An interpretation of the Sigma-2 fragment of classical Analysis in System T

Danko Ilik

To cite this version:

Danko Ilik. An interpretation of the Sigma-2 fragment of classical Analysis in System T. 2014. hal-01092411

HAL Id: hal-01092411
https://inria.hal.science/hal-01092411
Preprint submitted on 8 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
AN INTERPRETATION OF THE SIGMA-2 FRAGMENT
OF CLASSICAL ANALYSIS IN SYSTEM T

DANKO ILIK

ABSTRACT. We show that it is possible to define a realizability interpretation for the \(\Sigma^2\)-fragment of classical Analysis using Gödel’s System T only. This supplements a previous result of Schwichtenberg regarding bar recursion at types 0 and 1 by showing how to avoid using bar recursion altogether. Our result is proved via a conservative extension of System T with an operator for composable continuations from the theory of programming languages due to Danvy and Filinski. The fragment of Analysis is therefore essentially constructive, even in presence of the full Axiom of Choice schema: Weak Church’s Rule holds of it in spite of the fact that it is strong enough to refute the formal arithmetical version of Church’s Thesis.

1. Introduction

In the middle of the 20th century, Kurt Gödel showed how to give a computational interpretation and a relative consistency proof of intuitionistic Arithmetic via his System T of equations between functionals definable by primitive recursion (in higher types) \cite{10,11}. Thanks to the fact that the induction axiom intuitionistically proves its own double negation translation, the interpretation also applies to classical Arithmetic. However, since the Axiom of Choice,

\[
\forall x \exists y A(x, y) \rightarrow \exists f \forall x A(x, f(x)),
\]

does not intuitionistically prove its double negation translation, the interpretation does not apply to classical Analysis. Gödel was of course aware of this fact and suggested \cite{21} §2.43 that an extension of System T is needed which can interpret the logical schema,

\[
\neg\neg\forall x (A(x) \lor \neg A(x)),
\]

known as Kuroda’s Conjecture \cite{24}, intuitionistically equivalent to the nowadays better known schema of Double Negation Shift,

\[
\forall x \neg\neg B(x) \rightarrow \neg\forall x B(x).
\]

Gödel must have also been aware of the difficulty involved in giving a computational interpretation to Kuroda’s Conjecture, for already his 1941 lecture at Yale \cite{11} considered the special case when the formula \(A(x)\) is \(\exists y T(x, x, y)\) — where \(T\) is Kleene’s predicate verifying that the Turing machine with code \(x\), when run on input \(x\), terminates with code \(y\) — which directly proves (see \cite{33}) the negation of the formal arithmetical version of Church’s Thesis,

\[
\forall x \neg\neg \exists y \exists u \forall T(e, x, u) \wedge A(x, U(u)).
\]

In spite of that, Spector and Kreisel \cite{31} managed to give a computational interpretation of DNS by extending System T with bar recursion, the computational adequacy of which was shown using a formal version of Brouwer’s principle of
Bar Induction [31, 14]. This approach to extracting computational content from proofs in Analysis via an extension of the primitive recursive System T with a *general recursive* schema, has been much refined over the 50 years since it appeared [22, 32, 20, 21, 8, 1, 3, 4, 29] and has been applied to obtain results in Analysis proper, notably in Kohlenbach’s Proof Mining programme [20].

Nevertheless, as Schwichtenberg showed already in 1979 [27], higher type primitive recursion is closed over the schema of bar recursion at types 0 and 1, and since a previous analysis of Kreisel [31, §12.2] shows that those low types are sufficient for interpreting the classical Axiom of Choice for formulas of the form \( \exists \alpha \in \mathbb{N} \rightarrow \forall x \in \mathbb{N} A_0(\alpha, x) \), \( A_0 \)-quantifier-free, we in fact known that we should need no more than the primitive recursive functionals themselves in order to give a realizability interpretation of the uniformly realizable part of Analysis. Yet, it has remained unclear up to this day how to avoid using general recursive schemata altogether.

One alternative is offered by the use of so called computational side-effects (control operators) from the theory of programming languages. Krivine [23] used a realizability interpretation based on a virtual machine which can execute lambda calculus terms, extended with a control operator and a special machine instruction called “quote”. Herbelin [13] gave a more direct approach based on rewrite rules for a type theory extended with a control operator and a coinductive treatment of the existential quantifier. Both approaches rely on a proper extension of System T that can give a computational interpretation to full classical logic. However, given that there are classically true arithmetic statements that do not have a recursive realizer, it is not clear what the meaning of control operators outside the \( \Sigma_0^1 \)-fragment is.

In this paper, we show that computational side-effects are not needed in the language of realizers, that is, although they are conceptually essential, control operators can be seen as a *meta-mathematical* technique. Proofs of the \( \Sigma_2 \)-fragment of Analysis, with the full Axiom of Choice, are essentially constructive and realizable by System T terms only. The soundness of the interpretation relies on a form of Markov’s Principle (the \textit{Shift} rule), rather than full classical logic [32, 23, 13] or continuity principles [31, 14, 1, 3].

2. **Conservative extension of System T with operators for composable continuations**

The constructive interpretation of proofs of Section 3 is based on Gödel’s System T in its lambda-calculus formulation [33, 28]. Nevertheless, we consider it a conceptual advantage to use an intermediate system, the System T obtained when System T is extended with a control operator for composable continuations, the so called call-by-name variant of the \textit{shift} operator of Danvy and Filinski [7]. The control operator is a key conceptual ingredient that led us to the interpretation [12, 15, 16, 19].

The goal of this technical section is to prove the following conservativity theorem, as well as prove that suitable equations important for Section 5 hold of System T([+]) (Proposition 1).

**Normalization Theorem.** *Every term of System T*[+] maps to a term in normal form of System T.*
The types of System T\(^{(+)\}}\) are \(\mathbb{N}\), functions \(\mathbb{N} \to \mathbb{N}\), functionals (ex. \((\mathbb{N} \to \mathbb{N}) \to \mathbb{N}\)), and Cartesian products of these. Types will be denoted by \(\sigma, \tau\):

\[T \ni \sigma, \tau ::= \mathbb{N} \mid \sigma \to \tau \mid \sigma \ast \tau.\]

Terms of System T\(^{+}\) are constants associated with a sequent \(\sigma_1; \ldots; \sigma_n \vdash \tau\) (also written \(\gamma \vdash \tau\) for \(\gamma\) a finite ordered list of types) which means that a term is of type \(\tau\) and the free variables that appear in the term are of types \(\sigma_1, \ldots, \sigma_n\). The terms, marked in sans-serif face, are defined inductively as follows.

- \(\text{hyp} \frac{\sigma; \gamma}{\sigma}\)
- \(\text{wkn} \frac{\gamma \vdash \sigma}{\tau; \gamma \vdash \sigma}\)
- \(\text{lam} \frac{\sigma; \gamma}{\gamma \vdash \sigma \to \tau}\)
- \(\text{app} \frac{\gamma \vdash \sigma \to \tau}{\gamma \vdash \tau}\)
- \(\text{pair} \frac{\gamma \vdash \sigma}{\gamma \vdash \sigma \ast \tau}\)
- \(\text{fst} \frac{\gamma \vdash \sigma \ast \tau}{\gamma \vdash \sigma}\)
- \(\text{snd} \frac{\gamma \vdash \sigma \ast \tau}{\gamma \vdash \tau}\)
- \(\text{zero} \frac{\gamma \vdash \mathbb{N}}{\gamma \vdash \mathbb{N}}\)
- \(\text{succ} \frac{\gamma \vdash \mathbb{N}}{\gamma \vdash \mathbb{N}}\)
- \(\text{rec} \frac{\gamma \vdash \mathbb{N}}{\gamma \vdash \mathbb{N} \to \sigma \to \sigma} \frac{\gamma \vdash \sigma}{\gamma \vdash \sigma \to \mathbb{N} \to \sigma \to \mathbb{N}}\)
- \(\text{shift} \frac{\mathbb{N} \to \sigma \to \mathbb{N}; \gamma}{\gamma \vdash \sigma}\)

For example, the Ackermann function of type \(\mathbb{N} \to \mathbb{N} \to \mathbb{N}\) in lambda calculus notation,

\[A := \lambda m. \mathcal{R} m (\lambda n.m + 1)(\lambda n'. \lambda u. \lambda n. \mathcal{R} n (u1)(\lambda n'. \lambda w. uw)),\]

where \(\mathcal{R}\) is a constant such that

\[\mathcal{R} 0ab = a\]
\[\mathcal{R}(n + 1)ab = bn(\mathcal{R} nab),\]

is defined using the following term:

\[\text{lam} \frac{(\text{rec hyp}(\text{lam}(\text{succ hyp}))}{(\text{lam \text{hyp}})}\]

\[\text{lam} \frac{(\text{lam \text{hyp}})}{(\text{lam \text{hyp}})}\]

\[\text{lam} \frac{(\text{rec hyp}(\text{app wkn hyp})(\text{succ zero}))}{(\text{lam (lam (app wkn hyp)) hyp))})}\]

This style of presenting lambda calculus terms, known as deBruijn convention, keeps the language of terms first-order, that is, it avoids problems related to handling variables as names that we have to keep track of externally to the system: \(\text{hyp}\) denotes the variable corresponding to the nearest preceding lambda abstraction \(\text{lam}\) or a \(\text{shift}\), \(\text{wkn hyp}\) denotes the second most recent introduced variable, \(\text{wkn wkn hyp}\) the third one, and so on. Seen as natural numbers (\(\text{hyp}\) is 0, \(\text{wkn}\) is successor), these are just so called deBruijn indices.

The terms of System T have a great computational potential, they can compute any higher-type primitive recursive function. There is, however, a subclass of terms
which denote computations that have finished, the so called terms in normal form. These will be sufficient for denoting programs and data extracted from proofs in classical Analysis in Section 3. Technically, the normal forms are known as \( \beta \)-normal in reference to the associated \( \beta \)-reduction relation. Here we give a direct inductive characterization of normal terms

\[
\begin{align*}
\text{e} & \quad \frac{\text{e} \vdash \sigma}{\gamma \vdash \sigma} \\
\text{lam} & \quad \frac{\text{e} \vdash \tau}{\gamma \vdash \sigma \rightarrow \tau} \\
\text{pair} & \quad \frac{\text{e} \vdash \sigma}{\gamma \vdash \sigma \ast \tau} \\
\text{zero} & \quad \frac{}{\gamma \vdash \mathbb{N}} \\
\text{succ} & \quad \frac{}{\gamma \vdash \mathbb{N}} \\
\text{hyp} & \quad \frac{(\sigma; \gamma) \vdash \sigma}{\gamma \vdash \sigma} \\
\text{wkn} & \quad \frac{(\tau; \gamma) \vdash \sigma}{\gamma \vdash \sigma} \\
\text{app} & \quad \frac{\gamma \vdash \tau}{\gamma \vdash \sigma \rightarrow \tau} \\
\text{fst} & \quad \frac{\gamma \vdash \sigma \ast \tau}{\gamma \vdash \sigma} \\
\text{snd} & \quad \frac{\gamma \vdash \sigma \ast \tau}{\gamma \vdash \tau} \\
\text{rec} & \quad \frac{\gamma \vdash \mathbb{N}}{\gamma \vdash \mathbb{N}} \\
\end{align*}
\]

defined at the same time with the so called neutral terms

\[
\begin{align*}
\text{hyp} & \quad \frac{(\sigma; \gamma) \vdash \sigma}{\gamma \vdash \sigma} \\
\text{wkn} & \quad \frac{(\tau; \gamma) \vdash \sigma}{\gamma \vdash \sigma} \\
\text{app} & \quad \frac{\gamma \vdash \tau}{\gamma \vdash \sigma \rightarrow \tau} \\
\text{fst} & \quad \frac{\gamma \vdash \sigma \ast \tau}{\gamma \vdash \sigma} \\
\text{snd} & \quad \frac{\gamma \vdash \sigma \ast \tau}{\gamma \vdash \tau} \\
\text{rec} & \quad \frac{\gamma \vdash \mathbb{N}}{\gamma \vdash \mathbb{N}} \\
\end{align*}
\]

Neutral terms correspond to computation that are “blocked”: neutral terms are those that contain open/free variables that block a \( \beta \)-reduction step from happening.

A property that will be later used (proof of Corollary 1) follows directly from the shape of normal forms: any closed normal term of type \( \mathbb{N} \), i.e. a term of type \( \emptyset \vdash \mathbb{N} \), is actually a numeral, that is, built only from zero and succ-terms. This follows because closed normal terms cannot be neutral: a neutral term necessarily has at least one free variable.

We are now ready to state our first theorem precisely.

**Normalization Theorem.** There is a normalization procedure \( \downarrow \llbracket - \rrbracket \) such that, for every term \( p \) of System \( T^+ \) of type \( \gamma \vdash \tau \), the term \( \downarrow \llbracket p \rrbracket \) is a term in normal form of System \( T \) of the same type \( \gamma \vdash \tau \).

This theorem is not a standard fact from the theory of lambda calculus with control operators. As a matter of fact, we present the first proof that control operators can be completely eliminated from System \( T \).

Specialists will recognize the proof method as a normalization-by-evaluation [5] or type-directed partial evaluation [6] argument. Nevertheless, the addition of control operators requires us to perform the proof in so called continuation-passing style; a similar technique has been used to provide a constructive completeness proof for non-minimal intuitionistic logic when disjunction and the existential quantifier are present [15, 17].

The proof is constructive and can be formalized in a suitable predicative meta-theory such as Martin-Löf Type Theory (for a modern formulation see [34]). At her convenience, the reader may find this full mechanization of the proof in Agda notation in [38]; one can also use this machine-checked constructive proof directly in order to compute, for example, that the term for the Ackermann function \( A(3,2) \) really evaluates to \( \text{succ} \ldots \text{succ} \text{zero} \) 29 times.

**Proof of Normalization Theorem.** Our goal is to define an evaluation function,

\[
\llbracket (-) \rrbracket : \gamma \vdash \sigma \Rightarrow \gamma \vdash \sigma
\]
that maps a term \( p \) of type \( \gamma \vdash \sigma \) to a forcing set \( \gamma \models \sigma \), together with a reification function,

\[
\downarrow (-) : \gamma \models \sigma \Rightarrow \gamma \models_! \sigma ,
\]

that extracts normal forms (without shift!) from the forcing set. Composing the two function, \( p \mapsto \downarrow [p] \), gives a normalization procedure. (The notation \( a \ a \ a \mapsto \downarrow \) will be a compact form of \( a \mapsto \downarrow \).)

We first need a precise way to speak about extensions of type contexts \( \gamma \) (imposed by the fact that we want to normalize potentially open terms). This is formalized by the initial segment, or prefix, preorder \( \geq \) as follows.

\[
\geq_{\mathsf{refl}} \gg \geq \quad \geq_{\mathsf{cons}} \gg (\sigma; \gamma_2) \gg \gamma_1
\]

For example, the proof of \( \tau_1; \tau_2; \gamma \geq \gamma \) will be denoted by \( \geq_{\mathsf{cons}} \geq_{\mathsf{refl}} \geq_{\mathsf{refl}} \).

The transitivity of \( \geq \) is proven as the (right-associative) operation \( (\cdot) \cdot \), defined by recursion on the construction of the proofs of \( \gamma_3 \geq \gamma_2 \) and \( \gamma_2 \geq \gamma_1 \).

\[
(\cdot) \cdot (\cdot) : \gamma_3 \geq \gamma_2 \Rightarrow \gamma_2 \geq \gamma_1 \Rightarrow \gamma_3 \geq \gamma_1
\]

\[
\geq_{\mathsf{refl}} \cdot \geq_{\mathsf{refl}} = \geq_{\mathsf{refl}} \geq_{\mathsf{refl}} \cdot \geq_{\mathsf{refl}} = \geq_{\mathsf{refl}}\geq_{\mathsf{refl}}
\]

In this definition, and henceforth, the notation \( \geq_n \) will be used to denote a (hypothetical) proof of \( \gamma_n \geq \gamma_k \). For example, given a proof \( \geq_2 \) of \( \gamma_2 \geq \sigma; \gamma_1 \), one can prove \( \gamma_2 \geq \gamma_1 \) by the denotation \( \geq_2 \cdot \geq_{\mathsf{refl}} \).

We can now define precisely the forcing set \( \gamma \models \sigma \). This is done simultaneously with the strong forcing set \( \gamma \models_! \sigma \), itself defined inductively following the construction of the type \( \sigma \).

\[
\gamma \models \sigma = \forall \gamma_1 \geq \gamma (\forall \gamma_2 \geq \gamma_1 (\gamma_2 \models_! \sigma \Rightarrow \gamma_2 \models_! N) \Rightarrow \gamma_1 \models_! N)
\]

\[
\gamma \models_! N = \gamma \models_! N
\]

\[
\gamma \models_! (\sigma \to \tau) = \forall \gamma' \geq \gamma (\gamma' \models_! \sigma \Rightarrow \gamma' \models_! \tau)
\]

\[
\gamma \models_! (\sigma \ast \tau) = \gamma \models_! \sigma \times \gamma \models_! \tau
\]

We will also need \( \gamma' \models_! \gamma \), the component-wise extension of the forcing relation defined as follows (\( \boxempty \) denotes the empty context).

\[
\gamma' \models_! \boxempty = \top
\]

\[
\gamma' \models_! (\sigma; \gamma) = (\gamma' \models_! \sigma) \times (\gamma' \models_! \gamma)
\]

The first equation defines the forcing of the empty context to be the singleton set \( \top \), whose unique inhabitant is denoted \( \bot \). The symbol \( \times \) constructs a Cartesian product i.e. pair type in the ambient type theory; the components of a Cartesian product can be accessed by the projection operations \( \text{proj}_1 \) and \( \text{proj}_2 \).

---

1The terminology “forcing” comes from similarity of our construction with Kripke models which does not intentionally refer to the forcing from Set Theory.
The following operations (lemmas) show that ⊢ \( r \), \( \vdash e \), \( \models s \), \( \models \) are all monotone with respect to the prefix preorder. The last two operations are defined by induction on types and contexts, respectively.

\[
\begin{align*}
\Gamma (\cdot) : \gamma_2 \geq \gamma_1 \Rightarrow \gamma_1 \not\vdash \sigma \Rightarrow \gamma_2 \not\vdash \sigma \\
\Gamma H \geq \text{refl} = H \\
\Gamma H \geq \text{cons} \geq 2 = e(\text{wkn}(\Gamma H \geq 2)) \\
\end{align*}
\]

\[
\begin{align*}
\Upsilon (\cdot) : \gamma_2 \geq \gamma_1 \Rightarrow \gamma_1 \not\vDash \sigma \Rightarrow \gamma_2 \not\vDash \sigma \\
\Upsilon H \geq \text{refl} = H \\
\Upsilon H \geq \text{cons} \geq 2 = \text{wkn}(\Upsilon H \geq 2)) \\
\end{align*}
\]

\[
\begin{align*}
[\cdot] (\cdot) : \gamma_2 \geq \gamma_1 \Rightarrow \gamma_1 \vDash \sigma \Rightarrow \gamma_2 \vDash \sigma \\
[H] \geq \text{refl} = H \\
[H] \geq \text{cons} \geq 2 = \text{wkn}(H \geq 2) \\
\end{align*}
\]

\[
\begin{align*}
[\cdot] (\cdot) : \gamma_2 \geq \gamma_1 \Rightarrow \gamma_1 \vDash \gamma \Rightarrow \gamma_2 \vDash \gamma \\
[H] \geq \text{refl} = H \\
[H] \geq \text{cons} \geq 2 = \text{wkn}(H \geq 2) \\
\end{align*}
\]

Finally, we also need lemmas relating the forcing sets and derivability, the return (\( \eta \)) and run (\( \mu \)) operations.

\[
\begin{align*}
\eta (\cdot) : \gamma \not\vDash \sigma \Rightarrow \gamma \vDash \sigma \\
\eta H = \geq 1 \kappa \mapsto \kappa \geq \text{refl} [H] \geq 1 \\
\mu (\cdot) : \gamma \vDash N \Rightarrow \gamma \not\vDash N \\
\mu H = H \geq \text{refl} (\geq 1 \alpha \mapsto \alpha) \\
\end{align*}
\]

Return shows that we can always lift a member of the strong forcing set to a member of the forcing set. Run shows that, whenever we have a member of a set forcing type \( N \), we can actually obtain a term of System T in normal form from it; note that by definition the sets \( \gamma \not\vDash N \) and \( \gamma \not\vDash N \) are the same.
The \textit{reify} function \( \downarrow (-) \) shows that we can actually run any forcing set, for any type \( \sigma \), and not just for \( \sigma = \mathbb{N} \). It is defined by induction on the type.

\[
\begin{align*}
\gamma \downarrow^\sigma (-) : \gamma \upharpoonright \sigma \Rightarrow \gamma \uparrow \sigma \\
\gamma \downarrow^\mathbb{N} H = \mu H \\
\gamma \downarrow^{\sigma \rightarrow \tau} H = \text{lam}(\gamma) \\
(\geq 1 \kappa \mapsto H(\geq 1 \cdot \geq \text{cons} \geq \text{refl})) \\
(\geq 2 \phi \mapsto \phi \geq \text{refl} (\{\sigma \uparrow \sigma \uparrow \text{hyp} \geq \geq 1 \geq \text{refl} \\
(\geq 3 \mapsto \kappa(\geq 3 \cdot \geq 2)))))
\end{align*}
\]

\[
\begin{align*}
\gamma \downarrow^{\sigma \ast \tau} H = \text{pair} \\
\alpha \uparrow^\sigma (\geq 1 \kappa \mapsto H(\geq 1 \cdot \geq \text{refl} (\geq 3 \mapsto \kappa(\geq 3 \cdot \geq 2))))
\end{align*}
\]

The case of arrow type forces us to define, at the same time with the \textit{reify} function, the \textit{reflect} function \( \uparrow \) (-):

\[
\begin{align*}
\gamma \uparrow^\sigma (-) : \gamma \subseteq \sigma \Rightarrow \gamma \uparrow \sigma \\
\gamma \uparrow^\mathbb{N} p = \eta(e_p) \\
\gamma \uparrow^{\sigma \rightarrow \tau} p = \eta(\geq 2 \alpha \mapsto \gamma \uparrow \text{app}(\{p, \geq 2, \gamma \uparrow \sigma \alpha) \\
\gamma \uparrow^{\sigma \ast \tau} p = \eta(\gamma \uparrow \text{fst} p, \gamma \uparrow \text{snd} p)
\end{align*}
\]

Note that this function needs as domain only the neutral terms.

Finally, we are ready to define the \textit{evaluation} function \( \llbracket - \rrbracket \) that constructs a member of the forcing set for any input term of System \textit{T} \(+\). The definition is by
recursion on the construction of the term $p$.

\[
\gamma \vdash (-) : \gamma \vdash \sigma \Rightarrow \forall \gamma' \vdash \gamma (\gamma' \vdash \sigma)
\]

\[
\text{[hyp]} = \text{proj}_1 \rho
\]

\[
\text{[wkn]} = \text{proj}_2 \rho
\]

\[
\text{[lam]} = \eta(\gamma \vdash \alpha \rightarrow [\rho]_{\langle \alpha, \gamma \rangle})
\]

\[
\text{[app]} = \geq 1 \kappa \vdash [p] \geq 1
\]

\[
(\geq 2 \kappa \vdash \phi \rightarrow \geq \text{refl} (\eta \geq \text{refl} (\geq 3 \kappa \geq \text{refl} (\geq 3 \kappa)))
\]

\[
\text{[pair]} = \eta([p], [q])
\]

\[
\text{[fst]} = \geq 1 \kappa \vdash \geq 1 \kappa \vdash \text{proj}_1 \alpha \geq \text{refl} (\geq 3 \kappa \leq \text{refl} (\geq 3 \kappa))
\]

\[
\text{[snd]} = \geq 1 \kappa \vdash \geq 1 \kappa \vdash \text{proj}_2 \alpha \geq \text{refl} (\geq 3 \kappa \leq \text{refl} (\geq 3 \kappa))
\]

\[
\text{[shift]} = \eta(\mu)\eta(\geq 2 \nu \rightarrow \eta(\geq 3 \alpha \rightarrow \eta(\alpha \geq \text{refl} (\geq 3 \kappa \geq \text{refl} (\geq 3 \kappa))))).
\]

\[
\text{[zero]} = \eta(\text{zero})
\]

\[
\text{[succ]} = \eta(\text{succ}) ([\mu]_{\rho})
\]

\[
\text{[rec]} = \geq 1 \kappa \vdash \geq 1 \kappa \vdash \nu \rightarrow \text{rec}(e, \downarrow [a]_{\rho}, \downarrow [f]_{\rho})
\]

Note that the argument $\rho$ is of type $\gamma' \vdash \gamma$.

For $\gamma = \gamma$, such a $\rho$ can always be constructed by reflecting the term hyp:

\[
\uparrow \gamma : \gamma \vdash \gamma
\]

\[
\uparrow [] = \text{tt}
\]

\[
\uparrow \sigma : \emptyset \vdash \text{hyp}, \text{[hyp]} \geq \text{refl}
\]

We have therefore shown that, given $p : \gamma \vdash \sigma$ of System $T^+$, there is a term in normal form $[p]_{\rho} : \gamma \vdash \sigma$ of System T, for every $\rho$, and in particular one such term is $[p]_{\rho_{\gamma}}$.

The following proposition characterizes the equational theory generated by the normalization procedure. It will be used in the proof of Soundness Theorem of Section 3. This has also been machine checked and is available in Agda notation from 18.
Proposition 1. The following definitional equalities hold,

\( \downarrow [wkn p]_{\alpha, \rho} = \downarrow [p]_{\rho} \quad \text{for} \ \alpha \in \tau; \gamma \vdash \tau \)

\( \downarrow [hyp]_{\alpha, \rho} = \downarrow \alpha \quad \text{for} \ \alpha \in \gamma \vdash \tau \)

\( \downarrow [fst pair(p, q)]_{\rho} = \downarrow [p]_{\rho} \)

\( \downarrow [snd pair(p, q)]_{\rho} = \downarrow [q]_{\rho} \)

\( \downarrow [app(lam p, q)]_{\rho} = \downarrow [p]_{[q], \rho} \)

\( \downarrow [rec(zero, p, q)]_{\rho} = \downarrow [p]_{\rho} \)

\( \downarrow [rec(succ r, p, q)]_{\rho} = \downarrow [app(app(q, r), rec(r, p, q))]_{\rho} \)

\( \downarrow^N [shift p]_{\rho} = \downarrow^N [p]_{\phi, \rho} \)

\( \downarrow^N [app(app(hyp, x), y)]_{\phi, \rho} = \downarrow^N [y]_{\phi, \rho} \)

where for the last two equations,

\( \phi := \eta(\geq 2 \nu \mapsto \eta(\geq 3 \alpha \mapsto \eta(\mu \alpha ))) \),

and \( x, y : \mathbb{N} \rightarrow \mathbb{N} \vdash \mathbb{N} \).

Proof. Equations (1)–(7) follow from the ones that hold already of the \( \downarrow (-) \) function. This is because, as an argument to the \( \downarrow (-) \) function, the evaluation function is always applied to some \( \geq 1 \) and \( \kappa \).

\[
[p]_{\rho} \geq_1 \kappa = [q]_{\rho} \geq_1 \kappa
\]

\[
[wkn p]_{\alpha, \rho} \geq_1 \kappa = [p]_{\rho} \geq_1 \kappa
\]

\[
[hyp]_{\alpha, \rho} \geq_1 \kappa = \alpha \geq_1 \kappa
\]

\[
[fst pair(p, q)]_{\rho} \geq_1 \kappa = [p]_{\rho} \geq_1 \kappa
\]

\[
[snd pair(p, q)]_{\rho} \geq_1 \kappa = [q]_{\rho} \geq_1 \kappa
\]

\[
[app(lam p, q)]_{\rho} \geq_1 \kappa = [p]_{[q], \rho} \geq_1 \kappa
\]

\[
[rec(zero, p, q)]_{\rho} \geq_1 \kappa = [p]_{\rho} \geq_1 \kappa
\]

\[
[rec(succ r, p, q)]_{\rho} \geq_1 \kappa = [app(app(q, r), rec(r, p, q))]_{\rho} \geq_1 \kappa
\]

These equations come out by unfolding the definition and occasionally using an \( \eta \)-equality step of the form \( (\alpha \mapsto \phi \alpha) = \phi \).

Equations (8)–(9) also follow by definition, this time reification being applied for only one concrete type, \( \mathbb{N} \).

3. A modified realizability interpretation of Analysis

By a logical theory sufficient to formalize proofs of Analysis we have in mind the System HA\(^{++}\) of Figure 1 together with the full axiom of choice schema AC = \( \bigcup_{\tau \in T} AC^{\tau} \),

\[ (AC^{\tau}) \quad \forall x^{\sigma} \exists y A(x, y) \rightarrow \exists^{\sigma \rightarrow \tau} f \forall x^{\sigma} A(x, f(x)). \]

This formulation of the axiom is strictly stronger than the Axiom of Dependent Choices which usually treated in the context of realizability interpretations for Analysis. One can also consider the axioms of Figure 2 to be part of HA\(^{++}\). It is known that due to their logical form, the proof interpretation that we are going to
employ has no modifying effect on them. On the other hand, they can indicate in which sense the realizability model supports extensionality.

HA\textsuperscript{ω+} is a first-order, predicate logic which is multi-sorted, that is, has variables and quantifiers that range over the types of System T. The system is “minimal” in the sense of Schwichtenberg [28] – one wants to know that the method works even if we do not have a special treatment of the absurdity symbol \(\perp\); one can work with any fixed formula \(N\) as if it were \(\perp\).

HA\textsuperscript{ω+} has explicit rules for dealing with the existential quantifier (disjunction has not been included for the sake of simplicity). This, together with the special \textsc{Shift} rule, justifies the plus superscript \(^+\) in the name. The rule \textsc{Shift} is a general form of the more usual double-negation elimination rule, restricted to \(\Sigma_2\)-formulas, that is precisely suitable for a simple proof of the Soundness Theorem. It has previously been used by Nakata and the author in a semi-classical logic context [16, 19].

The class of \(\Sigma_2\)-formulas consists of formulas \(S\) of the following form,

\[
S ::= N \mid \exists x^\tau N \mid N \rightarrow S \mid N \land S \mid S \land N,
\]

where \(N\) stands for so called \textit{computationally irrelevant} formulas [29], defined inductively by

\[
N ::= P \mid N \land N \mid N \lor N \mid \forall x^\tau N \mid A \rightarrow N,
\]

where \(P\) stands for prime formulas (predicates) and \(A\) has no restrictions on the form.

**Figure 1:** A natural deduction system for the theory HA\textsuperscript{ω+}
As our interpretation we will use a version of Kreisel’s so called modified realizability interpretation [33, 20], optimised similarly to the work of Berger, Buchholz, Schwichtenberg, and Seisenberger [2, 29]. However, the key addition is that the realizing terms will also be computed by normalized instances of the shift term.

Definition 1. Given a context $\Gamma$ and an interpretation of hypotheses $\rho : |\Gamma| \vdash |\Gamma|$, the modified realizability interpretation “$\mathbf{mr} A$” of a formula $A$ by a term $p$ of type $|\Gamma| \vdash |A|$ of System T is defined by the following formula translation,

\[
\begin{align*}
\mathbf{mr} N &:= N \\
\mathbf{mr} N \land B &:= N \land (\mathbf{mr} B) \\
\mathbf{mr} A \land N &:= (\mathbf{mr} A) \land N \\
\mathbf{mr} A \land B &:= (\downarrow[[\text{fst} p]]_\rho \mathbf{mr} A) \land (\downarrow[[\text{snd} p]]_\rho \mathbf{mr} B) \\
\mathbf{mr} N \rightarrow B &:= N \rightarrow (\mathbf{mr} B) \\
\mathbf{mr} A \rightarrow B &:= \forall x(\downarrow[[x]]_\rho \mathbf{mr} A) \rightarrow (\downarrow[[\text{app}(p, x)]]_\rho \mathbf{mr} B)) \\
\mathbf{mr} \forall x^\tau A(x) &:= \forall x^\tau (\downarrow[[\text{app}(p, x)]]_\rho \mathbf{mr} A(x)) \\
\mathbf{mr} \exists x^\tau N(x) &:= N(p) \\
\mathbf{mr} \exists x^\tau A(x) &:= \downarrow[[\text{snd} p]]_\rho \mathbf{mr} A(\downarrow[[\text{fst} p]]_\rho),
\end{align*}
\]

in which $N$ denotes a computationally irrelevant formula, and where the type $|A|$ of the realizing term $p$ is computed as follows:

\[
\begin{align*}
|N| &:= N \\
|N \land B| &:= |B| \\
|A \land N| &:= |A| \\
|A \land B| &:= |A| \ast |B| \\
|N \rightarrow B| &:= |B| \\
|A \rightarrow B| &:= |A| \rightarrow |B| \\
|\forall x^\tau A| &:= \tau \rightarrow |A| \\
|\exists x^\tau N| &:= \tau \\
|\exists x^\tau A| &:= \tau \ast |A|
\end{align*}
\]

The map $| \cdot |$ is extended to contexts $\Gamma$ by $|C_1, \ldots, C_n| := |C_1| \ast \cdots \ast |C_n|$.

Note that $\Sigma_2$-formulae are exactly those ones that are realized by a term of type $N$. 
Our main result is the following one.

**Soundness Theorem.** If HA⁺⁺ + AC proves \( C_1, C_2, \ldots, C_n \vdash A \), and \( A \) is computationally relevant, then there exists a term \( p \) of System T⁺⁺ such that HA⁺⁺ alone proves that, for every \( \rho: |C_1|, |C_2|, \ldots, |C_n| \vdash |C_1|, |C_2|, \ldots, |C_n| \),

\[
\downarrow [\text{hyp}]_{\rho} \quad \text{mr} \quad C_1, \downarrow [\text{wkn hyp}]_{\rho} \quad \text{mr} \quad C_2, \ldots, \downarrow [\text{wkn}^n \text{ hyp}]_{\rho} \quad \text{mr} \quad C_n \vdash \downarrow [p]_{\rho} \quad \text{mr} \quad A.
\]

**Proof.** The proof is by induction on the derivation of \( C_1, C_2, \ldots, C_n \vdash A \) and provides the same realizing terms as Kreisel’s modified realizability interpretation of HA⁺⁺ + AC. The additional rule of **Shift** (treated in more detail below) is realized via the **shift** term of System T⁺⁺, nevertheless normalized to System T using \( \downarrow [\ ] \).

We will denote by \( p \) realizing terms provided by the induction hypothesis for each proof rule. If there are two induction hypotheses, the term corresponding to the second, right premise of the proof rule will be denoted by \( q \).

As a general guide, the elimination rules are enough to prove their own soundness, while the introduction rules and the rules AC, WKN, IND, and **Shift** also need to use the definitional equalities of Proposition 1.

In general, the axiom \( +\text{AC} \) is realized by the term

\[
\text{lam pair}(\text{lam app}(\text{fst wkn hyp}, \text{hyp}), \text{lam app}(\text{snd wkn hyp}, \text{hyp})).
\]

When the formula \( A(x, y) \) is computationally irrelevant, the realizer is the term \( \text{lam hyp} \). The proof in both cases is a trivial intuitionistic implication and does not require AC⁺⁺⁺ itself. Equations \([11]-[15]\) of Proposition 1 are nevertheless used.

**AX** is realized by \( \text{hyp} \).

**WKN** is realized by \( \text{wkn p} \) and verified using equation \([1]\) and \([2]\).

The general case of \( \to_1 \), when \( A \) from \( A \to B \) is computationally relevant, is realized by \( \text{lam p} \) and verified using equations \([5]\), \([2]\), and \([1]\). The induction hypothesis needs to be used with the context \( \rho := ([x]_\rho, \rho) \), where \( x \) comes from the unfolding of the mr-definition for implication. The special case, when \( A \) is computationally irrelevant, is rather realized by the term \( p \) only.

When \( A \) from \( A \to B \) is computationally relevant, the case \( \to_E \) is realized by \( \text{app}(p, q) \). When \( A \) is irrelevant, the realizing term is just \( p \).

\( \land_1^1 \) is realized using \( \text{fst p} \), in general, while in the case where one of the conjuncts of \( A \land B \) is irrelevant, the realizer is just \( p \).

\( \land_2^1 \) is realized using \( \text{snd p} \), in general, while in the case where one of the conjuncts of \( A \land B \) is irrelevant, the realizer is just \( p \).

In general, \( \land_1 \) is realized using \( \text{pair}(p, q) \) and verified via equations \([3]\) and \([4]\).

When \( A \) from \( A \land B \) is irrelevant, the realizer is \( q \), while when \( B \) is irrelevant, the realizer is \( p \).

In general, \( \exists_1 \) is realized by \( \text{pair}(p, q) \) and verified via equations \([3]\) and \([4]\).

When \( A(x) \) from \( \exists x. A(x) \) is computationally irrelevant, then the realizer is \( r \), the witnessing term.

\( \exists_1 \) is realized by \( \text{app}(\text{app}(q, \text{fst p}), \text{snd p}) \), in general. When \( A(x) \) from \( \exists x. A(x) \) is computationally irrelevant, then the realizer is \( \text{app}(q, p) \).

\( \forall_1 \) is realized by \( \text{lam p} \). For verification, it is necessary to apply equation \([5]\) and to use the induction hypothesis with context \( \rho := ([x]_\rho, \rho) \).

\( \forall_1 \) is realized by \( \text{app}(p, r) \).

**IND** is realized by \( \text{lam rec}(\text{hyp}, p, q) \) and using equations \([6]\) and \([4]\).
Shift is realized by shift_\rho (normalized to System T using \downarrow [-]). The goal is to prove
\[
\downarrow \operatorname{hyp}_\rho \operatorname{mr} C_1, \ldots, \downarrow \operatorname{wkn}^n \operatorname{hyp}_\rho \operatorname{mr} C_n \vdash \downarrow \operatorname{shift}_\rho \operatorname{mr} A(r).
\]
Using equation (3), we obtain \phi and the goal becomes
\[
\downarrow \operatorname{hyp}_\rho \operatorname{mr} C_1, \ldots, \downarrow \operatorname{wkn}^n \operatorname{hyp}_\rho \operatorname{mr} C_n \vdash \downarrow [p]_{\phi, \rho} \operatorname{mr} A(r).
\]
We can now use the induction hypothesis with \rho := (\phi, \rho),
\[
\downarrow \operatorname{hyp}_\rho \operatorname{mr} \forall x^n (A(x) \rightarrow S(x)), \downarrow \operatorname{wkn} \operatorname{hyp}_\rho \operatorname{mr} C_1, \ldots,
\downarrow \operatorname{wkn}^{n+1} \operatorname{hyp}_\rho \operatorname{mr} C_n \vdash \downarrow [p]_{\phi, \rho} \operatorname{mr} S(r).
\]
Thanks to equation (11), the induction hypothesis becomes
\[
\downarrow \operatorname{hyp}_\rho \operatorname{mr} \forall x^n (A(x) \rightarrow S(x)), \downarrow \operatorname{hyp}_\rho \operatorname{mr} C_1, \ldots,
\downarrow \operatorname{wkn}^n \operatorname{hyp}_\rho \operatorname{mr} C_n \vdash \downarrow [p]_{\phi, \rho} \operatorname{mr} S(r).
\]
Finally, thanks to equation (12), we can finish the proof by applying the Shift rule for:
\[
S'(x, y) := \downarrow [y]_{\phi, \rho} \operatorname{mr} S(x)
\]
\[
A'(x, y) := \downarrow [y]_{\phi, \rho} \operatorname{mr} A(x).
\]

\[\square\]

Remark 1. The Shift case in the proof of the Soundness Theorem only uses the case where shift is reified at type \(N\). This use does not exhaust the possibilities of the realizability model. For example, one can prove the soundness of the Shift rule for \(A = A_1 \land A_2\) or \(A = \exists z \in A_2(z)\), when \(A_i \in \Sigma_2\), by using the equations
\[
\downarrow^{\pi_\sigma} [p]_\rho = \text{pair}(\downarrow^\pi [\text{fst}_\rho p], \downarrow^\pi [\text{snd}_\rho p])
\]
\[
\downarrow^{N^{\pi_\sigma}} [\text{fst}_\rho p]_\rho = \downarrow^N [p]_{\phi_1, \rho}
\]
\[
\downarrow^{N^{\pi_\sigma}} [\text{snd}_\rho p]_\rho = \downarrow^N [p]_{\phi_2, \rho}
\]
\[
\downarrow^N [\text{app}(\text{app}(\text{hyp}, x), y)]_{\phi_1, \rho} = \downarrow^N [\text{fst}_\rho y]_{\phi_1, \rho}
\]
\[
\downarrow^N [\text{app}(\text{app}(\text{hyp}, x), y)]_{\phi_2, \rho} = \downarrow^N [\text{snd}_\rho y]_{\phi_2, \rho}
\]
where
\[x : N \rightarrow N \times N \rightarrow N; \gamma \vdash N\]
\[y : N \rightarrow N \times N \rightarrow N; \gamma \vdash N \times N\]
\[\phi_1 := \eta(\geq_2 \nu \rightarrow \eta(\geq_3 \alpha \rightarrow \eta(\geq_4 \gamma \rightarrow (\text{proj}_1 \gamma))))\]
\[\phi_2 := \eta(\geq_2 \nu \rightarrow \eta(\geq_3 \alpha \rightarrow \eta(\geq_4 \gamma \rightarrow (\text{proj}_2 \gamma))))\]

The induction hypothesis needs to be used twice, once for \(\phi_1\) and once for \(\phi_2\).

Similar equations hold for function types:
\[
\gamma \downarrow^N [\text{app}(\text{shift}_\rho, z)]_\rho = \downarrow^N [\text{app}(\text{shift}_\rho, z)]_\rho
\]
\[
\phi_3 := \eta(\geq_2 \nu \rightarrow \eta(\geq_3 \alpha \rightarrow \eta(\geq_4 \gamma \rightarrow \mu(\gamma \geq_\text{refl} [z]_{\gamma, \rho} z' \geq_3 z'))))
\]

Nevertheless, it does not appear to be possible to prove the soundness of the Shift rule even for the case \(|A| = N \rightarrow N\) in general, since it is well known that there are already classically true \(\Sigma_3^0\)-formulas which do not have a recursive realizer.
Still, it may be the case that the realizability model can be used to give a sound computational interpretation of particular $\Sigma_3^0$ (or more complex) formulas.

**Corollary 1.** The $\Sigma_2$-fragment of classical Analysis satisfies the Existence Property.

Given a derivation of $\Gamma \vdash \exists x^\tau A(x)$, there exists a term $p$ of type $\tau$ of System $T$ such that $\Gamma \vdash A(p)$.

and, consequently, the Weak Church’s Rule,

Given a (closed) derivation of $\emptyset \vdash \forall x^N \exists y^N A(x, y)$, there exists a total recursive function $f : \mathbb{N} \to \mathbb{N}$ such that, for all $n \in \mathbb{N}$, we have that $\emptyset \vdash A(\overline{m}, \overline{m})$, where $\overline{m}$ denotes the term $\underbrace{\text{succ} \cdots \text{succ} \text{zero}}_m$.

**Proof.** The proof method is not new (see Corollary 5.24 of [20] and paragraph 1.11.7 of [33]). If the formula $A$ is of the class $\Gamma_1$ [33, 20],

$$\Gamma_1 \ni G := N | G \land G | \forall x G | \exists x G | S \rightarrow G,$$

(where $S$ is a $\Sigma_2$-formula and $N$ is computationally irrelevant) then already intuitionistic logic shows that $\vdash (p \text{ mrt } A) \rightarrow A$. If $A$ is outside this class, one first needs to define the “with truth” variant of modified realizability in which one replaces the clause for implication of mr-interpretation of Definition 1 by

$$p \text{ mrt } A \rightarrow B := \forall x ([\downarrow [x]_p, \text{ mrt } A] \rightarrow [\downarrow [\text{app}(p, x)]_p, \text{ mrt } B]) \land (A \rightarrow B).$$

The Soundness Theorem is provable for mrt with the same realizing terms, but now we also have, for any formula $A$, $(p \text{ mrt } A) \rightarrow A$. This directly implies the Existence Property.

For the special case when $\Gamma = \emptyset$ and $\tau = \mathbb{N}$, we get the Numerical Existence Property:

Given a (closed) derivation of $\emptyset \vdash \exists y^N A(y)$, there exists $n : \mathbb{N}$ such that $\emptyset \vdash A(\overline{n})$.

This follows from the fact that a closed derivation has a realizer that is a closed term (does not have non-bound $\text{hyp}$ subterms). Since the realizer is necessarily in normal form, and since it is not neutral (all neutral terms have at least one non-bound occurrence of $\text{hyp}$), then the realizer must be of the required form $\underbrace{\text{succ} \cdots \text{succ} \text{zero}}_n$.

To show the Weak Church’s Rule, we use elementary Recursion Theory. Like all theories over countable languages, HA$_{\omega+1}+\text{AC}$ is recursively axiomatizable, that is, there exists a recursive predicate $\text{Proof}(k, l)$ formalizing the fact that $k \in \mathbb{N}$ is a code for a derivation of the formula coded by $l \in \mathbb{N}$.

Let $g(n) = \min_{m} \text{ Proof}(j_1 m, (\Gamma A(\overline{m}, \underline{j_2 m})))$, where $j_1$ and $j_2$ are the projections of some surjective pairing function. As defined, $g$ is a partial recursive function.

Now, given $\emptyset \vdash \forall x^N \exists y^N A(x, y)$ and $n \in \mathbb{N}$, we obtain $\emptyset \vdash \exists y^N A(\overline{n}, y)$, and by the Numerical Existence Property we obtain $m \in \mathbb{N}$ such that $\vdash A(\overline{m}, \overline{m})$. We proved that, for every $n$, there exists $m$ such that $\emptyset \vdash A(\overline{m}, \overline{m})$ which shows that the function $g$ is total recursive. We may now take $f(n) := j_2(g(n))$ and by definition we have that, for any $n$, $\emptyset \vdash A(\overline{m}, f(n))$. \[\square\]

Note that the class $\Sigma_2$ includes the following schemata,

$$(\text{MP}) \quad \neg_N \neg_N \exists x^\tau M(x) \rightarrow \exists x^\tau M(x),$$
\[ \forall x \neg N \neg N A(x) \rightarrow \neg N \forall x N A(x), \]

where \( M, N \) denote computationally irrelevant formulas and \( \neg N A \) denotes negation in minimal logic, that is \( A \rightarrow N \) for a fixed \( N \).

The Existence Property implies that principles like MP and DNS can justly be considered as constructive even in presence of induction and the full Axiom of Choice, partly extending previous works [33, 8, 30, 12, 16]. Similar conclusions follow from the work of Rand Moschovakis that uses a version of Kleene’s general-recursive realizability [26].

Weak Church’s Rule seems to justify why, even constructively, \( CT_0 \) deserves the name “the false Church’s Thesis” [25].

Acknowledgments

This work was funded by Kurt Gödel Research Prize Fellowship 2011 and ERC Advanced Grant ProofCert. I would also like to thank Martín Escardó, Jaime Gaspar, Keiko Nakata, and Dirk Pattinson for comments on an earlier version of this paper, and Dale Miller for providing scientific liberty.

References


