Proving Isomorphism of First-Order Logic Proof Systems in HOL

Anna Mikhajlova and
Joakim von Wright

Turku Centre for Computer Science
TUCS Technical Report No 169
March 1998
ISBN 952-12-0183-5
ISSN 1239-1891

Abstract

We prove in HOL that three proof systems for classical first-order predicate logic, the Hilbertian axiomatization, the system of natural deduction, and a variant of sequent calculus, are isomorphic. The isomorphism is in the sense that provability of a conclusion from hypotheses in one of these proof systems is equivalent to provability of this conclusion from the same hypotheses in the others. Proving isomorphism of these three proof systems allows us to guarantee that meta-logical provability properties about one of them would also hold in relation to the others. We prove the deduction, monotonicity, and compactness theorems for Hilbertian axiomatization, and the substitution theorem for the system of natural deduction. Then we show how these properties can be translated between the proof systems. Besides, by proving a theorem which states that provability in flattened sequent calculus implies provability in standard sequent calculus, we show how some meta-logical results about Hilbertian axiomatization and natural deduction can be translated to sequent calculus.

Keywords: Formalised mathematics, theorem-proving in higher-order logic, first-order proof theories, meta-logical reasoning, Hilbertian axiomatization, natural deduction, sequent calculus

TUCS Research Group
Programming Methodology Research Group

1 Introduction

We use the theorem proving system HOL [4] to prove meta-logical results about proof systems for classical first-order predicate logic. In particular, we establish isomorphism among the Hilbertian axiomatization (HA), the system of natural deduction (ND), and a variant of sequent calculus, which we call *flattened sequent calculus*.

We formalize syntactic constructs of first-order logic, terms and formulae, as well as the notions of an inference rule, an inference and a derivation. Then, for every axiomatization that we study, we define a set of its basic inference rules. Every proof system takes different sets of basic connectives. We present inference rules for expanding the defined connectives into the basic and contracting combinations of basic connectives into the defined connectives.

We define in HOL a proof-checking relation which is used to verify correctness of first-order logic proofs in all the proof systems. Using this relation, it is possible to check proofs based not only on basic inference rules, but on derived ones as well. When a particular derivation is shown to be a proof, its hypotheses and conclusion are used to construct a derived inference rule, which is given a name and used further on in derivations. In this manner we can build compositional derivations and check their correctness.

Sequent calculus (SC) represents a special case, being a proof system for sequents with several (more than one) formulae in the succedent. It is well-known [3] that the classical sequent calculus is more general than the other proof systems we consider, in the sense that to every SC proof there correspond several proofs in Hilbertian style or natural deduction. We formalize this idea by defining flattened sequent calculus (FlatSC) and proving that its every basic inference rule is provable in sequent calculus and that flattened sequent calculus is isomorphic to the Hilbertian axiomatization and the system of natural deduction.

Every proof system we consider has features which make it particularly suitable for a collection of applications. For instance, `PROLOG` is an implementation of a fragment of sequent calculus, and the "tableaux" used in automatic theorem-proving are just a special case of this calculus. The natural deduction technique is widely used for teaching of logic but makes the demonstration of meta-logical results somewhat difficult. On the other hand, Hilbertian axiomatization is particularly suited for meta-logical proofs but is quite cumbersome to work with. All of these proof systems, in one form or another, form the basis for various theorem-proving systems. HOL itself is based on a sequent formulation of natural deduction which is fairly close to our formalization; Isabelle is also based on a variant of natural deduction, and it allows formal meta-logical reasoning. Sequent calculus forms the basis for PVS, and the Metamath system [7] is based on a Hilbert-style axiomatization.

There are intrinsic interdependencies among all these mainstream proof systems, for instance, every deduction in the system of natural deduction is a translation of some proof in the sequent calculus. The latter is effectively a calculus of proof search for natural deductions. In this work we focus on these interdependencies and the implicit isomorphism among these proof systems, and analyze these issues formally in HOL. We use higher-order logic as the metalogic for reasoning about first-order proof systems, whereas, usually this kind of reasoning is done in a natural language, e.g. [6, 3, 9]. By using a formal system as the metalogic and formalizing the proofs in a theorem-proving environment, we reduce susceptibility to errors.

The main theorem we prove here states that whenever a conclusion is provable from hypotheses in Hilbertian axiomatization, it is also provable from the same hypotheses in the system of natural deduction, and in flattened sequent calculus. Proving isomorphism of these three proof systems allows us to guarantee that meta-logical properties proved about one of them would hold for the others as well. The properties we prove about Hilbertian axiomatization are the deduction, monotonicity, and compactness theorems. The substitution theorem is proved to hold for the system of natural deduction. We show how these properties can be translated between the systems. Besides, by proving a theorem which states that provability in flattened sequent calculus implies provability in SC, we show how some meta-logical results about Hilbertian axiomatization and natural deduction can be translated to sequent calculus.

Notation

When referring to HOL objects, we use `typewriter font`. The ASCII character combinations used by HOL `/\`, `\/`, and `==>` correspond to ordinary logical symbols $\wedge, \vee$ and $\Rightarrow$, whereas `?` and `!` stand for the existential and the universal quantification.

2 Representing First-Order Logic Concepts in HOL

To reason about proofs in first-order logic and the relationship among different proof systems, we need to formalize syntactic constructs of the logic. These constructs are built from the underlying alphabet.

In this alphabet, we distinguish a logical alphabet and a symbol set. The logical alphabet includes the symbols of logic used in every first-order language, namely, a set of individual variables ranging over the elements of the structure under consideration, the logical constant $\perp$, the logical connectives $\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg$, the universal quantifier $\forall$, and the existential quantifier $\exists$. We also use a symbol $=$ for expressing the equality of two elements. Another part of our first-order alphabet, the symbol set S, contains special symbols used in the mathematical theory to be discussed, representing sets of constants, function symbols, and relation symbols. Out of the alphabet elements we build two kinds of expressions of the first-order language, terms and formulae.

The term and formula structures have little impact on proofs and, hence, proof systems. Being modular, our formalization of first-order logic allows us to slot in alternative representations of terms and formulae. We have chosen to formalize terms in the style most convenient for mechanized manipulation, namely that of de Bruijn [1].

2.1 Terms and Formulae

Our main motivation for the chosen representation of terms is to simplify handling of bound variables. Using the "traditional" representation leads to a very complicated definition of substitution of one term for a variable in another term and greatly complicates proofs involving term substitutions. All these problems arise from the necessity to deal with bound variables which have no individuality and only serve as place markers. Following [1], we no longer give bound variables names from the set of individual variables. Instead, they are distinguished from free variables and indexed by natural numbers, the number indicating which outer $\lambda-$binding they correspond to. For example, instead of $\lambda\, f\, x \cdot f\, x$ we write $\lambda\,(\lambda\,(1\;0))$, because f corresponds to two binders out. Since we number binders from the inside out, none of the numbers change when a given term becomes part of some larger term. We may, of course, have free variables e.g., $\lambda\, x\;\cdot\; f\;x$ is just $\lambda\,(f\;\;0)$. Using this representation of terms not only simplifies their substitution, but also eliminates the need for side conditions in inference rules, because no free variables can be captured in substitution or quantification introduction.

We define a term as a constant, a variable, a bound variable, a lambda abstraction or an application of terms. In HOL we define a new type `Sterm`:

```
Sterm  = Sconst of string
        | Svar of string
        | Sbvar of num
        | Sabs of Sterm
        | Scomb of Sterm => Sterm
```

Notice that the formal syntax now has separate constructs for free variables and bound variables, `Svar` and `Sbvar` respectively. As opposed to the "traditional" representation, we don't have an identification of a bound variable in an abstraction. In general, the formal syntax of non-typed $\lambda-$calculus allows construction of ill-formed or meaningless terms, like an application of a variable term to itself, `Scomb (Svar "x") (Svar "x")`. We do not check for well-formedness of terms, but for our purposes this condition is not critical.

To manipulate terms, we define a number of functions recursing on the structure of a term. A function `STermSubst` is used to substitute a variable in a term with another term:

```
|- (!t x str. STermSubst t x (Sconst str) = Sconst str) /\
   (!t x str. STermSubst t x (Svar str) =
       ((str = x) => t | (Svar str))) /\
   (!t x n. STermSubst t x (Sbvar n) = Sbvar n) /\
   (!t x s. STermSubst t x (Sabs s) = Sabs (STermSubst t x s)) /\
   (!t x s r. STermSubst t x (Scomb r s) =
       Scomb (STermSubst t x r) (STermSubst t x s))
```

We also define two functions for binding and "unbinding" variables in a term. The binding function `STermBind` takes a variable and a term, and returns a lambda abstraction where all occurrences of the variable are bound with a corresponding `Sbvar` instance. The function `STermBeta` performs a beta-conversion on the given `Scomb` term.

Now, after providing the definition of `Sterm`, we can represent the type of first-order logic formulae. A formula over a symbol set S is a falsum $\perp$, an n-ary relation among `Sterm`s, an equality of two `Sterm`s, or a formula constructed from other formulae by using logical connectives and quantifiers. Following [9], we consider first-order logic with equality and, accordingly, distinguish a binary equality relation from other relations and treat it as a special case. In HOL the new type `Sformula` is defined as follows:

```
Sformula = False
          | Srel of string => (Sterm list)
          | Seq of Sterm => Sterm
          | Snot of Sformula
          | Sand of Sformula => Sformula
          | Sor of Sformula => Sformula
          | Simp of Sformula => Sformula
          | Sequiv of Sformula => Sformula
          | Sforall of Sformula
          | Sexists of Sformula
```

To perform substitutions in a formula, we define two functions, `SFormulaTermSubst`, which substitutes a term for a variable, and `SubFormulaSubst`, which substitutes a subformula with another formula. Functions `SFUnivQuantify` and `SFExistQuantify` are used to quantify a formula universally and existentially. Both take a variable to be quantified on and a formula, and return a respectively quantified formula. A function `SFBetaReduce` is used to specialize a quantified formula.

Having defined formulae and operations on them in this way, we can now build syntactic constructs based on formulae.

2.2 Sequents

A sequent is an expression $\Gamma \vdash \Delta$, where Γ and Δ are finite sets of formulae $p_1, ..., p_n$ and $q_1, ..., q_m$. We refer to Γ as the antecedent of a sequent and to Δ as its succedent. In our presentation we will simplify the notation and write p and Γ, p for $\{p\}$ and $\Gamma \cup \{p\}$ respectively. We use sequents to model derivability of Δ from Γ in all proof systems. In HOL we define the type of sequents to be

```
Ssequent = Sseq of (Sformula set) => (Sformula set)
```

In both Hilbertian axiomatization and the system of natural deduction Δ is a singleton set, and its only element is derivable from Γ. In sequent calculus the denotational interpretation is that the conjunction of the p_i implies the disjunction of the q_j. In particular, if Γ and Δ are both empty, the sequent asserts contradiction.

2.3 Inference Rules and Inferences

Inference rules are represented as named sets of pairs (premise set, conclusion), where the premises and the conclusion are sequents. Axiomatic rules have empty sets of premises. In HOL the type `InfRule` is defined to represent inference rules:

```
InfRule = Rule of string => ((Ssequent set) # Ssequent) set
```

Every individual rule is described using an inductive unary relation of type `InfRule -> bool`. For example, the conjunction introduction rule in the system of natural deduction

$$\frac{\Gamma \vdash p \qquad \Delta \vdash q}{\Gamma, \Delta \vdash p \wedge q}\ \mathtt{ConjI}$$

is defined in HOL by the following theorem:

```
|- ConjI
   (Rule "ConjI" {hyps,concl | ?p q Gamma Delta.
      (hyps = {Sseq Gamma {p}; Sseq Delta {q}}) /\
      (concl = Sseq (Gamma UNION Delta) {p Sand q})})
```

Individual inferences are instantiated by the type `Sinference` as named pairs (premise set, conclusion):

```
   Sinference = Sinf of string => ((Ssequent set) # Ssequent)
```

Functions `InfHyps` and `InfConcl` give respectively the hypotheses and the conclusion of an inference.

All individual inferences are supposed to be instances of certain inference rules. A function `CorrectRuleInst:(InfRule set) -> Sinference -> bool` is used to check whether a particular inference is an instance of the identically named inference rule in a certain set of rules.

2.4 Connective Exchange Rules

Every proof system is based on a certain set of basic logical connectives. Other connectives are defined in terms of the basic ones. The connective exchange rules pertinent to a particular proof system arise from these definitions. For example, Hilbertian axiomatization takes implication and falsum as basic; this is a functionally complete set and all other connectives, negation, conjunction, disjunction, and equivalence, can be defined in terms of these two. The connective exchange rules are symmetric, as they represent contraction and expansion of connectives of interest in terms of the others. For example, since the logical negation is defined by $\neg p \mathrel{\widehat{=}} p \Rightarrow \bot$, we can formulate rules for contraction and expansion of $\neg$ as follows:

$$\frac{\Gamma \ \vdash \ \varphi[p \Rightarrow \bot]}{\Gamma \ \vdash \ \varphi[\neg p]}\ \texttt{NotContr} \qquad \frac{\Gamma \ \vdash \ \varphi[\neg p]}{\Gamma \ \vdash \ \varphi[p \Rightarrow \bot]}\ \texttt{NotExp}$$

For every connective exchange rule of such form there is a corresponding one with exchange in the antecedents of the sequents:

$$\frac{\Gamma, \varphi[p \Rightarrow \bot] \ \vdash \ \Delta}{\Gamma, \varphi[\neg p] \ \vdash \ \Delta}\ \texttt{NotContr} \qquad \frac{\Gamma, \varphi[\neg p] \ \vdash \ \Delta}{\Gamma, \varphi[p \Rightarrow \bot] \ \vdash \ \Delta}\ \texttt{NotExp}$$

In HOL we define a unary inductive relation `NotExch` which holds of the rules `NotContr` and `NotExp`. The definitions of these rules employ the subformula substitution function `SubFormulaSubst`.

The exchange rules for the other connectives arise from the definitions

$$\begin{aligned} p \wedge q &\mathrel{\widehat{=}} (p \Rightarrow \ (q \Rightarrow \bot)) \Rightarrow \bot \\ p \vee q &\mathrel{\widehat{=}} \neg(\neg p \wedge \neg q) \\ p \Leftrightarrow q &\mathrel{\widehat{=}} (p \Rightarrow q) \wedge (q \Rightarrow p) \end{aligned}$$

and are as follows:

$$\frac{\Gamma \ \vdash \ \varphi[(p \Rightarrow (q \Rightarrow \bot)) \Rightarrow \bot]}{\Gamma \ \vdash \ \varphi[p \wedge q]}\ \texttt{AndContr} \qquad \frac{\Gamma \ \vdash \ \varphi[p \wedge q]}{\Gamma \ \vdash \ \varphi[(p \Rightarrow (q \Rightarrow \bot)) \Rightarrow \bot]}\ \texttt{AndExp}$$

$$\frac{\Gamma \ \vdash \ \varphi[\neg(\neg p \wedge \neg q)]}{\Gamma \ \vdash \ \varphi[p \vee q]}\ \texttt{OrContr} \qquad \frac{\Gamma \ \vdash \ \varphi[p \vee q]}{\Gamma, \ \vdash \ \varphi[\neg(\neg p \wedge \neg q)]}\ \texttt{OrExp}$$

$$\frac{\Gamma \ \vdash \ \varphi[(p \Rightarrow q) \wedge (q \Rightarrow p)]}{\Gamma \ \vdash \ \varphi[p \Leftrightarrow q]}\ \texttt{EqContr} \qquad \frac{\Gamma \ \vdash \ \varphi[p \Leftrightarrow q]}{\Gamma \ \vdash \ \varphi[(p \Rightarrow q) \wedge (q \Rightarrow p)]}\ \texttt{EqExp}$$

We define the set of all connective exchange rules as `ConnExch`:

```
|- ConnExch = {r | NotExch r \/ AndExch r \/ OrExch r \/ EqExch r}
```

2.5 Derivations and Proofs

A derivation is represented as a tree of inferences. The conclusion of the derivation appears at the root of the tree and the hypotheses appear at the leaves. In HOL derivations are of type `Sinference ltree`, where `(*)ltree` is the type of labeled trees. The theory of labeled trees in HOL defines tree constructors as

$$\texttt{Node} : * \to ((*)\texttt{ltree})\texttt{list} \to (*)\texttt{ltree}$$

A derivation is defined to be a proof if every inference in this derivation is a correct instance of a certain inference rule and all hypotheses of this inference are conclusions of some inferences appearing earlier in the proof.

We define an inductive relation `IsProof` which identifies those derivations that prove their conclusions from the premises with respect to a given set of inference rules. We can use `IsProof` to check correctness of derivations in all proof systems by supplying it with the corresponding set of inference rules. Given a derivation, a set of inference rules, and a conclusion, `IsProof` traverses the derivation tree and checks that every inference in this derivation is an instance of the specified inference rule. It also makes sure that the set of subderivation conclusions for a particular inference includes every premise of this inference. In HOL `IsProof` is a relation of type

```
(Sinference ltree) list -> InfRule set -> Ssequent set -> bool
```

The first argument is a list because at derivation levels other than the root level we need to deal with lists of derivations. A set of inference rules of the current axiomatization should be provided as the second argument. The third argument is the set of inference conclusions at the current level in the derivation tree. The following rules constitute the definition of `IsProof`:

```
IsProof_RULE =
|- (!rSet. IsProof [] rSet {}) /\
   (!S tlConclSet rSet tl. IsProof tl rSet tlConclSet ==>
      IsProof (CONS (Node (Sinf "hyp" ({S},S)) []) tl)
         rSet (S INSERT tlConclSet)) /\
   (!tlConclSet tl inf rSet.
      CorrectRuleInst rSet inf /\ IsProof tl rSet tlConclSet ==>
      IsProof (CONS (Node inf []) tl) rSet
         (InfConcl inf INSERT tlConclSet)) /\
   (!tlConclSet tl dList inf rSet.
      CorrectRuleInst rSet inf /\ IsProof tl rSet tlConclSet /\
      (?upConclSet. InfHyps inf SUBSET upConclSet /\
         IsProof dList rSet upConclSet) ==>
      IsProof (CONS (Node inf dList) tl) rSet
         (InfConcl inf INSERT tlConclSet))
```

In addition to the notion of correct derivation identified by the relation `IsProof`, we define the notion of provability. The inductive relation `IsProvable` specifies when a sequent is provable from a set of sequents with respect to a set of inference rules:

```
IsProvable_RULE =
|- (!h c rSet.
   (?d h'. IsProof [d] rSet {c} /\ DerHyps d h' /\ h' SUBSET h)  ==>
      IsProvable rSet h c) /\
```

```
(!h c hyps rSet. IsProvable rSet hyps c /\
   (!el. el IN hyps ==> IsProvable rSet h el) ==>
   IsProvable rSet h c) /\
(!c h derived rSet. IsProvable (rSet UNION derived) h c /\
   (!r. r IN derived ==> (?inf. CorrectRuleInst {r} inf /\
      IsProvable rSet (InfHyps inf) (InfConcl inf))) ==>
   IsProvable rSet h c) /\
(!c h rSet. IsProvable (rSet UNION ConnExch) h c ==>
   IsProvable rSet h c)
```

We distinguish four cases. First of all, a sequent `c` is provable from the set of sequents `h` in the proof system with inference rules `rSet` if there exist a derivation `d` and a set of hypotheses `h'` such that `d` is a proof of the conclusion `c` from the hypotheses `h'` and all hypotheses `h'` are in `h`. Secondly, a sequent `c` is provable from `h` in `rSet` if it is provable using the same set of inference rules from some set of hypotheses `hyps`, such that every element in this set is in its turn provable from `h`. This rule encodes transitivity of provability. We choose to formalize it as a rule rather than prove it as a theorem, using the first provability rule, because of the difficulty presented by building a derivation as a tree of inferences. Namely, the proof would require splitting a derivation tree into transitive subtrees, which appears to be impractical. The third rule formalizes provability of conclusion `c` from the hypotheses `h` using the set of basic and derived rules. The rule states that whenever every derived rule, used in the proof of `c` from `h`, has itself a proof in the system with basic rules only, we can conclude that `c` is provable from the hypotheses `h` using just these basic rules. And, finally, the last rule formalizes provability using the set of connective exchange rules in addition to the logical rules. The connective exchange rules are often used implicitly, and this rule allows us to "hide" their application.

In practice derived rules are constructed in the following way. When a derivation of a certain conclusion from certain hypotheses is shown to be a proof, we define a derived rule similarly to the way basic rules have been defined. The hypotheses sequents and the conclusion sequent of the derivation are taken to be the hypotheses and the conclusion of the new inference rule. This new rule is given a name and can further on be used in derivations along with basic rules. In this manner we can construct modular derivations and prove their correctness much easier.

3 The Proof Systems of First-Order Logic and Their Isomorphism

The main theorem proving the isomorphism states that whenever a conclusion sequent `c` is provable from hypotheses sequents `h` in Hilbertian axiomatization, given by the corresponding set of basic inference rules `HARules`, it is also provable from the same hypotheses in the system of natural deduction, given by `NDRules`, and in the flattened sequent calculus, given by `FlatSCRules`:

```
IsomorphismThm =
|- !h c. (IsProvable NDRules h c = IsProvable HARules h c) /\
         (IsProvable FlatSCRules h c = IsProvable NDRules h c)
```

We prove this theorem by transitivity of implication, showing that provability in ND implies provability in HA, and provability in HA implies provability in flattened sequent calculus, and, finally, that provability in FlatSC implies provability in natural deduction. These proofs, in turn, reduce to proving that every basic inference of natural deduction has a proof of its conclusion from its hypotheses in Hilbertian axiomatization, and similarly for the other two cases.

The second main result we prove here relates provability in (standard) sequent calculus to provability in the other three proof systems. As we have already mentioned in the introduction, sequent calculus manipulates sequents with more than one formulae in the succedent, whereas, the other proof systems have exactly one formula there. We define the notion of flattening a sequent so that it has a singleton succedent set, and prove that if a conclusion sequent `c` is provable from hypotheses sequents `h` in any of FlatSC, HA, and ND, then it is also provable from the same hypotheses in sequent calculus. We depict these relationships diagrammatically in Fig.1.

3.1 Hilbertian Axiomatization

The first formal system we present is the Hilbertian axiomatization, which we formalize following [2]. This axiomatization is a good starting point for reasoning about properties of the logic because of its formal simplicity. It is particularly convenient for proving the deduction, monotonicity, and compactness theorems which have important implications.

As we have mentioned above, Hilbertian axiomatization takes $\perp$ and $\Rightarrow$ as basic connectives. The following rules constitute the set of axioms and

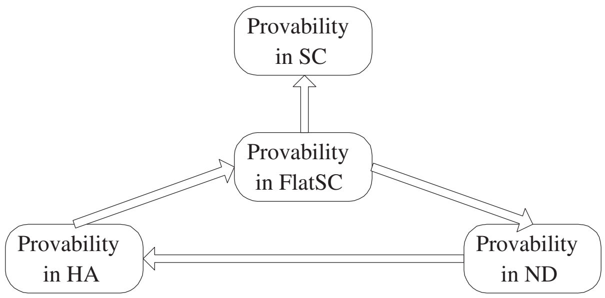

Figure 1: Relationship among provability in different proof systems

basic inference rules of HA:

$\dfrac{}{\Gamma\ \vdash\ p}\ (p \in \Gamma)$	`IdentRule`
$\dfrac{}{\Gamma\ \vdash\ p \Rightarrow (q \Rightarrow p)}$	`Ax1`
$\dfrac{}{\Gamma\ \vdash\ (p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))}$	`Ax2`
$\dfrac{}{\Gamma\ \vdash\ ((p \Rightarrow \bot) \Rightarrow \bot) \Rightarrow p}$	`Ax3`
$\dfrac{\Gamma\ \vdash\ p \qquad \Delta\ \vdash\ p \Rightarrow q}{\Gamma, \Delta\ \vdash\ q}$	`MP`
$\dfrac{}{\Gamma\ \vdash\ p \Rightarrow \forall x.p[x]}$	`ForallIntro`
$\dfrac{}{\Gamma\ \vdash\ \forall x.p[x] \Rightarrow p[t]}$	`ForallElim`
$\dfrac{}{\Gamma\ \vdash\ (\forall x.p \Rightarrow q) \Rightarrow ((\forall x.p) \Rightarrow (\forall x.q))}$	`ForallDistr`

Having defined for every rule a corresponding identically named unary relation, we define the set of basic inference rules as **`HARules`**:

```
|- HARules = {r | IdentRule r \/ Ax1 r \/ Ax2 r \/ Ax3 r \/ MP r \/
                  ForallIntro r \/ ForallElim r \/ ForallDistr r}
```

We also derive the corresponding provability theorems, which we call `HAProvability_RULES`:

```
|- !hyps Gamma p. p IN Gamma ==>
   IsProvable HARules hyps (Sseq Gamma {p}),
|- !hyps Gamma p q.
   IsProvable HARules hyps (Sseq Gamma {p Simp q Simp p}),
|- !hyps Gamma p q r. IsProvable HARules hyps
      (Sseq Gamma {(p Simp q Simp r) Simp (p Simp q) Simp p Simp r}),
|- !hyps Gamma p. IsProvable HARules hyps
      (Sseq Gamma {((p Simp False) Simp False) Simp p}),
|- !hyps Delta1 Delta2 p q.
   IsProvable HARules hyps (Sseq Delta1 {p}) /\
   IsProvable HARules hyps (Sseq Delta2 {p Simp q}) ==>
   IsProvable HARules hyps (Sseq (Delta1 UNION Delta2) {q}),
|- !hyps Gamma p x.
   IsProvable HARules hyps (Sseq Gamma {p Simp (SFUnivQuantify x p)}),
|- !hyps Gamma p tm. IsProvable HARules hyps
      (Sseq Gamma {(Sforall p) Simp (SFBetaReduce (Sforall p) tm)}),
|- !hyps Gamma p q. IsProvable HARules hyps (Sseq Gamma
      {(Sforall (p Simp q)) Simp ((Sforall p) Simp (Sforall q))})
```

The formal simplicity of Hilbertian axiomatization facilitates proving useful properties about this proof system. In what follows we prove the deduction theorem, the monotonicity theorem, and the compactness theorem.

3.1.1 The Deduction Theorem for Hilbertian Axiomatization

The deduction theorem states that for all formulae p and q and all sets of formulae Γ if q is provable from Γ, then $p \Rightarrow q$ is provable from Γ without p:

$$\frac{\Gamma \ \vdash \ q}{\Gamma \setminus p \ \vdash \ p \Rightarrow q} \ \texttt{DeductRule}$$

In HOL the corresponding theorem is expressed as follows:

```
DeductionThm =
|- !Gamma p q.
   IsProvable HARules {Sseq Gamma {q}}
     (Sseq (Gamma DELETE p) {p Simp q}))
```

A general theorem `ProvImplThm` states that a sequent `s2` is provable from a sequent `s1` if provability of `s1` from hypotheses `hyps` implies provability of `s2` from the same hypotheses:

```
ProvImplThm =
|- !rules s1 s2.
   (!h. IsProvable rules h s1 ==> IsProvable rules h s2) ==>
   IsProvable rules {s1} s2)
```

This theorem allows us to reduce the proof of `DeductionThm` to a proof of a lemma

```
|- !hyps Gamma p q. IsProvable HARules hyps (Sseq Gamma {q}) ==>
   IsProvable HARules hyps (Sseq (Gamma DELETE p) {p Simp q})
```

which we prove by induction on `HARules`. The subgoals we should prove are as follows:

1. ```
   q IN Gamma ==>
   IsProvable HARules hyps (Sseq (Gamma DELETE p) {p Simp q})
   ```
2. ```
   IsProvable HARules hyps (Sseq (Gamma DELETE p)
       {p' Simp q Simp p'})
   ```
3. ```
   IsProvable HARules hyps (Sseq (Gamma DELETE p)
       {p Simp ((q Simp False) Simp False) Simp q})
   ```
4. ```
   IsProvable HARules hyps (Sseq (Delta1 DELETE p)
       {p Simp p' Simp q}) /\
   IsProvable HARules hyps (Sseq (Delta2 DELETE p)
       {p Simp p'})  ==>
   IsProvable HARules hyps
      (Sseq ((Delta1 UNION Delta2) DELETE p) {p Simp q})
   ```
5. ...

and similarly for the axioms concerning quantification manipulation. We resolve the axiomatic subgoals by matching them with an auxiliary theorem

```
DeductAxThm =
|- !hyps p q Gamma. IsProvable HARules hyps (Sseq Gamma {q}) ==>
   IsProvable HARules hyps (Sseq Gamma {p Simp q})
```

and rewriting with the corresponding theorems in HAProvability_RULES. To resolve the subgoal based on the rule Modus Ponens, we prove a theorem

```
DeductMPThm =
|- !hyps p q r Gamma Delta.
   IsProvable HARules hyps (Sseq Gamma {p Simp q Simp r}) /\
   IsProvable HARules hyps (Sseq Delta {p Simp q}) ==>
   IsProvable HARules hyps (Sseq (Gamma UNION Delta) {p Simp r})
```

Rewriting with this theorem completes the proof of the deduction lemma, and, consequently, the proof of DeductionThm.

3.1.2 The Monotonicity Theorem for Hilbertian Axiomatization

The second fundamental property of our proof system states that adding formulae to the antecedent set of a sequent is monotonic, in the sense that if $\Gamma \vdash p$ then $\Gamma \cup \Delta \vdash p$, for a finite Δ. We prove this monotonicity property as a theorem in HOL:

```
MonoThm =
|- !Gamma Delta p. FINITE Delta =>
   IsProvable HARules {Sseq Gamma {p}} (Sseq (Gamma UNION Delta) {p})
```

The proof of MonoThm is performed by induction on both the set Delta and the basic inference rules of Hilbertian axiomatization, HARules. Inducting first on HARules results in the following subgoals:

```
1. p IN Gamma ==> !Delta. FINITE Delta ==>
      IsProvable HARules hyps (Sseq (Gamma UNION Delta) {p})

2. !Delta. FINITE Delta ==>
      IsProvable HARules hyps
         (Sseq (Gamma UNION Delta) {p Simp q Simp p})

3. !Delta. FINITE Delta ==> FINITE Delta2 ==>
      IsProvable HARules hyps (Sseq (Delta2 UNION Delta) {p}) /\
   !Delta. FINITE Delta ==> FINITE Delta1 ==>
      IsProvable HARules hyps
         (Sseq (Delta1 UNION Delta) {p Simp q}) ==>
   !Delta. FINITE Delta ==>  FINITE (Delta1 UNION Delta2) ==>
      IsProvable HARules hyps
            (Sseq ((Delta1 UNION Delta2) UNION Delta) {q})

4. ...
```

To solve the first subgoal, we apply the induction principle for finite sets and resolve the case when Delta is equal to the empty set by matching it against the first of HAProvability_RULES. To resolve the induction case, we derive in HA a new rule

$$\frac{\Gamma \ \vdash \ p}{\Gamma, q \ \vdash \ p}\ \texttt{MonoOne}$$

and prove the corresponding theorem:

```
MonoOneThm =
|- !hyps Gamma p q. IsProvable HARules hyps (Sseq Gamma {p}) ==>
   IsProvable HARules hyps (Sseq (q INSERT Gamma) {p})
```

Rewriting the first subgoal using this theorem completes its proof. All the remaining subgoals are resolved by inducting on finiteness of `Delta` and rewriting with the corresponding rule in `HAProvability_RULES`.

3.1.3 The Compactness Theorem for Hilbertian Axiomatization

Compactness is another fundamental property that we prove about Hilbertian axiomatization. The compactness theorem states that if a formula `p` is provable from a set of formulae `Gamma`, then there exists a finite subset `Delta` of `Gamma`, such that `p` is provable from `Delta`.

```
CompactThm =
|- !hyps Gamma p. IsProvable HARules hyps (Sseq Gamma {p}) ==>
   (?Delta. Delta SUBSET Gamma /\ FINITE Delta /\
      IsProvable HARules hyps (Sseq Delta {p}))
```

This theorem is also proved by induction on `HARules`.

Despite its formal simplicity, Hilbertian axiomatization is quite inconvenient to work with. Proofs and derivations are unintuitive and it's usually very difficult to invent them even in simple cases. In this respect the system of natural deduction is by far superior to HA. We demonstrate that higher-level modular rules of the system of natural deduction are the derived rules of HA. For this purpose, we first present ND and then derive its basic inference rules in the Hilbertian axiomatization.

3.2 System of Natural Deduction

The fundamental property of the system of natural deduction is the symmetry of its introduction and elimination rules. The only axiomatic rule that we have in our system states that from a set of assumptions containing a formula we can always conclude that the formula is true. The basic introduction and elimination rules are restricted to those for three connectives, $\wedge, \Rightarrow$ and $\bot$. The rules for other connectives are derived. We also have a pair of introduction and elimination rules for universal quantification. Existential quantification rules are derived from these rules.

$$\frac{}{\Gamma \vdash p}\ (p \in \Gamma)\ \texttt{IdRule}$$

$$\frac{\Gamma \vdash p \qquad \Delta \vdash q}{\Gamma, \Delta \vdash p \wedge q}\ \texttt{ConjI} \qquad \frac{\Gamma \vdash p \wedge q}{\Gamma \vdash p} \quad \frac{\Gamma \vdash p \wedge q}{\Gamma \vdash q}\ \texttt{ConjE}$$

$$\frac{\Gamma, p \vdash q}{\Gamma \vdash p \Rightarrow q}\ \texttt{ImpI} \qquad \frac{\Gamma \vdash p \qquad \Delta \vdash p \Rightarrow q}{\Gamma, \Delta \vdash q}\ \texttt{ImpE}$$

$$\frac{\Gamma, p \Rightarrow \bot \vdash \bot}{\Gamma \vdash p}\ \texttt{RAA} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash p}\ \texttt{FalseE}$$

$$\frac{\Gamma \vdash p}{\Gamma \vdash \forall x.p[x]}\ \texttt{ForallI} \qquad \frac{\Gamma \vdash \forall x.p[x]}{\Gamma \vdash p[t]}\ \texttt{ForallE}$$

The set of these inference rules is called **NDRules** and given as follows:

```
|- NDRules = {r | IdRule r \/ ConjI r \/ ConjE r \/ ImpI r \/
   ImpE r \/ RAA r \/ FalseE r \/ ForallI r \/ ForallE r}
```

Based on the exchange rules for the connectives $\neg$ and $\vee$, we derive the corresponding introduction and elimination rules:

$$\frac{\Gamma, p \vdash \bot}{\Gamma \vdash \neg p}\ \texttt{NotIntro} \qquad \frac{\Gamma \vdash p \qquad \Delta \vdash \neg p}{\Gamma, \Delta \vdash \bot}\ \texttt{NotElim}$$

$$\frac{\Gamma \vdash p}{\Gamma \vdash p \vee q} \quad \frac{\Gamma \vdash q}{\Gamma \vdash p \vee q}\ \texttt{OrIntro} \qquad \frac{\Gamma \vdash p \vee q \quad \Delta, p \vdash r \quad \Theta, q \vdash r}{\Gamma, \Delta, \Theta \vdash r}\ \texttt{OrElim}$$

3.2.1 The Substitution theorem for Natural Deduction

The equivalence connector $\Leftrightarrow$ is of special interest, because using equivalences we can make substitutions. Having a substitution rule of the form

$$\frac{\Gamma \vdash p \Leftrightarrow q \qquad \Delta \vdash \varphi}{\Gamma, \Delta \vdash \varphi[p\backslash q]}\ \texttt{SubstRule}$$

significantly facilitates derivations in natural deduction. For deriving this rule we prove the substitution theorem

```
SubstThm =
|- !hyps p q phi  phi_with_q.
   SubFormulaSubst phi p q phi_with_q ==>
      IsProvable NDRules hyps
         (Sseq {p Sequiv q} {phi Sequiv phi_with_q})
```

where `SubFormulaSubst` substitutes a subformula `q` for `p` in the formula `phi` to produce the formula `phi_with_q`. The proof of this theorem requires deriving a few intermediate rules, namely

$$\frac{}{\Gamma \vdash p \Leftrightarrow p}\ \texttt{EqId} \qquad \frac{\Gamma \vdash p \Leftrightarrow q}{\Gamma \vdash \neg p \Leftrightarrow \neg q}\ \texttt{EqNot}$$

and a number of rules of the form

$$\frac{\Gamma \vdash p1 \Leftrightarrow q1 \qquad \Delta \vdash p2 \Leftrightarrow q2}{\Gamma, \Delta \vdash p1\ *\ p2\ \Leftrightarrow\ q1\ *\ q2}\ \texttt{Eq}\ *$$

where $*$ is one of the connectives $\wedge, \vee, \Rightarrow$ and $\Leftrightarrow$.

Using the substitution rule, we can derive a number of useful connective exchange rules. The first rule allows us to introduce and eliminate double negation:

$$\frac{\Gamma \vdash \varphi[\neg\neg p]}{\Gamma \vdash \varphi[p]}\ \texttt{DblNegContr} \qquad \frac{\Gamma \vdash \varphi[p]}{\Gamma \vdash \varphi[\neg\neg p]}\ \texttt{DblNegExp}$$

The second rule handles exchanging the connectives $\vee$ and $\Rightarrow$:

$$\frac{\Gamma \vdash \varphi[\neg p \vee q]}{\Gamma \vdash \varphi[p \Rightarrow q]}\ \texttt{OrToImp} \qquad \frac{\Gamma \vdash \varphi[p \Rightarrow q]}{\Gamma \vdash \varphi[\neg p \vee q]}\ \texttt{ImpToOr}$$

For every derived connective exchange rule there is naturally a corresponding one with exchange in the antecedents of the sequents.

3.3 Sequent Calculus

Finally, let us present formalizations of the sequent calculus and its flattened version. The sequent calculus illustrates the symmetries of logic [3], and has numerous analogies with the system of natural deduction. The rules of the calculus are in fact more or less complex combinations of the rules of ND. We formalize sequent calculus without structural rules, resulting in the following set of inference rules:

$$\frac{}{\Gamma, p \vdash p, \Delta}\ \texttt{IdAx}$$

$$\frac{\Gamma \vdash p, \Delta \qquad \Phi \vdash q, \Theta}{\Gamma, \Phi \vdash p \wedge q, \Delta, \Theta}\ \texttt{ConjR} \qquad \frac{\Gamma, p \vdash \Delta}{\Gamma, p \wedge q \vdash \Delta} \quad \frac{\Gamma, q \vdash \Delta}{\Gamma, p \wedge q \vdash \Delta}\ \texttt{ConjL}$$

$$\frac{\Gamma \vdash p, \Delta}{\Gamma \vdash p \vee q, \Delta} \quad \frac{\Gamma \vdash q, \Delta}{\Gamma \vdash p \vee q, \Delta}\ \texttt{DisjR} \qquad \frac{\Gamma, p \vdash \Delta \qquad \Phi, q \vdash \Theta}{\Gamma, \Phi, p \vee q \vdash \Delta, \Theta}\ \texttt{DisjL}$$

$$\frac{\Gamma, p \vdash q, \Delta}{\Gamma \vdash p \Rightarrow q, \Delta}\ \texttt{ImpR} \qquad \frac{\Gamma \vdash p, \Delta \qquad \Phi, q \vdash \Theta}{\Gamma, \Phi, p \Rightarrow q \vdash \Delta, \Theta}\ \texttt{ImpL}$$

$$\frac{\Gamma, p \vdash \Delta}{\Gamma \vdash \neg p, \Delta}\ \texttt{NotR} \qquad \frac{\Gamma \vdash p, \Delta}{\Gamma, \neg p \vdash \Delta}\ \texttt{NotL}$$

$$\frac{\Gamma \vdash p, \Delta}{\Gamma \vdash \forall x.p[x], \Delta}\ \texttt{ForallR} \qquad \frac{\Gamma, p \vdash \Delta}{\Gamma, \forall x.p[x] \vdash \Delta}\ \texttt{ForallL}$$

$$\frac{\Gamma \vdash p, \Delta \qquad \Phi, p \vdash \Theta}{\Gamma, \Phi \vdash \Delta, \Theta}\ \texttt{Cut}$$

The set `SCRules` of these rules is defined as follows:

```
SCRules = {r | (IdAx r) \/ (ConjR r) \/ (ConjL r) \/
      (ImpR r) \/ (ImpL r) \/ (NotR r) \/ (NotL r) \/
      (ForallR r) \/ (ForallL r) \/ (Cut r)}
```

3.3.1 Flattening Sequents

Sequent calculus is the only axiomatization where succedents of sequents are not necessarily singleton sets. Sequents of SC are interpreted so that the conjunction of elements in the antecedent set asserts the disjunction of elements in the succedent set. When proving basic rules of sequent calculus as theorems of natural deduction, such sequents with non-singleton succedent sets create complications because the system of natural deduction cannot manipulate them using its basic rules. We resolve this problem by "flattening" sequents of sequent calculus. Namely, we map every sequent of SC to a corresponding one with a singleton succedent set whose only element is a disjunction of succedent elements of the original. Hence, a sequent $\Gamma \vdash \{\}$ is mapped to $\Gamma \vdash \{\bot\}$ and a sequent $\Gamma \vdash p, \Delta$ is mapped to $\Gamma \vdash \{p \vee \Delta'\}$, where Δ' is a disjunction of elements in Δ. By flattening `SCRules` in this manner, we define a proof system `FlatSCRules` representing flattened sequent calculus:

```
FlatSCRules = {r | (FlatIdAx r) \/ (FlatConjR r) \/ (FlatConjL r) \/
      (FlatImpR r) \/ (FlatImpL r) \/ (FlatNotR r) \/ (FlatNotL r) \/
      (FlatForallR r) \/ (FlatForallL r) \/ (FlatCut r)}
```

where, e.g., `FlatImpL` is given by

$$\frac{\Gamma \ \vdash \ p \vee \delta \qquad \Phi, q \ \vdash \ \theta}{\Gamma, \Phi, p \Rightarrow q \ \vdash \ \delta \vee \theta}\ \texttt{FlatImpL}$$

and show that it is a subset of the standard sequent calculus, in the sense that anything provable in FlatSC is also provable in SC. In HOL we formulate and prove the corresponding theorem

```
Provability_of_FlatSC_in_SC =
|- !h c. IsProvable FlatSCRules h c ==> IsProvable SCRules h c
```

The proof of this theorem is reduced, using first the inference rule set monotonicity theorem

```
RuleSetMono =
|- !rSet1 rSet2 h c.
   IsProvable rSet1 h c ==> IsProvable (rSet1 UNION rSet2) h c
```

and then the third rule in the set defining `IsProvable`, to a theorem stating that every basic inference rule of FlatSC can be derived in sequent calculus:

```
BasicFlatSCRules_Provable_in_SC =
|- !r. r IN FlatSCRules ==> (?inf. CorrectRuleInst {r} inf /\
   IsProvable SCRules (InfHyps inf) (InfConcl inf))
```

Rewriting with the definition of `FlatSCRules`, we get ten subgoals corresponding to the rules of flattened sequent calculus. For example, the subgoal corresponding to the rule `FlatImpL` states that

```
IsProvable SCRules
   {Sseq Gamma {p Sor delta}; Sseq (q INSERT Eta) {theta}}
   (Sseq ((p Simp q) INSERT (Gamma UNION Eta)) {delta Sor theta})
```

Due to the lack of space we present the complete derivations corresponding to this and the other `FlatSCRules` in the appendix.

3.4 Proving Isomorphism of Natural Deduction, Hilbertian Axiomatization, and Flattened Sequent Calculus

As it was already mentioned above, we prove the `IsomorphismThm` by transitivity of implication, showing that provability in ND implies provability in HA, and provability in HA implies provability in flattened sequent calculus, and, finally, that provability in FlatSC implies provability in natural deduction. Therefore, we need to prove three theorems:

```
Provability_of_ND_in_HA =
|- !h c. IsProvable NDRules h c ==> IsProvable HARules h c

Provability_of_HA_in_FlatSC =
|- !h c. IsProvable HARules h c ==> IsProvable FlatSCRules h c

Provability_of_FlatSC_in_ND =
|- !h c. IsProvable FlatSCRules h c ==> IsProvable NDRules h c
```

As was the case with provability of FlatSC in SC, proofs of these theorems reduce to showing that every basic inference rule of ND, HA, and FlatSC has a proof of its conclusion from its hypotheses in respectively HA, FlatSC, and ND. For example, the theorem stating provability of basic inference rules of natural deduction in Hilbertian axiomatization is formulated as follows:

```
BasicNDRules_Provable_in_HA =
|- !r. r IN NDRules ==> (?inf. CorrectRuleInst {r} inf /\
   IsProvable HARules (InfHyps inf) (InfConcl inf))
```

Proving this theorem (and the others are proved similarly) amounts to showing that for every basic inference rule of the system of natural deduction there exists a proof in Hilbertian axiomatization, using only the basic rules of HA and, possibly, the derived rules of HA corresponding to the basic rules of NA derived in HA earlier. Proving, for instance, correctness of derivation of the rule `RAA`, which can be derived in Hilbertian axiomatization, using the rule `ImpI` of natural deduction

$$\dfrac{\dfrac{\Gamma, p \Rightarrow \bot \;\vdash\; \bot}{\Gamma \;\vdash\; (p \Rightarrow \bot) \Rightarrow \bot}\,\texttt{ImpI} \qquad \dfrac{}{\vdash\; ((p \Rightarrow \bot) \Rightarrow \bot) \Rightarrow p}\,\texttt{Ax3}}{\Gamma \;\vdash\; p}\,\texttt{MP}$$

requires that correctness of derivation of `ImpI` in HA has already been proved.

We present all the derivations required for the proof of the theorems `BasicHARules_Provable_in_FlatSC`, `BasicFlatSCRules_Provable_in_ND` and `BasicNDRules_Provable_in_HA` in the appendix.

3.5 Translating Meta-logical Results Among the Proof Systems

The most important consequence of the theorems `IsomorphismThm` and `Provability_of_FlatSC_in_SC` is that the meta-logical properties we have proved about one proof system hold for the other systems as well. In particular, we have that the deduction, the monotonicity, and the substitution theorems hold for all the proof systems we study, including sequent calculus, whereas the compactness theorem holds, in addition to the Hilbertian axiomatization, for the system of natural deduction, and flattened sequent calculus.

For example, compactness in natural deduction is expressed as a theorem

```
CompactThmND =
|- !hyps Gamma p. IsProvable NDRules hyps (Sseq Gamma {p}) ==>
   (?Delta. Delta SUBSET Gamma /\ FINITE Delta /\
      IsProvable NDRules hyps (Sseq Delta {p}))
```

and proved in two steps by rewriting with `IsomorphismThm` and `CompactThm`.

4 Conclusions and Related Work

We formalize first-order logic in the theorem proving system HOL and establish isomorphism among three first-order proof systems, namely, the Hilbertian axiomatization, the system of natural deduction, and flattened sequent calculus. Moreover, we prove that classical sequent calculus is more general than these proof systems, in the sense that anything provable in either of them is also provable in SC.

Our interest in studying the relationship among first-order proof systems is by far not original. For example, Girard et al. [3] note that to a proof of $A \vdash B$ in sequent calculus there corresponds a deduction of B under the hypotheses A in natural deduction, and, conversely, a deduction of B from A can be represented in sequent calculus, but not uniquely. Kleene in [6] shows that provability in G4, which is a variant of sequent calculus, implies provability in a Hilbert-type formal system H. Also, he proves the deduction theorem for the system H; the latter differs from our proof system HA in taking all logical connectives as basic and, therefore, having three inference rules and 15 axiomatic rules. All the meta-logical reasoning in [3, 6] is carried out in a natural language, whereas we do it in a mechanized theorem-proving system using higher-order logic.

Formalization of mathematics, and, in particular, logic, in a theorem-proving system is another area of related work. In particular, the formalization of higher-order logic proofs in HOL [10] was a starting point for this project. Formalization of sequent calculus in the logical framework LF and a structural proof of Cut elimination are reported in [8]. Finally, Harrison in [5] gives a thorough account of approaches to formalization of logic in a historical perspective.

Acknowledgment

We would like to thank John Harrison for his encouragement to carry out this project, and practical help in formalization of concepts and proofs.

References

[1] Nicolas G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. Math.*, 34(5):381–392, 1972.

[2] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, Inc., Orlando, Florida, 1972.

[3] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1989.

[4] M. J. C. Gordon and Thomas F. Melham. *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press, 1993.

[5] John Harrison. Formalized mathematics. Technical Report TUCS-TR-36, Turku Centre for Computer Science, Finland, August 14, 1996.

[6] S. C. Kleene. *Mathematical Logic*. Wiley and Sons, 1968.

[7] N.D. Megill. Metamath: A computer language for pure mathematics. 1996. Unpublished; available on the Web from `ftp://sparky.shore.net/members/ndm/metamath.tex.Z`.

[8] Frank Pfenning. A structural proof of cut elimination and its representation in a logical framework. Technical Report CMU-CS-94-218, Department of Computer Science, Carnegie Mellon University, November 1994.

[9] Dirk van Dalen. *Logic and Structure*. Universitext. Springer-Verlag, Berlin, 3rd edition, 1994.

[10] J. von Wright. Representing higher order logic proofs in HOL. *Lecture Notes in Computer Science*, 859:456–469, 1994.

Appendix 1

Here we present derivations of the basic inference rules of flattened sequent calculus in ordinary sequent calculus. In these derivations we use auxiliary rules

$$\dfrac{\Gamma \vdash p \lor p}{\Gamma \vdash p}\ \texttt{DisjId} \qquad \dfrac{\Gamma \vdash p \lor q}{\Gamma \vdash q \lor p}\ \texttt{DisjComm} \qquad \dfrac{\Gamma \vdash p \lor (q \lor r)}{\Gamma \vdash (p \lor q) \lor r}\ \texttt{DisjAssoc}$$

as well as a pair of double negation contraction rules

$$\dfrac{\Gamma, \neg\neg p \vdash \Delta}{\Gamma, p \vdash \Delta} \qquad \dfrac{\Gamma \vdash \neg\neg p \lor q}{\Gamma \vdash p \lor q}\ \texttt{DblNegCtr}$$

derived is SC. The derivations of these rules are straightforward and we omit them for brevity.

The identity rule `FlatIdAx` is provable using the rules `IdAx` and `DisjR` as follows:

$$\dfrac{\dfrac{}{\Gamma, p \vdash p}\ \texttt{IdAx}}{\Gamma, p \vdash p \lor \delta}\ \texttt{DisjR}$$

Availability of `FlatCut` significantly simplifies further derivations, and so we construct its proof in SC next:

$$\dfrac{\Gamma \vdash p \lor \delta \qquad \dfrac{\dfrac{\Phi, p \vdash \theta}{\Phi, p \vdash \delta \lor \theta}\ \texttt{DisjR} \qquad \dfrac{}{\delta \vdash \delta \lor \theta}\ \texttt{FlatIdAx}}{\Phi, p \lor \delta \vdash \delta \lor \theta}\ \texttt{DisjL}}{\Gamma, \Phi \vdash \delta \lor \theta}\ \texttt{Cut}$$

Next we derive the disjunction rules. The right disjunction rule `FlatDisjR` is derived using `DisjR`:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash p \lor \delta}{\Gamma \vdash q \lor (p \lor \delta)}\ \texttt{DisjR}}{\Gamma \vdash (q \lor p) \lor \delta}\ \texttt{DisjAssoc}}{\Gamma \vdash (p \lor q) \lor \delta}\ \texttt{DisjComm} \qquad \dfrac{\dfrac{\Gamma \vdash q \lor \delta}{\Gamma \vdash p \lor (q \lor \delta)}\ \texttt{DisjR}}{\Gamma \vdash (p \lor q) \lor \delta}\ \texttt{DisjAssoc}$$

and the left disjunction rule `FlatDisjL` is derived using `DisjL`:

$$\dfrac{\dfrac{\Gamma, p \vdash \delta}{\Gamma, p \vdash \delta \lor \theta}\ \texttt{DisjR} \qquad \dfrac{\Phi, q \vdash \theta}{\Phi, q \vdash \delta \lor \theta}\ \texttt{DisjR}}{\Gamma, \Phi, p \lor q \vdash \delta \lor \theta}\ \texttt{DisjL}$$

The rule `NotR` is used to derive `FlatNotR`:

$$\dfrac{\dfrac{\dfrac{\Gamma, p \vdash \delta}{\Gamma, p \vdash \neg p \lor \delta}\ \texttt{DisjR}}{\Gamma \vdash \neg p, \neg p \lor \delta}\ \texttt{NotR} \qquad \dfrac{}{\neg p \vdash \neg p \lor \delta}\ \texttt{FlatIdAx}}{\Gamma \vdash \neg p \lor \delta}\ \texttt{Cut}$$

and the rule NotL is used to derive FlatNotL:

$$\dfrac{\dfrac{\Gamma \vdash p \vee \delta \qquad \dfrac{\dfrac{\dfrac{}{\neg p \vdash \neg p, \delta}\,\texttt{IdAx}}{\neg p, \neg\neg p \vdash \delta}\,\texttt{NotL}}{\neg p, p \vdash \delta}\,\texttt{DblNegCtr}}{\Gamma, \neg p \vdash \delta \vee \delta}\,\texttt{FlatCut}}{\Gamma, \neg p \vdash \delta}\,\texttt{DisjId}$$

The rules FlatNotR and FlatNotL can now be used in derivations of FlatImpR and FlatImpL:

$$\dfrac{\dfrac{\dfrac{\Gamma, p \vdash q \vee \delta}{\Gamma \vdash \neg p \vee (q \vee \delta)}\,\texttt{FlatNotR}}{\Gamma \vdash (\neg p \vee q) \vee \delta}\,\texttt{DisjAssoc}}{\Gamma \vdash (p \Rightarrow q) \vee \delta}\,\texttt{OrToImp} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash p \vee \delta}{\Gamma \vdash \delta \vee p}\,\texttt{DisjComm}}{\Gamma, \neg\delta \vdash p}\,\texttt{FlatNotL} \qquad \Phi, q \vdash \theta}{\Gamma, \Phi, p \Rightarrow q, \neg\delta \vdash \theta}\,\texttt{ImpL}}{\Gamma, \Phi, p \Rightarrow q \vdash \neg\neg\delta \vee \theta}\,\texttt{FlatNotR}}{\Gamma, \Phi, p \Rightarrow q \vdash \delta \vee \theta}\,\texttt{DblNegCtr}$$

The rules FlatConjL and FlatForallL are instances of the rules ConjL and ForallL, and the rules FlatConjR and FlatForallR are derived as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash p \vee \delta}{\Gamma \vdash \delta \vee p}\,\texttt{DisjComm}}{\Gamma, \neg\delta \vdash p}\,\texttt{FlatNotL} \qquad \dfrac{\dfrac{\Phi \vdash q \vee \theta}{\Phi \vdash \theta \vee q}\,\texttt{DisjComm}}{\Phi, \neg\theta \vdash q}\,\texttt{FlatNotL}}{\Gamma, \Phi, \neg\delta, \neg\theta \vdash p \wedge q}\,\texttt{ConjR}}{\Gamma, \Phi, \neg\delta \vdash \neg\neg\theta \vee p \wedge q}\,\texttt{FlatNotR}}{\Gamma, \Phi \vdash \neg\neg\delta \vee \neg\neg\theta \vee p \wedge q}\,\texttt{FlatNotR}}{\Gamma, \Phi \vdash \delta \vee \theta \vee p \wedge q}\,\texttt{DblNegCtr}}{\Gamma, \Phi \vdash (\delta \vee \theta) \vee p \wedge q}\,\texttt{DisjAssoc}}{\Gamma, \Phi \vdash p \wedge q \vee \delta \vee \theta}\,\texttt{DisjComm} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash p \vee \delta}{\Gamma \vdash \delta \vee p}\,\texttt{DisjComm}}{\Gamma, \neg\delta \vdash p}\,\texttt{FlatNotL}}{\Gamma, \neg\delta \vdash \forall x.p[x]}\,\texttt{ForallR}}{\Gamma \vdash \neg\neg\delta \vee \forall x.p[x]}\,\texttt{FlatNotR}}{\Gamma \vdash \delta \vee \forall x.p[x]}\,\texttt{DblNegCtr}}{\Gamma \vdash \forall x.p[x] \vee \delta}\,\texttt{DisjComm}$$

Appendix 2

Here we present derivations of the basic inference rules of natural deduction in Hilbertian axiomatization.

IdRule is a derived rule of HA trivially, since it is equivalent to IdentRule. So is ImpE rule since it is equivalent to the rule Modus Ponens of Hilbertian axiomatization. The rule ImpI is derived in HA using the Deduction Theorem.

The rule Reducium Ad Absurdum of natural deduction, `RAA`, is derived in HA using the derived rule `ImpI`:

$$\dfrac{\dfrac{\Gamma, p \Rightarrow \bot \ \vdash\ \bot}{\Gamma \ \vdash\ (p \Rightarrow \bot) \Rightarrow \bot}\ \texttt{ImpI} \qquad \dfrac{}{\vdash\ ((p \Rightarrow \bot) \Rightarrow \bot) \Rightarrow p}\ \texttt{Ax3}}{\Gamma \ \vdash\ p}\ \texttt{MP}$$

In the following derivations it is more convenient to use the following form of `RAA`:

$$\dfrac{\Gamma, \neg p \ \vdash\ \bot}{\Gamma \ \vdash\ p}\ \texttt{NegRAA}$$

We derive this rule using `RAA` and the connective exchange rule for negation. The rule `FalseE` is also derived using `RAA`:

$$\dfrac{\dfrac{\Gamma \ \vdash\ \bot}{\Gamma, p \Rightarrow \bot \ \vdash\ \bot}\ \texttt{MonoOne}}{\Gamma \ \vdash\ p}\ \texttt{RAA}$$

The conjunction introduction rule of natural deduction is derived in HA as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \ \vdash\ p \qquad \dfrac{}{p \Rightarrow (q \Rightarrow \bot) \ \vdash\ p \Rightarrow (q \Rightarrow \bot)}\ \texttt{IdentRule}}{\Gamma, p \Rightarrow (q \Rightarrow \bot) \ \vdash\ q \Rightarrow \bot}\ \texttt{MP} \qquad \Delta \ \vdash\ q}{\Gamma, \Delta, p \Rightarrow (q \Rightarrow \bot) \ \vdash\ \bot}\ \texttt{MP}}{\Gamma, \Delta \ \vdash\ (p \Rightarrow (q \Rightarrow \bot)) \Rightarrow \bot}\ \texttt{ImpI}}{\Gamma, \Delta \ \vdash\ p \wedge q}\ \texttt{AndContr}$$

Derivations of the conjunction elimination rules are rather intricate. To carry out these derivations we introduce a few new rules of Hilbertian axiomatization. The first rule represents transitivity of implication:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p \ \vdash\ p}\ \texttt{IdentRule} \qquad \Gamma \ \vdash\ p \Rightarrow q}{\Gamma, p \ \vdash\ q}\ \texttt{MP} \qquad \Delta \ \vdash\ q \Rightarrow r}{\Gamma, \Delta, p \ \vdash\ r}\ \texttt{MP}}{\Gamma, \Delta \ \vdash\ p \Rightarrow r}\ \texttt{ImpI}$$

We call this rule `ImpTran` and record it as a new derived rule of HA. The next two rules are variations of the contrapositive rule for implication:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, q \ \vdash\ q}\ \texttt{IdentRule} \qquad \dfrac{\dfrac{\dfrac{}{\neg p \ \vdash\ \neg p}\ \texttt{IdentRule} \qquad \dfrac{}{\neg p \ \Rightarrow\ \neg q \ \vdash\ \neg p \ \Rightarrow\ \neg q}\ \texttt{IdentRule}}{\neg p \ \Rightarrow\ \neg q, \neg p \ \vdash\ \neg q}\ \texttt{MP}}{\neg p \ \Rightarrow\ \neg q, \neg p \ \vdash\ q \Rightarrow \bot}\ \texttt{NotExp}}{\Gamma, \neg p \ \Rightarrow\ \neg q, \neg p, q \ \vdash\ \bot}\ \texttt{MP}}{\dfrac{\Gamma, \neg p \ \Rightarrow\ \neg q, q \ \vdash\ p}{\dfrac{\Gamma, \neg p \ \Rightarrow\ \neg q \ \vdash\ q \Rightarrow p}{\Gamma \ \vdash\ (\neg p \ \Rightarrow\ \neg q) \Rightarrow (q \Rightarrow p)}\ \texttt{ImpI}}\ \texttt{ImpI}}\ \texttt{NegRAA}$$

and

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg p \vdash \neg p}\;\texttt{IdentRule} \quad \dfrac{}{\neg p \Rightarrow q \vdash \neg p \Rightarrow q}\;\texttt{IdentRule}}{\neg p \Rightarrow q, \neg p \vdash q}\;\texttt{MP} \quad \dfrac{}{\Gamma, q \Rightarrow \bot \vdash q \Rightarrow \bot}\;\texttt{IdentRule}}{\Gamma, q \Rightarrow \bot, \neg p \Rightarrow q, \neg p \vdash \bot}\;\texttt{MP}}{\Gamma, q \Rightarrow \bot, \neg p \Rightarrow q \vdash p}\;\texttt{NegRAA}}{\Gamma, \neg p \Rightarrow q \vdash (q \Rightarrow \bot) \Rightarrow p}\;\texttt{ImpI}}{\Gamma, \neg p \Rightarrow q \vdash \neg q \Rightarrow p}\;\texttt{NotContr}}{\Gamma \vdash (\neg p \Rightarrow q) \Rightarrow (\neg q \Rightarrow p)}\;\texttt{ImpI}$$

We call these rules `ImpContr` and derive the first conjunction elimination rule using the second `ImpContr` rule as follows:

$$\dfrac{\dfrac{\Gamma \vdash p \wedge q}{\Gamma \vdash \neg(p \Rightarrow \neg q)}\;\texttt{AndExp} \quad \dfrac{\dfrac{}{\vdash \Psi}\;\texttt{Ax1} \quad \dfrac{}{\vdash \Psi \Rightarrow (\neg(p \Rightarrow \neg q) \Rightarrow q)}\;\texttt{ImpContr}}{\vdash \neg(p \Rightarrow \neg q)) \Rightarrow q}\;\texttt{MP}}{\Gamma \vdash q}\;\texttt{MP}$$

where $\Psi \;\widehat{=}\; \neg q \Rightarrow (p \Rightarrow \neg q)$. Derivation of the second conjunction elimination rule is based on a new axiomatic rule `ImpProp`,

$$\dfrac{\dfrac{}{\Gamma \vdash \neg p \Rightarrow (\neg q \Rightarrow \neg p)}\;\texttt{Ax1} \quad \dfrac{}{\vdash (\neg q \Rightarrow \neg p) \Rightarrow (p \Rightarrow q)}\;\texttt{ImpContr}}{\Gamma \vdash \neg p \Rightarrow (p \Rightarrow q)}\;\texttt{ImpTran}$$

and is as follows:

$$\dfrac{\dfrac{\Gamma \vdash p \wedge q}{\Gamma \vdash \neg(p \Rightarrow \neg q)}\;\texttt{AndExp} \quad \dfrac{\dfrac{}{\vdash \Psi}\;\texttt{ImpProp} \quad \dfrac{}{\vdash \Psi \Rightarrow (\neg(p \Rightarrow \neg q) \Rightarrow p)}\;\texttt{ImpContr}}{\vdash \neg(p \Rightarrow \neg q)) \Rightarrow p}\;\texttt{MP}}{\Gamma \vdash p}\;\texttt{MP}$$

where $\Psi \;\widehat{=}\; \neg p \Rightarrow (p \Rightarrow \neg q))$.

After proving the correctness of conjunction elimination derivations, the derivation of quantification rules is rewardingly simple. The rule `ForallI` is derived using `ForallIntro`:

$$\dfrac{\Gamma \vdash p \quad \dfrac{}{\vdash p \Rightarrow \forall x.p[x]}\;\texttt{ForallIntro}}{\Gamma \vdash \forall x.p[x]}\;\texttt{MP}$$

and the rule `ForallE` is derived using `ForallElim`:

$$\dfrac{\Gamma \vdash \forall x.p[x] \quad \dfrac{}{\vdash \forall x.p[x] \Rightarrow p[t]}\;\texttt{ForallElim}}{\Gamma \vdash p[t]}\;\texttt{MP}$$

Appendix 3

Here we prove that every basic inference rule of Hilbertian axiomatization is a derived rule of flattened sequent calculus. In the derivations we will use the rules `DisjId, DisjComm` and `DisjAssoc`, derived in FlatSC.

The identity rule of Hilbertian axiomatization `IdentRule` is derived in FlatSC using `FlatIdAx`:

$$\dfrac{\dfrac{}{\Gamma, p \;\vdash\; p \vee p}\,\texttt{FlatIdAx}}{\Gamma, p \;\vdash\; p}\,\texttt{DisjId}$$

The first axiomatic rule `Ax1` is derived as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, p \;\vdash\; p \vee \neg q}\,\texttt{FlatIdAx}}{\Gamma, p \;\vdash\; \neg q \vee p}\,\texttt{DisjComm}}{\Gamma \;\vdash\; \neg p \vee (\neg q \vee p)}\,\texttt{FlatNotR}}{\Gamma \;\vdash\; p \Rightarrow (q \Rightarrow p)}\,\texttt{OrToImp}$$

The second axiomatic rule `Ax2` is a little more complicated:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \neg p \;\vdash\; \neg p \vee r}\,\texttt{FlatIdAx} \qquad \dfrac{\dfrac{\dfrac{}{\neg p \vee q \;\vdash\; \neg p \vee q}\,\texttt{IdentRule}}{\neg p \vee q \;\vdash\; q \vee \neg p}\,\texttt{DisjComm} \qquad \dfrac{}{r \;\vdash\; r}\,\texttt{IdentRule}}{\neg p \vee q, q \Rightarrow r \;\vdash\; \neg p \vee r}\,\texttt{FlatImpL}}{\Gamma, \neg p \vee (q \Rightarrow r), (\neg p \vee q) \;\vdash\; (\neg p \vee r) \vee (\neg p \vee r)}\,\texttt{FlatDisjL}}{\Gamma, \neg p \vee (q \Rightarrow r), (\neg p \vee q) \;\vdash\; \neg p \vee r}\,\texttt{DisjId}}{\Gamma, \neg p \vee (q \Rightarrow r) \;\vdash\; \neg(\neg p \vee q) \vee (\neg p \vee r)}\,\texttt{FlatNotR}}{\Gamma \;\vdash\; \neg(\neg p \vee (q \Rightarrow r)) \vee (\neg(\neg p \vee q) \vee (\neg p \vee r))}\,\texttt{FlatNotR}}{\Gamma \;\vdash\; (p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))}\,\texttt{OrToImp}$$

The last axiomatic rule is derived as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, p \;\vdash\; p}\,\texttt{IdentRule}}{\Gamma \;\vdash\; \neg p \vee p}\,\texttt{FlatNotR}}{\Gamma \;\vdash\; p \Rightarrow p}\,\texttt{OrToImp}}{\Gamma \;\vdash\; \neg\neg p \Rightarrow p}\,\texttt{DblNegExp}}{\Gamma \;\vdash\; ((p \Rightarrow \bot) \Rightarrow \bot) \Rightarrow p}\,\texttt{NotExp}$$

In the derivation of Modus Ponens we use `FlatCut`:

$$\dfrac{\dfrac{\dfrac{\Gamma \;\vdash\; p}{\Gamma \;\vdash\; p \vee q}\,\texttt{FlatDisjR} \qquad \dfrac{\dfrac{\dfrac{\Delta \;\vdash\; p \Rightarrow q}{\Delta \;\vdash\; \neg p \vee q}\,\texttt{ImpToOr}}{\Delta, \neg\neg p \;\vdash\; q}\,\texttt{FlatNotL}}{\Delta, p \;\vdash\; q}\,\texttt{DblNegContr}}{\Gamma, \Delta \;\vdash\; q \vee q}\,\texttt{FlatCut}}{\Gamma, \Delta \;\vdash\; q}\,\texttt{DisjId}$$

Two new rules of flattened sequent calculus

$$\dfrac{\Gamma, p \;\vdash\; q}{\Gamma \;\vdash\; p \Rightarrow q}\,\texttt{ImpOneR} \quad \text{and} \quad \dfrac{\Gamma \;\vdash\; p}{\Gamma \;\vdash\; \forall x.p[x]}\,\texttt{ForallOneR}$$

are used to derive the quantification rules of Hilbertian axiomatization. The derivations of `ForallIntro` and `ForallElim` are straightforward, using these rules and `FlatForallL`. The distributivity rule is more complicated and we present its derivation below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p \;\vdash\; p}\,\texttt{IdentRule}}{\forall x.p \;\vdash\; p}\,\texttt{FlatForallL} \qquad \dfrac{\dfrac{}{\Gamma, p \Rightarrow q \;\vdash\; p \Rightarrow q}\,\texttt{IdentRule}}{\Gamma, \forall x.p \Rightarrow q \;\vdash\; p \Rightarrow q}\,\texttt{FlatForallL}}{\Gamma, \forall x.p \Rightarrow q, \forall x.p \;\vdash\; q}\,\texttt{MP}}{\Gamma, \forall x.p \Rightarrow q, \forall x.p \;\vdash\; \forall x.q}\,\texttt{ForallOneR}}{\Gamma, \forall x.p \Rightarrow q \;\vdash\; (\forall x.p) \Rightarrow (\forall x.q)}\,\texttt{ImpOneR}}{\Gamma \;\vdash\; (\forall x.p \Rightarrow q) \Rightarrow ((\forall x.p) \Rightarrow (\forall x.q))}\,\texttt{ImpOneR}$$

Appendix 4

Finally, we prove that every inference rule of flattened sequent calculus has a corresponding derivation in the system of natural deduction. We begin by proving the "identity group" `FlatIdAx` and FlatCut:

$$\dfrac{\dfrac{}{\Gamma, p \;\vdash\; p}\,\texttt{IdRule}}{\Gamma, p \;\vdash\; p \vee \delta}\,\texttt{OrIntro} \qquad \dfrac{\Gamma \;\vdash\; p \vee \delta \qquad \dfrac{\Phi, p \;\vdash\; \theta}{\Phi, p \;\vdash\; \delta \vee \theta}\,\texttt{OrIntro} \qquad \dfrac{\dfrac{}{\delta \;\vdash\; \delta}\,\texttt{IdRule}}{\delta \;\vdash\; \delta \vee \theta}\,\texttt{OrIntro}}{\Gamma, \Phi \;\vdash\; \delta \vee \theta}\,\texttt{OrElim}$$

We also derive a rule

$$\dfrac{\Gamma \;\vdash\; p \qquad \Phi, p \;\vdash\; \theta}{\Gamma, \Phi \;\vdash\; \theta}\,\texttt{CutOne}$$

which is useful for further derivations. The next two rules to be derived are `FlatNotR` and `FlatNotL` corresponding to negation introduction and elimination respectively:

$$\dfrac{\dfrac{\Gamma, p \;\vdash\; \delta}{\Gamma \;\vdash\; p \Rightarrow \delta}\,\texttt{ImpI}}{\Gamma \;\vdash\; \neg p \vee \delta}\,\texttt{ImpToOr} \qquad \dfrac{\dfrac{\dfrac{\Gamma \;\vdash\; p \vee \delta}{\Gamma \;\vdash\; \neg\neg p \vee \delta}\,\texttt{DblNegExp}}{\Gamma \;\vdash\; \neg p \Rightarrow \delta}\,\texttt{OrToImp} \qquad \dfrac{}{\neg p \;\vdash\; \neg p}\,\texttt{IdRule}}{\Gamma, \neg p \;\vdash\; \delta}\,\texttt{ImpE}$$

Before deriving the other rules of flattened sequent calculus in ND, we find it useful to introduce a symmetric pair of sign switching rules:

$$\dfrac{\Gamma, \neg p \;\vdash\; \delta}{\Gamma \;\vdash\; p \vee \delta}\,\texttt{SwSignR} \qquad \dfrac{\Gamma \;\vdash\; \neg p \vee \delta}{\Gamma, p \;\vdash\; \delta}\,\texttt{SwSignL}$$

Derivations of these rules are based on `FlatNotR` and `FlatNotL` respectively and on the rule for contracting double negation.

Next we derive the rules corresponding to implication introduction and elimination, `FlatImpR`:

$$\dfrac{\dfrac{\Gamma, p \;\vdash\; q \vee \delta}{\Gamma \;\vdash\; \neg p \vee q \vee \delta}\,\texttt{FlatNotR}}{\Gamma \;\vdash\; (p \Rightarrow q) \vee \delta}\,\texttt{OrToImp}$$

and `FlatImpL`:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash p \lor \delta}{\Gamma, \neg\delta \vdash p}\,\texttt{FlatNotL} \quad \dfrac{}{p \Rightarrow q \vdash p \Rightarrow q}\,\texttt{IdRule}}{\Gamma, \neg\delta, p \Rightarrow q \vdash q}\,\texttt{ImpE} \qquad \Phi, q \vdash \theta}{\Gamma, \neg\delta, p \Rightarrow q, \Phi \vdash \theta}\,\texttt{CutOne}}{\Gamma, \Phi, p \Rightarrow q \vdash \delta \lor \theta}\,\texttt{SwSignR}$$

The sign switching rules are also employed in derivations of `FlatConjR`:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash p \lor \delta}{\Gamma, \neg\delta \vdash p}\,\texttt{FlatNotL} \quad \dfrac{\Phi \vdash q \lor \theta}{\Phi, \neg\theta \vdash q}\,\texttt{FlatNotL}}{\Gamma, \Phi, \neg\theta, \neg\delta \vdash p \land q}\,\texttt{ConjI}}{\Gamma, \Phi, \neg\theta \vdash p \land q \lor \delta}\,\texttt{SwSignR}}{\Gamma, \Phi \vdash p \land q \lor \delta \lor \theta}\,\texttt{SwSignR}$$

and `FlatConjL`:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, p \vdash \delta}{\Gamma \vdash \neg p \lor \delta}\,\texttt{FlatNotR}}{\Gamma, \neg\delta \vdash \neg p}\,\texttt{SwSignL} \quad \dfrac{\dfrac{}{p \land q \vdash p \land q}\,\texttt{IdRule}}{p \land q \vdash p}\,\texttt{ConjE}}{\Gamma, p \land q, \neg\delta \vdash \bot}\,\texttt{NotElim}}{\Gamma, p \land q \vdash \delta}\,\texttt{RAA}$$

The right rule for disjunction of FlatSC is essentially the same as the disjunction introduction rule of ND:

$$\dfrac{\Gamma \vdash p \lor q \lor \delta}{\Gamma \vdash p \lor \delta}\,\texttt{OrIntro} \qquad \dfrac{\Gamma \vdash p \lor q \lor \delta}{\Gamma \vdash q \lor \delta}\,\texttt{OrIntro}$$

The left rule for disjunction is only slightly more involved and is based on the corresponding disjunction elimination rule of ND:

$$\dfrac{\dfrac{}{p \lor q \vdash p \lor q}\,\texttt{IdRule} \quad \dfrac{\Gamma, p \vdash \delta}{\Gamma, p \vdash \delta \lor \theta}\,\texttt{OrIntro} \quad \dfrac{\Phi, q \vdash \theta}{\Phi, q \vdash \delta \lor \theta}\,\texttt{OrIntro}}{\Gamma, \Phi, p \lor q \vdash \delta \lor \theta}\,\texttt{OrElim}$$

Finally, the right rule for universal quantification is derived using the universal quantification introduction rule of natural deduction:

$$\dfrac{\Gamma \vdash p \lor \delta \quad \dfrac{\dfrac{\dfrac{}{p \vdash p}\,\texttt{IdRule}}{p \vdash \forall x.p[x]}\,\texttt{ForallI}}{p \vdash \forall x.p[x] \lor \delta}\,\texttt{OrIntro} \quad \dfrac{\dfrac{}{\delta \vdash \delta}\,\texttt{IdRule}}{\delta \vdash \forall x.p[x] \lor \delta}\,\texttt{OrIntro}}{\Gamma \vdash \forall x.p[x] \lor \delta}\,\texttt{OrElim}$$

and the left rule for universal quantification is derived in ND using `ForallE`:

$$\dfrac{\dfrac{\overline{\forall x.p[x] \;\vdash\; \forall x.p[x]}\;\texttt{IdRule}}{\forall x.p[x] \;\vdash\; p}\;\texttt{ForallE} \qquad \Gamma, p \;\vdash\; \delta}{\Gamma, \forall x.p[x] \;\vdash\; \delta}\;\texttt{CutOne}$$

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**