

Induction and Cut

Chad E. Brown

`cebrown@ags.uni-sb.de`

Universität des Saarlandes, Saarbrücken, Germany

Using Cut

1. $Q A$

2. $\forall x. Q x \supset Q [F x]$

Goal: $Q . \underbrace{F \dots F}_{256} 0$

Proof 1 (cut-free): 256 applications of (2); 1 application of (1)

Using Cut

1. $Q A$

2. $\forall x.Q x \supset Q [F x]$

Goal: $Q . \underbrace{F \dots F}_{256} 0$

Proof 2 (cut): Show 8 Lemmas:

Lemma 1: $\forall x.Q x \supset Q . F . F x \quad \dots$

Lemma n: $\forall x.Q x \supset Q . \underbrace{F \dots F}_{2^n} x$

Using Lemmas as cut-formulas, the proof uses 2 applications of (2) (to show Lemma 1); 1 application of (1) to Lemma 8. (Lemma n is used twice to show Lemma $n + 1$.)

Using Cut

Cut leads to shorter proofs, but search for proofs with cut are harder to mechanize.

Where did the 8 lemmas come from?

Simulating Cut by Induction

With induction, we can simulate the proof with cut.

1. $Q A$
2. $\forall x. Q x \supset Q [F x]$

Plus Induction (*IND*)

$$(IND) \quad \forall P. [P 0 \wedge \forall n [P n \supset P [S n]]] \supset \forall m . P m$$

Suppose we want a cut-free proof of

$$(IND) \wedge (1) \wedge (2) \supset Q . \underbrace{F \dots F}_{256} 0$$

Simulating Cut by Induction

With induction, we can simulate the proof with cut.

$$(IND) \quad \forall P. [P 0 \wedge \forall n [P n \supset P [S n]]] \supset \forall m . P m$$

$$(IND) \wedge (1) \wedge (2) \supset Q . \underbrace{F \dots F}_2 0$$

For each lemma L^n we can use the induction assumption with the “set” $\{n | \mathbf{L}\}$. This set does *not* depend on n . We obtain (deleting vacuous quantifiers):

$$[\mathbf{L}^n \wedge [\mathbf{L}^n \supset \mathbf{L}^n]] \supset \mathbf{L}^n$$

Introduces lemmas via (abuse of) induction axiom

Simple Types

Simple Types \mathcal{T} :

- o (truth values)
- ι (individuals)
- $(\alpha\beta)$ (functions from β to α)

Simple Types

Simple Types \mathcal{T} :

o	(truth values)
ι	(individuals)
$(\alpha\beta)$	(functions from β to α)

$(\alpha\beta\gamma)$ abbreviates $((\alpha\beta)\gamma)$

Simply Typed λ -Terms

Terms:

x_α	Variables (\mathcal{V})
A_α	Parameters (\mathcal{P})
c_α	Logical Constants (\mathcal{S})
$[\mathbf{F}_{\alpha\beta} \mathbf{B}_\beta]_\alpha$	Application
$[\lambda y_\beta \mathbf{A}_\alpha]_{\alpha\beta}$	λ -abstraction

Simply Typed λ -Terms

Terms:

x_α	Variables (\mathcal{V})
A_α	Parameters (\mathcal{P})
c_α	Logical Constants (\mathcal{S})
$[\mathbf{F}_{\alpha\beta} \mathbf{B}_\beta]_\alpha$	Application
$[\lambda y_\beta \mathbf{A}_\alpha]_{\alpha\beta}$	λ -abstraction

Equality of terms: $\alpha\beta\eta$

Simply Typed λ -Terms

Terms:	x_α	Variables (\mathcal{V})
	A_α	Parameters (\mathcal{P})
	c_α	Logical Constants (\mathcal{S})
	$[\mathbf{F}_{\alpha\beta} \mathbf{B}_\beta]_\alpha$	Application
	$[\lambda y_\beta \mathbf{A}_\alpha]_{\alpha\beta}$	λ -abstraction

Equality of terms: $\alpha\beta\eta$

α -conversion Changing Bound Variables

β -reduction $[[\lambda y_\beta \mathbf{A}_\alpha] \mathbf{B}] \xrightarrow{\beta} [\mathbf{B}/y] \mathbf{A}$

η -reduction $[\lambda y_\beta [\mathbf{F}_{\alpha\beta} y]] \xrightarrow{\eta} \mathbf{F} \quad (y_\beta \notin \mathbf{Free}(\mathbf{F}))$

Simply Typed λ -Terms

Terms:

x_α	Variables (\mathcal{V})
A_α	Parameters (\mathcal{P})
c_α	Logical Constants (\mathcal{S})
$[\mathbf{F}_{\alpha\beta} \mathbf{B}_\beta]_\alpha$	Application
$[\lambda y_\beta \mathbf{A}_\alpha]_{\alpha\beta}$	λ -abstraction

Equality of terms: $\alpha\beta\eta$

Every term has a unique $\beta\eta$ -normal form,
up to α -conversion.

Logical Constants

\neg_{oo} – negation

\vee_{ooo} – disjunction

$\prod_{o(o\alpha)}^\alpha$ – universal quantification over type α

Abbreviations for Logical Operators

$[A_o \vee B_o]$ means $[\vee_{ooo} A_o B_o]$

$[A_o \supset B_o]$ means $[\neg A_o \vee B_o]$

$[\forall x_\alpha A_o]$ means $[\Pi_{o(o\alpha)}^\alpha \cdot \lambda x_\alpha A_o]$.

$[\exists x_\alpha A_o]$ means $[\neg \forall x_\alpha \neg A_o]$.

Church's Type Theory

Church's Type Theory:

- Simply typed λ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

Church's Type Theory

Church's Type Theory:

- Simply typed λ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

- Axioms of Extensionality

Church's Type Theory

Church's Type Theory:

- Simply typed λ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

- Axioms of Extensionality
- Axiom of Description or Choice

Church's Type Theory

Church's Type Theory:

- Simply typed λ -calculus with the signature

$$\{\neg, \vee\} \cup \{\Pi^\alpha \mid \alpha \in \mathcal{T}\}$$

(and perhaps a description or choice operator).

- Axioms of Extensionality
- Axiom of Description or Choice
- Axiom of Infinity

Elementary Type Theory

Fragment of Church's Type Theory:

- Simply typed λ -calculus

Elementary Type Theory

Fragment of Church's Type Theory:

- Simply typed λ -calculus
- No Extensionality

Elementary Type Theory

Fragment of Church's Type Theory:

- Simply typed λ -calculus
- No Extensionality
- No Axiom of Description or Choice

Elementary Type Theory

Fragment of Church's Type Theory:

- Simply typed λ -calculus
- No Extensionality
- No Axiom of Description or Choice
- No Axiom of Infinity

Sequents

Sequents = Finite Multiset of Sentences

$$M^1, \dots, M^n$$

Sequents

Sequents = Finite Multiset of Sentences

$$M^1, \dots, M^n$$

Intuitively,

$$M^1 \vee \dots \vee M^n$$

Sequents

Sequents = Finite Multiset of Sentences

$$M^1, \dots, M^n$$

Intuitively,

$$M^1 \vee \dots \vee M^n$$

Next: Rules for Deriving Sequents
in Elementary Type Theory

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init)$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init)$$

$$\frac{}{\Gamma, \mathbf{M}}$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \qquad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr)$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \quad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \quad \frac{}{\Gamma, \mathbf{M}}$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \quad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \quad \frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta)$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init)$$

$$\frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr)$$

$$\frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta)$$

$$\frac{\Gamma, \mathbf{M}}{\Gamma, \neg \neg \mathbf{M}} \mathcal{G}(\neg \neg)$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \quad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \quad \frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta)$$

$$\frac{\Gamma, \mathbf{M}}{\Gamma, \neg \neg \mathbf{M}} \mathcal{G}(\neg \neg) \quad \frac{}{\Gamma, [\mathbf{M} \vee \mathbf{N}]}$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \quad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \quad \frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta)$$

$$\frac{\Gamma, \mathbf{M}}{\Gamma, \neg \neg \mathbf{M}} \mathcal{G}(\neg \neg) \quad \frac{\Gamma, \mathbf{M}, \mathbf{N}}{\Gamma, [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\vee)$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \quad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \quad \frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta)$$

$$\frac{\Gamma, \mathbf{M}}{\Gamma, \neg \neg \mathbf{M}} \mathcal{G}(\neg \neg) \quad \frac{\Gamma, \mathbf{M}, \mathbf{N}}{\Gamma, [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\vee) \quad \frac{}{\Gamma, \neg [\mathbf{M} \vee \mathbf{N}]}$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \quad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \quad \frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta)$$
$$\frac{\Gamma, \mathbf{M}}{\Gamma, \neg \neg \mathbf{M}} \mathcal{G}(\neg\neg) \quad \frac{\Gamma, \mathbf{M}, \mathbf{N}}{\Gamma, [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\vee) \quad \frac{\Gamma, \neg \mathbf{M} \quad \Gamma, \neg \mathbf{N}}{\Gamma, \neg [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\neg\vee)$$

Sequent Calculus

$$\frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \quad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \quad \frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta)$$
$$\frac{\Gamma, \mathbf{M}}{\Gamma, \neg \neg \mathbf{M}} \mathcal{G}(\neg\neg) \quad \frac{\Gamma, \mathbf{M}, \mathbf{N}}{\Gamma, [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\vee) \quad \frac{\Gamma, \neg \mathbf{M} \quad \Gamma, \neg \mathbf{N}}{\Gamma, \neg [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\neg\vee)$$

$$\Gamma, [\forall x_\alpha \mathbf{M}]$$

Sequent Calculus

$$\begin{array}{c} \frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \quad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \quad \frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta) \\ \\ \frac{\Gamma, \mathbf{M}}{\Gamma, \neg \neg \mathbf{M}} \mathcal{G}(\neg\neg) \quad \frac{\Gamma, \mathbf{M}, \mathbf{N}}{\Gamma, [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\vee) \quad \frac{\Gamma, \neg \mathbf{M} \quad \Gamma, \neg \mathbf{N}}{\Gamma, \neg [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\neg\vee) \\ \\ \frac{\Gamma, [[W/x]\mathbf{M}] \quad W \in \mathcal{P}_\alpha \setminus \mathbf{Params}(\Gamma, [\forall x_\alpha \mathbf{M}])}{\Gamma, [\forall x_\alpha \mathbf{M}]} \mathcal{G}(\forall^W) \end{array}$$

Sequent Calculus

$$\begin{array}{c} \frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \quad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \quad \frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta) \\ \\ \frac{\Gamma, \mathbf{M}}{\Gamma, \neg \neg \mathbf{M}} \mathcal{G}(\neg\neg) \quad \frac{\Gamma, \mathbf{M}, \mathbf{N}}{\Gamma, [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\vee) \quad \frac{\Gamma, \neg \mathbf{M} \quad \Gamma, \neg \mathbf{N}}{\Gamma, \neg [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\neg\vee) \\ \\ \frac{\Gamma, [[W/x]\mathbf{M}] \quad W \in \mathcal{P}_\alpha \setminus \mathbf{Params}(\Gamma, [\forall x_\alpha \mathbf{M}])}{\Gamma, [\forall x_\alpha \mathbf{M}]} \mathcal{G}(\forall^W) \\ \\ \hline \Gamma, \neg [\forall x_\alpha \mathbf{M}] \end{array}$$

Sequent Calculus

$$\begin{array}{c} \frac{}{\Gamma, \mathbf{M}, \neg \mathbf{M}} \mathcal{G}(Init) \qquad \frac{\Gamma, \mathbf{M}, \mathbf{M}}{\Gamma, \mathbf{M}} \mathcal{G}(Contr) \qquad \frac{\Gamma, \mathbf{M}^\downarrow}{\Gamma, \mathbf{M}} \mathcal{G}(\beta\eta) \\ \\ \frac{\Gamma, \mathbf{M}}{\Gamma, \neg \neg \mathbf{M}} \mathcal{G}(\neg \neg) \qquad \frac{\Gamma, \mathbf{M}, \mathbf{N}}{\Gamma, [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\vee) \qquad \frac{\Gamma, \neg \mathbf{M} \quad \Gamma, \neg \mathbf{N}}{\Gamma, \neg [\mathbf{M} \vee \mathbf{N}]} \mathcal{G}(\neg \vee) \\ \\ \frac{\Gamma, [[W/x]\mathbf{M}] \quad W \in \mathcal{P}_\alpha \setminus \mathbf{Params}(\Gamma, [\forall x_\alpha \mathbf{M}])}{\Gamma, [\forall x_\alpha \mathbf{M}]} \mathcal{G}(\forall^W) \\ \\ \frac{\Gamma, \neg [[\mathbf{C}/x]\mathbf{M}] \quad \mathbf{C} \in \mathit{cwff}_\alpha(\mathcal{S})}{\Gamma, \neg [\forall x_\alpha \mathbf{M}]} \mathcal{G}(\neg \forall) \end{array}$$

Cut-Elimination

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- Let \mathcal{G} be the sequent calculus without cut.
- Let \mathcal{G}^+ be the sequent calculus with cut.

Cut-Elimination

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- Let \mathcal{G} be the sequent calculus without cut.
- Let \mathcal{G}^+ be the sequent calculus with cut.
- Andrews proved cut-elimination for \mathcal{G} (essentially).
- If a sequent Γ has a derivation in \mathcal{G}^+ (with cut), then Γ has a derivation in \mathcal{G} (without cut).

Cut-Elimination

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- Let \mathcal{G} be the sequent calculus without cut.
- Let \mathcal{G}^+ be the sequent calculus with cut.
- In general, any proof without cut may be much larger than a proof with cut.

Cut-Elimination

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- Let \mathcal{G} be the sequent calculus without cut.
- Let \mathcal{G}^+ be the sequent calculus with cut.
- In general, any proof without cut may be much larger than a proof with cut.
- In first-order, the blow-up of cut-elimination is hyperexponential.

Cut-Elimination

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- Let \mathcal{G} be the sequent calculus without cut.
- Let \mathcal{G}^+ be the sequent calculus with cut.
- In general, any proof without cut may be much larger than a proof with cut.
- In first-order, the blow-up of cut-elimination is hyperexponential.
- In higher-order, the blow-up of cut-elimination is much worse (more than Ackermann).

Cut-Free Proofs

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- Q: Why do we search for cut-free proofs?

Cut-Free Proofs

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- Q: Why do we search for cut-free proofs?
- A: To avoid searching for the cut formula C .

Cut-Free Proofs

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- Q: Why do we search for cut-free proofs?
- A: To avoid searching for the cut formula C .
- Q: What is the cost?

Cut-Free Proofs

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- Q: Why do we search for cut-free proofs?
- A: To avoid searching for the cut formula C .
- Q: What is the cost?
- A: Some theorems have short proofs with cut but no short cut-free proof.

Cut-Free Proofs

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- In first-order logic, cut-free proofs satisfy a “subformula property”.

Cut-Free Proofs

$$\frac{\Gamma, C \quad \Gamma, \neg C}{\Gamma} \textit{Cut}$$

- In first-order logic, cut-free proofs satisfy a “subformula property”.
- In higher-order logic, such a “subformula property” is less restrictive since any formula is a subformula of $\forall P_o P$.

Induction

- Let \mathbb{N} be a type for natural numbers.

Induction

- Let \mathbf{N} be a type for natural numbers.
- Induction:

$$(IND) \quad \forall P_{o\mathbf{N}}. [P\ 0_{\mathbf{N}} \wedge \forall n_{\mathbf{N}} [P\ n \supset P\ [S\ n]]] \supset \forall m_{\mathbf{N}}. P\ m$$

Induction

- Let \mathbf{N} be a type for natural numbers.

- Induction:

$$(IND) \quad \forall P_{o\mathbf{N}}. [P\ 0_{\mathbf{N}} \wedge \forall n_{\mathbf{N}} [P\ n \supset P\ [S\ n]]] \supset \forall m_{\mathbf{N}}. P\ m$$

- Any term A_{α} is a “subformula” of (IND) since we can substitute $[\lambda n_{\mathbf{N}}. Q_{o\alpha} A]$ for P .

Original Problem

1. $Q_{oi} A_i$
2. $\forall x_i. Q x \supset Q [F_{ii} x]$

$$(IND) \quad \forall P_{oN}. [P 0_N \wedge \forall n_N [P n \supset P [S n]]] \supset \forall m_N. P m$$

Sequent:

$$\neg(IND), \neg(1), \neg(2), Q . \underbrace{F \dots F}_2 0$$

Original Problem

1. $Q_{o\iota} A_{\iota}$
2. $\forall x_{\iota}. Q x \supset Q [F_{\iota\iota} x]$

$$(IND) \quad \forall P_{oN}. [P 0_N \wedge \forall n_N [P n \supset P [S n]]] \supset \forall m_N. P m$$

Sequent:

$$\neg(IND), \neg(1), \neg(2), Q . \underbrace{F \dots F}_{256} 0$$

Lemma L^k :

$$\forall x_{\iota}. Q x \supset Q . \underbrace{F \dots F}_{2^k} x$$

Cut-Simulation

Sequent Γ :

$$\neg(IND), \neg(1), \neg(2), Q . \underbrace{F \cdots F}_{256} 0$$

Cut-Simulation

Sequent Γ :

$$\neg(IND), \neg(1), \neg(2), Q . \underbrace{F \cdots F}_{256} 0$$

Let \mathbf{L} be one of the lemmas \mathbf{L}^k . Apply Contraction and $\mathcal{G}(\neg\forall)$ to (IND) to instantiate P with $[\lambda n_{\mathbf{N}} \mathbf{L}]$.

Cut-Simulation

Sequent Γ :

$$\neg(IND), \neg(1), \neg(2), Q . \underbrace{F \cdots F}_{256} 0$$

Let \mathbf{L} be one of the lemmas \mathbf{L}^k . Apply Contraction and $\mathcal{G}(\neg\forall)$ to (IND) to instantiate P with $[\lambda n_{\mathbf{N}} \mathbf{L}]$. After a few rules we have three subgoals:

Cut-Simulation

Sequent Γ :

$$\neg(IND), \neg(1), \neg(2), Q . \underbrace{F \cdots F}_{256} 0$$

Let \mathbf{L} be one of the lemmas \mathbf{L}^k . Apply Contraction and $\mathcal{G}(\neg\forall)$ to (IND) to instantiate P with $[\lambda n_{\mathbf{N}} \mathbf{L}]$. After a few rules we have three subgoals:

1. Γ, \mathbf{L} (show 0 in $\{n_{\mathbf{N}} | \mathbf{L}\}$)

Cut-Simulation

Sequent Γ :

$$\neg(IND), \neg(1), \neg(2), Q . \underbrace{F \cdots F}_{256} 0$$

Let \mathbf{L} be one of the lemmas \mathbf{L}^k . Apply Contraction and $\mathcal{G}(\neg\forall)$ to (IND) to instantiate P with $[\lambda n_{\mathbf{N}} \mathbf{L}]$. After a few rules we have three subgoals:

1. Γ, \mathbf{L} (show 0 in $\{n_{\mathbf{N}} | \mathbf{L}\}$)
2. $\Gamma, \mathbf{L} \supset \mathbf{L}$ (show $\{n_{\mathbf{N}} | \mathbf{L}\}$ is closed under successor)

Cut-Simulation

Sequent Γ :

$$\neg(IND), \neg(1), \neg(2), Q . \underbrace{F \cdots F}_{256} 0$$

Let \mathbf{L} be one of the lemmas \mathbf{L}^k . Apply Contraction and $\mathcal{G}(\neg\forall)$ to (IND) to instantiate P with $[\lambda n_{\mathbf{N}} \mathbf{L}]$. After a few rules we have three subgoals:

1. Γ, \mathbf{L} (show 0 in $\{n_{\mathbf{N}} | \mathbf{L}\}$)
2. $\Gamma, \mathbf{L} \supset \mathbf{L}$ (show $\{n_{\mathbf{N}} | \mathbf{L}\}$ is closed under successor)
3. $\Gamma, \neg\mathbf{L}$ (use $\{n_{\mathbf{N}} | \mathbf{L}\}$ is true for every n).

Cut-Simulation

Sequent Γ :

$$\neg(IND), \neg(1), \neg(2), Q . \underbrace{F \cdots F}_{256} 0$$

Let \mathbf{L} be one of the lemmas \mathbf{L}^k . Apply Contraction and $\mathcal{G}(\neg\forall)$ to (IND) to instantiate P with $[\lambda n_{\mathbf{N}} \mathbf{L}]$. After a few rules we have three subgoals:

1. Γ, \mathbf{L} (show 0 in $\{n_{\mathbf{N}} | \mathbf{L}\}$)
2. $\Gamma, \mathbf{L} \supset \mathbf{L}$ (show $\{n_{\mathbf{N}} | \mathbf{L}\}$ is closed under successor)
3. $\Gamma, \neg\mathbf{L}$ (use $\{n_{\mathbf{N}} | \mathbf{L}\}$ is true for every n).

The second is trivial (by $\mathcal{G}(Init)$).

Cut-Simulation

- Given a derivation of Γ, \mathbf{L} and a derivation of $\Gamma, \neg \mathbf{L}$, we have a derivation of Γ .

Cut-Simulation

- Given a derivation of Γ, \mathbf{L} and a derivation of $\Gamma, \neg \mathbf{L}$, we have a derivation of Γ .
- The derivation adds only a finite number of new rules.

Cut-Simulation

- Given a derivation of Γ, \mathbf{L} and a derivation of $\Gamma, \neg \mathbf{L}$, we have a derivation of Γ .
- The derivation adds only a finite number of new rules.
- In general, we may use (or abuse) induction to simulate cut.

Cut-Simulation

- Given a derivation of Γ, \mathbf{L} and a derivation of $\Gamma, \neg \mathbf{L}$, we have a derivation of Γ .
- The derivation adds only a finite number of new rules.
- In general, we may use (or abuse) induction to simulate cut.
- More generally, many impredicative formulas (other than induction) can be used to simulated cut.

Cut-Simulation

- Given a derivation of Γ, \mathbf{L} and a derivation of $\Gamma, \neg \mathbf{L}$, we have a derivation of Γ .
- The derivation adds only a finite number of new rules.
- In general, we may use (or abuse) induction to simulate cut.
- More generally, many impredicative formulas (other than induction) can be used to simulated cut.
- In the presence of such a formula, one may perform cut-elimination without blow-up in size.

Cut-Simulation

- Given a derivation of Γ, \mathbf{L} and a derivation of $\Gamma, \neg \mathbf{L}$, we have a derivation of Γ .
- The derivation adds only a finite number of new rules.
- In general, we may use (or abuse) induction to simulate cut.
- More generally, many impredicative formulas (other than induction) can be used to simulated cut.
- In the presence of such a formula, one may perform cut-elimination without blow-up in size.
- Q: Is this good or bad?

Cut-Simulation

- Given a derivation of Γ, \mathbf{L} and a derivation of $\Gamma, \neg \mathbf{L}$, we have a derivation of Γ .
- The derivation adds only a finite number of new rules.
- In general, we may use (or abuse) induction to simulate cut.
- More generally, many impredicative formulas (other than induction) can be used to simulated cut.
- In the presence of such a formula, one may perform cut-elimination without blow-up in size.
- Q: Is this good or bad?
- Q: “Good”? “Bad”?

Consequences of Cut-Simulation

- In the presence of induction (or some other such impredicative formula) the search space without cut is essentially the same as the search space without cut.

Consequences of Cut-Simulation

- In the presence of induction (or some other such impredicative formula) the search space without cut is essentially the same as the search space without cut.
- Cut-elimination is practically meaningless in such situations (much like the subformula property).

Consequences of Cut-Simulation

- In the presence of induction (or some other such impredicative formula) the search space without cut is essentially the same as the search space without cut.
- Cut-elimination is practically meaningless in such situations (much like the subformula property).
- Finding instantiations for P corresponds to finding cut formulas.

Options

Option 1: “Build in” the induction axiom. For example, we could have new rules for dealing with type N.

Options

Option 1: “Build in” the induction axiom. For example, we could have new rules for dealing with type N.

- The idea of “building in” axioms has already been followed for comprehension axioms, extensionality axioms and properties of (Leibniz) equality.

Options

Option 1: “Build in” the induction axiom. For example, we could have new rules for dealing with type N.

- The idea of “building in” axioms has already been followed for comprehension axioms, extensionality axioms and properties of (Leibniz) equality.
- In some sense λ -calculus “builds in” comprehension axioms.

Options

Option 1: “Build in” the induction axiom. For example, we could have new rules for dealing with type N.

- The idea of “building in” axioms has already been followed for comprehension axioms, extensionality axioms and properties of (Leibniz) equality.
- In some sense λ -calculus “builds in” comprehension axioms.
- Including comprehension axioms permits cut-simulation.

Options

Option 1: “Build in” the induction axiom. For example, we could have new rules for dealing with type N.

- The idea of “building in” axioms has already been followed for comprehension axioms, extensionality axioms and properties of (Leibniz) equality.
- In some sense λ -calculus “builds in” comprehension axioms.
- Including comprehension axioms permits cut-simulation.
- With λ -calculus (and logical constants), there is no need to include comprehension axioms explicitly.

Options

Option 1: “Build in” the induction axiom. For example, we could have new rules for dealing with type N.

- The idea of “building in” axioms has already been followed for comprehension axioms, extensionality axioms and properties of (Leibniz) equality.
- In some sense λ -calculus “builds in” comprehension axioms.
- Including comprehension axioms permits cut-simulation.
- With λ -calculus (and logical constants), there is no need to include comprehension axioms explicitly.
- This avoids the problem by avoiding adding the formula to the sequent.

Options

Option 1: “Build in” the induction axiom.

Drawbacks:

Options

Option 1: “Build in” the induction axiom.

Drawbacks:

- Even if we build in induction, one can still make new inductive definitions (for example, the even numbers) and this definition could be used for cut-simulation.

Options

Option 1: “Build in” the induction axiom.

Drawbacks:

- Even if we build in induction, one can still make new inductive definitions (for example, the even numbers) and this definition could be used for cut-simulation.
- Taken to an extreme this program suggests simply using a predicative logic with extra built-in properties. Impredicativity is hard, but nice.

Options

Option 2: Restrict the instantiations for the induction axiom. For example, if $[\lambda n_N A_o]$ is instantiated for P , then n must occur free in A .

- Such a restriction is natural.

Options

Option 2: Restrict the instantiations for the induction axiom. For example, if $[\lambda n_N A_o]$ is instantiated for P , then n must occur free in A .

Drawbacks:

Options

Option 2: Restrict the instantiations for the induction axiom. For example, if $[\lambda n_{\mathbf{N}} \mathbf{A}_o]$ is instantiated for P , then n must occur free in \mathbf{A} .

Drawbacks:

- We can still perform cut-simulation by instantiating with

$$[\lambda n. \forall q_{o\mathbf{N}}. q n \supset q n]$$

and using the (new) quantifier to introduce the cut formula.

Options

Option 2: Restrict the instantiations for the induction axiom. For example, if $[\lambda n_{\mathbf{N}} \mathbf{A}_o]$ is instantiated for P , then n must occur free in \mathbf{A} .

Drawbacks:

- We can still perform cut-simulation by instantiating with

$$[\lambda n. \forall q_{o\mathbf{N}}. q n \supset q n]$$

and using the (new) quantifier to introduce the cut formula.

- To avoid cut-simulation one would need far more serious restrictions on instantiations (not simply on induction).

Options

Option 2: Restrict the instantiations for the induction axiom. For example, if $[\lambda n_{\mathbf{N}} \mathbf{A}_o]$ is instantiated for P , then n must occur free in \mathbf{A} .

Drawbacks:

- We can still perform cut-simulation by instantiating with

$$[\lambda n. \forall q_{o\mathbf{N}}. q n \supset q n]$$

and using the (new) quantifier to introduce the cut formula.

- To avoid cut-simulation one would need far more serious restrictions on instantiations (not simply on induction).
- Not clear even the simple restriction on induction is complete.

Options

Option 3: Accept cut. Look for other restrictions.

Options

Option 3: Accept cut. Look for other restrictions.

- Inverse Method? (Forward search with sequent calculus)

Options

Option 3: Accept cut. Look for other restrictions.

- Inverse Method? (Forward search with sequent calculus)
- How do we prevent blind search?