

# Formalizing Henkin-Style Completeness of an Axiomatic System for Propositional Logic

Asta Halkjær From

Technical University of Denmark

**Abstract.** We formalize a Henkin-style completeness proof for an axiomatic system for propositional logic in the proof assistant Isabelle/HOL. Our formalization precisely details the structure of this proof method.

**Keywords:** Propositional logic · Henkin-style completeness · Isabelle/HOL.

## 1 Introduction

Hilbert proved the completeness of an axiomatic system for propositional logic in 1917-18 [34], Gödel proved the completeness of first-order logic in 1929 [12] and Henkin simplified this proof in 1947 [13], devising what we now know as the Henkin-style method [14]. In this paper we study the structure of a Henkin-style completeness proof for an axiomatic Hilbert system for propositional logic by formalizing it in the proof assistant Isabelle/HOL [21].

Isabelle is a generic proof assistant and Isabelle/HOL is the instance based on higher-order logic. With it, we can state every definition, proposition and proof in the precise language of higher-order logic rather than in natural language. Our proof language is then completely formal which makes it possible for the machine to assist us in our endeavour. By writing our proofs in the Isar language, an acronym of *intelligible semi-automated reasoning* [32], we can have Isabelle check everything that we type. In particular, Isar contains commands such as **assume** to introduce assumptions, **have** to state a partial result and **moreover** to chain several of these together. After these commands, we typically write so-called «cartouches» delimited by angle brackets that contain our higher-order logic terms: definitions, statements and so on [33]. Our proofs are checked by the trusted Isabelle/HOL kernel but we do not typically write proofs directly using the kernel's axioms and inference rules. Instead we give the name of a prover that implements a proof search procedure like tableaux or resolution and Isabelle will run the prover to obtain the proof object. By formalizing our proofs like this we know that our conclusions always follow.

Of course, Isabelle cannot verify that our definitions match our intentions, that part is up to us, but formalization still reduces the possibility of mistakes. In particular, it reduces the surface area where mistakes can happen since the proofs themselves are checked by the machine. Not only does a formalization like the one we present increase the trust in our result, it can also serve as a reference to understand the proof since every detail is given: no case can be omitted as

“trivial” or left as an “exercise for the reader.” Our work can also act as starting point for formalizing other results based on the same techniques.

The full formalization, just below 400 lines, is available online:

<https://github.com/logic-tools/axiom>

We reproduce the essential pieces of it here and introduce parts of the syntax as we go along but forgo any thorough explanation.

## 1.1 Structure of the paper

After giving a brief history of formalized completeness proofs we start off by formalizing the syntax and semantics of our propositional logic (§ 2) and defining a sound proof system (§ 3). The idea of the completeness proof is as follows: given a formula  $\phi$  valid under assumptions  $\psi_1, \dots, \psi_k$ , assume for the sake of contradiction that there is no corresponding derivation:

$$\not\vdash \psi_1 \longrightarrow \dots \longrightarrow \psi_k \longrightarrow \phi$$

This means we cannot derive falsity,  $\perp$ , when also assuming  $\neg\phi$ , i.e.:

$$\not\vdash \neg\phi \longrightarrow \psi_1 \longrightarrow \dots \longrightarrow \psi_k \longrightarrow \perp$$

The set  $\{\neg\phi, \psi_1, \dots, \psi_k\}$  is therefore *consistent* and can be turned into a *maximal consistent set* (§ 4) through an *extension* (§ 5). Such sets are *Hintikka* sets (§ 6) and their elements have a model. This contradicts the validity assumption, proving that a derivation must exist. The proof system is therefore complete (§ 7) and we conclude with possible extensions (§ 8).

## 1.2 A history of formalized completeness proofs

Our formalization is only one in a long line of formalized completeness proofs.

Completeness proofs can generally be split into two categories based on their approach: semantic proofs in the style of Gödel [12] and Henkin [13] on the one hand and syntactic proofs in the style of Beth and Hintikka [17] and Gallier [11] on the other. Fitting and Mendelsohn call the semantic proofs “synthetic” because they start from a formula and *synthesize* new ones, building up larger and larger sets of formulas that are consistent with the starting point [9]. Formulas in such sets are then shown to have a model and this is the approach we take in this paper. Fitting and Mendelsohn contrast this with the syntactic proofs that they dub “analytic” because they work by *analyzing* the given formula, breaking it into smaller and smaller subformulas and reasoning from those. In these proofs we typically construct a counterexample from the open leaves or an infinite path of a failed derivation attempt. The synthetic approach is remarked to have a *mathematical*, abstract feeling whereas the analytic approach is more *computational* and often resembles an actual prover for the logic [5].

The Henkin-style completeness method has been applied to modal logic from the beginning, notably to system S5 as early as 1959 by Bayart (in French) [1]. Bentley recently formalized such a proof in the proof assistant Lean [2]. Jørgensen et al. adapted the synthetic approach to a tableau system for hybrid logic [16] with a formalization in Isabelle/HOL due to the present author [10].

In 1985, Shankar formalizes Shoenfield’s first-order logic and axiomatic proof system in the Boyer-Moore theorem prover [28]. They show propositional completeness of the system analytically by defining a tautology checker for a fragment of the syntax based on negation and disjunction.

In 1996, Persson shows constructive completeness for intuitionistic first-order logic in Martin-Löf type theory using the proof assistant ALF [24]. Their proof has a synthetic flavor and their result is constructive: they obtain a program that transforms a proof of validity into a derivation in either natural deduction or sequent calculus. Persson also formalizes an axiomatic system but without proving its completeness.

By early 2000, Margetson formalizes the completeness of first-order logic and the cut elimination theorem for sequent calculus in Isabelle/HOL and Ridge later updates the formalization to the Isar language [18]. Their completeness proof is in the Beth-Hintikka style and based on analyzing failing branches in proof trees.

In 2005, Braselmann and Koepke follow in the Mizar system but using a Henkin-style argument for their sequent calculus [6].

In 2007, Berghofer formalizes Fitting’s synthetic work on natural deduction [8] in Isabelle/HOL [3]. The formalized model existence theorem is based on Smullyan’s abstract consistency properties [30] and Berghofer follows Fitting in reusing the result to show the Löwenheim-Skolem theorem. The present author has extended the completeness result in that formalization to also cover open formulas [3]. In 2016, Schlichtkrull extended Berghofer’s work in another direction, namely to prove the completeness of first-order resolution [26,27].

In 2010, Ilik investigates Henkin-style arguments for both classical and intuitionistic first-order logic in the proof assistant Coq [15].

In 2017, Michaelis and Nipkow formalize a number of proof systems for propositional logic in Isabelle/HOL: natural deduction, sequent calculus, an axiomatic Hilbert system similar to ours and resolution [19,20]. They give a syntactic completeness proof for the sequent calculus and show that sequent calculus derivations can be translated into natural deduction and further into their Hilbert system, obtaining completeness for the three proof systems. Independently of this approach, they formalize the propositional model existence theorem by Fitting [8] and use this result to reprove completeness of the sequent calculus and Hilbert system, respectively. Their formalization is more ambitious than ours and therefore more involved. We start from a smaller syntax and focus on only one proof system and one approach. This leads to a simpler formalization and helps us understand the essential pieces of the approach.

Blanchette, Popescu and Traytel have recently advanced the state of completeness proofs for sequent calculus and tableau systems in Isabelle/HOL [5]. They explicitly shy away from Henkin in favor of the Beth-Hintikka style and

use codatatypes to model possibly infinite derivation trees. Their result can be instantiated for different variations of sequent calculus or tableau and various flavors of first-order logic.

Blanchette gives an overview of the formalized metatheory of various other logical calculi and automatic provers in Isabelle [4].

If we move to Gödel’s incompleteness theorems, the first one has been formalized in the Boyer-Moore theorem prover by Shankar in 1986 [29] and in Coq by O’Connor in 2003 [22]. Both incompleteness theorems have been formalized in Isabelle/HOL by Paulson in 2013 [23] and by Popescu and Traytel in 2019 [25].

In summation, the Henkin style is ubiquitous and we have seen it applied to examples such as sequent calculus and natural deduction for first-order logic, system S5 for modal logic and a tableau system for hybrid logic. Most work either extends the technique to cover more advanced logics or abstracts it so that it applies to several at once. Our contribution is to boil this proof style down to its essence, motivating each step as we present it and using a proof assistant to ensure precision, correctness and comprehensiveness. Our work may also serve as a fast-paced introduction to proof assistants.

## 2 Syntax and Semantics

We pick a minimal syntax consisting of a logical constant representing falsity, natural numbers as propositional symbols, and implication. We model the syntax as a datatype, *form*, with a constructor for each case separated by “|”:

**datatype** *form* = *Falsity* ( $\perp$ ) | *Pro nat* | *Imp form form* (**infixr**  $\longrightarrow$ ) 25)

The annotations in parentheses allow us to construct formulas using standard notation in bold. Our definition of negation as an abbreviation makes use of this:

**abbreviation** *Neg* ( $\neg$ ) [40] 40) **where**  $\neg p \equiv p \longrightarrow \perp$

We define the semantics as a primitive recursive predicate on formulas given an interpretation of propositional symbols:

**primrec** *semantics* ::  $\langle nat \Rightarrow bool \rangle \Rightarrow form \Rightarrow bool$  ( $\cdot \models \cdot$ ) [50, 50] 50) **where**  
 $\langle (I \models \perp) = False \rangle$   
 $| \langle (I \models Pro\ n) = I\ n \rangle$   
 $| \langle (I \models (p \longrightarrow q)) = ((I \models p) \longrightarrow (I \models q)) \rangle$

The first line gives the type and infix notation while the remaining lines define the predicate by each case of the syntax. The first case states that no interpretation models  $\perp$ , the second case that the semantics of a propositional symbol is given by the interpretation and finally we delegate to the meta-logic implication,  $\longrightarrow$ , to interpret the object logic implication  $\longrightarrow$ .

### 3 Proof System

We pick a simple axiomatic proof system for our purposes, consisting of modus ponens and three axiom schemas. This is Church's axiom system  $P_1$  [7]:

**inductive** *Axiomatics* ::  $\langle \text{form} \Rightarrow \text{bool} \rangle (\vdash \rightarrow [50] 50)$  **where**

*MP*:  $\langle \vdash p \Rightarrow \vdash (p \rightarrow q) \Rightarrow \vdash q \rangle$   
 $|$  *Imp1*:  $\langle \vdash (p \rightarrow q \rightarrow p) \rangle$   
 $|$  *Imp2*:  $\langle \vdash ((p \rightarrow q \rightarrow r) \rightarrow (p \rightarrow q) \rightarrow p \rightarrow r) \rangle$   
 $|$  *Neg*:  $\langle \vdash (((p \rightarrow \perp) \rightarrow \perp) \rightarrow p) \rangle$

The proof system is sound with respect to the semantics, which means that every derivable formula is true under any interpretation:

**theorem** *soundness*:  $\langle \vdash p \Rightarrow I \models p \rangle$

**by** (*induct rule: Axiomatics.induct*) *simp-all*

The second line shows that the simplifier can easily verify the theorem once we state that the proof should be performed by induction over the rules.

### 4 Consistency and Maximality

We want to work with sets of formulas where no finite subset  $S'$  syntactically entails falsity, i.e. we cannot derive  $\perp$  given  $S'$ . Our provability predicate,  $\vdash$ , has no notion of entailment but we can use implication,  $\rightarrow$ , to serve the same purpose. As such, we use the following function, *imply*, to build a chain of implications from a given list of assumptions to a conclusion. A list is a finite sequence and is either empty,  $[]$ , or built from an element, the separator  $\#$ , and a smaller list. We say that  $q$  can be *derived from*  $ps$  when we can derive  $\vdash \text{imply } ps \ q$ .

**primrec** *imply* ::  $\langle \text{form list} \Rightarrow \text{form} \Rightarrow \text{form} \rangle$  **where**

$\langle \text{imply } [] \ q = q \rangle$   
 $|$   $\langle \text{imply } (p \# ps) \ q = (p \rightarrow \text{imply } ps \ q) \rangle$

The set  $S$  is consistent exactly when there is no list  $S'$  that, when treated as a *set*, is a subset of  $S$  and that entails  $\perp$  in the sense of *imply*:

**definition** *consistent* ::  $\langle \text{form set} \Rightarrow \text{bool} \rangle$  **where**

$\langle \text{consistent } S \equiv \nexists S'. \text{ set } S' \subseteq S \wedge \vdash \text{imply } S' \ \perp \rangle$

A set is maximal when any proper extension makes it inconsistent:

**definition** *maximal* ::  $\langle \text{form set} \Rightarrow \text{bool} \rangle$  **where**

$\langle \text{maximal } S \equiv \forall p. p \notin S \rightarrow \neg \text{consistent } (\{p\} \cup S) \rangle$

Note that we allow for inconsistent maximal sets to separate concerns.

## 5 Extension

We need to grow a consistent set into a *maximal* one while preserving consistency. According to Lindenbaum's lemma, attributed to him by Tarski [31], we can always do this. Given an enumeration of formulas,  $(\phi_n)$ , we construct a corresponding sequence of consistent sets  $(S_n)$ .

Assuming  $S_n$  has been constructed, its immediate extension is given by:

$$S_{n+1} = \begin{cases} \{\phi_n\} \cup S_n & \text{if } \{\phi_n\} \cup S_n \text{ is consistent,} \\ S_n & \text{otherwise.} \end{cases}$$

That is, we only add the corresponding formula to the previous set if consistency is preserved. In the Isabelle code, we use the function *extend S f n* to construct  $S_n$  from  $S = S_0$  given an enumeration of formulas represented by  $f$ :

```
primrec extend :: ⟨form set ⇒ (nat ⇒ form) ⇒ nat ⇒ form set⟩ where
  ⟨extend S f 0 = S⟩
| ⟨extend S f (Suc n) =
  (if consistent ({f n} ∪ extend S f n)
   then {f n} ∪ extend S f n
   else extend S f n)⟩
```

To construct our *maximal consistent set* we take the infinite union  $\bigcup S_n$ :

```
definition Extend :: ⟨form set ⇒ (nat ⇒ form) ⇒ form set⟩ where
  ⟨Extend S f ≡  $\bigcup n.$  extend S f n⟩
```

It is easy to see that the starting set is a subset of the union:

```
lemma Extend-subset: ⟨S ⊆ Extend S f⟩
unfolding Extend-def by (metis Union-upper extend.simps(1) range-eqI)
```

And that any element  $S_m$  is a superset of previous elements:

```
lemma extend-bound: ⟨( $\bigcup n \leq m.$  extend S f n) = extend S f m⟩
by (induct m) (simp-all add: atMost-Suc)
```

### 5.1 Consistency

When the initial  $S$  is consistent, so is any  $S_n$  by construction:

```
lemma consistent-extend: ⟨consistent S ⇒ consistent (extend S f n)⟩
by (induct n) simp-all
```

Finally, we show that the limit,  $\bigcup S_n$ , is also consistent:

```
lemma consistent-Extend:
  assumes ⟨consistent S⟩
  shows ⟨consistent (Extend S f)⟩
```

We prove this by classical contradiction using the *ccontr* rule:

**unfolding** *Extend-def*  
**proof** (*rule ccontr*)

Assuming the union is inconsistent, we can derive  $\perp$  from some subset  $S'$ :

**assume**  $\langle \neg \text{consistent } (\bigcup n. \text{extend } S \text{ f } n) \rangle$   
**then obtain**  $S'$  **where**  $\langle \vdash \text{imply } S' \perp \rangle$   $\langle \text{set } S' \subseteq (\bigcup n. \text{extend } S \text{ f } n) \rangle$   
**unfolding** *consistent-def* **by** *blast*

This subset is finite so it must be a subset of a finite segment of the union, say  $S_0 \cup \dots \cup S_m$  for some  $m$ :

**then obtain**  $m$  **where**  $\langle \text{set } S' \subseteq (\bigcup n \leq m. \text{extend } S \text{ f } n) \rangle$   
**using** *UN-finite-bound* **by** (*metis List.finite-set*)

But every element in  $(S_n)$  is a subset of the next, so  $S'$  is a subset of  $S_m$ :

**then have**  $\langle \text{set } S' \subseteq \text{extend } S \text{ f } m \rangle$   
**using** *extend-bound* **by** *blast*

And we already established that any such element is consistent:

**moreover have**  $\langle \text{consistent } (\text{extend } S \text{ f } m) \rangle$   
**using** *assms consistent-extend* **by** *blast*

So there cannot be an inconsistent subset  $S'$  and we have our contradiction:

**ultimately show** *False*  
**unfolding** *consistent-def* **using**  $\langle \vdash \text{imply } S' \perp \rangle$  **by** *blast*  
**qed**

In conclusion,  $\bigcup S_n$  is consistent when  $S_0$  is.

## 5.2 Maximality

Importantly, the union  $\bigcup S_n$  is also maximal (regardless of the choice of  $S_0$ ):

**lemma** *maximal-Extend*:  
**assumes**  $\langle \text{surj } f \rangle$   
**shows**  $\langle \text{maximal } (\text{Extend } S \text{ f}) \rangle$   
*(proof omitted)*

The proof is similar to the one for consistency. If the union is not maximal then there is some  $\phi_k \notin \bigcup S_n$  such that  $\{\phi_k\} \cup \bigcup S_n$  is consistent. Since  $\phi_k \notin \bigcup S_n$ , it was not added to the sequence, i.e  $\phi_k \notin S_{k+1}$ , and by construction this must be because  $\{\phi_k\} \cup S_k$  is inconsistent. But  $\{\phi_k\} \cup \bigcup S_n$  is a superset of  $\{\phi_k\} \cup S_k$ , so  $\{\phi_k\} \cup \bigcup S_n$  must be inconsistent too, contradicting our assumption.

## 6 Hintikka Sets

The completeness proof works by showing that every maximal consistent set is a Hintikka set, where Hintikka sets are defined as follows:

```

locale Hintikka =
  fixes  $H :: \langle \text{form set} \rangle$ 
  assumes
    NoFalsity:  $\langle \perp \notin H \rangle$  and
    Pro:  $\langle \text{Pro } n \in H \implies (\neg \text{Pro } n) \notin H \rangle$  and
    ImpP:  $\langle (p \longrightarrow q) \in H \implies (\neg p) \in H \vee q \in H \rangle$  and
    ImpN:  $\langle (\neg (p \longrightarrow q)) \in H \implies p \in H \wedge (\neg q) \in H \rangle$ 

```

The idea is to ensure that every formula in a set is satisfiable by ensuring through syntactic criteria that the set is *downwards saturated* [30], i.e. that the satisfiability of any complex formula is guaranteed by conditions on its sub-formulas. Since  $\perp$  is unsatisfiable it should never occur (*NoFalsity*), and if a propositional symbol occurs then its negation should not (*Pro*). An implication is satisfied if either the antecedent is false or the consequent is true, so if an implication occurs in a Hintikka set, then either the negated antecedent or the consequent should too (*ImpP*). If a negated implication occurs in a Hintikka set then so should both the antecedent and negated consequent (*ImpN*).

### 6.1 Model existence

The downwards saturation ensures that if we interpret every proposition in a Hintikka set as true then every larger formula in the set will be modelled by this interpretation. We therefore base the interpretation on set membership:

**abbreviation** (*input*)  $\langle \text{model } H \ n \equiv \text{Pro } n \in H \rangle$

This models any formula in a Hintikka set:

```

lemma Hintikka-model:
   $\langle \text{Hintikka } H \implies (p \in H \longrightarrow \text{model } H \models p) \wedge ((\neg p) \in H \longrightarrow \neg \text{model } H \models p) \rangle$ 
  by (induct p) (simp; unfold Hintikka-def, blast)+

```

### 6.2 Maximal consistency

Our final task is to show that a maximal consistent set is a Hintikka set:

```

lemma Hintikka-Extend:
  assumes  $\langle \text{maximal } S \rangle \langle \text{consistent } S \rangle$ 
  shows  $\langle \text{Hintikka } S \rangle$ 

```

The proof has four cases based on the cases of the Hintikka definition and we show two of them here. Consider first propositional symbols:



```

fix  $n$ 
assume  $\langle \text{Pro } n \in S \rangle$ 
moreover have  $\langle \vdash \text{ imply } [\text{Pro } n, \neg \text{Pro } n] \perp \rangle$ 
  by (simp add: FalsityE)
ultimately show  $\langle (\neg \text{Pro } n) \notin S \rangle$ 
  using assms(2) unfolding consistent-def
  by (metis bot.extremum empty-set insert-subset list.set(2))

```

We have assumed a fixed but arbitrary propositional symbol  $n$  that occurs positively in  $S$ . We can derive  $\perp$  from this in combination with a negative occurrence. Thus, both cannot appear in the consistent  $S$  and this case of the Hintikka definition is satisfied.

Next, assume that a negated implication occurs in  $S$ . We show half of the Hintikka condition by contradiction, namely that so does the antecedent:

```

assume *:  $\langle (\neg (p \longrightarrow q)) \in S \rangle$ 
show  $\langle p \in S \wedge (\neg q) \in S \rangle$ 
proof (rule conjI; rule ccontr)

```

The set  $S$  is maximal, so if it does not contain  $p$  there must be some finite subset  $S'$  of  $S$  that we can derive falsity from when adding  $p$ :

```

assume  $\langle p \notin S \rangle$ 
then obtain  $S'$  where  $S'$ :  $\langle \vdash \text{ imply } (p \# S') \perp \rangle$   $\langle \text{set } S' \subseteq S \rangle$ 
  using assms inconsistent-head by blast

```

We can *cut* out  $p$  and derive  $\perp$  directly from the negated implication:

```

moreover have  $\langle \vdash \text{ imply } ((\neg (p \longrightarrow q)) \# S') p \rangle$ 
  using add-imp ImpE1 deduct by blast
ultimately have  $\langle \vdash \text{ imply } ((\neg (p \longrightarrow q)) \# S') \perp \rangle$ 
  using cut' by blast

```

These assumptions, however, are a subset of  $S$ , contradicting its consistency:

```

moreover have  $\langle \text{set } ((\neg (p \longrightarrow q)) \# S') \subseteq S \rangle$ 
  using *(1) S'(2) by fastforce
ultimately show False
  using assms unfolding consistent-def by blast

```

## 7 Completeness

Isabelle can automatically prove the countability of formulas, providing a surjective function *from-nat* for obtaining specific elements of the enumeration  $(\phi_n)$ :

```

instance form :: countable by countable-datatype

```

Finally we reach the completeness lemma itself. We assume that  $p$  is valid under the assumptions  $ps$  and show that we can derive  $p$  from  $ps$ :

**lemma** *imply-completeness*:  
**assumes** *valid*:  $\langle \forall I \ s. \text{list-all } (\lambda q. I \models q) \ ps \longrightarrow I \models p \rangle$   
**shows**  $\langle \vdash \text{imply } ps \ p \rangle$

We proceed by contradiction and the application of a similar derivation rule:

**proof** (*rule ccontr*)  
**assume**  $\langle \neg \vdash \text{imply } ps \ p \rangle$   
**then have** \*:  $\langle \neg \vdash \text{imply } ((\neg p) \# ps) \ \perp \rangle$   
**using** *Boole* **by** *blast*

We abbreviate the starting consistent set  $?S$  and its maximal extension  $?H$ :

**let**  $?S = \langle \text{set } ((\neg p) \# ps) \rangle$   
**let**  $?H = \langle \text{Extend } ?S \text{ from-nat} \rangle$

And use the previous results to show that  $?H$  is a Hintikka set:

**have**  $\langle \text{consistent } ?S \rangle$   
**unfolding** *consistent-def* **using** \* *imply-weaken* **by** *blast*  
**then have**  $\langle \text{consistent } ?H \rangle \langle \text{maximal } ?H \rangle$   
**using** *consistent-Extend maximal-Extend surj-from-nat* **by** *blast+*  
**then have**  $\langle \text{Hintikka } ?H \rangle$   
**using** *Hintikka-Extend* **by** *blast*

We have seen that we have a model for any formula in such an  $?H$ :

**have**  $\langle \text{model } ?H \models p \rangle$  **if**  $\langle p \in ?S \rangle$  **for**  $p$   
**using** *that Extend-subset Hintikka-model*  $\langle \text{Hintikka } ?H \rangle$  **by** *blast*

So in particular for  $\neg p$  and all of  $ps$ :

**then have**  $\langle \text{model } ?H \models (\neg p) \rangle \langle \text{list-all } (\lambda p. \text{model } ?H \models p) \ ps \rangle$   
**unfolding** *list-all-def* **by** *fastforce+*

Our validity assumption then gives us that *model*  $?H$  also models  $p$ :

**then have**  $\langle \text{model } ?H \models p \rangle$   
**using** *valid* **by** *blast*

But this is a contradiction:

**then show** *False*  
**using**  $\langle \text{model } ?H \models (\neg p) \rangle$  **by** *simp*  
**qed**

As such, we must be able to derive any valid formula:

**theorem** *completeness*:  $\langle \forall I. I \models p \implies \vdash p \rangle$   
**using** *imply-completeness* [**where**  $ps = \langle [] \rangle$ ] **by** *simp*

## 8 Conclusion

We have shown how to formalize the soundness and completeness of a simple axiomatic proof system for propositional logic in Isabelle/HOL. The proof assistant is sophisticated enough that we can do the soundness proof almost automatically and use constructions like infinite sets in the proof of completeness.

Our choice of propositional logic means that we miss out on an aspect of Henkin’s original proof: the use of special constants to witness existential statements. In return, our formalization is more manageable.

The formalization is simple to extend. The supplementary material contains a file where we have added binary disjunction and conjunction operators to the syntax and updated the proof system and so on accordingly. The result is only around 130 lines longer and we did not have to modify any existing line, only to add new ones. The biggest changes are in the Hintikka definition and maximal consistency lemma while model existence is still completely automatic.

*Acknowledgements* We thank Jørgen Villadsen, Alexander Birch Jensen, Frederik Jacobsen and the anonymous reviewers for valuable comments.

## References

1. Bayart, A.: Quasi-adéquation de la logique modale du second ordre S5 et adéquation de la logique modale du premier ordre S5. *Logique et Analyse* **2**(6/7), 99–121 (1959)
2. Bentzen, B.: A Henkin-style completeness proof for the modal logic S5 (2019), <http://arxiv.org/abs/1910.01697>, CoRR,
3. Berghofer, S.: First-Order Logic According to Fitting. *Archive of Formal Proofs* (Aug 2007), <http://isa-afp.org/entries/FOL-Fitting.html>
4. Blanchette, J.C.: Formalizing the metatheory of logical calculi and automatic provers in Isabelle/HOL (invited talk). In: Mahboubi, A., Myreen, M.O. (eds.) *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019*. pp. 1–13. ACM (2019)
5. Blanchette, J.C., Popescu, A., Traytel, D.: Soundness and completeness proofs by coinductive methods. *Journal of Automated Reasoning* **58**(1), 149–179 (2017)
6. Braselmann, P., Koepke, P.: Gödel’s completeness theorem. *Formalized Mathematics* **13**(1), 49–53 (2005)
7. Church, A.: *Introduction to Mathematical Logic*. Princeton Mathematical Series, Princeton University Press (1956)
8. Fitting, M.: *First-Order Logic and Automated Theorem Proving*, Second Edition. Graduate Texts in Computer Science, Springer (1996)
9. Fitting, M., Mendelsohn, R.L.: *First-Order Modal Logic*. Springer (2012)
10. From, A.H.: Formalizing a Seligman-style tableau system for hybrid logic. *Archive of Formal Proofs* (Dec 2019), <http://isa-afp.org/entries/Hybrid-Logic.html>, Formal proof development
11. Gallier, J.H.: *Logic for computer science: foundations of automatic theorem proving*. Courier Dover Publications (2015)
12. Gödel, K.: *Über die Vollständigkeit des Logikkalküls*. Ph.D. thesis, University of Vienna (1929)

13. Henkin, L.: The Completeness of Formal Systems. Ph.D. thesis, Princeton University (1947)
14. Henkin, L.: The Discovery of My Completeness Proofs. *Bulletin of Symbolic Logic* **2**(2), 127–158 (1996)
15. Ilik, D.: Constructive completeness proofs and delimited control. Ph.D. thesis, École polytechnique (2010)
16. Jørgensen, K.F., Blackburn, P., Bolander, T., Braüner, T.: Synthetic completeness proofs for Seligman-style tableau systems. In: *Proceedings of the 11th conference on Advances in Modal Logic*. pp. 302–321 (2016)
17. Kleene, S.C.: *Mathematical Logic*. Wiley, London (1967)
18. Margetson, J., Ridge, T.: Completeness theorem. *Archive of Formal Proofs* (Sep 2004), <http://isa-afp.org/entries/Completeness.html>, Formal proof development
19. Michaelis, J., Nipkow, T.: Propositional proof systems. *Archive of Formal Proofs* (Jun 2017), [http://isa-afp.org/entries/Propositional\\_Proof\\_Systems.html](http://isa-afp.org/entries/Propositional_Proof_Systems.html), Formal proof development
20. Michaelis, J., Nipkow, T.: Formalized proof systems for propositional logic. In: Abel, A., Forsberg, F.N., 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 (2018)
21. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL - A Proof Assistant for Higher-Order Logic, *Lecture Notes in Computer Science*, vol. 2283. Springer (2002)
22. O’Connor, R.: Essential incompleteness of arithmetic verified by Coq. In: *International Conference on Theorem Proving in Higher Order Logics*. pp. 245–260. Springer (2005)
23. Paulson, L.C.: A machine-assisted proof of Gödel’s incompleteness theorems for the theory of hereditarily finite sets. *The Review of Symbolic Logic* **7**(3), 484–498 (2014)
24. Persson, H.: Constructive completeness of intuitionistic predicate logic. *Licentiate thesis*, Chalmers University of Technology (1996)
25. Popescu, A., Traytel, D.: A formally verified abstract account of Gödel’s incompleteness theorems. In: Fontaine, P. (ed.) *Automated Deduction – CADE 27*. pp. 442–461. Springer International Publishing, Cham (2019)
26. Schlichtkrull, A.: The resolution calculus for first-order logic. *Archive of Formal Proofs* (Jun 2016), [http://isa-afp.org/entries/Resolution\\_FOL.html](http://isa-afp.org/entries/Resolution_FOL.html), Formal proof development
27. Schlichtkrull, A.: Formalization of the resolution calculus for first-order logic. *Journal of Automated Reasoning* **61**(1-4), 455–484 (2018)
28. Shankar, N.: Towards mechanical metamathematics. *Journal of Automated Reasoning* **1**(4), 407–434 (1985)
29. Shankar, N.: *Metamathematics, machines and Gödel’s proof*. No. 38, Cambridge University Press (1997)
30. Smullyan, R.M.: *First-Order Logic*. Springer-Verlag (1968)
31. Tarski, A.: *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*. Hackett Publishing (1983)
32. Wenzel, M.: Isabelle/Isar—a generic framework for human-readable proof documents. *From Insight to Proof—Festschrift in Honour of Andrzej Trybulec* **10**(23), 277–298 (2007)
33. Wenzel, M.: *The Isabelle/Isar Reference Manual* (2020), part of the Isabelle distribution.
34. Zach, R.: Completeness before Post: Bernays, Hilbert, and the development of propositional logic. *Bulletin of Symbolic Logic* **5**(3), 331–366 (1999)