"Proofs *are* programs" in MLTT

Pierre-Marie Pédrot

INRIA

Martin-Löf type theory truly is the paradise of constructive mathematics, as it is indeed both a logical foundation *and* a programming language. Yet the previous sentence does not do justice to the profound essence of MLTT: it is not just some fancy logical reasoning strapped onto a run-of-the-mill programming language. In MLTT, computation and logic are literally identified into a single monistic view, and the choice to consider an object as logical or computational can be framed as an opinion. It is thus a truism that "proofs *are* programs" in MLTT, a self-evident truth which we will dub the *credo* of Church's church for reasons that will become clear soon. This identification holds by construction and does not require e.g. any realizability model.

With this in mind, it should come as a surprise that MLTT enjoys non-computational models, such as the **Set** model where MLTT functions are interpreted as ZFC functions. The reason for this discrepancy is that, in MLTT, the proof-as-program identification is an external fact not reflected in the theory itself. Thankfully, there is a well-known solution to bridge this gap: the internal Church Thesis [8]. In higher-order arithmetic, this principle can be stated as

$$\mathsf{CT} : \forall(f : \mathbb{N} \to \mathbb{N}).\, \exists(p : \mathbb{N}).\, \forall(n : \mathbb{N}).\, \exists(k : \mathbb{N}).\, \mathsf{T}\ p\ n\ k\ (f\ n)$$

where T is the decidable Kleene predicate. Namely, $\mathsf{T}\ p\ n\ k\ v$ holds whever p is the code of some Turing machine, and running p on the input n terminates in less than k steps and returns the value v. Said otherwise, CT guarantees that any internal function $f : \mathbb{N} \to \mathbb{N}$ is reflected by an actual algorithm $p : \mathbb{N}$ from within the logic, i.e. is extensionally a program.

The CT principle was heavily used by the Russian constructivist school [6], and is known to be consistent with higher-order arithmetic. The typical way to prove this is via Kleene realizability, where proofs are interpreted as concrete codes. One has to be wary that CT is quite an oddball, though. Assuming enough choice, it contradicts both weak forms of excluded middle like LLPO *and* function extensionality. This is not a problem for MLTT, which does not validate either.

After having accumulated that much evidence, the reader could rightfully assume that the compatibility of MLTT with CT is a classic, if not folklore result. As a matter of fact, MLTT+CT is the foundation for synthetic computability [3], another offshoot of the synthetic trend that trivializes the annoying parts of computability proofs by working directly and implicitly with computable functions. Surely one does not add axioms lightly when it comes to developping a sizable formalized library. So, MLTT + CT ought to be known to be consistent. Right?

Interestingly, the answer to this question so far was: *it depends*[1]. The problem boils down to the precise definition of CT in our theory. Many type theories feature several kind of existential types, typically contrasting actual existence $\Sigma x : A.\, B$ with propositional existence $\exists x : A.\, B$. Since the arithmetic statement of CT features an existential quantification, there are as many ways to interpret CT as there are existential types[2], i.e. we have two principles

$$\begin{aligned} \mathsf{CT}_{\Sigma} &: \Pi(f : \mathbb{N} \to \mathbb{N}).\, \Sigma(p : \mathbb{N}).\, \Pi(n : \mathbb{N}).\, \Sigma(k : \mathbb{N}).\, \mathsf{T}\ p\ n\ k\ (f\ n) \\ \mathsf{CT}_{\exists} &: \Pi(f : \mathbb{N} \to \mathbb{N}).\, \exists(p : \mathbb{N}).\, \Pi(n : \mathbb{N}).\, \Sigma(k : \mathbb{N}).\, \mathsf{T}\ p\ n\ k\ (f\ n) \end{aligned}$$

[1] One could not have expected less from a dependent type theory.

[2] The translation choice for the second existential quantification does not matter in most settings.

which are in general not equivalent. As $\exists$ is usually intended to be uncomputational and thus does not satisfy choice, $\mathsf{CT}_\exists$ is the nicest of the two, i.e. it does not contradict classical logic nor function extensionality. As a result, it was showed to be consistent not only with MLTT, but also with univalence [7]. By contrast, as Σ-types validate non-choice automatically, CT_Σ is the portal to an algorithmic hell featuring a quote function $\mathfrak{q} : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$.

At the risk of repeating ourselves, "proofs *are* programs" in MLTT, even if only externally so. It should thus be easy to extend MLTT with CT_Σ. At this point the waters become extremely murky. The few published results on the topic [5, 4] are only able to prove the consistency of CT_Σ with a crippled subset of MLTT deprived of the ξ rule, i.e. congruence for λ-abstractions, which prevents conversion to proceed under binders. Furthermore, they hint that handling full-blown MLTT is a hard problem.

This was an unbearable situation for us. First, we believe that the ξ rule is a critical feature of MLTT. Second, $\mathsf{MLTT} + \mathsf{CT}_\Sigma$ is obviously consistent, because remember that "proofs *are* programs". So, it was a categorical imperative to actually prove it, and not merely on paper, as nobody trusts paper proofs about type theory. Therefore, as the only reasonable path forward, we formalized in Coq the consistency of "MLTT", an extension of MLTT that proves CT_Σ. This settles the question for good.

In a nutshell, "MLTT" is a variant of MLTT with one universe, negative Π and Σ types with definitional η-rules, additionally featuring empty, identity and natural number types. It features three additional quotation primitives

$$\frac{\Gamma \vdash M : \mathbb{N} \to \mathbb{N}}{\Gamma \vdash \mathfrak{q}\ M : \mathbb{N}} \qquad \frac{\Gamma \vdash M : \mathbb{N} \to \mathbb{N} \qquad \Gamma \vdash N : \mathbb{N}}{\Gamma \vdash \mathfrak{s}\ M\ N : \mathbb{N}} \qquad \frac{\Gamma \vdash M : \mathbb{N} \to \mathbb{N} \qquad \Gamma \vdash N : \mathbb{N}}{\Gamma \vdash \varrho\ M\ N : M\ \mathfrak{Q}\ N}$$

where $M\ \mathfrak{Q}\ N := \mathsf{T}\ (\mathfrak{q}\ M)\ N\ (\mathfrak{s}\ M\ N)\ (M\ N)$. In other words, these three operations implement the skolemization of CT_Σ. In particular, "MLTT" proves CT_Σ trivially, hence

""proofs *are* programs" in "MLTT"".

For brevity, we will not present in detail the computational rules of these operations, but give the general idea. Basically, these operations will only compute on closed deep normal forms. For instance, the conversion rule for $\mathfrak{q}$ is given as

$$\mathfrak{q}\ M \equiv \lceil M \rceil \quad \text{when } M \text{ closed deep normal form}$$

where $\lceil \cdot \rceil : \mathtt{Term} \to \mathbb{N}$ is a quotation function in the metatheory, which together with T defines a *computational model* for "MLTT".

Under mild hypotheses on this model, it is possible to show that not only "MLTT" is consistent, but is also strongly normalizing and enjoys canonicity. We prove these facts through essentially the same logical relation used by Abel et al. to prove decidability of conversion [1]. The main difference is that we annotate our semantic proofs of conversions with the fact that not only they are normalizing for head reduction, but also for iterated head (i.e. deep) reduction. This addition is virtually transparent and did not require any non-trivial change to the relation. The only additional material needed for the proof is a fair amount of rewriting theory for the untyped fragment of "MLTT", including confluence and standardization. Moreover, some care has to be taken to properly handle the definitional η-rules of negative types, which adds some unwanted technicity. The Coq formalization is based on the `logrel-coq` project [2] and can be found at https://github.com/ppedrot/quote-mltt. Although we did not prove decidability of type-checking for "MLTT", this should be an easy byproduct of this development.

It appears that the solution to the internalization of CT in MLTT is conceptually trivial: simply restrict computation to closed normal forms. While this seems to go against the type-theoretical ethos, it turns out that this plays well with the usual expectations on MLTT such as canonicity and strong normalization. As a result, we believe that this cheap trick can go a long way to internalize externally derivable rules in MLTT. We leave the study of the class of axioms that can be implemented this way to future work.

References

[1] A. Abel, J. Öhman, and A. Vezzosi. Decidability of conversion for type theory in type theory. *Proc. ACM Program. Lang.*, 2(POPL), Dec. 2017.

[2] A. Adjedj, M. Lennon-Bertrand, K. Maillard, P.-M. Pédrot, and L. Pujet. Martin-Löf à la Coq. In S. Blazy, B. Pientka, A. Timany, and D. Traytel, editors, *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2024, London, UK, January 15-16, 2024*. ACM, 2024.

[3] Y. Forster. *Computability in constructive type theory*. PhD thesis, Saarland University, Germany, 2021.

[4] H. Ishihara, M. E. Maietti, S. Maschio, and T. Streicher. Consistency of the intensional level of the minimalist foundation with Church's thesis and axiom of choice. *Arch. Math. Log.*, 57(7-8):873–888, 2018.

[5] M. E. Maietti. A minimalist two-level foundation for constructive mathematics. *Ann. Pure Appl. Log.*, 160(3):319–354, 2009.

[6] M. Margenstern. L'école constructive de Markov. *Revue d'histoire des mathématiques*, 1995.

[7] A. W. Swan and T. Uemura. On Church's thesis in cubical assemblies. *Math. Struct. Comput. Sci.*, 31(10):1185–1204, 2021.

[8] A. Troelstra and D. Dalen. *Constructivism in Mathematics: An Introduction*. Number vol. 1 in Constructivism in Mathematics. North-Holland, 1988.