Refutation of shallow embedding in Martin-Löf type theory

Abstract: In Martin-Löf type theory (MLTT), we evaluate shallow embedding as the following conjecture: "if we add the rewrite rule ∀*x*. f *x* (**not** *x*) = **true**, the expression f **true false** will not be rewritten to true, since it does not rigidly match the **not** *x* on the left hand side". The conjecture is *no*t tautologous, hence refuting shallow embedding in MLTT and forming a *non* tautologous fragment of the universal logic VŁ4.

We assume the method and apparatus of Meth8/VŁ4 with `T`autology as the designated proof value, **`F`** as contradiction, `N` as truthity (non-contingency), and `C` as falsity (contingency). The 16-valued truth table is row-major and horizontal, or repeating fragments of 128-tables, sometimes with table counts, for more variables. (See ersatz-systems.com.)

LET ~ Not, ¬ ; + Or, ∨, ∪, ⊔ ; - Not Or; & And, ∧, ∩, ⊓, ·; \ Not And;
> Imply, greater than, →, ⇒, ↦, ≻, ⊃, ↠ ; < Not Imply, less than, ∈, ≺, ⊂, ⊬, ⊭, ↞, ≲ ;
= Equivalent, ≡, :=, ⇔, ↔, ≜, ≈, ≃ ; @ Not Equivalent, ≠;
% possibility, for one or some, ∃, ◊, M; # necessity, for every or all, ∀, □, L;
(z=z) `T` as tautology, ⊤, ordinal 3; (z@z) **`F`** as contradiction, Ø, Null, ⊥, zero;
(%z>#z) `N` as non-contingency, Δ, ordinal 1; (%z<#z) `C` as contingency, ▽, ordinal 2;
~(y < x) (x ≤ y), (x ⊆ y), (x ⊑ y); (A=B) (A~B).
Note for clarity, we usually distribute quantifiers onto each designated variable.

From: Kaposi, A.; András Kovács, A.; Kraus, N. (2019). Shallow embedding of type theory is morally correct. arxiv.org/ftp/arxiv/papers/1907/1907.07562.pdf nicolai.kraus@gmail.com

1 Introduction Martin-Löf type theory .. (MLTT) is a formal system which can be used for writing and verifying programs, and also for formalising mathematics. Proof assistants and dependently typed programming languages such as Agda .., Coq .., Idris .., and Lean .. are based on MLTT and its variations. (1.1.1)

Remark 1.1.1: We refute Martin-Löf type theory (MLTT) and Coq elsewhere as *not* tautologous. This implies that Agda, Idris, and Lean as used in this context are suspicious.

1.2 Reflecting definitional equality To eliminate explicit derivations of conversion, the most promising approach is to reflect object-level definitional equality as meta-level definitional equality. If this is achieved, then all conversion derivations can be essentially replaced by proofs of reflexivity, and the meta-level typechecker would implicitly construct all derivations for us. How can we achieve this? We might consider extensional type theory with general equality reflection, or proof assistants with limited equality reflection. In Agda there is support for the latter using rewrite rules .., which we have examined in detail for the previously described purposes. In Agda, we can just postulate the syntax of the object theory, and try to reflect the equations. This approach does work to some extent, but there are significant limitations: …

– In the current Agda implementation (version 2.6), rewrite rules are not flexible enough to capture all desired computational behavior. For example, the left hand side of a rewrite rule is treated as a rigid expression which is not refined during the matching of the rule. Given an *f* : Bool → Bool → Bool function, if we add the rewrite rule ∀*x*. f *x* (**not** *x*) = **true**, the expression f **true false** will not be rewritten to true, since it does not rigidly match the **not** *x* on the left hand side. (1.2.1.1)

LET p, q, r, s: x, f, r, s.

((((q&#p)&~#p)=(s=s))>(q&((s=s)&(s@s)))) = ~(s=s) ;
FFFF FFFF FFFF FFFF (1.2.1.2)

Remark 1.2.1.2: Eq. 1.2.1.2 as rendered is *not* tautologous. For the outer antecedent expression, the inner antecedent and consequent are both **F**, contradictory, meaning the conjecture is in fact T, tautologous or (s=s), as **F**>**F**=T.

In practice, this means that an unbounded number of special-cased rules are required to reflect equalities for a type theory. Lifting all the restricting assumptions in the implementation of rewrite rules would require non-trivial research effort. (1.2.2.1)

Remark 1.2.2.1: The "non-trivial research effort" is already implemented using the universal logic Ł4 in the model logic model checker Meth8/VŁ4.

It seems to be difficult to capture the equational theory of a dependent object theory with general-purpose implementations of equality reflection. In the future, robust equality reflection for conversion rules may become available, but until then we have to devise workarounds. If the object theory is similar enough to the metatheory, we can reuse meta-level conversion checking using a shallow embedding. In this paper we describe such a shallow embedding. The idea is that in the standard model of the object theory equations already hold definitionally, and so it would be convenient to reason about expressions built from the standard model as if they came from arbitrary models, e.g. from the syntax.

However, we should only use shallow embeddings in morally correct ways: only those equations should hold in the shallow embedding that also hold in the deeply embedded syntax. (1.2.3.1)

Remark 1.2.3.1: The expression "morally correct" is a mixed metaphor because morality is either good or bad, but logic is either correct or incorrect, hence the figure of speech should read either "morally good" or "logically correct".

To address this, first we prove that shallow embedding is injective up to definitional equality: the metatheory can only believe two embedded terms definitionally equal if they are already equal in the object theory. This requires us to look at both the object theory and the metatheory from an external point of view and reason about embedded meta-level terms as pieces of syntax.

Second, we describe a method for hiding implementation details of the standard model, which prevents constructing terms which do not have syntactic counterparts and which also disallows morally incorrect propositional equalities. This hiding is realised with import mechanisms; we do not formally model it, but it is reasonable to believe that it achieves the intended purposes. (1.2.4.1)

Remark 1.2.4.1: This definition of metatheory and definitional equality is built into the universal logic VŁ4 and implemented in the model logic model checker Meth8/VŁ4.