An Arithmetic for non-Size-Increasing Polynomial-Time Computation

Klaus Aehlig Ulrich Berger Martin Hofmann

Helmut Schwichtenberg

The Question

We want to propose an answer to the question

$$\frac{\text{Heyting arithmetic}}{\text{Gödel's } T} = \frac{?}{\text{Hofmann's } \texttt{Ptime} \text{ system}}$$

Of course we first have to explain what that means:

- Gödel's T
- Heyting Arithmetic
- the relation between them
- Martin Hofmann's system for polynomial time

Based on:

- Ulrich Berger, Martin Hofmann, Helmut Schwichtenberg. A logic for nonincreasing PTIME computation. (unpublished draft)
- Klaus Aehlig. An arithmetic for non-size-increasing polynomial-time computation (ICC 2000).

Gödel's T

- simply typed λ-calculus with pairs
  - types: o, $\rho \to \sigma$, $\rho \times \sigma$
  - terms: $(\lambda x^\rho.t^\sigma)^{\rho\to\sigma}$, $(t^{\rho\to\sigma}s^\rho)^\sigma$, $\langle t^\rho; s^\sigma\rangle^{\rho\times\sigma}$, $\pi_i(t^{\rho_0\times\rho_1})^{\rho_i}$
  - β-reduction: $(\lambda x.t)s \mapsto t\,[x{:=}s]$, $\pi_i(\langle t_0; t_1\rangle) \mapsto t_i$
- ground type is that of natural numbers: 0^o, $\mathrm{S}^{o\to o}$
- recursion:
  - recursor $\mathrm{R}_\rho^{\rho\to(o\to\rho\to\rho)\to o\to\rho}$
  - reductions $\quad \mathrm{R}_\rho ts0 \mapsto t$

    $\mathrm{R}_\rho ts(\mathrm{S}r) \mapsto sr(\mathrm{R}_\rho tsr)$
- Gödel's T is confluent and strongly normalizing, in particular equality is decidable.

Heyting arithmetic

- terms: Gödel's T
  - many-sorted language
  - identify terms with the same normal form
- formulae: $P\overrightarrow{t}$, $A \wedge B$, $A \to B$, $\forall x^{\rho}.\, A$, $\exists x^{\rho}.\, A$
- (many-sorted) intuitionistic predicate logic
- induction axiom

$$A\,[x{:=}0] \to (\forall x^{o}.A \to A\,[x{:=}\mathrm{S}x]) \to \forall x^{o} A$$

Realizability Interpretation

- Philosophical background: What does $\exists x^\rho .\, A$ mean? Can we give an example?
- For every formula A we define a (possibly empty) type $[\![A]\!]$ and a statement (formula) $t^{[\![A]\!]}\ \mathbf{mr}\, A$ by induction on A:

$[\![A]\!] = \star$	$\varepsilon\, \mathbf{mr}\, A = A$	A atomic
$[\![A \to B]\!] = [\![A]\!] \to [\![B]\!]$	$t\, \mathbf{mr}\, A \to B = \forall x^{[\![A]\!]}.(x\, \mathbf{mr}\, A) \to tx\, \mathbf{mr}\, B$	
$[\![\exists x^\rho .\, A]\!] = \rho \times [\![A]\!]$	$\langle t; s\rangle\, \mathbf{mr}\, \exists x.A = s\, \mathbf{mr}\, A\,[x{:=}t]$	
$[\![\forall x^\rho .\, A]\!] = \rho \to [\![A]\!]$	$t\, \mathbf{mr}\, \forall x.A = \forall x.\, tx\, \mathbf{mr}\, A$	

- Special case of a Π_2-statement

$$t^{\rho\to\sigma}\ \mathbf{mr}\, \forall x^\rho \exists y^\sigma A = \forall x A\,[y{:=}tx]$$

functions as realizers $\rightsquigarrow$ computer science

Realizability Interpretation — The Theorem

For every derivation of a formula A from free assumptions among $A_1, \ldots, A_n$ and free object variables $x_1, \ldots, x_n$ we can effectively find a term t of type $[\![A]\!]$ with free variables among $x_{A_1}^{[\![A_1]\!]}, \ldots, x_{A_n}^{[\![A_n]\!]}; x_1, \ldots, x_n$ and a derivation of

$$t\,\mathbf{mr}\,A$$

with free assumptions among $x_{A_1}\,\mathbf{mr}\,A_1, \ldots, x_{A_n}\,\mathbf{mr}\,A_n$.

x_{A_i} is a "new" object variable uniquely related to the "assumption A_i".

Proof (sketch): Induction on the derivation. Construct realizers for every rule out of the realizers for the subderivations. Induction for A is realized by $\mathrm{R}_{[\![A]\!]}$, $(\exists^+)$ by the pair constructor $\langle\cdot;\cdot\rangle$, $(\to^+)$ and $(\forall^+)$ by λ-abstraction, $(\to^-)$ by application, ...

Proof (some examples)

Derivation $\mathfrak{D}$		**Proof of $\mathfrak{D}' \,\mathbf{mr}\, A$**
$$\cfrac{\begin{array}{c}[A]\\ \vdots \\ \mathfrak{D} \\ \vdots \\ B\end{array}}{A \to B}$$	$$\begin{array}{c} x_A \,\mathbf{mr}\, A \\ \vdots \\ \mathfrak{D}' \,\mathbf{mr}\, B \end{array} \quad \Rightarrow$$	$$\begin{array}{rl} & \forall x_A.\,(x_A \,\mathbf{mr}\, A) \to \boxed{\lambda x_A.\mathfrak{D}'}\, x_A \,\mathbf{mr}\, B \\ = & \lambda x_A.\mathfrak{D}' \,\mathbf{mr}\,(A \to B) \end{array}$$
$$\cfrac{\begin{array}{c}\mathfrak{D} \\ \vdots \\ A\,[x{:=}t]\end{array} \quad t}{\exists x.\, A}$$		$$\begin{array}{rl} & \vdots \\ & \mathfrak{D}' \,\mathbf{mr}\, A\,[x{:=}t] \\ = & \boxed{\langle t; \mathfrak{D}' \rangle}\,\mathbf{mr}\,\exists x.\, A \end{array}$$

Implicit Computational Complexity (ICC)

- Gödel's T is *not* a universal model of computation (for example it's strongly normalizing!)

$\leadsto$ some complexity bound on Heyting arithmetic (e.g. on Π_2-statements)

- However, Gödel's T has quite high computational complexity (e.g. Ackermann's function is definable)
- Can we get some better bound for some **more restricted form of Heyting Arithmetic**?
- Something feasible, like polynomial time??

A Term System for `Ptime`

- based on Hofmann 1999
- every definable term is a polynomial time computable function
- however, not every Ptime function is definable (growth restriction)
- "simple" restrictions of Gödel's T (no tiering!)

Example: $\texttt{insertion sort}$ & $x \mapsto 2^x$

$$
\begin{array}{lcl}
\texttt{sort}([]) & = & [] \\
\texttt{sort}([a \mid l]) & = & \texttt{insert}(a, \texttt{sort}(l)) \\
 & & \\
\texttt{insert}(a, []) & = & [a] \\
\texttt{insert}(a, [b \mid \ell]) & = & [b' \mid \texttt{insert}(a', \ell)]
\end{array}
\qquad
\begin{array}{lcl}
\texttt{exp}(0) & = & S0 \\
\texttt{exp}(Sx) & = & \texttt{double}(\texttt{exp}(x)) \\
 & & \\
\texttt{double}(0) & = & 0 \\
\texttt{double}(Sx) & = & S(S(\texttt{double}(x)))
\end{array}
$$

a', b' a permutation of a, b with $b' \leq a'$

Controlling the Size of the Recursive Values?

- "They're small because everything is small!"
- Therefore only non-size-increasing functions are considered.
- We need a *syntactical* bound on the "size" of every term of an inductive type (e.g. numbers, lists, …).
- Non-growing syntactical measure?!

$$t \to t' \qquad \Rightarrow \qquad \mathrm{FV}(t) \supset \mathrm{FV}(t')$$

Defining a Term System out of this Idea?

- **first step:** Enforce free variables
  - define a new type $\diamond$ with
$$t^{\diamond} \qquad \Rightarrow \qquad \mathrm{FV}(t) \neq \emptyset$$
  - successor of Type
$$\diamond \multimap \mathbf{N} \multimap \mathbf{N}$$

- **problem:**

$$\mathbf{S}_1 d^{\diamond}(\mathbf{S}_1 d^{\diamond}(\ldots(\mathbf{S}_1 d^{\diamond} 0)))$$

- **second step:** *linear* typing, i.e. only form (st) if

$$\mathrm{FV}(s) \cap \mathrm{FV}(t) = \emptyset$$

Overview of the term system

- **Abstractions** (Introduced by special syntax)
  - λ-abstraction: $\lambda x.\, t$
  - Pair: $\langle s; t \rangle$
- **Data types:** (Introduced by constants)
  - Binary numerals: 0, $\mathbf{S}_0$, $\mathbf{S}_1$
  - Lists: $\mathbf{nil}_\rho{}^{\mathbf{L}(\rho)}$, $\mathbf{cons}_\rho{}^{\diamond \multimap \rho \multimap \mathbf{L}(\rho) \multimap \mathbf{L}(\rho)}$
  - Tensor: $\otimes_{\rho,\sigma}{}^{\rho \multimap \sigma \multimap \rho \otimes \sigma}$
  - Booleans: $\mathbf{t}$, $\mathbf{f}$

Overview of the term system (cont'd)

- **Eliminations** written as application ("Vector Notation")
  - Application: $t^{\rho \multimap \sigma} s$
  - Iteration: $t^{\mathbf{L}(\rho)} \{s\}\, r$

    $t^{\mathbf{N}} \{s_0, s_1\}\, r$
  - Projections: $t^{\rho \times \sigma}\mathbf{t}$ and $t^{\rho \times \sigma}\mathbf{f}$
  - Tensor-Elimination: $t^{\rho \otimes \sigma}(\lambda x^{\rho}, y^{\sigma}\ s)$
  - Cases: $t^{\mathrm{bool}} \langle s_0; s_1 \rangle$
- Type $\diamond$: $x^{\diamond}$

Linear typing rules

$$\frac{\Gamma_1 \vdash t^{\rho \multimap \sigma} \qquad \Gamma_2 \vdash s^{\rho}}{\Gamma_1, \Gamma_2 \vdash (ts)^{\sigma}} \qquad (\multimap^-)$$

$$\frac{\Gamma \vdash t^{\mathbf{L}(\rho)} \qquad \emptyset \vdash s^{\diamond \multimap \rho \multimap \sigma \multimap \sigma}}{\Gamma \vdash (t\,\{s\})^{\sigma \multimap \sigma}} \qquad (\mathbf{L}(\rho)^-)$$

Γ_1, Γ_2 is the union of Γ_1 and Γ_2, expressing

$$\Gamma_1 \cap \Gamma_2 = \emptyset$$

Ptime correctness

Every term denotes a polynomial time computable function.

Proof:

- define explicit polynomial(ly bound function) $\vartheta(t)\colon \mathbb{N} \to \mathbb{N}$ for every term t
- show that $t \to t'$ implies $\vartheta(t)(|\mathrm{FV}(t)|) > \vartheta(t')(|\mathrm{FV}(t)|)$

Induction versus Iteration

If we took induction like this

$$\frac{\forall x^{\mathbf{L}(\rho)} \forall y^{\diamond} \forall z^{\rho}. A \to A\,[x := \mathbf{cons}_{\rho} y z x] \qquad t^{\mathbf{L}(\rho)}}{A\,[x := \mathbf{nil}_{\rho}] \to A\,[x := t]}$$

we would get a step term of type

$$\mathbf{L}(\rho) \multimap \diamond \multimap \rho \multimap [\![A]\!] \multimap [\![A]\!]$$

So we need a proof *uniform in* $x^{\mathbf{L}(\rho)}$!

Linear Logic?

The logical cut

$$\frac{\Gamma \vdash A \to B \qquad \Delta \vdash A}{\Gamma \cup \Delta \vdash B}$$

is mapped to the application of terms ts

- Should we restrict ourselfs to $\Gamma \cap \Delta = \emptyset$?
- Sharing of computational irrelevant assumptions?
- Flexibility (mathematical strength) in the negativ (compuational irrelevant) fragment??

So much for the motivation. Technical details after the ...

C O F F E E   B R E A K

Remember: Iteration versus Induction

If we took induction like this

$$\frac{\forall x^{\mathbf{L}(\rho)} \forall y^{\diamond} \forall z^{\rho}. A \to A\,[x{:=}\mathbf{cons}_{\rho} y z x] \qquad t^{\mathbf{L}(\rho)}}{A\,[x{:=}\mathbf{nil}_{\rho}] \to A\,[x{:=}t]}$$

we would get a step term of type

$$\mathbf{L}(\rho) \multimap \diamond \multimap \rho \multimap [\![A]\!] \multimap [\![A]\!]$$

So we need a proof *uniform in* $x^{\mathbf{L}(\rho)}$!

Uniformity?

- Need to express uniformity

$\rightsquigarrow$ new quantifier $\forall^*$

  - "strong all-quantifier"
  - $\forall^* x.\, A$ should express that there is a proof of A not using x in a "computational relevant" way

- What is used in a computational relevant way?

$\rightsquigarrow$ Add a second context to control this!

The *two* Contexts

- “Normal” Context:
  - variables for the free assumptions
  - free variables of object terms used as
    - witnesses in $\exists^+$
    - specializations in $\forall^-$
  - the same as in ordinary Heyting arithmetic
- “Essential” Context:
  - subset of the normal context (think of it as underlinings in the normal context!)
  - measures the computational relevant part

  $\rightsquigarrow$ allows to distinguish between $\forall$ and $\forall^*$, in introduction *and* elimination rules

Notation for two Contexts

notation		normal		essential
Γ	$\rightsquigarrow$	Γ	;	$\emptyset$
$\underline{\Gamma}$	$\rightsquigarrow$	Γ	;	Γ
$\Gamma, \underline{\Gamma'}, \Delta, \underline{\Delta'}$	$\rightsquigarrow$	$\Gamma \cup \Gamma' \cup \Delta \cup \Delta'$	;	Γ', Δ'
				$\Gamma' \cap \Delta' = \emptyset$!

- underlined elements are in *both* contexts
- non-underlinded elements only in the *normal* context
- *linearity* restrictions in the *essential* context
- *no* linearity in the the *normal* context

$\forall$ **versus** $\forall^*$

$$\frac{\Delta, \underline{\Delta'}, \underline{x} \vdash A \qquad \texttt{Vars!}}{\Delta, \underline{\Delta'} \vdash \forall x.A} \qquad\qquad \frac{\Delta, \underline{\Delta'}, x \vdash A \qquad \texttt{Vars!} \quad x \notin \Delta'}{\Delta, \underline{\Delta'} \vdash \forall^* x.A}$$

$$\frac{\Delta, \underline{\Delta'} \vdash \forall x^\rho A \qquad \Gamma \vdash t^\rho}{\Delta, \underline{\Delta'}, \underline{\Gamma} \vdash A\,[x{:=}t]} \qquad\qquad \frac{\Delta, \underline{\Delta'} \vdash \forall^* x^\rho A \qquad \Gamma \vdash t^\rho}{\Delta, \underline{\Delta'}, \Gamma \vdash A\,[x{:=}t]}$$

$$[\![\forall x^\rho A]\!] = \rho \multimap [\![A]\!] \qquad\qquad [\![\forall^* x^\rho A]\!] = [\![A]\!]$$

Computational Context

- weakening

$$\frac{\Delta, \underline{\Delta'} \vdash A}{\Gamma, \Delta, \underline{\Gamma'}, \underline{\Delta'} \vdash A}$$

- pruning

$$\frac{\Delta, \underline{\Gamma} \vdash A \qquad [\![A]\!] = \star}{\Delta, \Gamma \vdash A}$$

- This shows $(\forall^* x A) \to (\forall x A)$, i.e. the "strong all-quantifier" really *is* stronger
- However, $(\forall x A) \to (\forall^* x A)$ if $[\![A]\!] = \star$.

Induction

$$\frac{\Delta \vdash \forall^* x^{\mathbf{L}(\rho)} \forall y^{\diamond} \forall z^{\rho}. A \to A\,[x{:=}\mathbf{cons}_\rho yzx] \qquad \Gamma \vdash t^{\mathbf{L}(\rho)}}{\Delta, \underline{\Gamma} \vdash A\,[x{:=}\mathbf{nil}_\rho] \to A\,[x{:=}t]}$$

- Note that the step case has to be shown "essentially closed"!
- Type of a realizer of the step case

$$\diamond \multimap \rho \multimap [\![A]\!] \multimap [\![A]\!]$$

The Theorem

For every derivation $\mathfrak{D}$ of $\overrightarrow{y}, \overrightarrow{B}, \underline{\overrightarrow{z}}, \underline{\overrightarrow{C}} \vdash A$ there is a term $\mathfrak{D}'$ and a derivation $\tilde{\mathfrak{D}}$, effectively (and feasibly) computable from $\mathfrak{D}$ such that

- "soundness of typing":
$$\overrightarrow{x_C}, \overrightarrow{z} \vdash (\mathfrak{D}')^{[\![A]\!]}$$
- "soundness of realizability": $\tilde{\mathfrak{D}}$ is a derivation of
$$\overrightarrow{y}, \overrightarrow{x_B}, \overrightarrow{x_B\,\mathbf{mr}\,B}, \overrightarrow{z}, \overrightarrow{x_C}, \overrightarrow{x_C\,\mathbf{mr}\,C} \vdash \mathfrak{D}'\,\mathbf{mr}\,A$$

Notes:

- x_C is a "new" variable of type $[\![C]\!]$ uniquely related to the "assumption C".
- technically: "decorate" derivations with assumption variables and proof terms, then define $\mathfrak{D}'$ and $\tilde{\mathfrak{D}}$ by induction on these structures.

Examples of Extracted Terms

$$\frac{\begin{matrix}\mathfrak{D}\\ A\end{matrix}}{\forall^* x A} \quad \rightsquigarrow \quad \mathfrak{D}' \qquad\qquad \frac{\begin{matrix}\mathfrak{D}\\ A\end{matrix}}{\forall x A} \quad \rightsquigarrow \quad \lambda x.\mathfrak{D}'$$

$$\frac{\begin{matrix}\mathfrak{D}\\ \forall^* x A\end{matrix} \qquad t}{A\,[x{:=}t]} \quad \rightsquigarrow \quad \mathfrak{D}' \qquad\qquad \frac{\begin{matrix}\mathfrak{D}\\ \forall x A\end{matrix} \qquad t}{A\,[x{:=}t]} \quad \rightsquigarrow \quad \mathfrak{D}' t$$

$$(\text{Induction}) \quad \rightsquigarrow \quad (\mathbf{L}(\llbracket A \rrbracket)^-) \qquad\qquad (\exists^+) \quad \rightsquigarrow \quad \otimes$$

Soundness of Typing

$$\overrightarrow{x_C}, \overrightarrow{z} \vdash (\mathfrak{D}')^{[\![A]\!]}$$

- The *new* part of the statement, linking the arithmetic to the computational complexity!
- main work: careful design of the rules (already done!)
- Proof: simple induction on the derivation

Soundness of Realizability

$$\overrightarrow{y}, \overrightarrow{x_B}, \overrightarrow{x_B \,\mathbf{mr}\, B}, \overrightarrow{z}, \overrightarrow{x_C}, \overrightarrow{x_C \,\mathbf{mr}\, C} \vdash \mathfrak{D}' \,\mathbf{mr}\, A$$

- Since we know
$$[\![t \,\mathbf{mr}\, A]\!] = \star$$
(by induciton on A) and we have the rule
$$\text{(pruning)}$$
we're as flexible as in normal Heyting Arithmetic.
- Proof: By induction on the derivation; details as usual (for normal Heyting Arithmetic)

Example (Append)

- Axioms (all-closures of):

$$\mathtt{APPEND}(\mathbf{nil}_\rho, \ell^{\mathbf{L}(\rho)}, \ell^{\mathbf{L}(\rho)})$$

$$\mathtt{APPEND}(\ell^{\mathbf{L}(\rho)}, \ell'^{\mathbf{L}(\rho)}, \ell''^{\mathbf{L}(\rho)}) \to \mathtt{APPEND}(\mathbf{cons}_\rho x^\diamond y^\rho \ell, \ell', \mathbf{cons}_\rho x y \ell'')$$

- We show $\forall \ell'^{\mathbf{L}(\rho)}, \ell^{\mathbf{L}(\rho)} \exists \ell''^{\mathbf{L}(\rho)} \mathtt{APPEND}(\ell, \ell', \ell'')$.
- Induction on ℓ. Step case

  Prove $\exists \ell''^{\mathbf{L}(\rho)} \mathtt{APPEND}(\mathbf{cons}_\rho x^\diamond y^\rho \ell, \ell', \ell'')$ from ℓ', ℓ, $\underline{x^\diamond}$, $\underline{y^\rho}$ and $\underline{\exists \ell''^{\mathbf{L}(\rho)} \mathtt{APPEND}(\ell, \ell', \ell'')}$.
- Extracted program

$$\begin{aligned} \lambda \ell', \ell \quad \ell\{\lambda x^\diamond, z^\rho, x_u^{\mathbf{L}(\rho)} \quad & (\lambda \ell'' \\ & (\lambda y^{\mathbf{L}(\rho)}\, y)(\mathbf{cons}_\rho x z \ell''))x_u\} \\ & ((\lambda y^{\mathbf{L}(\rho)}\, y)\ell') \end{aligned}$$

Work in Progress

All the statements so far are proven theorems . . .

. . . however, there's still some work in progress!

Alternative Ideas: $\mathrm{E}(\cdot)$-Predicates, Single Context

- So far, we used $\forall^*$ as a special quantifier. An alternative could be the use of just the quantifiers $\forall$, $\exists$ and interpret them both as computationally *irrelevant*. Computational relevance comes in form of a special predicate symbol $\mathrm{E}(\cdot)$. The present calculus is a special case of this:

$$
\begin{array}{ccc}
\forall x.\ \dots & \rightsquigarrow & \forall x.\, \mathrm{E}(x) \to \dots \\
\forall^* x.\ \dots & \rightsquigarrow & \forall x.\ \dots \\
\exists x.\ \dots & \rightsquigarrow & \exists x.\, \mathrm{E}(x) \wedge \dots
\end{array}
$$

- So far, we used two contexts. One could also try a single context and

$$
\frac{\Delta, \Gamma \vdash A \to B \qquad \Gamma, \Delta' \vdash A \qquad \Gamma \text{ computationally irrelevant}}{\Delta, \Gamma, \Delta' \vdash B}
$$

  This is a derived rule of the present calculus, however simpler(?). Maybe this suffices for "enough interessting consequences"??

T h e

E N D