Formally Computing with the Non-Computable

Liron Cohen

Dept. of Computer Science, Ben-Gurion University, Israel
cliron@bgu.ac.il

Abstract. Church–Turing computability, which is the standard notion of computation, is based on functions for which there is an effective method for constructing their values. However, intuitionistic mathematics, as conceived by Brouwer, extends the notion of effective algorithmic constructions by also admitting constructions corresponding to human experiences of mathematical truths, which are based on temporal intuitions. In particular, the key notion of infinitely proceeding sequences of freely chosen objects, known as *free choice sequences*, regards functions as being constructed over time. This paper describes how free choice sequences can be embedded in an implemented formal framework, namely the constructive type theory of the Nuprl proof assistant. Some broader implications of supporting such an extended notion of computability in a formal system are then discussed, focusing on formal verification and constructive mathematics.

1 Introduction

Church–Turing computability is the standard notion of computation. It defines the computable functions as those for which there is an effective method for obtaining the values of the function. Turing used the term 'purely mechanical', whereas Church used 'effectively calculable':

> "define the notion ... of an effectively calculable function of positive integers by identifying it with the notion of a recursive function of positive integers (or of a λ-definable function of positive integers)." [13]

Intuitionistic mathematics, which originated in the ideas of L.E.J. Brouwer, extends the Church–Turing notion of computability by putting forward novel forms of computation, namely the bar induction principle and the continuity principle. *Bar induction* is a strong intuitionistic induction principle which is equivalent to the classical principle of transfinite induction [34] [1], while the *continuity principle for numbers* states that all functions from $\mathbb{N} \to \mathbb{N}$ to $\mathbb{N}$ are continuous. Brouwer used the bar induction principle to derive the fan theorem, which was used in turn, together with the continuity principle for numbers, to derive the *uniform continuity principle* [12, Thm 3]. The uniform continuity principle states that every continuous function on a closed interval of the reals into

[1] Variants of bar induction were shown to be compatible with constructive type theory, and used to enhance the logical functionality implemented by proof assistants [29,32].

the reals is *uniformly continuous* and has a supremum. Historians of mathematics consider that "in just ten lines a revolution was launched" [40].

But to obtain these foundational principles, the standard function space had to be expanded to include non-recursive functions. For this, Brouwer proposed accepting non-lawlike computations, and thus he introduced the bold notion of *choice sequences*. Choice sequences are fundamental objects that are at the core of intuitionistic mathematics. They are never-finished sequences of objects created *over time* by continuously picking elements from a previously well-defined collection, e.g., the natural numbers.[2] Choice sequences can be *lawlike*, in the sense that they are determined by an algorithm (i.e., standard computable functions), or *lawless* (i.e., *free*), in the sense that they are not subject to any law (e.g., generated by throwing dice), or a combination of both. Free choice sequences are described as

> "new mathematical entities ... in the form of infinitely proceeding sequences, whose terms are chosen more or less freely from mathematical entities previously acquired..." [11]

While this notion clearly steps out of the realm of sequences constructed by an algorithm, there is a mental conception of how to create such sequences: the ideal mathematician, or creative subject, can simply pick elements as time proceeds. Brouwer used the concept of choice sequences to develop a novel theory of the continuum, defining real numbers as choice sequences of nested rational intervals.

The foundations of Brouwer's intuitionistic mathematics have been widely studied [21,38,35,34,41,26]. These works have examined Brouwer's ideas from a theoretical, foundational point of view. However, the focus of this paper is the study of intuitionistic mathematics and its extended notion of computability in a formal setting, namely that of a proof assistant. We show that while choice sequences, and in particular *free* choice sequences, are considered non-computable in the traditional sense, they can be integrated into a mechanized system and used in computations. This will not only extend the standard notion of computation in theory, but will in practice provide us with a mechanized system in which such forms of computation are supported and utilized, which, in turn, will enable the exploration of the wider implications of the resulting computational theory.

Currently, the standard Church–Turing notion of computability is the one that underlies the computational theories invoked by standard constructive type theories, which in turn are the basis of extant proof assistants such as Nuprl [14,2], Coq [17], and Agda [1].[3] Thus, for example, the elements of the function type $\mathbb{N} \to T$ are taken to be the effective (computable) functions from the type of the natural numbers, $\mathbb{N}$, to the type T. The integration of the notion of choice sequences into the constructive type theory would entail, among other things, that choice sequences whose elements are chosen from T become first-class citizens of the function type $\mathbb{N} \to T$.

[2] For simplicity, throughout this paper we focus on choice sequences of natural numbers.

[3] For a survey of the status of Church Thesis in type-theory-based proof assistants see [20].

While many of Brouwer's intuitionistic principles and theorems have already been implemented in the Nuprl proof assistant [30,29,28], the constructive type theory underlying Nuprl has only recently been extended to fully integrate the notion of choice sequences. This extension required major modifications to the type theory, starting from the structure of the underlying library of definitions and lemmas, and working up to the semantics of the type system. The current paper presents the main components of that extension, with full details available in [6,7]. We further discuss the wider implications of the resulting mechanized theory, especially with respect to formal verification and constructive mathematics.

2 Integrating Choice Sequences into a Proof Assistant

The Nuprl proof assistant [14,2] implements a type theory called constructive type theory, which is a dependent type theory in the spirit of Martin-Löf's extensional theory [25], based on an untyped functional programming language. It has a rich type theory including equality types, W-types, quotient types, set types, union and (dependent) intersection types, partial equivalence relation (PER) types, approximation and computational equivalence types, and partial types. This section demonstrates how the constructive type theory implemented by the Nuprl proof assistant can be consistently extended to an intuitionistic type theory, that is, a type theory that supports Brouwer's intuitionistic principles. In particular, we focus on the integration of Brouwer's broader sense of computability through an embedding of choice sequences [6,7]. This extended theory provides a formal account of the notion of choice sequences driven by the design constraints of their implementation in a theorem prover.[4]

The Nuprl proof assistant can be (very roughly) described as consisting of the following components. Underlying the whole system is the library, which stores all the definitions and proofs the system currently holds. The computation system encapsulates the operational semantics of the system. The type system defines the type constructors, the behaviors of types and their associated equalities, based on the semantics of types employed. Then there is a set of axioms and inference rules for manipulating the terms and types of the system.[5] Fully integrating choice sequences into Nuprl required a major overhaul to all of the aforementioned components, as we will describe below.

2.1 Storing Choice Sequences in the Library

Choice sequences are implemented as a new type of entry in the digital library of facts and definitions underlying Nuprl, which holds a (finite) list of terms. Thus, the library is used as a *state* in which we store the choices of values that have been made for a particular choice sequence at a given point in time ([31] provides details on the treatment of choice sequences in the library). We utilize the library

[4] The extended framework described was formalized in Coq's formalization of Nuprl's constructive type theory [3,27].

[5] This simplified description omits many components of the system which are not relevant to the current paper.

to perform what is known as memoization, a programming language method originally designed to improve efficiency. In this scheme, we allow the values of a choice sequence to be chosen freely, but once the fifth element of the sequence has been chosen to be 7, say, we store that in the library, and from that point on we return it whenever the input is 5. Thus, the Nuprl library can be extended in two orthogonal directions: by adding more entries to the library, or by adding more values to a specific choice sequence entry. The fact that the library can always be extended allows choice sequences to be represented as finite at any given point in time (i.e., the state of the library), but as infinitely proceeding as the library extends over time. This corresponds to Brouwer's notion of a choice sequence progressing over *time*, as implemented by progressing over *library extensions*.

Concretely, each choice sequence entry in the library has a name, taken from a nominal set of atoms, and it also comes equipped with a restriction[6], in the form of a predicate, which specifies which sequence extensions are valid (e.g., the restriction '$\lambda \mathtt{n}, \mathtt{t}.\ \mathtt{if}\ \mathtt{n} < 10\ \mathtt{then}\ \mathtt{true}\ \mathtt{else}\ 2 \leq \mathtt{t}$' forces the choices starting from position 10 to be greater than or equal to 2). The restriction constitutes a proof obligation that has to be enforced when adding more values to a choice sequence. Using the restriction mechanism we can also define the lawlike choice sequences, by simply posing their generating rule as the restriction.

Since choice sequences are open-ended objects, it may be the case that, to prove a theorem or carry out a computation, the value of a choice sequence at a certain point may need to be known, but at that stage it has yet to be defined. There are different implementation approaches in such cases. In the intuitionistic theory of choice sequences, a reasonable answer is to 'wait until the creative subject picks enough values in the sequence' (consistent with thinking about a choice sequence as the advancement of knowledge over time). This suggests one possible implementation: the system can print out a message to the user asking for more values until there is sufficient data. Another possibility is to have the system automatically fill in values up to the desired place in the sequence, using some number generator. The generator could be random or not, or even probabilistic. The current implementation in Nuprl takes the first approach, but can be combined with the second approach if needed.

2.2 Extending the Computation System

Nuprl's programming language is an untyped (à la Curry), lazy λ-calculus with pairs, injections, a fixpoint operator, etc. For efficiency, integers are primitive and Nuprl provides operations on integers as well as comparison operators. Nuprl's computation system also had to be revised to support choice sequences and, in particular, to make explicit the tight dependency on the library. Fig. 1 presents a subset of Nuprl's extended syntax and small-step operational semantics, where the additional components related to choice sequences are highlighted in blue.

Choice sequences are incorporated as values of the form η, and the new type `Free` is the type of choice sequences. The operational semantics is then extended so that all small-step reduction rules are parameterized by a library, *lib*. In

[6] See, e.g., [37,38,35] for discussions on the various types of restrictions.

$$
\begin{array}{lll}
T \in \texttt{Type} & ::= & \mathbb{N} \mid \mathbb{U}_i \mid \boldsymbol{\Pi} x{:}t.t \mid \boldsymbol{\Sigma} x{:}t.t \mid \{x : t \mid t\} \mid t = t \in t \mid t{+}t \mid \ldots \\
& & \mid \texttt{Free} \;\; \text{(choice sequence type)} \\
v \in \texttt{Value} & ::= & T \mid \star \mid \underline{n} \mid \lambda x.t \mid \langle t, t \rangle \mid \texttt{inl}(t) \mid \texttt{inr}(t) \mid \ldots \\
& & \mid \eta \;\; \text{(choice sequence name)} \\
t \in \texttt{Term} & ::= & x \mid v \mid t\ t \mid \texttt{fix}(t) \mid \texttt{let}\ x := t\ \texttt{in}\ t \mid \texttt{case}\ t\ \texttt{of inl}(x) \Rightarrow t\ \texttt{|}\ \texttt{inr}(y) \Rightarrow t \\
& & \mid \texttt{if}\ t{=}t\ \texttt{then}\ t\ \texttt{else}\ t \mid \ldots
\end{array}
$$

$$
\begin{array}{llll}
(\lambda x.t_1)\ t_2 & \longmapsto_{lib} & t_1[x\backslash t_2] & \texttt{let}\ x_1, x_2 = \langle t_1, t_2 \rangle\ \texttt{in}\ t \longmapsto_{lib} t[x_1\backslash t_1; x_2\backslash t_2] \\
\texttt{fix}(v) & \longmapsto_{lib} & v\ \texttt{fix}(v) & \ldots \\
\eta(i) & \longmapsto_{lib} & \eta[i] \quad \text{, if } \eta[i] \text{ is defined in } lib &
\end{array}
$$

Fig. 1: Extended syntax (top) and operational semantics (bottom)

particular, an application of the form $\eta(i)$ reduces to $\eta[i]$ if $0 \leq i$ and the ith value in the choice sequence named η is available in the current library, in which case $\eta[i]$ returns that value; otherwise it is left undefined.

2.3 Possible Library Semantics

The introduction of choice sequences entails a radical shift in our understanding of mathematical truth. The meaning of proposition $P(\eta)$ mentioning a choice sequence η may not be determined by our current knowledge of η, and so mathematical truth is no longer a *timeless* concept. Instead, truth now depends on current knowledge of η and the possible ways that η may be extended in the future. To support this, the semantics of the Nuprl system was turned into a possible-world-style semantics [24,18], in which the possible worlds correspond to extensions of the library (thus providing a computational interpretation of the possible-world semantics in terms of libraries). In any particular state of the library the semantics is induced by Nuprl's standard realizability semantics.

Nonetheless, the standard Kripke semantics, in which a statement is true in a library only if it is true in *all possible extensions* of the current library, is insufficient to support choice sequences. To demonstrate the problem, consider the claim "there is some value in a given place of a choice sequence" (e.g., formally, $\exists x.\eta(100) = x$). This should be a valid statement in the theory of choice sequences, based on their "infinitely proceeding" nature. However, if in the current stage of the library the choice sequence a has only three values, this will be false under the Kripke-like semantics, since there are extensions of the library in which the 100th value is yet to be filled in. Thus, to support the evolving nature of choice sequences the possible-world semantics has to be more subtle in its treatment of possible extensions.

Two different possible-world semantics that depend on the current Nuprl library lib and its possible extensions, $lib \mapsto lib'$, have been considered [6,7]. The two semantics are especially well-suited to model choice sequences because in both, expressions only need to "eventually" compute to values, which is compatible with the "eventual" nature of choice sequences that are only partially available at a given time, with the promise that they can always be extended in the future. The first semantics is a *Beth-style semantics* [5,18], where $P(\eta)$ is true in library lib when, roughly speaking, there is a bar for lib (i.e., a collection of libraries such

that each path in the tree of library extensions that goes through lib intersects it) in which $P(\eta)$ is true [6]. This is equivalent to saying that there is a proof of $P(\eta)$ by bar induction on the tree of possible extensions of lib. Another semantics is a variant of the Beth-style semantics called *open bar* semantics [7]. In the open bar semantics, $P(\eta)$ is true in library lib when, for each extension lib' of lib, P holds for some extension of lib'. The open bar semantics enables a more general bar induction argument and hence validates some classical principles (see Section 3).

2.4 Extending the Type System

These new semantics entail new interpretations of Nuprl's type system, in which types are interpreted as PERs on closed terms. The resulting type systems satisfy all the standard properties (e.g., transitivity and symmetry), but also two additional properties that are unique to such a possible-world interpretation: monotonicity and locality. Monotonicity ensures that true facts remain true in the future, and locality allows one to deduce a fact about the current library if it is true in a bar of that library. While monotonicity is a general feature of possible-world semantics (including Kripke semantics), locality is a distinctive feature of Beth-like models.

3 The Resulting Theories of Choice Sequences

The two semantics induce two theories: the one based on the Beth-style semantics is called BITT, for 'Brouwerian Intuitionistic Type Theory', and the one based on the open bar semantics is called OTT. Both theories fully embed choice sequences as first-class citizens, in the sense that choice sequences inhabit the extended function type $\mathbb{N} \to \mathbb{N}$ (also called the Baire space, $\mathcal{B}$). That is, in both BITT and OTT the following holds: $\eta \in \texttt{Free} \to \eta \in \mathcal{B}$.

3.1 Axioms for Choice Sequences

Both BITT and OTT validate (variants) of the following key properties governing choice sequences that have been suggested in the literature (see, e.g., [39,23]). In what follows we write $\mathcal{B}_n$ for $\mathbb{N}_n \to \mathbb{N}$, where $\mathbb{N}_n = \{k : \mathbb{N} \mid k < n\}$.

Density Axiom	$\boldsymbol{\Pi} n{:}\mathbb{N}.\boldsymbol{\Pi} f{:}\mathcal{B}_n.\boldsymbol{\Sigma}\alpha{:}\texttt{Free}.f = \alpha \in \mathcal{B}_n$
Discreteness Axiom	$\boldsymbol{\Pi}\alpha, \beta{:}\texttt{Free}.(\alpha{=}\beta \in \mathcal{B}){+}(\neg\alpha{=}\beta \in \mathcal{B})$
Open Data Axiom	$\boldsymbol{\Pi}\alpha{:}\texttt{Free}.P(\alpha) \to \boldsymbol{\Sigma} n{:}\mathbb{N}.\boldsymbol{\Pi}\beta{:}\texttt{Free}.(\alpha{=}\beta \in \mathcal{B}_n \to P(\beta))$

The Density Axiom intuitively states that, for any finite list of values, there is a choice sequence that extends it. In BITT, proving its validity required an additional machinery of name spaces for choice sequences (see [6] for full details). In OTT, however, such machinery is not necessary for validating the variant of the statement in which the existential quantifier is 'squashed'. The squashing mechanism erases the evidence that a type is inhabited by squashing it down to a single constant inhabitant using set types: $\downarrow T = \{x : \texttt{True} \mid T\}$ [14, p. 60]. Intuitively, a squashed existential quantifier asserts the existence of an object without specifying how it can be computed. The Discreteness Axiom states that

intensional equality over choice sequences is decidable, and it is easily validated since choice sequences are identified by their names in the library, which are unique.

The Open Data Axiom, roughly speaking, states that if a property (with certain side-conditions essentially ensuring that α is the only free choice sequence in $P(\alpha)$) holds for a free choice sequence, then there is a finite initial segment of that sequence such that this property holds for all free choice sequences with the same initial segment. Since the axiom does not provide information on a specific choice sequence, but rather on the collection of all sequences determined by an initial segment, it constitutes a continuity principle in a sense. The non-squashed continuity principle is, however, incompatible with Nuprl (following similar arguments to those in [22,33,19]). Nonetheless, in OTT two squashed variants of the Open Data Axiom have been validated. The key observation is that when the $\boldsymbol{\Sigma}$ type is $\downarrow$-squashed, there is no need to provide a witness for the modulus of continuity of P at α. Instead, one can simply find a suitable meta-theoretical number in the proof of its validity, without having to provide an expression from the object theory that computes that number.

3.2 Classical Axioms

One main difference between BITT and OTT relates to compatibility with classical logic. BITT is incompatible with classical logic in the sense that it validates the negation of many classically valid principles. In particular, it proves the negation of the $\downarrow$-squashed law of excluded middle, $\neg\boldsymbol{\Pi} P.\downarrow(P+\neg P)$, the negation of Markov's principle (a principle of constructive recursive mathematics [9, ch. 3]), and the negation of the independence of premise axiom (a controversial axiom which is classically true but generally not accepted by constructivists, which was used by Gödel in his famous Dialectica interpretation [4]). Proofs of these negated properties follow similar arguments such as in [16,15]. For example, notice that in order to prove the validity of the $\downarrow$-squashed law of excluded middle, we would have to prove in the metatheory that for all propositions, there exists a bar of the current library such that either the proposition is true at the bar, or that it is false in all extensions of the bar. To prove its negation we show that neither option is valid anymore because choice sequences can always evolve differently when multiple choices are possible. The open bar semantics invoked by OTT, on the other hand, is based on a more relaxed notion of time that is flexible enough to be compatible with classical reasoning. In particular, it enables the validation of the $\downarrow$-squashed law of excluded middle.

4 Implications of the Formalization of Choice Sequences

The integration of Brouwer's ex ed notion of computation into a mechanized proof assistant is not only important from a foundational standpoint, but also has interesting consequences and practical applications. This section informally and briefly discusses such implications in two main fields, namely formal verification and constructive mathematics.

4.1 Formal Verification

Leveraging the foundational, novel computational capabilities that go beyond the Church–Turing notion of computation has the potential to facilitate significant advances in the *internal* verification of complex systems, e.g., distributed protocols. The dominant approach in verification of such systems is *external*; that is, one develops a model of the system and then proves that the system behaves correctly according to its desired specification, assuming this model is correct. While this strategy is extremely flexible (a model can describe any kind of computational system), it has the major disadvantage that the model may be incorrect. Extending the computation system with Brouwer's broader notion of computation enables an *internal* approach in such verifications. This is because the embedding of the notion of choice sequences within the computational system provides a means to internally formalize non-deterministic behaviors. In large distributed systems, a lot of information is, de facto, 'lawless', in the sense that it is unpredictable. It is far too expansive (source and money-wise) to keep track of all the computation steps, and the environment cannot be controlled, and so in practice there is no way to determine, e.g., the order of messages sent. Therefore, one can use standard computable functions to model the processes of a distributed system, and free choice sequences to model sensors (or unpredictable environmental inputs).

4.2 Intuitionistic Mathematics

Standard mathematical discourse is based on classical mathematics, and thus standard textbooks in mathematics contain, e.g., proofs by contradiction or by cases, as well as impredicative structures. Constructive mathematics (e.g., [8]), on the other hand, does not allow "non-constructive" methods of formal proof, and in particular rejects the law of excluded middle. Because of this restriction, the practice of constructive mathematics is often quite remote from the (classical) standard practice of mathematics, and proofs tend to require more elaborate arguments. For example, without some version of the compactness theorem (which, classically, requires the axiom of choice), point-wise versions of continuity and the derivative are of no use and the more complicated notions of uniform continuity and a uniform version of the derivative must be used.

Most works in constructive mathematics adopt E. Bishop's approach [10,8] and remain agnostic towards the fundamental intuitionistic principles such as choice sequences, bar induction and the uniform continuity principle.[7] But these intuitionistic principles, which go beyond constructive mathematics, have the potential to simplify the practice of mathematical theories. For one, the uniform continuity principle obviates the need for the compactness theorem, thus making intuitionistic calculus more elegant than constructive calculus, because restrictions on key theorems can be eliminated. For example, in intuitionistic mathematics we can again use the point-wise versions of continuity and the derivative in a manner similar to the way they are employed in classical mathematics. Thus, Brouwer's intuitionistic mathematics has the computational advantages of constructive

[7] Notable exceptions include, e.g., [36,42].

mathematics, while at the same time enabling proofs that resemble those of classical mathematics to a greater extent than constructive ones.

The computational account of choice sequences in Nuprl also provides a natural framework for the formalization of the Brouwerian, choice-sequence-based constructive real numbers, and, in turn, the development of the corresponding real analysis, and the exploration of its computational benefits. Brouwer used choice sequences to define the constructive real numbers as sequences of nested rational intervals. The standard formalization of the reals, even in classical or constructive mathematics, is also achieved via converging sequences. Nonetheless, there are two major differences between the standard formalizations and the intuitionistic (Brouwerian) formalization. First, in the intuitionistic formalization, a real number *is* the choice sequence itself, as opposed to it being the limit point (i.e., equivalence class). Second, the notion of what these sequences can be incorporates, in the intuitionistic setting, the *free* choice sequences.

Acknowledgments. The author thanks Vincent Rahli, Robert Constable and Mark Bickford as the framework described in the paper is based on a joint ongoing work with them.

References

1. Agda wiki. `http://wiki.portal.chalmers.se/agda/pmwiki.php`.
2. S. F. Allen, M. Bickford, R. L. Constable, R. Eaton, C. Kreitz, L. Lorigo, and E. Moran. Innovations in Computational Type Theory using Nuprl. *Journal of Applied Logic*, 4(4):428–469, 2006.
3. A. Anand and V. Rahli. Towards a Formally Verified Proof Assistant. In G. Klein and R. Gamboa, editors, *ITP 2014*, volume 8558 of *LNCS*, pages 27–44, 2014.
4. J. Avigad and S. Feferman. Gödel's Functional ("Dialectica") Interpretation. *Handbook of proof theory*, 137:337–405, 1998.
5. E. W. Beth. Semantic Construction of Intuitionistic Logic. *Journal of Symbolic Logic*, 22(4):363–365, 1957.
6. M. Bickford, L. Cohen, R. L. Constable, and V. Rahli. Computability Beyond Church-Turing via Choice Sequences. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 245–254, 2018.
7. M. Bickford, L. Cohen, R. L. Constable, and V. Rahli. Open Bar – a Brouwerian Intuitionistic Logic with a Pinch of Excluded Middle. In C. Baier and J. Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL)*, volume 183 of *LIPIcs*, pages 11:1–11:23, 2021.
8. E. Bishop and D. Bridges. *Constructive Analysis*. Springer, New York, 1985.
9. D. Bridges and F. Richman. *Varieties of Constructive Mathematics*. London Mathematical Society Lecture Notes Series. Cambridge University Press, 1987.
10. D. Bridges and F. Richman. *Varieties of Constructive Mathematics*. Cambridge University Press, Cambridge, 1988.
11. L.E.J. Brouwer. Begründung der mengenlehre unabhängig vom logischen satz vom ausgeschlossen dritten. zweiter teil: Theorie der punkmengen. *Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam*, 12(7), 1919.
12. L.E.J. Brouwer. *From frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*, chapter On the Domains of Definition of Functions. 1927.

13. A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363, 1936.
14. R. L. Constable, S. F. Allen, M. Bromley, R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, and S. F. Smith. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Inc., 1986.
15. T. Coquand and B. Mannaa. The Independence of Markov's Principle in Type Theory. In Delia Kesner and Brigitte Pientka, editors, *FSCD 2016*, volume 52 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
16. T. Coquand, B. Mannaa, and F. Ruch. Stack semantics of type theory. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–11, 2017.
17. The Coq Proof Assistant. `http://coq.inria.fr/`.
18. V. H. Dyson and G. Kreisel. *Analysis of Beth's Semantic Construction of Intuitionistic Logic*. Stanford University, 1961.
19. M. H. Escardó and C. Xu. The Inconsistency of a Brouwerian Continuity Principle with the Curry-Howard Interpretation. In *13th International Conference on Typed Lambda Calculi and Applications (TLCA)*, pages 153–164, 2015.
20. Y. Forster. Church's Thesis and Related Axioms in Coq's Type Theory. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
21. S. C. Kleene and R. E. Vesley. *The Foundations of Intuitionistic Mathematics, especially in relation to recursive functions*. North-Holland Publishing Company, 1965.
22. G. Kreisel. On Weak Completeness of Intuitionistic Predicate Logic. *J. Symb. Log.*, 27(2):139–158, 1962.
23. G. Kreisel. Lawless Sequences of Natural Numbers. *Compositio Mathematica*, 20:222–248, 1968.
24. S. A. Kripke. Semantical Considerations on Modal Logic. *Acta Philosophica Fennica*, 16(1963):83–94, 1963.
25. P. Martin-Löf. Constructive Mathematics and Computer Programming. In *Proceedings of the Sixth International Congress for Logic, Methodology, and Philosophy of Science*, pages 153–175, Amsterdam, 1982. North Holland.
26. J. R. Moschovakis. An Intuitionistic Theory of Lawlike, Choice and Lawless Sequences. In *Logic Colloquium'90: ASL Summer Meeting in Helsinki*, pages 191–209. Association for Symbolic Logic, 1993.
27. V. Rahli and M. Bickford. A Nominal Exploration of Intuitionism. In *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs*, CPP 2016, page 130–141, New York, NY, USA, 2016.
28. V. Rahli and M. Bickford. Validating Brouwer's Continuity Principle for Numbers using Named Exceptions. *Mathematical Structures in Computer Science*, 28(6):942–990, 2018.
29. V. Rahli, M. Bickford, L. Cohen, and R. L. Constable. Bar Induction is Compatible with Constructive Type Theory. *J. ACM*, 66(2):13:1–13:35, April 2019.
30. V. Rahli, M. Bickford, and R. L. Constable. Bar induction: The Good, the Bad, and the Ugly. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2017.

31. V. Rahli, L. Cohen, and M. Bickford. A Verified Theorem Prover Backend Supported by a Monotonic Library. In G. Barthe, G. Sutcliffe, and M. Veanes, editors, *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 57 of *EPiC Series in Computing*, pages 564–582, 2018.
32. M. Rathjen. A note on Bar Induction in Constructive Set Theory. *Mathematical Logic Quarterly*, 52(3):253–258, 2006.
33. A. S. Troelstra. A Note on Non-Extensional Operations in Connection With Continuity and Recursiveness. *Indagationes Mathematicae*, 39(5):455–462, 1977.
34. A. S. Troelstra. *Choice Sequences: a Chapter of Intuitionistic Mathematics*. Clarendon Press Oxford, 1977.
35. A. S. Troelstra. Choice Sequences and Informal Rigour. *Synthese*, 62(2):217–227, 1985.
36. A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics, An Introduction*, volume I, II. North-Holland, Amsterdam, 1988.
37. M. van Atten. *On Brouwer*. Wadsworth Philosophers. Cengage Learning, 2004.
38. M. van Atten and D. van Dalen. Arguments for the Continuity Principle. *Bulletin of Symbolic Logic*, 8(3):329–347, 2002.
39. D. van Dalen. An Interpretation of Intuitionistic Analysis. *Annals of mathematical logic*, 13(1):1–43, 1978.
40. D. van Dalen. *L.E.J. Brouwer: Topologist, Intuitionist, Philosopher: How Mathematics is Rooted in Life*. Springer, 2013.
41. W. Veldman. Understanding and Using Brouwer's Continuity Principle. In *Reuniting the Antipodes — Constructive and Nonstandard Views of the Continuum*, volume 306 of *Synthese Library*, pages 285–302. Springer Netherlands, 2001.
42. W. Veldman. Some applications of Brouwer's Thesis on Bars. In Bourdeau H. Atten, B., editor, *One Hundred Years of Intuitionism (1907-2007)*, Publications of the Henri Poincaré Archives, pages 326–340. Berkhäuser, Berlin, 2008.