# ULF – Ultrafinitist and Ultraconstructivist Logical Framework*

Michał J. Gajda

Migamake Pte Ltd
mjgajda@migamake.com

**Abstract**

Ultrafinitism postulates that we can only compute on relatively short objects, and numbers beyond certain value are not available. This approach would also forbid many forms of infinitary reasoning and allow to remove certain paradoxes stemming from enumeration theorems. For computational application of1 ultrafinitist logic, we need more than a proof system, but a logical framework to express both proofs, programs, and theorems in a single framework. In spirit of ultraconstructivism, the framework forbids use of empty datatypes like *falsum*. We present its inference rules and propose facilities to make their use just as simple as logical frameworks used in conventional theorem provers.

## 1 Introduction

Ultrafinitism(Kornai 2003; Podnieks 2005; Yessenin-Volpin 1970; Gefter 2013) postulates that we can only reason and compute relatively short objects(Krauss and Starkman 2004; Sazonov 1995; Lloyd 2002; Gorelik 2010), and numbers beyond certain value are not available. This approach would also forbid many forms of infinitary reasoning and allow to remove many from paradoxes stemming from countable enumeration.

However, philosophers still disagree of whether such a finitist logic could be consistent(Magidor 2007), while constructivist mathematicians claim that *"no satisfactory developments exist"*(Troelstra 1988). We present preliminary work on a proof system based on Curry-Howard isomorphism(Howard 1980).

We believe that this approach may present certain impossibility results as logical paradoxes stemming from a profligate use of transfinite reasoning.

## 2 Definitions

### 2.1 Variables, and polynomials

We assume a set of variables $x \in X$, polynomial variables $v \in V$, and integers $i \in \mathbb{Z}$. We will use polynomials:

$$\rho ::= v \mid i \mid \rho + \rho \mid \rho * \rho \mid \rho^{\rho} \mid \rho^{\circ\rho}[v]$$

Note that our polynomials will be ever used when they have a positive natural value.

Here $\rho^{\rho}$ is an ordinary exponentation, and $\rho^{\circ\rho}[v]$ is an iterated function composition with respect to argument variable $v$.

In case that we are not interested in the exact constants, we may replace each polynomial with big-oh notation $\mathcal{O}(\rho_1)$. In this case we just state that there is a judgement that has all

*

polynomials $\mathcal{O}(\rho_1), ..., \mathcal{O}(\rho_2)$ replaced by by some polynomial that is majorized by the $c * \rho_1$ for *small constant* $|c| < 1000$.

The polynomials will be standing on one of two roles: as an upper bound on the proof complexity, and there we will use symbol $\alpha$ as placeholder, or to state an upper bound on the number of constructors in the proof indicated by symbol $\beta$. That is because number of constructors may sometimes bound a recursive examination of the proof of a proposition.

Terms are given by:

$$p ::= v \,\big|\, \lambda v.p \,\big|\, \forall_\alpha v : p \to p \,\big|\, \exists v_\beta.p$$

Environments of assumptions are given by a list of propositions with a constructor depth attached to them.

$$\Gamma ::= \epsilon \,\big|\, x_v : A, \Gamma$$

Judgements are of the form: $\Gamma \vdash^{\alpha}_{\beta} A$ .

In theory we could attach a pair to each proposition and judgement $A_{(\alpha,\beta)}$ that would describe both complexity $\alpha$ of computing the proof and a depth $\beta$ of the resulting witness. However in most cases one of these would be 1 or could be inferred from the remaining information.

## 2.2 Inference rules

$$\frac{\Gamma \vdash^{\alpha}_{\beta} y : T \quad v \in V \quad w \in V \quad x \in X \quad A \in X}{\Gamma, x_v : A \vdash^{1}_{v} x : A}\ \textit{polyvar} \qquad \frac{\Gamma \vdash^{\alpha}_{\beta} y : A \quad v \in V}{\Gamma, x_v : A \vdash^{1}_{v} x : A}\ \textit{var}$$

$$\frac{\Gamma \vdash^{\alpha_1}_{\beta_1} a^1 : A^1 \quad \Gamma \vdash^{\alpha_2}_{\beta_2} a^2 : A^2}{\Gamma \vdash^{\alpha_1+\alpha_2}_{\max(\beta_1,\beta_2)+1} (a^1, a^2) : A^1 \wedge A^2}\ \textit{pair} \qquad \frac{\Gamma \vdash^{\alpha}_{\max(\beta_1,\beta_2)} e : A^1 \wedge A^2}{\Gamma \vdash^{\alpha+1}_{\beta-1} \mathit{prj}_i\ e : A^i}\ \textit{prj}_i$$

$$\frac{\Gamma \vdash^{\alpha}_{\beta} e : A^i}{\Gamma \vdash^{\alpha+1}_{\beta+1} \mathit{inj}_i\ e : A^1 \vee A^2}\ \textit{inj}_i \qquad \frac{\Gamma \vdash^{\alpha_1}_{\beta_1} e : A \qquad \alpha_1 \le \alpha_2 \qquad \beta_1 \le \beta_2}{\Gamma \vdash^{\alpha_2}_{\beta_2} e : A}\ \textit{bound}$$

$$\frac{\Gamma \vdash^{\alpha_\vee}_{\beta_\vee} a : A^1 \vee A^2 \qquad \Gamma, x : {A^1}_{\beta_\vee - 1} \vdash^{\alpha_1}_{\beta_1} b : B \qquad \Gamma, y : {A^2}_{\beta_\vee - 1} \vdash^{\alpha_2}_{\beta_2} c : B}{\Gamma \vdash^{\alpha_\vee + max(\alpha_1,\alpha_2)+1}_{\max(\beta_1,\beta_2)} \mathit{case}\ a\ \mathit{of}\ \mathit{inj}_1\ x \to b;\ \mathit{inj}_2\ y \to c : B}\ \textit{case}$$

$$\frac{\Gamma, x_v : A \vdash^{\alpha(v)}_{\beta(v)} e : B}{\Gamma \vdash^{\alpha(1)+1}_{\beta(1)+1} \lambda x.e : \forall a_v : A \Longrightarrow^{\alpha(v)}_{\beta(v)} B}\ \textit{abs} \qquad \frac{\Gamma \vdash^{\alpha_1}_{\beta_1} e : \forall a : A_v \Longrightarrow^{\alpha_2(v)}_{\beta_2(v)} B \quad \Gamma \vdash^{\alpha_3}_{\beta_3} a : A}{\Gamma \vdash^{\alpha_1+\alpha_2(\beta_3)+\alpha_3}_{\beta_2(\beta_3)} e\ a : B}\ \textit{app}$$

Please note that notation $\forall x_v : A \implies^{\alpha(v)}_{\beta(v)} B$ has variable $x$ declared as depth of $A$, and then bound in polynomials $\alpha(v)$ and $\beta(v)$

With exception of *bound* these are all reinterpretations of rules for intuitionistic logic, enriched with bounds on the proof length $\alpha$ and normalized term depth $\beta$.

Please note that these rules all maintain bounded depth with no unbounded recursion. We may add rule for recursive definitions (like definition of the closure):

$$\frac{\Gamma \vdash^{\alpha_1}_{\beta_1} f : A_v \Rightarrow^{\alpha_2(v))}_{\beta_2(v)} A \quad \Gamma \vdash^{\alpha_3}_{\beta_3} a : A \quad v > \beta_2(v)}{\Gamma \vdash^{\alpha_1+\alpha_2^{\circ\beta_3}[\beta_3]+\alpha_3}_{\beta_2^{\circ\beta_3}[\beta_3]} rec\, f\ a : B}\ rec$$

### 2.3 Simplifying bound polynomials

Our inference rules rely on computing polynomial bounds and their inequality. Here we note a few inequalities that simplify reasoning about these bounds, albeit at the cost of making them somewhat looser.

First we note that all variables are positive naturals because they represent the data of non-zero size: $x \geq 1$.

That means that the following laws are true, assuming that $x, y, ... \geq 1$ are data size variables in the environment, $e \leq f \leq 1$ and $g \leq h \leq 1$ are arbitrary positive expressions, and $a, b, c... \geq 1$ are constants. For easier use the rules are presented in left-to-right order, just like conventional rewrite rules.

$$\begin{array}{lrcl}
(1) & x^e & \leq & x^f \\
(2) & a * x^e + b * x^f & \leq & (a+b) * x^f \\
(3) & x^e & \leq & x^f \\
(4) & a * x^e + b * x^f & \leq & (a+b) * x^f \\
(5) & a * x^e * y^g & \leq & a * x^f * y^h \\
(6) & e^{\circ g}[x] & \leq & f^{\circ h}[x] \\
(7) & (a * x)^{\circ e}[x] & = & a^e * x \\
(8) & (x + a)^{\circ e}[x] & = & x + a * e \\
(9) & (x^e)^{\circ g}[x] & = & x^{e^g}
\end{array}$$

We may thus use these identities to loosen the bound in such way as to reduce the size of polynomial and make it a sum of a single term in all variables and additional constant term. This reduction may be delayed until we have bound to verify.

Note that all the inference rules can be used leaving polynomials as the holes to be filled by the framework interpreter.

## 3 Sketching the proofs

### 3.1 Strong normalization

Please note that all rules increase polynomial bound on computation effort: $\alpha$. Thus if we are able to infer a judgement with a valid bound, we get a bounded judgement.

The depth bound $\beta$ is used for delimiting recursion depth, since it has to decrease at each stage of loop application *rec*.

Please note that after elision of bounds the exposure follows the intuitionistic logic. Thus the consistency

## 4 Facilities

Here we discuss possible extensions that will make it easier to use the framework.

### 4.1 Inductive types

Inductive types can be encoded with finite pairs and sums above, but we can also have mutually recursive declarations of the form:

$$I ::= data\ t\ \tau^* = [c\ a^*]^+$$

Where:

- $t \in T$ is a type identfier,
- $\tau = V$ is a type variable,
- $c \in C$ is a constructor identifier,
- $a ::= \tau \mid t\ a^*$ which means that argument to a constructor is either a type variable, or a type constructor and a list of arguments.

For each type constructor we have a fixed arity of type arguments, and ditto for any constructor $c$ for this type. We also need to compute a *minimum depth of the type*, which a minimum of *minimal depths for each constructor*. Minimal depth for each constructor is a maximum of all its arguments, or 1 if there are no arguments. For a set of datatypes to be valid, they have to all possess a *minimum depth* that is finite number.

Please note that this definition explicitly forbids inductive types that are empty, or do not have a finite construction.

For each inductive type $I$ with constructors $C_1, ...C_n$ with arities $k_1, ..., k_n$ we can apply a *case* rule:

$$\frac{\Gamma \vdash^{\alpha_\vee}_{\beta_\vee} a : I\ a^1\ ...\ a^n \qquad \Gamma, r_v^{1,1} : A^1, ..., r_v^{1,k_1} : A^n \vdash^{\alpha_1(v)}_{\beta_1(v)} e_1 : B \quad ... \qquad \Gamma, r_v^{n,1} : A^1, ..., r_v^{n,k_n} : A^n \vdash^{\alpha_n(v)}_{\beta_n(v)} e_n : B}{\Gamma \vdash^{\alpha_\vee + max_k(\alpha_1(\beta_\vee - 1),...,\alpha_n(\beta_\vee - 1))}_{\max(\beta_1(\beta_\vee - 1),...,\beta_n(\beta_\vee - 1))} case\ a\ of\ C_1\ r^{1,1}\ ...r^{1,k_1} \to e_1;\ ...; C_n\ r^{n,1}\ ...r^{n,k_n} \to e_n : B}\ case$$

### 4.2 Metareasoning

We may now decode the type constructors:

$$[\![A \vee B]\!] = inj_1(pair[\![A]\!][\![B]\!])$$
$$[\![A \wedge B]\!] = inj_2(pair[\![A]\!][\![B]\!])$$
$$[\![\forall x : A.B]\!] = \lambda x : A.[\![B]\!]$$

This decoding allows us to make operations on types akin to generic programming in Haskell(**generic?**)

## 5 Discussion

We have shown a possible consistent logic for inference with strictly bounded number of steps. This allows us to limit our statements by the length of acceptable proof, and thus define statements that are not just true, but computable within Bremermann-Gorelik limit[1](Gorelik 2010).

This in turn allows us to conduct a reasonable ultrafinitist discourse without inconsistency. The underlying proof system is also much strong than weak subtheories like Bounded Arithmetic(Krajicek 1995).

[1]Computation run by computer the size of Earth within the lifespan of Earth so far. Of the order of $10^{93}$.

In future work we will attempt to characterize the realm of ultrafinitist theorems. For example the famous undecidability example that refers to arbitrary program seems impossible to define in this framework. In particular any program bounded by $\alpha(x)$ for any input $\beta(x)$ might be checked for termination within a significantly larger but still bounded limit of $\alpha^{\beta(x)}(x)$. Another avenue of future work would be to define full type theory, dependently typed language and automatic prover for this inference rules. Since polynomials $\alpha(x)$ are monotonic, it is likely that the proof strategy will be also bounded in this case. A penultimate application would be defining automated inference system for upper bounds of (eager) functional programs and thus warn user about possible non-termination of the program in absence of infinite input.

## References

10 Gefter, Amanda. 2013. "Mind-Bending Mathematics: Why Infinity Has to Go." *New Scientist* 219 (2930): 32–35. https://doi.org/https://doi.org/10.1016/S0262-4079(13)62043-6.

Gorelik, Gennady. 2010. "Bremermann's Limit and cGh-Physics." http://arxiv.org/abs/0910.3424.

Howard, William A. 1980. "The Formulae-as-Types Notion of Construction." In *To h.b. Curry: Essays on Combinatory Logic, λ-Calculus and Formalism*, edited by J. Hindley and J. Seldin, 479–90. Academic Press.

Kornai, Andras. 2003. "Explicit Finitism." *International Journal of Theoretical Physics* 42 (February): 301–7. https://doi.org/10.1023/A:1024451401255.

Krajicek, Jan. 1995. *Bounded Arithmetic, Propositional Logic and Complexity Theory*. Encyclopedia of Mathematics and Its Applications. Cambridge University Press. https://doi.org/10.1017/CBO9780511529948.

Krauss, Lawrence, and Glenn Starkman. 2004. "Universal Limits on Computation," May.

Lloyd, S. 2002. "Computational Capacity of the Universe." *Physical Review Letters* 88 23: 237901.

Magidor, Ofra. 2007. "Strict Finitism Refuted?"

Podnieks, Karlis. 2005. "Towards a Real Finitism?" 2005. http://www.ltn.lv/%C2%A0podnieks/finitism.htm.

Sazonov, Vladimir Yu. 1995. "On Feasible Numbers." In *Logic and Computational Complexity*, edited by Daniel Leivant, 30–51. Berlin, Heidelberg: Springer Berlin Heidelberg.

Troelstra, A. S. 1988. *Constructivism in Mathematics: An Introduction*. Elsevier.

Yessenin-Volpin, Aleksandr S. 1970. "The Ultra-Intuitionistic Criticism and the Antitraditional Program for Foundations of Mathematics." In *Studies in Logic and the Foundations of Mathematics*, 60:3–45. Elsevier.