Extend the Intuitionistic Logic Natural Deduction with Kolmogorov Double Negation

Yantian Yin
Yifei Che

1 Idea

The key difference between intuitionistic logic and classical logic lies in the acceptance of the law of excluded middle. Our goal is to explore the connection between the two systems, showing that they are essentially equivalent under double negation translation.

1.1 Nature Deduction

For the natural deduction we have these rules for introduction and elimination.

$$\frac{\vdash A \quad \vdash B}{\vdash A \wedge B}\ \wedge I \qquad\qquad \frac{\vdash A \wedge B}{\vdash A}\ \wedge E_L \quad \frac{\vdash A \wedge B}{\vdash B}\ \wedge E_R$$

$$\frac{\begin{array}{c}\overline{\vdash A}^{\ u}\\ \vdots \\ \vdash B\end{array}}{\vdash A \supset B}\ \supset I^u \qquad\qquad \frac{\vdash A \supset B \quad \vdash A}{\vdash B}\ \supset E$$

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

$$\frac{\begin{array}{c}\overline{\vdash A}^{\ u}\\ \vdots \\ \vdash \bot\end{array}}{\vdash \neg A}\ \neg I^{p,u} \qquad\qquad \frac{\vdash A \quad \vdash \neg A}{\vdash C}\ \neg E$$

$$\frac{}{\vdash \top}\ \top I \qquad\qquad \frac{\vdash \bot}{\vdash C}\ \bot E$$

Natural deduction (ND) corresponds to intuitionistic logic because it does not enforce that every proposition has a fixed complement. For example, if we apply *negation introduction* ($\neg I$) to a proposition A and derive $\neg A$, we cannot simply repeat this process on $\neg A$ to recover A. This asymmetry is a key reason why ND is not classical logic.

To extend ND into classical logic, we need to incorporate the law of excluded middle. One way to do this is by adding the *Kolmogorov double negation rule*, which allows us to derive A from $\neg\neg A$. We refer to this rule as $\neg\mathrm{kd}E$ (short for *not Kolmogorov double Elimination*, or **nkdE**). With this rule added, we define a new system—ND plus the Kolmogorov rule—which we call **KND**.

To fully complete KND as classical logic, we must also specify the unique complement of $\top$, which is $\bot$. We treat $\bot$ as an atomic formula. With these additions, KND forms a proper classical logic system.

$$\frac{\vdash \neg\neg A}{\vdash A}\ \neg\neg_{kd}E$$

$$\boxed{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \neg(A \lor \neg A)}\,u \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \neg(A \lor \neg A)}\,u \quad \dfrac{\dfrac{}{\vdash A}\,\nu}{\vdash A \lor \neg A}\,\lor I_1}{\vdash p}\,\neg E}{\vdash \neg A}\,\neg I^{p,\nu}}{\vdash A \lor \neg A}\,\lor I_2}{\vdash p}}{\vdash \neg\neg(A \lor \neg A)}\,\neg I^{p,u}}{\vdash A \lor \neg A}\,\neg\neg E}$$

prove of excluding middle

Now in our hand we have the intuitionistic logic ND and classical logic KND. We show that both these two logics are "the same" by proving that for any logical formula provable in KND it can also be proven in ND (soundness) and any logical formula provable in ND can also be proven in KND (completeness).

To do this first we need to define a translation function that translates formulas in ND or KND to their counterparts. We use the ktrans–Kolmogorov translation and it is defined as (n for $\neg\neg$):

$$
\begin{aligned}
A^* &= nA \quad \text{if } A \text{ is atomic} \\
(A \wedge B)^* &= n(A^* \wedge B^*) \\
(A \supset B)^* &= n(A^* \supset B^*) \\
(A \vee B)^* &= n(A^* \vee B^*) \\
(\neg A)^* &= n(\neg A^*) \\
\top^* &= n\top \\
\bot^* &= n\bot
\end{aligned}
$$

2 Inference Rules

Before we step into the proof of soundness and completeness we need some inference rules to help us eliminate double negation in ND.

First, since we can prove $\neg\neg A$ from A in ND we have the inference rule $\neg\neg X$. Same with $\neg\neg_k X$.

Also we can eliminate $\neg\neg\neg A$ to $\neg A$, we make it as $\neg\neg\neg R$.

$$
\dfrac{\dfrac{\dfrac{}{\vdash \neg A}\,u \qquad \vdash A}{\vdash p}\,\neg E}{\vdash \neg\neg A}\,\neg I^{p,u} \quad \Rightarrow \quad \boxed{\dfrac{\vdash A}{\vdash \neg\neg A}\,\neg\neg X}
$$

$$
\dfrac{\dfrac{\dfrac{}{\vdash \neg A}\,u \qquad \vdash A}{\vdash p}\,\neg kE}{\vdash \neg\neg A}\,\neg kI^{p,u} \quad \Rightarrow \quad \boxed{\dfrac{\vdash A}{\vdash \neg\neg A}\,\neg\neg kX}
$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A}\,u}{\vdash \neg\neg A}\,\neg\neg X \qquad \dfrac{}{\vdash \neg\neg\neg A}\,v}{\vdash q}\,\neg E}{\vdash \neg A}\,\neg I^{q,u} \quad \Rightarrow \quad \boxed{\dfrac{\vdash \neg\neg\neg A}{\vdash \neg A}\,\neg\neg\neg R}
$$

$$
\dfrac{\dfrac{\vdash \neg\neg\bot \qquad \dfrac{\dfrac{}{\vdash \bot}\,u}{\vdash \neg\bot}\,\neg I^u}{\vdash p}\,\neg E}{\vdash C}\,\bot E \quad \Rightarrow \quad \boxed{\dfrac{\vdash \neg\neg\bot}{\vdash C}\,\neg\neg\bot E}
$$

3 Lemma

Lemma 1 (Existence of Kolmogorov Translation)**.** *For every propositional formula A (built from atoms with $\neg$, $\wedge$, $\vee$, and $\supset$), there exists a formula A^*—its Kolmogorov translation—such that*

$$\mathrm{kolm}(A, A^*)$$

holds.

Proof. Define A^* by structural induction on A:

$$\begin{array}{rcll} p^* & := & \neg\neg p & (p \text{ atomic}) \\ (A \wedge B)^* & := & \neg\neg\big(A^* \wedge B^*\big) & \\ (A \vee B)^* & := & \neg\neg\big(A^* \vee B^*\big) & \\ (A \supset B)^* & := & \neg\neg\big(A^* \supset B^*\big) & \\ (\neg A)^* & := & \neg\neg\neg A^* & \end{array}$$

Base case. If A is atomic ($A = p$), set $A^* = \neg\neg p$; then $\mathrm{kolm}(p, \neg\neg p)$ is immediate.

Inductive step. Assume A^* and B^* already exist. The table above defines translations for each composite constructor, and the corresponding kolm derivations follow from the induction hypotheses.

Since the construction terminates for every syntax tree, the mapping $A \mapsto A^*$ is total; hence $\mathrm{kolm}(A, A^*)$ holds for all formulas A. □

Lemma 2 (Every Kolmogorov image is a double negation)**.** *Let A and B be propositional formulas. If $\mathrm{kolm}(A, B)$ holds, then there exists a formula C such that*

$$B = \neg\neg C \quad \textit{and} \quad \mathrm{kolm}(A, C).$$

Proof. Proceed by induction on the derivation of $\mathrm{kolm}(A, B)$.

Atomic case. If $A = p$ and the derivation ends with the rule for atoms, we have $B = \neg\neg p$. Set $C := p$. Then $\mathrm{kolm}(p, C)$ is the very last step of the given derivation, so the claim holds.

Conjunction. Suppose $\mathrm{kolm}(A \wedge B_0,\, B)$ is derived via

$$B = \neg\neg\big(D_1 \wedge D_2\big), \qquad \mathrm{kolm}(A, D_1), \qquad \mathrm{kolm}(B_0, D_2).$$

Take $C := D_1 \wedge D_2$. Both sub-derivations satisfy the induction hypothesis, hence $\mathrm{kolm}(A \wedge B_0, C)$ holds and $B = \neg\neg C$.

Disjunction and implication. The arguments are identical: each rule in the definition of kolm produces the outer pattern $\neg\neg(\,\cdot\,)$, so we peel off that pair of negations and re-assemble the inner sub-translations using the induction hypothesis.

Negation. From $\mathrm{kolm}(\neg A_0, B)$ we get $B = \neg\neg\neg D$ with $\mathrm{kolm}(A_0, D)$. Let $C := \neg D$; then again $B = \neg\neg C$ and $\mathrm{kolm}(\neg A_0, C)$.

Thus every derivation of $\mathrm{kolm}(A, B)$ factors through an inner formula C with $B = \neg\neg C$, proving the claim. $\square$

4 Soundness

Now with the help of our derivation rule, and lemma we can prove our soundness theorem. We do structural proof on the last used operation.

Case:

$$\frac{A \quad B}{A \wedge^k B}\,\wedge kI \quad\Rightarrow\quad \dfrac{\dfrac{\neg\neg A \quad \neg\neg B}{\neg\neg A \wedge \neg\neg B}\,\wedge I}{\neg\neg(\neg\neg A \wedge \neg\neg B)}\,\neg\neg X$$

Case:

$$\frac{A \wedge^k B}{A}\,\wedge kE1 \quad\Rightarrow\quad \dfrac{\dfrac{\dfrac{}{\neg\neg(\neg\neg A \wedge \neg\neg B)}\,u \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg\neg A \wedge \neg\neg B}\,u}{\neg\neg A}\,\wedge E1 \quad \dfrac{}{\neg A}\,v}{q}\,\neg I^{q,u}}{\neg(\neg\neg A \wedge \neg\neg B)}}{p}\,\neg E}{\neg\neg A}\,\neg I^{p,v}$$

Case:

$$\frac{A \wedge^k B}{B}\,\wedge kE2 \quad\Rightarrow\quad \dfrac{\dfrac{\dfrac{}{\neg\neg(\neg\neg A \wedge \neg\neg B)}\,u \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg\neg A \wedge \neg\neg B}\,u}{\neg\neg B}\,\wedge E2 \quad \dfrac{}{\neg B}\,v}{q}\,\neg I^{q,u}}{\neg(\neg\neg A \wedge \neg\neg B)}}{p}\,\neg E}{\neg\neg B}\,\neg I^{p,v}$$

Case:

$$\dfrac{\begin{array}{c}\overline{A}^{\;u}\\ \vdots\\ B\end{array}}{A \supset^k B}\supset kI^u \qquad\Rightarrow\qquad \dfrac{\dfrac{\begin{array}{c}\overline{\neg\neg A}^{\;u}\\ \vdots \\ \neg\neg B\end{array}}{\neg\neg A\supset\neg\neg B}\supset I^u}{\neg\neg(\neg\neg A\supset\neg\neg B)}\neg\neg X$$

Case:

$$\dfrac{A\supset^k B \quad A}{B}\supset kE \qquad\Rightarrow\qquad \dfrac{\dfrac{\overline{\neg\neg(\neg\neg A\supset\neg\neg B)}^{\;u} \qquad \dfrac{\dfrac{\dfrac{\overline{\neg\neg A\supset\neg\neg B}^{\;u} \quad \neg\neg A}{\neg\neg B}\supset E \quad \overline{\neg B}^{\;v}}{p}\neg E}{\neg(\neg\neg A\supset\neg\neg B)}\neg I^{p,u}}{q}\neg E}{\neg\neg B}\neg I^{q,v}$$

Case:

$$\dfrac{A}{A\vee^k B}\vee kI1 \qquad\Rightarrow\qquad \dfrac{\dfrac{\neg\neg A}{\neg\neg A\vee\neg\neg B}\vee I1}{\neg\neg(\neg\neg A\vee\neg\neg B)}\neg\neg X$$

Case:

$$\dfrac{B}{A\vee^k B}\vee kI2 \qquad\Rightarrow\qquad \dfrac{\dfrac{\neg\neg B}{\neg\neg A\vee\neg\neg B}\vee I2}{\neg\neg(\neg\neg A\vee\neg\neg B)}\neg\neg X$$

Case:

$$\dfrac{A\vee^k B \qquad \begin{array}{c}\overline{A}^{\;u}\\ \vdots\\ C\end{array} \qquad \begin{array}{c}\overline{B}^{\;v}\\ \vdots\\ C\end{array}}{C}\vee kE^{u,v} \qquad\Longrightarrow\qquad \dfrac{\dfrac{\neg\neg\big(\neg\neg A\vee\neg\neg B\big)}{\neg\neg A\ \vee\ \neg\neg B}\neg\neg E \qquad \begin{array}{c}\overline{\neg\neg A}^{\;u}\\ \vdots\\ \neg\neg C\end{array} \qquad \begin{array}{c}\overline{\neg\neg B}^{\;v}\\ \vdots\\ \neg\neg C\end{array}}{\neg\neg C}\vee E^{u,v}$$

Case:

$$\dfrac{\neg\neg A}{A}\neg\neg E \qquad\Rightarrow\qquad \dfrac{\neg\neg\neg\neg A}{\neg\neg A}\neg\neg\neg R$$

Case:

$$\frac{\neg\neg A}{A}\;\neg\neg E \qquad\Rightarrow\qquad \frac{\neg\neg\neg\neg A}{\neg\neg A}\;\neg\neg\neg R$$

5 Completeness

Case:

$$\frac{A \quad B}{A \wedge B}\;\wedge I \qquad\Rightarrow\qquad \frac{\dfrac{\neg\neg A'}{A'}\;\neg\neg E \qquad \dfrac{\neg\neg B'}{B'}\;\neg\neg E}{A' \wedge^k B'}\;\wedge kI$$

Case:

$$\frac{A \wedge B}{A}\;\wedge E1 \qquad\Rightarrow\qquad \frac{\dfrac{\neg\neg A' \wedge^k \neg\neg B'}{\neg\neg A'}\;\wedge kE1}{A'}\;\neg\neg E$$

Case:

$$\frac{A \wedge B}{A}\;\wedge E2 \qquad\Rightarrow\qquad \frac{\dfrac{\neg\neg A' \wedge^k \neg\neg B'}{\neg\neg B'}\;\wedge kE2}{B'}\;\neg\neg E$$

Case:

$$\frac{\begin{array}{c}\overline{A}^{\;u}\\ \vdots\\ B\end{array}}{A \supset B}\;\supset I^u \qquad\Rightarrow\qquad \frac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\overline{\neg\neg A'}^{\;u}\\ \vdots\\ \neg\neg B'\end{array}}{\neg\neg A' \supset^k \neg\neg B'}\;\supset kI^u \quad \dfrac{\overline{A'}^{\;v}}{\neg\neg A'}\;\neg\neg X}{\neg\neg B'}\;\supset kE}{B'}\;\neg\neg E}{A' \supset^k B'}\;\supset kI^v$$

Case:

$$\frac{A \supset B \quad A}{B}\;\supset E \qquad\Rightarrow\qquad \frac{\dfrac{\neg\neg A' \supset^k \neg\neg B' \qquad \neg\neg A'}{\neg\neg B'}\;\supset kE}{B'}\;\neg\neg E$$

Case:

$$\dfrac{A}{A \vee B}\;\vee I1 \qquad \Rightarrow \qquad \dfrac{\dfrac{\neg\neg A'}{A'}\;\neg\neg E}{A' \vee^k B'}\;\vee kI1$$

Case:

$$\dfrac{B}{A \vee B}\;\vee I2 \qquad \Rightarrow \qquad \dfrac{\dfrac{\neg\neg B'}{B'}\;\neg\neg E}{A' \vee^k B'}\;\vee kI2$$

Case:

$$\dfrac{A \vee B \qquad \begin{array}{c}\overline{A}^{\;u}\\ \vdots \\ C\end{array} \qquad \begin{array}{c}\overline{B}^{\;v}\\ \vdots \\ C\end{array}}{C}\;\vee E^{u,v} \quad \Longrightarrow \quad \dfrac{\dfrac{\neg\neg A' \vee^k \neg\neg B' \qquad \begin{array}{c}\neg\neg A'\\ \vdots \\ \neg\neg C'\end{array} \qquad \begin{array}{c}\neg\neg B'\\ \vdots \\ \neg\neg C'\end{array}}{\neg\neg C'}\;\vee kE^{u,v}}{C'}\;\neg\neg E$$

6 Beluga

```
i : type. % Type of individuals; we leave elements abstract.

LF o : type = % formulas
  |  : o -> o -> o
  |  : o
  |  : o
  |  : o →o →o
  |  : o →o →o
  | ¬ : o -> o
  |  : (i →o) →o %  \x. A(x)  B  \x. A(x)  B(x)
  |  : (i →o) →o %  \x. A(x)  B  \x. A(x)  B(x)
;

--prefix ¬ 10.
--infix  6 right.
--infix  5 right.
--infix  4 right.
--prefix  8.
--prefix  8.


LF nd : o -> type =
  | I : (nd A -> nd B)
   -> nd (A  B)

  | E : nd (A  B) -> nd A
```

```
   -> nd B

  | ¬I : ({p:o} nd A -> nd p)
   -> nd (¬ A)

  | ¬E : nd (¬ A) -> nd A
   -> nd C

  |  I : nd A →nd B→
    nd (A  B)

  |  El : nd (A  B)→
     nd A

  |  Er : nd (A  B)→
     nd B

  |  Il : nd A→
     nd (A  B)
  |  Ir : nd B →nd (A  B)
  |  E : nd (A  B)→
        (nd A →nd C)→
        (nd B →nd C)→
        nd C
  |  I : nd
  |  E : nd  →
         nd C
;


LF knd : o →type =
  |  kI : (knd A →knd B)→
         knd (A  B)

  |  kE : knd (A  B) →knd A→
          knd B

  |  kI : knd A →knd B→
         knd (A  B)

  |  kEl : knd (A  B)→
         knd A

  |  kEr : knd (A  B)→
         knd B

  |  kIl : knd A→
         knd (A  B)

  |  kIr : knd B→
         knd (A  B)

  |  kE : knd (A  B)→
         (knd A →knd C)→
         (knd B →knd C)→
         knd C
```

```
  | kI : knd

  | kE : knd →
        knd C

  | ¬kI : ({p:o} knd A →knd p)→
           knd (¬ A)

  | ¬kE : knd (¬ A) -> knd A
        -> knd C

  | ¬¬kE : knd (¬ (¬ A))→
           knd A
;

LF ktrans : o -> o -> type =
  | ktrans_I : ktrans A A* -> ktrans B B* ->
             ktrans (A  B) (¬ (¬ (A*  B*)))

  | ktrans_El : ktrans (A  B) (¬ (¬ (A*  B*))) ->
               ktrans A A*

  | ktrans_Er : ktrans (A  B) (¬ (¬ (A*  B*))) ->
               ktrans B B*

  | ktrans_I : ktrans A A* -> ktrans B B* ->
             ktrans (A  B) (¬ (¬ (A*  B*)))

  | ktrans_E : ktrans (A  B) (¬ (¬ (A*  B*))) -> ktrans A A* ->
             ktrans B B*

  | ktrans_I : ktrans A A* -> ktrans B B* ->
             ktrans (A  B) (¬ (¬ (A*  B*)))

  | ktrans_E : ktrans (A  B) (¬ (¬ (A*  B*))) -> ktrans A A* -> ktrans B B* -> ktrans C C*
       ->
             ktrans C C*

  | ktrans_¬I : ktrans A A* ->
             ktrans (¬ A) (¬ (¬ (¬ A*)))

  | ktrans_¬E : ktrans (¬ A) (¬ (¬ (¬ A*))) -> ktrans A A* -> ktrans C C* ->
             ktrans C C*

  | ktrans_ : ktrans  (¬ (¬  ))

  | ktrans_ : ktrans  (¬ (¬  ))
;

%{

Translation context

}%
schema ctx = knd A ;
schema nctx = nd A ;
```

```
inductive Rel : Γ{:ctx} Γ{' : nctx} ctype =
  | empty : Rel [ ] [ ]
  | cons : Rel Γ[] Γ[']→
           Rel Γ[, a:knd A[]] Γ[', a:nd A[]];

rec ktrans_of :  [ o] → [ o] =
fn A =>
  case A of
    |  [ A1  B1] =>
       let  [ A1*] = ktrans_of  [ A1] in
       let  [ B1*] = ktrans_of  [ B1] in
       [ ¬ (¬ (A1*  B1*))]
    |  [ A1  B1] =>
       let  [ A1*] = ktrans_of  [ A1] in
       let  [ B1*] = ktrans_of  [ B1] in
       [ ¬ (¬ (A1*  B1*))]
    |  [ A1  B1] =>
       let  [ A1*] = ktrans_of  [ A1] in
       let  [ B1*] = ktrans_of  [ B1] in
       [ ¬ (¬ (A1*  B1*))]
    |  [ ¬ A1] =>
       let  [ A1*] = ktrans_of  [ A1] in
       [ ¬ (¬ (¬ A1*))]
    |  [  ] =>
       [ ¬ (¬  )]
    |  [  ] =>
       [ ¬ (¬  )]

;

rec existsktrans : {A :  [ o]} {A* :  [ o]}  [ ktrans A A*] =
mlam A => mlam A* =>
  case  [ A] of
  |  [ A1  B1] =>
    (case  [ A*] of
    |  [ ¬ (¬ (A1*  B1*))] =>
     let  [ KA] = existsktrans  [ A1]  [ A1*] in
     let  [ KB] = existsktrans  [ B1]  [ B1*] in
     [  ktrans_I KA KB])
  |  [ A1  B1] =>
    (case  [ A*] of
    |  [ ¬ (¬ (A1*  B1*))] =>
     let  [ KA] = existsktrans  [ A1]  [ A1*] in
     let  [ KB] = existsktrans  [ B1]  [ B1*] in
     [  ktrans_I KA KB])
  |  [ A1  B1] =>
    (case  [ A*] of
    |  [ ¬ (¬ (A1*  B1*))] =>
     let  [ KA] = existsktrans  [ A1]  [ A1*] in
     let  [ KB] = existsktrans  [ B1]  [ B1*] in
     [  ktrans_I KA KB])
  |  [ ¬ A1] =>
    (case  [ A*] of
    |  [ ¬ (¬ (¬ A1*))] =>
     let  [ KA] = existsktrans  [ A1]  [ A1*] in
     [ ktrans_¬I KA])
  |  [  ] =>
```

```
    (case  [ A*] of
    |  [ ¬ (¬  )] =>
      [  ktrans_])
  |  [  ] =>
    (case  [ A*] of
    |  [ ¬ (¬  )] =>
      [  ktrans_])
;

rec dnotx : Rel Γ[] Γ['] →Γ['  nd A] →Γ['  nd (¬ (¬ A))] =
fn rel => fn prf =>
  let Γ['  D[]] = prf inΓ
  ['  ¬I (\p. \u. ¬E u D[])]
;

rec triple_neg_red : Rel Γ[] Γ['] →Γ['  nd (¬ (¬ (¬ A[])))] →Γ['  nd (¬ A[])] =
fn rel => fn prf =>
  let Γ['  D[]] = prf inΓ
  ['  ¬I (\p. \u. ¬E D[] (¬I (\q. \v. ¬E v u)))]
;

rec dneg_falser : Rel Γ[] Γ['] →Γ['  nd (¬ (¬  ))] →Γ['  nd C[]] =
fn rel => fn prf =>
  let Γ['  D[]] = prf inΓ
  ['  ¬E D[] (¬I \p. \u.  (E u))]
;

rec sound : Rel Γ[] Γ['] →Γ[  knd A[]] →[  ktrans A A*] →Γ['  nd A*[]] =
fn rel => fn prf => fn kolm =>
  case (prf, kolm) of
  | Γ([  kI NKA NKB], [  ktrans_I KA KB]) =>
    let Γ['  NJA] = sound rel Γ[  NKA] [  KA] in
    let Γ['  NJB] = sound rel Γ[  NKB] [  KB] in
    dnotx rel Γ['  I NJA NJB]
  | Γ([  kEl (NK1 : knd (A[]  B[]))], [ (KA : (ktrans A (¬(¬ A**))))]) =>
    let  [ B*] = ktrans_of  [ B] in
    let  [ KB] = existsktrans  [ B]  [ B*] in
    let Γ['  NJ1[]] = sound rel Γ[  NK1]  [  ktrans_I KA KB] inΓ
    ['  ¬I (\p. \u.
      ¬E NJ1[] (
        ¬I (\q. \v.
          ¬E  (El v) u)))]
  | Γ([  kEr (NK1 : knd (A[]  B[]))], [ (KB : (ktrans B (¬(¬ B**))))]) =>
    let  [ A*] = ktrans_of  [ A] in
    let  [ KA] = existsktrans  [ A]  [ A*] in
    let Γ['  NJ1[]] = sound rel Γ[  NK1]  [  ktrans_I KA KB] inΓ
    ['  ¬I (\p. \u.
      ¬E NJ1[] (
        ¬I (\q. \v.
          ¬E  (Er v) u)))]
  | Γ([  kI \u. (NKB : knd B[])], [  ktrans_I (KA : (ktrans A A*)) (KB : (ktrans B B*))])
        =>
    let Γ[', v:nd A*[]  v] = sound (cons rel) Γ[, u:knd A[]  u]  [ KA] in
    let Γ[', v:nd A*[]  NJB] = sound (cons rel) Γ[, u:knd _  NKB]  [ KB] in
    dnotx rel Γ['  I \u. NJB]
  | Γ([  kE (NKI : knd (A[]  B[])) (NKA : knd A[])], [ (KB : (ktrans B (¬(¬ B**))))]) =>
    let  [ A*] = ktrans_of  [ A] in
```

```
    let  [ KA] = existsktrans  [ A]  [ A*] in
    let Γ[' NJA[]] = sound rel Γ[ NKA]  [ KA] in
    let Γ[' NJI[]] = sound rel Γ[ NKI]  [ ktrans_I KA KB] inΓ
    [' ¬I (\p. \u.
      ¬E NJI[] (
        ¬I (\q. \v.
          ¬E  (E v NJA[]) u)))]
  | Γ([  kIl NK],  [ ktrans_I KA KB]) =>
    let Γ[' NJ] = sound rel Γ[ NK]  [ KA] in
    dnotx rel Γ['  Il NJ]
  | Γ([  kIr NK],  [ ktrans_I KA KB]) =>
    let Γ[' NJ] = sound rel Γ[ NK]  [ KB] in
    dnotx rel Γ['  Ir NJ]
  | Γ([  kE (NK[] : knd (A[]  B[])) (\ua. NKA[]) (\ub. NKB[]) ],[ ktrans_E (KO : (ktrans (A
         B) (¬ (¬ (¬(¬ A**)  ¬(¬ B**)))))) (KA : (ktrans A (¬(¬ A**)))) (KB : (ktrans B (¬(
       ¬ B**)))) (KC : (ktrans C (¬(¬ C**))))]) =>
    let Γ[' NJ[]] = sound rel Γ[ NK[]]  [ KO] in
    let Γ[', va:nd (¬(¬ A**[]))  va] = sound (cons rel) Γ[, ua:knd A[]  ua]  [ KA] in
    let Γ[', vb:nd (¬(¬ B**[]))  vb] = sound (cons rel) Γ[, ub:knd B[]  ub]  [ KB] in
    let Γ[', va:nd (¬(¬ A**[]))  NJA[]] = sound (cons rel) Γ[, ua:knd A[]  NKA[]]  [ KC] in
    let Γ[', vb:nd (¬(¬ B**[]))  NJB[]] = sound (cons rel) Γ[, ub:knd B[]  NKB[]]  [ KC] inΓ
    [' ¬I (\p. \u.
      ¬E NJ[] (
        ¬I (\q. \v.
          ¬E  (E v (\va. NJA[]) (\vb.NJB[])) u)))]
  %{| Γ([ ¬kI \p. \u. NK1],  [ ktrans_¬I (K : (ktrans A A*))]) =>
    let  [ kp] = existsktrans  [ ]  [ ¬ (¬  )] in
    let Γ[', v: nd A*[]  v] = sound (cons rel) Γ[, u:knd A[]  u]  [ K] in
    let Γ[', v: nd A*[]  NJ1[..,_,_]] = sound (cons rel) Γ[, u: knd A[]  NK1 [..,,_]]  [ kp]
          in
    let Γ[' NJ1N[]] = dneg_falser rel Γ [' NJ1] in
    dnotx rel Γ['  ¬I (\p. \u. NJ1N[..,p])]}%
  | Γ([ ¬kE NK1[] (NK2[] : knd A[])],  [ ktrans_¬E (KN : (ktrans (¬ A) (¬ (¬ (¬ A*))))) (KA
         : (ktrans A A*)) (KC : (ktrans C C*))]) =>
    let Γ[' NJ1] = sound rel Γ[ NK1[]]  [ KN] in
    let Γ[' NJ2] = sound rel Γ[ NK2[]]  [ KA] in
    let Γ[' NJ1'] = triple_neg_red rel Γ[' NJ1] inΓ
    [' ¬E NJ1' NJ2]
;

rec nj_nk : Rel Γ[] Γ['] →Γ['  nd A[]] →Γ[  knd A[]] =
fn rel => fn prf =>
  case prf of
  | Γ['  I NJA NJB] =>
    let Γ[ NKA] = nj_nk rel Γ[' NJA] in
    let Γ[ NKB] = nj_nk rel Γ[' NJB] inΓ
    [  kI NKA NKB]
  | Γ['  El NJ] =>
    let Γ[ NK] = nj_nk rel Γ['  NJ] inΓ
    [  kEl NK]
  | Γ['  Er NJ] =>
    let Γ[ NK] = nj_nk rel Γ['  NJ] inΓ
    [  kEr NK]
  | Γ['  I \u. NJ] =>
    let Γ[, a: knd A[]  NK] = nj_nk (cons rel) Γ[', a:nd _  NJ] inΓ
    [  kI \u. NK]
  | Γ['  E NJI NJA] =>
```

```
   let Γ[  NKI] = nj_nk rel Γ['  NJI] in
   let Γ[  NKA] = nj_nk rel Γ['  NJA] inΓ
   [  kE NKI NKA]
  | Γ['  Il NJ] =>
   let Γ[  NK] = nj_nk rel Γ['  NJ] inΓ
   [  kIl NK]
  | Γ['  Ir NJ] =>
   let Γ[  NK] = nj_nk rel Γ['  NJ] inΓ
   [  kIr NK]
  | Γ['  E NJ (\ua. NJ1) (\ub. NJ2)] =>
   let Γ[  NK] = nj_nk rel Γ['  NJ] in
   let Γ[, a: knd A[]  NK1] = nj_nk (cons rel) Γ[', a:nd _  NJ1] in
   let Γ[, a: knd B[]  NK2] = nj_nk (cons rel) Γ[', a:nd _  NJ2] inΓ
   [  kE NK (\va.NK1) (\vb.NK2)]
 %{| Γ['  ¬I \p. \u. NJ] =>
   let Γ[, a: knd A[]  NK] = nj_nk (cons rel) Γ[', a:nd _  NJ[..]] inΓ
   ['  kI \q. \v. NK]}%
  | Γ['  ¬E NJ1 NJ2] =>
   let Γ[  NK1] = nj_nk rel Γ['  NJ1] in
   let Γ[  NK2] = nj_nk rel Γ['  NJ2] inΓ
   [  ¬kE NK1 NK2]
;

rec equiv : Rel Γ[] Γ['] → [ ktrans A A*] →Γ[  knd A*[]] →Γ[  knd A[]] =
fn rel => fn kolm => fn prf =>
 case (kolm, prf) of
  |  ([  ktrans_I KA KB], Γ[  (u : knd (¬(¬(A*  B*))))]) =>
   let Γ[  NKA] = equiv rel  [ KA] Γ[  kEl (¬¬kE u)] in
   let Γ[  NKB] = equiv rel  [ KB] Γ[  kEr (¬¬kE u)] inΓ
   [  kI NKA NKB]
;
```