. Formalizing the Metatheory of Logical Calculi and Automatic Provers in Isabelle/HOL (Invited Talk). In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2019, page 1–13, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293880.3294087.

7 George Boolos. *The logic of provability*. Cambridge university press, 1995.

8 Alonzo Church. *Introduction to Mathematical Logic*. Princeton: Princeton University Press, 1956.

9 B. Jack Copeland. The genesis of possible worlds semantics. *Journal of Philosophical logic*, 31(2):99–137, 2002.

10 Melvin Fitting and Richard L Mendelsohn. *First-order modal logic*, volume 277. Springer Science & Business Media, 2012.

11 Kurt Gödel. Eine Interpretation des Intuitionistischen Aussagenkalküls. Ergebnisse eines Mathematischen Kolloquiums, 4: 39–40, 1933. english translation, with an introductory note by A.S. Troelstra. *Kurt Gödel, Collected Works*, 1:296–303, 1986.

12 Robert Goldblatt. Mathematical modal logic: A view of its evolution. In *Handbook of the History of Logic*, volume 7, pages 1–98. Elsevier, 2006.

13 Raul Hakli and Sara Negri. Does the deduction theorem fail for modal logic? *Synthese*, 187(3):849–867, 2012.

14 John Harrison. The HOL Light Theorem Prover. Web site: https://github.com/jrh13/hol-light.

15 John Harrison. HOL Light tutorial. http://www.cl.cam.ac.uk/~jrh13/hol-light/tutorial.pdf, 2017.

16 David Hilbert and Paul Bernays. *Grundlagen der Mathematik, Vol. I (1934), Vol II (1939)*. Berlin: Springer, 1939.

17 Joachim Lambek and Philip J. Scott. *Introduction to higher-order categorical logic*, volume 7. Cambridge University Press, 1988.

18 Julius Michaelis and Tobias Nipkow. Formalized proof systems for propositional logic. In A. Abel, F. Nordvall Forsberg, and A. Kaposi, editors, *23rd Int. Conf. Types for Proofs and Programs (TYPES 2017)*, volume 104 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

19 Sally Popkorn. *First steps in modal logic*. Cambridge University Press, 1994.

20 Natarajan Shankar. Towards mechanical metamathematics. *Journal of Automated Reasoning*, 1(4):407–434, 1985.

21 Robert M. Solovay. Provability interpretations of modal logic. *Israel journal of mathematics*, 25(3-4):287–304, 1976.

22 Yiming Xu and Michael Norrish. Mechanised modal model theory. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning*, pages 518–533, Cham, 2020. Springer International Publishing.