A Sequent Calculus for First-Order Logic Formalized in Isabelle/HOL [⋆]

Asta Halkjær From[1][0000−0002−3601−0804],
Anders Schlichtkrull[2][0000−0001−9212−6150], and
Jørgen Villadsen[1][0000−0003−3624−1159]

[1] Technical University of Denmark, Kongens Lyngby, Denmark
{ahfrom,jovi}@dtu.dk
[2] Aalborg University Copenhagen, Copenhagen, Denmark andsch@cs.aau.dk

Abstract. We formalize in Isabelle/HOL soundness and completeness of a one-sided sequent calculus for first-order logic. The completeness is shown via a translation from a semantic tableau calculus, whose completeness proof we base on the theory entry "First-Order Logic According to Fitting" by Berghofer in the Archive of Formal Proofs (AFP). The calculi and proof techniques are taken from Ben-Ari's textbook *Mathematical Logic for Computer Science* (Springer 2012). We thereby demonstrate that Berghofer's approach works not only for natural deduction but constitutes a framework for mechanically-checked completeness proofs for a range of proof systems.

1 Introduction

We formalize in the proof assistant Isabelle/HOL [18] the soundness and completeness proofs of a tableau calculus as well as a sequent calculus for first-order logic with functions. We thereby also establish semantic cut-elimination since our sequent calculus is cut-free. Our formalization is a contribution to the meta-program of IsaFoL (Isabelle Formalization of Logic) [3]:

> *IsaFoL (Isabelle Formalization of Logic) is an undertaking that aims at developing formal theories about logics, proof systems, and automatic provers, using Isabelle/HOL.*
>
> *At the heart of the project is the conviction that proof assistants have become mature enough to actually help researchers in automated reasoning when they develop new calculi and tools.*

Along with the resolution calculus, the sequent calculus is a central topic in automated reasoning but we do not present an automatic prover. We follow the soundness and completeness proofs in the textbook on mathematical logic by Ben-Ari [1]. Our formalization of the soundness and completeness proofs (1000+ lines) is available online in the *Archive of Formal Proofs* [10].

In recent years we have used our formalization for teaching logic in computer science and described this use in a number of papers [11–13]. These report on teaching logic for hundreds of students but this is not the topic of our present paper. Also, these papers focus on derivations in the sequent calculus and how Isabelle/HOL verifies these derivations of first-order formulas. That is, unlike the present paper, none of these papers detail the formalization of the soundness and completeness proofs for the sequent calculus or the tableau calculus.

We find that for teaching, the one-sided sequent calculus is a good starting point, especially if the students have previously worked with tableau calculi; the traditional two-sided sequent calculus can be covered afterwards if necessary. Without a formalization, students and researchers have to manually recheck the soundness and completeness proofs if they experiment with adding or removing rules of the calculus. With our work, they can instead let Isabelle/HOL show which proofs need to be changed in order to account for the new set of rules and use Isabelle/HOL to help them with updating these proofs.

Our formalization approach is slightly unusual in that we have started from the formalization of natural deduction by Berghofer (4000+ lines) [2]. Except for comments in his Isabelle/HOL files, Berghofer has not published about this and we have extended the formalized result to cover open formulas (around 1000 of the 4000+ lines). Our approach shows how Berghofer's work can be used as a framework for formalizing logical systems. It is common in mathematics and computer science to build on the work of others by using their theorems and lemmas, but this has the danger that assumptions may be forgotten and results may be applied incorrectly. When we use Berghofer's result here, there is no such danger, because Isabelle keeps track of all of this. Along the lines of Ben-Ari [1] our proofs work by exploring the relationships between a tableau calculus and the sequent calculus.

We start by discussing related work on formalizing sequent calculi in Section 2. Then we briefly review Berghofer's formalization of natural deduction in Section 3 and our extension to open formulas in Section 4. In Section 5 we prove soundness and completeness of our tableau calculus and continue with open formulas in Section 6. We prove soundness and completeness of our sequent calculus in Section 7, using the completeness result for our tableau calculus. Lastly, we conclude with thoughts about future work in Section 8.

2 Related Work

There are other formalizations of sequent calculi in proof assistants, all of which differ from the one we present. We discuss them here.

One is in the supplementary work of a paper by Blanchette and Popescu [4], but the paper itself does not describe this development. In the supplementary work the authors formalize in Isabelle/HOL a tableau calculus for many-sorted first-order logic for formulas in negation normal form. This supplementary material is unfortunately not up to date with recent Isabelle versions. The advantages of our formalization are that it works in the newest version of Isabelle and that it is not limited to formulas in negation normal form. We expect that it will be easy to keep our development up to date with new versions of Isabelle, because our formalization stays within the usual features of Isabelle such as its default proof language Isar, default proof methods methods/tactics and default tools for defining types, constants and functions. Furthermore our formalization is part of the Archive of Formal Proofs in which Isabelle developers keep the entries up to date with new releases of Isabelle. E.g. Berghofer's natural deduction formalization has been kept up to date with new Isabelle releases since 2007. On the other hand, the advantages of Blanchette and Popescu's formalization are that it supports many-sorted first-order logic with equality whereas ours supports only plain first-order logic. This limitation of our work is a consequence of using Berghofer's natural deduction formalization which is also based on this choice. Another difference is that Blanchette and Popescu base their formalization on a framework [5] for proving logical calculi complete using so-called analytic or syntactic completeness proofs. Such proofs work by using failed attempts at deriving an unprovable formula to construct a countermodel. We instead base our formalization on Berghofer's formalization of synthetic completeness adopted from a book by Fitting [8]. In the synthetic approach, completeness is proved by constructing maximal consistent sets of formulas and showing that formulas in such sets have a model. Instead of reasoning about failed proof attempts, including concerns of fairness, we reason about the simple process of extending a consistent set of formulas to be maximally consistent. Thus, we prove completeness in an entirely different way than Blanchette and Popescu.

Another formalization of a sequent calculus is by Ridge and Margetson [20]. Like Blanchette and Popescu they use the analytic approach, and their calculus is limited to negation normal form without equality. Additionally, their term language consists exclusively of variables rather than full first-order terms.

Braselmann and Koepke [6] also formalized a sequent calculus for first-order logic without equality, but they used Mizar rather than Isabelle/HOL. They base their work on the textbook by Ebbinghaus, Flum and Thomas [7] which like our work employs the synthetic approach. Their formalization uses Mizar's Tarski-Grothendieck set theory as the metalogic whereas we have Isabelle/HOL's higher-order logic as metalogic. In their work they have a definition of proofs as lists of sequents whereas we define derivability directly as an inductive predicate.

A more exotic result is Ilik's formalization [16], in Coq, of the completeness of a sequent calculus with respect to a Kripke-semantics for classical first-order logic [17]. This is in contrast to the other works presented here which use the usual semantics for first-order logic. Lastly, we mention here some results from intuitionistic logic, namely Persson's formalization [19] of the soundness of intuitionistic first-order logic, and Herbelin, Kim and Lee's formalization [15] of soundness and completeness of intuitionistic first-order logic with respect to Kripke models for a first-order logic with only universal quantification and implication.

3 Natural Deduction

Our point of departure is Berghofer's formalization of natural deduction [2] based on Fitting's presentation of the completeness of first-order logic [8]. In his formalization, Berghofer first defines the syntax and semantics of first-order logic. He then defines a natural deduction proof system and proves it sound. He goes on to prove the model existence theorem and thereafter proves completeness.

Our formalization uses the same definition of first-order logic's syntax and semantics, and we also use the model existence theorem to prove completeness. We therefore briefly review Berghofer's formalization in this section.

3.1 FOL Syntax and Semantics

Berghofer formalizes terms using a simple datatype with one constructor, *Var*, for variables and one, *App*, for composite terms:

```
datatype 'a term
  = Var nat
  | App 'a ‹'a term list›
```

Variables are represented by natural numbers because Berghofer is using the so-called de Bruijn notation. In the de Bruijn notation the idea is that a variable *Var i* exists in the scope of a number of quantifiers. Of these quantifiers, *Var i* is bound by the one that has the ith innermost of these scopes, counting from 0. In case no such scope exists, the variable is considered free.

Berghofer formalizes formulas as a datatype:

```
datatype ('a, 'b) form
  = FF
  | TT
  | Pred 'b ‹'a term list›
  | And ‹('a, 'b) form› ‹('a, 'b) form›
  | Or ‹('a, 'b) form› ‹('a, 'b) form›
  | Impl ‹('a, 'b) form› ‹('a, 'b) form›
  | Neg ‹('a, 'b) form›
  | Forall ‹('a, 'b) form›
  | Exists ‹('a, 'b) form›
```

FF represents $\bot$ (falsity), *TT* represents $\top$ (truth), *Pred p ts* represents the predicate $p(ts_1, \ldots, ts_n)$, *And* represents $\wedge$ (conjunction), *Or* represents $\vee$ (disjunction), *Impl* represents $\rightarrow$ (implication), *Neg* represents $\neg$ (negation), *Forall* represents $\forall$ (universal quantification) and *Exists* represents $\exists$ (existential quantification). Notice that the quantifiers contain a subformula but no explicit binding of a variable symbol – this is because with the de Bruijn notation an explicit binding would not be meaningful.

Berghofer defines the semantics of terms and lists of terms by mutual recursion on these:

```
primrec
  evalt :: ⟨(nat ⇒ 'c) ⇒ ('a ⇒ 'c list ⇒ 'c) ⇒ 'a term ⇒ 'c⟩ and
  evalts :: ⟨(nat ⇒ 'c) ⇒ ('a ⇒ 'c list ⇒ 'c) ⇒ 'a term list ⇒ 'c list⟩ where
  ⟨evalt e f (Var n) = e n⟩
| ⟨evalt e f (App a ts) = f a (evalts e f ts)⟩
| ⟨evalts e f [] = []⟩
| ⟨evalts e f (t # ts) = evalt e f t # evalts e f ts⟩
```

Here, e is a variable denotation and f is a function denotation.

Berghofer defines the semantics of formulas by recursion on the formula datatype:

```
primrec eval :: ⟨(nat ⇒ 'c) ⇒ ('a ⇒ 'c list ⇒ 'c) ⇒
  ('b ⇒ 'c list ⇒ bool) ⇒ ('a, 'b) form ⇒ bool⟩ where
  ⟨eval e f g FF = False⟩
| ⟨eval e f g TT = True⟩
| ⟨eval e f g (Pred a ts) = g a (evalts e f ts)⟩
| ⟨eval e f g (And p q) = ((eval e f g p) ∧ (eval e f g q))⟩
| ⟨eval e f g (Or p q) = ((eval e f g p) ∨ (eval e f g q))⟩
| ⟨eval e f g (Impl p q) = ((eval e f g p) ⟶ (eval e f g q))⟩
| ⟨eval e f g (Neg p) = (¬ (eval e f g p))⟩
| ⟨eval e f g (Forall p) = (∀ z. eval (e⟨0:z⟩) f g p)⟩
| ⟨eval e f g (Exists p) = (∃ z. eval (e⟨0:z⟩) f g p)⟩
```

Here, e is a variable denotation, f is a function denotation and g is a predicate denotation. Since Berghofer is formalizing the object logic FOL in the meta-logic HOL, he can use HOL's *True*, *False*, $\wedge$, $\vee$, $\longrightarrow$, $\neg$, $\forall$ and $\exists$ when defining the semantics of FOL. The notation $e\langle 0 : z\rangle$ represents the variable denotation $\{0 \mapsto z, 1 \mapsto e\ 0, 2 \mapsto e\ 1, \ldots\}$.

Berghofer also formalizes what it means for a formula p to be a consequence of a list of formulas *ps* with respect to a variable denotation, a function denotation and a predicate denotation:

```
definition model :: ⟨(nat ⇒ 'c) ⇒ ('a ⇒ 'c list ⇒ 'c) ⇒ ('b ⇒ 'c list ⇒ bool) ⇒
   ('a, 'b) form list ⇒ ('a, 'b) form ⇒ bool⟩ (-,-,-,- ⊨ - [50,50] 50) where
  ⟨(e,f,g,ps ⊨ p) = (list-all (eval e f g) ps ⟶ eval e f g p)⟩
```

3.2 Berghofer's Natural Deduction System

Berghofer formalizes a natural deduction system as an inductive predicate consisting of a number of rules.

```
inductive deriv :: ⟨('a, 'b) form list ⇒ ('a, 'b) form ⇒ bool⟩ (- ⊢ - [50,50] 50) where
  Assum: ⟨a ∈ set G ⟹ G ⊢ a⟩
| TTI: ⟨G ⊢ TT⟩
| FFE: ⟨G ⊢ FF ⟹ G ⊢ a⟩
| NegI: ⟨a # G ⊢ FF ⟹ G ⊢ Neg a⟩
| NegE: ⟨G ⊢ Neg a ⟹ G ⊢ a ⟹ G ⊢ FF⟩
| Class: ⟨Neg a # G ⊢ FF ⟹ G ⊢ a⟩
| AndI: ⟨G ⊢ a ⟹ G ⊢ b ⟹ G ⊢ And a b⟩
| AndE1: ‹G ⊢ And a b ⟹ G ⊢ a›
| AndE2: ‹G ⊢ And a b ⟹ G ⊢ b›
| OrI1: ‹G ⊢ a ⟹ G ⊢ Or a b›
| OrI2: ‹G ⊢ b ⟹ G ⊢ Or a b›
| OrE: ‹G ⊢ Or a b ⟹ a # G ⊢ c ⟹ b # G ⊢ c ⟹ G ⊢ c›
| ImplI: ‹a # G ⊢ b ⟹ G ⊢ Impl a b›
| ImplE: ‹G ⊢ Impl a b ⟹ G ⊢ a ⟹ G ⊢ b›
| ForallI: ‹G ⊢ a[App n []/0] ⟹ list-all (λp. n ∉ params p) G ⟹
    n ∉ params a ⟹ G ⊢ Forall a›
| ForallE: ‹G ⊢ Forall a ⟹ G ⊢ a[t/0]›
| ExistsI: ‹G ⊢ a[t/0] ⟹ G ⊢ Exists a›
| ExistsE: ‹G ⊢ Exists a ⟹ a[App n []/0] # G ⊢ b ⟹
    list-all (λp. n ∉ params p) G ⟹ n ∉ params a ⟹ n ∉ params b ⟹ G ⊢ b›
```

For example, the rule *ForallI* would be the following rule in a more conventional style for writing out proof rules:

$$\frac{G \vdash a[n/0] \quad n \text{ does not occur in } G\text{, nor in } a}{G \vdash \forall.a}$$

Here $a[n/0]$ represents substitution of variable 0 with the constant (0-ary composite term) n in a. The rest of the rules are built in a similar way, so we will not write them out here.

3.3 Soundness of Natural Deduction

Berghofer proves natural deduction sound:

```
theorem correctness: ‹G ⊢ p ⟹ ∀ e f g. e,f,g,G ⊨ p›
```

The proof is by induction on the rules of the proof system. Berghofer's original proof was in apply-style, but the current version of the proof by From is an Isar-style proof in which all cases are handled by the automation of Isabelle except for *ForallI* and *ExistsE*. However, these cases are not too difficult.

3.4 Model Existence and Consistency Properties

In order to prove model existence, Fitting and Berghofer first define the notion of what it means for a set of sets of formulas to be a consistency property:

```
definition consistency :: ‹('a, 'b) form set set ⇒ bool› where
 ‹consistency C = (∀ S. S ∈ C ⟶
    (∀ p ts. ¬ (Pred p ts ∈ S ∧ Neg (Pred p ts) ∈ S)) ∧
    FF ∉ S ∧ Neg TT ∉ S ∧
    (∀ Z. Neg (Neg Z) ∈ S ⟶ S ∪ {Z} ∈ C) ∧
    (∀ A B. And A B ∈ S ⟶ S ∪ {A, B} ∈ C) ∧
    (∀ A B. Neg (Or A B) ∈ S ⟶ S ∪ {Neg A, Neg B} ∈ C) ∧
    (∀ A B. Or A B ∈ S ⟶ S ∪ {A} ∈ C ∨ S ∪ {B} ∈ C) ∧
    (∀ A B. Neg (And A B) ∈ S ⟶ S ∪ {Neg A} ∈ C ∨ S ∪ {Neg B} ∈ C) ∧
    (∀ A B. Impl A B ∈ S ⟶ S ∪ {Neg A} ∈ C ∨ S ∪ {B} ∈ C) ∧
    (∀A B. Neg (Impl A B) ∈ S ⟶ S ∪ {A, Neg B} ∈ C) ∧
    (∀P t. closedt 0 t ⟶ Forall P ∈ S ⟶ S ∪ {P[t/0]} ∈ C) ∧
    (∀P t. closedt 0 t ⟶ Neg (Exists P) ∈ S ⟶ S ∪ {Neg (P[t/0])} ∈ C) ∧
    (∀P. Exists P ∈ S ⟶ (∃x. S ∪ {P[App x []/0]} ∈ C)) ∧
    (∀P. Neg (Forall P) ∈ S ⟶ (∃x. S ∪ {Neg (P[App x []/0])} ∈ C)))›
```

Fitting and Berghofer's model existence theorem then states the following:

Theorem 1. *If C is a consistency property, $S \in C$, $\phi \in S$ and ϕ is closed, then ϕ has a model.*

We will not repeat the proof here, and instead refer to Berghofer [2] and Fitting [8]. We note that the result only applies to closed formulas.

Berghofer shows that the collection (set) of natural deduction consistent sets of formulas is a consistency property and he can therefore obtain the following model existence theorem for natural deduction in particular:

Theorem 2. *If S is a natural deduction consistent set, then there is a model for the closed formulas in S.*

3.5 Completeness

Proving completeness means showing that if $q_1, ..., q_n \models p$ then $q_1, ..., q_n \vdash p$. In other words, if a conclusion follows logically from a set of premises then the conclusion can also be proved from the set of premises. Fitting's proof is by contraposition, i.e. he shows that if $q_1, ..., q_n \nvdash p$ then $q_1, ..., q_n \not\models p$. He therefore assumes that $q_1, ..., q_n \nvdash p$, i.e. there is no natural deduction proof of the conclusion from the premises. Natural deduction allows proof by contradiction, so in particular there is no natural deduction proof by contradiction, i.e. we also know that $q_1, ..., q_n, \neg p \nvdash \bot$. Thus, $q_1, ..., q_n, \neg p$ is consistent with respect to natural deduction. The model existence theorem for natural deduction states that if a set of formulas is consistent with respect to natural deduction then it is satisfiable. We can therefore use it to conclude that natural deduction consistent formulas are satisfiable, and thus $q_1, ..., q_n, \neg p$ has a model. That model is the evidence that $q_1, ..., q_n \not\models p$. This concludes the completeness proof by contraposition.

In the following we will see that as soon as we have shown that the model existence theorem can be applied to the tableau calculus then the above completeness argument can largely be repeated to prove the tableau calculus complete.

4 Open Formulas and Natural Deduction

In this section we show how to extend a completeness result for sentences to one for open formulas. We presented this technique at the Tenth Scandinavian Logic Symposium in 2018 [9] (abstract only) and at the (informal, no-proceedings) Isabelle Workshop 2020 [14]. The text in this section is adapted from the latter.

We illustrate the five-step technique by showing the lemma that combines the steps. The result we prove is the following:

assumes $\langle\forall\,(e :: \mathit{nat} \Rightarrow \mathit{nat\ hterm})\ f\ g.\ e,\ f,\ g,\ z \models p\rangle$
shows $\langle z \vdash p\rangle$

We assume that formula p is valid under assumptions z and want to derive $z \vdash p$.

1. We simplify the problem by turning the meta-level assumptions into object-level implications (*put-imps p (rev z)* builds the chain $z_n \to \ldots \to z_1 \to p$):

let $\langle ?p = \langle \mathit{put\text{-}imps}\ p\ (\mathit{rev}\ z)\rangle$

Importantly, this preserves validity:

have $*$: $\langle\forall\,(e :: \mathit{nat} \Rightarrow \mathit{nat\ hterm})\ f\ g.\ \mathit{eval}\ e\ f\ g\ ?p\rangle$

2. Next, we universally close the formula by prefixing it with a sufficient number (m) of universal quantifiers and note that this too preserves validity:

obtain m **where** $**$: $\langle \mathit{closed}\ 0\ (\mathit{put\text{-}unis}\ m\ ?p)\rangle$
moreover have $\langle\forall\,(e :: \mathit{nat} \Rightarrow \mathit{nat\ hterm})\ f\ g.\ e,\ f,\ g,\ [] \models \mathit{put\text{-}unis}\ m\ ?p\rangle$

3. The resulting formula is a valid sentence so we know from the existing completeness result that it can be derived:

ultimately have $\langle [] \vdash \mathit{put\text{-}unis}\ m\ ?p\rangle$

4. By working within the proof system we can then derive the open formula:

then have $\langle [] \vdash ?p\rangle$

5. And finally we use a deduction theorem to turn the introduced implications back into meta-level assumptions:

then show $\langle z \vdash p\rangle$

Steps 1 through 3 are not particularly difficult so we focus on steps 4 and 5.

4.1 Deriving the Open Formula

We have a derivation of a formula with m universal quantifiers in front and want a derivation without them. We could use the *Uni-E* rule to substitute the original variables for the freshly quantified ones but this requires some tricky reasoning due to our use of de Bruijn indices. The following example starts from the formula $\forall\forall\forall p(0,1,2)$ and illustrates how the variables shift when eliminating the quantifiers:

$$\begin{aligned}
(\forall\forall p(0,1,2))[2/0] \rightsquigarrow \forall((\forall p(0,1,2))[3/1]) \rightsquigarrow \forall\forall(p(0,1,2)[4/2]) \rightsquigarrow \forall\forall p(0,1,4)\\
(\forall p(0,1,4))[1/0] \rightsquigarrow \forall(p(0,1,4)[2/1]) \rightsquigarrow \forall p(0,2,3)\\
p(0,2,3)[0/0] \rightsquigarrow p(0,1,2)
\end{aligned}$$

To avoid having to reason about these shifts, we instead eliminate the universal quantifiers with fresh constants. The function *subc c s p* replaces occurrences of c with s in p, adjusting the variables in s when substituting under a quantifier. It is admissible to perform such a substitution uniformly across the formula and assumptions of a derivation:

shows $\langle z \vdash p \Longrightarrow \mathit{subcs}\ c\ s\ z \vdash \mathit{subc}\ c\ s\ p\rangle$

The proof goes by induction on the derivation and is mechanical, except for the quantifier rules where care must be taken to treat the involved constants correctly. The following renaming result is useful. Here *psubst* applies a function to every constant/function symbol:

shows $\langle z \vdash p \Longrightarrow \mathit{map}\ (\mathit{psubst}\ f)\ z \vdash \mathit{psubst}\ f\ p\rangle$

When we compose closure elimination (*sub*) with constant substitution (*subc*) in the right order we get the following *telescoping* sequence of substitutions:

$$\mathit{subc}\ c_0\ \ (m\text{-}1)\ \ (\mathit{subc}\ c_1\ \ (m\text{-}2)\ \ (\ldots\ \ (\mathit{subc}\ c_{m-1}\ \ 0\ (\mathit{sub}\ 0\ c_{m-1}\ \ldots))))$$

Each introduced constant is immediately replaced by the correct variable and since subsequent substitutions are of constants, they do not adjust previously inserted variables. We can then prove our desired result:

lemma *remove-unis-sentence*:
  assumes *inf-params*: $\langle \mathit{infinite}\ (-\ \mathit{params}\ p)\rangle$
    and $\langle \mathit{closed}\ 0\ \ (\mathit{put\text{-}unis}\ m\ p)\rangle\ \langle[] \vdash \mathit{put\text{-}unis}\ m\ p\rangle$
  shows $\langle[] \vdash p\rangle$

4.2 Shifting Implications

Step 5 of our technique is to derive $z \vdash p$ from $[] \vdash \mathit{put\text{-}imps}\ p\ (\mathit{rev}\ z)$. We do so with the following lemma, which shifts just one formula from an implication to the assumptions:

assumes *inf-params*: $\langle \mathit{infinite}\ (\mathit{UNIV} :: {'a}\ \mathit{set})\rangle$
  and $\langle z \vdash \mathit{Impl}\ p\ q\rangle$
shows $\langle p\ \#\ z \vdash q\rangle$

In the proof, we weaken z with p and apply modus ponens (*Imp-E*). The weakening is shown by induction over the inference rules:

shows $\langle z \vdash p \Longrightarrow \mathit{set}\ z \subseteq \mathit{set}\ z' \Longrightarrow z' \vdash p\rangle$

The proof is trivial except for the cases for *Exi-E* and *Uni-I*, where the Skolem constant fixed by the induction hypothesis is only new to the smaller set of premises, not necessarily the larger ones. Again, it is necessary to perform renaming using *psubst*. We can now shift a list of implications:

shows $\langle z' \vdash \mathit{put\text{-}imps}\ p\ z \Longrightarrow \mathit{rev}\ z\ @\ z' \vdash p\rangle$

4.3 Completeness

In conclusion, we have completeness for any valid formula, open or closed:

assumes $\langle \forall\, (e :: \mathit{nat} \Rightarrow \mathit{nat}\ \mathit{hterm})\ f\, g.\ e,\, f,\ g,\ z \models p\rangle$
shows $\langle z \vdash p\rangle$

The recipe makes no particular use of natural deduction – in the places where the recipe does derivations, one expects similar derivations to be possible in other calculi. Indeed we will see that the recipe applies to our semantic tableau calculus as well.

5 Soundness and Completeness of Tableau

The Isabelle theory `Tableau` contains the tableau formalization. We use this as a stepping stone to show completeness for the sequent calculus since the tableau rules match the consistency property making it almost trivial to show completeness.

5.1 Definition

Tableau calculi can be understood as dual to sequent calculi. Where a one-sided sequent is interpreted as a disjunction, each node in a tableau stands for a conjunction of formulas. In a sequent calculus proof we end each sub-derivation with an axiom, but in a closed tableau we mark each branch with a contradiction. As will become apparent, the following rules are derived from this duality:

```
inductive TC :: ⟨('a, 'b) form list ⇒ bool⟩ (⟨⊣ -⟩ 0) where
  Basic: ⟨⊣ Pred i l # Neg (Pred i l) # G⟩
| BasicFF: ⟨⊣ ⊥ # G⟩
| BasicNegTT: ⟨⊣ Neg ⊤ # G⟩
| AlphaNegNeg: ⟨⊣ A # G ⟹ ⊣ Neg (Neg A) # G⟩
| AlphaAnd: ⟨⊣ A # B # G ⟹ ⊣ And A B # G⟩
| AlphaNegOr: ⟨⊣ Neg A # Neg B # G ⟹ ⊣ Neg (Or A B) # G⟩
| AlphaNegImpl: ⟨⊣ A # Neg B # G ⟹ ⊣ Neg (Impl A B) # G⟩
| BetaNegAnd: ⟨⊣ Neg A # G ⟹ ⊣ Neg B # G ⟹ ⊣ Neg (And A B) # G⟩
| BetaOr: ⟨⊣ A # G ⟹ ⊣ B # G ⟹ ⊣ Or A B # G⟩
| BetaImpl: ⟨⊣ Neg A # G ⟹ ⊣ B # G ⟹ ⊣ Impl A B # G⟩
| GammaForall: ⟨⊣ subst A t 0 # G ⟹ ⊣ Forall A # G⟩
| GammaNegExists: ⟨⊣ Neg (subst A t 0) # G ⟹ ⊣ Neg (Exists A) # G⟩
| DeltaExists: ⟨⊣ subst A (App n []) 0 # G ⟹ news n (A # G) ⟹ ⊣ Exists A #
G⟩
| DeltaNegForall: ⟨⊣ Neg (subst A (App n []) 0) # G ⟹ news n (A # G) ⟹ ⊣ Neg
(Forall A) # G⟩
| Order: ⟨⊣ G ⟹ set G = set G' ⟹ ⊣ G'⟩
```

A closed tableau should prove unsatisfiability of the starting formulas. Since ⊥ is never satisfiable, *BasicFF* allows us to close a branch directly if it occurs. More interestingly, the *BetaOr* rule says that both *A # G* and *B # G* have to be unsatisfiable for *Or A B # G* to be so. The remaining rules are similar.

5.2 Soundness

The lemma *TC-soundness* states the soundness property:

```
⟨⊣ G ⟹ ∃ p ∈ set G. ¬ eval e f g p⟩
```

If a closing tableau exists for the list of formulas G, then the conjunction of G is unsatisfiable: every interpretation falsifies some formula p in G.

The proof of *TC-soundness* is fairly straightforward and goes by induction on the inference rules (for an arbitrary function denotation). It is written in

Isar and relies only on existing lemmas introduced by Berghofer. Only two of the inductive cases cannot be proven automatically by Isabelle: *DeltaExists* and *DeltaNegForall*. They have similar proofs and we consider the former case here.

We need to show $\exists p \in \mathit{set}\ (\mathit{Exi}\ A\ \#\ G).\ \neg\ \mathit{eval}\ e\ f\ g\ p$ and have by the induction hypothesis: $\exists p \in \mathit{set}\ (A\,[\mathit{Fun}\ n\ []\,/\,0]\ \#\ G).\ \neg\ \mathit{eval}\ e\ \mathit{?f}\ g\ p$ where n is a new constant and $\mathit{?f}$ represents any function denotation.

We proceed by contradiction and assume that every formula holds in the given model: (*) $\forall p \in \mathit{set}\ (\mathit{Exi}\ A\ \#\ G).\ \mathit{eval}\ e\ f\ g\ p$. Since $\mathit{Exi}\ A$ holds, there must be a member of the universe, say, x that satisfies $A[\mathit{Fun}\ n\ []\,/\,0]$ when n is interpreted as x: (1) $\mathit{eval}\ e\ (f(n := \lambda w.\ x))\ g\ p.\ A\,[\mathit{Fun}\ n\ []\,/\,0]$.

By applying the induction hypothesis at the same updated function denotation we know: (2) $\exists p \in \mathit{set}\ (A\,[\mathit{Fun}\ n\ []\,/\,0]\ \#\ G).\ \neg\ \mathit{eval}\ e\ (f(n := \lambda w.\ x))\ g\ p.$

From (2) we have two cases: either $A\,[\mathit{Fun}\ n\ []\,/\,0]$ is false or G contains the false formula. The first case contradicts (1) directly. Alternatively, if G contains the false formula, this contradicts our starting assumption (*).

The following abbreviates that p can be proved from assumptions ps:

$\langle \mathit{tableauproof}\ ps\ p \equiv (\dashv \mathit{Neg}\ p\ \#\ ps)\rangle$

With it, we can state soundness more easily:

$\langle \mathit{tableauproof}\ ps\ p \Longrightarrow \mathit{list\text{-}all}\ (\mathit{eval}\ e\ f\ g)\ ps \Longrightarrow \mathit{eval}\ e\ f\ g\ p\rangle$

5.3 Completeness

We show completeness of the tableau calculus in the same way as for natural deduction: by proving a consistency property and using the model existence result to contradict the nonexistence of a closing tableau for a valid formula. In natural deduction, the consistent sets are those from which we cannot derive a contradiction ($\bot$). In tableau calculi, deriving contradictions is exactly the point, so the consistent sets are simply those without closing tableaux. The statement of our consistency theorem *TCd-consistency* thus becomes:

shows $\langle \mathit{consistency}\ \{S{::}('a,\ 'b)\ \mathit{form\ set}.\ \exists\, G.\ S = \mathit{set}\ G \wedge \neg\ (\dashv G)\}\rangle$

Our tableau rules are dual to the definition of when a set is consistent [2]. For instance, we can close a tableau if a predicate appears both positively and negatively with the same arguments, while such a pair cannot occur in a consistent set. This makes the consistency simple and mechanical to show.

Take the case when $\neg\neg\phi$ occurs in a consistent set S, so $S \cup \{\phi\}$ should also be consistent. No closing tableau can exist for $S \cup \{\phi\}$. If it did, one would also exist for S because we can extend S with ϕ by applying the *AlphaNegNeg* rule to $\neg\neg\phi \in S$. The corresponding proofs for natural deduction generally require more creativity and more than one rule application.

We obtain the completeness result *tableau-completeness'* for closed formulas:

assumes $\langle \mathit{closed}\ 0\ p\rangle$
  and $\langle \mathit{list\text{-}all}\ (\mathit{closed}\ 0)\ ps\rangle$
  and mod: $\langle \forall\,(e :: \mathit{nat} \Rightarrow \mathit{nat\ hterm})\ f\ g.\ \mathit{list\text{-}all}\ (\mathit{eval}\ e\ f\ g)\ ps \longrightarrow \mathit{eval}\ e\ f\ g\ p\rangle$
shows $\langle \mathit{tableauproof}\ ps\ p\rangle$

We assume that p is valid under assumptions ps (in the universe of Herbrand terms) but that we cannot close the corresponding tableau. Then the set formed by $\neg p$ and ps is consistent and we have seen that any formula in it has a model. By our validity assumption the same model satisfies p, and this leads to a contradiction.

6 Open Formulas and Tableau

We use the same recipe as for natural deduction to extend the tableau completeness result to cover open formulas. However, we employ one simplification that would also work for natural deduction but is slightly less intuitive. Instead of closing the formula with universal quantifiers, we close it by substituting fresh constants directly for the free variables. At that point we are still operating semantically, so we only need to show that this preserves validity and doing so is simple. We still make use of the telescoping sequence of substitutions but save the introduction of the universal quantifiers. Our application of this recipe to both natural deduction and a tableau calculus indicate that it can be applied to a wider range of proof systems to strengthen existing completeness results.

7 Sequent Calculus

The Isabelle theory `Sequent` contains the sequent formalization.

7.1 Definition

We define the sequent calculus as yet another inductive collection of rules:

```
inductive SC :: ‹('a, 'b) form list ⇒ bool› (‹⊢ -› 0) where
  Basic: ‹⊢ Pred i l # Neg (Pred i l) # G›
| BasicNegFF: ‹⊢ Neg ⊥ # G›
| BasicTT: ‹⊢ ⊤ # G›
| AlphaNegNeg: ‹⊢ A # G ⟹ ⊢ Neg (Neg A) # G›
| AlphaNegAnd: ‹⊢ Neg A # Neg B # G ⟹ ⊢ Neg (And A B) # G›
| AlphaOr: ‹⊢ A # B # G ⟹ ⊢ Or A B # G›
| AlphaImpl: ‹⊢ Neg A # B # G ⟹ ⊢ Impl A B # G›
| BetaAnd: ‹⊢ A # G ⟹ ⊢ B # G ⟹ ⊢ And A B # G›
| BetaNegOr: ‹⊢ Neg A # G ⟹ ⊢ Neg B # G ⟹ ⊢ Neg (Or A B) # G›
| BetaNegImpl: ‹⊢ A # G ⟹ ⊢ Neg B # G ⟹ ⊢ Neg (Impl A B) # G›
| GammaExists: ‹⊢ subst A t 0 # G ⟹ ⊢ Exists A # G›
| GammaNegForall: ‹⊢ Neg (subst A t 0) # G ⟹ ⊢ Neg (Forall A) # G›
| DeltaForall: ‹⊢ subst A (App n []) 0 # G ⟹ news n (A # G) ⟹ ⊢ Forall A # G›
| DeltaNegExists: ‹⊢ Neg (subst A (App n []) 0) # G ⟹ news n (A # G) ⟹ ⊢ Neg
(Exists A) # G›
| Order: ‹⊢ G ⟹ set G = set G' ⟹ ⊢ G'›
```

7.2 Soundness

The lemma *SC-soundness* states the soundness property:

lemma *SC-soundness*: $\langle \vdash\ G \Longrightarrow \exists\, p \in \mathit{set}\ G.\ \mathit{eval}\ e\ f\ g\ p \rangle$

If the sequent G has a derivation then the disjunction of G is valid: every interpretation satisfies some formula p in G. By taking the sequent with a single formula we get the expected theorem.

Dually to the tableau proof, the only two cases that are not solved automatically by Isabelle are: *DeltaForall* and *DeltaNegExists*. Consider the first one.

We need to show $\exists p \in \mathit{set}\ (\mathit{Uni}\ A\ \#\ G).\ \mathit{eval}\ e\ f\ g\ p$ and we can assume the induction hypothesis: $\exists p \in \mathit{set}\ (A\,[\mathit{Fun}\ n\ []\,/\,0]\ \#\ G).\ \mathit{eval}\ e\ \mathit{?f}\ g\ p$.

When we instantiate the induction hypothesis at the function denotation $f(n := \lambda w.\ x)$ we get that for all witnesses x, some formula holds:

$$\forall x.\ \exists p \in \mathit{set}\ (A\,[\mathit{Fun}\ n\ []\,/\,0]\ \#\ G).\ \mathit{eval}\ e\ (f(n := \lambda w.\ x))\ g\ p$$

Now there are two cases. Either it is the case that the first formula holds for all witnesses, i.e. $\forall x.\ \mathit{semantics}\ e\ f(n := \lambda w.\ x)\ g\ (A\,[\mathit{Fun}\ n\ []\,/\,0])$, or it is the case that there exists a witness that makes some formula in G hold, i.e. $\exists x.\ \exists p \in \mathit{set}\,G.\ \mathit{semantics}\ e\ (f(n := \lambda w.\ x))\ g\ p$.

The first case corresponds to the semantics of the universal quantifier, so we know *Uni A* holds and we are done. In the second case we can recover the unmodified function denotation f since n is new in G, and thereby know that something in G holds, so we are done.

7.3 Completeness

We show completeness of the sequent calculus by proving that for every closed tableau over a list of formulas $[p_1, \ldots, p_n]$, there exists a sequent calculus derivation of the list of complementary formulas $[\neg p_1, \ldots, \neg p_n]$. This is stated in Isabelle by mapping the function *compl* across the list, where *compl* is defined as follows: $\mathit{compl}\ (\neg\, p) = p$, and $\mathit{compl}\ p = \neg\, p$ for any formula p that is not a negation. Thus in Isabelle we want to show $\dashv\ G \Longrightarrow\ \vdash\ \mathit{map}\ \mathit{compl}\ G$.

The proof goes by induction over the tableau rules with a bit of massaging of each induction hypothesis to make the corresponding sequent calculus rule applicable. Consider for instance the *AlphaAnd* case where we assume $\dashv\ A\ \#\ B\ \#\ G$ and, inductively, $\vdash\ \mathit{map}\ \mathit{compl}\ (A\ \#\ B\ \#\ G)$ and need to show $\vdash\ \mathit{map}\ \mathit{compl}\ (\mathit{Con}\ A\ B\ \#\ G)$. We get $\vdash\ \mathit{compl}\ A\ \#\ \mathit{compl}\ B\ \#\ \mathit{map}\ \mathit{compl}\ G$ from the induction hypothesis and definition of *map*. There are two cases for A (and similarly B). Either it is of the form $A = \neg A'$ or it starts with a different connective. In the first case, $\mathit{compl}\ A = A'$. We want to apply *AlphaNegAnd* which requires a derivation containing $\neg A$, so in the first case we apply *AlphaNegNeg* first. We can hereafter derive $\vdash\ \mathit{Neg}\ (\mathit{Con}\ A\ B)\ \#\ \mathit{map}\ \mathit{compl}\ G$ using *AlphaNegAnd*. In the second case $\mathit{compl}\ A = \neg A$, and we need only to apply *AlphaNegAnd*. Using the definition of *map* again we see that it is the goal.

Ben-Ari [1] also shows this relationship between a tableau calculus and a sequent calculus. However, he neglects the complications around *compl* and how the *AlphaNegNeg* rule can be necessary to make the cases line up. By carrying out our work in a proof assistant, we cannot accidentally make such omissions.

The theorem *SC-completeness* states the completeness property:

theorem *SC-completeness*:
  fixes p :: ‹(*nat*, *nat*) *form*›
  assumes ‹$\forall$ (e :: *nat* $\Rightarrow$ *nat hterm*) f g. *list-all* (*eval* e f g) *ps* $\longrightarrow$ *eval* e f g p›
  shows ‹$\vdash$ p # *map compl ps*›

Assume p is valid under assumptions *ps*. Any of the formulas can be open. From completeness for tableau, the tableau for $\neg\, p$ # *ps* closes, so by the equivalence to sequent calculus, p # *map compl ps* has a sequent calculus derivation.

8 Conclusions and Future Work

We have shown that the natural deduction system formalized by Berghofer is also complete when considering open formulas. We have also formalized the soundness and a synthetic completeness proof of a one-sided sequent calculus for first-order logic. This was via a translation from a tableau calculus. Standing on the shoulders of Berghofer, we proved the tableau calculus complete based on his natural deduction formalization. This shows that Berghofer's work is more general than just a natural deduction formalization and can indeed be used as an offset for proving calculi complete.

As future work we envision formalizing the resolution system from Fitting's book [8]. This should easily fit our approach since Berghofer's work is based on this book.

With the present paper, detailing our novel formalization in Isabelle/HOL of the soundness and completeness proofs for a sequent calculus and a tableau calculus for first-order logic with functions, we hope that more people will get inspired and take up the gauntlet to formalize logical systems and build frameworks to aid this task. There are plenty of systems out there waiting to be formalized and plenty of room to build upon the frameworks currently available.

Acknowledgements

We thank Agnes Moesgård Eschen, Frederik Krogsdal Jacobsen, Alexander Birch Jensen, Simon Tobias Lund, François Schwarzentruber and Freek Wiedijk for comments on drafts.

References

1. Ben-Ari, M.: Mathematical logic for computer science (2. ed.). Springer (2001)
2. Berghofer, S.: First-order logic according to Fitting. Archive of Formal Proofs (Aug 2007), https://isa-afp.org/entries/FOL-Fitting.html, Formal proof development

3. 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, January 14-15, 2019. pp. 1–13. ACM (2019)
4. Blanchette, J.C., Popescu, A.: Mechanizing the metatheory of Sledgehammer. In: Frontiers of Combining Systems - 9th International Symposium, FroCoS 2013, Nancy, France, September 18-20, 2013. Proceedings. pp. 245–260 (2013)
5. Blanchette, J.C., Popescu, A., Traytel, D.: Soundness and completeness proofs by coinductive methods. J. Autom. Reason. **58**(1), 149–179 (2017). https://doi.org/10.1007/s10817-016-9391-3
6. Braselmann, P., Koepke, P.: Gödel's completeness theorem. Formalized Mathematics **13**(1), 49–53 (2005)
7. Ebbinghaus, H., Flum, J., Thomas, W.: Mathematical logic. Undergraduate texts in mathematics, Springer (1984)
8. Fitting, M.: First-Order Logic and Automated Theorem Proving, Second Edition. Graduate Texts in Computer Science, Springer (1996)
9. From, A.H.: Formalized Soundness and Completeness of Natural Deduction for First-Order Logic (2018), tenth Scandinavian Logic Symposium (SLS 2018), http://scandinavianlogic.org/material/book_of_abstracts_sls2018.pdf
10. From, A.H.: A sequent calculus for first-order logic. Archive of Formal Proofs (Jul 2019), https://isa-afp.org/entries/FOL_Seq_Calc1.html, Formal proof development
11. From, A.H., Jensen, A.B., Schlichtkrull, A., Villadsen, J.: Teaching a formalized logical calculus. In: Quaresma, P., Neuper, W., Marcos, J. (eds.) Proceedings 8th International Workshop on Theorem Proving Components for Educational Software, ThEdu@CADE 2019, 25th August 2019. EPTCS, vol. 313, pp. 73–92 (2019)
12. 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, 29th June 2020. EPTCS, vol. 328, pp. 18–34 (2020)
13. From, A.H., Jacobsen, F.K., Villadsen, J.: SeCaV: A sequent calculus verifier in Isabelle/HOL. In: Pre-proceedings of the 16th International Workshop on Logical and Semantic Frameworks with Applications, LSFA 2021, Buenos Aires, Argentina (online), 23–24 July (2021)
14. From, A.H., Villadsen, J.: A concise sequent calculus for teaching first-order logic, Isabelle Workshop 2020 (informal, no proceedings)
15. Herbelin, H., Kim, S.Y., Lee, G.: Formalizing the meta-theory of first-order predicate logic. Journal of the Korean Mathematical Society **54**(5), 1521–1536 (2017)
16. Ilik, D.: Constructive Completeness Proofs and Delimited Control. Ph.D. thesis, École Polytechnique (2010), https://tel.archives-ouvertes.fr/tel-00529021/document
17. Ilik, D., Lee, G., Herbelin, H.: Kripke models for classical logic. Annals of Pure and Applied Logic **161**(11), 1367–1378 (2010)
18. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
19. Persson, H.: Constructive completeness of intuitionistic predicate logic. Ph.D. thesis, Chalmers University of Technology (1996), http://web.archive.org/web/20001011101511/http://www.cs.chalmers.se/~henrikp/Lic/
20. Ridge, T., Margetson, J.: A mechanically verified, sound and complete theorem prover for first order logic. In: Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005, Oxford, UK, August 22-25, 2005, Proceedings. pp. 294–309 (2005)