The Kolmogorov Interpretation of Classical Logic in Intuitionistic Logic

Chad E. Brown

November 30, 1998

1 Introduction

We indicate how to represent the Kolmogorov interpretation of classical first-order logic in intuitionistic first-order logic using the logical framework LF (see [Pfe]). We also implement this representation in the Twelf programming language (see [PS98]) Furthermore, the proofs of two metatheorems (soundness and completeness of the interpretation) are represented as higher-level judgments. These proofs are also implemented in Twelf.

In 1925, Kolmogorov showed how to interpret classical logic in intuitionistic logic using a double negation translation of formulas (see [Kol67]). Kolmogorov worked with a Hilbert-style axiomatization of classical logic and intuitionistic logic (using only implication and negation for the propositional case). Actually, Kolmogorov's axiomatization of intuitionistic logic gives only the minimal calculus. In this report, we work with Gentzen-style natural deduction formulations of classical logic and intuitionistic logic (see [Gen69]).

2 Representing Classical and Intuitionistic First-Order Logic

The language of first-order logic is partitioned into terms and formulas. Terms are constructed by induction from variables, parameters, and constant symbols using function symbols in the usual way. An atomic formula is either one of the propositional constants $\top$ and $\bot$, a propositional parameter

p, or is of the form $P(t_1, \ldots, t_n)$ for a predicate symbol P and an n-tuple of terms $(t_1, \ldots, t_n)$. Formulas are constructed by induction from atomic formulas using the propositional connectives $\wedge$, $\supset$, $\vee$, $\neg$, and the quantifiers $\forall$ and $\exists$. Note that the constant symbols, function symbols, and predicate symbols available depend upon the particular signature. The basic Twelf implementation will assume an empty signature. It is straightforward to include new symbols in the signature.

The type of terms in the LF representation is i and the type of formulas is o. The language of first-order logic is represented in LF by including constants for propositional connectives and constants (e.g., and : $\mathrm{o} \to \mathrm{o} \to \mathrm{o}$, true : o) and constants for the quantifiers (e.g., forall : $(\mathrm{i} \to \mathrm{o}) \to \mathrm{o}$). We use the principle of abstract syntax to represent object variables as meta-variables, and object-level bindings as meta-level (lambda) bindings. In accordance with this principle, the constants forall and exists are applied to *functions* which take terms to formulas (i.e., objects of type $\mathrm{i} \to \mathrm{o}$) and return formulas.

We will not describe the full representation function $\ulcorner\urcorner$ taking terms and formulas to their LF representation (see [Pfe] for more details). Such a function can be defined by a straightforward induction. The only interesting cases are the quantifiers.

$$\begin{array}{rcl} \lceil \forall x.\ A \rceil & = & \mathrm{forall}\ (\lambda x{:}\mathrm{i}.\ \lceil A \rceil) \\ \lceil \exists x.\ A \rceil & = & \mathrm{exists}\ (\lambda x{:}\mathrm{i}.\ \lceil A \rceil) \end{array}$$

The Twelf implementation of first-order logic is given below.

```
i : type.  %name i T.
o : type.  %name o A.

and    : o -> o -> o.   %infix right 11 and.
imp    : o -> o -> o.   %infix right 10 imp.
or     : o -> o -> o.   %infix right 11 or.
not    : o -> o.        %prefix 12 not.
true   : o.
false  : o.
forall : (i -> o) -> o.
exists : (i -> o) -> o.
```

Note that for the sake of readability we have declared `and`, `imp`, and `or` to be infix operators.

For both classical and intuitionistic natural deduction, we need a judgment of the derivability of $\vdash C$ from hypotheses $\vdash A_1, \ldots, \vdash A_n$. To distinguish the two judgments, we will use the symbols $\vdash_{\mathrm{NJ}}$ and $\vdash_{\mathrm{NK}}$. The symbol $\vdash$ may be used ambiguously to stand for either.

Each inference rule for intuitionistic natural deduction has a counterpart for classical natural deduction. For example, the conjunction introduction rule for intuitionistic logic

$$\frac{\vdash_{\mathrm{NJ}} A \qquad\qquad \vdash_{\mathrm{NJ}} B}{\vdash_{\mathrm{NJ}} A \wedge B} \wedge\mathrm{I}$$

has a classical counterpart

$$\frac{\vdash_{\mathrm{NK}} A \qquad\qquad \vdash_{\mathrm{NK}} B}{\vdash_{\mathrm{NK}} A \wedge B} \wedge\mathrm{I}.$$

(Note that we label both of these rules $\wedge$I.) Writing the rule as

$$\frac{\vdash A \qquad\qquad \vdash B}{\vdash A \wedge B} \wedge\mathrm{I}$$

is meant to represent both. Using this convention, we can simultaneously introduce all the inference rules of intuitionistic natural deduction and all but one inference rule of classical natural deduction. These rules are given below separated into introduction and elimination rules. These diagrams are taken from [Pfe], which can be consulted for futher details.

Introduction Rules | Elimination Rules

$$\frac{\vdash A \qquad \vdash B}{\vdash A \wedge B}\,\wedge\mathrm{I}$$

$$\frac{\vdash A \wedge B}{\vdash A}\,\wedge\mathrm{E_L} \qquad \frac{A \wedge B}{B}\,\wedge\mathrm{E_R}$$

$$\frac{\begin{array}{c}\overline{\vdash A}^{\,u} \\ \vdots \\ \vdash B\end{array}}{\vdash A \supset B}\,\supset\mathrm{I}^u$$

$$\frac{\vdash A \supset B \qquad \vdash A}{\vdash B}\,\supset\mathrm{E}$$

$$\frac{\vdash A}{\vdash A \vee B}\,\vee\mathrm{I_L} \qquad \frac{\vdash B}{\vdash A \vee B}\,\vee\mathrm{I_R}$$

$$\frac{\vdash A \vee B \qquad \begin{array}{c}\overline{\vdash A}^{\,u_1} \\ \vdots \\ \vdash C\end{array} \qquad \begin{array}{c}\overline{\vdash B}^{\,u_2} \\ \vdots \\ \vdash C\end{array}}{\vdash C}\,\vee\mathrm{E}^{u_1,u_2}$$

$$\frac{\begin{array}{c}\overline{\vdash A}^{\,u} \\ \vdots \\ \vdash p\end{array}}{\vdash \neg A}\,\neg\mathrm{I}^{p,u}$$

$$\frac{\vdash A \qquad \vdash \neg A}{\vdash C}\,\neg\mathrm{E}$$

$$\frac{}{\vdash \top}\,\top\mathrm{I}$$

$$\frac{\vdash \bot}{\vdash C}\,\bot\mathrm{E}$$

Introduction Rules | Elimination Rules

$$\frac{\vdash [a/x]A}{\vdash \forall x.\ A}\,\forall\mathrm{I}^* \qquad\qquad \frac{\vdash \forall x.\ A}{\vdash [t/x]A}\,\forall\mathrm{E}$$

$$\frac{\vdash [t/x]A}{\vdash \exists x.\ A}\,\exists\mathrm{I} \qquad\qquad \frac{\vdash \exists x.\ A \qquad \begin{array}{c}\overline{\vdash [a/x]A}\,u\\ \vdots\\ \vdash C\end{array}}{\vdash C}\,\exists\mathrm{E}^{a,u}$$

In the case of classical deduction, we further have the double negation reduction rule (which is equivalent to excluded middle).

$$\frac{\vdash_{\mathrm{NK}} \neg\neg A}{\vdash_{\mathrm{NK}} A}\,\neg\neg\mathrm{R}$$

These judgments are represented in LF by dependent types. We have separate types for classical provability and intuitionistic provability. The two judgments are defined similarly, with the only exception that classical provability includes the double negation reduction rule. The two dependent types in LF are

nk : o $\rightarrow$ type.
nj : o $\rightarrow$ type.

where a term of type nk A represents a classical natural deduction proof of A and a term of type nj A represents an intuitionistic natural deduction proof of A. The corresponding types in the Twelf implementation are

```
nk : o -> type.
nj : o -> type.
```

Each inference rule is represented by a constant. Many of the inference rules are straightforward. For example,

$$\frac{\vdash_{\mathrm{NJ}} A \qquad \vdash_{\mathrm{NJ}} B}{\vdash_{\mathrm{NJ}} A \wedge B}\,\wedge\mathrm{I}$$

is represented by a constant

$$\text{nj\_andi} : \Pi A{:}\text{o}.\ \Pi B{:}\text{o}.\ \text{nj}\ A \rightarrow \text{nj}\ B \rightarrow \text{nj}\ (A\ \text{and}\ B).$$

In the Twelf implementation, we can leave many of the Π bindings implicit. The Twelf implementation of the LF constant above is

```
nj_andi    : nj A -> nj B -> nj (A and B).
```

Other rules employ hypothetical judgments. A typical case is

$$\dfrac{\begin{array}{c}\overline{\vdash A}^{\,u} \\ \vdots \\ \vdash B\end{array}}{\vdash A \supset B} \supset\mathrm{I}^u$$

which is represented by a constant

$$\text{nj\_impi} : \Pi A{:}\text{o}.\ \Pi B{:}\text{o}.\ (\text{nj}\ A \rightarrow \text{nj}\ B) \rightarrow \text{nj}\ (A\ \text{imp}\ B)$$

and implemented in Twelf by

```
nj_impi    : (nj A -> nj B) -> nj (A imp B).
```

Other rules require parametric judgments. A typical case is

$$\dfrac{\vdash [a/x]A}{\vdash \forall x.\ A} \forall\mathrm{I}^*$$

represented by a constant

$$\text{nj\_foralli} : \Pi A{:}(\text{i} \rightarrow \text{o}).\ (\Pi a{:}\text{i}.\ \text{nj}\ (A\ a)) \rightarrow \text{nj}\ (\text{forall}\ A)$$

and implemented in Twelf by

```
nj_foralli : ({a:i} nj (A a)) -> nj (forall A).
```

Finally, some rules require judgments which are both hypothetical and parametric. Typical cases include

$$\dfrac{\begin{array}{c}\overline{\vdash A}^{\,u}\\ \vdots \\ \vdash p\end{array}}{\vdash \neg A}\,\neg\mathrm{I}^{p,u} \qquad\qquad \dfrac{\vdash \exists x.\ A \qquad \begin{array}{c}\overline{\vdash [a/x]A}^{\,u}\\ \vdots \\ \vdash C\end{array}}{\vdash C}\,\exists\mathrm{E}^{a,u}$$

represented by constants

$$\begin{array}{lcl} \mathrm{nj\_noti} & : & \Pi A{:}\mathrm{o}.\ (\Pi p{:}\mathrm{o}.\ \mathrm{nj}\ A \to \mathrm{nj}\ p) \to \mathrm{nj}\ (\mathrm{not}\ A) \\ \mathrm{nj\_existse} & : & \Pi A{:}\mathrm{o}.\ \Pi C{:}\mathrm{o}.\ \mathrm{nj}\ (\mathrm{exists}\ A) \to (\Pi a{:}\mathrm{i}.\ \mathrm{nj}\ (A\ a) \to \mathrm{nj}\ C) \to \mathrm{nj}\ C \end{array}$$

and implemented in Twelf by

```
nj_noti    : ({p:o} nj A -> nj p) -> nj (not A).
nj_existse : nj (exists A) -> ({a:i} nj (A a) -> nj C) -> nj C.
```

The full Twelf implementations are shown in the appendix.

3 Derived Rules of Inference

In both classical and intuitionistic logic, we can derive $\vdash \neg\neg A$ given a derivation of $\vdash A$. The hypothetical derivation which allows this inference is shown below.

$$\dfrac{\dfrac{\overline{\vdash \neg A}^{\,u} \qquad \overline{\vdash A}}{\vdash p}\,\neg\mathrm{E}}{\vdash \neg\neg A}\,\neg\mathrm{I}^{p,u}$$

Since there is such a hypothetical derivation (for both $\vdash_{\mathrm{NK}}$ and $\vdash_{\mathrm{NJ}}$), we have the following (double negation expansion) derived rules of inference.

$$\dfrac{\vdash_{\mathrm{NK}} A}{\vdash_{\mathrm{NK}} \neg\neg A}\,\neg\neg\mathrm{X} \qquad\qquad \dfrac{\vdash_{\mathrm{NJ}} A}{\vdash_{\mathrm{NJ}} \neg\neg A}\,\neg\neg\mathrm{X}$$

In Twelf, these derived rules of inference are implemented as the following definitions.

```
nk_dnotx = ([NK] (nk_noti [p:o] [u:nk (not A)] (nk_note u p NK))).
nj_dnotx = ([NJ] (nj_noti [p:o] [u:nj (not A)] (nj_note u p NJ))).
```

Derived rules of inference should be contrasted with admissible rules of inference. An admissible rule of inference is shown to be valid by induction over the rules of inference. As a result, adding a new rule of inference can invalidate what was previously an admissible rule. A derived rule of inference will be valid given any extension of the rules of inference.

We will also need two other derived rules of inference in intuitionistic logic. The first allows us to reduce a triple negation to a single negation. The second allows us to reduce a double negation of false to prove anything. The derived rules of inference are shown below.

$$\frac{\vdash_{\mathrm{NJ}} \neg\neg\neg A}{\vdash_{\mathrm{NJ}} \neg A}\,\neg\neg\neg\mathrm{R} \qquad\qquad \frac{\vdash_{\mathrm{NJ}} \neg\neg\bot}{\vdash_{\mathrm{NJ}} C}\,\neg\neg\bot\mathrm{R}$$

The derivation of $\neg\neg\neg$R is as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_{\mathrm{NJ}} A}\,u}{\vdash_{\mathrm{NJ}} \neg\neg A}\,\neg\neg\mathrm{X} \quad \dfrac{}{\vdash_{\mathrm{NJ}} \neg\neg\neg A}}{\vdash_{\mathrm{NJ}} q}\,\neg\mathrm{E}}{\vdash_{\mathrm{NJ}} \neg A}\,\neg\mathrm{I}^{q,u}$$

The derivation of $\neg\neg\bot$R is as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_{\mathrm{NJ}} \bot}\,u}{\vdash_{\mathrm{NJ}} p}\,\bot\mathrm{E}}{\vdash_{\mathrm{NJ}} \neg\bot}\,\neg\mathrm{I}^{p,u} \quad \dfrac{}{\vdash_{\mathrm{NJ}} \neg\neg\bot}}{\vdash_{\mathrm{NJ}} C}\,\neg\mathrm{E}$$

The Twelf definitions are shown in the appendix.

4 The Kolmogorov Translation

Given a formula, we form its Kolmogorov translation by recursively applying double negation to each subformula. We will follow Kolmogorov's original notation by using A^* to represent the translation of A. We will also follow Kolmogorov's notation by using nA as shorthand for $\neg\neg A$. The inductive definition of the translation is given by

$$\begin{array}{rcll}
A^* & = & nA & \text{if } A \text{ is atomic} \\
(A \wedge B)^* & = & n(A^* \wedge B^*) & \\
(A \supset B)^* & = & n(A^* \supset B^*) & \\
(A \vee B)^* & = & n(A^* \vee B^*) & \\
(\neg A)^* & = & n(\neg A^*) & \\
\top^* & = & n\top & \\
\bot^* & = & n\bot & \\
(\forall x.A)^* & = & n\forall x.A^* & \\
(\exists x.A)^* & = & n\exists x.A^* &
\end{array}$$

Although this translation is defined above as a function from formulas to formulas, in LF we must represent it as a relation. In particular, we have a dependent type

$$\text{kolm} : \text{o} \to \text{o} \to \text{type}.$$

The judgment kolm A A^* means that A^* is the Kolmogorov translation of A. Two typical LF constants defining the judgment are

$$\begin{array}{lcl}
\text{kolm_not} & : & \Pi A{:}\text{o}.\ \Pi A^*{:}\text{o}.\ \text{kolm } A\ A^* \to \text{kolm (not } A\text{) (not not not } A^*\text{)} \\
\text{kolm_forall} & : & \Pi A{:}(\text{i} \to \text{o}).\ \Pi A^*{:}(\text{i} \to \text{o}). \\
& & \qquad (\Pi a{:}\text{i}.\ \text{kolm } (A\ a)\ (A^*\ a)) \\
& & \qquad\qquad \to \text{kolm (forall } A\text{) (not not (forall } A^*\text{))}.
\end{array}$$

Before we implement this in Twelf, we make the definition

```
n = [p:o] (not not p).
```

This simplifies the syntax by allowing us to write `n A` for `not not A`. The Twelf constant declarations corresponding to the two cases shown above are

```
kolm_not    : kolm (not A) (n (not A*))
    <- kolm A A*.
kolm_forall : kolm (forall A) (n (forall A*))
    <- ({a:i} kolm (A a) (A* a)).
```

The full Twelf implementation is shown in the appendix.

Since the Kolmogorov translation is a *function* from formulas to formulas, we know that for every A there is a translation A^*. Unfortunately, the LF representation (and Twelf implementation) of the translation as a relation fails to capture this fact. Since we need to know arbitrary formulas have Kolmogorov translations to prove the soundness result (in the next section), we must explicitly represent the functional character of `kolm` by defining a judgment that there exists a Kolmogorov translation. We skip directly to the Twelf implementation. First, we have a dependent type

```
existskolm : {A:o} {A*:o} kolm A A* -> type.
```

Then we define constants which mimic the cases in the definition of `kolm`. For example, we have constants

```
existskolm_not    : existskolm (not A) (n not A*)
                               (kolm_not KA)
     <- existskolm A A* KA.

existskolm_forall : existskolm (forall A) (n (forall A*))
                               (kolm_forall KA)
     <- ({a:i} existskolm (A a) (A* a) (KA a)).
```

Using mode and termination checking in Twelf (see [PS98]), we know that given any formula A, the question of the existence of a Kolmogorov translation A^* of A is decided by `existskolm`. We further have the coverage property that for every A there is an A^*, but this has not been verified automatically.

Another fact we will need in the proof of soundness is that every A^* is of the form nA^+ for some A^+. This is true by an appeal to the inversion principle based on the definition of `kolm`.

It remains to show that A is provable in classical logic iff its translation A^* is provable in intuitionistic logic. The "only if" direction is soundness of the interpretation. The "if" direction is completeness. These results are proven in the following sections, along with indications of how to implement the proofs in Twelf.

5 Soundness

Soundness of the interpretation means that whenever a formula A is classically provable, its translation A^* is intuitionistically provable. The proof of this fact is by induction over the derivation of $\vdash_{\mathrm{NK}} A$, but with a generalized induction hypothesis. We can separate this argument as a lemma.

Notation: Since the diagrammatic notation can be cumbersome, we localize parameters and hypotheses by introducing the notation

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NK}} H_1, \ldots, \vdash_{\mathrm{NK}} H_k \triangleright NK :: \vdash_{\mathrm{NK}} A$$

to mean NK is a derivation of $\vdash_{\mathrm{NK}} A$ depending on parameters $a_1, \ldots, a_n, p_1, \ldots, p_m$ and hypotheses $\vdash_{\mathrm{NK}} H_1, \ldots, \vdash_{\mathrm{NK}} H_k$.

Lemma 5.1 *Given formulas $H_1, \ldots, H_k, A$, individual parameters $a_1, \ldots, a_n$, propositional parameters $p_1, \ldots, p_m$, and a derivation NK so that*

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NK}} H_1, \ldots, \vdash_{\mathrm{NK}} H_k \triangleright NK :: \vdash_{\mathrm{NK}} A,$$

there is a derivation NJ so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ :: \vdash_{\mathrm{NJ}} A^*.$$

Proof: The proof is by induction over NK. The proof is represented in LF and implemented in Twelf using a higher level judgement of type

```
sound : nk A -> kolm A A* -> nj A* -> type.
```

Each case in the induction is represented and implemented by a constant.

We only show a few representative cases. The introduction rule cases are generally straightforward, requiring induction and the derived rule of inference $\neg\neg$X.

Case : This is a typical introduction case. Suppose NK is of the form

$$\dfrac{\begin{array}{c}\overline{\vdash_{\mathrm{NK}} A}\,u \\ NK_B \\ \vdash_{\mathrm{NK}} B\end{array}}{\vdash_{\mathrm{NK}} A \supset B}\supset\mathrm{I}^u$$

Then, by induction we have a hypothetical derivation NJ_B satisfying

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^*, \vdash_{\mathrm{NJ}} A^* \triangleright NJ_B :: \vdash_{\mathrm{NJ}} B^*.$$

Recall that $(A \supset B)^* = n(A^* \supset B^*)$. Using NJ_B, we can construct NJ so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ :: \vdash_{\mathrm{NJ}} n(A^* \supset B^*)$$

as shown below. (Note the use of the derived rule of inference $\neg\neg$X.)

$$\dfrac{\dfrac{\begin{array}{c}\overline{\vdash_{\mathrm{NJ}} A^*}\,v \\ NJ_B \\ \vdash_{\mathrm{NJ}} B^*\end{array}}{\vdash_{\mathrm{NJ}} A^* \supset B^*}\supset\mathrm{I}^v}{\vdash_{\mathrm{NJ}} n(A^* \supset B^*)}\neg\neg\mathrm{X}$$

This case is implemented in Twelf by

```
sound_impi : sound (nk_impi NKB) (kolm_imp KB KA)
                   (nj_dnotx (nj_impi NJB))
    <- ({u:nk A} {v:nj A*} sound u KA v ->
          sound (NKB u) KB (NJB v)).
```

Case : Suppose NK is of the form

$$\dfrac{\begin{array}{c}\overline{\vdash_{\mathrm{NK}} A}\,u \\ NK_1 \\ \vdash_{\mathrm{NK}} p\end{array}}{\vdash_{\mathrm{NK}} \neg A}\neg\mathrm{I}^{p,u}$$

By induction we have a derivation NJ_1 hypothetical in $\vdash_{\mathrm{NJ}} A^*$ and parametric in p so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, p, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^*, \vdash_{\mathrm{NJ}} A^* \triangleright NJ_1 :: \vdash_{\mathrm{NJ}} np$$

Instantiating the parameter p with $\bot$, we obtain a derivation $NJ_1^{\bot}$ satisfying

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^*, \vdash_{\mathrm{NJ}} A^* \triangleright NJ_1^{\bot} :: \vdash_{\mathrm{NJ}} n\bot$$

Recall that $(\neg A)^* = n(\neg A^*)$. Using the derived rules of inference $\neg\neg\bot$R and $\neg\neg$X we can construct NJ so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ :: \vdash_{\mathrm{NJ}} n\neg A^*$$

as shown below.

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\dfrac{}{\vdash_{\mathrm{NJ}} A^*}\,v \\ NJ_1^{\bot} \\ \vdash_{\mathrm{NJ}} n\bot\end{array}}{q}\,\neg\neg\bot\mathrm{R}}{\vdash_{\mathrm{NJ}} \neg A^*}\,\neg\mathrm{I}^{v,q}}{\vdash_{\mathrm{NJ}} n\neg A^*}\,\neg\neg\mathrm{X}$$

This case is implemented in Twelf by

```
sound_noti : sound (nk_noti NK1) (kolm_not K)
                   (nj_dnotx
                     (nj_noti ([q] [v:nj A*]
                       (nj_dneg_falser (NJ1 false v) q))))
    <- ({p} {u:nk A} {v:nj A*} {kp:kolm p (n p)}
        existskolm p (n p) kp ->
        sound u K v ->
         sound (NK1 p u) kp (NJ1 p v)).
```

The elimination rule cases generally require more work. We will show one typical case and two cases which are rather special. Notice that in the Twelf implementation of the elimination rule cases, we must appeal to the `existskolm` lemma to know that the formulas in the induction step have Kolmogorov translations. (In the introduction rule cases we can obtain the Kolmogorov translations by inversion.)

Case : This is a typical case for elimination rules. Suppose we have NK is of the form

$$\dfrac{\begin{array}{c} NK_1 \\ \vdash_{\mathrm{NK}} A \wedge B \end{array}}{\vdash_{\mathrm{NK}} A} \wedge\mathrm{E_L}$$

By inversion we have $A^* = nA^+$ for some A^+. So,

$$(A \wedge B)^* = n(A^* \wedge B^*) = n(nA^+ \wedge B^*).$$

By induction on NK_1 we have a derivation NJ_1 so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ_1 :: \vdash_{\mathrm{NJ}} n(nA^+ \wedge B^*).$$

Given NJ_1, we can construct a derivation NJ satisfying

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ :: \vdash_{\mathrm{NJ}} nA^+$$

as shown below.

$$\dfrac{\dfrac{\begin{array}{c} (*) \\ \vdash_{\mathrm{NJ}} \neg(nA^+ \wedge B^*) \end{array} \qquad \begin{array}{c} NJ_1 \\ \vdash_{\mathrm{NJ}} \neg\neg(nA^+ \wedge B^*) \end{array}}{\vdash_{\mathrm{NJ}} q} \neg\mathrm{E}}{\vdash_{\mathrm{NJ}} nA^+} \neg\mathrm{I}^{q,u}$$

where $(*)$ is the derivation

$$\dfrac{\dfrac{\dfrac{}{\vdash_{\mathrm{NJ}} \neg A^+} u \qquad \dfrac{\dfrac{}{\vdash_{\mathrm{NJ}} nA^+ \wedge B^*} v}{\vdash_{\mathrm{NJ}} \neg\neg A^+} \wedge\mathrm{E_L}}{\vdash_{\mathrm{NJ}} p} \neg\mathrm{E}}{\vdash_{\mathrm{NJ}} \neg(nA^+ \wedge B^*)} \neg\mathrm{I}^{p,v}$$

The Twelf implementation of this case is given by

```
sound_andel : sound (nk_andel (NK1:nk (A and B))) KA
                    (nj_noti ([q:o] [u:nj (not A+)]
                      (nj_note NJ1 q
                        (nj_noti ([p:o] [v:nj ((n A+) and B*)]
                          (nj_note (nj_andel v) p u))))))
    <- existskolm B B* KB
    <- sound NK1 (kolm_and KB KA) NJ1.  % NJ1::- - (- - A+ and B*)
```

Case : Suppose we have NK is of the form

$$\dfrac{\begin{matrix} NK_1 \\ \vdash_{\mathrm{NK}} A \end{matrix} \qquad \begin{matrix} NK_2 \\ \vdash_{\mathrm{NK}} \neg A \end{matrix}}{\vdash_{\mathrm{NK}} C}\ \neg\mathrm{E}$$

By induction we have derivations NJ_1 and NJ_2 so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ_1 :: \vdash_{\mathrm{NJ}} A^*$$

and

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ_2 :: \vdash_{\mathrm{NJ}} n(\neg A^*).$$

Using the derived rule of inference $\neg\neg\neg$R, we can construct a derivation NJ satisfying

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ :: \vdash_{\mathrm{NJ}} C^*$$

as shown below.

$$\dfrac{\begin{matrix} NJ_1 \\ \vdash_{\mathrm{NJ}} A^* \end{matrix} \qquad \dfrac{\begin{matrix} NJ_2 \\ \vdash_{\mathrm{NJ}} \neg\neg\neg A^* \end{matrix}}{\vdash_{\mathrm{NJ}} \neg A^*}\ \neg\neg\neg\mathrm{R}}{\vdash_{\mathrm{NJ}} C^*}\ \neg\mathrm{E}$$

In Twelf, this case is implemented by the constant declaration

```
sound_note : sound (nk_note NK2 C (NK1:nk A)) KC
                   (nj_note (nj_triple_neg_red NJ2)
                            (n C*) NJ1)
      <- existskolm A A* KA
      <- sound NK2 (kolm_not KA) NJ2
      <- sound NK1 KA NJ1.
```

Case : Finally, let us consider the case of the double negation rule. Suppose NK is of the form

$$\dfrac{\begin{matrix} NK_1 \\ \vdash_{\mathrm{NK}} \neg\neg A \end{matrix}}{\vdash_{\mathrm{NK}} A}\ \neg\neg\mathrm{R}$$

By inversion we have $A^* = nA^+$ for some A^+. Note that

$$(\neg\neg A^*) = n(\neg(n(\neg(nA^+)))) = n(n(n(nA^+))).$$

By induction we have a derivation NJ_1 satisfying

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ_1 :: \vdash_{\mathrm{NJ}} n(n(n(nA^+))).$$

Applying the derived rule of inference $\neg\neg\neg$R three times, we obtain a derivation NJ so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1^*, \ldots, \vdash_{\mathrm{NJ}} H_k^* \triangleright NJ :: \vdash_{\mathrm{NJ}} nA^+$$

as shown below.

$$\cfrac{\cfrac{\cfrac{\begin{array}{c} NJ_1 \\ \vdash_{\mathrm{NJ}} n(n(n(nA^+))) \end{array}}{\vdash_{\mathrm{NJ}} n(n(nA^+))}\,\neg\neg\neg\mathrm{R}}{\vdash_{\mathrm{NJ}} n(nA^+)}\,\neg\neg\neg\mathrm{R}}{\vdash_{\mathrm{NJ}} nA^+}\,\neg\neg\neg\mathrm{R}$$

The Twelf implementation in this case is shown below.

```
sound_dnotr    : sound (nk_dnotr NK1) KA
                       (nj_triple_neg_red
                          (nj_triple_neg_red
                             (nj_triple_neg_red NJ1)))
    <- sound NK1 (kolm_not (kolm_not KA)) NJ1.
```

□

Theorem 5.2 (Soundness) *Given any formula A (with parameters $a_1, \ldots, a_n$) and derivation $a_1, \ldots, a_n \triangleright NK :: \vdash_{\mathrm{NK}} A$, there is a derivation $a_1, \ldots, a_n \triangleright NJ :: \vdash_{\mathrm{NJ}} A^*$.*

Proof: Apply the lemma with an empty list of propositional parameters and hypotheses. □

The full Twelf implementation of the inductive proof of the lemma is shown in the appendix.

6 Completeness

Completeness of the interpretation means that whenever a translation A^* is intuitionistically provable, then A is classically provable. We do not prove this by induction. Instead we prove two lemmas by induction and the proof of completeness follows easily.

Lemma 6.1 (Intuitionistic Derivations "are" Classical Derivations) *Given any formula A and derivation NJ so that*

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1, \ldots, \vdash_{\mathrm{NJ}} H_k \triangleright NJ :: \vdash_{\mathrm{NJ}} A,$$

there is a derivation NK so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NK}} H_1, \ldots, \vdash_{\mathrm{NK}} H_k \triangleright NK :: \vdash_{\mathrm{NK}} A.$$

Proof: This is proven by a trivial induction on NJ, using the fact that every inference rule for $\vdash_{\mathrm{NJ}}$ has a counterpart inference rule for $\vdash_{\mathrm{NK}}$. In Twelf, this proof is implemented by a higher level judgement of type

```
nj_nk : nj A -> nk A -> type.
```

We will show one case to demonstrate the induction.

Case : Suppose NJ is of the form

$$\dfrac{\begin{matrix} NJ_1 \\ \vdash_{\mathrm{NJ}}\ \exists x.B \end{matrix} \qquad \begin{matrix} \overline{\vdash_{\mathrm{NJ}}\ [a/x]B}\ u \\ NJ_2 \\ \vdash_{\mathrm{NJ}}\ A \end{matrix}}{\vdash_{\mathrm{NJ}}\ A}\ \exists\mathrm{E}^{a,u}$$

By induction we have derivations NK_1 and NK_2 such that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1, \ldots, \vdash_{\mathrm{NJ}} H_k \triangleright NK_1 :: \vdash_{\mathrm{NK}} \exists x.B$$

and

$$a_1, \ldots, a_n, a, p_1, \ldots, p_m, \vdash_{\mathrm{NJ}} H_1, \ldots, \vdash_{\mathrm{NJ}} H_k, \vdash_{\mathrm{NK}} [a/x]B \triangleright NK_2 :: \vdash_{\mathrm{NK}} A.$$

Using $\exists$E (for $\vdash_{\mathrm{NK}}$) we obtain NK defined by

$$\dfrac{\begin{array}{c} NK_1 \\ \vdash_{\mathrm{NK}} \exists x.B \end{array} \qquad \begin{array}{c} \dfrac{}{\vdash_{\mathrm{NK}} [a/x]B}\,v \\ NK_2 \\ \vdash_{\mathrm{NK}} A \end{array}}{\vdash_{\mathrm{NK}} A}\,\exists\mathrm{E}^{a,v}$$

as required.

The Twelf constant declaration implementing this case is given by

```
nj_nk_existse : nj_nk (nj_existse NJ1 NJ2) (nk_existse NK1 NK2)
    <- nj_nk NJ1 NK1
    <- ({a} {u} {v} nj_nk u v -> nj_nk (NJ2 a u) (NK2 a v)).
```

□

The full Twelf implementation is shown in the appendix. Note that we could avoid implementing this proof at all if we could implement `nj` as a subtype of `nk`.

Lemma 6.2 (Classical Equivalence of A and A^*) *Given any formula A (with individual parameters $a_1, \ldots, a_n$ and propositional parameters $p_1, \ldots, p_m$), there are hypothetical derivations NK and NK^* so that*

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NK}} A^* \triangleright NK :: \vdash_{\mathrm{NK}} A$$

and

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NK}} A \triangleright NK^* :: \vdash_{\mathrm{NK}} A^*.$$

Proof: The proof is by induction on the formula A. In contrast, The Twelf implementation works by induction over the judgment that A^* is the Kolmogorov translation of A. This is equivalent to induction on A, since the Kolmogorov translation function was defined by induction on A. The type of the higher level judgement implementing the proof is

```
equiv : kolm A A* -> (nk A -> nk A*) -> (nk A* -> nk A) -> type.
```

The inference rules which play the most important roles in the proof are (not surprisingly) $\neg\neg$R and $\neg\neg$X. It is important to notice that we need to prove the existence of NK and NK^* by a mutual induction. I will show two cases that demonstrate the need for the mutual induction.

Case : Suppose the formula is $(A \supset B)$. Recall $(A \supset B)^* = n(A^* \supset B^*)$. By induction we have NK_A, NK_A^*, NK_B, and NK_B^* so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NK}} A^* \triangleright NK_A :: \vdash_{\mathrm{NK}} A$$

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NK}} A \triangleright NK_A^* :: \vdash_{\mathrm{NK}} A^*$$

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NK}} B^* \triangleright NK_B :: \vdash_{\mathrm{NK}} B$$

and

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\mathrm{NK}} B \triangleright NK_B^* :: \vdash_{\mathrm{NK}} B^*.$$

From these we can construct NK^* by

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\dfrac{}{\vdash_{\mathrm{NK}} A^*}\,u \\ NK_A \\ \vdash_{\mathrm{NK}} A\end{array} \qquad \dfrac{}{\vdash_{\mathrm{NK}} A \supset B}\,\mathrm{hyp}}{\begin{array}{c}\vdash_{\mathrm{NK}} B \\ NK_B^* \\ \vdash_{\mathrm{NK}} B^*\end{array}}\,{\supset}\mathrm{E}}{\vdash_{\mathrm{NK}} (A^* \supset B^*)}\,{\supset}\mathrm{I}^u}{\vdash_{\mathrm{NK}} n(A^* \supset B^*)}\,\neg\neg\mathrm{X}$$

and NK by

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{}{\vdash_{\mathrm{NK}} A}\,u \\ NK_A^* \\ \vdash_{\mathrm{NK}} A^*\end{array} \qquad \dfrac{\dfrac{}{\vdash_{\mathrm{NK}} n(A^* \supset B^*)}\,\mathrm{hyp}}{\vdash_{\mathrm{NK}} A^* \supset B^*}\,\neg\neg\mathrm{R}}{\begin{array}{c}\vdash_{\mathrm{NK}} B^* \\ NK_B \\ \vdash_{\mathrm{NK}} B\end{array}}\,{\supset}\mathrm{E}}{\vdash_{\mathrm{NK}} (A \supset B)}\,{\supset}\mathrm{I}^u$$

Notice that we needed NK_A to define NK^* and we needed NK^*_A to define NK.

This case is implemented in Twelf by the constant declaration

```
equiv_imp : equiv (kolm_imp KB KA)
                  ([v:nk (A imp B)]
                     (nk_dnotx
                       (nk_impi [u:nk A*]
                          (NKB* (nk_impe v (NKA u))))))
                  ([v:nk (n (A* imp B*))]
                     (nk_impi [u:nk A]
                       (NKB (nk_impe (nk_dnotr v) (NKA* u)))))
    <- equiv KA NKA* NKA
    <- equiv KB NKB* NKB.
```

Case : Suppose the formula is $\neg A$. Recall $(\neg A)^* = n(\neg A^*) = \neg\neg\neg A^*$. By induction we have NK_A and NK^*_A so that

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\text{NK}} A^* \triangleright NK_A :: \vdash_{\text{NK}} A$$

and

$$a_1, \ldots, a_n, p_1, \ldots, p_m, \vdash_{\text{NK}} A \triangleright NK^*_A :: \vdash_{\text{NK}} A^*.$$

Using these derivations, we can define NK^* by

$$\cfrac{\cfrac{\begin{array}{c}\cfrac{\cfrac{}{\vdash_{\text{NK}} \neg\neg A^*}\,u}{\vdash_{\text{NK}} A^*}\,\neg\neg\text{R} \\ NK_A \\ \vdash_{\text{NK}} A\end{array} \qquad \cfrac{}{\vdash_{\text{NK}} \neg A}\,\text{hyp}}{p}\,\neg\text{E}}{\neg\neg\neg A^*}\,\neg\text{I}^{p,u}$$

and NK by

$$\cfrac{\cfrac{\begin{array}{c}\cfrac{}{\vdash_{\text{NK}} A}\,u \\ NK^*_A \\ \vdash_{\text{NK}} A^*\end{array} \qquad \cfrac{\cfrac{}{\vdash_{\text{NK}} n\neg A^*}\,\text{hyp}}{\vdash_{\text{NK}} \neg A^*}\,\neg\neg\text{R}}{p}\,\neg\text{E}}{\neg A}\,\neg\text{I}^{p,u}$$

Again, the fact that we need NK_A to define NK^* and NK_A^* to define NK demonstrates the need for mutual induction.

The Twelf implementation of this case is given by

```
equiv_not : equiv (kolm_not K)
                  ([v:nk (not A)]
                     (nk_noti [p] [u:nk (not not A*)]
                        (nk_note
                          v p (NKA (nk_dnotr u)))))
                  ([v:nk (n not A*)]
                     (nk_noti [p] [u:nk A]
                        (nk_note
                           (nk_dnotr v) p (NKA* u))))
    <- equiv K NKA* NKA.
```

$\square$

As usual, the full Twelf implementation of this proof is shown in the appendix.

Theorem 6.3 (Completeness) *Given any formula A (with parameters $a_1, \ldots, a_n$) and a derivation*

$$a_1, \ldots, a_n \triangleright NJ ::\vdash_{\mathrm{NJ}} A^*,$$

there is a derivation

$$a_1, \ldots, a_n \triangleright NK ::\vdash_{\mathrm{NK}} A.$$

Proof: Given A and NJ, by the first lemma there is an NK_1^* so that

$$a_1, \ldots, a_n \triangleright NK_1^* ::\vdash_{\mathrm{NK}} A^*.$$

Now, by the second there is also a hypothetical derivation NK_2 so that

$$a_1, \ldots, a_n, \vdash_{\mathrm{NK}} A^* \triangleright NK_2 ::\vdash_{\mathrm{NK}} A.$$

Combining NK_1^* and NK_2 we have a derivation NK so that

$$a_1, \ldots, a_n \triangleright NK ::\vdash_{\mathrm{NK}} A$$

as desired. $\square$

The Twelf implementation of completeness uses the type

```
complete : kolm A A* -> nj A* -> nk A -> type.
```

The judgment is defined by one constant (representing appeals to the two lemmas).

```
complete1   : complete K NJ (NK2 NK1*)
         <- nj_nk NJ NK1*
         <- equiv K NK2* NK2.
```

7 Examples

7.1 A Simple Example

Let us first demonstrate the implementation in an example which can be easily followed. Consider the following natural deduction proof of $\vdash_{\text{NK}} \bot \supset \bot$.

$$\dfrac{\dfrac{\overline{\vdash_{\text{NK}} \bot}\, u}{\vdash_{\text{NK}} \bot}\,\bot\text{E}}{\vdash_{\text{NK}} \bot \supset \bot}\,\supset\text{I}^u$$

(The alert reader will notice there is an even shorter derivation.) This derivation is represented in Twelf by the term

```
(nk_impi ([u:nk false] nk_falsee u)).
```

By soundness, there should be a corresponding intuitionistic natural deduction proof of $(\bot \supset \bot)^*$, where

$$(\bot \supset \bot)^* = n(n\bot \supset n\bot).$$

We can query Twelf to find the corresponding proof as follows.

```
%query 1 * D:sound (nk_impi ([u:nk false] nk_falsee u))
                   (kolm_imp kolm_false kolm_false)
                   NJ.
```

Notice that this query contains free variables D and NJ. Upon success, NJ will be instantiated to a term representing the corresponding intuitionistic natural deduction and D will be instantiated to the derivation of the soundness result for this instance. The term Twelf finds is

```
(nj_noti
  ([p:o] [u:nj (not (not not false imp not not false))]
    nj_note u p
     (nj_impi
       ([X1:nj (not not false)]
         nj_note X1 (not not false)
          (nj_noti [p1:o] [u1:nj false]
            (nj_falsee u1)))))).
```

The natural deduction diagram is

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}(*)\\ \vdash_{\mathrm{NJ}} n\bot\end{array}}{\vdash_{\mathrm{NJ}} n\bot \supset n\bot}\supset\mathrm{I}^{X1} \quad \dfrac{}{\vdash_{\mathrm{NJ}} \neg(n\bot \supset n\bot)}u}{\vdash_{\mathrm{NJ}} p}\neg\mathrm{E}}{\vdash_{\mathrm{NJ}} n(n\bot \supset n\bot)}\neg\mathrm{I}^{p,u}$$

where $(*)$ is the derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_{\mathrm{NJ}} \bot}u1}{\vdash_{\mathrm{NJ}} p1}\bot\mathrm{E}}{\vdash_{\mathrm{NJ}} \neg\bot}\neg\mathrm{I}^{p1,u1} \dfrac{}{\vdash_{\mathrm{NJ}} n\bot}X1}{\vdash_{\mathrm{NJ}} n\bot}\neg\mathrm{E}$$

As the corresponding natural deductions in the examples to follow are quite large, we will only show the terms representing them.

By completeness, there must be a derivation of $\vdash_{\mathrm{NK}} \bot \supset \bot$ corresponding to the derivation above of $\vdash_{\mathrm{NJ}} (\bot \supset \bot)^*$. The Twelf query

```
%query 1 * D:complete (kolm_imp kolm_false kolm_false)
   (nj_noti
     ([p:o] [u:nj (not (not not false imp not not false))]
```

```
    nj_note u p
     (nj_impi
       ([X1:nj (not not false)]
         nj_note X1 (not not false)
          (nj_noti ([p1:o] [u1:nj false] nj_falsee u1))))))
NK.
```

computes this derivation (as the instantiation of NK). The term representing the derivation is

```
(nk_impi
  ([u:nk false]
    nk_dnotr
     (nk_impe
       (nk_dnotr  % Excluded Middle Used!
         (nk_noti
           ([A1:o] [X1:nk (not (n false imp n false))]
             nk_note X1 A1
              (nk_impi
                ([X2:nk (not not false)]
                  nk_note X2 (not not false)
                   (nk_noti
                     ([A2:o] [X3:nk false]
                       nk_falsee X3)))))))
       (nk_dnotx u)))).
```

This proof is considerably more complicated and roundabout than the two-step proof with which we started. Notice that the double negation reduction rule (`nk_dnotr`) is used in the proof. The Kolmogorov translation is not minimal, as one can see by comparing the size of the translated proof with the size of the original proof.

7.2 Excluded Middle

The formula $a \vee \neg a$ is classically provable (but not intuitionistically). The proof of soundness of the Kolmogorov translation allows us to translate any classical proof of $a \vee \neg a$ to an intuitionistic proof of $(a \vee \neg a)^*$. The proof of completeness allows us to translate any intuitionistic proofs of $(a \vee \neg a)^*$

to a classical proof of $a \vee \neg a$. The Twelf implementation calculates these translations as demonstrated in the examples below.

We can derive $\vdash_{\mathrm{NK}} a \vee \neg a$ by the derivation NK shown below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}(*)\\ \vdash_{\mathrm{NK}} \neg a\end{array}}{\vdash_{\mathrm{NK}} a \vee \neg a}\vee\mathrm{I_R} \qquad \dfrac{}{\vdash_{\mathrm{NK}} \neg(a \vee \neg a)}\,u}{p}\,\neg\mathrm{E}}{\vdash_{\mathrm{NK}} \neg\neg(a \vee \neg a)}\,\neg\mathrm{I}^{p,u}}{\vdash_{\mathrm{NK}} a \vee \neg a}\,\neg\neg\mathrm{R}$$

where $(*)$ is the derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_{\mathrm{NK}} a}\,v}{\vdash_{\mathrm{NK}} a \vee \neg a}\vee\mathrm{I_L} \qquad \dfrac{}{\vdash_{\mathrm{NK}} \neg(a \vee \neg a)}\,u}{\vdash_{\mathrm{NK}} q}\,\neg\mathrm{E}}{\vdash_{\mathrm{NK}} \neg a}\,\neg\mathrm{I}^{q,v}$$

In Twelf, this derivation is represented by the term

```
(nk_dnotr
  (nk_noti
    ([p] [u:nk (not (a or (not a)))]
      nk_note u p
        (nk_orir
          (nk_noti
            ([q] [v:nk a]
              nk_note u q (nk_oril v)))))))
```

By soundness, there is a corresponding NJ derivation of the translation $n(np \vee n(\neg np))$. The implementation of the soundness proof in Twelf allows us to find the corresponding NJ derivation automatically. The Twelf query is of the form

```
%query 1 * D:({a:o} {ka:kolm a (not not a)}
  existskolm a (not not a) ka ->
  sound
```

```
  (nk_dnotr
    (nk_noti
      ([p] [u:nk (not (a or (not a)))]
        nk_note u p
         (nk_orir
           (nk_noti
             ([q] [v:nk a]
               nk_note u q (nk_oril v)))))))
  (kolm_or (kolm_not ka) ka)
  (NJ a)).
```

Notice that the query contains a free variable NJ (which may depend on the propositional parameter a). This variable NJ should be instantiated to the appropriate derivation as the goal is solved. In fact, Twelf finds the (parametric) derivation represented by the term shown below.

```
([a:o]
  nj_triple_neg_red
   (nj_triple_neg_red
     (nj_triple_neg_red
       (nj_dnotx
         (nj_noti
           ([q:o] [v:nj (n (not not not
                              (not not a or n (not not not a))))]
             nj_dneg_falser
              (nj_note (nj_triple_neg_red v) (n false)
                (nj_dnotx
                  (nj_orir
                    (nj_dnotx
                      (nj_noti
                        ([q1:o] [v1:nj (not not a)]
                          nj_dneg_falser
                           (nj_note (nj_triple_neg_red v)
                             (n false)
                             (nj_dnotx (nj_oril v1))) q1))))))
              q)))))).
```

The implementation of completeness converts this intuitionistic natural deduction (or any other intuitionistic natural deduction of $(a \vee \neg a)^*$) into a classical natural deduction of $a \vee \neg a$. For example, the intuitionistic natural deduction of $(a \vee \neg a)^*$ represented by the term

```
(nj_noti
 ([p:o] [u:nj (not (not not a or not not not not not a))]
   nj_note u p
    (nj_orir
     (nj_dnotx
      (nj_noti
       ([q:o] [v:nj (not not a)]
         nj_note u q (nj_oril v)))))))
```

is converted to the classical natural deduction represented by the term

```
([a:o]
  nk_ore
   (nk_dnotr
     (nk_noti
       ([A2:o] [X2:nk (not (not not a or n (not not not a)))]
         nk_note X2 A2
          (nk_orir
            (nk_noti
              ([A3:o] [X3:nk (not not not not a)]
                nk_note X3 A3
                 (nk_noti
                   ([A4:o] [X4:nk (not not a)]
                     nk_note X2 A4 (nk_oril X4)))))))))
   ([u:nk (not not a)] nk_oril (nk_dnotr u))
   ([u:nk (n (not not not a))]
     nk_orir
      (nk_noti
        ([p:o] [u1:nk a]
          nk_note (nk_dnotr u) p (nk_dnotx u1))))).
```

7.3 DeMorgan's Laws

It is well-known that one of DeMorgan's Laws is classically true, but not intuitionistically true. The law in question is $\neg(A \wedge B) \supset \neg A \vee \neg B$. The following term represents a derivation of $\vdash_{\mathrm{NK}} \neg(A \wedge B) \supset \neg A \vee \neg B$ (where a and b are parameters representing A and B)

```
(nk_impi
  ([u:nk (not (a and b))]
    nk_dnotr
     (nk_noti
       ([p:o] [v:nk (not (not a or not b))]
         nk_note v p
          (nk_orir
            (nk_noti
              ([q:o] [w:nk b]
                nk_note v q
                 (nk_oril
                   (nk_noti
                     ([r:o] [z:nk a]
                       nk_note u r
                        (nk_andi z w)))))))))))
```

Twelf computes the corresponding derivation of

$$\vdash_{\mathrm{NJ}} (\neg(A \wedge B) \supset \neg A \vee \neg B)^*.$$

While this term is large, the reader with a sufficient amount of patience can translate it into a natural deduction proof. The term representing this derivation is

```
(nj_dnotx
 (nj_impi
  ([X2:nj (n (not n (n a and n b)))]
   nj_triple_neg_red
   (nj_triple_neg_red
    (nj_triple_neg_red
     (nj_dnotx
      (nj_noti
       ([q:o] [v:nj (n (not not not
                          (n (not n a)
                            or n (not n b))))]
```

```
nj_dneg_falser
(nj_note (nj_triple_neg_red v) (n false)
 (nj_dnotx
  (nj_orir
   (nj_dnotx
    (nj_noti
     ([q1:o] [v1:nj (n b)]
      nj_dneg_falser
      (nj_note (nj_triple_neg_red v) (n false)
       (nj_dnotx
        (nj_oril
         (nj_dnotx
          (nj_noti
           ([q2:o] [v2:nj (n a)]
            nj_dneg_falser
            (nj_note (nj_triple_neg_red X2) (n false)
             (nj_dnotx (nj_andi v2 v1))) q2))))))
      q1))))))
q)))))))).
```

The analagous quantifier rule which is classically, but not intuitionistically, true is $\neg(\forall x.A) \supset \exists x.\neg A$. A derivation of $\vdash_{\mathrm{NK}} \neg(\forall x.A) \supset \exists x.\neg A$ is represented by the term below.

```
(nk_impi
  ([u:nk (not forall A)]
    nk_dnotr
     (nk_noti
       ([p:o] [v:nk (not exists ([x:i] not A x))]
         nk_note u p
          (nk_foralli
            ([a:i]
              nk_dnotr
               (nk_noti
                 ([q:o] [w:nk (not A a)]
                   nk_note v q
                    (nk_existsi a w)))))))))
```

The corresponding derivation of $\vdash_{\mathrm{NJ}} (\neg(\forall x.A) \supset \exists x.\neg A)^*$ found by Twelf is represented by the term

```
(nj_dnotx
 (nj_impi
  ([X2:nj (n (not n (forall ([T1:i] n (A T1)))))]
   nj_triple_neg_red
   (nj_triple_neg_red
    (nj_triple_neg_red
     (nj_dnotx
      (nj_noti
       ([q:o] [v:nj (n (not not not exists ([T2:i] n (not n (A T2)))))]
        nj_dneg_falser
        (nj_note (nj_triple_neg_red X2) (n false)
         (nj_dnotx
          (nj_foralli
           ([T3:i]
            nj_triple_neg_red
            (nj_triple_neg_red
             (nj_triple_neg_red
              (nj_dnotx
               (nj_noti
                ([q1:o] [v1:nj (n (not not not A T3))]
                 nj_dneg_falser
                 (nj_note (nj_triple_neg_red v)
                  (n false)
                  (nj_dnotx (nj_existsi T3 v1)))
                 q1)))))))))
        q)))))))).
```

8 Conclusion

The logical framework LF and the Twelf programming language can be used to represent and implement the syntax of a logical system, natural deduction proofs in the logical system, and meta-theorems about natural deduction. By representing and implementing both intuitionistic and classical natural deduction for first-order logic, we have demonstrated the power of higher-order abstract syntax for dealing with bound variables. By further representing and implementing soundness and completeness of the Kolmogorov translation, we have demonstrated the power of higher-level judgements.

In particular, this example shows that the logical framework LF can be used not only to study an object language and its meta-theoretical properties, but also to study multiple object languages and their meta-theoretical relationships. The fact that these relationships can be represented with relative ease and precision is an indication of the scope and importance of such a representation technique. At the same time, however, there are some idiosyncratic aspects to the representation. The most obvious example is the relational representation of the Kolmogorov translation function, and the corresponding need to explicitly represent the fact that every formula has a Kolmogorov translation.

A Twelf Implementations

This appendix contains the Twelf code referred to in the body of the report.

A.1 Signature for First-Order Logic

```
% File: fol.elf

i : type.  %name i T.
o : type.  %name o A.

and    : o -> o -> o.   %infix right 11 and.
imp    : o -> o -> o.   %infix right 10 imp.
or     : o -> o -> o.   %infix right 11 or.
not    : o -> o.        %prefix 12 not.
true   : o.
false  : o.
forall : (i -> o) -> o.
exists : (i -> o) -> o.
```

A.2 Classical Natural Deduction

```
% File: nk.elf

nk : o -> type.

nk_andi    : nk A -> nk B -> nk (A and B).
nk_andel   : nk (A and B) -> nk A.
nk_ander   : nk (A and B) -> nk B.
nk_impi    : (nk A -> nk B) -> nk (A imp B).
nk_impe    : nk (A imp B) -> nk A -> nk B.
nk_oril    : nk A -> nk (A or B).
nk_orir    : nk B -> nk (A or B).
nk_ore     : nk (A or B) -> (nk A -> nk C) -> (nk B -> nk C) -> nk C.
nk_noti    : ({p:o} nk A -> nk p) -> nk (not A).
nk_note    : nk (not A) -> {C:o} nk A -> nk C.
nk_truei   : nk true.
nk_falsee  : nk false -> nk C.
nk_foralli : ({a:i} nk (A a)) -> nk (forall A).
nk_foralle : nk (forall A) -> {T:i} nk (A T).
nk_existsi : {T:i} nk (A T) -> nk (exists A).
nk_existse : nk (exists A) -> ({a:i} nk (A a) -> nk C) -> nk C.

nk_dnotr   : nk A
     <- nk (not (not A)).  % double negation version of excluded middle
```

A.3 Intuitionistic Natural Deduction

```
% File: nj.elf

nj : o -> type.

nj_andi    : nj A -> nj B -> nj (A and B).
nj_andel   : nj (A and B) -> nj A.
nj_ander   : nj (A and B) -> nj B.
nj_impi    : (nj A -> nj B) -> nj (A imp B).
nj_impe    : nj (A imp B) -> nj A -> nj B.
```

```
nj_oril    : nj A -> nj (A or B).
nj_orir    : nj B -> nj (A or B).
nj_ore     : nj (A or B) -> (nj A -> nj C) -> (nj B -> nj C) -> nj C.
nj_noti    : ({p:o} nj A -> nj p) -> nj (not A).
nj_note    : nj (not A) -> {C:o} nj A -> nj C.
nj_truei   : nj true.
nj_falsee  : nj false -> nj C.
nj_foralli : ({a:i} nj (A a)) -> nj (forall A).
nj_foralle : nj (forall A) -> {T:i} nj (A T).
nj_existsi : {T:i} nj (A T) -> nj (exists A).
nj_existse : nj (exists A) -> ({a:i} nj (A a) -> nj C) -> nj C.
```

A.4 Derived Rules of Inference

The first three of the following derived rules of inference can be proven automatically by the Twelf theorem prover.

```
% Files: nk.elf,nj.elf

nk_dnotx = ([NK] (nk_noti [p:o] [u:nk (not A)] (nk_note u p NK))).

nj_dnotx = ([NJ] (nj_noti [p:o] [u:nj (not A)] (nj_note u p NJ))).

nj_triple_neg_red = ([NJ]
                       (nj_noti [q:o] [u:nj A]
                          (nj_note NJ q (nj_dnotx u)))).

nj_dneg_falser = ([NJ1] [C] (nj_note
                               NJ1 C
                               (nj_noti ([p] [u:nj false]
                                  (nj_falsee u))))).
```

A.5 Kolmogorov Translation

```
% File: kolmtrans.elf

% First, define n as shorthand for double negation.
```

```
n = [p:o] (not not p).

kolm : o -> o -> type.
%mode kolm +A -A*.

kolm_and    : kolm (A and B) (n (A* and B*))
    <- kolm A A*
    <- kolm B B*.

kolm_imp    : kolm (A imp B) (n (A* imp B*))
    <- kolm A A*
    <- kolm B B*.

kolm_or     : kolm (A or B) (n (A* or B*))
    <- kolm A A*
    <- kolm B B*.

kolm_not    : kolm (not A) (n (not A*))
    <- kolm A A*.

kolm_true   : kolm true (n true).

kolm_false  : kolm false (n false).

kolm_forall : kolm (forall A) (n (forall A*))
    <- ({a:i} kolm (A a) (A* a)).

kolm_exists : kolm (exists A) (n (exists A*))
    <- ({a:i} kolm (A a) (A* a)).

%terminates A (kolm A A*).
```

A.6 Existence of Kolmogorov Translations

```
% File: kolmtrans.elf
```

```
existskolm : {A:o} {A*:o} kolm A A* -> type.
%mode existskolm +A -A* -K.

existskolm_and   : existskolm (A and B) (n (A* and B*))
                              (kolm_and KB KA)
     <- existskolm A A* KA
     <- existskolm B B* KB.

existskolm_imp   : existskolm (A imp B) (n (A* imp B*))
                              (kolm_imp KB KA)
     <- existskolm A A* KA
     <- existskolm B B* KB.

existskolm_or    : existskolm (A or B) (n (A* or B*))
                              (kolm_or KB KA)
     <- existskolm A A* KA
     <- existskolm B B* KB.

existskolm_not    : existskolm (not A) (n not A*)
                               (kolm_not KA)
     <- existskolm A A* KA.

existskolm_true   : existskolm true  (n true) kolm_true.

existskolm_false  : existskolm false (n false) kolm_false.

existskolm_forall : existskolm (forall A) (n (forall A*))
                               (kolm_forall KA)
     <- ({a:i} existskolm (A a) (A* a) (KA a)).

existskolm_exists : existskolm (exists A) (n (exists A*))
                               (kolm_exists KA)
     <- ({a:i} existskolm (A a) (A* a) (KA a)).

%terminates A (existskolm A A* K).
```

A.7 Soundness

```
% File: sound.elf

% Translation of classical derivations to intuitionistic derivations of transformed formula.

% The induction is on the classical natural deduction NK of A.

sound : nk A -> kolm A A* -> nj A* -> type.
%mode sound +NK +K -NJ.

% andi -- easy, andi and double negate
sound_andi     : sound (nk_andi NKA NKB) (kolm_and KB KA)
                       (nj_dnotx (nj_andi NJA NJB))
    <- sound NKA KA NJA
    <- sound NKB KB NJB.

% This uses a derivation from - - (- - A+ and - - B*) to - - A*
sound_andel : sound (nk_andel (NK1:nk (A and B))) KA
                    (nj_noti ([q:o] [u:nj (not A+)]
                      (nj_note NJ1 q
                        (nj_noti ([p:o] [v:nj ((n A+) and B*)]
                          (nj_note (nj_andel v) p u))))))
    <- existskolm B B* KB
    <- sound NK1 (kolm_and KB KA) NJ1.  % NJ1::- - (- - A+ and B*)

% as in previous case
sound_ander : sound (nk_ander (NK1:nk (A and B))) KB
                    (nj_noti ([q:o] [u:nj (not B+)]
                      (nj_note NJ1 q
                        (nj_noti ([p:o] [v:nj (A* and (n B+))]
                          (nj_note (nj_ander v) p u))))))
    <- existskolm A A* KA
    <- sound NK1 (kolm_and KB KA) NJ1.  % NJ1::- - (A* and - - B+)

% impi -- apply impi and double negate
sound_impi : sound (nk_impi NKB) (kolm_imp KB KA)
```

```
                   (nj_dnotx (nj_impi NJB))
    <- ({u:nk A} {v:nj A*} sound u KA v ->
          sound (NKB u) KB (NJB v)).

sound_impe : sound (nk_impe NKI (NKA:nk A)) KB
                   (nj_noti [q:o] [v:nj (not B+)]
                     (nj_note NJI q
                       (nj_noti [p:o] [u:nj (A* imp (n B+))]
                         (nj_note (nj_impe u NJA) p v))))
    <- existskolm A A* KA
    <- sound NKA KA NJA
    <- sound NKI (kolm_imp KB KA) NJI.


% or-introductions -- simply apply or-intro and double negate
sound_oril : sound (nk_oril NK) (kolm_or KB KA)
                   (nj_dnotx (nj_oril NJ))
    <- sound NK KA NJ.

sound_orir : sound (nk_orir NK) (kolm_or KB KA)
                   (nj_dnotx (nj_orir NJ))
    <- sound NK KB NJ.

sound_ore  : sound (nk_ore (NK:nk (A or B)) NKA NKB) KC
                   (nj_noti [q:o] [v:nj (not C+)]
                     (nj_note NJ q
                       (nj_noti [p:o] [u:nj (A* or B*)]
                         (nj_note (nj_ore u NJA NJB) p v))))
    <- existskolm A A* KA
    <- existskolm B B* KB
    <- sound NK (kolm_or KB KA) NJ
    <- ({u:nk A} {v:nj A*} sound u KA v ->
          sound (NKA u) KC (NJA v))
    <- ({u:nk B} {v:nj B*} sound u KB v ->
          sound (NKB u) KC (NJB v)).

% noti -- use derived rule of inference (nj_dneg_falser)  - - false |- q
```

```
sound_noti : sound (nk_noti NK1) (kolm_not K)
                   (nj_dnotx
                     (nj_noti ([q] [v:nj A*]
                       (nj_dneg_falser (NJ1 false v) q))))
    <- ({p} {u:nk A} {v:nj A*} {kp:kolm p (n p)}
        existskolm p (n p) kp ->
        sound u K v ->
         sound (NK1 p u) kp (NJ1 p v)). % (NJ1 p v):nj (n p)

% note -- use derived rule of inference to derive - - - A from - - - - - A
% and apply note
sound_note : sound (nk_note NK2 C (NK1:nk A)) KC
                   (nj_note (nj_triple_neg_red NJ2)
                            (n C*) NJ1)
      <- existskolm A A* KA
      <- sound NK2 (kolm_not KA) NJ2
      <- sound NK1 KA NJ1.

% truei -- easy!
sound_truei : sound nk_truei kolm_true (nj_dnotx nj_truei).

% falsee -- use derived rule of inference
sound_falsee : sound (nk_falsee NK) KC
                     (nj_dneg_falser NJ C*)
    <- sound NK kolm_false NJ.

% foralli -- foralli and double negate
sound_foralli : sound (nk_foralli NK) (kolm_forall KA)
                      (nj_dnotx (nj_foralli NJ))
    <- ({a:i} sound (NK a) (KA a) (NJ a)).

sound_foralle : sound (nk_foralle NK T) (KA T)
                      (nj_noti [q:o] [v:nj (not (A+ T))]
                        (nj_note NJ q
                          (nj_noti [p:o] [u:nj (forall ([x:i] (n (A+ x))))]
                            (nj_note (nj_foralle u T) p v))))
    <- existskolm (forall A) (n (forall ([x] (n (A+ x)))))
```

```
                 (kolm_forall KA)
    <- sound NK (kolm_forall KA) NJ.

% existsi -- easy, existsi and double negate
sound_existi : sound (nk_existsi T NK) (kolm_exists KA)
                     (nj_dnotx (nj_existsi T NJ))
    <- sound NK (KA T) NJ.

% existse -- compare to ore
sound_existse : sound (nk_existse (NKEA:nk (exists A)) NKC) KC
                      (nj_noti [q:o] [v:nj (not C+)]
                        (nj_note NJnnEA q
                          (nj_noti [p:o] [u:nj (exists A*)]
                            (nj_note (nj_existse u NJC) p v))))
    <- existskolm (exists A) (n (exists A*))
                 (kolm_exists KA)
    <- sound NKEA (kolm_exists KA) NJnnEA
    <- ({a:i} {u:nk (A a)} {v:nj (A* a)}
         sound u (KA a) v ->
           sound (NKC a u) KC (NJC a v)).

% dnotr (excluded middle as the inference rule - - A |- A)
sound_dnotr    : sound (nk_dnotr NK1) KA
                       (nj_triple_neg_red
                          (nj_triple_neg_red
                             (nj_triple_neg_red NJ1)))
    <- sound NK1 (kolm_not (kolm_not KA)) NJ1.

%terminates NK (sound NK K NJ).
```

A.8 Intuitionistic Derivations "are" Classical Derivations

```
% File: complete.elf

% For completeness, we need a lemma that every nj proof "is" an nk proof.
```

```
nj_nk : nj A -> nk A -> type.
%mode nj_nk +NJ -NK.

nj_nk_andi    : nj_nk (nj_andi NJA NJB) (nk_andi NKA NKB)
    <- nj_nk NJA NKA
    <- nj_nk NJB NKB.

nj_nk_andel   : nj_nk (nj_andel NJ) (nk_andel NK)
    <- nj_nk NJ NK.

nj_nk_ander   : nj_nk (nj_ander NJ) (nk_ander NK)
    <- nj_nk NJ NK.

nj_nk_impi    : nj_nk (nj_impi NJ) (nk_impi NK)
    <- ({u} {v} nj_nk u v -> nj_nk (NJ u) (NK v)).

nj_nk_impe    : nj_nk (nj_impe NJI NJA) (nk_impe NKI NKA)
    <- nj_nk NJI NKI
    <- nj_nk NJA NKA.

nj_nk_oril    : nj_nk (nj_oril NJ) (nk_oril NK)
    <- nj_nk NJ NK.

nj_nk_oril    : nj_nk (nj_orir NJ) (nk_orir NK)
    <- nj_nk NJ NK.

nj_nk_ore     : nj_nk (nj_ore NJ NJ1 NJ2) (nk_ore NK NK1 NK2)
    <- nj_nk NJ NK
    <- ({u} {v} nj_nk u v -> nj_nk (NJ1 u) (NK1 v))
    <- ({u} {v} nj_nk u v -> nj_nk (NJ2 u) (NK2 v)).

nj_nk_noti    : nj_nk (nj_noti NJ) (nk_noti NK)
    <- ({p} {u} {v} nj_nk u v -> nj_nk (NJ p u) (NK p v)).

nj_nk_note    : nj_nk (nj_note NJ1 C NJ2) (nk_note NK1 C NK2)
    <- nj_nk NJ1 NK1
```

```
    <- nj_nk NJ2 NK2.

nj_nk_truei   : nj_nk nj_truei nk_truei.

nj_nk_falsee  : nj_nk (nj_falsee NJ) (nk_falsee NK)
    <- nj_nk NJ NK.

nj_nk_foralli : nj_nk (nj_foralli NJ) (nk_foralli NK)
    <- ({a} nj_nk (NJ a) (NK a)).

nj_nk_foralle : nj_nk (nj_foralle NJ T) (nk_foralle NK T)
    <- nj_nk NJ NK.

nj_nk_existsi : nj_nk (nj_existsi T NJ) (nk_existsi T NK)
    <- nj_nk NJ NK.

nj_nk_existse : nj_nk (nj_existse NJ1 NJ2) (nk_existse NK1 NK2)
    <- nj_nk NJ1 NK1
    <- ({a} {u} {v} nj_nk u v -> nj_nk (NJ2 a u) (NK2 a v)).

%terminates NJ (nj_nk NJ NK).
```

A.9 Classical Equivalence of A and A^*

```
% File: complete.elf

equiv : kolm A A* -> (nk A -> nk A*) -> (nk A* -> nk A) -> type.
%mode equiv +K -NK* -NK.

equiv_and : equiv (kolm_and KB KA)
                  ([u:nk (A and B)] (nk_dnotx (nk_andi (NKA* (nk_andel u))
                                                       (NKB* (nk_ander u)))))
                  ([u:nk (n (A* and B*))]
                    (nk_andi (NKA (nk_andel (nk_dnotr u)))
                             (NKB (nk_ander (nk_dnotr u)))))
    <- equiv KA NKA* NKA
    <- equiv KB NKB* NKB.
```

```
% This case clearly demonstrates why we need to prove
% the existence of NK* and NK together.
equiv_imp : equiv (kolm_imp KB KA)
                  ([v:nk (A imp B)]
                     (nk_dnotx
                       (nk_impi [u:nk A*]
                          (NKB* (nk_impe v (NKA u))))))
                  ([v:nk (n (A* imp B*))]
                     (nk_impi [u:nk A]
                        (NKB (nk_impe (nk_dnotr v) (NKA* u)))))
    <- equiv KA NKA* NKA
    <- equiv KB NKB* NKB.

equiv_or  : equiv (kolm_or KB KA)
                  ([v:nk (A or B)]
                     (nk_dnotx
                       (nk_ore v
                          ([u:nk A] (nk_oril (NKA* u)))
                          ([u:nk B] (nk_orir (NKB* u))))))
                  ([v:nk (n (A* or B*))]
                     (nk_ore (nk_dnotr v)
                        ([u:nk A*] (nk_oril (NKA u)))
                        ([u:nk B*] (nk_orir (NKB u)))))
    <- equiv KA NKA* NKA
    <- equiv KB NKB* NKB.

equiv_not : equiv (kolm_not K)
                  ([v:nk (not A)]
                     (nk_noti [p] [u:nk (not not A*)]
                        (nk_note
                          v p (NKA (nk_dnotr u)))))
                  ([v:nk (n not A*)]
                     (nk_noti [p] [u:nk A]
                        (nk_note
                           (nk_dnotr v) p (NKA* u))))
    <- equiv K NKA* NKA.
```

```
% An alternative formulation is
% equiv_true : equiv kolm_true ([v:nk true] (nk_dnotx nk_truei))
%                               ([v:nk (n true)] nk_truei).
equiv_true : equiv kolm_true ([v:nk true] (nk_dnotx v))
                             ([v:nk (n true)] (nk_dnotr v)).

equiv_false : equiv kolm_false ([v:nk false] (nk_dnotx v))
                               ([v:nk (n false)] (nk_dnotr v)).

equiv_forall : equiv (kolm_forall K)
                     ([v:nk (forall A)]
                       (nk_dnotx
                         (nk_foralli [a:i]
                           (NK* a (nk_foralle v a)))))
                     ([v:nk (n (forall A*))]
                       (nk_foralli [a:i]
                         (NK a (nk_foralle (nk_dnotr v) a))))
   <- ({a} equiv (K a) (NK* a) (NK a)).

equiv_exists : equiv (kolm_exists K)
                     ([v:nk (exists A)]
                        (nk_dnotx
                          (nk_existse v
                             ([a] [u:nk (A a)]
                                (nk_existsi a (NK* a u))))))
                     ([v:nk (n (exists A*))]
                        (nk_existse (nk_dnotr v)
                           ([a] [u:nk (A* a)]
                              (nk_existsi a (NK a u)))))
   <- ({a} equiv (K a) (NK* a) (NK a)).

%terminates K (equiv K NK* NK).
```

A.10 Completeness

```
% File: complete.elf
```

```
complete : kolm A A* -> nj A* -> nk A -> type.
%mode complete +K +NJ -NK.

% Completeness reduces to nj_nk and equiv.
complete1   : complete K NJ (NK2 NK1*)
         <- nj_nk NJ NK1*
         <- equiv K NK2* NK2.
```

References

[Gen69] G. Gentzen. Investigations into logical deduction. In *The Collected Works of Gerhard Gentzen*, pages 68–131. Amsterdam, North Holland Pub. Co., 1969.

[Kol67] A. Kolmogorov. On the principle of excluded middle. In *From Frege to Gödel*, pages 414–437. Harvard University Press, 1967.

[Pfe] Frank Pfenning. Computation and deduction. Course Notes.

[PS98] Frank Pfenning and Carsten Schuermann. Twelf user's guide, version 1.2. Technical Report CMU-CS-98-173, Carnegie Mellon University, September 1998.