Formalizing Axiomatic Systems for Propositional Logic in Isabelle/HOL

Asta Halkjær From[1][0000−0002−3601−0804], Agnes Moesgård Eschen[1], and Jørgen Villadsen[1][0000−0003−3624−1159]

DTU Compute - Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark

`ahfrom@dtu.dk` `s151952@student.dtu.dk` `jovi@dtu.dk`

Abstract. We formalize soundness and completeness proofs for a number of axiomatic systems for propositional logic in the proof assistant Isabelle/HOL.

Keywords: Propositional Logic · Axiomatic Systems · Isabelle/HOL · Completeness · Soundness.

1 Introduction

With the proof assistant Isabelle/HOL [10] we can create canonical reference documents for logics and their metatheory. The formal language of Isabelle/HOL, namely higher-order logic, is precise and unambiguous. This means every proof can be mechanically checked. We consider here two (functionally complete) fragments of propositional logic and various axiomatic systems for these fragments. Table 1 gives an overview of the systems and fragments. Our focus is mostly syntactic and we showcase the benefits of doing this work in Isabelle. We write down both the syntax and semantics of our languages, with infix syntax and abbreviations as desired. Furthermore we specify various inference systems by their rules and axioms. The systems here are all axiomatic but the techniques work for proof systems in general.

This sets the stage for our investigations. We can easily verify that the proof systems are sound with respect to the semantics, with Isabelle doing almost all the work. We can verify completeness by adapting a formalization for a similar system or by finding derivations for the axioms of one system in the other one (and similarly for the rules). Here, Isabelle helps out: instead of painstakingly writing down each derivation, a sometimes daunting task in an axiomatic system, we can let one of its sophisticated proof methods prove its existence for us. We can even let Isabelle find the right proof method and a suitable collection of needed axioms and previously derived formulas for us with its *Sledgehammer* technology. With these tools at hand we can verify historical claims such as how some axiom can be omitted because it follows from the others.

As an example we formalize Łukasiewicz's shortest axiom for implicational propositional logic and provide, in full, his derivation of Wajsberg's axioms, for

Table 1. The formalized axiomatic systems. The first three are formalized in the theory *System-W* and use $\bot, \to$ as primitive symbols. The last three are formalized in the theory *System-R* and use $\neg, \bigvee$ as primitives, with the abbreviation $p \to q \equiv \neg p \bigvee q$.

System	Source	Page [3]	Axioms
Axiomatics	Wajsberg 1937	159	$p \to (q \to p)$ $(p \to q) \to (q \to r) \to (p \to r)$ $((p \to q) \to p) \to p)$ $\bot \to p$
FW	Wajsberg 1939	163	$p \to (q \to p)$ $(p \to (q \to r)) \to (p \to q) \to (p \to r)$ $((p \to \bot) \to \bot) \to p)$
WL	Łukasiewicz 1948	159	$((p \to q) \to r) \to ((r \to p) \to (s \to p))$ $\bot \to p$
Axiomatics	Rasiowa 1949	157	$p \bigvee p \to p$ $p \to p \bigvee q$ $(p \to q) \to (r \bigvee p) \to (q \bigvee r)$
RB	Russell 1908, Bernays 1926	157	$p \bigvee p \to p$ $p \to p \bigvee q$ $p \bigvee q \to q \bigvee p$ $(p \to q) \to (r \bigvee p) \to (q \bigvee r)$
PM	Whitehead & Russell 1910	-	$p \bigvee p \to p$ $p \to q \bigvee p$ $p \bigvee q \to q \bigvee p$ $(p \bigvee (q \bigvee r)) \to (q \bigvee (p \bigvee r))$ $(p \to q) \to (r \bigvee p) \to (q \bigvee r)$

which we have formalized completeness. In this example we also show how to seamlessly use his notation in Isabelle and have the proof assistant translate it to the more familiar one. As another example we consider the exchangeability of two axioms.

We reproduce parts of our formalizations in the paper. The full Isabelle/HOL formalizations, 669 lines (535 sloc, source lines of code, not counting blank lines) in file `System_W.thy` and 631 lines (510 sloc) in file `System_R.thy`, are available here:

`https://github.com/logic-tools/axiom`

The paper strives to be self-contained so consulting the formalizations is optional. However, the availability enables the reader to investigate the formalizations on their own and, if curious, to look up anything we have omitted for reasons of space. The files can also be extended with other proof systems or taken as inspiration for different fragments of propositional logic or expansions to other logic. To verify properties of some new axiomatic system, it could be shown equivalent to one formalized here, so that soundness and completeness can be carried over. All of this with automation available to aid the process and a trusted kernel that guarantees correctness.

We modify our existing work [5] to formalize completeness of the axiomatic systems we consider. The existing completeness proof uses Henkin's synthetic technique based on maximal consistent sets of formulas to build a model for underivable formulas. We have adapted this proof to two representatives of the fragments we consider in this paper (the two systems dubbed *Axiomatics* in Table 1). A lot of this work involves showing that the proof system can derive certain formulas that are used in the completeness proof. Similarly, to reuse the completeness result for the other axiomatic systems for the same fragment, we show that certain formulas can be derived using their axioms. In short, much of this work is about proving that specific formulas can be derived. The classic book by Church [3] has been an excellent source for relevant formulas and instead of fiddling with instantiating the axioms ourselves and finding the right sequence of rule applications, we call upon Isabelle's tool *Sledgehammer* [2]. Often the built-in provers *meson* and *metis* can assemble the pieces for us.

Unfortunately, we are not always lucky enough to find a proof in the first attempt and *Sledgehammer* simply times out. We are then faced with a choice: either make a manual attempt to derive the formula or take a guess that some other formula should be derived first. We generally prefer the latter approach since it lets the proof assistant do more of the menial work for us, while leaving the more creative role of finding the right stepping stones to us.

In cases where we need to derive more than one formula to aid us, we typically mark each one of them with **sorry** before trying to prove them. This *fake proof* is accepted by Isabelle, so that *Sledgehammer* will pick up the lemma as usable in further derivations, but provides no guarantee that the formula can actually be derived. It saves time because we can make sure that the formulas marked by **sorry** are actually useful for our derivation before we try to find derivations for them in turn.

The paper is organized as follows. We continue with a discussion of the closest related work (§2). We move on to formalize the first three systems (§3) including the completeness of Łukasiewicz' single shortest axiom. We follow up by formalizing the remaining three systems (§4) for our other fragment of propositional logic and discuss historical concerns about the independence of certain axioms. Finally we describe the main challenges and benefits of using the proof assistant Isabelle/HOL (§5) and we conclude (§6) by placing our work in the context of the IsaFoL (Isabelle Formalization of Logic) project.

2 Related Work

We see two main pieces of related work explained below: that of Michaelis and Nipkow [9] and of Fitelson and Wos [4, 13]. We distinguish ourselves by considering completeness of a number of systems based on different primitives using the same approach.

- Michaelis and Nipkow [9] formalized a number of proof systems for propositional logic in Isabelle/HOL: resolution, natural deduction, sequent calculus and an axiomatic system. They used a much larger syntax with falsity, negation, conjunction, disjunction and implication. They both gave a syntactic completeness proof for the sequent calculus and showed completeness of the other systems by translations, but also showed completeness of the sequent calculus and axiomatic system with a Henkin-style [5] proof akin to ours. They only considered an axiomatic system similar to the Wajsberg axioms from 1939, where we consider the range of systems in Table 1. Their larger scope also means they go into fewer details than us, especially regarding the role of Isabelle in deriving formulas.
- Fitelson and Wos [4, 13] used the OTTER theorem prover to find axiomatic proofs for a range of formulas, similar to our use of Isabelle. They start from a clause with the disjunction of the negated Wajsberg 1937 axioms and a clause consisting of the Łukasiewicz 1948 axiom. Then they ask OTTER to derive the empty clause, causing it to derive each of the Wajsberg axioms along the way. We take a different approach and verify the correctness of the inference steps given by Łukasiewicz directly in Isabelle. Moreover, we show how to use Łukasiewicz's notation directly, instead of translating it into the clausal form of OTTER. Finally, Isabelle allows us to formalize semantics as well as proof systems.

We have recently [5] presented the details of a direct Henkin-style completeness proof for the Wajsberg axioms from 1939. In the present paper we elaborate on our use of derivations and equivalences instead of describing the Henkin-style completeness proof. We have used preliminary versions of our formalizations in the files `System_W.thy` and `System_R.thy` in our course on automated reasoning in 2020 and 2021 with, respectively, 27 and 37 MSc computer science students. The focus of the exercises was on our approach to formalization of syntax, semantics and axiomatic systems using Isabelle/HOL. As an introductory example we included a very brief description of the approach in our paper [6] about our main Isabelle/HOL tools for teaching logic, namely the Natural Deduction Assistant (NaDeA) and the Sequent Calculus Verifier (SeCaV), both much larger developments for first-order logic with functions.

3 Implication and Falsity

We start by considering Wajsberg's axioms for the fragment of propositional logic built from propositional symbols, implication and falsity.

3.1 Language

The following datatype *form* embeds our syntax into Isabelle:

datatype *form* = *Falsity* (‹⊥›) | *Pro nat* | *Imp form form* (**infix** ‹→› *0*)

Vertical bars separate the three constructors. The first one introduces $\bot$ as a primitive, the next one propositional symbols with natural numbers as identifiers and the final one is implication between two formulas, with the infix symbol $\to$.

Besides these primitive connectives, Isabelle allows us to introduce abbreviations as we would do with pen and paper. Here for the trivially true formula and for negation:

abbreviation *Truth* (‹⊤›) **where** ‹$\top \equiv (\bot \to \bot)$›
abbreviation (*input*) ‹$\mathit{Neg}\ p \equiv (p \to \bot)$›

To give our syntax meaning, we write a primitive recursion function in higher-order logic that uses an interpretation of the propositional symbols to map a formula into a truth value:

primrec *semantics* (**infix** ‹⊨› *0*) **where**
‹$(I \models \bot) = \mathit{False}$› |
‹$(I \models (\mathit{Pro}\ n)) = I\ n$› |
‹$(I \models (p \to q)) = (\mathit{if}\ I \models p\ \mathit{then}\ I \models q\ \mathit{else}\ \mathit{True})$›

We use Isabelle's *if-then-else* to interpret implication but we could also use the built-in higher-order logic implication ($\longrightarrow$).

We can define what it means for a formula to be valid by quantifying over all interpretations:

definition ‹$\mathit{valid}\ p \equiv \forall I.\ (I \models p)$›

3.2 Wajsberg 1937

Consider first Wajsberg's proof system from 1937 [3, p. 159]:

inductive *Axiomatics* (‹⊢›) **where**
‹$\vdash q$› **if** ‹$\vdash p$› **and** ‹$\vdash (p \to q)$› |
‹$\vdash (p \to (q \to p))$› |
‹$\vdash ((p \to q) \to ((q \to r) \to (p \to r)))$› |
‹$\vdash (((p \to q) \to p) \to p)$› |
‹$\vdash (\bot \to p)$›

The $\vdash$ predicate holds for a given formula if it can be derived from the specified rule and axioms. Notably, the axioms are schemas, where p and q can be instantiated for any formula. The only rule, here and later, is modus ponens (*MP*). The first axiom (*Imp1*) corresponds to the K combinator, the second (*Tran*) expresses transitivity of implication and the third (*Clas*), Peirce's law, implies the law of the excluded middle. Finally we have the principle of explosion (*Expl*).

As an example, we can derive $\top$ from this last axiom:

theorem ‹$\vdash \top$› **using** *Axiomatics.intros(5)* **.**

Isabelle automatically instantiates the given axiom correctly.

Since we have specified the meaning of our formulas in Isabelle, we can verify the soundness of the proof system:

```
theorem soundness: ‹⊢ p ⟹ I ⊨ p›
  by (induct rule: Axiomatics.induct) auto
```

The proof works by induction over the proof system, an induction principle that Isabelle automatically provides. The proof method *auto* discharges each of the resulting proof obligations. Such checks are cheap and easy in Isabelle, helping to prevent typos or other mistakes.

Completeness follows the synthetic recipe due to Henkin and, together with soundness, results in the following theorem:

```
theorem main: ‹valid p = ⊢ p›
proof
  assume ‹valid p›
  with completeness show ‹⊢ p›
    unfolding valid-def .
next
  assume ‹⊢ p›
  with soundness show ‹valid p›
    unfolding valid-def by (intro allI)
qed
```

The proof is shown in its entirety to showcase features of the Isabelle syntax.

3.3 Wajsberg 1939

Consider a later proof system by Wajsberg with different axioms [3, p. 163]:

```
inductive FW (‹⊩›) where
  ‹⊩ q› if ‹⊩ p› and ‹⊩ (p → q)› |
  ‹⊩ (p → (q → p))› |
  ‹⊩ ((p → (q → r)) → ((p → q) → (p → r)))› |
  ‹⊩ (((p → ⊥) → ⊥) → p)›
```

We still have the *Imp1* axiom corresponding to the K combinator, but as second axiom we now have a correspondence to the S combinator (both axioms used by Frege). Finally, with the abbreviation for *Neg*, we see that this last axiom eliminates a double negation.

We can now verify that the two systems prove the same formulas:

```
theorem Axiomatics-FW: ‹⊢ p ⟷ ⊩ p›
proof
  have *: ‹⊩ ((p → q) → ((q → r) → (p → r)))› for p q r
    by (metis FW.intros(1–3))
  then have **: ‹⊩ (((p → q) → p) → p)› for p q
    by (metis FW.intros(1–4))
  show ‹⊢ p› if ‹⊩ p›
    using that by induct (use Imp1 Imp2 Neg Axiomatics.intros in meson)+
  show ‹⊩ p› if ‹⊢ p›
    using that by induct (use * ** FW.intros in meson)+
qed
```

As part of the proof we find derivations for the transitivity principle (*Tran*) and Peirce's law in the latter system.

3.4 Shortest Axiom

Considering the fragment of classical logic with implication but without a symbol for falsity, Łukasiewicz found a shortest single axiom from which you can derive the rest [12].

To obtain completeness for our fragment with a symbol for falsity, we also need the principle of explosion [3, p. 159]:

```
inductive WL (‹≫›) where
  ‹≫ q› if ‹≫ p› and ‹≫ (p → q)› |
  ‹≫ (((p → q) → r) → ((r → p) → (s → p)))› |
  ‹≫ (⊥ → p)›
```

Łukasiewicz writes $C\ p\ q$ for $p \to q$. This prefix notation allows him to avoid parentheses. We can use it in Isabelle via the following specification:

```
abbreviation (input) C :: ‹form ⇒ form ⇒ form› (‹C - -› [0, 0] 1) where
  ‹(C p q) ≡ (p → q)›
```

We set the symbol C up with the mixfix specification $[0, 0],\ 1$, giving the two arguments higher precedence (0) than the full expression (1). This means that e.g. $C\ C\ p\ q\ r$ is parsed correctly into $(p \to q) \to r$. Since we specified the abbreviation as *input* only, any Isabelle output will display the formulas in the conventional $\to$-notation.

Łukasiewicz shows in 29 lines how to derive the Wajsberg axioms (*Axiomatics* above). With our abbreviation we reproduce his derivations almost verbatim in figures 1 and 2 on pages 14 and 15. The formalization follows the original faithfully: each line is only derived from the specified lines and modus ponens as passed to the *meson* prover. Łukasiewicz carefully describes how to instantiate each previous formula in order to arrive at the current formula but we leave this to Isabelle to figure out. While Łukasiewicz's paper must be hand-checked to ensure there are no errors, Isabelle instantly verifies the correctness of our formalization. Given those derivations we can prove equivalence between this proof system and the first Wajsberg axioms:

```
theorem equivalence: ‹≫ p ⟷ ⊢ p›
proof
  have *: ‹⊢ (((p → q) → r) → ((r → p) → (s → p)))› for p q r s
    using completeness by simp
```

  show ‹$\vdash p$› **if** ‹$\gg p$›
    using *that* **by** *induct* (*auto simp*: $*$ *intro*: *Axiomatics.intros*)
  show ‹$\gg p$› **if** ‹$\vdash p$›
    using *that* **by** *induct* (*auto simp*: *l27 l28 l29 intro*: *WL.intros*)
qed

We use the completeness of the Wajsberg axioms to show that Łukasiewicz's formula can be derived. In the other direction we use the formulas in lines 27–29 of figure 2.

4 Disjunction and Negation

We now wipe the slate clean and consider Rasiowa's axioms for a different fragment of propositional logic built from propositional symbols, negation ($\neg$) and disjunction ($\bigvee$).

4.1 Language

Again we specify the syntax as a datatype in Isabelle:

datatype *form* = *Pro nat* | *Neg form* | *Dis form form* (**infix** ‹$\bigvee$› *0*)

We regain implication through its classical interpretation:

abbreviation *Imp* (**infix** ‹$\rightarrow$› *0*) **where** ‹$(p \rightarrow q) \equiv (\mathit{Neg}\ p \bigvee q)$›

We again define the trivially true formula, this time more abstractly since we no longer have $\bot$ available (in Isabelle/HOL, by formulation, each type has one designated value that is *undefined* but we do not know which value it is):

abbreviation *Truth* (‹$\top$›) **where** ‹$\top \equiv (\mathit{undefined} \rightarrow \mathit{undefined})$›

Given $\top$, however, defining $\bot$ becomes simple:

abbreviation *Falsity* (‹$\bot$›) **where** ‹$\bot \equiv \mathit{Neg}\ \top$›

We specify the semantics similarly to before:

primrec *semantics* (**infix** ‹$\models$› *0*) **where**
  ‹$(I \models \mathit{Pro}\ n) = I\ n$› |
  ‹$(I \models \mathit{Neg}\ p) = (\mathit{if}\ I \models p\ \mathit{then}\ \mathit{False}\ \mathit{else}\ \mathit{True})$› |
  ‹$(I \models (p \bigvee q)) = (\mathit{if}\ I \models p\ \mathit{then}\ \mathit{True}\ \mathit{else}\ (I \models q))$›

4.2 Rasiowa 1949

Consider the following proof system by Rasiowa [3, p. 157]:

inductive *Axiomatics* (‹⊢›) **where**
  ‹$\vdash q$› **if** ‹$\vdash p$› **and** ‹$\vdash (p \to q)$› |
  ‹$\vdash ((p \bigvee p) \to p)$› |
  ‹$\vdash (p \to (p \bigvee q))$› |
  ‹$\vdash ((p \to q) \to ((r \bigvee p) \to (q \bigvee r)))$›

To aid readability we write the rules using the abbreviation for implication introduced above ($p \to q \equiv \mathit{Neg}\ p \bigvee q$), but we recall that it is not a primitive. If we expand the abbreviation for the modus ponens rule (*MP*), it infers $\vdash q$ from $\vdash p$ and $\vdash \neg p \bigvee q$.

The first axiom (*Idem*) expresses idempotence of disjunction. The second (*AddR*) builds a disjunction from a given formula by adding an arbitrary formula on the right-hand side. Finally, the last axiom (*Swap*) does two things: it replaces the formula on right-hand side of the disjunction with an implied formula and then it swaps the two sides of the disjunction.

The principle of explosion is not a built-in axiom but Isabelle can quickly find a derivation:

theorem ‹$\vdash (\bot \to p)$› **using** *Axiomatics.intros* **by** *metis*

We can just as quickly verify the soundness:

theorem *soundness*: ‹$\vdash p \Longrightarrow I \models p$›
  by (*induct rule*: *Axiomatics.induct*) *auto*

The axiom *AddR* forms a disjunction with the given formula on the left and an arbitrary one on the right. We might wonder if this is essential or whether we could add the arbitrary formula on the left instead (i.e. *AddL*). Isabelle can help answer this question:

proposition *alternative-axiom*: ‹$\vdash (p \to (p \bigvee q))$› **if** ‹$\bigwedge p\ q.\ \vdash (p \to (q \bigvee p))$›
  by (*metis MP Idem Swap that*)

We see that *AddR* can be derived from *AddL* (in Isabelle given after **if** and referred to as *that*) alongside the remaining proof system. Note that *AddR* is not made available to *metis*.

Likewise, we can derive *AddL* from the full proof system:

lemma *AddL*: ‹$\vdash (p \to (q \bigvee p))$›
  by (*metis MP Idem Swap AddR*)

Thus, we can quickly answer questions about different variants of the axioms.

A notable derivable formula is the following that substitutes a formula on the right-hand side of a disjunction with an implied formula:

```
lemma SubR: ‹⊢ ((p → q) → ((r ⋁ p) → (r ⋁ q)))›
  by (meson MP SwapCon Swap)
```

Again, we can prove the completeness of the system:

```
theorem main: ‹valid p = ⊢ p›
  (proof omitted)
```

4.3 Russell 1908 & Bernays 1926

Consider now another proof system over the same fragment [3, p. 157]:

```
inductive RB (‹⊩›) where
  ‹⊩ q› if ‹⊩ p› and ‹⊩ (p → q)› |
  ‹⊩ ((p ⋁ p) → p)› |
  ‹⊩ (p → (q ⋁ p))› |
  ‹⊩ ((p ⋁ q) → (q ⋁ p))› |
  ‹⊩ ((p → q) → ((r ⋁ p) → (r ⋁ q)))›
```

Here we have first *Idem* and *AddL*, then a permutation or commutativity principle for disjunction (*Perm*) and finally *SubR*. We only need the derived *SubR* to show equivalence:

```
theorem Axiomatics-RB: ‹⊢ p ⟷ ⊩ p›
proof
  show ‹⊢ p› if ‹⊩ p›
    using that by induct (use SubR Axiomatics.intros in meson)+
  show ‹⊩ p› if ‹⊢ p›
    using that by induct (use RB.intros in meson)+
qed
```

4.4 Whitehead & Russell 1910

Consider next the system for propositional logic that appears in the first volume of the three-volume *Principia Mathematica* (often abbreviated PM):

```
inductive PM (‹≫›) where
  ‹≫ q› if ‹≫ p› and ‹≫ (p → q)› |
  ‹≫ ((p ⋁ p) → p)› |
  ‹≫ (p → (q ⋁ p))› |
  ‹≫ ((p ⋁ q) → (q ⋁ p))› |
  ‹≫ ((p ⋁ (q ⋁ r)) → (q ⋁ (p ⋁ r)))› |
  ‹≫ ((p → q) → ((r ⋁ p) → (r ⋁ q)))›
```

Here we have *Idem*, *AddL*, *Perm*, a distributivity principle and *SubR*. We can easily show that we can derive at least as many formulas when we have the extra axiom:

```
proposition PM-extends-RB: ‹⊩ p ⟹ ≫ p›
  by (induct rule: RB.induct) (auto intro: PM.intros)
```

To show the equivalence in both directions, we use the completeness of *RB* to prove the existence of a derivation for the extra axiom:

```
theorem equivalence: ‹≫ p ⟷ ⊢ p›
proof
  have ∗: ‹⊢ ((p ⋁ (q ⋁ r)) → (q ⋁ (p ⋁ r)))› for p q r
    using completeness by simp
  show ‹⊢ p› if ‹≫ p›
    using that by induct (use ∗ SubR Axiomatics.intros in meson)+
  show ‹≫ p› if ‹⊢ p›
    using that by induct (use PM.intros in meson)+
qed
```

5 Challenges and Benefits

Isabelle helped enormously in adapting the completeness proof to each of the two fragments, since it is easy to define abbreviations for non-primitive connectives and the proof assistant gives an error everywhere something needs to be changed. What the proof assistant cannot tell us, is what sub-derivations are needed to derive a key formula. This proved a particular challenge for the Rasiowa axioms for the fragment $\neg, \bigvee$. Those axioms are concerned with these two operators but we usually think in terms of implication, $\to$, and want to derive formulas like the Wajsberg 1937/1939 axioms (cf. Table 1). However, the starting point does not give us much help. For instance, to derive the following transitivity of implication, a useful lemma for further derivations, we must first derive several other formulas:

lemma *Tran*: ‹$\vdash ((p \to q) \to ((q \to r) \to (p \to r)))$›

One of the formulas we found useful is the following, somewhat unintuitive, *SwapAnte* lemma:

lemma *SwapAnte*: ‹$\vdash (((p \bigvee q) \to r) \to ((q \bigvee p) \to r))$›

Luckily, Isabelle makes it easy to quickly derive a range of formulas (or pretend to derive them with **sorry**) and figure out which ones are useful after the fact. As we have shown throughout the paper, this allows us to quickly investigate connections between various proof systems, with Isabelle keeping track of the details for us.

6 Conclusion

We have seen two languages and a range of axiomatic proof systems with various derivations and equivalences. To our knowledge, most of the systems have been formalized here for the first time with soundness and completeness proofs.

Our work is part of the IsaFoL (Isabelle Formalization of Logic) project [1] which aims at developing formalizations in Isabelle/HOL of logics, proof systems, and automatic/interactive provers. Other work in the same line includes completeness of epistemic [7] and hybrid [8] logic and an ordered resolution prover for first-order logic [11]. The project collects formalizations such as ours that can be used as reference documents to verify historical claims, in teaching logic, or to aid in the formalization of other logics and potentially executable provers. Our own formalization could serve as starting point for a student project to formalize the completeness of some other axiomatic proof system.

A notable thing about our approach is that while we show that several formulas are derivable in one system or another, we do not give the derivation itself. Instead, we let an automated prover like *metis* or *meson* find it. This allows us to move quickly and at a higher level than if we spelled out each step in full: faced with a formula that is hard to derive we can experiment with simpler formulas that the automation can handle and try to piece things together afterwards. As mentioned, this was exactly how we worked to derive many of the formulas. However, it also means that even if we prove that a formula can be derived, we have no derivation to inspect; we must simply trust Isabelle that it exists. Meanwhile, we argue that Isabelle is at least as trustworthy as a human author whose work we might not check in the first place. If we do wish to spell out the derivation, Isabelle can help us do so, by proving that derivations exist for our stepping stones.

The ability to introduce abbreviations can provide interesting perspectives on formulas. Consider the usual axiom for disjunction elimination:

lemma *DisE*: ‹$\vdash ((p \to r) \to ((q \to r) \to ((p \bigvee q) \to r)))$›

We might think of it as "if both p and q imply r, then if we know either then we know r." In the language with disjunction and negation, the implication is an abbreviation and expanding the inner ones gives us:

$$(\neg p \bigvee r) \to (\neg q \bigvee r) \to (p \bigvee q) \to r$$

This has another natural reading: "either p is false or r holds, and either q is false or r holds, but either p or q is in fact true, so r must hold." An interactive system like Isabelle makes it simple to hide away details like the abbreviation for implication but also to peek at them if we want to.

Acknowledgements

We thank Alexander Birch Jensen, Frederik Krogsdal Jacobsen, Osman Hasan and the anonymous reviewers for comments on drafts.

References

1. 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, Cascais, Portugal, January 14-15, 2019. pp. 1–13. ACM (2019)

2. Blanchette, J.C., Böhme, S., Paulson, L.C.: Extending sledgehammer with SMT solvers. J. Autom. Reason. **51**(1), 109–128 (2013). https://doi.org/10.1007/s10817-013-9278-5
3. Church, A.: Introduction to Mathematical Logic. Princeton: Princeton University Press (1956)
4. Fitelson, B., Wos, L.: Finding missing proofs with automated reasoning. Studia Logica **68**(3), 329–356 (2001). https://doi.org/10.1023/A:1012486904520
5. From, A.H.: Formalizing Henkin-style completeness of an axiomatic system for propositional logic. In: Proceedings of the Web Summer School in Logic, Language and Information (WeSSLLII) and the European Summer School in Logic, Language and Information (ESSLLI) Virtual Student Session (2020), pages 1–12, preliminary paper, `https://www.brandeis.edu/nasslli2020/pdfs/student-session-proceedings-compressed.pdf\#page=8`, accepted for Springer post-proceedings.
6. From, A.H., Villadsen, J., Blackburn, P.: Isabelle/HOL as a meta-language for teaching logic. In: Quaresma, P., Neuper, W., Marcos, J. (eds.) Proceedings 9th International Workshop on Theorem Proving Components for Educational Software, ThEdu@IJCAR 2020, Paris, France, 29th June 2020. EPTCS, vol. 328, pp. 18–34 (2020). https://doi.org/10.4204/EPTCS.328.2
7. From, A.H.: Epistemic logic: Completeness of modal logics. Archive of Formal Proofs (Oct 2018), `https://devel.isa-afp.org/entries/Epistemic_Logic.html`, Formal proof development
8. From, A.H.: Formalizing a Seligman-style tableau system for hybrid logic. Archive of Formal Proofs (Dec 2019), `https://isa-afp.org/entries/Hybrid_Logic.html`, Formal proof development
9. Michaelis, J., Nipkow, T.: Formalized proof systems for propositional logic. In: Abel, A., Forsberg, F.N., Kaposi, A. (eds.) 23rd International Conference on Types for Proofs and Programs, TYPES 2017, May 29-June 1, 2017, Budapest, Hungary. LIPIcs, vol. 104, pp. 5:1–5:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
10. 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)
11. Schlichtkrull, A., Blanchette, J., Traytel, D., Waldmann, U.: Formalizing Bachmair and Ganzinger's ordered resolution prover. J. Autom. Reason. **64**(7), 1169–1195 (2020)
12. Łukasiewicz, J.: The shortest axiom of the implicational calculus of propositions. Proceedings of the Royal Irish Academy. Section A: Mathematical and Physical Sciences **52**, 25–33 (1948)
13. Wos, L., Pieper, G.W.: Automated Reasoning and the Discovery of Missing and Elegant Proofs. Rinton Press (2003)

```
lemma l1: ‹≫ (C C C p q r C C r p C s p)›
  using WL.intros(2) .

lemma l2: ‹≫ (C C C C r p C s p C p q C r C p q)›
  using l1 by (meson WL.intros(1))

lemma l3: ‹≫ (C C C r C p q C C r p C s p C t C C r p C s p)›
  using l1 l2 by (meson WL.intros(1))

lemma l4: ‹≫ (C C C p q p C s p)›
  using l3 l1 by (meson WL.intros(1))

lemma l5: ‹≫ (C C C s p C p q C r C p q)›
  using l1 l4 by (meson WL.intros(1))

lemma l6: ‹≫ (C C C r C p q C s p C t C s p)›
  using l1 l5 by (meson WL.intros(1))

lemma l7: ‹≫ (C C C t C s p C r C p q C u C r C p q)›
  using l1 l6 by (meson WL.intros(1))

lemma l8: ‹≫ (C C C s q p C q p)›
  using l7 l1 by (meson WL.intros(1))

lemma l9: ‹≫ (C r C C r p C s p)›
  using l8 l1 by (meson WL.intros(1))

lemma l10: ‹≫ (C C C C C r q p C s p r C t r)›
  using l1 l9 by (meson WL.intros(1))

lemma l11: ‹≫ (C C C t r C C C r q p C s p C u C C C r q p C s p)›
  using l1 l10 by (meson WL.intros(1))

lemma l12: ‹≫ (C C C u C C C r q p C s p C t r C v C t r)›
  using l1 l11 by (meson WL.intros(1))

lemma l13: ‹≫ (C C C v C t r C u C C C r q p C s p C w C u C C C r q p C s p)›
  using l1 l12 by (meson WL.intros(1))

lemma l14: ‹≫ (C C C t r C s p C C C r q p C s p)›
  using l13 l1 by (meson WL.intros(1))

lemma l15: ‹≫ (C C C r q C s p C C r p C s p)›
  using l14 l1 by (meson WL.intros(1))
```

Fig. 1. Lines 1–15 of Łukasiewicz's derivation.

```
lemma l16: ‹≫ (C C r C s p C C C r q p C s p)›
  using l15 l9 by (meson WL.intros(1))

lemma l17: ‹≫ (C C C C C p q r t C s p C C r p C s p)›
  using l16 l1 by (meson WL.intros(1))

lemma l18: ‹≫ (C C C C r p C s p C C C p q r t C u C C C p q r t)›
  using l1 l17 by (meson WL.intros(1))

lemma l19: ‹≫ (C C C C s p q C r p C C C p q r C s p)›
  using l18 by (meson WL.intros(1))

lemma l20: ‹≫ (C C C C r p p C s p C C C p q r C s p)›
  using l14 l19 by (meson WL.intros(1))

lemma l21: ‹≫ (C C C C p r q q C C q r C p r)›
  using l20 l15 by (meson WL.intros(1))

lemma l22: ‹≫ (C p p)›
  using l5 l4 by (meson WL.intros(1))

lemma l23: ‹≫ (C C C p q r C C r p p)›
  using l20 l22 by (meson WL.intros(1))

lemma l24: ‹≫ (C r C C r p p)›
  using l8 l23 by (meson WL.intros(1))

lemma l25: ‹≫ (C C p q C C C p r q q)›
  using l15 l24 by (meson WL.intros(1))

lemma l26: ‹≫ (C C C C p q C C q r C p r C C C p r q q C C C p r q q)›
  using l25 by (meson WL.intros(1))

lemma l27: ‹≫ (C p C q p)›
  using l8 by (meson WL.intros(1))

lemma l28: ‹≫ (C C C p q p p)›
  using l25 l22 by (meson WL.intros(1))

lemma l29: ‹≫ (C C p q C C q r C p r)›
  using l21 l26 by (meson WL.intros(1))
```

Fig. 2. Lines 16–29 of Łukasiewicz's derivation.