Synthetic Turing Reducibility in CIC

Yannick Forster[1,2] and Dominik Kirst[1]

[1] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2] Inria, Gallinette Project-Team, Nantes, France

Abstract

We discuss a definition of Turing reducibility in synthetic computability carried out in CIC, the type theory underlying Coq. CIC distinguishes between functions (which can be assumed to all be computable) and total functional relations, allowing for a definition of Turing reducibility via continuous Turing functionals, based on an idea of Bauer.

Turing reductions A problem P is Turing-reducible to a problem Q if P can be solved by a machine which has access to an oracle for Q (the notion was introduced in Turing's PhD thesis [21], but popularised by Post [16]). In less operational models of computation like μ-recursive functions, Turing reductions can be described via functionals $F\colon(\mathbb{N}\rightharpoonup\mathbb{N})\to(\mathbb{N}\rightharpoonup\mathbb{N})$ where the value $F(\alpha)x$ is obtained from the potentially non-computable α and the natural number x solely by the usual computable operations of μ-recursive functions. As a consequence, a μ-recursive functional sends a μ-recursive function α to a μ-recursive function $F(\alpha)$. Kleene [12] and Davis [5] both proved that any μ-recursive functional F is continuous: Whenever $F(\alpha)x \triangleright_\mu y$, then there is a list L included in the domain of α such that whenever β returns the same values on L as $alpha$, then also $F(\beta)x \triangleright_\mu y$. Formally:

$$F(\alpha)x \triangleright_\mu y \to \exists L\colon \mathbb{L}\mathbb{N}.\ (\forall x \in L.\ \exists y.\ \alpha x \triangleright y) \land \forall \beta.\ (\forall x \in L.\ \alpha x = \beta x) \to F(\beta)x \triangleright_\mu y$$

To the best of our knowledge Turing reducibility and (continuous) μ-recursive functionals have not been studied in type theory or constructive (reverse) mathematics, or formalised using machine-checked proofs in a proof assistant. The main hinderance regarding computability theory is the ubiquitous use of the (informal) Church-Turing thesis, which connects "intuitive calculability" with a formal notion (computability in a defined model of computation).

Synthetic computability Synthetic mathematics in general offers a solution for situations where analytic encodings of notions muddle the clear view at the essence of concepts. Introduced by Richman [18] and popularised by Richman, Bridges, and Bauer [4, 1, 2, 3] in synthetic computability one assumes an axiom amounting to imposing a universal function for the space $\mathbb{N}\to\mathbb{N}$, or, equivalently, for $\mathbb{N}\rightharpoonup\mathbb{N}$. For instance, one can state the axiom EPF as

$$\exists \theta\colon \mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N}).\ \forall f\colon \mathbb{N}\rightharpoonup\mathbb{N}.\ \exists c.\ \theta_c \equiv f$$

for a suitable notion of partial functions (e.g. where partial values over X are modelled by step-indexing as monotonous sequences from $\mathbb{N}$ to the option type over X). EPF is closely related to the well-studied axiom CT ("Church's thesis" [14, 20]) in constructive mathematics, which postulates that *all* functions are μ-recursive. CT immediately implies EPF, because θ can be taken to be the universal function for μ-recursive functions, see [8].

In synthetic computability, one can define an undecidable but enumerable problem $\mathcal{K}$, where both enumerability and undecidability are formalised solely in terms of functions. Results like Rice's theorem [1, 8] and Myhill's isomorphism theorem [9], are relatively easy to prove in synthetic computability theory. Synthetic definitions of many-one and truth-table reducibility can be given by just dropping "computable" from textbook definitions, and we have constructed synthetic solutions for Post's problem for many-one and truth-table reducibility [9]. Turing reducibility is less easily synthesised, due to the notion of an oracle. For instance, under the presence of EPF, a functional $F\colon(\mathbb{N}\rightharpoonup\mathbb{N})\to(\mathbb{N}\rightharpoonup\mathbb{N})$ only acts on computable input, thus it cannot be seen as a Turing reduction acting on a (certainly non-computable) oracle input.

Synthetic Turing reducibility We use the intuition behind the mentioned Kleene/Davis theorem to define synthetic Turing reducibility, based on *continuous Turing functionals*. We follow an idea by Bauer [3] and define Turing functionals based on a two-layered approach: They consist of a (continuous) functional $F\colon(Y\rightsquigarrow\mathbb{B})\to(X\rightsquigarrow\mathbb{B})$ mapping functional relations $Y \rightsquigarrow \mathbb{B}$ to functional relations $X \rightsquigarrow \mathbb{B}$ factoring through a (then also continuous) computational core $F'\colon(Y\rightharpoonup\mathbb{B})\to(X\rightharpoonup\mathbb{B})$ (see also [7] for a more detailed treatment).

Formally a Turing functional $F\colon(Y\rightsquigarrow\mathbb{B}) \to (X \rightsquigarrow \mathbb{B})$ ...

1. ... is *continuous* if: $\forall R\colon Y\rightsquigarrow\mathbb{B}.\forall x\colon X.\forall b\colon\mathbb{B}.\ FRxb \to \exists L : \mathbb{L}Y.\ (\forall y \in L.\exists b.\ Ryb) \wedge$ $\forall R'\colon Y\rightsquigarrow\mathbb{B}.\ (\forall y \in L.\forall b.\ Ryb \to R'yb) \to FR'xb$
2. ... factors through a *computational core* $F'\colon(Y\rightharpoonup\mathbb{B})\to(X\rightharpoonup\mathbb{B})$ if:

$$\forall f\colon Y\rightharpoonup\mathbb{B}.\forall R\colon Y\rightsquigarrow\mathbb{B}.\ f\ \textit{computes}\ R \to F'f\ \textit{computes}\ FR$$

where $f\colon Z_1\rightharpoonup Z_2$ *computes* a functional relation $R\colon Z_1\rightsquigarrow Z_2$ if $\forall xy.\ Rxy \leftrightarrow fx \triangleright y$.

Note that we follow the same intuition as for μ-recursive functionals to define continuity here. A synthetic Turing reduction from a predicate $p\colon X\to\mathbb{P}$ to a predicate $q\colon Y\to\mathbb{P}$ maps the characteristic relation of q ($\lambda xb.\ qx \leftrightarrow b = \mathsf{true}$) to the characteristic relation of p.

The definition makes crucial use of the fact that functional relations are completely distinct from functions since unique choice is not provable in CIC.

Our definition of Turing reducibility is work-in-progress: We were able to prove various results regarding Turing reducibility with machine-checked proofs in Coq, see below.

Bauer's definition of Turing reducibility Our definition is based on (computable) functional relations $X\rightsquigarrow\mathbb{B}$, whereas Bauer's definition [3] is based on disjoint pairs of (enumerable) predicates $X\to\mathbb{P}$ and using an order-theoretic definition of continuity. Proving our definition of Turing reducibility equivalent without considering continuity is straightforward [7, §9.6]. We also have an equivalence proof w.r.t. continuity now for a variant of Bauer's definition of continuity, following well-known ideas explained for instance in Rogers' book [19, II.3.2].

Machine-checked results Turing reducibility is reflexive, transitive, and transports undecidability. Every truth-table reduction gives rise to a Turing reduction. When a Turing reduction is total for all total inputs and the L in compactness is computable, it is in fact a truth-table reduction (a result Rogers attributes to Nerode [19, Thm. XIX]). Lastly, the hypersimple deficiency predicate of $\mathcal{K}$ is Turing reduction complete, showing that Turing reducibility is strictly more general than truth-table reducibility. All results are explained in [7, §10].

Open questions However, at least three more results for Turing reducibility are needed to have confidence that our synthetic rendering is correct. The Kleene-Post theorem [13], stating that there are incomparable Turing-reducibility degrees, Post's theorem [17], connecting Turing reducibility via the Turing jump to the arithmetical hierarchy, and the Friedberg-Muchnik theorem [10, 15], settling Post's problem by proving that there exists an enumerable, undecidable, but Turing-reducibility incomplete predicate. Central for all three results is, in contrast to the results we have already proved, an enumerator of cores of Turing functional.

Towards a universal core We require a function $\zeta\colon\mathbb{N}\to((\mathbb{N}\rightharpoonup\mathbb{N})\to\mathbb{N}\rightharpoonup\mathbb{N})$ which enumerates at least all possible cores of Turing functionals. Certain variants of such a function are known to be impossible: A computable modulus of continuity applying to itself would be inconsistent in type theory [6]. We are attacking the problem from two directions: First, we are trying to construct ζ from θ either directly or by enriching the definition of Turing reductions with more computational requirements for cores, e.g. by asking for a computable modulus of continuity for F', which possibly has to be continuous itself. Secondly, in a concurrently submitted abstract we are describing our work-in-progress proofs of synthetic variants of the Kleene-Post and Post's theorem, identifying sufficient properties of a universal function ζ [11].

References

[1] Andrej Bauer. First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31, 2006. doi:10.1016/j.entcs.2005.11.049.

[2] Andrej Bauer. On fixed-point theorems in synthetic computability. *Tbilisi Mathematical Journal*, 10(3):167–181, 2017. doi:10.1515/tmj-2017-0107.

[3] Andrej Bauer. Synthetic mathematics with an excursion into computability theory (slide set). *University of Wisconsin Logic seminar*, 2020. URL: http://math.andrej.com/asset/data/madison-synthetic-computability-talk.pdf.

[4] Douglas Bridges and Fred Richman. *Varieties of constructive mathematics*, volume 97. Cambridge University Press, 1987. doi:10.1017/CBO9780511565663.

[5] Martin D. Davis. *Computability and Unsolvability.* McGraw-Hill Series in Information Processing and Computers. McGraw-Hill, 1958.

[6] Martín Hötzel Escardó. Computability of continuous solutions of higher-type equations. In Klaus Ambos-Spies, Benedikt Löwe, and Wolfgang Merkle, editors, *Mathematical Theory and Computational Practice, 5th Conference on Computability in Europe, CiE 2009, Heidelberg, Germany, July 19-24, 2009. Proceedings*, volume 5635 of *Lecture Notes in Computer Science*, pages 188–197. Springer, 2009. doi:10.1007/978-3-642-03073-4_20.

[7] Yannick Forster. *Computability in Constructive Type Theory.* PhD thesis, Saarland University, 2021. URL: https://ps.uni-saarland.de/~forster/thesis.

[8] Yannick Forster. Parametric Church's Thesis: Synthetic computability without choice. In *International Symposium on Logical Foundations of Computer Science*, pages 70–89. Springer, 2022. doi:10.1007/978-3-030-93100-1_6.

[9] Yannick Forster, Felix Jahn, and Gert Smolka. A Constructive and Synthetic Theory of Reducibility: Myhill's Isomorphism Theorem and Post's Problem for Many-one and Truth-table Reducibility in Coq (Full Version). preprint, February 2022. URL: https://hal.inria.fr/hal-03580081.

[10] Richard M Friedberg and Hartley Rogers Jr. Reducibility and completeness for sets of integers. *Mathematical Logic Quarterly*, 5(7-13):117–125, 1959. doi:10.1002/malq.19590050703.

[11] Dominik Kirst, Niklas Mück, and Yannick Forster. Synthetic versions of the Kleene-Post and Post's theorem. In *28th International Conference on Types for Proofs and Programs (TYPES 2022)*, 2022.

[12] Stephen C. Kleene. *Introduction to metamathematics*, volume 483. van Nostrand New York, 1952.

[13] Steven C. Kleene and Emil L. Post. The upper semi-lattice of degrees of recursive unsolvability. *The Annals of Mathematics*, 59(3):379, May 1954. doi:10.2307/1969708.

[14] Georg Kreisel. Mathematical logic. *Lectures in modern mathematics*, 3:95–195, 1965. doi:10.2307/2315573.

[15] Albert Abramovich Muchnik. On strong and weak reducibility of algorithmic problems. *Sibirskii Matematicheskii Zhurnal*, 4(6):1328–1341, 1963.

[16] Emil L. Post. Recursively enumerable sets of positive integers and their decision problems. *bulletin of the American Mathematical Society*, 50(5):284–316, 1944. doi:10.1090/S0002-9904-1944-08111-1.

[17] Emil L. Post. Degrees of recursive unsolvability - preliminary report. In *Bulletin of the American Mathematical Society*, volume 54, pages 641–642. American Mathematical Society (AMS), 1948.

[18] Fred Richman. Church's thesis without tears. *The Journal of symbolic logic*, 48(3):797–803, 1983. doi:10.2307/2273473.

[19] Hartley Rogers. Theory of recursive functions and effective computability. 1987.

[20] Anne Sjerp Troelstra and Dirk van Dalen. Constructivism in mathematics. vol. i. *Studies in Logic and the Foundations of Mathematics*, 26, 1988.

[21] Alan Mathison Turing. Systems of logic based on ordinals. *Proceedings of the London mathematical society*, 2(1):161–228, 1939. doi:10.1112/plms/s2-45.1.161.