Formalized Proof Systems for Propositional Logic

Julius Michaelis[1] **and Tobias Nipkow**[1]

1 **Fakultät für Informatik, Technische Universität München, Boltzmannstr. 3, 85748 Garching**
michaeli@in.tum.de **http://www.in.tum.de/~nipkow**

Abstract

We have formalized a range of proof systems for classical propositional logic (sequent calculus, natural deduction, Hilbert systems, resolution) in Isabelle/HOL and have proved the most important meta-theoretic results about semantics and proofs: compactness, soundness, completeness, translations between proof systems, cut-elimination, interpolation and model existence.

1998 ACM Subject Classification F.4.1 Theory of computation–Proof theory

Keywords and phrases formalization of logic, proof systems, sequent calculus, natural deduction, resolution

Digital Object Identifier 10.4230/LIPIcs.TYPES.2017.6

1 Introduction

This paper presents a unified formalization in Isabelle/HOL [29] of the classical proof systems for classical propositional logic (sequent calculus, natural deduction, Hilbert systems, resolution) and their meta-theory: compactness, soundness, completeness, translations between proof systems, cut-elimination, interpolation and model existence. The motivation is the desire to develop (in the long run) a counterpart to the emerging *Encyclopaedia of Proof Systems* [51], including all formalized meta-theory. Our work is also part of IsaFoL [3], a collection of formalizations of logics in Isabelle. A second motivation comes from teaching logic. Most textbooks cover only specific proof systems and specific results proved in a specific manner, and it is hard to modify the setup. That is why we cover multiple proof systems with their interconnections. In particular we give multiple proofs of some key results, e.g. three different completeness proofs. The longer term goal is a formalized logic textbook in the style of *Concrete Semantics* [28] but with a web of proof systems and multiple proofs of key results such that one can select one's own subset of the material. It is also possible to obtain verified executable code from such formalizations, e.g. our toy sequent calculus and resolution provers and our Craig interpolator.

Our paper differs from each of the related papers below in at least one of the following aspects: we do not reason about deductive systems alone but also about semantics; we do not consider a specific calculus but a collection of canonical calculi; we do not prove a specific result but most if not all the results covered in the propositional section of an introductory logic course. That is, we provide a unique, unified library of definitions and proofs. The formalization of the translations between all the proof systems appears to be new.

The complete development is available online [26]. It consists of 5000 lines of Isabelle text that translate into more than 100 A4 pages of definitions and proofs. For readability the notation in this paper differs in places from the Isabelle text.

1.1 Related Work

Formalization of logic was probably started by Shankar's proof of Gödel's first incompleteness theorem [42]. Harrison formalized basic first-order model theory [16]. There are many other formalizations of results about propositional and first-order logics (e.g. cut elimination [33, 49], interpolation [9], completeness [36, 8, 39, 21, 6], incompleteness [31], resolution [37, 38], quantifier elimination [27]) and temporal logic (e.g. [11]). Berghofer [1] formalized part of [13] but this was never published. There is also significant work on formalized SAT solvers [50, 25, 43, 24, 30, 5]. Harrison [17] started to verify a theorem prover in itself, which was later extended [23]. The above publications rely on general purpose theorem provers. Alternatively, a number of metalogical frameworks for reasoning about deductive systems have been implemented (e.g. [32, 34, 14, 35]) and proof theoretic results were proved in them, e.g. cut elimination in intuitonistic and classical logic [33], cut admissibility for a number of propositional linear logics [10], and translations between various calculi for minimal propositional logic of implication (in the Abella [14] library).

2 Isabelle Notation

The logic of Isabelle/HOL conforms largely to everyday mathematical notation. This section summarizes non-standard notation.

The type of truth values is called *bool*. The function space is denoted by $\Rightarrow$. Type variables are denoted by $'a$, $'b$, etc. The notation $t :: \tau$ means that term t has type τ. Type constructors follow postfix syntax, e.g. $'a\ set$ is the type of sets of elements of type $'a$. Lists over type $'a$, type $'a\ list$, come with the empty list $[]$ and the infix constructor "$\cdot$", the infix append operator @. Function *set* converts a list into a set.

Type $'a\ multiset$ is the type of finite multisets over type $'a$. The empty multiset is $\emptyset_\#$, comprehension is written $\{...\}_\#$, multiset union is denoted by $+$, insertion by a comma, difference by $-$ and membership by $\in_\#$.

Horizontal lines in rules are displayed versions of the implication $\longrightarrow$.

Many of our theorems are proved by induction on the structure of formulas or derivations. If we do not say anything about how the induction cases are dealt with, this means that Isabelle can come up with proofs for them automatically (often with the help of Sledgehammer [2]).

3 Formulas and Their Semantics

A formula (which is simply a term of a recursive data type) is either an atom $A_0, A_1, \ldots$[1], or $\bot$, or constructed from other formulas using the connectives $\neg$, $\wedge$, $\vee$ and $\rightarrow$; note the bold font to distinguish them from $\neg$, $\wedge$, $\vee$ and $\longrightarrow$ in the metalogic. The variables F, G and H always stand for formulas. The function *atoms* returns the set of all atom indices in a formula (or similar object).

The semantics of a formula F is defined via a *valuation* or *assignment* $\mathcal{A} :: nat \Rightarrow bool$ (a function from atom indices to truth values) and a recursively defined operator $\mathcal{A} \models F$. The $\models$ operator canonically maps the constructors of the formulas to their equivalent boolean operators in the metalogic, e.g.

$$(\mathcal{A} \models F \rightarrow G) = (\mathcal{A} \models F \longrightarrow \mathcal{A} \models G).$$

[1] We use natural numbers to index atoms, but any countably infinite set would suffice.

$$\frac{}{A_k, \Gamma \Rightarrow A_k, \Delta}\ \textsc{Ax} \qquad \frac{}{\bot, \Gamma \Rightarrow \Delta}\ \textsc{BotL}$$

$$\frac{\Gamma \Rightarrow F, \Delta}{\neg F, \Gamma \Rightarrow \Delta}\ \textsc{NotL} \qquad \frac{F, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg F, \Delta}\ \textsc{NotR}$$

$$\frac{F, G, \Gamma \Rightarrow \Delta}{F \wedge G, \Gamma \Rightarrow \Delta}\ \textsc{AndL} \qquad \frac{\Gamma \Rightarrow F, \Delta \qquad \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \wedge G, \Delta}\ \textsc{AndR}$$

$$\frac{F, \Gamma \Rightarrow \Delta \qquad G, \Gamma \Rightarrow \Delta}{F \vee G, \Gamma \Rightarrow \Delta}\ \textsc{OrL} \qquad \frac{\Gamma \Rightarrow F, G, \Delta}{\Gamma \Rightarrow F \vee G, \Delta}\ \textsc{OrR}$$

$$\frac{\Gamma \Rightarrow F, \Delta \qquad G, \Gamma \Rightarrow \Delta}{F \to G, \Gamma \Rightarrow \Delta}\ \textsc{ImpL} \qquad \frac{F, \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \to G, \Delta}\ \textsc{ImpR}$$

Figure 1 Rules of sequent calculus

Atoms are evaluated by looking up their index in $\mathcal{A}$: $\mathcal{A} \models A_k = \mathcal{A}\ k$. Assignments are total functions and thus suitable for any formula.

4 Proof Systems

4.1 Sequent Calculus

We formalize a Sequent Calculus (SC) inspired by Troelstra and Schwichtenberg [48]. But whereas they define derivation trees inductively, we merely define derivable sequents, where sequents are pairs of multisets. That is, we formalize an inductively defined predicate $\Gamma \Rightarrow \Delta$ where Γ and Δ are multisets. Hence $\Gamma \Rightarrow \Delta$ means that the sequent is derivable. We begin with the system that [48] refers to as **G3c**: its rules, adjusted for our definition of formulas, are shown in Figure 1.

In the literature one finds three types of sequents: lists, multisets and sets. After some experiments with all three representations we found that multisets were most convenient, although we did not conduct an in-depth study. However, we did formalize Gentzen's original system (approximately **G2c** in [48]) based on lists (with all the structural rules) and proved the equivalence with the multiset-based system SC from Figure 1. From now on we consider the latter system only.

Many of the proofs in [48] are by induction over the depth of a derivation tree. In order to be able to follow this style we defined a second, ternary predicate written $\Gamma \Rightarrow_n \Delta$, where n is the depth of the derivation, and proved $(\exists\, n.\ \Gamma \Rightarrow_n \Delta) \longleftrightarrow \Gamma \Rightarrow \Delta$. A canonical definition of depth would have Ax and BotL start with 0 (or 1) and use the maximum in rules with two premises. Instead, we require both antecedents to have the same "depth", e.g.:

$$\frac{\Gamma \Rightarrow_n F, \Delta \qquad G, \Gamma \Rightarrow_n \Delta}{F \to G, \Gamma \Rightarrow_{n+1} \Delta} \qquad \frac{F, \Gamma \Rightarrow_n G, \Delta}{\Gamma \Rightarrow_{n+1} F \to G, \Delta} \qquad \frac{}{A_k, \Gamma \Rightarrow_{n+1} A_k, \Delta}$$

We found this modification to be more suitable for Isabelle's automated reasoning tools.

While this allows the proofs in the next section to follow [48], we additinally found proofs that do not require depth arguments. We present only those.

4.1.1 Cut Admissibility

We prove weakening, inversion, contraction and cut as admissible rules in the meta-logic. The proofs of the eight inversion rules can be partly automated, but they are still quite long. One major issue is exchange: given a sequent of the form $F,\ G \to H,\ \Gamma \Rightarrow \Delta$, we need to

exchange F and $G \to H$ before we can apply ImpL. The problem is that Isabelle's unification algorithm does not know about multiset laws. Although we do not need to apply an explicit exchange rule of SC, the proof still needs to apply the multiset equation $F,\ G,\ \Gamma = G,\ F,\ \Gamma$. Luckily, ordered rewriting (which Isabelle's simplifier supports) with this equation sorts sequents into a canonical order, thus automating the application of this equation in many cases. Another reason for our proofs to be long is that we formalized derivable sequents. When a proof in the literature makes a case distinction on whether a formula is principal, we can only reproduce that by making a case distinction on which rule was applied last. That requires discharging ten instead of two cases.

The proof of the admissibility of the cut rule is by induction on the cut formula as in [48]. The induction steps are entirely taken care of automatically. For the base cases, we need two auxiliary lemmas:

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta - \{\bot\}_\#} \qquad \frac{A_k,\ \Gamma \Rightarrow \Delta \qquad \Gamma \Rightarrow A_k,\ \Delta}{\Gamma \Rightarrow \Delta}$$

They are proved by induction on the derivations of $\Gamma \Rightarrow \Delta$ and $A_k,\ \Gamma \Rightarrow \Delta$. While the second lemma is just the cut rule for atoms, it is easier to show than the cut rule itself since we know that the cut formula cannot be principal in the induction steps of its proof.

The proof of the two contraction rules proceeds in exactly the same manner, unlike the proof by induction on the derivation tree depth in [48]. The proof by induction on the depth does not require the two auxiliary lemmas in the base cases but is slightly more unwieldy in Isabelle/HOL.

4.1.2 Soundness and Completeness

We define the semantics of a sequent as follows:

$$\mathcal{A} \models \Gamma \Rightarrow \Delta \stackrel{\text{def}}{=} (\forall F{\in_\#}\Gamma.\ \mathcal{A} \models F) \longrightarrow (\exists F{\in_\#}\Delta.\ \mathcal{A} \models F).$$

Note that $\_ \models \_ \Rightarrow \_$ is a ternary operator. Validity is defined as usual:

$$\models \Gamma \Rightarrow \Delta \stackrel{\text{def}}{=} \forall \mathcal{A}.\ \mathcal{A} \models \Gamma \Rightarrow \Delta$$

Soundness ($\Gamma \Rightarrow \Delta \longrightarrow \models \Gamma \Rightarrow \Delta$) is proved by induction on $\Gamma \Rightarrow \Delta$. The proof of completeness

$$\models \Gamma \Rightarrow \Delta \longrightarrow \Gamma \Rightarrow \Delta \tag{1}$$

is more involved and conceptually similar to Gallier's [15, §3.4.6]. Gallier presents a search procedure that constructs derivation trees. Once the procedure terminates (for propositional logic, it always does), the set of open leaves in the derivation tree corresponds to the set of counterexamples. Our proof search procedure *sc* dispenses with derivation trees and works on lists of formulas and atoms. Replacing multisets with lists is necessary to formulate *sc* as a simple structurally recursive function. In a call *sc* Γ A Δ B, the parameters Γ and Δ are lists of formulas and (ignoring A and B) we are trying to prove *mset* $\Gamma \Rightarrow$ *mset* Δ, where *mset* converts a list into a multiset. In each step, the first formula in Γ or Δ is decomposed according to a rule of SC. Atoms (more precisely, their indices) are moved into A and B. If $\Gamma = \Delta = [\,]$ and *set* $A \cap$ *set* $B = \emptyset$, then a counterexample has been found. The defining equations for *sc* are shown in Figure 2.

From a pair (A, B) returned by *sc*, a countermodel can be constructed: map all elements of A to *True* and all elements of B to *False*. The following propositions are not hard to prove:

$$
\begin{array}{lcl}
sc\ []\ A\ []\ B & = & \mathsf{if}\ set\ A \cap set\ B = \emptyset\ \mathsf{then}\ \{(A,\ B)\}\ \mathsf{else}\ \emptyset \\
sc\ (A_k \cdot \Gamma)\ A\ \Delta\ B & = & sc\ \Gamma\ (k \cdot A)\ \Delta\ B \\
sc\ \Gamma\ A\ (A_k \cdot \Delta)\ B & = & sc\ \Gamma\ A\ \Delta\ (k \cdot B) \\
sc\ (\bot \cdot \Gamma)\ A\ \Delta\ B & = & \emptyset \\
sc\ \Gamma\ A\ (\bot \cdot \Delta)\ B & = & sc\ \Gamma\ A\ \Delta\ B \\
sc\ (\neg\ F \cdot \Gamma)\ A\ \Delta\ B & = & sc\ \Gamma\ A\ (F \cdot \Delta)\ B \\
sc\ \Gamma\ A\ (\neg\ F \cdot \Delta)\ B & = & sc\ (F \cdot \Gamma)\ A\ \Delta\ B \\
sc\ (F \wedge G \cdot \Gamma)\ A\ \Delta\ B & = & sc\ (F \cdot G \cdot \Gamma)\ A\ \Delta\ B \\
sc\ \Gamma\ A\ (F \wedge G \cdot \Delta)\ B & = & sc\ \Gamma\ A\ (F \cdot \Delta)\ B \cup sc\ \Gamma\ A\ (G \cdot \Delta)\ B \\
sc\ (F \vee G \cdot \Gamma)\ A\ \Delta\ B & = & sc\ (F \cdot \Gamma)\ A\ \Delta\ B \cup sc\ (G \cdot \Gamma)\ A\ \Delta\ B \\
sc\ \Gamma\ A\ (F \vee G \cdot \Delta)\ B & = & sc\ \Gamma\ A\ (F \cdot G \cdot \Delta)\ B \\
sc\ (F \to G \cdot \Gamma)\ A\ \Delta\ B & = & sc\ \Gamma\ A\ (F \cdot \Delta)\ B \cup sc\ (G \cdot \Gamma)\ A\ \Delta\ B \\
sc\ \Gamma\ A\ (F \to G \cdot \Delta)\ B & = & sc\ (F \cdot \Gamma)\ A\ (G \cdot \Delta)\ B
\end{array}
$$

Figure 2 Search procedure sc

- The search procedure always terminates in at most n steps, where $n = 2 * \sum (map\ size\ (\Gamma\ @\ \Delta)) + length\ (\Gamma\ @\ \Delta)$ and the *size* of a formula is the number of connectives in it. For the same n we proved:
- $sc\ \Gamma\ []\ \Delta\ [] = \emptyset \longrightarrow mset\ \Gamma \Rightarrow_n mset\ \Delta$
- $(A,\ B) \in sc\ \Gamma\ []\ \Delta\ [] \longrightarrow \lambda k.\ k \in set\ A \not\models mset\ \Gamma \Rightarrow mset\ \Delta$

Generalizations of the latter two facts (with variables for the empty lists) follow by induction on the computation of *sc*. The completeness of SC follows easily. Note that in the final proposition the countermodel is based only on A because disjointness of A and B implies that $\lambda k.\ k \in set\ A$ maps elements in B to *False*.

4.2 Natural Deduction

We have chosen a presentation of Natural Deduction (ND) with explicit contexts.[2] A context Γ is a set of formulas. The set of ND rules is shown in Figure 3. Explicit contexts require rule Ax to get started. Streamlined proofs with explicit contexts require a weakening rule which is proved by induction on $\Gamma \vdash_N F$:

$$\frac{\Gamma \vdash_N F \qquad \Gamma \subseteq \Gamma'}{\Gamma' \vdash_N F}$$

4.2.1 Soundness and Completeness

The semantic meaning of $\Gamma \vdash_N F$ is best described by entailment[3]:

$$\Gamma \Vdash F \stackrel{\text{def}}{=} \forall \mathcal{A}.\ (\forall\, G{\in}\Gamma.\ \mathcal{A} \models G) \longrightarrow \mathcal{A} \models F.$$

Soundness, $\Gamma \vdash_N F \longrightarrow \Gamma \Vdash F$, is shown by induction on $\Gamma \vdash_N F$.

For showing completeness, we follow an approach presented, for example, by Huth and Ryan [20]: show that ND can "simulate truth tables". The translation from assignments to formulas is captured by the notation $F^{\mathcal{A}}$ defined by

[2] This is indispensable for a positive inductive definition: without contexts, ImpI becomes $(\vdash_N F \longrightarrow \vdash_N G) \longrightarrow \vdash_N F \to G$ where the inductive predicate occurs in a negative position $(\vdash_N F)$.

[3] We chose a separate symbol $\Vdash$ for entailment because further overloading of $\models$ necessitates type annotations for disambiguation in too many places.

$$\frac{F \in \Gamma}{\Gamma \vdash_N F}\ \textsc{Ax} \qquad \frac{\{\neg\ F\} \cup \Gamma \vdash_N \bot}{\Gamma \vdash_N F}\ \textsc{CC}$$

$$\frac{\Gamma \vdash_N \neg\ F \qquad \Gamma \vdash_N F}{\Gamma \vdash_N \bot}\ \textsc{NotE} \qquad \frac{\{F\} \cup \Gamma \vdash_N \bot}{\Gamma \vdash_N \neg\ F}\ \textsc{NotI}$$

$$\frac{\Gamma \vdash_N F \wedge G}{\Gamma \vdash_N F}\ \textsc{AndE}_1 \qquad \frac{\Gamma \vdash_N F \wedge G}{\Gamma \vdash_N G}\ \textsc{AndE}_2 \qquad \frac{\Gamma \vdash_N F \qquad \Gamma \vdash_N G}{\Gamma \vdash_N F \wedge G}\ \textsc{AndI}$$

$$\frac{\Gamma \vdash_N F}{\Gamma \vdash_N F \vee G}\ \textsc{OrI}_1 \qquad \frac{\Gamma \vdash_N G}{\Gamma \vdash_N F \vee G}\ \textsc{OrI}_2$$

$$\frac{\Gamma \vdash_N F \vee G \qquad \{F\} \cup \Gamma \vdash_N H \qquad \{G\} \cup \Gamma \vdash_N H}{\Gamma \vdash_N H}\ \textsc{OrE}$$

$$\frac{\{F\} \cup \Gamma \vdash_N G}{\Gamma \vdash_N F \to G}\ \textsc{ImpI} \qquad \frac{\Gamma \vdash_N F \to G \qquad \Gamma \vdash_N F}{\Gamma \vdash_N G}\ \textsc{ImpE}$$

Figure 3 Rules of natural deduction

1. $F \to G \to F$
2. $(F \to G \to H) \to (F \to G) \to F \to H$
5. $(F \wedge G) \to F$
6. $(F \wedge G) \to G$
7. $F \to G \to F \wedge G$
8. $F \to F \vee G$
9. $G \to F \vee G$
10. $(F \to H) \to (G \to H) \to (F \vee G) \to H$
11. $(F \to \bot) \to \neg\ F$
12. $\neg\ F \to F \to \bot$
13. $(\neg\ F \to \bot) \to F$

$$\frac{F \in \Gamma}{\Gamma \vdash_H F}\ \textsc{Ax}$$

$$\frac{\Gamma \vdash_H F \qquad \Gamma \vdash_H F \to G}{\Gamma \vdash_H G}\ \textsc{MP}$$

Figure 4 Axioms and rules of Hilbert systems

$$F^{\mathcal{A}} \stackrel{\text{def}}{=} \mathsf{if}\ \mathcal{A} \models F\ \mathsf{then}\ F\ \mathsf{else}\ \neg\ F$$

First we show two lemmas:

$$\mathit{atoms}\ F \subseteq Z \longrightarrow \{{A_k}^{\mathcal{A}} \mid k \in Z\} \vdash_N F^{\mathcal{A}}$$
$$(\forall \mathcal{A}.\ \{{A_k}^{\mathcal{A}} \mid k \in \{l\} \cup Z\} \vdash_N F) \longrightarrow (\forall \mathcal{A}.\ \{{A_k}^{\mathcal{A}} \mid k \in Z\} \vdash_N F)$$

The first one is proved by induction on F, the second one requires a few lines of careful manual reasoning involving the derived rule

$$\frac{\{F\} \cup \Gamma \vdash_N H \qquad \{\neg\ F\} \cup \Gamma \vdash_N H}{\Gamma \vdash_N H}$$

A limited completeness theorem $\models F \longrightarrow \emptyset \vdash_N F$ follows by downward induction on the size of *atoms* F (removing atom by atom). The general completeness theorem requires compactness and is postponed to Section 6.

4.3 Hilbert Systems

We consider Hilbert systems (HS) based on the axioms and rules shown in Figure 4. Axioms 5 to 10 correspond to those originally postulated by Hilbert [19]. Axioms 1 and 2 are merely

a more compact way of expressing Hilbert's first four axioms. The axioms from 11 onward needed to be adjusted since Hilbert does not use formulas with $\bot$.

The deduction theorem is proved in the standard inductive fashion:

$$\frac{\{F\} \cup \text{Axs}_0 \cup \Gamma \vdash_H G}{\text{Axs}_0 \cup \Gamma \vdash_H F \to G}$$

Here Axs_0 denotes the set of axioms 1 and 2 only. In Section 5 we relate HS to ND and SC, thus obtaining a syntactic proof of soundness and completeness. A semantic proof of completeness is given in Section 9.

5 Translating Between Proof Systems

This section presents results relating derivations in SC, ND and HS. The key results are:

$$\Gamma \vdash_N F \longrightarrow \text{Axs}_1 \cup \Gamma \vdash_H F \tag{2}$$
$$\text{Axs}_1 \cup \{F \mid F \in_\# \Gamma\} \vdash_H G \longrightarrow \Gamma \Rightarrow \{G\}_\# \tag{3}$$
$$\Gamma \Rightarrow \Delta \longrightarrow \{F \mid F \in_\# \Gamma\} \cup \{\neg F \mid F \in_\# \Delta\} \vdash_N \bot \tag{4}$$

where Axs_1 is the full set of axioms from Figure 4. All of these proofs are by induction on the rules of the system in the premise, simulating each rule in the system of the conclusion.

For the proof of (2) the deduction theorem is required wherever ND modifies its context Γ. The only case that needs some manual attention is that of simulating OrE, simply because of its slightly higher complexity.

For (3), we need to show that all Axs_1 axioms are derivable in SC, which can be done by blindly applying matching SC rules. To simulate MP, admissibility of the cut rule is necessary.

For (4), we need a set of rules that execute the reasoning of SC in the context Γ of ND, prefixing a $\neg$ to formulas that would appear in Δ, for example:

$$\frac{\{\neg F, \neg G\} \cup \Gamma \vdash_N \bot}{\{\neg (G \vee F)\} \cup \Gamma \vdash_N \bot} \qquad \frac{\{\neg F\} \cup \Gamma \vdash_N \bot \qquad \{G\} \cup \Gamma \vdash_N \bot}{\{F \to G\} \cup \Gamma \vdash_N \bot}$$

The proofs of these rules are constructed automatically.

Overall we obtain that the following three propositions are equivalent:

$$\text{Axs}_1 \vdash_H F \qquad \emptyset \vdash_N F \qquad \emptyset_\# \Rightarrow \{F\}_\#.$$

6 Compactness

The compactness theorem states that *satisfiability* and *finite satisfiability* of a set of formulas coincide:

$$\textit{sat}\ S \stackrel{\text{def}}{=} \exists \mathcal{A}.\ \forall F \in S.\ \mathcal{A} \models F$$
$$\textit{fin\_sat}\ S \stackrel{\text{def}}{=} \forall s \subseteq S.\ \textit{finite}\ s \longrightarrow \textit{sat}\ s$$

We follow Enderton's proof [12] which is based on an enumeration of all formulas. There is an Isabelle library theory that can automatically derive countability of the type of formulas and define a surjective function from a natural number n to a formula F_n. This enables us to define the saturation of a set of formulas:

$$\frac{C \in S}{S \vdash_R C} \qquad \frac{S \vdash_R C \qquad S \vdash_R D \qquad P_k \in C \qquad N_k \in D}{S \vdash (C - \{k^+\}) \cup (D - \{k^-\})}$$

Figure 5 Rules of Resolution

$$\begin{array}{l} \mathit{saturate}\ S\ 0 = S \\ \mathit{saturate}\ S\ (n+1) = \\ (\mathsf{let}\ S' = \mathit{saturate}\ S\ n;\ S_t = \{F_n\} \cup S';\ S_f = \{\neg\ F_n\} \cup S' \\ \ \mathsf{in\ if}\ \mathit{fin\_sat}\ S_f\ \mathsf{then}\ S_f\ \mathsf{else}\ S_t) \\ \mathit{Saturate}\ S \stackrel{\text{def}}{=} \bigcup_n \mathit{saturate}\ S\ n \end{array}$$

Compactness follows after proving the following lemmas:

$$\begin{array}{l} S \subseteq \mathit{Saturate}\ S \\ \mathit{fin\_sat}\ S \longrightarrow F \in \mathit{Saturate}\ S \longleftrightarrow \neg\ F \notin \mathit{Saturate}\ S \\ \mathit{fin\_sat}\ S \longrightarrow \mathit{fin\_sat}\ (\mathit{Saturate}\ S) \\ \mathit{fin\_sat}\ S \longrightarrow (\lambda k.\ A_k \in \mathit{Saturate}\ S) \models F \longleftrightarrow F \in \mathit{Saturate}\ S \end{array}$$

The proofs are standard.

As an example application of compactness, we show a general completeness theorem for ND: $\Gamma \models F \longrightarrow \Gamma \vdash_N F$. We have proved this both via the completeness result from Section 4.2, compactness and weakening, and via compactness, the SC completeness result from Section 4.1 and (4) from Section 5.

7 Resolution

We have dedicated a separate section to resolution since it is quite different from the other proof systems. To begin with, it uses a different type of formulas: CNFs. A CNF is a set of clauses, where a clause is a set of positive (P_k) or negative (N_k) literals. We use S, T for CNFs, C, D, E for clauses and L for literals. We write the empty clause as $\Box$. The semantics of CNFs (and literals) is defined in a similar fashion to that of formulas, and we will use the same $\models$ for it here:

$$\begin{array}{l} \mathcal{A} \models S \stackrel{\text{def}}{=} \forall C{\in}S.\ \exists L{\in}C.\ \mathcal{A} \models L, \\ \mathcal{A} \models P_k = \mathcal{A}\ k, \quad \mathcal{A} \models N_k = (\neg\ \mathcal{A}\ k). \end{array}$$

To convert a formula into CNF, we first convert it into NNF ($\neg$ is applied only to atoms) by pushing $\neg$ downward and eliminating $\rightarrow$ in the usual fashion. To convert a formula in NNF into CNF, we define another recursive function: *cnf* :: *formula* $\Rightarrow$ *literal set set*. Its only interesting equation is

$$\mathit{cnf}\ (F \vee G) = \{C \cup D \mid C \in \mathit{cnf}\ F \wedge D \in \mathit{cnf}\ G\}.$$

The semantic correctness of these transformations ($\mathcal{A} \models \mathit{cnf}\ (\mathit{nnf}\ F) \longleftrightarrow \mathcal{A} \models F$) is trivial.

Our formalization of resolution as an inductively defined predicate requires two rules, as shown in Figure 5. A resolution refutation of S is a derivation $S \vdash_R \Box$. The following two weakening rules will come in handy:

$$\frac{S \vdash_R C}{T \cup S \vdash_R C}\ (5) \qquad \frac{S \cup T \vdash_R C}{\exists D{\subseteq}\{L\}.\ \{\{L\} \cup E \mid E \in S\} \cup T \vdash_R D \cup C}\ (6)$$

A weaker version of (6) is presented by Gallier [15, §4.3.4] roughly as (we leave out any formal notation): take a resolution refutation graph and insert an additional atom into some of the start nodes' clauses, then apply the same resolution steps as in the original graph. We have to generalize this from assuming a refutation to an arbitrary resolution $S \cup T \vdash_R C$ to show the claim by induction on the derivation.

7.1 Soundness and Completeness

Soundness ($S \vdash_R \square \longrightarrow \forall \mathcal{A}.\ \mathcal{A} \not\models S$) is an easy corollary of the following lemma which is proved by induction on $S \vdash_R C$:

$$S \vdash_R C \wedge \mathcal{A} \models S \longrightarrow \mathcal{A} \models \{C\}$$

We give two proofs of completeness. The first one follows Schöning [40]. Since the proof is an induction on the set of atoms in the CNF, we need the following definitions to manipulate that set:

$$\begin{array}{lcl} k^v & \stackrel{\text{def}}{=} & \mathsf{case}\ v\ \mathsf{of}\ \mathit{True} \Rightarrow P_k \mid \mathit{False} \Rightarrow N_k \\ S[v/k] & \stackrel{\text{def}}{=} & \{C - \{k^{\neg v}\} \mid C \in S \wedge k^v \notin C\} \end{array}$$

The notational similarity with a substitution is deliberate, but beware that there is no new formula or clause that gets substituted in. Instead, the CNF is modified as if that atom index k had been set to the value v:

$$\mathcal{A} \models S[v/k] \longleftrightarrow \mathcal{A}(k := v) \models S$$

We follow Schöning and show

$$(\forall \mathcal{A}.\ \mathcal{A} \not\models S) \wedge \mathit{finite}\ (\mathit{atoms}\ S) \longrightarrow S \vdash_R \square$$

by induction on $\mathit{atoms}\ S$. The base case is trivial since S can only contain the empty clause. In the step case we conclude that if $\forall \mathcal{A}.\ \mathcal{A} \not\models S$, then also $\forall \mathcal{A}.\ \mathcal{A} \not\models S[v/k]$, and hence $S[v/k] \vdash_R \square$ for any v and some $k \in \mathit{atoms}\ S$. For a slick formal proof of the final $S \vdash_R \square$, it is necessary to find a suitable lemma that relates a CNF $S[v/k]$ back to the original CNF S:

$$\{\mathsf{if}\ \{k^{\neg v}\} \cup C \in S\ \mathsf{then}\ \{k^{\neg v}\} \cup C\ \mathsf{else}\ C \mid C \in S[v/k]\} \subseteq S$$

After splitting the set comprehension into two separate sets of clauses based on whether $\{k^{\neg v}\} \cup C \in S$, we can use the two weakening lemmas (5,6) and resolution to finish the proof.

7.2 A Translation Between SC and Resolution

We provide a translation between SC proofs and resolution refutations. This yields a second soundness and completeness argument. The idea is to translate from an SC proof $\Gamma \Rightarrow \emptyset_\#$ into a resolution refutation $\Gamma' \vdash_R \square$ where Γ' is a CNF corresponding to Γ. SC in the form presented in Section 4.1 is not very suitable for this kind of reasoning. Instead, we follow an idea by Gallier [15] and introduce *LSC* (shown in Figure 6), our own variant of Schütte's one-sided calculus K_1 [41]. We prove

$$\Gamma \Rightarrow \emptyset_\# \longleftrightarrow \Gamma \Rightarrow_L$$

To go from LSC to resolution we require a special normal form of formulas. The formula should be a conjunction of disjunctions of literals, but with a twist: the conjunctions can be arbitrarily nested but the disjunctions have to be nested to one side (we chose the right side). This format is captured by the predicate *is_cnf* on formulas. With that, we can show

$$\frac{}{\neg\ A_k,\ A_k,\ \Gamma \Rightarrow_L}\ \textsc{Ax} \qquad \frac{}{\bot,\ \Gamma \Rightarrow_L}\ \textsc{BotL}$$

$$\frac{F,\ \Gamma \Rightarrow_L}{\neg\ (\neg\ F),\ \Gamma \Rightarrow_L}$$

$$\frac{F,\ G,\ \Gamma \Rightarrow_L}{F \wedge G,\ \Gamma \Rightarrow_L}\ \textsc{AndL} \qquad \frac{\neg\ F,\ \Gamma \Rightarrow_L \qquad \neg\ G,\ \Gamma \Rightarrow_L}{\neg\ (F \wedge G),\ \Gamma \Rightarrow_L}$$

$$\frac{F,\ \Gamma \Rightarrow_L \qquad G,\ \Gamma \Rightarrow_L}{F \vee G,\ \Gamma \Rightarrow_L}\ \textsc{OrL} \qquad \frac{\neg\ F,\ \neg\ G,\ \Gamma \Rightarrow_L}{\neg\ (F \vee G),\ \Gamma \Rightarrow_L}$$

$$\frac{\neg\ F,\ \Gamma \Rightarrow_L \qquad G,\ \Gamma \Rightarrow_L}{F \rightarrow G,\ \Gamma \Rightarrow_L} \qquad \frac{F,\ \neg\ G,\ \Gamma \Rightarrow_L}{\neg\ (F \rightarrow G),\ \Gamma \Rightarrow_L}$$

Figure 6 Rules of LSC

$$(\forall\, F{\in_\#}\Gamma.\ \mathit{is\_cnf}\ F) \wedge \Gamma \Rightarrow_L \longrightarrow \bigcup\{\mathit{cnf}\ F \mid F \in_\# \Gamma\} \vdash_R \square$$

by induction on $\Gamma \Rightarrow_L$. The motivation for the special nesting of disjunctions is the OrL case. In order to apply (6) one of the disjuncts must be a literal that can match L in (6). The idea for this trick is borrowed from Gallier [15], but the details of the proofs differ (cf. [26]).

In a second step we can use the lemma above to prove our main result:

$$\{F\}_\# \Rightarrow \emptyset_\# \longrightarrow \mathit{cnf}\ (\mathit{nnf}\ F) \vdash_R \square$$

Doing so requires a number of auxiliary lemmas with laborious proofs about converting deductions of $\Gamma \Rightarrow_L$ to deductions of $\Gamma' \Rightarrow_L$ where Γ' is a variant of Γ such that *is_cnf* holds for all elements of Γ'.

We have also shown $S \vdash_R \square \longrightarrow \exists\, F.\ \mathit{cnf}\ (\mathit{nnf}\ F) \subseteq S \wedge \{F\}_\# \Rightarrow \emptyset_\#$. We do not go into this proof.

7.3 A Toy Resolution Prover

It is possible to generate code from executable Isabelle definitions; inductive definitions are interpreted in a Prolog-like manner. Although applicable to $\vdash_R$, it leads to the usual nontermination issue due to DFS. Thus we implement resolution by hand in two steps. First we define a function *res* that computes all resolvents of a clause set:

$$\begin{array}{l}\mathit{res}\ S \stackrel{\text{def}}{=} (\bigcup C_1 \in S.\ \bigcup C_2 \in S.\ \bigcup L_1 \in C_1.\ \bigcup L_2 \in C_2.\\ \quad \mathsf{case}\ (L_1,L_2)\ \mathsf{of}\\ \quad\ (P_i,N_j) \Rightarrow \mathit{if}\ i = j\ \mathit{then}\ \{(C_1 - \{P_i\}) \cup (C_2 - \{N_j\})\}\ \mathit{else}\ \emptyset\\ \quad \mid \_ \qquad\ \Rightarrow \emptyset)\end{array}$$

Then we iterate *res* until no new clauses are generated:

$$\mathit{Res}\ S = (\mathsf{let}\ R = \mathit{res}\ S \cup S\ \mathsf{in}\ \mathsf{if}\ R = S\ \mathsf{then}\ \mathit{Some}\ S\ \mathsf{else}\ \mathit{Res}\ R)$$

The result is wrapped in **datatype** $'a\ \mathit{option} = \mathit{None} \mid \mathit{Some}\ 'a$ to express if the computation diverged or produced a result a by returning *None* or *Some a*. For more details see [22, 7]. Because of Isabelle's automatic data refinement of sets by lists this is an executable function. We proved soundness and completeness wrt. $\vdash_R$ and termination:

$$\begin{array}{l}\mathit{Res}\ S = \mathit{Some}\ T \longrightarrow (C \in T) = S \vdash_R C\\ \mathit{finite}\ S \wedge (\forall\, C{\in}S.\ \mathit{finite}\ C) \longrightarrow \exists\, T.\ \mathit{Res}\ S = \mathit{Some}\ T\end{array}$$

Of course we can only show termination for finite sets of finite clauses. The termination proof relies on the fact that there is a finite bounding set, the set of all clauses that can be constructed from the atoms in S, because resolution does not introduce new atoms.

Of course *Res* is inefficient and only useful for demonstration purposes. Efficient variants would need much further refinement as in [4, 5].

8 Craig Interpolation

The interpolation theorem states: given an SC derivation $\Gamma_1 + \Gamma_2 \Rightarrow \Delta_1 + \Delta_2$, there exists an *interpolant* G s.t.

$\Gamma_1 \Rightarrow G$, Δ_1 and $G, \Gamma_2 \Rightarrow \Delta_2$ and
atoms $G \subseteq$ *atoms* $(\Gamma_1 + \Delta_1)$ and *atoms* $G \subseteq$ *atoms* $(\Gamma_2 + \Delta_2)$

(A slightly less general version uses $\Gamma_2 = \Delta_1 = \emptyset_\#$.) We follow the proof by Troelstra and Schwichtenberg [48], which is by induction on the depth of the derivation. Instead of formalizing the split sequents used in the original proof, we use multiset unions. Troelstra and Schwichtenberg only provide the cases for Ax, BotL, ImpL and ImpR.

To avoid the other six cases, we rewrite formulas into the implicative fragment, i.e we define a transformation from a formula F to a formula $\overrightarrow{F}$:

$$\begin{array}{lcllcl} \overrightarrow{A_k} & = & A_k & \overrightarrow{\bot} & = & \bot \\ \overrightarrow{F \to G} & = & \overrightarrow{F} \to \overrightarrow{G} & \overrightarrow{F \wedge G} & = & (\overrightarrow{F} \to \overrightarrow{G} \to \bot) \to \bot \\ \overrightarrow{\neg F} & = & \overrightarrow{F} \to \bot & \overrightarrow{F \vee G} & = & (\overrightarrow{F} \to \bot) \to \overrightarrow{G} \end{array}$$

Showing that this transformation preserves the semantics is trivial. Since we do not want to introduce a semantic argument into our proof of interpolation, we also need to show that derivability in SC is preserved by the transformation:

$$\Gamma \Rightarrow \Delta \longleftrightarrow \overrightarrow{\Gamma} \Rightarrow \overrightarrow{\Delta}$$

We show this using one induction over the SC rules for each direction of the logical equivalence. In the direction from left to right, derivations can be transformed easily. The direction from right to left is technically more difficult: the induction will produce two cases where $\to$ was the topmost connective of the last principal formula, but the $\to$ could be the result of rewriting any of the four connectives $\neg$, $\wedge$, $\vee$ or $\to$. Appropriately splitting this into cases requires some manual effort. This means that the reduced set of connectives does not reduce the total proof effort for Craig interpolation dramatically, but we can also apply it to contraction and cut admissibility.

The interpolation theorem is proved by induction on $\Gamma \Rightarrow \Delta$ where $\Gamma = \Gamma_1 + \Gamma_2$ and $\Delta = \Delta_1 + \Delta_2$. In the proof, we can now assume that $\neg$, $\wedge$ and $\vee$ do not occur anywhere, making the corresponding six cases vacuous. The remaining four cases need to be split into a total of 10 subcases distinguishing in which of the multisets Γ_1, Δ_1, Γ_2, Δ_2 the principal formula occurred. In each of the subcases, we provide a witness for G. [4]. Some of the difficulty of the proof lies in finding a way to instantiate the induction hypothesis and then finding a possible interpolation formula for the given instantiation. Consider, for example, the ImpL case with $F \to H$ principal and $F \to H \in_\# \Gamma_1$. For brevity, we define $\Gamma_1{}'$ s.t. $\Gamma_1 = F \to H, \Gamma_1{}'$. Now, we have (at least) two choices: We can follow Troelstra and Schwichtenberg and instantiate the IHs to obtain interpolants G_F and G_H such that:

[4] In [48] one of the witnesses for Ax is $\bot$ where it should be $\bot \to \bot$.

$\Gamma_2 \Rightarrow G_F$, Δ_2 and G_F, $\Gamma_1{}' \Rightarrow F$, Δ_1 and
H, $\Gamma_1{}' \Rightarrow G_H$, Δ_1 and G_H, $\Gamma_2 \Rightarrow \Delta_2$

The new interpolant is then $G_F \rightarrow G_H$. Note that this is not the only possible interpolant. Alternatively we can obtain a G_F that satisfies $\Gamma_1{}' \Rightarrow G_F$, F, Δ_1 and G_F, $\Gamma_2 \Rightarrow \Delta_2$ and use $G_F \vee G_H$ as the interpolant.

While the above proof is fully constructive and provides witnesses for G in all cases, splitting the sequent leads to a lot of technical detail. The next subsection shows an easy semantic alternative.

8.1 Craig Interpolation via Substitution

It is also possible to show that if $\models F \rightarrow H$, then there exists a formula G s.t.

$\models F \rightarrow G$ and $\models G \rightarrow H$ and
$\mathit{atoms}\ G \subseteq \mathit{atoms}\ F \cap \mathit{atoms}\ H$

Doing so is interesting because there is a much easier proof with a different approach to constructing the interpolant G, e.g. as presented by Harrison [18]: provide a uniform recursive function that computes the interpolant:

$\mathit{interpolate}\ F\ H =$
(let $K = \mathit{atoms}\ F - \mathit{atoms}\ H$
 in if $K = \emptyset$ then F
  else let $k = \mathit{Min}\ K$ in $\mathit{interpolate}\ (F[(\bot \rightarrow \bot)/k] \vee F[\bot/k])\ H$)

where $F[G/k]$ substitutes G for A_k in F and Min is (arbitrarily) used to select the minimal index. The proof is by induction on the set $\mathit{atoms}\ F - \mathit{atoms}\ H$. The induction step follows using $\mathcal{A} \models F \longrightarrow \mathcal{A} \models F[(\bot \rightarrow \bot)/k] \vee F[\bot/k]$.

9 The Model Existence Theorem

So far we have invested a lot of work into constructive, or at least insightful proofs. This section shows a way to derive many of the results with much less effort: the Model Existence Theorem, as presented by Fitting [13] (originally by Smullyan [44, 45]). We begin by defining when a set of formulas is a Hintikka set:

$$\begin{aligned}\mathit{Hintikka}\ S \stackrel{\text{def}}{=}\ & \bot \notin S \wedge (\forall k.\ A_k \in S \longrightarrow \neg\ A_k \in S \longrightarrow \mathit{False}) \wedge \\ & (\forall F\ G.\ F \wedge G \in S \longrightarrow F \in S \wedge G \in S) \wedge \\ & (\forall F\ G.\ F \vee G \in S \longrightarrow F \in S \vee G \in S) \wedge \\ & (\forall F\ G.\ F \rightarrow G \in S \longrightarrow \neg\ F \in S \vee G \in S) \wedge \\ & (\forall F.\ \neg\ (\neg\ F) \in S \longrightarrow F \in S) \wedge \\ & (\forall F\ G.\ \neg\ (F \wedge G) \in S \longrightarrow \neg\ F \in S \vee \neg\ G \in S) \wedge \\ & (\forall F\ G.\ \neg\ (F \vee G) \in S \longrightarrow \neg\ F \in S \wedge \neg\ G \in S) \wedge \\ & (\forall F\ G.\ \neg\ (F \rightarrow G) \in S \longrightarrow F \in S \wedge \neg\ G \in S)\end{aligned}$$

Hintikka's lemma shows that Hintikka sets are satisfiable:

$\mathit{Hintikka}\ S \longrightarrow \mathit{sat}\ S$

We continue by defining when a set T of sets of formulas is a propositional consistency property:

$$\begin{aligned} pcp\ T \stackrel{\text{def}}{=}\ & \forall S{\in}T.\ \bot \notin S \wedge (\forall k.\ A_k \in S \longrightarrow \neg\ A_k \in S \longrightarrow \mathit{False}) \wedge \\ & (\forall F\ G.\ F \wedge G \in S \longrightarrow \{F,\ G\} \cup S \in T) \wedge \\ & (\forall F\ G.\ F \vee G \in S \longrightarrow \{F\} \cup S \in T \vee \{G\} \cup S \in T) \wedge \\ & (\forall F\ G.\ F \rightarrow G \in S \longrightarrow \{\neg\ F\} \cup S \in T \vee \{G\} \cup S \in T) \wedge \\ & (\forall F.\ \neg\ (\neg\ F) \in S \longrightarrow \{F\} \cup S \in T) \wedge \\ & (\forall F\ G.\ \neg\ (F \wedge G) \in S \longrightarrow \{\neg\ F\} \cup S \in T \vee \{\neg\ G\} \cup S \in T) \wedge \\ & (\forall F\ G.\ \neg\ (F \vee G) \in S \longrightarrow \{\neg\ F,\ \neg\ G\} \cup S \in T) \wedge \\ & (\forall F\ G.\ \neg\ (F \rightarrow G) \in S \longrightarrow \{F,\ \neg\ G\} \cup S \in T) \end{aligned}$$

Fitting's proof [13] of the Model Existence Theorem

$$pcp\ T \wedge S \in T \longrightarrow sat\ S$$

shows that every *pcp* can be extended to one where every member is the start of a $\subseteq$-chain (constructed in a similar fashion to the sequence in Section 6). The limit of this sequence is a Hintikka set, and by Hintikka's lemma satisfiable. The theorem follows. Our formalization of this proof matches Fitting's proof very closely, and we do not want to repeat it here. Instead, we present three examples where we applied the Model Existence Theorem:

- For SC, we show the lemma

  $$pcp\ \{\{F \mid F \in_\# \Gamma\} \mid \Gamma \not\Rightarrow \emptyset_\#\}.$$

  The proof of this is constructed automatically after dealing with the slight discrepancies between sets and multisets. Completeness (1) follows easily.
- We show compactness using the fact that

  $$pcp\ \{W \mid \mathit{fin\_sat}\ W\}.$$

- We show completeness of HS using

  $$pcp\ \{\Gamma \mid \neg\ (\Gamma \cup \mathrm{Axs}_1 \vdash_H \bot)\}.$$

  This requires a significant amount of manual effort for proving derived rules, e.g.

  $$\Gamma \cup \mathrm{Axs}_1 \vdash_H F \rightarrow G \longrightarrow \Gamma \cup \mathrm{Axs}_1 \vdash_H \neg\ F \vee G$$

  in our Hilbert system.

10 Conclusion

We have presented the formalization of a broad spectrum of calculi and results for classical propositional logic. Although all of the constructions and proofs we formalized "worked" in principle, the distances between the informal starting points and the formal text varied considerably. On one end of the spectrum were beautiful abstract results like the Model Existence Theorem whose proofs could be formalized very easily. On the other end of the spectrum was the translation from sequent calculus proofs into resolution refutations because it required to relate CNFs represented as formulas to CNFs represented as sets of clauses. Proofs about resolution graphs can also become more complicated if they take the form of global modifications of the graph: such one-step proofs required more subtle inductive arguments. Somewhere in the middle of the spectrum are proofs about sequent calculus, e.g. admissibility of cut or syntactic Craig interpolation. These careful syntactic arguments either lead to long manual proofs or require special purpose automation to deal with multisets (or structural rules, depending on the formalization).

This work is a first step towards a basis for the growing collection of formalized logical calculi. There is a plethora of important but non-trivial extensions, e.g. first-order, intuitionistic, or modal logics, that we hope to see formalized. Thiemann *et al.*'s formalization IsaFoR/CeTA [46] of large parts of rewriting theory (starting with [47]) shows that the dream of a unified formalization of logic is achievable and ideally the two efforts will be linked one day.

Acknowledgement

We thank Peter Lammich for a first version of function *sc* and Jasmin C. Blanchette for valuable comments.

References

1 Stefan Berghofer. First-order logic according to Fitting. 2007. http://isa-afp.org/entries/FOL-Fitting.shtml, Formal proof development.

2 Jasmin C. Blanchette, Sascha Böhme, and Lawrence C. Paulson. Extending Sledgehammer with SMT solvers. In N. Bjørner and V. Sofronie-Stokkermans, editors, *Automated Deduction (CADE 2011)*, volume 6803 of *LNCS*, pages 116–130. Springer, 2011.

3 Jasmin C. Blanchette et al. IsaFoL: Isabelle Formalization of Logic. https://bitbucket.org/isafol/isafol.

4 Jasmin C. Blanchette, Mathias Fleury, Peter Lammich, and Christoph Weidenbach. A verified sat solver framework with learn, forget, restart, and incrementality. *Accepted for publication in J. of Automated Reasoning.*

5 Jasmin C. Blanchette, Mathias Fleury, and Christoph Weidenbach. A verified SAT solver framework with learn, forget, restart, and incrementality. In N. Olivetti and A. Tiwari, editors, *Automated Reasoning (IJCAR 2016)*, volume 9706 of *LNCS*, pages 25–44. Springer, 2016.

6 Jasmin C. Blanchette, Andrei Popescu, and Dmitriy Traytel. Unified classical logic completeness. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Automated Reasoning (IJCAR 2014)*, volume 8562 of *LNCS*, pages 46–60. Springer, 2014.

7 Ana Bove, Alexander Krauss, and Matthieu Sozeau. Partiality and recursion in interactive theorem provers - an overview. *Mathematical Structures in Computer Science*, 26(1):38–88, 2016.

8 Patrick Braselmann and Peter Koepke. Gödel's completeness theorem. *Formalized Mathematics*, 13(**1**):49–53, 2005.

9 Peter Chapman, James McKinna, and Christian Urban. Mechanising a Proof of Craig's Interpolation Theorem for Intuitionistic Logic in Nominal Isabelle. In *Proc. AISC/Calculemus/MKM Conference*, pages 38–52. Springer, 2008.

10 Kaustuv Chaudhuri, Leonardo Lima, and Giselle Reis. Formalized meta-theory of sequent calculi for substructural logics. In *Workshop on Logical and Semantic Frameworks, with Applications (LSFA-11)*, 2016.

11 Christian Doczkal and Gert Smolka. Completeness and decidability results for CTL in constructive type theory. *Automated Reasoning*, 56(3):343–365, 2016.

12 Herbert Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

13 Melvin Fitting. *First-Order Logic and Automated Theorem Proving.* Springer, 1990.

14 Andrew Gacek. *A Framework for Specifying, Prototyping, and Reasoning about Computational Systems.* PhD thesis, University of Minnesota, 2009.

15 Jean H Gallier. *Logic for computer science: foundations of automatic theorem proving.* Harper & Row, 1986.

16 John Harrison. Formalizing basic first order model theory. In J. Grundy and M. Newey, editors, *Theorem Proving in Higher Order Logics (TPHOLs '98)*, volume 1497 of *LNCS*, pages 153–170. Springer, 1998.

17 John Harrison. Towards self-verification of HOL Light. In U. Furbach and N. Shankar, editors, *Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNCS*, pages 177–191. Springer, 2006.

18 John Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

19 David Hilbert. Die Grundlagen der Mathematik. pages 1–21. Vieweg+Teubner Verlag, Wiesbaden, 1928.

20 Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press, 2004.

21 Danko Illik. *Constructive Completeness Proofs and Delimited Control*. PhD thesis, École Polytechnique, 2010.

22 Alexander Krauss. Recursive definitions of monadic functions. In Ekaterina Komendantskaya, Ana Bove, and Milad Niqui, editors, *Partiality and Recursion in Interactive Theorem Provers, PAR@ITP 2010*, volume 5 of *EPiC Series*, pages 1–13. EasyChair, 2012.

23 Ramana Kumar, Rob Arthan, Magnus Myreen, and Scott Owens. Self-formalisation of higher-order logic - semantics, soundness, and a verified implementation. *Autom. Reasoning*, 56(3):221–259, 2016.

24 Stephane Lescuyer. *Formalizing and Implementing a Reflexive Tactic for Automated Deduction in Coq*. PhD thesis, Université Paris Sud - Paris XI, 2011.

25 Filip Marić. Formalization and implementation of modern SAT solvers. *Automated Reasoning*, 43(1):81–119, 2009.

26 Julius Michaelis and Tobias Nipkow. Propositional proof systems. *Archive of Formal Proofs*, June 2017. http://isa-afp.org/entries/Propositional_Proof_Systems.html, Formal proof development.

27 Tobias Nipkow. Linear quantifier elimination. *Automated Reasoning*, 45:189–212, 2010.

28 Tobias Nipkow and Gerwin Klein. *Concrete Semantics with Isabelle/HOL*. Springer, 2014. http://concrete-semantics.org.

29 Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

30 Duckki Oe, Aaron Stump, Corey Oliver, and Kevin Clancy. versat: A verified modern SAT solver. In V. Kuncak and A. Rybalchenko, editors, *Verification, Model Checking, and Abstract Interpretation (VMCAI 2012)*, volume 7148 of *LNCS*, pages 24–38. Springer, 2012.

31 Lawrence C. Paulson. A mechanised proof of Gödel's incompleteness theorems using Nominal Isabelle. *Autom. Reasoning*, 55(1):1–37, 2015.

32 Frank Pfenning. Elf: A Language for Logic Definition and Verified Metaprogramming. In *Logic in Computer Science (LICS 1989)*, pages 313–322. IEEE Computer Society Press, 1989.

33 Frank Pfenning. Structural Cut Elimination: I. Intuitionistic and Classical Logic. *Inf. Comput.*, 157(1-2):84–141, 2000.

34 Frank Pfenning and Carsten Schürmann. System description: Twelf — A Meta-Logical Framework for Deductive Systems. In H. Ganzinger, editor, *Automated Deduction — CADE-16*, volume 1632 of *LNCS*, pages 202–206. Springer, 1999.

35 Brigitte Pientka and Andrew Cave. Inductive Beluga: Programming Proofs. In A. Felty and A. Middeldorp, editors, *Automated Deduction (CADE-25)*, volume 9195 of *LNCS*, pages 272–281. Springer, 2015.

36 Tom Ridge and James Margetson. A mechanically verified, sound and complete theorem prover for first order logic. In J. Hurd and T. Melham, editors, *Theorem Proving in Higher Order Logic (TPHOLs 2005)*, volume 3603 of *LNCS*, pages 294–309. Springer, 2005.

37 Anders Schlichtkrull. Formalization of the Resolution Calculus for First-Order Logic. In J. Blanchette and S. Merz, editors, *Interactive Theorem Proving (ITP 2016)*, volume 9807 of *LNCS*, pages 341–357. Springer, 2016.

38 Anders Schlichtkrull. Formalization of the resolution calculus for first-order logic. *J. of Automated Reasoning*, Jan 2018.

39 Julian J. Schlöder and Peter Koepke. The Gödel completeness theorem for uncountable languages. *Formalized Mathematics*, 20(**3**):199–203, 2012.

40 Uwe Schöning. *Logic for Computer Scientists*. Birkhäuser, 1989.

41 Kurt Schütte. Schlußweisen-Kalküle der Prädikatenlogik. *Mathematische Annalen*, 122(1):47–65, 1950.

42 Natarajan Shankar. *Metamathematics, Machines, and Gödel's Proof*. Cambridge University Press, 1994.

43 Natarajan Shankar and Marc Vaucher. The mechanical verification of a DPLL-based satisfiability solver. *Electr. Notes Theor. Comput. Sci.*, 269:3–17, 2011.

44 Raymond M Smullyan. A unifying principal in quantification theory. *Proceedings of the National Academy of Sciences*, 49(6):828–832, 1963.

45 Raymond M. Smullyan. *First-Order Logic*. Springer, 1968.

46 Christian Sternagel, René Thiemann, et al. IsaFoR/CeTA: An Isabelle/HOL formalization of rewriting for certified termination analysis. http://cl-informatik.uibk.ac.at/software/ceta/.

47 René Thiemann and Christian Sternagel. Certification of Termination Proofs Using CeTA. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logic (TPHOLs 2009)*, volume 5674 of *LNCS*, pages 452–468. Springer, 2009.

48 A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, 2nd edition, 2000.

49 Christian Urban and Bozhi Zhu. Revisiting Cut-Elimination: One Difficult Proof Is Really a Proof. In A. Voronkov, editor, *Rewriting Techniques and Applications (RTA 2008)*, volume 5117 of *LNCS*, pages 409–424. Springer, 2008.

50 Kumar Neeraj Verma, Jean Goubault-Larrecq, Sanjiva Prasad, and S. Arun-Kumar. Reflecting BDDs in Coq. In He Jifeng and Masahiko Sato, editors, *ASIAN 2000: 6th Asian Computing Science Conference*, volume 1961 of *LNCS*, pages 162–181. Springer, 2000.

51 Bruno Woltzenlogel Paleo, editor. *Towards an Encyclopaedia of Proof Systems*. College Publications, London, UK, 1 edition, 1 2017.