Logical- and Meta-Logical Frameworks
Lecture 2

Carsten Schürmann

August 8, 2006

Recall from last time

Summary

- Judgments. A true
- Evidence. $\mathcal{D} :: A \supset \neg\neg A$ true
- Principle of structural induction:

$$\text{If } \mathcal{D} :: A \text{ true and } \mathcal{E} :: dn(A) = A' \text{ then } \mathcal{F} :: A' \text{ true.}$$

- Inversion.
- Generalization of induction hypothesis.

Homework

- Finish the proof of cases impE, negE, orI, and orE.
- Finish the proof of the substitution lemma.

Challenge

How can we use the computer to carry out such arguments?
While minimizing human intervention?

Historical Overview

- Theorem prover. Nqthm, Otter
- Hereditary Harrop formulas. Isabelle, λProlog
- λ^{Π} (LF). Automath, LF, Elf, Twelf
- Substructural logical frameworks. Forum, LLF, OLF
- Equational logic, rewriting. Maude, ELAN
- Constructive type theories. ALF, Agda, Coq, LEGO, Nuprl

Representation function

Domain *Informal* mathematical domain

Range Computer internal format

- Binary
- Base types, e.g. integers, strings
- Datatypes
- Logical Framework
- Logic

Notation $\ulcorner \cdot \urcorner = \cdot$

Representation function (cont'd)

Definition The representation invariant states explicitly what is not in implicit in the representation

Example x integer, but $-3 \leq x \leq 9$.

Observation The more expressive the representation language the less need for representation invariants.

Example $x \in \{y | \text{int}(x) \wedge -3 \leq x \leq 9\}$

But The more expressive the representation language the more human intervention is required.

Question Can we strike a blance?

Representation function (cont'd)

A representation is adequate if it is

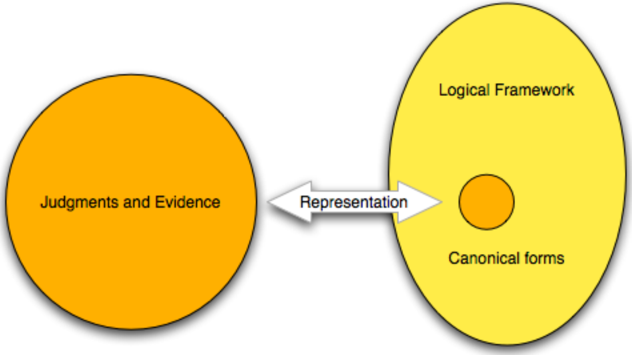
injective

- $\ulcorner \mathcal{J}_1 \urcorner = \ulcorner \mathcal{J}_2 \urcorner$ implies $\mathcal{J}_1 = \mathcal{J}_2$.
- $\ulcorner \mathcal{D}_1 :: \mathcal{J}_1 \urcorner = \ulcorner \mathcal{D}_2 :: \mathcal{J}_2 \urcorner$ implies $\mathcal{J}_1 = \mathcal{J}_2$ and $\mathcal{D}_1 :: \mathcal{J}_1 = \mathcal{D}_2 :: \mathcal{J}_2$,

surjective

- For every type A, there exists a judgment $\mathcal{J}$, sucht that $\ulcorner \mathcal{J} \urcorner = A$ exists.
- for every $M : A$, there exists evidence $\mathcal{D} : \mathcal{J}$, such that $\ulcorner \mathcal{J} \urcorner = A$ and $\ulcorner \mathcal{D} : \mathcal{J} \urcorner = M : A$.

Representation Methodology



Representation function

Lemma: If $\ulcorner \cdot \urcorner$ is adequate then its inverse exist.

Definition $\llcorner \cdot \lrcorner$ is the inverse of the representation function.

$$
\begin{array}{rcl}
\llcorner \ulcorner \mathsf{wff} \urcorner \lrcorner & = & \mathsf{wff} \\
\llcorner \ulcorner A \ \mathsf{true} \urcorner \lrcorner & = & A \ \mathsf{true} \\
\llcorner \ulcorner \neg\neg p :: \mathsf{wff} \urcorner \lrcorner & = & \neg\neg p
\end{array}
$$

$$
\llcorner \ulcorner \mathcal{D} = \cfrac{\cfrac{\cfrac{\cfrac{}{A \ \mathsf{true}}\, u \quad \cfrac{}{\neg A \ \mathsf{true}}\, v}{p \ \mathsf{true}}\, \mathsf{negE}}{\neg\neg A \ \mathsf{true}}\, \mathsf{impl}^{u}}{A \supset \neg\neg A \ \mathsf{true}}\, \mathsf{negI}^{p,v} \urcorner \lrcorner = \mathcal{D} :: A \supset \neg\neg A
$$

Advantages of adequate encodings

- Proof checking can be reduced to typechecking
- Example: Proof carrying code, typed assembly language.

Logical framework

- Type theory
- Dependent types
- Functions, definitional equality $\beta\eta$.
- No impredicativity
  - No polymorphism
  - No type constructors
- Signatures

Our goal is to understand all of this today.

Logical framework LF (Part 1)

Simply-typed

Types	$A, B ::= a \mid A \to B$
Objects	$M, N ::= c \mid M\ N$
Signatures	$\Sigma ::= \cdot \mid \Sigma, c : A \mid \Sigma, a : \text{type}$
Contexts	$\Gamma ::= \cdot \mid \Gamma, x : A$
Validity	

- Valid types: $\Gamma \vdash_\Sigma A : \text{type}$
- Valid objects: $\Gamma \vdash_\Sigma M : A$

Example (Part 1)

$$
\begin{array}{rcl}
\ulcorner \mathrm{wff} \urcorner & = & \mathrm{wff} \\
\ulcorner \neg A \urcorner & = & \mathrm{neg}\ \ulcorner A \urcorner \\
\ulcorner A \wedge B \urcorner & = & \mathrm{and}\ \ulcorner A \urcorner\ \ulcorner B \urcorner \\
\ulcorner A \vee B \urcorner & = & \mathrm{or}\ \ulcorner A \urcorner\ \ulcorner B \urcorner \\
\ulcorner A \supset B \urcorner & = & \mathrm{imp}\ \ulcorner A \urcorner\ \ulcorner B \urcorner
\end{array}
\qquad
\begin{array}{rcl}
\mathrm{wff} & : & \mathrm{type} \\
\mathrm{neg} & : & \mathrm{wff} \to \mathrm{wff} \\
\mathrm{and} & : & \mathrm{wff} \to \mathrm{wff} \to \mathrm{wff} \\
\mathrm{or} & : & \mathrm{wff} \to \mathrm{wff} \to \mathrm{wff} \\
\mathrm{imp} & : & \mathrm{wff} \to \mathrm{wff} \to \mathrm{wff}
\end{array}
$$

Lemma $\ulcorner\urcorner$ is adequate.

Proof By structural induction.

But: How do we represent arbitrary p?

$$
\dfrac{\begin{array}{c}\dfrac{}{A\ \mathrm{true}}\,u \\ \vdots \\ p\ \mathrm{true}\end{array}}{\neg A\ \mathrm{true}}\ \mathrm{negI}^{p,u}
$$

Motivation (Part 2)

Terms with *holes* The best way to represent a formula with a *hole*:

$$
\begin{aligned}
& \ulcorner p :: \mathsf{wff} \vdash \neg\neg p :: \mathsf{wff} \urcorner \\
= \quad & p : \mathsf{wff} \vdash \mathsf{neg}\ (\mathsf{neg}\ p) :: \mathsf{wff} \\
\text{iff} \quad & \cdot \vdash \lambda p : \mathsf{wff}.\, \mathsf{neg}\ (\mathsf{neg}\ p) :: \mathsf{wff} \to \mathsf{wff}
\end{aligned}
$$

The substitution principle

$$
\begin{aligned}
& \ulcorner [\neg A/p]\neg p \urcorner \\
= \quad & \ulcorner p :: \mathsf{wff} \vdash \neg p \urcorner\ \ulcorner \neg A \urcorner \\
= \quad & (\lambda p : \mathsf{wff}.\, \mathsf{neg}\ p)\ (\mathsf{neg}\ \ulcorner A \urcorner) \\
= \quad & \mathsf{neg}\ (\mathsf{neg}\ \ulcorner A \urcorner) \\
= \quad & \ulcorner \neg\neg A \urcorner
\end{aligned}
$$

Logical framework LF (Part 2)

Simply-typed

Types	$A, B ::= a \mid A \rightarrow B$
Objects	$M, N ::= x \mid c \mid M\ N \mid \lambda x : A.\ M$
Signatures	$\Sigma ::= \cdot \mid \Sigma, c : A \mid \Sigma, a : \text{type}$
Contexts	$\Gamma ::= \cdot \mid \Gamma, x : A$
Validity	

- Valid types: $\Gamma \vdash A : \text{type}$
- Valid objects: $\Gamma \vdash M : A$

Logical framework LF (Part 2) (cont'd)

Definitional equality

$$(\lambda x : A.\, M)\; N \;=\; [N/x]M \qquad (1)$$
$$(\lambda x : A.\, M\; x) \;=\; M \qquad (2)$$

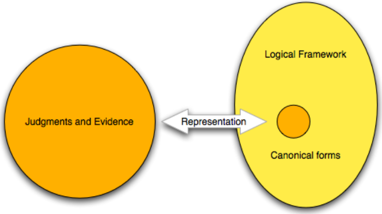
(1) is called β-rule. Does substitutions.

(2) is called η-rule. x not free in M.

Example $\ulcorner [A/p]B \urcorner$

Example $\ulcorner [\mathcal{D} :: A \text{ true}/u](\mathcal{E}(u) :: B \text{ true}) \urcorner$.

Representation Methodology



Property (Subject Reduction) If $\Gamma \vdash M : A$ and $M = N$ then $\Gamma \vdash N : A$.

Property (Subsitution) If $\Gamma, x : A \vdash M : B$ and $\Gamma \vdash N : A$ then $\Gamma \vdash [N/x]M : B$

Definition (Canonicity) An object in β normal η long form is called *canonical*.

Property (Weak normalization) For every object M there exists an canonical object N, such that $M = N$.

Canonical forms (cont'd)

Question Can we write uncountably many functions from wff $\longrightarrow$ wff:

with 1 constructor	$\lambda p : \mathrm{wff}.\, p$
with 2 constructors	$\lambda p : \mathrm{wff}.\, \mathrm{neg}\ p$
with 3 constructors	$\lambda p : \mathrm{wff}.\, \mathrm{neg}\ (\mathrm{neg}\ p)$
	$\lambda p : \mathrm{wff}.\, \mathrm{and}\ p\ p$
	$\lambda p : \mathrm{wff}.\, \mathrm{or}\ p\ p$
	$\lambda p : \mathrm{wff}.\, \mathrm{imp}\ p\ p$
	$\ldots$

Observation No, there are only countably many.

Canonical forms

Judgments

Canonical forms	$\Gamma \vdash M \Uparrow A$
Atomic forms	$\Gamma \vdash N \downarrow A$

Rules

$$\frac{x : A \in \Gamma}{\Gamma \vdash x \downarrow A} \qquad \frac{c : A \in \Sigma}{\Gamma \vdash c \downarrow A} \qquad \frac{\Gamma \vdash M \downarrow A \rightarrow B \qquad \Gamma \vdash N \Uparrow A}{\Gamma \vdash M\ N \downarrow B}$$

$$\frac{\Gamma \vdash M \downarrow a}{\Gamma \vdash M \Uparrow a} \qquad \frac{\Gamma, x : A \vdash M \Uparrow B}{\Gamma \vdash \lambda x : A.\, M \Uparrow A \rightarrow B}$$

Hereditary substitution [Watkins '02]

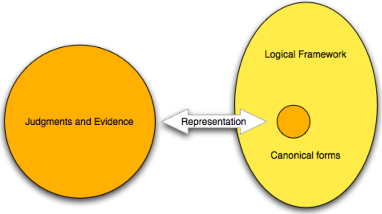
We write M for canonical forms, N for atomic forms, and P for either. We assume that all variable names are renamed away.

Judgments $[M'/x]M = M''$, $[M']N = P''$.

Rules

$$
\begin{array}{rcl}
[M'/x]y & = & y \\
[M'/x]x & = & M' \\
[M'/x](\lambda y : A.\, M) & = & \lambda y : A.\, [M'/x]M \\
[M'/x](N\ M) & = & \begin{cases} [([M'/x]M)/y]M'' & \text{if } [M'/x]N = \lambda y : A.\, M'' \\ N''\ ([M'/x]M) & \text{if } [M'/x]N = N'' \end{cases}
\end{array}
$$

Hereditary substitutions (cont'd)



Observation Hereditary substitutions allows us to stay *tangerine* sets.
We do not consider ill-typed objects,
We do not consider non-canonical objects.
We do not use β reduction for computation.

Conclusion We do use β reduction for substitutions.
Logical frameworks provides syntax for judgments and evidence.
It is a *meta-language* for deductive systems.

Motivation (Part 3)

Representation of schematic judgments

$$\ulcorner A \text{ true} \urcorner = \text{true}$$

$$\text{true} : \text{type}$$

$$\ulcorner \dfrac{\dfrac{\dfrac{}{A \text{ true}}u}{\genfrac{}{}{0pt}{}{\mathcal{D}}{p \text{ true}}}}{\neg A \text{ true}}\,\text{negI}^{p,u} \urcorner = \text{negI}\, \ulcorner A \urcorner\, (\lambda p : \text{wff}.\, \lambda u : \text{true}.\, \ulcorner \mathcal{D} \urcorner)$$

$$\text{negI} : \text{wff} \rightarrow (\text{wff} \rightarrow \text{true} \rightarrow \text{true}) \rightarrow \text{true}$$

Motivation (Part 3)

Problem: Adequacy is broken

$$\text{negI}\ \ulcorner A \urcorner\ (\lambda p : \text{wff}.\ \lambda u : \text{true}.\ u)$$

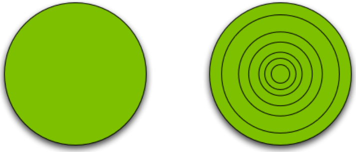
does not correspond to a real derivation.

Thus: $\ulcorner \cdot \urcorner$ is not surjective.

We need to fix this.

Motivation (Part 3)

Central idea: Dependent types.



$$
\begin{array}{rcl}
\ulcorner A \text{ true} \urcorner & = & \text{true} \\
\text{true} & : & \text{wff} \longrightarrow \text{type}
\end{array}
$$

$$
\begin{array}{rcl}
\ulcorner \dfrac{\begin{array}{c}\overline{A \text{ true}}^{\,u} \\ \mathcal{D} \\ p \text{ true}\end{array}}{\neg A \text{ true}}\,\text{negI}^{p,u} \urcorner & = & \text{negI}\ \ulcorner A \urcorner\ (\lambda p : \text{wff}.\ \lambda u : \text{true}\ \ulcorner A \urcorner.\ \ulcorner \mathcal{D} \urcorner) \\
\text{negI} & : & \Pi A : \text{wff}.\ (\Pi p : \text{wff}.\ \text{true}\ A \longrightarrow \text{true}\ p) \\
& & \quad \longrightarrow \text{true}\ (\text{neg}\ A)
\end{array}
$$

Logical framework LF (Part 3)

Dependently-typed

Kinds $K ::= \text{type} \mid A \to K \mid \Pi x : A.\, K$

Types $A, B ::= a \mid A \to B \mid \Pi x : A.\, B$

Objects $M, N ::= x \mid c \mid M\ N \mid \lambda x : A.\, M$

Signatures $\Sigma ::= \cdot \mid \Sigma, c : A \mid \Sigma, a : K$

Contexts $\Gamma ::= \cdot \mid \Gamma, x : A$

Validity

- Valid kinds: $\Gamma \vdash K \Uparrow \text{kind}$
- Valid types: $\Gamma \vdash A \Uparrow K$
- Valid objects: $\Gamma \vdash M \Uparrow A$

Canonical forms

Judgments

$$\begin{array}{rl} \text{Canonical forms} & \Gamma \vdash M \Uparrow A \\ \text{Atomic forms} & \Gamma \vdash N \downarrow A \end{array}$$

Rules

$$\frac{x : A \in \Gamma}{\Gamma \vdash x \downarrow A} \qquad \frac{c : A \in \Sigma}{\Gamma \vdash c \downarrow A} \qquad \frac{\Gamma \vdash M \downarrow \Pi x : A.\, B \qquad \Gamma \vdash N \Uparrow A}{\Gamma \vdash M\ N \downarrow [N/x]B}$$

$$\frac{\Gamma \vdash M \downarrow a}{\Gamma \vdash M \Uparrow a} \qquad \frac{\Gamma, x : A \vdash M \Uparrow B}{\Gamma \vdash \lambda x : A.\, M \Uparrow \Pi x : A.\, B}$$

Example

Implicit arguments

$$\left\ulcorner \dfrac{\begin{array}{cc} \mathcal{D}_1 & \mathcal{D}_2 \\ A \text{ true} & \neg A \text{ true} \end{array}}{B \text{ true}}\,\text{negE} \right\urcorner$$
$$= \quad \text{negE}\,\ulcorner A \urcorner\,\ulcorner B \urcorner\,\ulcorner \mathcal{D}_1 \urcorner\,\ulcorner \mathcal{D}_2 \urcorner$$

and thus

$$\text{neg} : \Pi A : \text{wff}.\, \text{true } A \rightarrow \Pi B : \text{wff}.\, \text{true } (\text{neg } A) \rightarrow \text{true } B$$

we can infer A from from the first argument, so from now on we will abbreviate

$$\text{neg} : \text{true } A \rightarrow \Pi B : \text{wff}.\, \text{true } (\text{neg } A) \rightarrow \text{true } B$$

That's how we implement it in Twelf.

$$\text{neg} : \text{true } A \rightarrow \{B : \text{wff}\}\text{true } (\text{neg } A) \rightarrow \text{true } B$$

Rules for conjunction

$$
\ulcorner \dfrac{\begin{array}{cc} \mathcal{D}_1 & \mathcal{D}_2 \\ A \text{ true} & B \text{ true} \end{array}}{A \wedge B \text{ true}} \text{andI} \urcorner
$$

$$
= \quad \text{andI} \ulcorner A \urcorner \ulcorner B \urcorner \ulcorner \mathcal{D}_1 \urcorner \ulcorner \mathcal{D}_2 \urcorner
$$

$$
\text{andI} : \text{true } A \rightarrow \text{true } B \rightarrow \text{true } (\text{and } A\ B)
$$

$$
\ulcorner \dfrac{\begin{array}{c} \mathcal{D} \\ A \wedge B \text{ true} \end{array}}{A \text{ true}} \text{andE}_1 \urcorner
$$

$$
= \quad \text{andE}_1 \ulcorner A \urcorner \ulcorner B \urcorner \ulcorner \mathcal{D} \urcorner
$$

$$
\text{andE}_1 : \text{true } (\text{and } A\ B) \rightarrow \text{true } A
$$

Rules for disjunction

$$\left\ulcorner \frac{A \text{ true}}{A \vee B \text{ true}}\,\mathsf{orI}_1 \right\urcorner$$

$$= \quad \mathsf{orI}_1\ \ulcorner A \urcorner\ \ulcorner B \urcorner\ \mathsf{true}\ A \to \mathsf{true}\ (\mathsf{or}\ A\ B)$$

$$\mathsf{orI}_1 \quad : \quad \mathsf{true}\ A \to \mathsf{true}\ (\mathsf{or}\ A\ B)$$

$$\left\ulcorner \frac{\begin{array}{c} \mathcal{D} \\ A \vee B \text{ true} \end{array} \quad \begin{array}{c} \overline{A \text{ true}}\,u \\ \mathcal{D}_1 \\ C \text{ true} \end{array} \quad \begin{array}{c} \overline{B \text{ true}}\,v \\ \mathcal{D}_2 \\ C \text{ true} \end{array}}{C \text{ true}}\,\mathsf{orE}^{u,v} \right\urcorner$$

$$= \quad \mathsf{orE}\ \ulcorner A \urcorner\ \ulcorner B \urcorner\ \ulcorner C \urcorner\ \ulcorner \mathcal{D} \urcorner ; \ulcorner \mathcal{D}_1 \urcorner\ \ulcorner \mathcal{D}_2 \urcorner$$

$$\mathsf{orE} \quad : \quad \mathsf{true}\ (\mathsf{or}\ A\ B) \to (\mathsf{true}\ A \to \mathsf{true}\ C) \to (\mathsf{true}\ B \to \mathsf{true}\ C) \to \mathsf{true}\ C$$

Implication

$$
\begin{aligned}
\mathsf{impI} &: (\mathsf{true}\ A \rightarrow \mathsf{true}\ B) \rightarrow \mathsf{true}\ (\mathsf{imp}\ A\ B) \\
\mathsf{impE} &: \mathsf{true}\ (\mathsf{imp}\ A\ B) \rightarrow (\mathsf{true}\ A) \rightarrow (\mathsf{true}\ B)
\end{aligned}
$$

That's the signature you need to feed to Twelf.
Let's look at it!

Example

$$\ulcorner \dfrac{\dfrac{\dfrac{\overline{A\ \text{true}}\,u \qquad \overline{\neg A\ \text{true}}\,v}{p\ \text{true}}\,\text{negE}}{\neg\neg A\ \text{true}}\,\text{impl}^u}{A \supset \neg\neg A\ \text{true}}\,\text{negI}^{p,v} \urcorner$$

$$= \text{negI}\ (\lambda p : \text{wff}.\ \lambda v : \text{true}\ (\text{neg}\ \ulcorner A \urcorner).\ \text{impl}(\lambda u : \text{true}\ \ulcorner A \urcorner.\ \text{negE}\ u\ p\ v))$$

Some syntactic comments on Twelf

- $\Pi x : A. B$ is written as $\{x : A\}B$.
- $\lambda x : A. M$ is written as $[x : A]M$.
- _ is stands for any object.
- Often you can omit type labels, Twelf will infer them.
- Capital letters meta variables.
- The result of type reconstruction: Π closure.

Adequacy

Theorem (Adequacy)

1. If in $p_1 :: \text{wff}, \ldots, p_n :: \text{wff}, u_1 :: A_1 \text{ true}, \ldots, u_m :: A_m \text{ true}$ we can provide evidence $\mathcal{D} :: A \text{ true}$ then there exists one unique M, such that $p_1 : \text{wff}, \ldots, p_n : \text{wff}, u_1 : \text{true} \ulcorner A_1 \urcorner, \ldots, u_m : \text{true} \ulcorner A_m \urcorner \vdash \ulcorner \mathcal{D} \urcorner \Uparrow \text{true} \ulcorner A \urcorner$.
2. If $\mathcal{E} :: p_1 : \text{wff}, \ldots, p_n : \text{wff}, u_1 : \text{true} \ulcorner A_1 \urcorner, \ldots, u_m : \text{true} \ulcorner A_m \urcorner \vdash M \Uparrow \text{true} \ulcorner A \urcorner$ then there exists evidence $\mathcal{D} :: A \text{ true}$ in $p_1 :: \text{wff}, \ldots, p_n :: \text{wff}, u_1 :: A_1 \text{ true}, \ldots, u_m :: A_m \text{ true}$, such that $\ulcorner \mathcal{D} \urcorner = M$.

Adequacy (cont'd)

1. Proof by induction on $\mathcal{D}$.

$$\text{Case}\quad \mathcal{D} = \dfrac{\begin{array}{c}\dfrac{}{A\ \mathsf{true}}\,u \\ \mathcal{D}' \\ p\ \mathsf{true}\end{array}}{\neg A\ \mathsf{true}}\ \mathsf{negI}^{p,u}$$

Assume $p :: \mathsf{wff}$ and $u :: A\ \mathsf{true}$.

$\ldots, p : \mathsf{wff}, u : \mathsf{true}\ \ulcorner A \urcorner \vdash \mathcal{D}' \Uparrow \mathsf{true}\ \ulcorner p \urcorner$ by ind. hyp. on $\mathcal{D}'$

$\ldots, p : \mathsf{wff} \vdash \lambda u : \mathsf{true}\ \ulcorner A \urcorner . \ulcorner \mathcal{D}' \urcorner$
$\Uparrow \mathsf{true}\ \ulcorner A \urcorner \to \mathsf{true}\ \ulcorner p \urcorner$ by canlam

$\cdots \vdash \lambda p : \mathsf{wff}. \lambda u : \mathsf{true}\ \ulcorner A \urcorner . \ulcorner \mathcal{D}' \urcorner$
$\Uparrow \Pi p : \mathsf{wff}. \mathsf{true}\ \ulcorner A \urcorner \to \mathsf{true}\ \ulcorner p \urcorner$ by canlam

$\cdots \vdash \mathsf{negI}\ (\lambda p : \mathsf{wff}. \lambda u : \mathsf{true}\ \ulcorner A \urcorner . \ulcorner \mathcal{D}' \urcorner)$
$\Uparrow \mathsf{true}\ (\mathsf{neg}\ \ulcorner A \urcorner)$ by cansig and atmapp

Adequacy (cont'd)

2. Proof by induction on $\mathcal{E}$.
Omitted (do the negE case as homework).

Conclusion

Conclusion LF, the dependently typed logical framework

One corner of the λ-cube.

No imprepdicativity, no induction principles thus adequate emcondings possible.

Canonical forms inductively defined.

All implemented in the Twelf system.

Homework Complete one case of the adequacy theorem proof for negE in one direction, and negE D_1 $D_2 \Uparrow$ true B in the other.