Proofs with feasible computational content

Helmut Schwichtenberg

Mathematisches Institut der Universität München

16th May 2006
1. Proof interpretations
2. Feasible computation with higher types
3. Proofs and proof terms
Proof interpretation

Hilbert (1925) introduced recursion in all finite types. Gödel (1958): system $T$, as an “extension of the finitary standpoint”.

**Dialectica** interpretation: assign to a proof of $A$ a formula

$$\exists x \forall y |A|^x_y,$$

$|A|^x_y$: $x$ is a realizer for $A$ w.r.t. a challenge $y$.

**Modified realizability** (Kreisel, 1959): emphasis on constructive truth rather than consistency. “Alternative interpretation”.

Oliva (2006) developed a unified proof interpretation, containing both as special cases.

Here: natural deduction treatment (soundness, contraction).
The type of a formula

\[
\rho \ast \sigma := \begin{cases} 
\sigma & \text{if } \rho = \varepsilon \\
\rho & \text{if } \sigma = \varepsilon \\
\rho \times \sigma & \text{otherwise,}
\end{cases}
\]

\[
\rho \rhd \sigma := \begin{cases} 
\varepsilon & \text{if } \rho = \varepsilon \\
\varepsilon & \text{if } \sigma = \varepsilon \\
\rho \rightarrow \sigma & \text{otherwise.}
\end{cases}
\]

We define \(\tau^+(A)\), \(\tau^-(A)\) (type of a realizer/challenge). In case \(\tau^+(A) = \varepsilon\) proofs of \(A\) have no computational content; such formulas \(A\) are called Harrop formulas.

\[
\tau^+(A \rightarrow B) := (\tau^+(A) \rhd \tau^+(B)) \ast (\tau^+(A) \ast \tau^-(B) \rhd \tau^-(A))
\]

\[
\tau^-(A \rightarrow B) := \tau^+(A) \ast \tau^-(B)
\]

\[
\tau^+(\forall x \rho A) := \rho \rhd \tau^+(A)
\]

\[
\tau^-(\forall x \rho A) := \rho \ast \tau^-(A)
\]

\[
\tau^+(\exists x \rho A) := \rho \ast \tau^+(A)
\]

\[
\tau^-(\exists x \rho A) := \tau^-(A)
\]
Formula interpretation

We extend our language by an uninterpreted bounded universal quantifier $\forall_{x \sqsubseteq t} A$; this should be viewed as an abbreviation rather than a new formula construct.

For example, $\forall_{x \sqsubseteq t} A(x)$ might stand for $A(t)$ or $\forall_x A(x)$. For a fixed abbreviation $\forall_{x \sqsubseteq t} A$, we define the $\sqsubseteq$-bounded formulas by

$$A_b, B_b ::= P \mid A_b \rightarrow B_b \mid \forall_{x \sqsubseteq t} A_b.$$ 

We require the following properties, which express that $\forall_{x \sqsubseteq t} A$ behaves like a bounded quantifier, w.r.t. $\sqsubseteq$-bounded formulas:

$$\forall_x (\tilde{A}_b \rightarrow B_b) \rightarrow \forall_{x \sqsubseteq y} \tilde{A}_b \rightarrow \forall_{x \sqsubseteq y} B_b,$$

(1)

$$\forall_{x \sqsubseteq y} A_b \leftrightarrow A_b \quad \text{if } x \notin \text{FV}(A_b).$$

(2)
Requirements on the bounded quantifier

We require the existence of constants $b_1, b_2, b_3$ such that

\[\forall y \subseteq b_1 \bar{a} x \quad A_b(y) \rightarrow A_b(x),\]  
(3)

\[\forall y \subseteq b_2 \bar{a} y_0 y_1 \quad A_b(y) \rightarrow \forall y \subseteq y_i \quad A_b(y) \quad \text{for } i \in \{0, 1\},\]  
(4)

\[\forall y \subseteq b_3 \bar{a} h u \quad A_b(y) \rightarrow \forall z \subseteq u \forall y \subseteq h z \quad A_b(y);\]  
(5)

$\bar{a}$ consists of the variables free in $A_b(x)$ other than $x$. Intuitively, (3) expresses the existence of a uniform $\sqsubseteq$-bound for every $x$. (4) says that a binary union of $\sqsubseteq$-bounded sets is $\sqsubseteq$-bounded, and (5) that a $\sqsubseteq$-bounded union of $\sqsubseteq$-bounded sets is $\sqsubseteq$-bounded.
Gödel interpretation $|A|^x_y$

Intuitively, $|A|^x_y$ means that $x$ is a realizer for $A$ w.r.t. a challenge $y$.

$$|P(\vec{s})|_y^x := P(\vec{s})$$ for a prime formula $P(\vec{s})$,

$$|A \rightarrow B|_{x,u}^{f,g} := \forall y \sqsubseteq gxu \quad |A|^x_y \rightarrow |B|^{fx}_u,$$

$$|\forall z A|_{z,y}^f := |A|^{fz}_y,$$

$$|\exists z A|_{y}^{z,x} := |A|^{x}_y.$$
Natural deduction

<table>
<thead>
<tr>
<th>derivation</th>
<th>term</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u: A$</td>
<td>$u^A$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$\lambda u^A M^B$</td>
<td>$(\lambda u^A M^B)^{A\rightarrow B}$</td>
</tr>
<tr>
<td>$\lambda u^A M^B$</td>
<td>$(\lambda u^A M^B)^{A\rightarrow B}$</td>
</tr>
<tr>
<td>$\lambda u^A M^B$</td>
<td>$(\lambda u^A M^B)^{A\rightarrow B}$</td>
</tr>
</tbody>
</table>
Natural deduction: \( \forall \)-rules

<table>
<thead>
<tr>
<th>derivation</th>
<th>term</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ( \frac{M}{A} ) ( \forall^+ x ) (VarC) ]</td>
<td>(( \lambda x M^A ))( \forall^x A ) (VarC)</td>
</tr>
<tr>
<td>[ ( \forall^x A ) ]</td>
<td></td>
</tr>
<tr>
<td>[ ( \frac{M}{\forall^x A(x)} ) ( r ) ( \forall^- ) ]</td>
<td>(( M^{\forall^x A(x)} r ))( A(r) )</td>
</tr>
</tbody>
</table>
Soundness

Theorem

Let $M$ be a derivation of $A$ from assumptions $u_i : C_i$. Then we can find terms $\left[M\right]^+ =: t$, $\left[M\right]^i =: r_i$ and a derivation of $|A|^t_w$ from assumptions $u'_i : \forall y_i \models r_i \mid C_i \mid y_i$, where $t$ does not depend on $w$.

Proof by induction on $M$. Case $M^A \rightarrow^B N^A$. Assume that both $M$, $N$ have only one free assumption $u : C$, which is shared. Let $\left[M\right]^+ =: r, t$ and $\left[M\right]^i =: p$. By IH we have a derivation of

$$|A \rightarrow B|^{r,t}_{x,w} = \forall y \models r x w \mid A\mid y \rightarrow \mid B\mid^{tx}_w$$

from $\forall z \models p \mid C\mid z$ (6)

and of

$$|A|_y^s$$

from $\forall z \models q \mid C\mid z$, (7)

with $s := \left[N\right]^+$ and $q := \left[N\right]^i$. Substituting $s$ for $x$ in (6) gives
Soundness (continued)

\[ \forall y \sqsubseteq rsw \ |A|_y^s \rightarrow |B|_w^{ts} \quad \text{from} \quad \forall z \sqsubseteq p' \ |C|_z^v, \]

where \( p' \) arises from \( p \) by substituting \( s \) for \( x \). Applying the property (3) of bounded quantification to (7) gives

\[ \forall y \sqsubseteq rsw \ |A|_y^s \quad \text{from} \quad \forall y \sqsubseteq rsw \ \forall z \sqsubseteq q \ |C|_z^v, \quad \text{hence from} \quad \forall z \sqsubseteq q' \ |C|_z^v \]

with \( q' := b_3 \vec{a}(\lambda y \ q)(rsw) \), by (5). Using \((\rightarrow^-)\) and (4) we obtain

\[ |B|_w^{ts} \quad \text{from} \quad \forall z \sqsubseteq b_2 \vec{a}_1 p' q' \ |C|_z^v, \]

where \( \vec{a}_1 \) (contained in \( \vec{a} \)) codes the free variables of \( |C|_z^v \) except \( z \). So we can define the required terms by \([MN]^+ := ts\) and \([MN]^− := b_2 \vec{a}_1 p' q'\).
Axioms

(1) Let \( B = \forall \vec{y} \ A(\vec{y}) \), with \( A(\vec{y}) \) quantifier-free. Then \( \tau^+(B) = \varepsilon \), \( |B|_y = A(\vec{y}) \). Hence such axioms are interpreted by themselves.

(2) Consider the natural numbers. Induction schema:

\[ A(0) \rightarrow \forall_m (A(m) \rightarrow A(m + 1)) \rightarrow \forall_n A(n). \quad (8) \]

Let \( B(n) := A(0) \rightarrow \forall_m (A(m) \rightarrow A(m + 1)) \rightarrow A(n) \). Clearly we can derive \( B(0) \) and \( B(n) \rightarrow B(n + 1) \). The Soundness Theorem gives us realizing terms \( s \) and \( t, r \) and derivations of \( |B(0)|^s_y \) and of \( |B(n) \rightarrow B(n + 1)|^{t,r}_{x,u} \), hence of

\[
\begin{align*}
\forall_{y \sqsubseteq rxu} |B(n)|^x_y &\rightarrow |B(n + 1)|^t_x u \\
\forall_y |B(n)|^x_y &\rightarrow |B(n + 1)|^t_x u \quad \text{by (1)} \\
\forall_y |B(n)|^x_y &\rightarrow \forall_y |B(n + 1)|^t_x y.
\end{align*}
\]

Define \( g(0) := s \) and \( g(n + 1) := t(g(n)) \). Then, by induction, \( \forall_y |B(n)|^{g(n)}_y \), so \( \exists_g \forall_y \forall_n B(n)|^g_y \). But \( \forall_n B(n) \) is equivalent to (8).
Instantiations of the Bounded Quantifier

Kreisel’s modified realizability is usually introduced as follows. For every formula \( A \) we define a new formula \( \text{xmr} A \) be induction on \( A \).

\[
\begin{align*}
\varepsilon \text{mr} P(\vec{s}) & := P(\vec{s}) \quad \text{for a prime formula} \ P(\vec{s}), \\
\text{f mr} (A \to B) & := \forall x \ (\text{x mr} A \to \text{fx mr} B), \\
\text{f mr} (\forall z A) & := \forall z \ \text{f z mr} A, \\
z, \text{x mr} (\exists z A) & := \text{x mr} A.
\end{align*}
\]

We can view the modified realizability interpretation as a special case of the unified functional interpretation, if we define \( \forall_x t A(x) \) simply as \( \forall_x A(x) \). Then the clause for implication in the definition of \( |A|^x_{y} \) becomes

\[
|A \to B|^f_{x,u} := \forall y |A|^x_{y} \to |B|^f_{u}.
\]

Conditions (1) – (5) trivially hold for this definition of the “bounded” quantifier.
Gödel’s Dialectica interpretation

Gödel’s original Dialectica interpretation is usually introduced as follows. For every formula $A$ we define new formulas $A^D$, $A_D$ such that $A^D = \exists x \forall y A_D(x, y)$ with $A_D$ quantifier-free, by induction on $A$. Let

$$P(\vec{s})^D := P(\vec{s})$$

for a prime formula $P(\vec{s})$.

Assume $A^D = \exists x \forall y A_D(x, y)$ and $B^D = \exists v \forall w B_D(v, w)$. Then

$$(A \rightarrow B)^D := \exists f, g \forall x, w (A_D(x, gxw) \rightarrow B_D(fx, w)),$$

$$(\forall z A(z))^D := \exists f \forall z, w A_D(fz, x, z),$$

$$(\exists z A(z))^D := \exists z, x \forall y A_D(x, y, z).$$

In each case, $(\cdot)^D$ is to be the largest quantifier-free subformula.
Gödel’s Dialectica interpretation (continued)

We can obtain the Gödel’s Dialectica interpretation as a special case of the unified functional interpretation, simply by viewing $\forall_x t A(x)$ as an abbreviation for $A(t)$. Then by definition

$$|A \rightarrow B|_{x,u}^{f,g} = |A|_{g|u}^x \rightarrow |B|_{u}^{fx}.$$ 

Conditions (1), (2) trivially hold for this definition of the “bounded” quantifier, and (3) – (5) above now read as

$$A_b(b_1 \bar{a}x) \rightarrow A_b(x),$$
$$A_b(b_2 \bar{a}y_0 y_1) \rightarrow A_b(y_i) \text{ for } i \in \{0, 1\},$$
$$A_b(b_3 \bar{a}hu) \rightarrow A_b(hu).$$

This is easy to achieve: Define $b_1 \bar{a}x := x$ and $b_3 \bar{a}hu := hu$. For $b_2$ we use that $A_b$ is quantifier-free, and hence that there exists a boolean-valued term $t_{A_b}$ such that $t_{A_b} = \text{tt} \iff A_b$. Then define

$$b_2 \bar{a}y_0 y_1 := \begin{cases} y_0 & \text{if } t_{A_b}(y_0) = \text{tt} \\ y_1 & \text{otherwise.} \end{cases}$$
Feasible computation with higher types


LT (Bellantoni, Niggl, S. 2000, 2002): restriction such that the definable functions are exactly the polynomial time computable ones.

Here:

\[
\begin{align*}
\text{Heyting Arithmetic} & \quad = \quad \frac{?}{\text{Gödel’s } T} \\
\end{align*}
\]

\[
\frac{?}{\text{LT}}
\]
Feasible computation with higher types


LT (Bellantoni, Niggl, S. 2000, 2002): restriction such that the definable functions are exactly the polynomial time computable ones.

Here:

$$\frac{\text{Heyting Arithmetic}}{\text{Gödel’s T}} = \frac{?}{\text{LT}}$$
Feasible computation with higher types


LT (Bellantoni, Niggl, S. 2000, 2002): restriction such that the definable functions are exactly the polynomial time computable ones.

Here:

\[
\frac{\text{Heyting Arithmetic}}{\text{Gödel’s T}} \overset{?}{=} \frac{\text{LT}}{}
\]
Types

\[ \rho, \sigma ::= U | B | L(\rho) | \rho \rightarrow \sigma | \rho \rightarrowtail \sigma | \rho \rightarrow \sigma | \rho \otimes \sigma | \rho \times \sigma. \]

The level of a type is defined by

\[
\begin{align*}
    l(U) := l(B) & := 0 \\
    l(L(\rho)) & := l(\rho) \\
    l(\rho \rightarrow \sigma) := l(\rho \rightarrow \sigma) & := \max\{l(\sigma), 1 + l(\rho)\} \\
    l(\rho \otimes \sigma) := l(\rho \times \sigma) & := \max\{l(\rho), l(\sigma)\}
\end{align*}
\]

Ground types are the types of level 0, and a higher type is any type of level at least 1. The \(\rightarrow\)-free types are called linear types.

In particular, each ground type is linear.
Types

\[ \rho, \sigma ::= U \mid B \mid L(\rho) \mid \rho \circ \sigma \mid \rho \to \sigma \mid \rho \otimes \sigma \mid \rho \times \sigma. \]

The **level** of a type is defined by

\[
\begin{align*}
    l(U) & := l(B) := 0 \\
    l(L(\rho)) & := l(\rho) \\
    l(\rho \to \sigma) & := \max\{l(\sigma), 1 + l(\rho)\} \\
    l(\rho \otimes \sigma) & := \max\{l(\rho), l(\sigma)\}
\end{align*}
\]

Ground types are the types of level 0, and a **higher** type is any type of level at least 1. The \( \to \)-free types are called **linear** types.

In particular, each ground type is linear.
Types

\[ \rho, \sigma ::= U \mid B \mid L(\rho) \mid \rho \to \sigma \mid \rho \to \sigma \mid \rho \otimes \sigma \mid \rho \times \sigma . \]

The level of a type is defined by

\[
\begin{align*}
l(U) & := l(B) := 0 \\
l(L(\rho)) & := l(\rho) \\
l(\rho \to \sigma) & := l(\rho \to \sigma) := \max\{l(\sigma), 1 + l(\rho)\} \\
l(\rho \otimes \sigma) & := l(\rho \times \sigma) := \max\{l(\rho), l(\sigma)\}
\end{align*}
\]

Ground types are the types of level 0, and a higher type is any type of level at least 1. The \(\to\)-free types are called linear types.

In particular, each ground type is linear.
Types

\[ \rho, \sigma ::= U \mid B \mid L(\rho) \mid \rho \rightarrow^\circ \sigma \mid \rho \rightarrow \sigma \mid \rho \otimes \sigma \mid \rho \times \sigma. \]

The level of a type is defined by

\[
\begin{align*}
    l(U) & := l(B) := 0 \\
    l(L(\rho)) & := l(\rho) \\
    l(\rho \rightarrow^\circ \sigma) & := l(\rho \rightarrow \sigma) := \max\{l(\sigma), 1 + l(\rho)\} \\
    l(\rho \otimes \sigma) & := l(\rho \times \sigma) := \max\{l(\rho), l(\sigma)\}
\end{align*}
\]

Ground types are the types of level 0, and a higher type is any type of level at least 1. The \( \rightarrow \)-free types are called linear types.

In particular, each ground type is linear.
Constants

\(\varepsilon : U\)
\(tt, ff : B\)
\(nil_\rho : L(\rho)\)
\(\text{cons}_\rho : \rho \rightarrow L(\rho) \rightarrow L(\rho)\)
\(\text{if}_\tau : B \rightarrow \tau \times \tau \rightarrow \tau \quad (\tau \text{ linear})\)
\(c^\rho_\tau : L(\rho) \rightarrow \tau \times (\rho \rightarrow L(\rho) \rightarrow \tau) \rightarrow \tau \quad (\tau \text{ linear})\)
\(R^\rho_\tau : L(\rho) \rightarrow (\rho \rightarrow L(\rho) \rightarrow \tau \rightarrow \tau) \rightarrow \tau \rightarrow \tau \quad (\rho \text{ ground, } \tau \text{ linear})\)
For linear $\rho, \sigma, \tau$:

\[ \otimes_{\rho\sigma}^+: \rho \multimap \sigma \multimap \rho \otimes \sigma \]
\[ \otimes_{\rho\sigma\tau}^-: \rho \otimes \sigma \multimap (\rho \multimap \sigma \multimap \tau) \multimap \tau \]
\[ \times_{\rho\sigma}^+: \rho \multimap \sigma \multimap \rho \times \sigma \quad \text{(if $\rho, \sigma$ ground)} \]
\[ \times_{\rho\sigma\tau}^+: (\tau \multimap \rho) \multimap (\tau \multimap \sigma) \multimap \tau \multimap \rho \times \sigma \quad \text{(if } l(\rho \times \sigma) > 0) \text{)} \]
\[ \text{fst}_{\rho\sigma}: \rho \times \sigma \multimap \rho \]
\[ \text{snd}_{\rho\sigma}: \rho \times \sigma \multimap \sigma \]
**LT-terms**

are built from constants and typed variables \( x^\sigma \) (incomplete) and \( \overline{x}^\sigma \) (complete) by introduction and elimination rules for the two type forms \( \rho \rightarrowo \sigma \) and \( \rho \rightarrow \sigma \), i.e.

\[
\begin{align*}
&c^\rho \quad \text{(constant)} \mid \\
&x^\rho \quad \text{(incomplete variable)} \mid \\
&\overline{x}^\rho \quad \text{(complete variable)} \mid \\
&(\lambda x^\rho r^\sigma)^{\rho \rightarrowo \sigma} \mid \\
&(r^{\rho \rightarrowo \sigma} s^\rho)^\sigma \quad \text{with higher type incomplete variables in } r, s \text{ distinct} \mid \\
&(\lambda \overline{x}^\rho r^\sigma)^{\rho \rightarrow \sigma} \mid \\
&(r^{\rho \rightarrow \sigma} s^\rho)^\sigma \quad \text{with } s \text{ complete}
\end{align*}
\]

A term \( s \) is **complete** if all of its free variables are complete, else **incomplete**. A term is **linear** or **ground** according as its type is.
Conversions

\((\lambda x r)s\) \quad \mapsto \quad r[x := s] \quad \beta\text{-conversion; similar for } \bar{x}

\text{if}_\tau \text{tt}s \quad \mapsto \quad \text{fst}_{\tau\tau}s

\text{if}_\tau \text{ff}s \quad \mapsto \quad \text{snd}_{\tau\tau}s

\text{c}_\tau^\rho \text{nil}_\rho s \quad \mapsto \quad \text{fst}_{\tau,\sigma}s \quad \text{for } \sigma := \rho \rightarrow \mathbf{L}(\rho) \rightarrow \tau

\text{c}_\tau^\rho (\text{cons}_\rho rl)s \quad \mapsto \quad \text{snd}_{\tau,\sigma}srl \quad \text{for } \sigma := \rho \rightarrow \mathbf{L}(\rho) \rightarrow \tau

\mathcal{R}_\tau^\rho \text{nil}_\rho st \quad \mapsto \quad t

\mathcal{R}_\tau^\rho (\text{cons}_\rho rl)st \quad \mapsto \quad srl(\mathcal{R}_\tau^\rho lst)

\otimes_{\rho\sigma\tau} (\otimes_{\rho\sigma}^+ rs)t \quad \mapsto \quad \text{trs}

\text{fst}_{\rho\sigma}(\times_{\rho\sigma}^+ rs) \quad \mapsto \quad r

\text{snd}_{\rho\sigma}(\times_{\rho\sigma}^+ rs) \quad \mapsto \quad s

\text{fst}_{\rho\sigma}(\times_{\rho\sigma\tau}^+ rst) \quad \mapsto \quad rt

\text{snd}_{\rho\sigma}(\times_{\rho\sigma\tau}^+ rst) \quad \mapsto \quad st
Computation in LT

Lemma (Sharing Normalization)

Let $t$ be an $\mathcal{R}$-free term whose higher-type variables are incomplete. Then a parse dag for $\text{nf}(t)$, of size at most $|t|$, can be computed from $t$ in time $O(|t|^2)$.

Proof.

Case $\text{fst}_{\rho\sigma}(\times^+_{\rho\sigma} rs) \mapsto r$ with $\rho \times \sigma$ a ground type.

The other cases are similar.
Lemma (Sharing Normalization)

Let $t$ be an $\mathcal{R}$-free term whose higher-type variables are incomplete. Then a parse dag for $\text{nf}(t)$, of size at most $|t|$, can be computed from $t$ in time $O(|t|^2)$.

Proof.

Case $\text{fst}_{\rho\sigma}(\times^+_{\rho\sigma} rs) \mapsto r$ with $\rho \times \sigma$ a ground type.

The other cases are similar.
Numerals

Terms of the form $\text{cons}_\rho r_1 \rho (\text{cons}_\rho r_2 \rho \ldots (\text{cons}_\rho r_n \rho \text{nil}_\rho) \ldots)$ are lists. Abbreviations for $\mathbf{N} := \mathbf{L}(\mathbf{U})$ and $\mathbf{W} := \mathbf{L}(\mathbf{B})$:

\[
\begin{align*}
0 & := \text{nil}_U \\
S & := \lambda^n \text{cons}_U \epsilon \\
1 & := \text{nil}_B \\
S_0 & := \lambda^W \text{cons}_B \text{ff} \\
S_1 & := \lambda^W \text{cons}_B \text{tt} \\
\end{align*}
\]

Particular lists are $\mathbf{S}(\ldots (\mathbf{S}_0) \ldots)$ and $\mathbf{S}_{i_1} (\ldots (\mathbf{S}_{i_n} 1) \ldots)$. The former are called \textit{unary numerals}, and the latter \textit{binary numerals}. 

Numerals

Terms of the form \( \text{cons}_\rho r_1^\rho (\text{cons}_\rho r_2^\rho \ldots (\text{cons}_\rho r_n^\rho \text{nil}_\rho) \ldots) \) are lists. Abbreviations for \( \mathbf{N} := \mathbf{L}(\mathbf{U}) \) and \( \mathbf{W} := \mathbf{L}(\mathbf{B}) \):

\[
\begin{align*}
0 & := \text{nil}_\mathbf{U} \\
S & := \lambda^\mathbf{N} \text{cons}_\mathbf{U}_\varepsilon_1 \\
1 & := \text{nil}_\mathbf{B} \\
S_0 & := \lambda^\mathbf{W} \text{cons}_\mathbf{B}_\text{ff} \\
S_1 & := \lambda^\mathbf{W} \text{cons}_\mathbf{B}_\text{tt}
\end{align*}
\]

Particular lists are \( S(\ldots(S_0)\ldots) \) and \( S_{i_1}(\ldots(S_{i_n}1)\ldots) \). The former are called \textit{unary numerals}, and the latter \textit{binary numerals}. 

Numerals

Terms of the form \( \text{cons}_\rho r_1^\rho (\text{cons}_\rho r_2^\rho \ldots (\text{cons}_\rho r_n^\rho \text{nil}_\rho) \ldots) \) are lists. Abbreviations for \( \mathbf{N} := \mathbf{L}(\mathbf{U}) \) and \( \mathbf{W} := \mathbf{L}(\mathbf{B}) \):

\[
\begin{align*}
0 & := \text{nil}_\mathbf{U} \\
S & := \lambda^\mathbf{N} \text{cons}_\mathbf{U} \varepsilon \mathbf{U} \\
1 & := \text{nil}_\mathbf{B} \\
S_0 & := \lambda^\mathbf{W} \text{cons}_\mathbf{B} \text{ff} \mathbf{B} \\
S_1 & := \lambda^\mathbf{W} \text{cons}_\mathbf{B} \text{tt} \mathbf{B}
\end{align*}
\]

Particular lists are \( S(\ldots(S_0)\ldots) \) and \( S_{i_1}(\ldots(S_{i_n}1)\ldots) \). The former are called **unary numerals**, and the latter **binary numerals**.
Terms of the form $\text{cons}_\rho r_1^\rho (\text{cons}_\rho r_2^\rho \ldots (\text{cons}_\rho r_n^\rho \text{nil}_\rho ) \ldots )$ are lists. Abbreviations for $\mathbf{N} := \mathbf{L}(\mathbf{U})$ and $\mathbf{W} := \mathbf{L}(\mathbf{B})$:

\[
\begin{align*}
0 & := \text{nil}_\mathbf{U} \\
S & := \lambda^\mathbf{N} \text{cons}_\mathbf{U} \varepsilon /
\end{align*}
\]
\[
\begin{align*}
1 & := \text{nil}_\mathbf{B} \\
S_0 & := \lambda^\mathbf{W} \text{cons}_\mathbf{B} \mathbf{ff} /
\end{align*}
\]
\[
\begin{align*}
S_1 & := \lambda^\mathbf{W} \text{cons}_\mathbf{B} \mathbf{tt} /
\end{align*}
\]

Particular lists are $S(\ldots (S0) \ldots )$ and $S_{i_1}(\ldots (S_{i_n}1) \ldots )$. The former are called **unary numerals**, and the latter **binary numerals**.
Polynomials

⊕: \( W \rightarrow W \rightarrow W \). \( x \oplus y \) concatenates \(|x|\) bits onto \( y \).

\[
1 \oplus y = S_0 y \\
(S_i x) \oplus y = S_0(x \oplus y)
\]

\( \bar{x} \oplus y := R_{W \rightarrow W} \bar{x}(\lambda \bar{z} \lambda \bar{l} \lambda p_{W \rightarrow W} \lambda y. S_0(py))S_0. \)

⊙: \( W \rightarrow W \rightarrow W \). \( x \odot y \) has output length \(|x| \cdot |y|\).

\[
x \odot 1 = x \\
x \odot (S_i y) = x \oplus (x \odot y)
\]

\( \bar{x} \odot \bar{y} := R_{W \rightarrow W} \bar{y}(\lambda \bar{z} \lambda \bar{l} \lambda p_{W} \bar{x} \oplus p)\bar{x}. \)
Polynomials

$\oplus : W \rightarrow W \rightarrow W$. $x \oplus y$ concatenates $|x|$ bits onto $y$.

$$1 \oplus y = S_0y$$

$$(S_i x) \oplus y = S_0(x \oplus y)$$

$$\bar{x} \oplus y := R_{w \circ w} \bar{x}(\lambda \bar{z} \lambda \bar{l} \lambda p^{w \circ w} \lambda y. S_0(py))S_0.$$

$\circ : W \rightarrow W \rightarrow W$. $x \circ y$ has output length $|x| \cdot |y|$.

$$x \circ 1 = x$$

$$x \circ (S_i y) = x \oplus (x \circ y)$$

$$\bar{x} \circ \bar{y} := R_{w \circ w} \bar{y}(\lambda \bar{z} \lambda \bar{l} \lambda p^w. \bar{x} \oplus p)\bar{x}.$$
Polynomials

⊕: \( W \rightarrow W \rightarrow W \). \( x \oplus y \) concatenates \(|x|\) bits onto \( y \).

\[1 \oplus y = S_0y\]
\[(S_i x) \oplus y = S_0(x \oplus y)\]

\(\bar{x} \oplus y := R_{W \odot W} \bar{x}(\lambda z \bar{l} \lambda p^{w \odot w} \lambda y. S_0(py)) S_0.\)

⊙: \( W \rightarrow W \rightarrow W \). \( x \odot y \) has output length \(|x| \cdot |y|\).

\[x \odot 1 = x\]
\[x \odot (S_i y) = x \oplus (x \odot y)\]

\(\bar{x} \odot \bar{y} := R_{W \bar{y}}(\lambda \bar{z} \bar{l} \lambda p^W. \bar{x} \oplus p) \bar{x}.\)
Polynomials

⊕: \( W \rightarrow W \rightarrow W \). \( x \oplus y \) concatenates \(|x|\) bits onto \( y \).

\[
1 \oplus y = S_0y
\]
\[
(S_i \cdot x) \oplus y = S_0(x \oplus y)
\]

\( \bar{x} \oplus y := \mathcal{R}_{w \circ w} \bar{x}(\lambda \bar{z} \lambda \bar{l} \lambda p^{w \circ w} \lambda y. S_0(py))S_0. \)

⊙: \( W \rightarrow W \rightarrow W \). \( x \circ y \) has output length \(|x| \cdot |y|\).

\[
x \circ 1 = x
\]
\[
x \circ (S_iy) = x \oplus (x \circ y)
\]

\( \bar{x} \circ \bar{y} := \mathcal{R}_w \bar{y}(\lambda \bar{z} \lambda \bar{l} \lambda p^w. \bar{x} \oplus p)\bar{x}. \)
Functions definable in **LT**

**Theorem (Normalization)**

Let \( r \) be a closed **LT**-term of type \( W \to \ldots W \to W \) \((\to \in \{\rightarrow, \Rightarrow\})\). Then \( r \) denotes a polytime function.

**Proof.**

Uses a model of computation via parse dags; ground type terms can be shared. Details in Bellantoni & S. 2002.

**Theorem (Sufficiency)**

Let \( f \) be a polytime function. Then \( f \) is denoted by a closed **LT**-term \( t \).

**Proof.**

Induction on the definition of \( f(x_1, \ldots, x_k; y_1, \ldots, y_l) \) in Bellantoni and Cook’s \( B \), associating to \( f \) a closed term \( t_f \) of type \( W^{(k)} \to W^{(l)} \to W \), such that \( t_f \) denotes \( f \).
Functions definable in $\text{LT}$

**Theorem (Normalization)**

Let $r$ be a closed $\text{LT}$-term of type $W \rightarrow \ldots W \rightarrow W$ ($\rightarrow \in \{\rightarrow, \to\}$). Then $r$ denotes a polytime function.

**Proof.**

Uses a model of computation via parse dags; ground type terms can be shared. Details in Bellantoni & S. 2002.

**Theorem (Sufficiency)**

Let $f$ be a polytime function. Then $f$ is denoted by a closed $\text{LT}$-term $t$.

**Proof.**

Induction on the definition of $f(x_1, \ldots, x_k; y_1, \ldots, y_l)$ in Bellantoni and Cook’s $B$, associating to $f$ a closed term $t_f$ of type $W^{(k)} \rightarrow W^{(l)} \to W$, such that $t_f$ denotes $f$. 

$\square$
Functions definable in \textbf{LT}

\textbf{Theorem (Normalization)}

Let \( r \) be a closed \textbf{LT}-term of type \( \mathbb{W} \xrightarrow{} \ldots \mathbb{W} \xrightarrow{} \mathbb{W} \) \((\xrightarrow{} \in \{\rightarrow, \twoheadrightarrow\})\). Then \( r \) denotes a polytime function.

\textbf{Proof.}

Uses a model of computation via parse dags; ground type terms can be shared. Details in Bellantoni & S. 2002.

\textbf{Theorem (Sufficiency)}

Let \( f \) be a polytime function. Then \( f \) is denoted by a closed \textbf{LT}-term \( t \).

\textbf{Proof.}

Induction on the definition of \( f(x_1, \ldots, x_k; y_1, \ldots, y_l) \) in Bellantoni and Cook’s \( B \), associating to \( f \) a closed term \( t_f \) of type \( \mathbb{W}^{(k)} \rightarrow \mathbb{W}^{(l)} \rightarrow \mathbb{W} \), such that \( t_f \) denotes \( f \).
Functions definable in $\mathbf{LT}$

**Theorem (Normalization)**

Let $r$ be a closed $\mathbf{LT}$-term of type $\mathcal{W} \rightarrow \ldots \mathcal{W} \rightarrow \mathcal{W}$ ($\rightarrow \in \{\to, \rightarrow\}$). Then $r$ denotes a polytime function.

**Proof.**

Uses a model of computation via parse dags; ground type terms can be shared. Details in Bellantoni & S. 2002.

**Theorem (Sufficiency)**

Let $f$ be a polytime function. Then $f$ is denoted by a closed $\mathbf{LT}$-term $t$.

**Proof.**

Induction on the definition of $f(x_1, \ldots, x_k; y_1, \ldots, y_l)$ in Bellantoni and Cook’s $B$, associating to $f$ a closed term $t_f$ of type $\mathcal{W}^{(k)} \rightarrow \mathcal{W}^{(l)} \rightarrow \mathcal{W}$, such that $t_f$ denotes $f$. 

\[\square\]
**LHA-formulas**

\[ P(\vec{r}) \mid A \rightarrow B \mid A \rightarrow \circ B \mid A \otimes B \mid A \land B \mid A \land_{x^\rho} A \mid A \land_{x^\rho} A \mid A \land_{x^\rho} A \mid \exists_{x^\rho} A. \]

In \( P(\vec{r}) \), the \( \vec{r} \) are terms from \( T \). Define falsity \( \bot \) by \( tt = ff \) and negation \( \neg A \) by \( A \rightarrow \circ \bot \). Disjunction can be defined by

\[ A \lor B := \exists_{x^B} \left( (x = tt \rightarrow A) \land (x = ff \rightarrow B) \right). \]
Ordinary proof terms

\[ c^A \quad \text{(axiom)} \mid \]
\[ \bar{u}^A, u^A \quad \text{(complete and incomplete assumption variables)} \mid \]
\[ (\lambda \bar{u}^A M^B)^{A \rightarrow B} \mid (M^{A \rightarrow B} N^A)^B \mid (\lambda u^A M^B)^{A \leftarrow B} \mid (M^{A \leftarrow B} N^A)^B \mid \]
\[ (\lambda x^\rho M^A)^{\forall^\rho x^A} \mid (M^{\forall^\rho x^A} r^\rho)^{A[x:=r]} \mid \]
\[ (\lambda \bar{x}^\rho M^A)^{\forall \bar{x}^A} \mid (M^{\forall \bar{x}^A} r^\rho)^{A[\bar{x}:=r]} \mid (\lambda x^\rho M^A)^{\forall x^A} \mid (M^{\forall x^A} r^\rho)^{A[x:=r]} . \]

Here we do not distinguish between

- \( \forall \) and \( \forall^{nc} \),
- complete and incomplete variables,
- \( \rightarrow \) and \( \leftarrow \).

**LHA** proof terms will be selected from the ordinary ones, by conditions similar to those distinguishing **LT**-terms from **T**-terms.
Ordinary proof terms

c^A  (axiom) |
ê^A, u^A  (complete and incomplete assumption variables) |
(\lambda ê^A M^B)^{A\rightarrow B} | (M^{A\rightarrow B} N^A)^B | (\lambda u^A M^B)^{A\rightarrow B} | (M^{A\rightarrow B} N^A)^B |
(\lambda x^\rho M^A)^{\forall_{x^\rho} A} | (M^{\forall_{x^\rho} A} r^\rho)^A[x:=r] |
(\lambda x^\rho M^A)^{\forall_x A} | (M^{\forall_{x^\rho} A} r^\rho)^A[x:=r] | (\lambda x^\rho M^A)^{\forall_x A} | (M^{\forall_{x^\rho} A} r^\rho)^A[x:=r].

Here we do not distinguish between

▶ \forall and \forall^{nc},
▶ complete and incomplete variables,
▶ \rightarrow and \rightarrow^\circ.

LHA proof terms will be selected from the ordinary ones, by conditions similar to those distinguishing LT-terms from T-terms.
Ordinary proof terms

\[ c^A \quad \text{(axiom)} \mid \]
\[ \bar{u}^A, u^A \quad \text{(complete and incomplete assumption variables)} \mid \]
\[ (\lambda \bar{u}^A M^B)^{A \to B} \mid (M^{A \to B} N^A)^B \mid (\lambda u^A M^B)^{A \to o B} \mid (M^{A \to o B} N^A)^B \mid \]
\[ (\lambda x^\rho M^A)^{\forall x^\rho A} \mid (M^{\forall x^\rho A} r^\rho)^{A[x:=r]} \mid \]
\[ (\lambda x^\rho M^A)^{\forall x A} \mid (M^{\forall x^\rho A} r^\rho)^{A[x:=r]} \mid (\lambda x^\rho M^A)^{\forall A} \mid (M^{\forall x^\rho A} r^\rho)^{A[x:=r]} . \]

Here we do not distinguish between

- \( \forall \) and \( \forall^{nc} \),
- complete and incomplete variables,
- \( \to \) and \( \to^o \).

**LHA** proof terms will be selected from the ordinary ones, by conditions similar to those distinguishing **LT**-terms from **T**-terms.
Restrict rules for generating proof terms, similarly as we did for object terms. Consequence: extracted term is in $\textbf{LT}$.

- If $\tau(A) = \varepsilon$, then every ordinary proof term $M^A$ is an LHA proof term, and $\text{CV}(M) := \emptyset$.

- Every assumption constant (axiom) $c^A$ and every complete or incomplete assumption variable $\bar{u}^A$ or $u^A$ is an LHA proof term. $\text{CV}(\bar{u}^A) := \{\bar{x}_\bar{u}\}$ and $\text{CV}(u^A) := \{x_u\}$.

- If $M^A$ is an LHA proof term, then so is $(\lambda\bar{u}^A M)^{A\rightarrow B}$ and $(\lambda u^A M)^{A\rightarrow oB}$. $\text{CV}(\lambda\bar{u}^A M) = \text{CV}(M) \setminus \{\bar{x}_\bar{u}\}$ and $\text{CV}(\lambda u^A M) = \text{CV}(M) \setminus \{x_u\}$.

- If $M^{A\rightarrow B}$ and $N^A$ are LHA proof terms, then so is $(MN)^B$, provided all variables in $\text{CV}(N)$ are complete. $\text{CV}(MN) := \text{CV}(M) \cup \text{CV}(N)$.

- If $M^{A\rightarrow oB}$ and $N^A$ are LHA proof terms, then so is $(MN)^B$, provided the higher type incomplete variables in $\text{CV}(M)$ and $\text{CV}(N)$ are disjoint. $\text{CV}(MN) := \text{CV}(M) \cup \text{CV}(N)$. 
**LHA proof terms**

Restrict rules for generating proof terms, similarly as we did for object terms. Consequence: extracted term is in $\mathbf{LT}$.

- If $\tau(A) = \varepsilon$, then every ordinary proof term $M^A$ is an LHA proof term, and $CV(M) := \emptyset$.

- Every assumption constant (axiom) $c^A$ and every complete or incomplete assumption variable $\bar{u}^A$ or $u^A$ is an LHA proof term. $CV(\bar{u}^A) := \{\bar{x}_u\}$ and $CV(u^A) := \{x_u\}$.

- If $M^A$ is an LHA proof term, then so is $(\lambda \bar{u}^A M)^{A \rightarrow B}$ and $(\lambda u^A M)^{A \leftarrow o B}$. $CV(\lambda \bar{u}^A M) = CV(M) \setminus \{\bar{x}_u\}$ and $CV(\lambda u^A M) = CV(M) \setminus \{x_u\}$.

- If $M^A \rightarrow B$ and $N^A$ are LHA proof terms, then so is $(MN)^B$, provided all variables in $CV(N)$ are complete. $CV(MN) := CV(M) \cup CV(N)$.

- If $M^A \leftarrow o B$ and $N^A$ are LHA proof terms, then so is $(MN)^B$, provided the higher type incomplete variables in $CV(M)$ and $CV(N)$ are disjoint. $CV(MN) := CV(M) \cup CV(N)$. 
If $M^A$ is an LHA proof term, $x \notin \text{FV}(B)$ for every formula $B$ of a free assumption variable in $M$, and moreover $x \notin \text{CV}(M)$, then $(\lambda xM)^\forall^{\text{nc}}xA$ is an LHA proof term. $\text{CV}(\lambda xM) := \text{CV}(M)$.

If $M^A$ is an LHA proof term, and $\tilde{x} \notin \text{FV}(B)$ for every formula $B$ of a free assumption variable in $M$, then so is $(\lambda \tilde{x}M)^\forall^{\text{nc}}A$. $\text{CV}(\lambda \tilde{x}M) := \text{CV}(M) \setminus \{\tilde{x}\}$ ($\tilde{x}$ a complete or incomplete variable).

If $M^\forall^{\text{nc}}xA$ is an LHA proof term and $r$ is a T-term, then $(Mr)^A[x:=r]$ is an LHA proof term. $\text{CV}(Mr) := \text{CV}(M)$.

If $M^\forall^{\text{nc}}A$ is an LHA proof term and $r$ is a complete LT-term, then $(Mr)^A[\tilde{x}:=r]$ is an LHA proof term. $\text{CV}(Mr) := \text{CV}(M) \cup \text{FV}(r)$.

If $M^\forall xA$ is an LHA proof term and $r$ is an LT-term, then $(Mr)^A[x:=r]$ is an LHA proof term, provided the higher type incomplete variables in $\text{CV}(M)$ are not free in $r$. $\text{CV}(Mr) := \text{CV}(M) \cup \text{FV}(r)$. 

Lemma

For every LHA proof term $M$, $CV(M) = FV(\llbracket M \rrbracket)$.

Proof.

Induction on $M^A$. We may assume $\tau(A) \neq \varepsilon$.

Case $M^A \to^B N^A$ with $\tau(A) \neq \varepsilon$. Then

$CV(MN) = CV(M) \cup CV(N) \stackrel{IH}{=} FV(\llbracket M \rrbracket) \cup FV(\llbracket N \rrbracket) = FV(\llbracket M \rrbracket[\llbracket N \rrbracket]) = FV(\llbracket MN \rrbracket)$.

Case $(\lambda xM)^{\forall^ncxA}$. Then $CV(\lambda xM) = CV(M) \stackrel{IH}{=} FV(\llbracket M \rrbracket)$. But by definition $\llbracket(\lambda xM)^{\forall^ncxA}\rrbracket = \llbracket M \rrbracket$.

The other cases are similar.

We can now give a simple characterization of LHA proof terms, which refers to extracted terms and LT, and moreover to the notion of an nc-correct (ordinary) proof term, which is defined as follows:
Lemma

For every LHA proof term $M$, $CV(M) = FV([M])$.

Proof.

Induction on $M^A$. We may assume $\tau(A) \neq \varepsilon$.

Case $M^A \vdash^0 N^A$ with $\tau(A) \neq \varepsilon$. Then

$CV(MN) = CV(M) \cup CV(N) =_{IH} FV([M]) \cup FV([N]) = FV([M][N]) = FV([MN])$.

Case $(\lambda xM)^{\forall^{nc} x A}$. Then $CV(\lambda xM) = CV(M) =_{IH} FV([M])$. But by definition $[(\lambda xM)^{\forall^{nc} x A}] = [M]$.

The other cases are similar.

We can now give a simple characterization of LHA proof terms, which refers to extracted terms and LT, and moreover to the notion of an nc-correct (ordinary) proof term, which is defined as follows:
Lemma

For every LHA proof term \( M \), \( CV(M) = FV([M]) \).

Proof.

Induction on \( M^A \). We may assume \( \tau(A) \neq \varepsilon \).

Case \( M^A \to^B N^A \) with \( \tau(A) \neq \varepsilon \). Then
\[
CV(MN) = CV(M) \cup CV(N) =_{\text{IH}} FV([M]) \cup FV([N]) = FV([M][N]) = FV([MN]).
\]

Case \((\lambda xM)^{\forall^{nc}xA}\). Then \( CV(\lambda xM) = CV(M) =_{\text{IH}} FV([M]) \). But by definition \([((\lambda xM)^{\forall^{nc}xA})] = [M] \).

The other cases are similar.

We can now give a simple characterization of LHA proof terms, which refers to extracted terms and LT, and moreover to the notion of an nc-correct (ordinary) proof term, which is defined as follows:
nc-correct (ordinary) proof terms

- If \( \tau(A) = \varepsilon \), then every ordinary proof term \( M^A \) is nc-correct.
- Every assumption constant (axiom) \( c^A \) and every complete or incomplete assumption variable \( \bar{u}^A \) or \( u^A \) is an nc-correct proof term.
- If \( M^A \) is nc-correct, then so is \( (\lambda \bar{u}^A M)^{A \to B} \) as well as \( (\lambda u^A M)^{A \to oB} \).
- If \( M^A \to B \) and \( N^A \) are nc-correct \((\to \in \{\to, \to o\})\), then so is \( (MN)^B \).
- If \( M^A \) is nc-correct, \( x \notin \text{FV}(B) \) for every \( u^B \in FA(M) \) and moreover \( x \notin \text{FV}([M]) \), then \( (\lambda x M)^{\forall \text{nc} x A} \) is nc-correct.
- If \( M^A \) is nc-correct, and \( \tilde{x} \notin \text{FV}(B) \) for every \( u^B \in FA(M) \), then \( (\lambda \tilde{x} M)^{\forall \tilde{x} A} \) is nc-correct.
- If \( M^{\forall \text{nc} x A} \) is nc-correct and \( r \) is a \( T \)-term, then \( (Mr)^A[x:=r] \) is nc-correct.
- If \( M^{\forall \tilde{x} A} \) is nc-correct and \( r \) is a \( T \)-term, then \( (Mr)^A[\tilde{x}:=r] \) is nc-correct.
Characterization of **LHA** proof terms

**Theorem**

An ordinary proof term $M^A$ is an **LHA** proof term iff $M$ is an nc-correct proof term such that $\llbracket M \rrbracket \in \text{LT}$.

**Proof.**

Induction on $M^A$, assuming that $M$ is an ordinary proof term. We can assume $\tau(A) \neq \varepsilon$.

**Case** $M^A \rightarrow^o B N^A$ with $\tau(A) \neq \varepsilon$. The following are equivalent.

- $MN$ is an **LHA** proof term
- $M$, $N$ are **LHA** proof terms, and the higher type incomplete variables in $\text{CV}(M)$ and $\text{CV}(N)$ are disjoint
- $\llbracket M \rrbracket$ and $\llbracket N \rrbracket$ are **LT**-terms, and the higher type incomplete variables in $\text{FV}(\llbracket M \rrbracket)$ and $\text{FV}(\llbracket N \rrbracket)$ are disjoint
- $\llbracket M \rrbracket \llbracket N \rrbracket$ ($= \llbracket MN \rrbracket$) is an **LT**-term.

The other cases are similar.
Theorem

An ordinary proof term $M^A$ is an LHA proof term iff $M$ is an nc-correct proof term such that $\llbracket M \rrbracket \in \text{LT}$.

Proof.

Induction on $M^A$, assuming that $M$ is an ordinary proof term. We can assume $\tau(A) \neq \varepsilon$.

Case $M^A \to^B N^A$ with $\tau(A) \neq \varepsilon$. The following are equivalent.

- $MN$ is an LHA proof term
- $M$, $N$ are LHA proof terms, and the higher type incomplete variables in $\text{CV}(M)$ and $\text{CV}(N)$ are disjoint
- $\llbracket M \rrbracket$ and $\llbracket N \rrbracket$ are LT-terms, and the higher type incomplete variables in $\text{FV}(\llbracket M \rrbracket)$ and $\text{FV}(\llbracket N \rrbracket)$ are disjoint
- $\llbracket M \rrbracket \llbracket N \rrbracket$ ($= \llbracket MN \rrbracket$) is an LT-term.

The other cases are similar. □
LHA and its provably recursive functions

An $n$-ary numerical function $f$ is called provably recursive in LHA if there is a $\Sigma_1$-formula $G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z)$ denoting the graph of $f$, and a derivation $M_f$ in LHA of

$$\forall \tilde{x}_1, \ldots, \forall \tilde{x}_n \exists z G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z).$$

Here the $\tilde{x}_i$ denote complete or incomplete variables of type $\mathbf{W}$.

Theorem

A function is provably recursive in LHA iff it is computable in polynomial time.

Proof.

$\Rightarrow$. Let $M$ be a derivation in LHA proving a formula of type $\vec{W}^k \rightarrow \vec{W}^l \rightarrow \vec{W}$. Then $[M]$ belongs to $\mathbf{LT}$, hence the claim follows from the Normalization Theorem.

$\Leftarrow$. Use Bellantoni and Cook’s 1992 characterization of the polynomial time computable functions.
**LHA and its provably recursive functions**

An $n$-ary numerical function $f$ is called **provably recursive** in **LHA** if there is a $\Sigma_1$-formula $G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z)$ denoting the graph of $f$, and a derivation $M_f$ in **LHA** of

$$\forall \tilde{x}_1, \ldots, \forall \tilde{x}_n \exists z \, G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z).$$

Here the $\tilde{x}_i$ denote complete or incomplete variables of type $W$.

**Theorem**

A function is provably recursive in **LHA** iff it is computable in polynomial time.

**Proof.**

$\Rightarrow$. Let $M$ be a derivation in **LHA** proving a formula of type $\vec{W}^k \rightarrow \vec{W}^l \rightarrow W$. Then $\llbracket M \rrbracket$ belongs to $LT$, hence the claim follows from the Normalization Theorem.

$\Leftarrow$. Use Bellantoni and Cook’s 1992 characterization of the polynomial time computable functions.
LHA and its provably recursive functions

An $n$-ary numerical function $f$ is called provably recursive in LHA if there is a $\Sigma_1$-formula $G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z)$ denoting the graph of $f$, and a derivation $M_f$ in LHA of

$$\forall \tilde{x}_1, \ldots, \forall \tilde{x}_n \exists z \, G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z).$$

Here the $\tilde{x}_i$ denote complete or incomplete variables of type $W$.

**Theorem**

A function is provably recursive in LHA iff it is computable in polynomial time.

**Proof.**

$\Rightarrow$. Let $M$ be a derivation in LHA proving a formula of type $\vec{W}^k \rightarrow \vec{W}^l \rightarrow W$. Then $[M]$ belongs to LT, hence the claim follows from the Normalization Theorem.

$\Leftarrow$. Use Bellantoni and Cook’s 1992 characterization of the polynomial time computable functions.
LHA and its provably recursive functions

An $n$-ary numerical function $f$ is called provably recursive in LHA if there is a $\Sigma_1$-formula $G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z)$ denoting the graph of $f$, and a derivation $M_f$ in LHA of

$$\forall \tilde{x}_1, \ldots, \forall \tilde{x}_n \exists z \, G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z).$$

Here the $\tilde{x}_i$ denote complete or incomplete variables of type $W$.

Theorem

A function is provably recursive in LHA iff it is computable in polynomial time.

Proof.

$\Rightarrow$. Let $M$ be a derivation in LHA proving a formula of type $\vec{W}^k \rightarrow \vec{W}^l \rightarrow W$. Then $[M]$ belongs to LT, hence the claim follows from the Normalization Theorem.

$\Leftarrow$. Use Bellantoni and Cook’s 1992 characterization of the polynomial time computable functions.
Future work

- Constructive analysis with witnesses of low type level. Type level 1 representation of continuous functions.
- Extraction of reasonable programs is possible.
- The Cauchy-Euler construction of approximate solutions to ODEs as a type level 1 process.
