

# Computer Supported Modeling and Reasoning

---

David Basin, Achim D. Brucker, Jan-Georg Smaus, and  
Burkhardt Wolff

April 2005

<http://www.infsec.ethz.ch/education/permanent/csmr/>

# Metatheory II: Rules

David Basin

ETH Zurich  
15.11.04

## Overview — Proof Rules in the LF

- Last lecture: simple types classify syntax
- Dependent types classify rules:  $pr : o \rightarrow Type$

$$\frac{A \wedge B}{A} \rightsquigarrow \mathit{andel} : \Pi x^o. \Pi y^o. pr(\mathit{and} \ x \ y) \rightarrow pr(x)$$

- Judgements as Types:

$$\frac{\vdots P}{\vdash \phi} \rightsquigarrow \lceil P \rceil \in pr(\lceil \phi \rceil)$$

- Models syntax:  $\phi \in Prop$  iff  $\lceil \phi \rceil \in o$
- Models provability:  $\vdash_L \phi$  iff  $\vdash_{LF} pr(\lceil \phi \rceil)$
- Models proofs:  $P$  iff  $\lceil P \rceil$

## Representing Proofs (cont.)

- Additional levels in the theory:  $\text{Term} \in \text{Type} \in \text{Kind}$

$$pr \in o \rightarrow \text{Type} \in \text{Kind}$$

- Add dependent  $\Pi$ -types:

$$\lambda x^N. x + 1 : \Pi x^N. \{y : N \mid x < y\}$$

- Captures dependencies in proof rules

$$andel : pr \rightarrow pr \text{ becomes } andel : \Pi a^o b^o. pr(andel\ a\ b) \rightarrow pr(a)$$

- Resulting type theory: minimal logic over  $\forall / \Rightarrow$  with  $\omega$ -order quantification

## LF Presentation — Pseudo-Terms

- Let  $Var$  and  $Const$  be sets of **variables** and **constants**.

$$\{Type, Kind\} \in Const$$

- Pseudo-terms**  $\mathcal{T}$

$$\mathcal{T} ::= Var \mid Const \mid \mathcal{T}\mathcal{T} \mid \lambda x^{\mathcal{T}}. \mathcal{T} \mid \Pi x^{\mathcal{T}}. \mathcal{T}$$

- In  $\Pi x^{\tau_1}. \tau_2$ ,  $x$  is bound in  $\tau_2$
- Write  $\Pi x^{\tau_1}. \tau_2$  as  $\tau_1 \rightarrow \tau_2$  when  $x$  not in  $\tau_2$
- Rules** prescribe which pseudo-terms are sensible, e.g., in  $\lambda^{\rightarrow}$ 
  - Some are **terms**:  $\lambda x^N. x$
  - Some are **types**:  $\Pi x^N. N \equiv N \rightarrow N$
  - Some are **nonsense**:  $\Pi x^{\lambda y^N. y}. x$

## LF — Signatures, $\vdash_{sig}$

- Let  $s$  range over sorts  $\{Type, Kind\}$
- Define valid **signature**  $\vdash_{sig}$ , **context**  $\vdash_{con}$ , and **type assignment**  $\vdash_{\Sigma}$
- A **signature**  $\Sigma$  is a sequence ( $c \in Const$ )

$$\Sigma ::= \langle \rangle \mid \Sigma, c : A$$

- Valid signature  $\Sigma$  satisfies  $\vdash_{sig}$  (all constants are well-typed or well-kinded)

$$\vdash_{sig} \langle \rangle \quad \frac{\vdash_{sig} \Sigma \quad \vdash_{\Sigma} A : s \quad c \notin dom(\Sigma)}{\vdash_{sig} \Sigma, c : A}$$

- Example (type environment for constants):

$$\Sigma = \langle o : Type, not : o \rightarrow o, \dots \rangle$$

## LF Context, $\vdash_{con}$

- A **context**  $\Gamma$  is a sequence (for  $x \in Var$ )

$$\Gamma ::= \langle \rangle \mid \Gamma, x : A$$

- $\Gamma$  valid w.r.t.  $\Sigma$  if it satisfies  $\vdash_{con}$  (all variables are well-typed or well-kinded)

$$\vdash_{con} \langle \rangle \quad \frac{\vdash_{con} \Gamma \quad \Gamma \vdash_{\Sigma} A : s \quad x \notin dom(\Gamma)}{\vdash_{con} \Gamma, x : A}$$

- Example ('symbol table for variables'):

$$\Gamma = \langle x : o, y : o, z : o \rightarrow o \rangle$$

## LF Type Assignment Relation, $\vdash_{\Sigma}$

*axiom*  $\Gamma \vdash_{\Sigma} \text{Type} : \text{Kind}$

*assumption*  $\frac{c : A \in \Sigma}{\Gamma \vdash_{\Sigma} c : A}$

*hypothesis*  $\frac{x : A \in \Gamma}{\Gamma \vdash_{\Sigma} x : A}$

*formation*  $\frac{\Gamma \vdash_{\Sigma} A : \text{Type} \quad \Gamma, x : A \vdash_{\Sigma} B : s}{\Gamma \vdash_{\Sigma} \Pi x^A. B : s}$

*abstraction*  $\frac{\Gamma, x : A \vdash_{\Sigma} b : B \quad \Gamma \vdash_{\Sigma} \Pi x^A. B : s}{\Gamma \vdash_{\Sigma} \lambda x^A. b : \Pi x^A. B}$

*application*  $\frac{\Gamma \vdash_{\Sigma} F : \Pi x^A. B \quad \Gamma \vdash_{\Sigma} a : A}{\Gamma \vdash_{\Sigma} F(a) : B[x \leftarrow a]}$



## Restriction of LF rules are $\lambda^{\rightarrow}$ rules

- Restrict typing assertion  $e : B$   
where  $e$  is  $\lambda^{\rightarrow}$  term and  $B$  is simple-type.
- LF rules formalize well-formed signature, context, and type.

$\tau ::= T \mid \tau \rightarrow \tau$  (Type Grammar)       $\Gamma, x : \tau, \Delta \vdash x : \tau$  hyp

$$\frac{\Gamma \vdash e : \sigma \rightarrow \tau \quad \Gamma \vdash e' : \sigma}{\Gamma \vdash ee' : \tau} \text{ app} \qquad \frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash \lambda x^{\sigma}. e : \sigma \rightarrow \tau} \text{ abs}$$

## LF and $\lambda^{\rightarrow}$ — Example

- Signature for logic over arrow (base type **explicit**):

$$\Sigma = \langle o : \mathit{Type}, \mathit{not} : o \rightarrow o, \mathit{imp} : o \rightarrow o \rightarrow o \rangle,$$

- Signature well-formed

$$\vdash_{\langle \rangle} \mathit{Type} : \mathit{Kind}$$

and

$$\frac{\frac{o : \mathit{Type} \in \langle o : \mathit{Type} \rangle}{\vdash_{\langle o : \mathit{Type} \rangle} o : \mathit{Type}} \quad \frac{o : \mathit{Type} \in \langle o : \mathit{Type} \rangle}{x : o \vdash_{\langle o : \mathit{Type} \rangle} o : \mathit{Type}}}{\vdash_{\langle o : \mathit{Type} \rangle} o \rightarrow o : \mathit{Type}}$$

similar for type of *imp*

## (Full) LF Type Theory

- Additional rules add power, e.g., formation (here  $s = Kind$ )

$$\frac{\Gamma \vdash_{\Sigma} A : Type \quad \Gamma, x : A \vdash_{\Sigma} B : Kind}{\Gamma \vdash_{\Sigma} \Pi x^A. B : Kind}$$

$$\Sigma_{prop} = \langle o : Type, pr : o \rightarrow Type, not : o \rightarrow o, \\ imp : o \rightarrow o \rightarrow o, and : o \rightarrow o \rightarrow o \rangle$$

- Type-valued functions in context are now well-typed

$$\frac{o : Type \vdash_{\Sigma} o : Type \quad o : Type, x : o \vdash_{\Sigma} Type : Kind}{o : Type \vdash_{\Sigma} o \rightarrow Type : Kind}$$

- Moreover, dependent types not possible in  $\lambda^{\rightarrow}$

# Example of Dependent Types — Rule Formalization

- Consider  $andel : \Pi x^o. \Pi y^o. pr(and\ x\ y) \rightarrow pr(x)$
- Part I

$$\frac{x : o, y : o \vdash_{\Sigma} and : o \rightarrow o \rightarrow o \quad x : o, y : o \vdash_{\Sigma} x : o}{x : o, y : o \vdash_{\Sigma} and\ x : o \rightarrow o} \text{app}$$

$$\frac{x : o, y : o \vdash_{\Sigma} pr : o \rightarrow Type \quad x : o, y : o \vdash_{\Sigma} and\ x\ y : o}{x : o, y : o \vdash_{\Sigma} pr(and\ x\ y) : Type} \text{app}$$

- Part II

$$\frac{x : o, y : o, z : pr(x) \vdash_{\Sigma} pr : o \rightarrow Type \quad x : o, y : o, z : pr(x) \vdash_{\Sigma} x : o}{x : o, y : o, z : pr(x) \vdash_{\Sigma} pr(x) : Type} \text{app}$$

$$\frac{\text{(Part I)} \quad x : o, y : o, z : pr(x) \vdash_{\Sigma} pr(x) : Type}{x : o, y : o \vdash_{\Sigma} pr(and\ x\ y) \rightarrow pr(x) : Type} \text{form}$$

$$\frac{x : o \vdash_{\Sigma} o : Type \quad x : o, y : o \vdash_{\Sigma} pr(and\ x\ y) \rightarrow pr(x) : Type}{x : o \vdash_{\Sigma} \Pi y^o. pr(and\ x\ y) \rightarrow pr(x) : Type} \text{form}$$

$$\frac{\vdash_{\Sigma} o : Type \quad x : o \vdash_{\Sigma} \Pi y^o. pr(and\ x\ y) \rightarrow pr(x) : Type}{\vdash_{\Sigma} \Pi x^o. \Pi y^o. pr(and\ x\ y) \rightarrow pr(x) : Type} \text{form}$$

## Metatheoretic Properties of LF

- $\forall/\Rightarrow$  fragment of intuitionistic  $\omega$ -order logic

$$\begin{array}{c} \prod x^o. \prod y^o. pr(\text{and } x \ y) \rightarrow pr(x) \\ \Downarrow \\ \forall x^o. \forall y^o. pr(\text{and } x \ y) \Rightarrow pr(x) \end{array}$$

- Unique normal forms and subject reduction  
Faithfulness/adequacy by analysis of normal forms
- $\Gamma \vdash_{\Sigma} t : \tau$  is decidable.
  - Well-formedness is decidable:  $\tau : \textit{Type}$  or  $\tau : \textit{Kind}$
  - Proof checking is decidable:  $t : \tau$

## Idea of LF Encodings

Rules organized around basic assertions (judgements)

- $pr(p)$ :  $p$  is provable
  - $valid(p)$ :  $p$  is true in all models
  - $true(p)$ :  $p$  is true in some model
  - $acc(w_1, w_2)$ : ‘world’  $w_1$  accesses  $w_2$
  - Multiple judgements, e.g. type theory and modal logics
- + higher-order judgements: ( $J_i$  judgement,  $C$  syntactic category)
- Schematic:  $\bigwedge_{x \in C} J(x)$
  - Hypothetical:  $J_1 \vdash_{\Sigma} J_2$  expresses consequence and allows hybrid notions of consequence

$$valid(p) \vdash_{\Sigma} true(p)$$

## Idea of LF Encodings: *Judgements as Types*

- Basic judgements associated with type families,  
e.g.  $pr : o \rightarrow Type$

$$\overline{p \text{ provable}} \rightsquigarrow pr(p)$$

- Higher-order forms available

$$\overline{\bigwedge_{x \in C} J(x)} \rightsquigarrow \Pi x^{\overline{C}}. \overline{J(x)}$$

$$\overline{J_1 \vdash J_2} \rightsquigarrow \overline{J_1} \rightarrow \overline{J_2}$$

- Truth = inhabitation

$t : pr(p)$  iff  $t$  represents proof that  $p$  is provable



## Example: Propositional Logic

- Formalize language ....

$$P ::= x \mid P \wedge P \mid P \Rightarrow P \dots$$

- ... and proof rules

$$\frac{A \wedge B}{A} \wedge\text{-}EL \quad \frac{A \wedge B}{B} \wedge\text{-}ER \quad \frac{A \quad B}{A \wedge B} \wedge\text{-}I$$

$$\frac{A \Rightarrow B \quad A}{B} \Rightarrow\text{-}E$$

$$\frac{\begin{array}{c} [A] \\ \vdots \\ B \end{array}}{A \Rightarrow B} \Rightarrow\text{-}I$$

## Propositional Logic in LF

- $\Sigma$  contains syntactic categories

$$o : Type$$

- and judgements

$$pr : o \rightarrow Type$$

- and operators (declaring syntax)

$$imp : o \rightarrow o \rightarrow o, \text{ and } : o \rightarrow o \rightarrow o, \dots$$

- and rules (next page)

## Propositional Logic in LF

- and-rules (and impe) are 1st-order judgements

$$\frac{A \wedge B}{A} \rightsquigarrow \Pi a^o b^o. pr(\text{and } a \ b) \rightarrow pr(a)$$

- $\Rightarrow$ -I is 2nd-order judgement

$$\frac{\begin{array}{c} [A] \\ \vdots \\ B \end{array}}{A \Rightarrow B} \Rightarrow\text{-I} \rightsquigarrow \Pi a^o b^o. (pr(a) \rightarrow pr(b)) \rightarrow pr(\text{imp } a \ b)$$

- Consequence over *hypothetical proofs*:  $(A \vdash B) \vdash A \Rightarrow B$
- Shifts assumption discharging to LF

$$\frac{x : pr(a) \vdash_{\Sigma} \dots : pr(b)}{\vdash_{\Sigma} \dots : pr(a) \rightarrow pr(b)} \text{abs}_{T,T}$$

$$\frac{\vdash_{\Sigma} \dots : pr(a) \rightarrow pr(b)}{\vdash_{\Sigma} \dots : pr(\text{imp } a \ b)} \text{impi}^*$$

## Propositional Logic — More Rules

$$\frac{A}{A \vee B} \vee\text{-}IL \quad \frac{B}{A \vee B} \vee\text{-}IR \quad \frac{A \vee B \quad \begin{array}{c} [A] \\ \vdots \\ C \end{array} \quad \begin{array}{c} [B] \\ \vdots \\ C \end{array}}{C} \vee\text{-}E$$

- Augment operators:  $or : o \rightarrow o \rightarrow o$

- Augment rules:

$$oril : \Pi a^o b^o. pr(a) \rightarrow pr(or\ a\ b)$$

$$orir : \Pi a^o b^o. pr(b) \rightarrow pr(or\ a\ b)$$

$$ore : \Pi a^o b^o c^o. pr(or\ a\ b) \rightarrow (pr(a) \rightarrow pr(c)) \\ \rightarrow (pr(b) \rightarrow pr(c)) \rightarrow pr(c)$$

## Propositional Proofs: $a \Rightarrow a$

Part I:

$$\begin{array}{c}
 a : o \vdash_{\Sigma} \text{impi} : \Pi a^o b^o. (pr(a) \rightarrow pr(b)) \rightarrow pr(\text{imp } a \ b) \quad a : o \vdash_{\Sigma} a : o \\
 \hline
 a : o \vdash_{\Sigma} \text{impi } a : \Pi b^o. (pr(a) \rightarrow pr(b)) \rightarrow pr(\text{imp } a \ b) \quad a : o \vdash_{\Sigma} a : o \\
 \hline
 a : o \vdash_{\Sigma} \text{impi } a \ a : (pr(a) \rightarrow pr(a)) \rightarrow pr(\text{imp } a \ a)
 \end{array}$$

Part II:

$$\begin{array}{c}
 a : o, y : pr(a) \vdash_{\Sigma} y : pr(a) \\
 \hline
 a : o \vdash_{\Sigma} \lambda y^{pr(a)}. y : pr(a) \rightarrow pr(a)
 \end{array}$$

Part III:

$$\begin{array}{c}
 \text{(Part I)} \quad \text{(Part II)} \\
 \hline
 a : o \vdash_{\Sigma} \text{impi } a \ a (\lambda y^{pr(a)}. y) : pr(\text{imp } a \ a) \\
 \hline
 \vdash_{\Sigma} \lambda a^o. \text{impi } a \ a (\lambda y^{pr(a)}. y) : \Pi a^o. pr(\text{imp } a \ a)
 \end{array}$$

## LF — Aside on System Support

Proof detail impractical. Implementations support:

- Implicit quantification/type reconstruction

$$\lambda a^o. \text{impi } a \ a \ (\lambda y^{pr(a)}. y) : \Pi a^o. pr(\text{imp } a \ a)$$

- Automatic type checking
- “Resolution” and refinement style proof.

$$\frac{\Gamma \vdash_{\Sigma} \dots : pr(\text{imp } a \ a)}{\Gamma, x : pr(a) \vdash_{\Sigma} \dots : pr(a)} \text{impi}^*$$

- Term synthesis via unification

$$\Gamma \vdash_{\Sigma} ?t : pr(\text{imp } a \ a)$$

- Pretty printing, implicit contexts, and implicit judgement coercions

$$\frac{\vdash a \Rightarrow a}{a \vdash a}$$

- Tactics

## LF — Terms Encode Proofs

- $a \Rightarrow a \rightsquigarrow \Pi a^o. pr(impa a)$

$$\lambda a^o. impia a (\lambda y^{pr(a)}. y)$$

$$\frac{[A]^y}{A \Rightarrow A} \Rightarrow -I^y$$

- $a \Rightarrow b \Rightarrow a \rightsquigarrow \Pi a^o b^o. pr(impa (impba))$

$$\lambda a^o. \lambda b^o. impia (impba) (\lambda x^{pr(a)}. impiba (\lambda y^{pr(b)}. x))$$

$$\frac{\frac{[A]^x}{B \Rightarrow A} \Rightarrow -I^x}{A \Rightarrow (B \Rightarrow A)} \Rightarrow -I^x$$

- Adequacy & faithfulness: There exists a bijection between proofs of  $\phi \in Prop$  and canonical members of  $pr(\phi)$

## Schematic Judgements — F.O. Arithmetic Example

- Review: syntax

$$\begin{array}{l}
 \text{Terms } T \quad ::= x \mid 0 \mid sT \mid T + T \mid T \times T \\
 \text{Formulae } F \quad ::= t = t \mid \neg F \mid F \wedge F \mid \dots \\
 \quad \quad \quad \quad \quad \forall x. A \mid \exists x. A
 \end{array}$$

- Type declarations for context

$$B = i : \text{Type}, o : \text{Type}$$

- Context  $\Gamma = \langle B, \Gamma_T, \Gamma_P, \Gamma_Q \rangle$ :

$$\Gamma_T = 0 : i, s : i \rightarrow i, \text{plus} : i \rightarrow i \rightarrow i, \text{times} : i \rightarrow i \rightarrow i$$

$$\Gamma_P = \text{eq} : i \rightarrow i \rightarrow o, \text{not} : o \rightarrow o, \text{and} : o \rightarrow o \rightarrow o, \dots$$

$$\Gamma_Q = \text{all} : (i \rightarrow o) \rightarrow o, \text{exists} : (i \rightarrow o) \rightarrow o$$



## FO Arithmetic — $\forall$ -Quantifier Rules

$$\frac{A}{\forall x. A} \forall-I^* \quad \frac{\forall x. A}{A[x \leftarrow t]} \forall-E$$

\*:  $x$  not free in any undischarged assumptions on which  $A$  depends

$$alli : \Pi A^{i \rightarrow o}. (\Pi x^i. pr(A(x))) \rightarrow pr(all(A))$$

$$alle : \Pi A^{i \rightarrow o} x^i. pr(all(A)) \rightarrow pr(A(x))$$

## Side Conditions Captured via $\Pi$ -binding

Prove  $(\forall x. A \Rightarrow B(x)) \Rightarrow (A \Rightarrow \forall x. B(x))$  in context  $\Gamma, A : o, B : i \rightarrow o, x : i$

Part I:

$$\frac{pr(all(\lambda x^i. A \Rightarrow B(x))), pr(A) \vdash_{\Sigma} pr(all(\lambda x^i. A \Rightarrow B(x)))}{pr(all(\lambda x^i. A \Rightarrow B(x))), pr(A) \vdash_{\Sigma} pr(A \Rightarrow B(x))} \textit{alle}$$

Part II:

$$\frac{\frac{\text{(Part I)} \quad pr(all(\lambda x^i. A \Rightarrow B(x))), pr(A) \vdash_{\Sigma} pr(A)}{pr(all(\lambda x^i. A \Rightarrow B(x))), pr(A) \vdash_{\Sigma} pr(B(x))} \textit{impe}}{pr(all(\lambda x^i. (A \Rightarrow B(x)))), pr(A) \vdash_{\Sigma} \Pi x^i. pr(B(x))} \textit{abs (discharge } x : i)}$$

$$\frac{\frac{\frac{pr(all(\lambda x^i. (A \Rightarrow B(x)))), pr(A) \vdash_{\Sigma} \Pi x^i. pr(B(x))}{pr(all(\lambda x^i. (A \Rightarrow B(x))), pr(A) \vdash_{\Sigma} pr(all(\lambda x^i. B(x))))} \textit{alli}}{pr(all(\lambda x^i. (A \Rightarrow B(x))), pr(A) \vdash_{\Sigma} pr(all(\lambda x^i. B(x))))} \textit{impi}}{pr(all(\lambda x^i. (A \Rightarrow B(x)))) \vdash_{\Sigma} pr(A \Rightarrow all(\lambda x^i. B(x)))} \textit{impi}}{pr(all(\lambda x^i. (A \Rightarrow B(x))) \Rightarrow (A \Rightarrow all(\lambda x^i. B(x))))} \textit{impi}$$

## FO Arithmetic — Substitution

$$\frac{t = u \quad A[t]}{A[u]} \text{ sub} \quad \frac{a = b \quad P(a, a, a)}{P(a, b, a)}$$

$A[t]$  means some distinguished (not necessarily all) occurrences of  $t$  in  $A$ . In above,  $A = \lambda x. P(a, x, a)$

$$\text{sub} : \prod t^i u^i A^{i \rightarrow o}. \text{pr}(eq \ t \ u) \rightarrow \text{pr}(A(t)) \rightarrow \text{pr}(A(u))$$

## Aside on Derived Rules

- Encoding of FOA adequate and faithful

But more than FOA sentences are expressible/derivable!

- Rules are derivable

$$\begin{array}{c}
 [A[x \leftarrow t]] \\
 \vdots \\
 B \\
 \hline
 \forall x. A \quad B \\
 B \quad \text{alle}_{SH}
 \end{array}$$

$$\text{alle}_{SH} : \Pi A^{i \rightarrow o} B^o. \text{pr}(\text{all}(A)) \rightarrow ((\Pi x^i. \text{pr}(A(x))) \rightarrow \text{pr}(B)) \rightarrow \text{pr}(B)$$

- Inhabitants are proof-construction functions.  $\text{alle}_{SH} =$

$$\lambda A^{i \rightarrow o} B^o. p^{\text{pr}(\text{all}(A))} q^{\Pi x^i. \text{pr}(A(x)) \rightarrow \text{pr}(B)}. q(\lambda x^i. \text{alle}(\lambda x^i. B(x))(x)(p))$$

- Can verify all *derivable* rules.

Judgements are open: no induction over encoded theory

## Summary of Proof/Rule Encoding

Object Language	Meta Language
Judgement $\vdash_{\Sigma} p$	Type Family $pr : o \rightarrow Type$
Inference Rule $\frac{A \wedge B}{A} \wedge\text{-}EL$	Constant Declaration $andel : \Pi a^o b^o. pr(andel\ a\ b) \rightarrow pr(a)$
Deduction	Typing Proof
Deductive System	Context Declaration

## Advantages/Disadvantages of LF

- + High-level representation: HOAS and assumption management for ND
- + Judgements as Types: Proof construction and adequacy arguments
- + Good metatheoretic properties
  - + Decidable type checking  $\Gamma \vdash_{\Sigma} t : \tau$
- +/- Supports some metatheoretic extensibility
  - No rules by induction over theory
  - + Derived rules hold in theory extensions
- +/- Biased towards certain logics:  $\vdash_L$  shares properties with  $\vdash_{LF}$

## Relation to Type Theories based on HOL (Isabelle)

- $\lambda^{\rightarrow}$  used to declare syntax

$$\Gamma_{prop} = \langle o : Type, pr : o \rightarrow Prop, not : o \rightarrow o, \\ imp : o \rightarrow o \rightarrow o, and : o \rightarrow o \rightarrow o \rangle$$

- $\forall/\Rightarrow$  fragment of HOL used to declare axioms

$$andel : \Pi x^o. \Pi y^o. pr(and\ x\ y) \rightarrow pr(x) \\ \downarrow \\ andel : \forall x^o. \forall y^o. pr(and\ x\ y) \Rightarrow pr(x)$$

- Formalizations similar, but no Judgements as Types
- Adequacy/faithfulness by proof (not term) normalization

## Further Reading

- Barendregt, “Introduction to Generalized Type Systems”, Journal of Functional Programming, April 1991.
- Harper, Honsell, and Plotkin, “A Framework for Defining Logics”, JACM, January 1993.
- Paulson, “Isabelle: A Generic Theorem Prover”, Springer LNCS 828.