Journal of Automated Reasoning (2023) 67:29
https://doi.org/10.1007/s10817-023-09677-z

# Mechanising Gödel–Löb Provability Logic in HOL Light

**Marco Maggesi**[1] · **Cosimo Perini Brogi**[2]

Received: 7 May 2022 / Accepted: 17 July 2023 / Published online: 29 August 2023


**Abstract**
We introduce our implementation in HOL Light of the metatheory for Gödel–Löb provability logic (GL), covering soundness and completeness w.r.t. possible world semantics and featuring a prototype of a theorem prover for GL itself. The strategy we develop here to formalise the modal completeness proof overcomes the technical difficulty due to the non-compactness of GL and is an adaptation—according to the formal language and tools at hand—of the proof given in George Boolos' 1995 monograph. Our theorem prover for GL relies then on this formalisation, is implemented as a tactic of HOL Light that mimics the proof search in the labelled sequent calculus G3KGL, and works as a decision algorithm for the provability logic: if the algorithm positively terminates, the tactic succeeds in producing a HOL Light theorem stating that the input formula is a theorem of GL; if the algorithm negatively terminates, the tactic extracts a model falsifying the input formula. We discuss our code for the formal proof of modal completeness and the design of our proof search algorithm. Furthermore, we propose some examples of the latter's interactive and automated use.

**Keywords** Provability logic · Higher-order logic · Mechanised mathematics · HOL Light theorem prover

## 1 Introduction

The origin of provability logic dates back to a short paper by Gödel [28] where propositions about provability are formalised through a unary operator `B` to give a classical reading of intuitionistic logic. 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. The problem has been settled since the 1970s for many formal systems of arithmetic employing Gödel–Löb logic GL: the abstract properties of the formal provability predicate of any

✉ Marco Maggesi
marco.maggesi@unifi.it

✉ Cosimo Perini Brogi
cosimo.perinibrogi@imtlucca.it

[1] Department of Mathematics and Computer Science, University of Florence, Tuscany, Italy

[2] SySMA Research Unit, IMT School for Advanced Studies Lucca, Tuscany, Italy

$\Sigma_1$-sound arithmetical theory $T$ extending $\mathsf{I}\Sigma_1$ [76] are captured by that modal system, as established by Solovay [73].

Solovay's technique 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.[1]

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.[2]

Our work starts then 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 soundness of the system w.r.t. irreflexive transitive finite frames.

A more mathematical part then follows: our goal has been proving formally

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

Since GL is not compact, the standard methodology based on canonical models [69] cannot be directly applied here. After Kozen and Parikh [43], it is common to restrict the construction to a finite subformula universe.[3] The same idea is basically used in Boolos [8] to prove modal completeness for GL. Proceeding in that same way, we have 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 often only proof-sketched in wide-adopted textbooks in logic—including [8]—for they mainly involve "standard" reasoning *within* the proof system we are dealing with. Nevertheless, when working in a formal setting, as we did with HOL Light, we need to split down the main goal into several subgoals, dealing with both the object- and the meta-level. Sometimes, the HOL Light automation does mirror—and, using automation mechanisms, simplify details of—the informal reasoning. On other occasions, we have to modify some aspects of the proof strategies, simplifying, through the computer tools at hand, some passages of the informal argument in Boolos [8, Chap. 5].

As it is known, for any logical calculus, a completeness result w.r.t. finite models—aka finite model property—implies the decidability of that very logic [36]. Therefore, the formal proof for Theorem 1 we discuss in the first part of the present work could be used, in principle, to develop a decision algorithm for $\mathbb{GL}$. That would be a valuable tool for automating in HOL Light the proof of theorems of GL.

Proof search in axiomatic calculi is a challenging task. In our formalisation, we had to develop several proofs in the axiomatic calculus for $\mathbb{GL}$, and only in a few cases, it has been possible to leave the proof search to the automation mechanism of HOL Light.

By having a formal proof of the finite model property, one could hope to solve the goal of checking whether a formula is a theorem of GL by shifting from the syntactic problem of finding a proof of that formula in the axiomatic calculus to the semantic problem of checking the validity of that formula in any finite model by applying automated first-order reasoning as implemented in the proof assistant.

[1] In contemporary research, this is still the main strategy to prove arithmetical completeness for other modalities for provability and related concepts, particularly for interpretability logics.

[2] Some of them are closely related to the field of proof theory; others point at developing a uniform proof-strategy to establish adequate semantics in formal theories of arithmetic having different strengths and flavours. The reader is referred to Beklemishev and Visser [5] for a survey of open problems in provability logics.

[3] We are grateful to an anonymous referee for pointing us to that work.

Such a strategy has many shortcomings, unfortunately. We document some of its limits in Sect. 4.4 and at the beginning of the subsequent section, but the main points are the following: From the complexity theory viewpoint, a decision procedure based on the countermodel construction would be far from being optimal, belonging to EXPSPACE, rather than PSPACE, to which decidability of GL belongs [46];[4] from the practical viewpoint, implementing it in HOL Light would consists in a relatively simple formalisation exercise.

Structural proof theory for modal logics—briefly summarised in Sect. 5.1—suggests a more promising strategy.

Many contemporary sequent calculi for non-classical logics are based on an "internalisation" of possible world semantics in Gentzen's original formalism [58, 67].

We resort to an explicit internalisation for developing our decision algorithm since our formalisation in HOL Light of Kripke semantics for GL is *per se* a labelling technique in disguise.

Therefore, in the second part of this paper, we introduce what might be considered a shallow embedding in HOL Light of Negri's labelled sequent calculus G3KGL [53, 54].

To state it clearly: while in the first part, we defined the axiomatic system $\mathbb{GL}$ within our proof assistant employing an inductive definition of the derivability relation for $\mathbb{GL}$, in the second part, we define *new tactics* of HOL Light in order to perform a proof search in G3KGL by using the automation infrastructure provided by HOL Light itself.

By relying on the meta-theory for G3KGL developed in [54], we can safely claim that such an embedding provides a decision algorithm for GL: if proof search terminates on a modal formula given as input, HOL Light produces a new theorem, stating that the input formula is a theorem of GL; otherwise, our algorithm prints all the information necessary to construct an appropriate countermodel for the input formula.

Our code is integrated into the current HOL Light distribution and freely available from [35]. The files we will occasionally refer to during the presentation are freely accessible from there.

The present paper is structured as follows:

- In Sect. 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 uniformly, i.e. without the restriction to the specific modal system we are interested in. The formalisation constitutes a large part of the file `modal.ml`;
- In Sect. 3, we formally define the axiomatic calculus $\mathbb{GL}$ and prove neatly 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 the contents of our code in `gl.ml`;
- In Sect. 4 we give our formal proof of modal completeness of $\mathbb{GL}$, starting with the definition of maximal consistent *lists* of formulas. In order to prove their syntactic properties—and, in particular, the extension lemma for consistent lists of formulas 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.
  At the end of the section, we give the formal definition of bisimilarity for our setup, and we prove the associated bisimulation theorem [69, Chap. 11]. Our notion of bisimilarity is polymorphic because it can relate classes of frames sitting on different types.

[4] Refer to Sect. 4.4 for the explicit bound one can obtain for the complexity of such a decision procedure.

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—this way obtaining the finite model property for GL w.r.t. frames having finite *sets* of formulas as possible worlds, as in the standard Henkin's construction. These results conclude the first part of the paper and are gathered in the file `completeness.ml`.

- Section 5 opens the part of the paper describing our original theorem prover for GL. We collect some basic notions and techniques for proof-theoretic investigations on modal logic. In particular, we recall the methodology of explicit internalisation of relational semantics and the results that provide the meta-theory for our decision algorithm.
- Finally, in Sect. 6, we describe our implementation of the labelled sequent calculus G3KGL—documented by the file `decid.ml`—to give a decision procedure for GL. We recover the formalisation of Kripke semantics for GL presented in Sect. 2 to define new tactics mimicking the rules for G3KGL. Then, we properly define our decision algorithm by designing a specific terminating proof search strategy in the labelled sequent calculus. This way, we extend the HOL Light automation toolbox with an "internal" theorem prover for GL that can also produce a countermodel for any formula for which proof search fails. We propose some hands-on examples of use by considering modal principles that have a certain relevance for meta-mathematical investigations; the interested reader will find further examples in the file `tests.ml`.
- Section 7 closes the paper and compares our results with related works on mechanised modal logic.

Our formalisation does not modify 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 endowed with the same automation toolbox.

***Revision notes.***
This article is an expanded version of the conference paper [48], presented at Interactive Theorem Proving (ITP) 2021. Changes include adding Sects. 5 and 6 on our implementation in HOL Light of the theorem prover and countermodel constructor for GL, and some minor local revisions. An intermediate version of the contents discussed in the present paper appeared in [65]. Both authors wish to thank two anonymous referees for their valuable feedback and comments towards improving the presentation of the material we discuss here.

### 1.1 HOL Light Notation

The HOL Light proof assistant [35] 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 [47], 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 (hopefully) simpler subgoals so that the interactive aspect of proving is highlighted. Proof scripts can then be constructed using *tacticals* that compact the proof into a few lines of code evaluated by the machine.

In what follows, we will report several code fragments to give the flavour of our development and to provide additional documentation and information to the reader interested in

the technical details. We partially edited the code to ease the reading of the mathematical formulas. For instance, we replaced the purely ASCII notations of HOL Light with the usual graphical notation. Every source code snippet in this paper has a link—indicated by the word "(sources)" on the right—to a copy of the source files stored on the Software Heritage[5] archive which is helpful to those who want to see the raw code and how it fits in its original context. Here is an example:

```
ADD_SYM                                                    (sources)
  ⊢ ∀m n. m + n = n + m
```

Note that we report theorems with their associated name (the name of its associated OCaml constant), and we write their statement prefixed with the turnstile symbol (⊢). In the expository style, we omit formal proofs, but the meaning of definitions, lemmas, and theorems in natural language is clear.

The HOL Light printing mechanism omits type information completely. Therefore, we warn the reader about types when it might be helpful, or even indispensable, to avoid ambiguity—including the case of our main results, `COMPLETENESS_THEOREM` and `COMPLETENESS_THEOREM_GEN`.

We also recall that a Boolean function `s : α -> bool` is also called a *set on* $\alpha$ 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`.

As mentioned, our contribution is part of the HOL Light distribution. The reader interested in performing these results on her machine—and perhaps building further formalisation 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, we deal with a logic that extends classical propositional reasoning using a single modal operator 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.

From a foundational perspective, we want to keep everything neat and clean; therefore, we will define *both* the object-language and its interpretation with no relation to the HOL Light environment. In other terms: our formulas and operators are *real* syntactic objects which we keep distinct from their semantic counterparts—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 $\mathcal{L}$ 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 add a modal unary

[5] https://www.softwareheritage.org/.

connective $\Box$. The starting point is, as usual, a denumerable infinite set of propositional atoms $a_0, a_1, \cdots$. Accordingly, formulas of this language will have one of the following forms

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

The following code extends the HOL system with an **inductive type of formulas** generated by the above syntactic constructions

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

Later in the code, the operators `&&`, `||`, `- ->`, `<->` are used infixed as usual. Next, we turn to semantics for our modal language. We use **relational models**—aka Kripke models.[6]

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 an evaluation 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   (sources)
  `(holds f V False (w:W) ⟺ ⊥) ∧
   (holds f V True w ⟺ ⊤) ∧
   (holds f V (Atom s) w ⟺ V s w) ∧
   (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 (□ p) w ⟺
    ∀u. u ∈ 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:W->bool, R:W->W->bool)` of a set of worlds and an accessibility relation—and `V:string->W->bool` 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                         (sources)
  `holds_in (W,R) p ⟺
   ∀V w. w ∈ W ⟹ holds (W,R) V p w`;;
```

[6] See Copeland [10] for the historical development of this notion.

```
let valid = new_definition                                    (sources)
  `L |= p <=> ∀f. L f ==> holds_in f p`;;
```

The above formalisation is essentially presented in Harrison's HOL Light Tutorial [34, § 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. However, in the next section, when we define the classes of frames, we are dealing with for GL, the requirement on `W` is correctly integrated with the corresponding types.

### 2.2 Frames for GL

For carrying out our formalisation, 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: then, 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, in order 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:[7]

```
let TRANSNT = new_definition                                  (sources)
  `TRANSNT (W:W->bool,R:W->W->bool) <=>
   ¬(W = {}) ∧
   (∀x y:W. R x y ==> x ∈ W ∧ y ∈ W) ∧
   (∀x y z:W. x ∈ W ∧ y ∈ W ∧ z ∈ W ∧ R x y ∧ R y z
              ==> R x z) ∧
   WF(λx y. R y x)`;;
```

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                                                (sources)
  ⊢ ∀W:W->bool R:W->W->bool.
      (∀x y:W. R x y ==> x ∈ W ∧ y ∈ W)
      ==> ((∀x y z. x ∈ W ∧ y ∈ W ∧ z ∈ W ∧ R x y ∧ R y z
                    ==> R x z) ∧
           WF (λx 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 [8, Theorem 10] and in [69, 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 [34, § 20.3].

[7] We warn the reader about a potentially misleading notation. In the following statement, two interrelated mathematical objects occur, both denoted by `W` for convenience: one is the type `W`, and the other is the set `W` on the former—in a sense explained in the introduction about the HOL Light notation.

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, e.g., for mechanically checking whether it does provide a countermodel for a formula of our logic. In fact, when aiming at a completeness theorem, one wants to consider models that are helpful for establishing further properties of the same logic. In the present case, the decidability of GL, which, as for any other normal modal logic, is a straightforward corollary of the *finite model property* [69, Chap. 13].

The frames we want to investigate are then those whose $W$ is **finite**, and whose $R$ is both **irreflexive** and **transitive**:

```
let ITF = new_definition                                    (sources)
  `ITF (W:W->bool,R:W->W->bool) ⟺
    ¬(W = ∅) ∧
   (∀x y:W. R x y ⟹ x ∈ W ∧ y ∈ W) ∧
   FINITE W ∧
   (∀x. x ∈ W ⟹ ¬ R x x) ∧
   (∀x y z. x ∈ W ∧ y ∈ W ∧ z ∈ 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                                                      (sources)
  ⊢ ∀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 and only 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.[8]

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 deal 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.

Therefore, to the 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: $\dfrac{A}{\Box A}\ \mathsf{NR}$,

[8] Notice that the lemma `ITF_NT` allows us to derive the former result as a corollary of the latter.

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

```
let GLaxiom_RULES,GLaxiom_INDUCT,GLaxiom_CASES =  (sources)
  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 $\vdash$), is also inductively defined in the expected way:[9]

```
let GLproves_RULES,GLproves_INDUCT,GLproves_CASES =
                                                    (sources)
  new_inductive_definition
  `(∀p. GLaxiom p ⟹ |-- p) ∧
   (∀p q. |-- (p --> q) ∧ |-- p ⟹ |-- q) ∧
   (∀p. |-- p ⟹ |-- (Box p))`;;
```

### 3.1 Soundness Lemma

We can now 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 result is obtained by simply unfolding the relevant definitions and applying theorems `TRANSNT_EQ_LOB` and `ITF_NT` of Sect. 2.2:

```
GL_TRANSNT_VALID                                    (sources)
  ⊢ ∀p. (|-- p) ⟹ TRANSNT |= p

GL_ITF_VALID                                        (sources)
  ⊢ ∀p. |-- p ⟹ ITF |= p
```

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

[9] Small modifications to limit the application of NR on the definition of `|- -` would introduce the notion of derivability from a set of assumptions, so that the deduction theorem would hold [32].

```
GL_consistent                                                    (sources)
  ⊢ ¬(|-- False)
```

We are now ready to consider the mechanised proof of completeness for the calculus w.r.t. this very class of frames.

### 3.2 $\mathbb{GL}$-Lemmas

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

Following 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 [35]—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 formalised proofs of those lemmas *in* $\mathbb{GL}$ that are useful for the formalised results we present in the next section. This file might be considered a "kernel" for further experiments in reasoning about axiomatic calculi using HOL Light. The lemmas we proved are, indeed, standard tautologies of classical propositional logic, along with specific theorems of minimal modal logic and its extension for transitive frames—i.e. of the systems $\mathbb{K}$ and $\mathbb{K}4$ [69]—, so that by applying minor changes in basic definitions, they are—so to speak—take-away proof scripts for extensions of that very minimal system within the realm of normal modal logics.

Whenever it was useful, we have also given a characterisation of classical operators in terms of an implicit (i.e. internal) deduction expressed by the connective `- ->`. When this internal deduction is from an empty set of assumptions, we named the HOL theorem with the suffix `_th`, and stated the deduction as a $\mathbb{GL}$-lemma, such as

```
GL_modusponens_th                                                (sources)
  ⊢ ∀p q. |-- ((p --> q) && p --> q)
```

Moreover, we introduced some derived rules of the axiomatic system mimicking the behaviour in Gentzen's formalism of classical connectives, as in e.g.

```
GL_and_elim                                                      (sources)
  ⊢ ∀p q r. |-- (r --> p && q)
            ⟹ |-- (r --> q) ∧ |-- (r --> p)
```

We had to prove about 120 such results of varying degrees of difficulty. We believe that this piece of code 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.

On the one hand, these lemmas simplify the subsequent formal proofs involving consistent lists of formulas since they let us work formally within the scope of `|- -` 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, 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 a notion of *theoremhood-as-tautology* for a specific logic, but its lack of naturalness w.r.t. the practice of developing proper derivations makes it an unsatisfactory model for structural aspects 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, even when dealing with trivial propositions.

We, therefore, left the proof search to the HOL Light toolbox as much as possible. Unfortunately, we have to express mixed feelings about the general experience. In most cases, relying on this specific proof assistant's automation tools did save our time and resources when trying to give a formal proof in $\mathbb{GL}$. Nevertheless, those automation tools did not turn out to be helpful at all in proving some $\mathbb{GL}$-lemmas. In those cases, we had to search for the specific instances of axioms from which deriving the lemmas,[10] so that interactive proving them had advantages as well as traditional instruments of everyday mathematicians.

To stress the general point: it is possible—and valuable 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 mechanising (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 particularly satisfactory for pursuing investigations on derivability within axiomatic systems.

## 4 Modal Completeness

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 approach can be summarised as a standard construction of countermodels made of maximal consistent sets of formulas and an appropriate accessibility relation, according to e.g. the textbooks [69] and [8].

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 formulas. We cannot, therefore, reduce the semantic notion of (in)coherent set of formulas to the syntactic one of (in)consistent set of formulas: when extending a consistent set of formulas to a maximal consistent one, we might end up with a *syntactically* consistent set that nevertheless cannot be *semantically* satisfied.

Despite this, it is possible to achieve a completeness result by

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

That is the fundamental idea behind the proof in Boolos' monograph [8, Chap. 5]. In that presentation, however, constructing a maximal consistent set from a simply consistent one is only proof-sketched and relies on a syntactic manipulation of formulas. By using HOL Light, we succeed in giving a detailed proof of completeness as directly as that by Boolos. Moreover, 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

[10] 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 formulas of a schema to apply to the goal in order to rewrite it.

the symbolic subtleties sketched in [8] that have the unpleasant consequence of making the formalised proof 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 other normal (mono)modal logics—we only need to modify the subsequent definition of `GL_STANDARD_REL` according to the specific system under consideration. Thus, we provide a way for formally establishing completeness á la Henkin *and* the finite model property without resorting to filtrations [69] of canonical models for those systems.

### 4.1 Maximal Consistent Lists

Following the standard practice, we need to consider consistent finite sets of formulas for our proof of completeness. In principle, we can employ general sets of formulas in the formalisation. However, from the practical viewpoint, 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 formulas in a list:[11]

```
let CONJLIST =                                                (sources)
  new_recursive_definition list_RECURSION
  `CONJLIST [] = True ∧
   (∀p X. CONJLIST (CONS p X) =
          if X = [] then p else p && CONJLIST X)`;;
```

We prove some properties on lists of formulas and some $\mathbb{GL}$-lemmas involving `CONJLIST`. These properties and lemma allow us to define the notion of **consistent list of formulas** and prove the main properties of this kind of objects:

```
let CONSISTENT = new_definition                               (sources)
  `CONSISTENT (l:form list) ⟺ ¬(|-- (Not (CONJLIST l)))`;;
```

In particular, we prove that:

- a consistent list cannot contain both `p` and `Not p` for any formula `p`, nor `False`;
- for any consistent list `X` and formula `p`, either `X + p` is consistent, or `X + Not p` is consistent, where + denotes the usual operation of appending an element to a list.

Our **maximal consistent lists** w.r.t. a given formula `p` will be consistent lists that do not contain repetitions and that contain, for any subformula of `p`, that very subformula or its negation:[12]

```
let MAXIMAL_CONSISTENT = new_definition                       (sources)
  `MAXIMAL_CONSISTENT p X ⟺
   CONSISTENT X ∧ NOREPETITION X ∧
   (∀q. q SUBFORMULA p ⟹ MEM q X ∨ MEM (Not q) X)`;;
```

[11] Notice that in 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.

[12] Here we define the set of subformulas of `p` as the reflexive, transitive closure of the set of formulas on which the main connective of `p` operates: this way, the definition is simplified, and it is easier to establish standard properties of the set of subformulas employing general higher-order lemmas in HOL Light for the closure of a given relation.

where X is a list of formulas and MEM q X is the membership relation for lists. We then establish the main closure property (MAXIMAL_CONSISTENT_LEMMA) of maximal consistent lists, namely closure under modus ponens (sources).

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—we can finally define the ideal (type of counter)model we are interested in.

We define first the relation of *subsentences* as

```
let SUBSENTENCE_RULES,SUBSENTENCE_INDUCT, (sources)
  SUBSENTENCE_CASES = new_inductive_definition
  `(∀p q. p SUBFORMULA q ⟹ p SUBSENTENCE q) ∧
   (∀p q. p SUBFORMULA q ⟹ Not p SUBSENTENCE q)`;;
```

Next, given a formula p, we take as "standard"—and define the class GL_STANDARD_MODEL—those models consisting of:

C1. the set of maximal consistent lists w.r.t. p made of *subsentences* of p—i.e. its subformulas or their negations—as possible worlds;

C2. an accessibility relation R such that

   a. it is irreflexive and transitive, and
   b. 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;

C3. an atomic evaluation that gives value T (true) to a in w iff a is a subformula of p.

The corresponding code is the following:

```
  let GL_STANDARD_FRAME = new_definition              (sources)
   `GL_STANDARD_FRAME p (W,R) ⟺
(*C1.  *) W = {w | MAXIMAL_CONSISTENT p w ∧
                   (∀q. MEM q w ⟹ q SUBSENTENCE p)} ∧
(*C2.a.*) ITF (W,R) ∧
(*C2.b.*) (∀q w. Box q SUBFORMULA p ∧ w ∈ W
            ⟹ (MEM (Box q) w ⟺ ∀x. R w x ⟹ MEM q x))`;;

  let GL_STANDARD_MODEL = new_definition              (sources)
    `GL_STANDARD_MODEL p (W,R) V ⟺
     GL_STANDARD_FRAME p (W,R) ∧
(*C3.*)(∀a w. w ∈ W
        ⟹ (V a w ⟺ MEM (Atom a) w ∧ Atom a SUBFORMULA p))`;;
```

Notice that the conditions C1, C2.b and C3 are very general and do not relate to the logic under consideration: they constitute a minimal set of conditions required for normal modal systems; on the contrary, the condition C2.a is specific to GL, and needs to be properly mirrrored at the syntactic level. That is the role played by the last two lines of the definition GL_STANDARD_REL discussed at the end of the next section.

### 4.2 Maximal Extensions

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

Our original strategy differs from the presentation in e.g. [8] for being closer to the standard Lindenbaum construction commonly used to prove completeness results. By doing so, we have been able to circumvent both many technicalities in formalising the combinatorial argument sketched by Boolos in [8, p. 79] *and* the problem – 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 `p`, we can construct a maximal consistent list of subsentences of `p` by extending (if necessary) `X`:

```
EXTEND_MAXIMAL_CONSISTENT                                    (sources)
  ⊢ ∀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
```

where `X SUBLIST M` denotes that each element of the list `X` is an elment of the list `M`.

The proof sketch is as follows: given a formula `p`, we proceed in a step-by-step construction by iterating over the subformulas `q` of `p` not contained in `X`. At each step, we append to the list `X` the subformula `q`—if the resulting list is consistent—or its negation `Not q`—otherwise.

This way, 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 `p` is not a $\mathbb{GL}$-lemma—the set of possible worlds in `GL_STANDARD_FRAME` w.r.t. `p` is non-empty, as required by the definition of relational structures:

```
NONEMPTY_MAXIMAL_CONSISTENT                                  (sources)
  ⊢ ∀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 conditions C2.a and C2b for a `GL_STANDARD_FRAME`; the following does the job:

```
let GL_STANDARD_REL = new_definition                         (sources)
  `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)`;;
```

Notice that the last two lines of this definition assure that the condition C2.a is satisfied, i.e., that we considering irreflexive transitive frames. Condition C2.b also needs to be satisfied: our `ACCESSIBILITY_LEMMA` assures that[13]

[13] Notice that we only need to prove the right-to-left direction of condition C2.b, since the converse is given by the second last requirement in the definition `GL_STANDARD_FRAMES`. We can formally prove the former by reasoning within $\mathbb{GL}$, and using, in particular, the scheme GL and the derivability of $\Box p \to \Box p \wedge \Box\Box p$ (`GL_dot_box`).

```
ACCESSIBILITY_LEMMA                                          (sources)
  ⊢ ∀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,
```

Such a "standard" accessibility relation (together with the set of the specific maximal consistent lists we are dealing with) defines then a structure in `ITF` with the required properties in order to satisfy the relation `GL_STANDARD_FRAME`:

```
ITF_MAXIMAL_CONSISTENT                                       (sources)
  ⊢ ∀p. ¬(|-- p)
        ⟹ ITF ({M  MAXIMAL_CONSISTENT p M ∧
                   (∀q. MEM q M ⟹ q SUBSENTENCE p)},
               GL_STANDARD_REL p),
```

Notice that we might easily modify the formal proofs of conditions C1, C2.b and C3 when dealing with different axiomatic systems, e.g. $\mathbb{K}$, $\mathbb{K}4$, $\mathbb{T}$, $\mathbb{S}4$, $\mathbb{B}$, $\mathbb{S}5$, as it happens at the informal level in Boolos [8]. In fact, for each of each systems, we would only have to modify the definition of the "standard relation" (in particular, the last two lines of our `GL_STANDARD_REL`) and the parts of code in the proof of the accessibility lemma where we need to reason within the specific axiomatic calculus under investigation. For each of these additional logics, a semantic condition on standard frames (line 4 of the definition `GL_STANDARD_FRAME`) would then been defined formalising what [8, Chap. 5] reports on the topic.

### 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 `q` of `p`—that if $\mathbb{GL} \nvdash$ `p`, then for any world `w` of the standard model, `q` holds in `w` iff `q` is member of `w`:[14]

```
GL_truth_lemma                                               (sources)
  ⊢ ∀W R p V q.
      ¬(|-- p) ∧
      GL_STANDARD_MODEL p (W,R) V ∧
      q SUBFORMULA p
      ⟹ ∀w. w ∈ W ⟹ (MEM q w ⟺ holds (W,R) V q w),
```

Finally, we can prove the main result: if $\mathbb{GL} \nvdash$ `p`, then the list `[Not p]` is consistent, and by applying `EXTEND_MAXIMAL_CONSISTENT`, we obtain a maximal consistent list `X` w.r.t. `p`

[14] As before, this formal proof can be adapted to the other six modal systems mentioned at the end of the previous section.

that extends it, so that, by applying `GL_truth_lemma`, we have that `X ⊮ p` in our standard model. The corresponding formal proof reduces to the application of those previous results and the appropriate instantiations:

```
COMPLETENESS_THEOREM                                          (sources)
  ⊢ ∀p. ITF |= 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`: the explicit type annotation would be

```
ITF : (form list->bool)#(form list->form list->bool)->bool
```

This is not an intrinsic limitation: the next section is devoted to generalising this theorem to frames on an arbitrary domain.

### 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`. This result would suffice to provide a (very non-optimal) bound on the complexity of $\mathbb{GL}$ : Testing the validity of a formula of size $n$ requires considering all models with cardinality $k$, for any $k \leq 2^n!$.

The same completeness result must also hold when considering 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 through the notion of *bisimulation* [75].

In our context, given two frames `(W1,R1)` and `(W2,R2)` sitting respectively on types `:A` and `:B`, each with an evaluation 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                                                (sources)
  ⊢ BISIMIMULATION (W1,R1,V1) (W2,R2,V2) Z ⟺
    (∀w1:A w2:B.
     Z w1 w2
     ⟹ w1 ∈ W1 ∧ w2 ∈ W2 ∧
          (∀a:string. V1 a w1 ⟺ V2 a w2) ∧
          (∀w1'. R1 w1 w1'
                 ⟹ ∃w2'. w2' ∈ W2 ∧ Z w1' w2'
                 ∧ R2 w2 w2') ∧
          (∀w2'. R2 w2 w2'
                 ⟹ ∃w1'. w1' ∈ 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                                (sources)
  `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                                                    (sources)
  ⊢ ∀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 bisimilarity preserves validity. The precise statements are the following:

```
BISIMILAR_HOLDS_IN                                                 (sources)
  ⊢ ∀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                                                    (sources)
  ⊢ ∀L1 L2.
      (∀W1 R1 V1 w1:A.
        L1 (W1,R1) ∧ w1 ∈ 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                                           (sources)
  ⊢ ∀p. INFINITE (:A) ∧ ITF:(A->bool)#(A->A->bool)->bool |= p
        ⟹ |-- p
```

Furthermore, from the proof that the relation

```
λw1 w2. MAXIMAL_CONSISTENT p w1 ∧
        (∀q. MEM q w1 ⟹ q SUBSENTENCE p) ∧
        w2 ∈ GL_STDWORLDS p ∧
        set_of_list w1 = w2
```

defines a bisimulation between the `ITF`-standard model based on maximal consistent *lists* of formulas and the model based on corresponding *sets* of formulas, we obtain the traditional version of modal completeness, corresponding to theorem `GL_COUNTERMODEL_FINITE_SETS` (sources) in our code. That, in turn, would enhance the complexity measure of derivability in $\mathbb{GL}$ as one would expect: now, in order to check whether a formula with size $n$ is a theorem of Gödel–Löb logic, one may forget about the order of the subsentences, and consider all `ITF`-models with cardinality $k$, for any $k \leq 2^n$.

## 5 Decidability via Proof Theory

By using our EXTEND_MAXIMAL_CONSISTENT lemma, we succeeded in giving a rather neat mechanised proof of completeness w.r.t. relational semantics for $\mathbb{GL}$.[15] Since the relational countermodel we construct is finite, we can safely claim[16] that the finite model property holds for $\mathbb{GL}$.

As an immediate corollary, we have that theoremhood in $\mathbb{GL}$ is decidable, and, in principle, we could implement a decision procedure for that logic in OCaml. Our theorem GL_COUNTERMODEL_FINITE_SETS would also provide an explicit bound on the complexity of that task.

A naive approach to the implementation would proceed as follows.

We define the tactic NAIVE_GL_TAC and its associated rule NAIVE_GL_RULE that perform the following steps:

1. Apply the completeness theorem w.r.t. finite sets;
2. Unfold some definitions;
3. Try to solve the resulting *semantic* problem using first-order reasoning.

Here is the corresponding code.

```
let NAIVE_GL_TAC : tactic = (sources)
  MATCH_MP_TAC GL_COUNTERMODEL_FINITE_SETS THEN
  REWRITE_TAC[valid; FORALL_PAIR_THM; holds_in; holds;
              ITF; GSYM MEMBER_NOT_EMPTY] THEN
  MESON_TAC[];;

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

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

```
GL_box_iff_th                                          (sources)
  ⊢ ∀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 script consists of ten lines of code. On the contrary, our rule is able to check it in a few steps:

```
# NAIVE_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))
```

[15] For the completeness w.r.t. transitive Noetherian frames, it is common—see the textbooks [8, 69]—to reason on irreflexive transitive finite structures and derive the result as a corollary of completeness w.r.t. the latter class.

[16] After Harrop [36].

In spite of this, the automation offered by `MESON` tactic is often unsuccessful, even on trivial lemmas. For instance, the previous procedure is not even able to prove the basic instance Gödel-Löb scheme

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

This suggests that `NAIVE_GL_TAC` is based on a very inadequate strategy.

A better approach would consist of introducing an OCaml bound function on the size of the frames on which the validity of a formula $A$ has to be checked, according to the cardinality remarks we made at the end of Sect. 4.4.

That is, for sure, a doable way to solve the task, but we were looking for a more principled and modern approach.

That is where structural proof theory has come to the rescue.

### 5.1 Bits of Structural Proof Theory

In very abstract terms, a deductive system consists of a set of starting formal expressions together with inference rules. Its principal aim is to find proofs of valid expressions w.r.t. a given logic L. A proof (or derivation) in a deductive system is obtained by application of the inference rules to starting expressions, followed by further application of the inference rules to the conclusion, and so on, recursively. A theorem (or lemma) in such a system is the formal expression obtained after a finite run of the procedure just sketched.

Such a definition captures the system $\mathbb{GL}$, and the derivability relation formalised in HOL Light by the predicate `|- -` of Sect. 3. Our previous remarks on $\mathbb{GL}$-lemmas make explicit that this kind of deductive system has shortcomings that its computerisation per se cannot overcome.

The proof-theoretic paradigm behind $\mathbb{GL}$ and, more generally, axiomatic calculi could be called *synthetic*: proof search in such systems is not guided by the components of the formula one wishes to prove. The human prover and the proof assistant dealing with a formal derivability relation have to guess both the correct instances of the axiom schemas and the correct application order of inference rules required in the proof.

A better paradigm is provided by Gentzen's sequent calculi, introduced first in [22, 23]. That work marks the advent of structural proof theory and the definite shift from investigations in synthetic deductive systems to *analytic* ones. Gentzen's original calculi have been further refined by Ketonen [41] and Kleene [42] into the so-called G3-style systems.

In those systems, a **sequent** is a formal expression with shape

$$\Gamma \Rightarrow \Delta\,,$$

where $\Gamma$, $\Delta$ are finite multisets—i.e. finite lists modulo permutations—of formulas of a given language. The symbol $\Rightarrow$ reflects the deducibility relation at the meta-level in the object language. $\Gamma$ is called the **antecedent** of the sequent; $\Delta$ is its **consequent**.

A derivation in a G3-style sequent calculus is a finite rooted tree labelled with sequents such that:

- its leaves are labelled by **initial sequents** (the starting formal expressions of the abstract deductive system);
- its intermediate nodes are labelled by sequents obtained from the sequent(s) labelling the node(s) directly above by a correct application of an inference rule of the calculus;
- its root is the conclusion of the derivation, and it is called the **end-sequent**.

**Initial sequents:**

$$p, \Gamma \Rightarrow \Delta, p$$

**Propositional rules:**

$$\frac{}{\bot, \Gamma \Rightarrow \Delta}\ \mathcal{L}\bot$$

$$\frac{A, B, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\wedge \qquad \frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B}\ \mathcal{R}\wedge$$

$$\frac{A, \Gamma \Rightarrow \Delta \qquad B, \Gamma \Rightarrow \Delta}{A \vee B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\vee \qquad \frac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow \Delta, A \vee B}\ \mathcal{R}\vee$$

$$\frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\neg \qquad \frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}\ \mathcal{R}\neg$$

$$\frac{\Gamma \Rightarrow \Delta, A \qquad B, \Gamma \Rightarrow \Delta}{A \to B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\to \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \to B}\ \mathcal{R}\to$$

**Fig. 1** Rules of the calculus G3cp

Figure 1 summarises the calculus G3cp for classical propositional logic. For each rule, one distinguishes:

- its **main formula**, which is the formula occurring in the conclusion and containing the logical connective naming the rule;
- its **active formulas**, which are the formulas occurring in the premise(s) of the rule;
- its **context**, which consists of the formulas occurring in the premise(s) *and* the conclusion, untouched by the rule.

G3-style systems are the best available option for (efficiently) automating decision procedures: once an adequate—i.e. sound and complete—G3-calculus for a given logic L has been defined, in order to decide whether a formula is a theorem of L it suffices to start a *root-first* proof search of that exact formula in the related G3-calculus.

This is so because, by design, good G3-style systems satisfy the following desiderata:

1. **Analyticity:** Each formula occurring in a derivation is a subformula of the formulas occurring lower in the derivation branch. This means that *no guesses are required* to the prover when developing a formal proof in the G3-calculus;
2. **Invertibility of all the rules:** For each rule of the system, the derivability of the conclusion implies the derivability of the premise(s). *This property avoids backtracking* on the proof search. It also means that at each step of the proof search procedure *no bit of information gets lost*, so that the construction of *one* derivation tree is enough to decide the derivability of a sequent;

3. **Termination:** Each proof search must come to an end. If the procedure's final state generates a derivation, the end-sequent is a theorem; otherwise, it is generally possible to extract a *refutation* of the sequent from the failed proof search.[17]

Satisfaction of all those desiderata by a pure Gentzen-style sequent calculus has been considered for long almost a mirage for non-classical logics. Times have changed with the advent of *internalisation* techniques of semantic notions in sequent calculi for non-classical logics.

The starting point of that perspective is still the basic G3-paradigm, but the formalism of the sequent system is extended either by

- enriching the language of the calculus itself (**explicit internalisation**); or by
- enriching the structure of sequents (**implicit internalisation**).

The implicit approach adds structural connectives to sequents other than '$\Rightarrow$' and commas.[18] Most G3-style calculi obtained this way provide, in general, very efficient decision procedures for the related logics. However, they are sometimes rather hard to design and might lack an additional and highly valuable desideratum of modularity.

The explicit approach reverses the situation. Explicit internalisation uses specific items to represent semantic elements; the formulas of the basic language are then **labelled** by those items and have shape e.g. $x : A$. That expression formalises the forcing relation we rephrased in Sect. 2.2, and classical propositional rules operate within the scope of labels. Furthermore, to handle modal operators, the antecedent of any sequent may now contain **relational atoms** of shape $xRy$, or any other expression borrowed from the semantics for the logic under investigation.

The rules for the modalities formalise the forcing condition for each modal connective. For instance, the rules in Fig. 2 define the labelled sequent calculus G3K.

Extensions of the minimal normal modal logic K are obtained by rules for relational atoms, formalising the characteristic properties of each specific extension. For instance, the system G3K4 for the logic K4 is defined by adding to G3K the rule

$$\frac{xRz, xRy, yRz, \Gamma \Rightarrow \Delta}{xRy, yRz, \Gamma \Rightarrow \Delta}\ \mathit{Trans}$$

Labelled sequent calculi do satisfy, in general, the basic desiderata of G3-style systems and are rather modular.[19] However, since analyticity holds in a less strict version than the subformula principle mentioned in the previous definition, termination of proof search is sometimes hard to prove and, in many cases, produces complexity results far from optimal.

In the present paper, internalisation is considered only in its explicit version for some syntactic economy: we adopted Negri's labelled sequent calculus [53] to achieve our goal of implementing a decision algorithm for GL.

Before proceeding, let us clarify the methodology behind labelled sequents for normal modal logics.

[17] Notice, however, that sometimes the invertibility of a rule could break termination of the proof search, as witnessed by $\mathcal{L} \rightarrow$ in G3ip for intuitionistic propositional logic [74].

[18] Refer to e.g. Avron[2], Mints [51, 52], and Pottinger [70], as well as Restall [71], Kurokawa [45], Marin and Straßburger [49]. For its direct generalisation by nested sequents refer to e.g. Kashima [40], Brünnler [9], as well as Poggiolesi [66–68] and Olivetti and Pozzato [63].

[19] Their origin dates back to Kanger [39], followed by the refinements in Kripke [44], Fitting [19], and Gabbay [20]. However, labelled sequent calculi have been established as a well-structured methodology after Negri [53]. Extensions, refinements, and further results have since been obtained for many logics. Refer to e.g. [55, 56], Poggiolesi [11, 68].

**Initial sequents:**

$$x : p, \Gamma \Rightarrow \Delta, x : p$$

**Propositional rules:**

$$\frac{}{x : \bot, \Gamma \Rightarrow \Delta}\ \mathcal{L}\bot$$

$$\frac{x : A, x : B, \Gamma \Rightarrow \Delta}{x : A \wedge B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\wedge \qquad \frac{\Gamma \Rightarrow \Delta, x : A \qquad \Gamma \Rightarrow \Delta, x : B}{\Gamma \Rightarrow \Delta, x : A \wedge B}\ \mathcal{R}\wedge$$

$$\frac{x : A, \Gamma \Rightarrow \Delta \qquad x : B, \Gamma \Rightarrow \Delta}{x : A \vee B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\vee \qquad \frac{\Gamma \Rightarrow \Delta, x : A, x : B}{\Gamma \Rightarrow \Delta, x : A \vee B}\ \mathcal{R}\vee$$

$$\frac{\Gamma \Rightarrow \Delta, x : A}{x : \neg A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\neg \qquad \frac{x : A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, x : \neg A}\ \mathcal{R}\neg$$

$$\frac{\Gamma \Rightarrow \Delta, x : A \qquad x : B, \Gamma \Rightarrow \Delta}{x : A \to B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\to \qquad \frac{x : A, \Gamma \Rightarrow \Delta, x : B}{\Gamma \Rightarrow \Delta, x : A \to B}\ \mathcal{R}\to$$

**Modal rules:**

$$\frac{y : A, xRy, x : \Box A, \Gamma \Rightarrow \Delta}{xRy, x : \Box A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\Box \qquad \frac{xRy, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \Box A}\ \mathcal{R}\Box_{(y!)}$$

where the annotation $(y!)$ in $\mathcal{R}\Box$ states that the label $y$ does not occur in $\Gamma, \Delta$.

**Fig. 2** Rules of the calculus G3K

For those logics, labelled sequent calculi are based on a language $\mathcal{L}_{LS}$ which extends $\mathcal{L}$ with a set of world labels and a set of relational atoms of the form $xRy$ for world labels $x$, $y$. Formulas of $\mathcal{L}_{LS}$ have only two possible forms

$$x : A \qquad \text{or} \qquad xRy,$$

where $A$ is a formula of $\mathcal{L}$.

Sequents are now pairs $\Gamma \Rightarrow \Delta$ of multisets of formulas of $\mathcal{L}_{LS}$.

Accordingly, the rules for *all* logical operators are based on the inductive definition of the forcing relation between worlds and formulas of $\mathcal{L}$, here denoted by $x : A$. In particular, the forcing condition for the $\Box$ modality

$$x \Vdash \Box A \qquad \text{iff} \qquad \text{for all } y, \text{ if } xRy, \text{ then } y \Vdash A$$

determines the following rules:

$$\frac{xRy, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \Box A}\ \mathcal{R}\Box_{(y!)} \qquad \frac{y : A, xRy, x : \Box A, \Gamma \Rightarrow \Delta}{xRy, x : \Box A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\Box$$

with the side condition for $\mathcal{R}\Box$ that $y$ does not occur in $\Gamma, \Delta$.

$$\frac{xRy, y : \Box A, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \Box A}\ \mathcal{R}\Box^{L\ddot{o}b}_{(y!)} \qquad \frac{}{xRx, \Gamma \Rightarrow \Delta}\ Irref \qquad \frac{xRz, xRy, yRz, \Gamma \Rightarrow \Delta}{xRy, yRz, \Gamma \Rightarrow \Delta}\ Trans$$

**Fig. 3** Additional rules for the calculus G3KGL

This is the very general framework behind the labelled sequent calculus G3K for the basic normal modal logic $\mathbb{K}$ we defined in Fig. 2. The results presented by Negri and von Plato in [57, Chap. 6] assure that any extension of $\mathbb{K}$ that is semantically characterised by (co)geometric frame conditions[20] can also be captured by an extension of G3K with appropriate (co)geometric rules.

Any of such extensions will keep the good structural properties of G3cp. Most relevantly, proving termination of the proof search in these calculi provides a neat proof of decidability for the corresponding logics, which allows the direct construction of a countermodel for a given formula generating a failed proof search. In many cases, such a proof of decidability is obtained as follows: because of the structural correspondence between the proof tree generated during a backward proof search and the relational frames at the semantic level, proving the finite model property (hence, decidability) for a given logic reduces to proving that the such a root-first search generates only a finite number of new labels; in general, a specific proof search strategy needs to be defined for each system under consideration, in order to prevent looping interactions of (some of) the rules of the given calculus. A detailed account of the methodology is given in Garg et al. [21].

## 6 Implementing G3KGL

The frame conditions characterising GL—i.e. Noetherianity and transitivity, or, equivalently, irreflexivity, transitivity and finiteness—cannot be expressed by a (co)geometric implication since finiteness and Noetherianity are intrinsically second order.

Therefore it is not possible to define a labelled sequent calculus for GL by simply extending G3K with naive semantic rules.

Nevertheless, it is still possible to internalise the possible world semantics by relying on a modified definition of the forcing relation—valid for `ITF` models—in which the standard condition for $\Box$ is substituted by the following

$$x \Vdash \Box A \quad \text{iff} \quad \text{for all } y, \text{ if } xRy \text{ and } y \Vdash \Box A, \text{ then } y \Vdash A. \tag{1}$$

Following Negri [53], this suggests modifying the $\mathcal{R}\Box$ rule according to the right-to-left direction of (1).

The resulting labelled sequent calculus G3KGL is then characterised by the initial sequents, the propositional rules and the $\mathcal{L}\Box$ in Fig. 2 together with the rules in Fig. 3.

The rule $\mathcal{R}\Box^{L\ddot{o}b}$ obeys to the side condition of not being $y$ free in $\Gamma$, $\Delta$, in line with the universal quantifier used in the forcing condition.

[20] Recall that a frame condition is said to be geometric [58] when it has shape $\forall \vec{x}(\varphi \rightarrow \psi)$, where $\varphi$, $\psi$ are first-order formulas that do not contain universal quantifiers or implications. This means that the semantics behind the logical system is a geometric theory in the sense explained in [15].

In Negri [54], it is proven that G3KGL has good structural properties. Moreover, in the same paper, it is easily shown that the proof search in this calculus is terminating. Its failure allows to construct a countermodel for the formula at the root of the derivation tree: it suffices to consider the top sequent of an open branch, and assume that all the labelled formulas and relational atoms in its antecedent are true, while all the labelled formulas in its consequent are false. Termination is assured by imposing saturation conditions on sequents that might prevent the application of useless rules, according to the Schütte-Takeuti-Tait reduction for classical G3 calculi. The reader is referred to Negri [54] for a detailed description of the proof search algorithm for G3KGL.
This way, a syntactic proof of decidability for GL is obtained.

### 6.1 How to Use G3KGL

It is not hard to see how to use both our formalisation of modal completeness and the already known proof theory for G3KGL to the aim of implementing a decision algorithm in HOL Light for GL: our predicate `holds (W,R) V A x` corresponds precisely to the labelled formula $x : A$. Thus we have three different ways of expressing the fact that a world $x$ forces $A$ in a given model $\langle W, R, V\rangle$:

| | |
|---|---|
| SEMANTIC NOTATION | $x \Vdash A$ |
| LABELLED SEQUENT CALCULUS NOTATION | $x : A$ |
| HOL LIGHT NOTATION | `holds (W,R) V A x` |

This correspondence suggests that a *deep* embedding of G3KGL is unnecessary. Since internalising possible world semantics in sequent calculi is, in fact, a syntactic formalisation of that very semantics, we can use *our own* formalisation in HOL Light of validity in Kripke frames and *adapt* the goal stack mechanism of the theorem prover to develop G3KGL proofs by relying on that very mechanism.

That adaptation starts with generalising the standard tactics for classical propositional logic already defined in HOL Light.

Let us call any expression of forcing in HOL Light notation a `holds`-proposition.

As an abstract deductive system, the logical engine underlying the proof development in HOL Light consists of a single-consequent sequent calculus for higher-order logic. We must work on a multi-consequent sequent calculus made of multisets of `holds`-propositions and (formalised) relational atoms. To handle commas, we recur to the logical operators of HOL Light: a comma in the antecedent is formalised by a conjunction; in the consequent, it is formalised by disjunction.

Since we cannot directly define multisets, we need to formalise the sequent calculus rules to operate on lists and handle permutations through standard conversions of a goal with the general shape of an $n$-ary disjunction of `holds`-propositions, for $n \geq 0$.

The intermediate tactics we now need to define operate

- on the components of a general goal-term consisting of a finite disjunction of `holds`-propositions: these correspond to the labelled formulas that appear on the right-hand side of a sequent of G3KGL so that some of our tactics can behave as the appropriate right rules of that calculus;

- on the list of hypotheses of the goal stack, which correspond to the labelled formulas and possible relational atoms that occur on the left-hand side of a sequent of G3KGL, so that some of our tactics can behave as the appropriate left rules of that calculus.

In order to make the automation of the process easier, among the hypotheses of each goal stack, we make explicit the assumptions about the transitivity of the accessibility relation, the left-to-right direction of the standard forcing condition for the $\Box$, and, separately, the right-to-left direction of the forcing condition for the same modality in `ITF` models. This way, the formal counterparts of, respectively, *Trans*, $\mathcal{L}\Box$, and $\mathcal{R}\Box^{Löb}$ can be executed adequately by combining standard tactics of HOL Light.

Our classical propositional tactics implement, for purely practical reasons, left and right rules for the (bi) implication-free fragment of $\mathcal{L}$. Therefore, the only intermediate tactics determining a branching in the derivation tree—i.e. the generation of subgoals in HOL Light goal-stack—are those corresponding to $\mathcal{L}\vee$ and $\mathcal{R}\wedge$. The translation of a formula of $\mathcal{L}_{LS}$, given as a goal, to its equivalent formula in the implemented fragment is produced by our conversion `HOLDS_NNFC_UNFOLD_CONV` (sources). A similar conversion is defined for the labelled formulas that appear among the hypotheses, i.e. on the left-hand side of our formal sequents.

Specific tactics handle left rules for propositional connectives: each is defined by using HOL Light theorem tactic(al)s, which can be thought of as operators on a given goal, taking theorems as input to apply a resulting tactic to the latter. For instance, the left rule for negation $\mathcal{L}\neg$ is defined by

```
let NEG_LEFT_TAC : thm_tactic =(sources)
    let pth = MESON [] `¬P ⟹ (P ∨ Q) ⟹ Q` in
    MATCH_MP_TAC o MATCH_MP pth
```

which uses the propositional tautology $\neg P \Longrightarrow (P \vee Q) \Longrightarrow Q$ in an MP rule instantiated with a negated `holds`-proposition occurring among the hypotheses; then it adds the `holds`-proposition among the disjuncts of the new goal, as expected. $\mathcal{R}\neg$ is defined analogously by `NEG_RIGHT_TAC` (sources).

On the contrary, $\mathcal{L}\vee$ and $\mathcal{R}\wedge$ are implemented by combining theorem tactic(al)s based on the basic operators `CONJ_TAC` and `DISJ_CASES`.

Modal rules are handled employing the explicit hypotheses we previously mentioned to deal with $\mathcal{L}\Box$ and $\mathcal{R}\Box^{Löb}$. The same approach also works for the rule *Trans*, which we handle through a theorem tactical `ACC_TCL` (sources) operating on the relational atoms among the hypotheses of a current goal stack.

This basically completes the formalisation—or shallow embedding—of G3KGL in HOL Light.

### 6.2 Design of the Proof Search

Our efforts now turn to run, in HOL Light, an automated proof search w.r.t. the implementation of G3KGL we have sketched in the previous section.
We can again rely on theorem tactic(al)s to build the main algorithm, but we need to define them recursively this time.

First, we have to apply recursively the left rules for propositional connectives, as well as the $\mathcal{L}\Box$ rule: this is made possible by the theorem tactic `HOLDS_TAC` (sources). Furthermore, we need to saturate the sequents w.r.t. the latter modal rule, by considering all the possible

relational atoms and applications of the rule for transitivity, and, eventually, further left rules: that is the job of the theorem tactic `SATURATE_ACC_TAC` (sources).

After that, it is possible to proceed with the $\mathcal{R}\Box^{\text{Löb}}$: that is trigged by the `BOX_RIGHT_TAC` (sources), which operates by applying (the implementation of) $\mathcal{R}\Box^{\text{Löb}}$ after `SATURATE_ACC_TAC` and `HOLDS_TAC`.

At this point, it is possible to optimise the application of `BOX_RIGHT_TAC` by applying the latter tactic after a "sorting tactic" `SORT_BOX_TAC` (sources): that tactic performs a conversion of the goal term and orders it so that priority is given to negated `holds`-propositions, followed by those `holds`-propositions formalising the forcing of a boxed formula. Each of these types of `holds`-propositions are sorted furthermore as follows:

> `holds WR V p w` precedes `holds WR V q w` if `p` occurs free in `q` and `q` does not occur free in `p`; or if `p` is "less than" `q` w.r.t. the structural ordering of types provided by the OCaml module `Pervasives`.

We programmed the tactic `GL_TAC` to perform the complete proof search; from that tactic, we define the expected `GL_RULE` (sources).

To keep it short, our tactic works as expected:

1. Given a formula $A$ of $\mathcal{L}$, `let`-terms are rewritten together with definable modal operators, and the goal is set to `|- A`;
2. A model $\langle W, R, V\rangle$ and a world $w \in W$—where $W$ sits on the type `num`—are introduced. The main goal is now `holds (W,R) V A w`;
3. Explicit additional hypotheses are introduced to be able to handle modal and relational rules, as anticipated before;
4. All possible propositional rules are applied after unfolding the modified definition of the predicate `holds` given by `HOLDS_NNFC_UNFOLD_CONV`. This assures that at each step of the proof search, the goal term is a finite conjunction of disjunctions of positive and negative `holds`-propositions. As usual, priority is given to non-branching rules, i.e. to those that do not generate subgoals. Furthermore, the hypothesis list is checked, and *Trans* is applied whenever possible; the same holds for $\mathcal{L}\Box$, which is applied any appropriate hypothesis after the tactic triggering transitivity. Each new goal term is reordered by `SORT_BOX_TAC`, which always precedes the implementation of $\mathcal{R}\Box^{Löb}$.

The procedure is repeated starting from step 2. The tactic governing this repetition is `FIRST o map CHANGED_TAC`, which triggers the correct tactic—corresponding to a specific step of the very procedure—in `GL_STEP_TAC` (sources) that does not fail.

At each step, moreover, the following condition is checked by calling `ASM_REWRITE_TAC`:

> *Closing* The same `holds`-proposition occurs both among the current hypotheses *and* the disjuncts of the (sub)goal; or `holds (W,R) V False x` occurs in the current hypothesis list for some label `x`.

This condition states that the current branch is closed, i.e. an initial sequent has been reached, or the sequent currently analysed has a labelled formula $x : \bot$ in the antecedent.

Termination of the proof search is assured by the results presented in [54]. Therefore, we can justify our choice to conclude `GL_TAC` by a `FAIL_TAC` that, when none of the steps 2–4 can be repeated during a proof search, our algorithm terminates, informing us that a countermodel for the input formula can be built.

That is exactly the job of our `GL_BUILD_COUNTERMODEL` (sources) tactic: it considers the goal state which the previous tactics of `GL_TAC` stopped at; collects all the hypotheses, discarding the meta-hypotheses; and negates all the disjuncts constituting the goal term.

Again, by referring to the results in [54, 58], we know that this information suffices to the user to construct a relational countermodel for the formula $A$ given as input to our theorem prover for GL.

### 6.3 Some Examples

Because of its adequate arithmetical semantics, Gödel–Löb logic reveals an exceptionally simple instrument to study the arithmetical phenomenon of self-reference, as well as Gödel's results concerning (in)completeness and (un)provability of consistency.

From a formal viewpoint, an arithmetical realisation $*$ in Peano arithmetic (PA)[21] of our modal language consists of a function commuting with propositional connectives and such that $(\Box A)^* := Bew(\ulcorner A^* \urcorner)$, where $Bew(x)$ is the formal provability predicate for PA.

Under this interpretation, we will read modal formulas as follows: We tested our procedure

| | |
|---|---|
| $\Box A$ | $A$ is provable in PA |
| $\neg\Box\neg A$ | $A$ is consistent with PA |
| $\neg\Box A$ | $A$ is unprovable in PA |
| $\Box\neg A$ | $A$ is refutable in PA |
| $(\Box A) \vee (\Box\neg A)$ | $A$ is decidable in PA |
| $(\neg\Box A) \wedge (\neg\Box\neg A)$ | $A$ is undecidable in PA |
| $\Box(A \leftrightarrow B)$ | $A$ and $B$ are equivalent over PA |
| $\Box\bot$ | PA is inconsistent |
| $\neg\Box\bot$, $\Diamond\top$ | PA is consistent |

GL_RULE on some examples.[22] First, as a sanity check, we applied it to all the schemas (including axioms but excluding rules) that were initially proven directly in the GL calculus (see our discussion in Sect. 3.2). In total, 56 such lemmas were (re-)proven by our procedure in about 3.5 seconds.

Next, we used the procedure to prove some other results of meta-mathematical relevance; for instance:

*Undecidability of consistency*. If PA does not prove its inconsistency, then its consistency is undecidable. The corresponding modal formula is

$$\neg(\Box\Box\bot) \rightarrow \neg(\Box\neg\Box\bot) \wedge \neg(\Box\neg\neg\Box\bot)$$

*Undecidability of Gödel's formula*. The formula stating its own unprovability is undecidable in PA, if the latter does not prove its inconsistency. The corresponding modal formula is

$$\Box(A \leftrightarrow \neg\Box A) \wedge \neg\Box\Box\bot \rightarrow \neg\Box A \wedge \neg\Box\neg A$$

*Reflection and iterated consistency.*

$$\Box((\Box p \rightarrow p) \rightarrow \Diamond\Diamond\top) \rightarrow \Diamond\Diamond\top \rightarrow \Box p \rightarrow p$$

*Formalised Gödel's second incompleteness theorem.* In PA, the following is provable: If PA is consistent, it cannot prove its own consistency. The corresponding modal formula is

$$\neg\Box\bot \rightarrow \neg\Box\Diamond\top$$

[21] Actually, we may consider any $\Sigma_1$-sound arithmetical theory $T$ extending $\mathsf{I}\Sigma_1$ [76].

[22] The sources of our tests are available in file `GL/tests.ml`.

For the reader's sake, we describe for this specific example the main steps constituting both the decision procedure (`GL_RULE`) *and* the shallow embedding of G3KGL in the interactive proof mechanism of HOL Light. First, the goal is set to

```
‘|---(Not Box False --> Not Box Diam True)‘
```

Then, by applying the completeness theorem and unfolding and sorting the disjuncts (`HOLDS_NNFC_UNFOLD_CONV`, and `SORT_BOX_TAC` discussed above), we obtain the following goal stack:

```
  0 [‘w IN W‘] (w)
‘¬holds (W,R) V (Box Not Box Not True) w ∨
 holds (W,R) V (Box False) w‘
```

Next, by using $\mathcal{R}\neg$ (`NEG_RIGHT_TAC`), we have

```
  0 [‘w IN W‘] (w)
  1 [‘holds (W,R) V (Box Not Box Not True) w‘] (holds)
‘holds (W,R) V (Box False) w‘
```

Now, we can trigger $\mathcal{R}\Box^{Löb}$ followed by $\mathcal{L}\Box$ (`BOX_RIGHT_TAC`), to obtain:

```
  0 [‘w IN W‘] (w)
  1 [‘holds (W,R) V (Box Not Box Not True) w‘] (holds)
  2 [‘y IN W‘]
  3 [‘R w y‘] (acc)
  4 [‘holds (W,R) V (Box False) y‘] (holds)
‘holds (W,R) V (Box Not True) y ∨
 holds (W,R) V False y‘
```

By now, the second disjunct is deleted, and by applying $\mathcal{R}\Box^{Löb}$ followed by $\mathcal{R}\neg$, we are done by $\mathcal{L}\bot$ (closing condition by `ASM_REWRITE_TAC`).

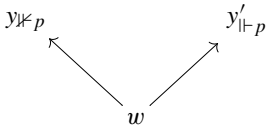
As already discussed, most of these proofs seem to be unfeasible using a generic proof search approach such as the one implemented by `MESON` or `METIS`, either axiomatically or semantically (see Sect. 3.2 and the beginning of Sect. 5, respectively). Thus, obtaining the formal proof in HOL Light of some of the above four results using more basic techniques can be challenging. Our procedure can prove them in a few seconds.

Finally, we tested the ability of our procedure to find countermodels. For instance, consider this reflection principle, which is a non-theorem in GL:

$$\Box(\Box p \vee \Box\neg p) \to (\Box p \vee \Box\neg p)$$

Our procedure fails (meaning it does not produce a theorem) on this formula and warns the user that a countermodel has been constructed. As expected, the structure that the counter-

model constructor `GL_BUILD_COUNTERMODEL` returns is the one that can be graphically rendered as




## 7 Related Work

Our formalisation 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 [18]. As we stated before, it is the core of the canonical model construction for most normal modal logics.

That approach does not work for $\mathbb{GL}$ because of its non-compactness. This issue is usually sidestepped by restricting to a finite subformula universe, as done in Boolos [8]. Nevertheless, we build the subformula universe in our mechanised proof by recurring to a restricted version of Henkin's construction without resorting to the syntactic manipulations that are proof-sketched in [8, Chap. 5].

As far as we know, *no other mechanised proof of modal completeness for* $\mathbb{GL}$ has been given before, despite there exist formalisations of similar results for several other logics.

In particular, both Doczkal and Bard [13], and Doczkal and Smolka [14] deal with completeness and decidability of non-compact modal systems—namely, converse PDL and CTL, respectively. They use Coq/SSReflect to give constructive proofs of completeness on the basis of Kozen and Parikh [43] for PDL and Emerson and Halpern [16] for CTL. Moreover, they propose a simulation of natural deduction for their formal axiomatic systems to make the proof development of lemmas in the Hilbert calculi that are required in the formalisation of completeness results easier. Their methodology is based on a variant of pruning [38], and the proposed constructive proof of completeness provides an algorithm to build derivations in the appropriate axiomatic calculi for valid formulas.

Formal proof of semantic completeness for *classical logic* has defined an established trend in interactive theorem proving since Shankar [72], where a Hintikka-style strategy is used to define a theoremhood checker for formulas built up by negation and disjunction only.

A very general treatment of systems for classical propositional logic is given in Michaelis and Nipkow [50]. They investigate an axiomatic calculus in Isabelle/HOL along with natural deduction, sequent calculus, and resolution system, and completeness is proven by Hintikka-style method for sequent calculus first, to be lifted then to the other formalisms through translations of each system into the others. Their formalisation 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 formalised in Isabelle/HOL is given in Blanchette [7], who provides a more general investigation, though unrelated to modal logics.

Regarding intuitionistic modalities, Bak [3] gives a constructive proof of completeness for IS4 w.r.t. a specific relational semantics verified in Agda. It uses natural deduction and applies modal completeness to obtain a normalisation result for the terms of the associated $\lambda$-calculus.

Bentzen [6] presents a Henkin-style completeness proof for $\mathbb{S}5$ formalised in Lean. That work applies the standard method of canonical models—since $\mathbb{S}5$ is compact.

More recently, Xu and Norrish [77] used the HOL4 theorem prover to treat model theory of modal systems. For future work, it might be interesting to use their formalisation along with the main lines of our implementation of axiomatic calculi to merge the two presentations—syntactic and semantic—exhaustively.

Our formalisation, however, has been led by the aim of developing a (prototypical) theorem prover in HOL Light for normal modal logics. The results concerning GL that we have presented here can be considered a case study of our original underlying methodology.

Automated deduction for modal logic has become a relevant scientific activity recently. An exhaustive comparison of our prover with other implementations of modal systems is beyond our scope. Nevertheless, we care to mention at least three different development lines on that trend.

The work [30] by Goré and Kikkert consists of a highly efficient hybridism of SAT solvers, modal clause-learning, and tableaux methods for modal systems. That prover deals with minimal modal logic K and its extensions T and S4. The current version of their implementation does not produce proof or a countermodel for the input formula; however, the code is publicly available, and minor tweaks should make it do so.

The conference paper [26] by Girlando and Straßburger presents a theorem prover for intuitionistic modal logics implementing proof search for Tait-style nested sequent calculi in Prolog. Because of the structural properties of those calculi, that prover returns, for each input formula, a proof in the appropriate calculus or a countermodel extracted from the failed proof search in the system. Similar remarks could be formulated for the implementation described in [27] concerning several logics for counterfactuals.

The latter formalisations are just two examples of an established modus operandi in implementing proof search in extended sequent calculi for non-classical logics by using the mere depth-first search mechanism of Prolog. Other instances of that line are e.g. [1, 24, 25][59–62], and [12]. None of those provers deals explicitly with GL, but that development approach would find no issue formalising G3KGL too.

Similar remarks about Papapanagiotou and Fleuriot [64] could be made. They propose a general framework for object-level reasoning with multiset-based sequent calculi in HOL Light. More precisely, they present a *deep* embedding of those systems by defining an appropriate relation between multisets in HOL Light and encode two G1 calculi: a fragment of intuitionistic propositional logic and its Curry-Howard analogous type theory. Specific tactics are then defined to perform an interactive proof search of a given sequent. For our purposes, it might be interesting to check whether their implementation may be of some help to enhance the performance and functionality of our prototypical theorem prover.[23]

It is worth noticing that the paper [31] by Goré et al. formalises directly in Coq the G3-style calculus GLS and proves that the system is structurally complete; most notably, it gives a computer checked proof that the cut rule is admissible in that calculus, and by this formalisation, it clarifies many aspects of the structural analysis of this logic that had remained opaque at the "pencil and paper" level.[24]

A specific tableaux based theorem prover for GL is described in Goré and Kelly [29], where they also discuss many efficiency related aspects of their artifact.

When writing this prototype, we aimed at high flexibility in experimenting with the certification of GL tautologies and with constructing possible countermodels to a modal input

[23] One might say that the framework of [64] is similar to the works in Prolog for aiming a direct deep embedding of a sequent calculus, but it is also close to our implementation for adopting the LCF approach and for choosing HOL Light as the environment.

[24] We are deeply grateful to an anonymous reviewer for pointing to this work on formalised proof theory for provability logics.

formula. The current stage of the artefact is heavily based on the HOL Light tactic machinery and is slightly naive in some aspects, most notably concerning efficiency. A more mature approach would provide a dedicated procedure for the search phase, which uses the tactic mechanism in the last stage of its execution, as done in tactics such as METIS, MESON or ARITH, implemented, for instance, by Harrison [33], Hurd [37], and Färber and Kaliszyk [17].

An immediate step would be to enhance the implementation of formal proofs in G3KGL so that derivation trees are represented as proof objects in the HOL Logic and checked by our procedure. Incidentally, this would constitute an alternative implementation of the labelled sequent calculus using the paradigm of deep embedding instead of shallow embedding. Moreover, we could use it to construct concrete proof trees for G3KGL. The ideal goal would be to make the prover run in PSPACE.

We also intend to measure the performance of our prototype—and its eventual definitive version—using a benchmark set for GL in Balsiger et al. Logics Workbenchs [4], and compare it with, e.g. Goré and Kelly's theorem prover.

Moving from the experiments about GL we propose in the present work, we plan to develop a more general mechanism to deal with (ideally) the whole set of normal modal logics within HOL Light. At the same time, we intend to add the necessary machinery to check the generated countermodels without affecting the program's performance.

**Author Contributions** Both authors contributed equally to this work.

**Funding** Open access funding provided by Università degli Studi di Firenze within the CRUI-CARE Agreement. First author is supported by project PRIN2017 "Real and Complex Manifolds: Topology, Geometry and holomorphic dynamics" (code 2017JZ2SW5), and by GNSAGA of INdAM. Second author is supported by project PRIN2017 "IT Matters: Methods and Tools for Trustworthy Smart systems" (code 2017FTXR7S).

## Declarations

**Conflict of interest** The authors declare that they have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.



## References

1. Alenda, R., Olivetti, N., Pozzato, G.L.: CSL-lean: a theorem-prover for the logic of comparative concept similarity. Electron. Notes Theor. Comput. Sci. **262**, 3–16 (2010)
2. Avron, A.: The method of hypersequents in the proof theory of propositional non-classical logics. In: Logic: from Foundations to Applications: European Logic Colloquium, pp. 1–32 (1996)
3. Bak, M.: Introspective Kripke models and normalisation by evaluation for the $\lambda^{\Box}$-calculus. In: 7th Workshop on Intuitionistic Modal Logic and Applications (IMLA 2017). https://github.com/mietek/imla2017/blob/master/doc/imla2017.pdf
4. Balsiger, P., Heuerding, A., Schwendimann, S.: A benchmark method for the propositional modal logics K, KT, S4. J. Autom. Reason. **24**(3), 297–317 (2000)

5. Beklemishev, L., Visser, A.: Problems in the logic of provability. In: Mathematical Problems from Applied Logic I, pp. 77–136. Springer, Cham (2006)
6. Bentzen, B.: A Henkin-style completeness proof for the modal logic S5. CoRR (2019). arXiv:1910.01697
7. Blanchette, J.C.: 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, pp. 1–13. Association for Computing Machinery, New York (2019)
8. Boolos, G.: The Logic of Provability. Cambridge University Press, Cambridge (1995)
9. Brünnler, K.: Deep sequent systems for modal logic. Arch. Math. Logic **48**(6), 551–577 (2009)
10. Copeland, B.J.: The genesis of possible worlds semantics. J. Philos. Logic **31**(2), 99–137 (2002)
11. Dalmonte, T., Olivetti, N., Negri, S.: Non-normal modal logics: Bi-neighbourhood semantics and its labelled calculi. In: Advances in Modal Logic 2018. AIX-MARSEILLE UNIVERSITÉ (2018)
12. Dalmonte, T., Negri, S., Olivetti, N., Pozzato, G.L.: Theorem proving for non-normal modal logics. In: OVERLAY 2020, Udine, Italy, September 2021
13. Doczkal, C., Bard, J.: Completeness and decidability of converse pdl in the constructive type theory of Coq. In: Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, pp. 42–52. Association for Computing Machinery, New York (2018)
14. Doczkal, C., Smolka, G.: Completeness and decidability results for CTL in constructive type theory. J. Autom. Reason. **56**(3), 343–365 (2016)
15. Dyckhoff, R., Negri, S.: Geometrisation of first-order logic. Bull. Symb. Logic **21**(2), 123–163 (2015)
16. Emerson, E.A., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. J. Comput. Syst. Sci. **30**(1), 1–24 (1985)
17. Färber, M., Kaliszyk, C.: Metis-based paramodulation tactic for HOL Light. GCAI **36**, 127–136 (2015)
18. Fitting, M., Mendelsohn, R.L.: First-Order Modal Logic, vol. 277. Springer, Dordrecht (2012)
19. Fitting, M.: Proof Methods for Modal and Intuitionistic Logics, vol. 169. Springer, Dordrecht (2013)
20. Gabbay, D.M.: Labelled Deductive Systems. Oxford Logic Guides, vol. 33(1). Oxford University Press, Oxford (1996)
21. Garg, D., Genovese, V., Negri, S.: Countermodels from sequent calculi in multi-modal logics. In: 2012 27th Annual IEEE Symposium on Logic in Computer Science, pp. 315–324 (2012)
22. Gentzen, G.: Untersuchungen über das logische Schließen I. Math. Z. **39**, 176–210 (1935)
23. Gentzen, G.: Untersuchungen über das logische Schließen II. Math. Z. **39**, 405–431 (1935)
24. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Analytic tableaux for KLM preferential and cumulative logics. In: International Conference on Logic for Programming Artificial Intelligence and Reasoning, pp. 666–681. Springer, Berlin (2005)
25. Giordano, L., Gliozzi, V., Pozzato, G.L.: KLMLean 2.0: a theorem prover for KLM logics of nonmonotonic reasoning. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, pp. 238–244. Springer, Cham (2007)
26. Girlando, M., Straßburger, L.: Moin: a nested sequent theorem prover for intuitionistic modal logics (system description). In: International Joint Conference on Automated Reasoning, pp. 398–407. Springer, Cham (2020)
27. Girlando, M., Lellmann, B., Olivetti, N., Pesce, S., Pozzato, G.L.: Calculi, countermodel generation and theorem prover for strong logics of counterfactual reasoning. J. Logic Comput., **01**, exab084 (2022)
28. Gödel, K.: 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, vol. 1, pp. 296–303 (1986)
29. Goré, R., Kelly, J.: Automated proof search in Gödel–Löb provability logic. In: Abstract, British Logic Colloquium (2007)
30. Goré, R., Kikkert, C.: CEGAR-Tableaux: improved modal satisfiability via modal clause-learning and SAT. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, pp. 74–91. Springer, Cham (2021)
31. Goré, R., Ramanayake, R., Shillito, I.: Cut-elimination for provability logic by terminating proof-search: formalised and deconstructed using coq. In: Das, A., Negri, S. (eds.) Automated Reasoning with Analytic Tableaux and Related Methods, pp. 299–313. Springer, Cham (2021)
32. Hakli, R., Negri, S.: Does the deduction theorem fail for modal logic? Synthese **187**(3), 849–867 (2012)
33. Harrison, J.: Optimizing proof search in model elimination. In: Automated Deduction-Cade-13: 13th International Conference on Automated Deduction New Brunswick. NJ, USA, July 30–August 3, 1996 Proceedings, vol. 13, pp. 313–327. Springer, Dordrecht (1996)
34. Harrison, J.: HOL Light tutorial (2017). http://www.cl.cam.ac.uk/~jrh13/hol-light/tutorial.pdf
35. Harrison, J.: The HOL Light Theorem Prover (2022). https://github.com/jrh13/hol-light
36. Harrop, R.: On the existence of finite models and decision procedures for propositional calculi. Math. Proc. Camb. Philos. Soc. **54**(1), 1–13 (1958)

37. Hurd, J.: First-order proof tactics in higher-order logic theorem provers. In: Design and Application of Strategies/Tactics in Higher Order Logics, Number NASA/CP-2003-212448 in NASA Technical Reports, pp. 56–68 (2003)
38. Kaminski, M., Schneider, T., Smolka, G.: Correctness and worst-case optimality of Pratt-style decision procedures for modal and hybrid logics. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, pp. 196–210. Springer, Berlin (2011)
39. Kanger, S.: Provability in Logic. Stockholm Studies in Philosophy. Almqvist and Wiksell, Stockholm (1957)
40. Kashima, R.: Cut-free sequent calculi for some tense logics. Studia Logica **53**, 119–135 (1994)
41. Ketonen, O.: Untersuchungen Zum Prädikatenkalkul. J. Symbol. Logic **10**(4), 127–130 (1945)
42. Kleene, S.C.: Permutability of inferences in Gentzen's calculi LK and LJ. Memoirs Am. Math. Soc. **10**, 1–26 (1952)
43. Kozen, D., Parikh, R.: An elementary proof of the completeness of PDL. Theor. Comput. Sci. **14**(1), 113–118 (1981)
44. Kripke, S.A.: Semantical analysis of intuitionistic logic I. In: Studies in Logic and the Foundations of Mathematics, vol. 40, pp. 92–130. Elsevier, Amsterdam (1965)
45. Kurokawa, H.: Hypersequent calculi for modal logics extending S4. In: JSAI International Symposium on Artificial Intelligence, pp. 51–68. Springer, Cham (2013)
46. Ladner, R.E.: The computational complexity of provability in systems of modal propositional logic. SIAM J. Comput. **6**(3), 467–480 (1977)
47. Lambek, J., Scott, P.J.: Introduction to Higher-Order Categorical Logic, vol. 7. Cambridge University Press, Cambridge (1988)
48. Maggesi, M., Perini Brogi, C.: A formal proof of modal completeness for provability logic. In: Cohen, L., Kaliszyk, C. (eds.) 12th International Conference on Interactive Theorem Proving (ITP 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 193, pp. 26:1–26:18, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern (2021)
49. Marin, S., Straßburger, L.: Label-free modular systems for classical and intuitionistic modal logics. In: Advances in Modal Logic, vol. 10. Groningen, The Netherlands (2014)
50. Michaelis, J., Nipkow, T.: Formalized proof systems for propositional logic. In: Abel, A., Nordvall Forsberg, F., Kaposi, A. (eds.) 23rd Int. Conf. Types for Proofs and Programs (TYPES 2017). LIPIcs, vol. 104, pp. 6:1–6:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Wadern (2018)
51. Mints, G.: Short Introduction to Modal Logic. Center for the Study of Language and Information Publication Lecture Notes, Cambridge University Press, Cambridge (1992)
52. Mints, G., Feys, R.: Sistemy Lyuisa i sistema T (Supplement to the Russian Translation). In: Feys, R. (ed.) Modal Logic, pp. 422–509. Nauka, Moscow (1974)
53. Negri, S.: Proof analysis in modal logic. J. Philos. Logic **34**(5), 507–544 (2005)
54. Negri, S.: Proofs and countermodels in non-classical logics. Logica Univ. **8**(1), 25–60 (2014)
55. Negri, S.: Proof theory for non-normal modal logics: the neighbourhood formalism and basic results. IfCoLog J. Logics Appl. **4**(4), 1241–1286 (2017)
56. Negri, S., Pavlović, E.: Proof-theoretic analysis of the logics of agency: the deliberative stit. Studia Logica **109**(3), 473–507 (2021)
57. Negri, S., von Plato, J.: Structural Proof Theory. Cambridge University Press, Cambridge (2008)
58. Negri, S., von Plato, J.: Proof Analysis: A Contribution to Hilbert's Last Problem. Cambridge University Press, Cambridge (2011)
59. Olivetti, N., Pozzato, G.L.: CondLean: a theorem prover for conditional logics. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, pp. 264–270. Springer, Berlin (2003)
60. Olivetti, N., Pozzato, G.L.: CondLean 3.0: Improving CondLean for stronger conditional logics. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, pp. 328–332. Springer, Berlin (2005)
61. Olivetti, N., Pozzato, G.L.: Theorem proving for conditional logics: CondLean and GoalDuck. J. Appl. Non-Classical Logics **18**(4), 427–473 (2008)
62. Olivetti, N., Pozzato, G.L.: NESCOND: an implementation of nested sequent calculi for conditional logics. In: International Joint Conference on Automated Reasoning, pp. 511–518. Springer, Cham (2014)
63. Olivetti, N., Pozzato, G.L.: A standard internal calculus for Lewis' counterfactual logics. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, pp. 270–286. Springer, Cham (2015)
64. Papapanagiotou, P., Fleuriot, J.: Object-level reasoning with logics encoded in HOL light. In: 15th Fifteenth Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, pp. 18–34. Open Publishing Association (2021)

65. Perini Brogi, C.: Investigations of proof theory and automated reasoning for non-classical logics. PhD thesis, DiMa – Department of Mathematics, Universitá degli studi di Genova (2022)
66. Poggiolesi, F.: The method of tree-hypersequents for modal propositional logic. In: Towards Mathematical Philosophy, pp. 31–51. Springer, Dordrecht (2009)
67. Poggiolesi, F.: Gentzen Calculi for Modal Propositional Logic, vol. 32. Springer, Dordrecht (2010)
68. Poggiolesi, F.: Natural deduction calculi and sequent calculi for counterfactual logics. Studia Logica **104**(5), 1003–1036 (2016)
69. Popkorn, S.: First Steps in Modal Logic. Cambridge University Press, Cambridge (1994)
70. Pottinger, G.: Uniform, cut-free formulations of T, S4 and S5. J. Symbol. Logic **48**(3), 900 (1983)
71. Restall, G.: Proofnets for S5: sequents and circuits for modal logic. In: Dimitracopoulos, C., Newelski, L., Normann, D. (eds.) Logic Colloquium 2005, pp. 151–172. Cambridge University Press, Cambridge (2007)
72. Shankar, N.: Towards mechanical metamathematics. J. Autom. Reason. **1**(4), 407–434 (1985)
73. Solovay, R.M.: Provability interpretations of modal logic. Isr. J. Math. **25**(3–4), 287–304 (1976)
74. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory, 2nd ed.. Cambridge Tracts in Theoretical Computer Science, vol. 43. Cambridge University Press, Cambridge (2000)
75. van Benthem, J., Blackburn, P.: Modal logic: a semantic perspective. In: Handbook of Modal Logic, vol. 3, pp. 1–84. Elsevier, Amsterdam (2007)
76. Verbrugge, R.L.C.: Provability logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy, fall, 2017th edn. Stanford University, Stanford, Metaphysics Research Lab (2017)
77. Xu, Y., Norrish, M.: Mechanised modal model theory. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) Automated Reasoning, pp. 518–533. Springer, Cham (2020)