CS4860-Applied Logic 2020 Course Introduction

Robert Constable

August 30, 2020

Abstract

This course in *Applied Logic* is co-listed in the Computer Science and Mathematics Departments as CS4860 and Math4860. We have been jointly teaching it since at least 1994 [81]. Key elements of the subject have been taught since Aristotle [6] in 350 BC, so for 2,350 years. A lot can happen on that scale, and we can't cover all of it. This offering of the course focuses on the modern period since Computer Science became an important new academic discipline, circa 1962 at Purdue University and then at Cornell University in 1965. Now in 2020, computer science is one of the most relevant and most popular subjects, attracting over 1,600 majors and requiring a large faculty, making CS departments among the largest in universities.

Over its long history, logic has become a deep and broad subject with strong ties to mathematics, philosophy, and computer science. The steadily increasing capabilities of Artificial Intelligence (AI) have substantially enriched the role of computer science in both logic and mathematics. One reason for this is that computers are capable of supporting rigorous reasoning and checking proofs and programs for errors. Computers can **check** formal proofs for correctness without"understanding" any of the concepts the way humans do. As computers become more able to help in checking proofs we seek further ways to engage them in helping us **discover** proofs. The use of computers has opened a game changing period in the history of mathematics, logic, and computer science [76]. In this context we can see that some logical proofs also act as computer programs that perform the computations implicit in the formal proofs. This connection will be a theme of this course under the heading "proofs as programs" [10].

During world war II Alan Turing worked on building a computer called *Colossus*. For 32 years the existence of this machine was a secret. After the war Turing began to speak about *mechanical intelligence*. He famously said that"if a machine is expected to be infallible, it cannot also be intelligent," (see page 70 of *Darwin among the Machines* which we discuss below). He also said that we want a machine that can learn from experience. This is a goal that we are working on at Cornell. We collect many *fragments* of successful formal proofs, and we apply them to help users make faster progress on creating other formal proofs. Our goal is to significantly speed up and extend this capability. It is one of many methods we are

exploring to find proofs more quickly. It is interesting that after WW2, John von Neumann was asked about what he was working on, and he said "I am thinking about something much more important than bombs, I am thinking about computers." Turing was building computers. His *Colossus* project was a secret for 32 years. Eventually Turing took up PhD studies and research at Princeton University under the supervision of Alonzo Church. His thesis title is *On Computable Numbers*. Commenting on this work around Princeton and quoted in the book *Darwin among the Machines* on page 54 A.K. Dewdney [56] said "By what species of madness might one suppose that all three notions, mechanical procedures, effectively calculable, and recursive functions would turn out to be the same." Now every computer science student knows this, see *Computing Machinery and Intelligence* [94].

Mathematics is one of the oldest academic disciplines. Geometry continues to be taught using Euclid's writings in Alexandria from 325 BC. Computer science has formalized many of these proofs and implemented their constructions [12, 13, 14, 15, 65]. Intuitionistic mathematics originates with L.E.J. Brouwer circa 1907 [29]. The idea of *implementing mathematics* dates to Alan Turing in the 1930's [94]. His goal was to use computers to solve mathematical problems. To enable that we must formulate mathematics so that computers can process it. We can then deploy them to help us calculate and find formal proofs [53], in Latin "to demonstrate". That is what it means to *implement* mathematics.

Computer scientists and mathematicians have created *formal mathematics* starting at least as long ago as Turing and de Bruijn, referenced above. Formal mathematics is highly reliable and increasingly useful as more elements of mathematics are formalized, as we improve our methods of formalization, and as computers steadily become more powerful. For a long time, L.E.J.Brouwer, the founder of intuitionism, stressed the limitations of formalism more than its advantages. It is quite likely he would be enthusiastic about the modern turn of events, e.g. automating elements of intuitionistic mathematics. His insights and writings strongly influence this modern course of events. He probably would have been pleased to learn that at Cornell we are creating *intuitionistic formal mathematics* [86, 20, 66].

We have been formalizing advanced mathematics including number theory, real analysis, automata theory, formal languages, and other computer science topics. We wrote the book *Implementing Mathematics with the Nuprl Proof Development System* published in 1986 [47]. Since then we have been active in using and improving the Nuprl system to formalize interesting mathematics and investigate open problems, some of which we have solved with the help of the Nuprl system. Other proof assistants are similarly engaged. We cite their major accomplishments as well. It is interesting to investigate the potential of more extensive *collaboration among proof assistants*. We touch on that briefly. Over the course of sixty five years, starting with *Automath* [109, 55], the field has rapidly advanced because of the joint engagement of computer science and mathematics. New AI techniques are now accelerating our progress [45]. We greatly benefit from these new methods, and we contribute to them as we explain in this course.

To date Nuprl is the only proof assistant to implement substantial elements of intuitionistic mathematics. Brouwer understood early on that no one could formalize **all** of "his mathematics", only certain elements. No one has yet tried to precisely delineate that boundary. Many computer science concepts in addition to AI, such as *computational complexity*, are relevant in intuitionistic mathematics and will further enrich it as we hope to explore for the first time in this course. This account will provide a novel and fertile examination of

the relationship *between intuitionistic mathematics and computer science.*

1 Background

One way to highlight what is of enduring importance in this course is to quote from George B. Dyson's well known 1997 book *Darwin among the Machines* [58]. He writes in the second paragraph of the preface (page ix) exactly this: "In the game of life and evolution there are three players at the table: human beings, nature, and machines. I am firmly on the side of nature. But nature, I suspect, is on the side of the machines." We want to explore this suspicion as we proceed. Another interesting comment he makes on the same page is that as a child he liked to build tree houses and play in them. He wondered whether trees could think. This author lives among many trees, and has thought about this question as well. I found a book by Peter Wohlleben entitled *The Hidden Life of TREES* [108] which tells us a lot about how trees communicate and help each other fend off termites and other insects that attack them.

Another resource for understanding the philosophical background influencing the foundations of mathematics is the book *Philosophy of mathematics, selected readings* edited by Benacerraf and Putnam [38]. In the adult "comic book" *Logicomix* [57], the heroes are all logicians.

Proof assistants are software systems we use to formalize and implement results in mathematics, to help researchers precisely formulate open problems and solve them, and to discover interesting new problems and conjectures. They are a relatively new technology, circa the 1980's. Proofs of theorems with *computational content* can be *executed*. Most results can be executed on laptops, but some rely on a cluster of processors to help find the proof. As a very simple example, if we formally prove that for any natural number n there is a least prime number p greater than n, then we can execute the proof on a number n and see exactly what the prime number is.

As another example, if we constructively prove that an equation has a real number solution, then we can compute that real number to arbitrary precision by executing the proof. This method of using *constructive proofs as programs* [10] can be provided for number theory, for significant parts of real and complex analysis, for geometry, for constructive and intuitionistic topology and for results in other branches of mathematics. There are now several excellent proof assistants and informative web pages that describe them. We have an especially close relationship with the *Coq proof assistant* [50, 52, 49] because we use it to prove the Nuprl rules correct. We call this a *dual prover technology.*

We were not expecting to create a proof assistant for full *intuitionistic analysis* in the style of

Luitzen Egbertus Jan Brouwer, *L.E.J. Brouwer* for short.[1] One reason is that Brouwer proved that there is no way to implement *all* of intuitionistic mathematics nor even *formalize* all of it. Moreover without using diagonalization techniques, Brouwer proved that the intuitionistic reals are not enumerable. We present that elegant argument later. There are many other results in his intuitionistic mathematics that *have not been formalized* and almost surely can be. Those are results of special interest to us, and we will discuss some of them in this course.

Next we mention the main topics of the course. Our approach is informed by a modern reaction to the quotation from Leopold Kronecker:

"Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk" ("The loving God made the integers, all else is people's work."). In the coming age of *intelligent machines*, we will say something more like this: "In the 20th century humans built *intelligent machines*, all else is our work with our colleagues, students, and these machines."

We can classify the main topics in the course into four somewhat overlapping categories:

LOGIC MATHEMATICS PHILOSOPHY COMPUTER SCIENCE

The elements of **logic** can be traced back to Aristotle, as we mentioned above, to 350 BC, 2,350 years ago. The elements of **mathematics** back to Euclid in 365 BC, 2,385 years ago. It is harder to define the origins of philosophy, possibly Pythagoras, 570 BCE, 2,590 years ago. For computer science we can trace the origin to *Perdue University* in 1962 and *Cornell University* in 1965. Can we imagine what it will be like 2,500 years from now?

2 Formal Intuitionistic Mathematics

Since 2014 we have been formalizing elements of L.E.J.Brouwer's intuitionistic mathematics and implementing them with the *Nuprl proof assistant*. Results are encouraging. For instance, *intuitionistic calculus* turns out to be technically simpler than classical calculus. The *Continuity Principle* is very expressive and useful, yet it contradicts classical mathematics. Brouwer's *Bar Induction* is a clean and elegant form of induction that is classically valid as well. We present formal versions of these key principles and theorems. The fact that continuous functions from closed intervals of the reals into the reals are uniformly continuous saves the work of proving uniform continuity case by case in applications. Many more examples of the *technical advantages* of formal intuitionistic mathematics are included in this course.

[1] His friends called him "Bertus."

The task of implementing mathematics brings together themes at the interface between mathematics and computer science. Computer science (CS) is a young and vibrant branch of science with roots in mathematics. There are at least four *research threads* that bring these two disciplines together. One is the **foundations of mathematics** thread, tracing its origins as far back as Euclid, at least 300 BCE. Another is the thread of **checking proofs using computers**, active since at least 1970 in the work of N.G. de Bruijn [53]. This theme led naturally to **formal proof creation with computer assistance** driven by the creation of proof assistants in computer science. We focus on two proof assistants, Coq [16] and Nuprl [47], because they are closely related and they work together. There are other excellent proof assistants, some mentioned later. The fourth theme is **theorem and proof discovery** using proof assistants. This is part of the thriving area of computer science called *Artificial Intelligence* (AI). It is deployed to make modern proof assistants "smarter." There have been new discoveries in mathematics as a result of this synergistic combination of research themes and computer engineering advances. We see no end in sight to the steadily advancing synergy between technology, mathematical discovery, human imagination. We will explore all four themes.

Starting circa 1985 the Nuprl proof assistant was used to implement *elements of constructive mathematics* including constructive *real analysis* as presented in Bishop's book *Foundations of Constructive Analysis* [23] and in its enrichment in the Bishop and Bridges version *Constructive Analysis* [25]. From that implementation, we began to verify textbook material for real analysis *using Nuprl*. In 1986 the PRL research group wrote the book *Implementing Mathematics with the Nuprl proof development system* [47]. For various reasons, that book has been cited almost every week since it was first published by *Prentice-Hall*. By now there are over 2,100 citations. A large team of doctoral students contributed to the proof assistant and the book. Our current effort to implement intuitionistic mathematics is undertaken with a smaller group of very experienced computer science researchers. They are the co-authors on the five articles that document our implementation of core intuitionistic mathematics at Cornell University from 2014 until 2019.

In 2020 we are now using this implementation to explore elements of intuitionistic mathematics including calculus, topology, homotopy theory, and Cubical Type Theory (CTT). This challenging and exciting effort motivates a search for new ways of making Nuprl "smarter". Human intuition has always been critical in creating new mathematics, and *we can now explore more deeply connections between human intuition and machine intelligence.*

We have formalized the Nuprl type theory proof rules using the *Coq proof assistant* [51]. Nowadays we do not alter the Nuprl logical rules or add new ones without verifying them in Coq. We call this a *dual prover technology.* The Coq and Nuprl teams have worked with each other in a variety of ways since 1985 until today. We have collaborated on projects some of which we mention in this course. One outcome is that Coq will have access to elements of intuitionistic mathematics that we formalize.

We began writing these notes to determine the plausibility of creating a textbook for logic in

computer science which provided appropriate. We are able to draw on Brouwer's extensive writings, also on the book *The Foundations of Intuitionistic Mathematics* [69] by Stephen Cole Kleene and Richard E. Vesley, the book by Heyting [62, 63] and the books by A.S. Troelstra [90, 91, 92] and by Dirk van Dalen [100]. We also depend on these resources [71, 67] as well as [37]. Moreover, Kleene's former PhD student, Professor Joan Rand Moschovakis, has written incisively on intuitionism [78, 77]. Her twenty eight page 1999 article with co-author Garyfallia Vafeiadou entitled *Intuitionistic Mathematics and Logic* [79] is in our view one of the very best insightful and compact accounts of intuitionistic mathematics written in English. We draw on this material and explicitly reference several of their insights and explanations.

3 Propositional Logic

The logical system for propositional logic is especially easy to understand. It uses variables such as P,Q,R,... to stand for *atomic propositions*. The logical *connectives* are: **and**, symbollically ($\wedge$), **or**, symbollically $\vee$, **implies**, symbollically $\Rightarrow$, and **not**,symbollically $\neg$, the constant **true**, symbollically T or $\top$, and the constant **false**, symbollically $\bot$.

On Aristotle: The *law of excluded middle* can be summarized as the idea that every proposition P must be either true or false but not both P and $\neg P$. In Aristotle's words, It is necessary for the affirmation or the negation to be true or false. This is an *exclusive or*: A is true or A is false. Exactly one and only one alternative holds. but we might not know which. We write this law as $A \vee \neg A$.

Once we have the axioms and theorems expressed in logical notation, we can ask for their precise meaning. The standard approach is to consider the *truth functional semantics* of these statements. We can use *truth tables* to give a precise semantics and then confirm that all of the axioms are tautologies, that is they have the value true no matter what truth values are assigned to the propositional variables.

We could show the truth table for some simple formulas. These are widely available in any logic book, such as Smullyan's classic [89] *First–Order Logic*. Smullyan was a PhD student of Alonzo Church, a professional magician, as well as a concert pianist who lived to be 97. His book is magical, and we will use parts of it in this course.

Instead of grounding validity in truth tables, we give the meaning of logical rules and propositions using *functional programs*. We do not take $A \vee \neg A$ as an axiom, as is customary, although we use it as a proposition. We see this in analyzing the proposition $\neg\neg A \Rightarrow A$. In some of our writings, the justification for $A \vee \neg A$ is given as $magic(A)$. This is appropriate given Smullyan's background. The operator *magic* uses a non computational "magical power" to determine which

disjunct is true. However the magic operator hides the evidence, reminding us of the way a magician hides how a trick is done. We provide the *computational evidence* for a list of 11 propositions below. This kind of *evidence semantics* will concern us more as we go further into the course [43].

Three of these rules, 2,9,1, are more than formulas known to be true because of evidence. They are called **inference rules**. They allow us to draw conclusions by combining formulas. For example in **rule** 2, we assume two propositions, A and $(A \Rightarrow B)$. We can see that if we have access to both of them as steps already established in a proof, then we could deduce B as well. A good way to think of this is to know that we have already established A, so we have access to evidence for that proposition. Call that evidence a. In addition, the evidence term for an implication such as $(A \Rightarrow B)$ is a computable function, call it f; it takes evidence a for A and computes evidence for B. So we know how to compute the evidence for B. It is obtained by computing the value of the expression $f(a)$. Once we do that, we have created evidence for B, and we can use it later in the proof. We keep track of this information because we now have B as a proved line in the proof under development.

- 1a $A \Rightarrow (B \Rightarrow A)$
- 1b $(A \Rightarrow B) \Rightarrow ((A \Rightarrow (B \Rightarrow C)) \Rightarrow (A \Rightarrow C))$
- 2. A, $(A \Rightarrow B)$ / B
- 3. $A \Rightarrow (B \Rightarrow A\&B)$
- 4a. $(A\&B) \Rightarrow A$
- 4b. $(A\&B) \Rightarrow B$
- 5a. $A \Rightarrow (A \vee B)$
- 5b. $B \Rightarrow (A \vee B)$
- 6. $(A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \vee B) \Rightarrow C))$
- 7. $(A \Rightarrow B) \Rightarrow ((A \Rightarrow \neg B) \Rightarrow \neg A)$
- 8^0. $A \vee \neg A$ (This is the classical *dreaded rule* 8.)

  Also see the article *Semantics of intuitionistic propositional logic* [82].

  There is another natural way to understand logical propositions that does not mention truth values. Instead we provide the *computational meaning* of the proposition by showing the "functional program" of this type. It is a program that takes evidence for the type A if there is such evidence, and makes that the value of a constant function of type $B \Rightarrow A$. We can easily write a functional program to accomplish this task. It is exactly this, $\lambda(x.\lambda(y.x))$. So the proof of this proposition is a program. This follows the discipline set out in the 1985

TOPLAS article *Proofs as Programs* [10]. We subsequently generalized this explanation to talk about *evidence semantics* [43].

We now provide functional programs for these 11 propositions listed above. It is likely that the programs can be understood without further explanation, but we provide brief explanations for those that are not elementary, after the list of the simplest cases. Two of the rules, **9** and **12** have a different format. They are *inference rules*. The format for rule 9 is normally like this:

$$\frac{C \Rightarrow A(x)}{C \Rightarrow \forall x.A(x)}$$

Instead we use this: $(C \Rightarrow A(x))\ /\ C \Rightarrow \forall x.A(x)$.

- 1a $A \Rightarrow (B \Rightarrow A)$ The functional program is $\lambda(a.\lambda(b.a))$.
- 1b $(A \Rightarrow B) \Rightarrow ((A \Rightarrow (B \Rightarrow C)) \Rightarrow (A \Rightarrow C))$
- 2. A, $(A \Rightarrow B)\ /\ B$ The program is $ap(\lambda(x.b); a) = b$.
- 3. $A \Rightarrow (B \Rightarrow A\&B)$ the program is $\lambda(x.\lambda(y. < x.y >))$.
- 4a. $(A\&B) \Rightarrow A$ the program is $\lambda(x.left(x))$.
- 4b. $(A\&B) \Rightarrow B$ the program is $\lambda(x.right(x))$.
- 5a. $A \Rightarrow (A \vee B)$ the program is $\lambda(x.inl(x))$.
- 5b. $B \Rightarrow (A \vee B)$ the program is $\lambda(x.inr(x))$.
- 6. $(A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \vee B) \Rightarrow C))$
- 7. $(A \Rightarrow B) \Rightarrow ((A \Rightarrow \neg B) \Rightarrow \neg A)$ The program is $\lambda(f.\lambda(g.\lambda(a.ap(g(a); f(a)))))$
- 8^0. $A \vee \neg A$ the program is $magicA$.

  Brouwer rejected Rule 8. He proposed the alternative $\neg\neg(A \vee \neg A)$ which he said is *intuitionistically correct*. It is easy to see that $\neg(A \vee \neg A)$ is impossible since if we know that $A \vee \neg A$ is not valid, then we know A is not valid, hence $\neg A$. Thus for Brouwer the right rule is the alternative he proposed. Kolmogorov commented that for Brouwer $\neg A$ is a positive existential statement. By assuming A we can find a contradiction, i.e. we can prove *False*, also written as $\perp$. In type theory we think of this as the *empty type* because there is no evidence for it. We can thus also write it as $Void$ so $\neg A$ also means $A \Rightarrow \perp$ or $A \Rightarrow Void$.

  To summarize the subtle point about $\neg\neg(A \vee \neg A)$, it is sensible to take this as an axiom. Another subtle claim is this $(A \vee \neg A) \Rightarrow (\neg\neg A \Rightarrow A$. For this claim, there is a program that provides the *computational evidence*. We will explore this theme further when we formalize the notion of evidence. Intuitively we see a program that provides the evidence. Here it is in a notation we explain more fully later. Except

for the operator *any* it is standard. The operator **any** formalizes the idea that from False, the empty type, anything follows. That is, we have the valid formula $Void \Rightarrow X$ for any proposition X. The witnessing evidence is this function: $\lambda(x.any(x))$. Next we give the complete evidence term for $(A \vee \neg A) \Rightarrow (\neg\neg A \Rightarrow A$, then we analyze it further intuitively.

$$\lambda(d; a.\lambda(x.a); na.\lambda(y.any(y(na))))$$

The variable d is evidence for the disjunction $A \vee \neg A$. The a evidence is for A and the na evidence is for $\neg A$, that is a function $\lambda(x.\ \bot)$. From $\bot$ *we can prove anything*, i.e., False implies anything. In the case of the a evidence, we use it to know $\neg\neg A \Rightarrow A$ by $\lambda(x.a)$. This function produces evidence for A regardless of the input, which is ignored.

We see that the program provides a function, given by a *lambda term*. This function takes evidence d for $(A \vee \neg A)$, the d is mnemonic for *decision*. Based on the evidence it distinguishes two cases. In one case, the evidence a for A is used to form the lambda term $\lambda(x.a)$ This lambda term provides evidence for A regardless of the input. In the case when we have evidence for na for $\neg A$, we see that the term $\lambda(y.any(y(na)))$ uses the fact that $y(na)$ belongs to the empty type $\bot$, and given any element of $\bot$, the *any* operator provides evidence for any type. This is the computational version of the fact that "false implies anything", i.e. we know that $\bot \Rightarrow A$ for any proposition A.

3.1 Brouwer's views on propositional logic

In general, Brouwer objected to formal reasoning and logic without it having a corresponding mathematical meaning. He made many very insightful observations, like noting that the *law of excluded middle*, $P \vee \neg P$, is valid only if we know that unsolvable mathematical problems exist. Other mathematicians were thinking along similar lines, one of them was Kolmogorov who said that $\neg P$ is a positive statement claiming that assuming P we can infer False. He noted that $\neg\neg P$ asserts the consistency of P.

Brouwer said that he only objected to formal reasoning that did not have intuitive mathematical meaning. Nowadays this issue might arise more broadly as Artificial Intelligence leads us to think of formal theories as *completely formal*, in some cases without any connection to mathematical meaning. This leads us to think more deeply about *the nature of machine intelligence.*[2] How deeply do we need to understand how the AI component works in order to have confidence in its contributions? It would be very good to know how Brouwer would have thought about this. *When does formalism lose its mathematical meaning?*

This was an issue between Hilbert and Brouwer, with Hilbert the champion of formal reasoning and Brouwer the critic. Brouwer insisted that intuitionistic mathematics could not be completely formalized. This seems intuitively true and thus far factually

[2]At one time there was a prestigious series of books under exactly this title [75].

true as well. We are only beginning to understand the issue as we attempt to formalize larger and larger parts of intuitionistic mathematics.

In modern times this issue will weigh more heavily as proof assistants make discoveries whose details challenge our abilities to understand them fully in our own "intuitive" way. We might need new concepts and new words to fully grasp what this key issue means for mathematics and for us.

3.2 Brouwer's Continuum

Brouwer says that the continuum is given to us by our *intuition of time*, and it is *inconceivable* that we can *construct* the continuum. Nevertheless the development of calculus and other areas of intuitionistic mathematics are natural in this new setting. We use the system to prove results such as Brouwer's famous *fixed-point theorems*. We can apply the very powerful **Continuity Principle** which is not valid in classical mathematics. Brouwer did not offer a proof of this strong result. We give a proof in this course based on articles by Wim Veldman [103, 104] and also referencing the article by Mark Bickford and Vincent Rahli, *Validating Brouwer's Continuity Principle for numbers using named exceptions* [85] .

3.3 Consequences of choice sequences for logic

In 1908 when Brouwer was writing his PhD thesis he expressed *reservations* about the Law of Excluded Middle, $P \vee \neg P$. However, he did not have a proof that this principle is intuitionistically false. After his work on the Continuity Principle, he could prove in 1918 that *the law of excluded middle is intuitionistically false*. Here is his argument. Using the Continuity Principle he could prove this:

$$\neg\forall x : \mathbf{R}.((x \epsilon \mathbf{Q}) \vee \neg(x \epsilon \mathbf{Q}))$$

We leave the proof as an advanced exercise and return to it later.

The appeal of implementing Bishop's *constructive analysis* [23, 24] and the Bishop and Bridges [25] version is that they wrote two textbooks on the subject. So we see a range of results from deep and surprising theorems, the kind of work Brouwer focused on, to the standard working mathematics of calculus, both for real and complex numbers. *We bring both threads together.* In addition, the elements of calculus are made formal and richer by Brouwer's deep novel insights not normally presented in calculus books.

Computer science has brought many of Brouwer's ideas on logic and mathematics to life by *implementing them* in proof assistants. We have applied these results to problems in computer science, mathematics, applied mathematics and to logic itself. In due course these concepts and theorems will be formalized in other proof assistants as well. In this and related ways, L.E.J.Brouwer has made significant contributions to computer science, a subject that did not exist for most of his life, 1881 to 1966. Joan Rand Moschovakis and Garyfallia Vafeiadou say the following on page 26 of their article cited above: "...in the irony of history, the interest in intuitionistic thought (except among those holding similar philosophical ideas) *springs today mostly from logic and computer science*."

Here is one way to view the ever broadening connection between computer science and mathematics. The synopsis is that computer science, an outgrowth of mathematics, is profoundly changing this centuries old academic subject, at least as old as Euclid who lived circa 300 BC, over 17 centuries ago. The idea to *formalize* mathematics could be dated from Leibniz circa 1700 just to give rough parameters. *Formalization provides a way to calculate with ideas as well as numbers.* We can use the processing power of computers to rapidly explore many plausible paths to formalizing definitions and propositions and to *executing their computational content.*

4 Comparing Bishop and Brouwer

Errett Bishop was well aware of Brouwer's results, but unlike Brouwer, he thought that we could *construct* the continuum. He made a memorable and somewhat dismissive remark about Brouwer's approach that we will quote later. After several years of study, we came to agree with Brouwer in part because of the elegance and usefulness of his results in applications and in streamlining other elements of type theory, and in part because we came to believe in Brouwer's insights and the research generated by them.

Here is a simple argument that shows that Bishop and Bridge's account of the reals does not capture the full continuum. We know that we can enumerate all programs since they are written in a formal language. The language will have a finite number of primitive symbols, and the programs for defining a real number as a converging sequence of rational numbers can be recursively enumerated, say $r_0, r_1, r_2, r_3, ..., r_n, ...$ Each algorithm defining a real number is in this countable list of programs. So in the metatheory we see that there are countably many programs defining real numbers. Bishop shows that from inside the theory, we see an uncountable collection of reals, but in the metatheory there are only countably many. Brouwer's real numbers are uncountable even in the metatheory, so they are actually uncountable, as we would expect.

Brouwer demonstrates that intuitionistic real analysis *cannot be completely formalized.* What is the difference and does it matter? One of the facts about real numbers

that we learn early is that we cannot enumerate them. We say that the set or type of real numbers $\mathbb{R}$ is *uncountable*. We can prove that by the method of diagonalization. Brouwer discovered a much stronger result which is proved entirely differently. He believes that the continuum is given to us *by our intuition*, and furthermore *a construction of the continuum is inconceivable*. He notes that our formalisms can only create denumerable (countable) sets (or types) of individual objects, and the continuum is not like that.

Brouwer's mathematics is also exceptionally well suited for implementing calculus and elements of topology. Moreover, it is also the case that there is something deeply compelling and mysterious about the very idea of the continuum, whether we are thinking about time or space. Moreover, the concept of a real number having "infinite precision" is already intriguing and useful. We find it easy to grasp the natural numbers and to build the rational numbers from them. We can perform the arithmetic operations and know exactly what they mean. But computing with real numbers is totally different. We need to think about "infinite precision" and *unbounded* elaboration of the approximations. Just understanding the square root of 2, $\sqrt{2}$, requires a new way of thinking. We can list the first million digits of this number, and people do that for fun. Below are a few digits. We can find a million digits on-line if we were so inclined. What we know about those million digits is that they are an *approximation* to $\sqrt{2}$. If we square the million digit approximation, the calculation will not give us 2. Indeed, there is an approximation of $\sqrt{2}$ computed to *10 trillion digits*. That is a lot of digits! If we square that one, it will not give us 2 either. So what does it mean that there is a real number approximation with 10 trillion digits? It means basically that we can continue the approximation to as many digits as we want, and that gets to the essence of what real numbers are.

Here is data about Nuprl's contributions to our work. Currently there are 20,000 theorems and 5,000 definitions in the Nuprl library. Creating definitions and proofs required 700,000,000 primitive proof steps used in 1.5 *billion primitive refinements* executed by the proof assistant. The new geometry library already provides 50 geometry tactics. These numbers are suggestive of the substantial contribution of proof assistants in creating extremely trustworthy and useable *implemented mathematics*. They also suggest the increasingly *high value of proof assistants in mathematical discovery*.

Some reals are given by algorithms that allow us to compute as many digits as we want, even beyond ten trillion. In intuitionistic mathematics, real numbers are also given by *free choice sequences*, so there is *no bound* on the number of digits we could produce in principle. We could provide a service that simply generates digits *ad infinitum*. These will define a real number, but it will never be "completely given" in any way we can imagine. There is thus a sense in which *the continuum of time or space is not made up of elements or components*. It is a shared mental construct common to humans just as space is. It may be both fun and interesting to contrast the *time primitive* with the *space primitive*. What is the reason that Brouwer thought time was the more fundamental of the two? What might Einstein say about that?

Listing decimal digits does not explain much. Brouwer found a new way to think about the continuum. He gave each of us a way to write down a possibly unique real number by simply producing a sequence like the one below where we are allowed to *freely make our choices of digits*, on a whim, with no "rhyme or reason". Below is one of those starting with 17, it is *our free choice sequence*, and we can extend it as far as we wish.

17.842664997881111111630000000082992753333777777700002755411990087994624479711349785

$e = 2.7182818284590452353602874713526624977572470936999595749669676277240766303.....$

$\sqrt{2} = 1.414213562373095048801688724209698078569671875376948073176679737990732478462$1....

Here is the square root of 2 out to 76 decimal digits.

$(1.4142135623730950488016887242096980785696718753769480731766797379907324784621)^2 =$
19999999999999999999999999999999999999999999999999999999999999999999999999999998009
1124636867675759782915147641524121975921079074512234828815055440381136 41

How do we add two approximations of a real number? Of what value are free choice sequences in practice? At first glance it could seem like a meaningless object. What makes it valuable? We will examine this question in detail as we explore intuitionistic real analysis further. These sequences are necessary for Brouwer's important and practical result on *uniform continuity*. A key general property of these sequences is that for any finite sequences of natural numbers, we can create a free choice sequence beginning with that sequence. This is called the *axiom of density*.

Physics is formulated in terms of timeless, axiomatic mathematics. A formulation on the basis of intuitionistic mathematics, built on time-evolving processes, would offer a perspective that is closer to our experience of physical reality [60].

Bishop proves that the reals are *uncountable* early in his book. The proof is also in Bishop and Bridges [25]. Here is the theorem he proved.

Theorem 2.19: Let (a_n) be a sequence of real numbers, and let x_0 and y_0 be real numbers where $x_0 < y_0$. Then there exists a real number x such that $x_0 \leq x \leq y_0$ and $x \neq a_n$ for all natural numbers n. Bishop says "This is the famous theorem of *Cantor* that the reals are uncountable."

$$x_0 \leq x \leq y_0 \text{ and } x \neq a_n$$

The basic idea expressed in this theorem is quite general. Brouwer might say that it is impossible to "fill up" *any* proper interval of the reals with a countable list of real numbers.

There is an even simpler proof of this result than what because Bishop and Bridges give because they define real numbers as certain computable sequences of rational numbers. They call the algorithms used to define the sequences *finite routines*. These algorithms are written using a *formal language of symbols*. We know that the type of all algorithms is countable since they are *formal linguistic objects*. So there are only countably many real number descriptions when we look at them in the meta-mathematics. This explanation of the continuum would be completely unacceptable to Brouwer. On the other hand, this account provides a way to compute with reals numbers to very high precision, higher than the notion of *floating point numbers* commonly used in numerical analysis. We read comments such as this about these approximations to real numbers: "there are 126×2^{23} in the half open interval $[0, 1)$. " This is a useful approximation to understanding real numbers, but in constructive and intuitionistic analysis we are looking for an account of the fundamental mathematical concept that underlies our understanding of time and space. *Brouwer took time as the more fundamental basic concept*. See this recent article by van Atten [98] in the *Stanford Encyclopedia of Philosophy*: https://plato.stanford.edu/entries/brouwer/

One view of this result is that it is a constructive version of *Cantor's Theorem* that the real numbers are uncountable in the sense that we can find a real number a not in the given sequence. Interested readers can examine both the Bridges and Bishop proof and Dr. Bickford's formalization of it. It is also interesting to examine another understanding of this issue, not as a version of Cantor's theorem but as an account of the intuitionistic continuum. We know from Brouwer that there are stronger results because the Bishop and Bridges reals leave uncountably many "holes" in the real line. A more informative intuitionistic account would be this theorem.
The No Gap Theorem: Given any two separated Bishop reals, r and t, with $r \neq t$, say $r < t$, we can construct a real b such that $r < b < t$. Moreover, there is a continuum of real numbers in this interval. This is a result of Brouwer's that can be found in many articles and books on intuitionism [1, 97, 98, 102, 32, 63, 30, 37, **?**].

Errett Bishop was well aware of Brouwer's results, but unlike Brouwer, he thought that we could *construct* the continuum. He made a memorable and somewhat dismissive remark about Brouwer's approach that we will quote later. After several years of study, we came to agree with Brouwer in part because of the elegance and usefulness of his results in applications and in streamlining other elements of type theory, and in part because we came to believe Brouwer's insights and the on-going ever broadening research agenda that follows from them [60, 88, 84, 105].

When intuitionist mathematics is used to describe the *evolution of physical systems*, it makes clear, according to Gisin, [60] that time really passes and new information is created. Moreover, with this formalism, the strict determinism implied by Einsteins equations gives way to a quantum-like unpredictability. If numbers are finite and limited in their precision, then nature itself is inherently imprecise, and thus unpredictable.

Here is a simple argument that shows that Bishop and Bridge's account of the reals

does not capture the full continuum. We know that we can *enumerate all programs* since they are written in a formal language. The language will have a finite number of primitive symbols, and the programs for defining a real number as a converging sequence of rational numbers can be recursively enumerated, say $r_0, r_1, r_2, r_3, ..., r_n, ...$ Each algorithm defining a real number is in this countable list of programs. So in the metatheory we see that there are countably many programs defining real numbers. Bishop shows that from inside the theory, we see an uncountable collection of reals, but in the metatheory there are only countably many. Brouwer's real numbers are uncountable even in the metatheory, so they are actually uncountable, as he knew.

In contrast, L.E.J.Brouwer demonstrates that his intuitionistic real analysis *cannot be completely formalized*. What is the difference and does it matter? One of the facts about real numbers that we learn early is that we cannot enumerate them. We say that the set or type of real numbers $\mathbb{R}$ is *uncountable*. We can prove that by the method of diagonalization. Brouwer discovered a much stronger result which is proved entirely differently. He believes that the continuum is given to us *by our intuition*, and furthermore *a construction of it is inconceivable*. Thus he claims that formalisms can only create denumerable (countable) sets or types of individual objects, *and the continuum is not like that*.

Brouwer's mathematics is also exceptionally well suited for implementing calculus and elements of topology. Moreover, it is also the case that there is something deeply *compelling and mysterious about the very idea of the continuum, whether we are thinking about* time *or space*. The concept of a real number having "infinite precision" is already intriguing and useful. It is easy to grasp the natural numbers and to build the rational numbers from them. School students can do this. We can perform the arithmetic operations and know exactly what they mean. But computing with real numbers is totally different. We need to think about "infinite precision" and *unbounded* elaboration of the approximations. Just understanding the square root of 2, $\sqrt{2}$, requires a new way of thinking. We can list the first million digits of this number, and people do that for fun. Below are a few digits. We can find a million digits on-line if we were so inclined. What we know about those million digits is that they are an *approximation* to $\sqrt{2}$. If we square the million digit approximation, the calculation will not give us 2. Indeed, there is an approximation of $\sqrt{2}$ computed to *10 trillion digits*. That is a lot of digits! If we square that one, it will not give us 2 either. So what does it mean that there is a real number approximation with 10 trillion digits? It means basically that we can continue the approximation to as many digits as we want, and that gets more to the essence of what real numbers are.

Some real numbers are given by algorithms that allow us to compute as many digits as we want, even beyond ten trillion. In intuitionistic mathematics, real numbers are also given by *free choice sequences*, so there is *no bound* on the number of digits we could produce in principle. We could provide a service that simply generates digits *ad infinitum*. These will define a real number, but it will never be "completely given" in any way we can imagine. There is this sense that the continuum of time or space is not made up of elements or components. It is a shared mental construct common

to humans just as space is. It may be both fun and interesting to contrast the *time primitive* with the *space primitive*. What is the reason that Brouwer thought time was the more fundamental of the two? What would Einstein say about that?

Physics is formulated in terms of timeless, axiomatic mathematics. A formulation on the basis of intuitionistic mathematics, built on time-evolving processes, would offer a perspective that is closer to our experience of physical reality [60].

5 Number Theory and Rational Numbers

In this section we will present the axioms for number theory expressed in first-order logic. This is often called *elementary number theory*(ENT). We take the axioms from Kleene's classic book *Introduction to Meta-Mathematics* [68] page 82 using his numbering of them.

Kleene's Number Theory Axioms

* 13. $A(0)\&\forall x.(A(x) \Rightarrow A(s(x))) \Rightarrow A(x)$
* 14. $(s(a) = s(b)) \Rightarrow a = b$
* 15. $\neg(s(a) = 0)$
* 16. $(a = b) \Rightarrow ((a = c) \Rightarrow (b = c))$
* 17. $(a = b) \Rightarrow ((s(a) = s(b))$
* 18. $(a + 0) = a$
* 19. $(a + s(b) = s(a + b)$
* 20. $(a \times 0) = 0$
* 21. $(a \times s(b)) = (a \times b) + a$

Here are the axioms from Heyting (it seems)

* 1a: $A \supset (B \supset A) \quad A \Rightarrow (B \Rightarrow A)$
* 1b: $(A \supset B) \supset ((A \supset (B \supset C)) \supset (A \supset C)$
* 1c: $(A \Rightarrow B) \Rightarrow ((A \Rightarrow (B \Rightarrow C)) \Rightarrow (A \Rightarrow C)$
* 2: $A \supset (A \supset B) \supset B \quad A \Rightarrow (A \Rightarrow B) \Rightarrow B$
* 3: $A \supset (B \supset A \wedge B) \quad A \Rightarrow (B \Rightarrow A \vee B)$
* 4a: $(A \wedge B) \supset A \quad (A\&B) \Rightarrow A$
* 4b: $(A \wedge B) \supset B \quad (A\&B) \Rightarrow B$
* 5a: $A \supset (A \vee B) \quad A \Rightarrow (A \vee B)$
* 5b: $B \supset (A \vee B) \quad B \Rightarrow (A \vee B)$

  —-
* 6a: $(A \supset ((B \supset C) \supset (A \vee B \supset C))$
* 6b: $(A \Rightarrow ((B \Rightarrow C) \Rightarrow (A \vee B \Rightarrow C))$
* 7a: $(A \supset B) \supset ((A \supset \neg B) \supset \neg A)$

* 7b: $(A \Rightarrow B) \Rightarrow ((A \Rightarrow \neg B) \Rightarrow \neg A)$
* 8a: $\neg\neg A \supset A \quad \neg\neg A \Rightarrow A$
* 8b: $\neg A \Rightarrow (A \Rightarrow B)$

6 Constructive Type Theory Rules

Before we extended Nuprl to implement intuitionistic mathematics, the theory it implemented was *Constructive Type Theory* (CTT). The Nuprl book [47] discusses this type theory in detail. For comparison to the other theories we mention, here are the elements of CTT. There are rules for these types: Atom, Void, Integers, Lists, Unions, Functions, Products, Quotients, Set-types, Equality rules, Universes, and eight miscellaneous rules. These remain in our intuitionistic theory. So we might call the theory *enriched intuitionistic mathematics*. Instead of our previous induction rule for recursive types [48] we now use *Bar Induction* even though we did not conduct an extensive investigation to see if we have lost expressiveness. We could retain the previous rule if we find good reasons for having two strong induction principles.

The rules are in the Nuprl book [47], available on line at the web page: http://www.nuprl.org/book/
They are in section 8.3 page 149. The rules are called *refinement rules* because they state a goal first, and then provide the subgoals needed to accomplish the goal. The types all occur in *universes*, U_i. There is an unbounded hierarchy of *universes* in the theory, $U_1, U_2, ..., U_i, ...$ Universe U_i belongs to universe U_{i+1}. The idea of universes goes back to *Principia Mathematica* [106]. The rational for this is explained in the Nuprl book and in the article [46]. It renders the Nuprl theory *predicative* in contrast to *impredicative* theories. There is a substantial literature on this topic, but one can understand and use Nuprl fluently without knowing this history.

6.1 Nuprl type theory rules

Here are the rules for the Nuprl type theory as presented in the book [47]. They are listed in Part II section 8, starting on page 155. The rules are organized by types. There are also arithmetical rules for addition, subtraction, multiplication, division, and operations modulo a number. There are other articles that examined the structure of Nuprl's type theory [41].

* First are the rules for **atoms**, the empty type (called **void**), and the **integers**, 0,1,-1,2,-2,3,-3, ... often written as **Z**.
* Atom Formation: $H \vdash$ *atom in* U_i *by intro.*
* Canonical atoms: $H \vdash$ *atom ext sequence of characters by intro.*
* Atom Equality: $H \vdash atom - eq(a; b; t, t')$ This operator tests for equality of atoms.
* Void Formation: $H \vdash$ *void in* U_i *by intro.* The empty type is in all universes.

* Void Hypothesis $H, z : void \vdash T$ *by* $any(z)$ If we assume false we can prove anything.
* **Z** Formation $H \vdash \mathbb{Z}$ *in* U_i *by intro.* The integers are a type.
* **Z** Negative $H \vdash -t$ *in* $\mathbb{Z}$ This is the canonical form of negative integers.
* **Z** Ops $H \vdash op(m, n)$ *in* $\mathbb{Z}$ by op for:$+, -, *, /, mod.$
  $H \vdash m, n$ *in* $\mathbb{Z}$
* **Z** Induction $H1, x : \mathbb{Z}, H2 \vdash T$ *ext* $ind(x; y, z.t1; t2; y, z.t2)$ by elim x new z, y

  $C, z : T(y + 1/x) \vdash T(y/x)$ *ext* $t1$

  $\vdash T(0/x)$ *ext* $t2$

  $y : \mathbf{Z}, 0 < y, z : T(y - 1/x) \vdash T(y/x)$ *ext* $t3$

  Note, if x is free in H2 the optional new z or y are used.
* These are the rules for **Lists**, a key data type for computer science. See page 160 of book.
* Formation 1. $H \vdash U_i$ *ext* $(A\ list)$ *by intro list* $H \vdash U_i$ *ext* A
* Formation 2. $H \vdash (A\ list)$ *in* U_i *by intro* $\vdash A$ *in* U_i
* Canonical 3. $H \vdash (A\ list)$ *ext nil by intro nil at* U_i
* Canonical 4. $H \vdash nil$ *in* A *list by intro at* U_i A *in* U_i
* Canonical 5. $H \vdash A\ listext\ h.t$ *by intro* $\vdash A ext\ h$ $A\ listext\ t$
* Canonical 6. $H \vdash a.b$ *in* A *list by intro*
  $\vdash a$ *in* A $\vdash b$ *in* A *list*
* Non-canonical 7. see Nuprl book page 160.
* Non-canonical 8. see Nuprl book page 160.
* Computation 9a. $H \vdash list_ind(nil; t_b; u, v, w.t_u) = t$ *in* T by reduce 1 $H \vdash t_b = t$ *in* T

The current Nuprl system is based on 311 rules. These include many very simple rules about arithmetic operations. The core type theory is covered in xxxxxx rules. Here is a list of the most interesting of these core rules. A full list is available at this url: http://nuprl.org/wip/rules/rules/ The most interesting rules come from formalizing Brouwer and from the several very strong mathematicians who study and develop intuitionism. This includes Joan R. Moschovakis and Garyfallia Vafeiadou's article *Intuitionistic Mathematics and Logic* [79] which strongly influenced our writing.

One of the most critical intuitionistic rules is the *Continuity Principle*. It contradicts classical mathematics and is perhaps the core idea of intuitionistic logic. We discuss it first before listing other principles and rules. We also discuss Bar Induction, another core principle that replaced the previous strong induction principle in

Nuprl [48, 74]. One of the best accounts of both the Continuity Principle and the Bar Induction principle is in the small book by Mark van Atten, *On Brouwer* [97]. He first discusses the *Weak Continuity Principle for Numbers*. He calls it WC-N. It is quite easy to understand and a good basis for exploring the full Continuity Principle.

The *Weak Continuity Principle* for numbers is this proposition where α and β range over choice sequences and $(\bar{\alpha})m$ stands for $< \alpha(0), \alpha(1), \alpha(2), ..., \alpha(m-1) >$

WC-N:

$$\forall\alpha.\exists x.A(\alpha, x) \Rightarrow \forall\alpha.\exists m.\exists x.\forall\beta.[(\bar{\beta}m = \bar{\alpha}m) \Rightarrow A(\beta, x)].$$

Here is Kleene's formal statement of the Continuity Principle [69].

* 1: Bottom diverges $H \vdash \neg(\bot)\downarrow$

* 2: Bar Induction
  $H \quad f(0)\epsilon\ (X\ 0\ c)$
  $H \vdash f\ 0\ c\ \epsilon\ (X\ 0\ c)$

Wim Weldman gives a nice account of the *Continuity Principle* [104]. We state it next after we introduce notation for the *Baire Space* of all infinite sequences of natural numbers. We use $(\mathbb{N} \Rightarrow \mathbb{N})$, the type of all functions from natural number to the natural numbers for the *Baire Space*.

$(\forall R : ((\alpha : \mathbb{N} \Rightarrow \mathbb{N})) \times \mathbb{N}).\forall\alpha : (\mathbb{N} \Rightarrow \mathbb{N})\ )$
$\Rightarrow (\ (\exists n, m : \mathbb{N}.\forall\beta : (\mathbb{N} \Rightarrow \mathbb{N}).\forall i : \mathbb{N}.((i < n) \wedge (\alpha(i) = \beta(i)))\ ) \Rightarrow R(\beta, m))\ )$

7 Integrating Constructive Synthetic Mathematics with Intuitionistic Type Theory using Cubical Type Theory

The Cornell Nuprl proof assistant now implements a fully *intuitionistic type theory* enhanced with concepts developed and implemented in Nuprl since 1984, e.g quotient types, dependent intersection, recursive types and novel types documented in the 1986 book on the theory and its implementation [47]. As far as we know, there is no other proof assistant that has formalized and implemented Brouwer's intuitionistic mathematics nor its enhancement with the types mentioned above. We call this theory *enhanced intuitionistic type theory*.

Recently we defined this rich new theory in the Coq proof assistant [4]. When we are exploring possible new types and rules we verify their consistency using Coq. We

call this a *dual-prover technology*. That methodology was independently advocated by the late Field's Medalist Vladimir Voevodsky. All of the Nuprl results are available at the library and web page of the proof assistant, www.nuprl.org. Currently there are 20,000 theorems and 5,000 definitions. Creating definitions and proofs required 700,000,000 primitive proof steps used in 1.5 *billion primitive refinements* executed by the proof assistant. The new geometry library already provides 50 geometry tactics. These numbers are suggestive of the substantial contribution of proof assistants in creating extremely trustworthy and useable *implemented mathematics*. They also suggest the increasingly *high value of proof assistants in mathematical discovery*.

The intuitionistic type theory we mentioned above was made possible by the foundational results of Kleene and Vesley in their ground breaking book *The Foundations of Intuitionistic Mathematics* [69] and Kleene's earlier results on intuitionistic number theory [70]. Kleene visited Brouwer in Holland and exchanged insights with him. What is especially interesting about our approach is that Brouwer is also one of the founders of homotopy theory. We know this theory well from our study of Brouwer's work and from our teaching. Virtually everyone knows of his famous *fixed point theorem*, which has many different proofs. In this book we will contribute early on a *constructive proof of Brouwer's epsilon fixed point theorem* in his intuitionistic type theory using homotopic methods. Elements of this work might attract attention from a broad swath of the mathematical community.

Nuprl supports both *analytic mathematics* as in the Real Analysis library and *synthetic mathematics* as in the Geometry library and the Cubical Type theory results [17]. We have considerable experience in translating synthetic results, say as in geometry and homotopy theory into analytic results. It is interesting that Brouwer was one of the key creators of homotopy theory. Our experience in this mode of work has provided the technology to rapidly explore the key notions in Homotopy Type Theory and their translation into intuitionistic analysis. One noteworthy consequence of this architecture is that we can provide an intuitionistic proof of Brouwer's *epsilon fixed point theorem*in all dimensions. From such a proof, we can compute any these epsilon fixed points. Brouwer did not have access to the computing power needed to bring his remarkable theorem to life, we did, cite Mark.

8 Chapter:Computability Beyond Church-Turing Using Choice Sequences

The Cornell Nuprl proof assistant implements a fully *intuitionistic type theory* enhanced with concepts implemented in Nuprl since 1984, e.g quotient types, dependent intersection, recursive types and novel types documented in the 1986 book on the theory and its implementation [47]. As far as we know, no other proof assistant that has formalized and implemented Brouwer's full intuitionistic mathematics nor its enhancement with the types mentioned above. We call this theory *enhanced intuitionistic type*

theory.

Recently we defined this rich new theory in the Coq proof assistant [4]. When we are exploring possible new types and rules we verify their consistency using Coq. We call this a *dual-prover technology.* That methodology was independently advocated by the late Field's Medalist Vladimir Voevodsky. All of the Nuprl results are available at the library and web page of the proof assistant, www.nuprl.org. Currently there are 20,000 theorems and 5,000 definitions. Creating definitions and proofs required 700,000,000 primitive proof steps used in 1.5 *billion primitive refinements* executed by the proof assistant. The new geometry library already provides 50 geometry tactics. These numbers are suggestive of the substantial contribution of proof assistants in creating extremely trustworthy and useable *implemented mathematics.* They also suggest the increasingly *high value of proof assistants in mathematical discovery.*

Nuprl supports both *analytic mathematics* as in the Real Analysis library and *synthetic mathematics* as in the Geometry library and the Cubical Type theory results [17]. We have considerable experience in translating synthetic results, say as in geometry and homotopy theory into analytic results. It is interesting that Brouwer was one of the key creators of homotopy theory. Our experience in this mode of work has provided the technology and experience to rapidly explore the key notions in Homotopy Type Theory and their translation into intuitionistic analysis. One noteworthy consequence of this architecture is that in this proposed project we can provide a constructive proof of Brouwer's *epsilon fixed point theorem* in all dimensions. From such a proof, we can compute any these epsilon fixed points.

9 AI Enhanced Proof Assistants

We have been working for many years to extend the reasoning and discovery capabilities of modern proof assistants. We use *artificial intelligence*, AI, to enhance their autonomous reasoning powers and their ability to collaborate with people and other provers. Proof assistants have been used to solve open problems in mathematics and computer science. As a result they are a platform for exploring advances in AI and for discovering new AI mechanisms useful in mathematics as well as computer science.

The role of proof assistants in computer science is particularly *synergistic.* We use them to extend their own capabilities, to guarantee that they work correctly, to expand their scope, and to discover new ways of making them "smarter."

The main proof assistant we use is Nuprl (pronounced as "new pearl"). It was built at Cornell starting in 1985, about thirty five years ago, and has been used and enhanced ever since. Over thirty graduate students worked to build and improve the system. Now we also use the Coq proof assistant to verify the correctness of Nuprl rules with respect to their intended meaning and to validate any new mechanisms or rules we

add [4, 20].

Nuprl implements an *intuitionistic type theory* based on the ideas of L.E.J. Brouwer and inspired by the *Automath* system of N.J. de Bruijn [54]. This body of work is not implemented in other proof assistants, and we believe this topic opens rich new opportunities to bring additional AI methods to bear on making Nuprl more capable. We like to think of making the system "smarter."

10 Computer Science 2025

Accelerating progress in computer science continues to stretch the bounds of human imagination. Research universities and major corporations such as Amazon, Apple, Dell, Google, Facebook, IBM, Intel, Microsoft, and others are changing daily life and modern science, infusing it with technologies that look increasingly magical.

The very nature of computer science and the mechanisms it builds create instances of intelligence that go beyond what individual humans can master. Already science-fiction visions look less imaginative than the reality that supplants them. We seek a deeper and bolder vision of computer science to explore before it comes upon us unforseen.

11 What the Future May Hold

Computer science, as a relatively young discipline is making considerable progress in helping mathematicians and computer scientists solve open problems and discover new ones. Proof assistants are a critical technology in this kind of research. That fact was very well understood by the Fields Medalist Vladimir Voevodsky who turned to computer science to help validate his ideas in homotopy theory. In particular he sought assistance in proving what he called his *Univalence Axiom*. This axiom is critical to understanding equality in homotopy theory. His conjecture was confirmed at Cornell by Mark Bickford working with Voevodsky using the Nuprl proof assistant [17, 19]. The PRL research group has been enriching and applying the Nuprl proof assistant since 1985 [47], for over thirty four years with at least 30 PhD students working on and with the system along with a strong technical support staff including Richard Eaton, Anne Trostle, Sarah Sernaker, and Ariel Kellison.

Proof assistants help researchers accomplish some of the most conceptually difficult yet essential tasks arising in mathematics and science, especially in computer science. They will help us build highly reliable self driving cars just as they now help pilots fly big jets. Nuprl has been used to precisely and formally define such tasks and then help accomplish them. It has been used to solve open problems in computing theory and mathematics, such as proving completeness for intuitionistic first-order logic,

iFOL, [42], as well as creating provably correct versions of some of the most conceptually complex algorithms needed in practice, e.g. the *Paxos* consensus protocol for distributed systems [87, 73, 83].

The Nuprl developers and researchers now use a *dual prover architecture* with the Coq proof assistant [16]. Coq is used to insure that the Nuprl rules are formally sound and that extensions of Nuprl preserve soundness. This capability helps extend the scope of Nuprl's reasoning power, applications, and discoveries. We discuss here the transformative use of this technology thus far, and we make predictions about its increasingly important roles, some of which we never imagined at the beginning in 1979 when we started work on proof assistants at Cornell, inspired by deBrujin's *Automath* system [109, 55, 80, 107, 44].

Stephen Hawking wrote about artificial intelligence (AI) in his 2018 book *Brief Answers to the Big Questions* [61]. In Chapter 9, *Will Artificial Intelligence Out Smart Us*, he writes on page 184:"If computers continue to obey Moore's Law, doubling their speed and memory capacity every eighteen months, the result is that computers are likely to overtake humans in intelligence at some point in the next one hundred years (2,118). When that happens we will need to ensure that computer have goals aligned with ours." He goes on to say "As development in these areas and others moves from laboratory research to economically viable technologies, a virtuous circle evolves, whereby even small improvements in performance are worth large sums of money, prompting further and greater investments in research." A bit further on he says "*Success in creating AI would be the biggest event in human history.*" Computer science is well on its way to achieving that. We already see examples in our five years of formalizing elements of intuitionistic mathematics. One of them is an elegant result by Mark Bickford on the continuum, *Connectedness of the continuum in intuitionistic mathematics* [18], which Nuprl helped him discover and formalize.

Bishop dismissed both of these "acts" or ideas. His theory of analysis can be seen as a plausible alternative to intuitionism that can be fully implemented, as we confirmed to a large extent using Nuprl. On the other hand, the practical value of Brouwer's results in analysis, such as the fact that all continuous functions from a closed interval of the reals to the reals are uniformly continuous, have *compelling value*. It turns out that *intuitionistic calculus* is very elegant and powerful as we will show. That fact is important when we are considering applications of real analysis. In addition Brouwer's principle of *Bar Induction* can be implemented, and it provides a *remarkably elegant and strong induction principle* that is widely useful as we learned from experience. Kleene showed that unrestricted Bar Induction conflicts with the continuity principles. To have *Bar Induction* and the *Continuity Principle* [103, 104], Kleene showed that we need computable but not recursive functions [69, 71].

The Nuprl book remains widely cited after 34 years because of the success of its methodology. Since publication of the book, Dr.Mark Bickford has formalized substantial parts of the Bishop and Bridges book. We obtained permission from Springer to link results from the book to his formalization in Nuprl, see http://www.nuprl.org/MathLibrary/C

This work created a *new paradigm* for publishing mathematics. The PRL group has also applied this methodology to a formalization of *Euclidean Geometry* led by Ariel Kellison with assistance from others in the group [65].

Our mind set at Cornell in 1986 was that with sufficient time and resources we could implement Bishop's *entire book*, and that would be a worthy project. But Bishop knew, and we learned first hand, that this would require an amazing amount of additional creativity and hard mathematics, not only filling in details, but also creating constructive versions of results in classical real analysis that were not well understood computationally.

Bishop was skeptical about some of Brouwer's ideas concerning the the full *continuum* of *real numbers*, $\Re$. He made a memorable comment on that topic in his book: "Brouwer thought the continuum would turn out to be discrete unless he personally intervened to prevent it." That is a catchy and clever remark, but it turned out to be true. Brouwer did "intervene," and he did explain why the continuum is not discrete. He did it using the novel concept of *free choice sequences*. We will examine them carefully since they are essential to our fully intuitionistic account of the real numbers. Fortunately there is a rich literature on this topic that we draw on [20, 91]. Brouwer changed mathematics in a profound way, including mathematical logic as well. We explain his ideas in this book and develop several of them further.

Our account draws on and integrates ideas from the exceptionally *outstanding* article on intuitionism by Joan Rand Moschovakis and Garyfallia Vafeiadou entitled *Intuitionistic Mathematics and Logic* [79] as well as other articles by Joan Rand Moschovakis [77, 78, 77]. She, like the first author, was a PhD student of Stephen Cole Kleene whose book with Richard Eugene Vesley on intuitionism, *Foundations of Intuitionistic Mathematics* [72], and his widely used book on logic, *Introduction to Meta-Mathematics* [67] made our work possible.

We have considerable experience implementing related ideas, namely a formalization of Bishop's constructive analysis in his 1967 book *Foundations of Constructive Analysis* [23] and related work [24]. We also implemented ideas from the follow-on book with Bridges [25] *Constructive Analysis* and related topics [28]. It is ironic that in Bishop's 1967 book he comments that Brouwer thought the continuum would turn out to be discrete "unless he personally intervened to prevent it." This is an *historically correct insight* that it took some of the best analysts in the world to grasp, even in this ironic way. We have permission from the publishers to display pages of the Bishop and Bridges book *Constructive Analysis* on the PRL group home page see the link below.

http://www.nuprl.org/MathLibrary/ConstructiveAnalysis/.

Dr. Bickford has created links between definitions and theorems in this on-line copy of Chapter 1 to the Nuprl formalization of the concepts. An important task for future work is to formalize this chapter using our implementation of the relevant concepts

from intuitionistic analysis, and compare the results. [3]

Some references

[78, 77, 95]
[31, 32, 36, 39, 37, 29, 40, 63, 33, 34, 35, 32, 99, 101, 102]
[96, 5, 2, 8, 9, 4, 7, 96, 3, 23, 10, 28, 11, 22, 26, 21, 27, 64, 77]
[90, 91, 93, 92]

References

[1] Peter Aczel. The type theoretic interpretation of constructive set theory: Choice principles. In S.S. Troelstra and D. van Dalen, editors, *The L.E.J. Brouwer Centenary Symposium*. North Holland, 1982.

[2] Peter Aczel. On relating type theories and set theories. In T. Altenkirch, W. Naraschewski, and B. Reus, editors, *Types for Proofs and Programs: International Workshop, TYPES '98, Kloster Irsee, Germany, March 1998*, volume 1657 of *Lecture Notes in Computer Science*, pages 1–18, 1999.

[3] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[4] Abhishek Anand and Vincent Rahli. Towards a formally verified proof assistant. In *International Conference on Interactive Theorem Proving*, pages 95–197, 2014.

[5] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: to Truth through Proof*. Academic Press, Inc., New York, 1986.

[6] Aristotle. *Organon*. Approx. 350 BCE.

[7] Jacob Aron. Beyond knowledge. *New Scientist*, pages 28 –31, 2015.

[8] Sergei Artemov. Uniform provability realization of intuitionistic logic, modality and lambda-terms. *Electronic Notes on Theoretical Computer Science*, 23(1), 1999. `http://www.elsevier.nl/entcs/`.

[9] Sergei Artemov. Explicit provability and constructive semantics. *The Bulletin for Symbolic Logic*, 6(1):1–36, 2001.

[10] Joseph L. Bates and Robert L. Constable. Proofs as programs. *ACM Transactions of Programming Language Systems*, 7(1):53–71, 1985.

[11] Andrej Bauer, Edmund M. Clarke, and Xudong Zhao. Analytica — an experiment in combining theorem proving and symbolic computation. *Journal of Automated Reasoning*, 21(3):295–325, 1998.

[12] Michael J. Beeson. Constructive Geometry. *Proceedings of the tenth Asian logic colloquium*, pages 19–84, 2009.

[3] A similar artifact was created by Ariel Kellison to link her Nuprl formalization of theorems from Euclidean geometry to their presentation with diagrams available in Heath's well known book on Euclid's *Elements* [59].

[13] Michael J. Beeson. Logic of ruler and compass constructions. In S. Barry Cooper, Anuj Dawar, and Benedict Loewe, editors, *Computability in Europe 2012*, Lecture Notes in Computer Science, pages 46–55. Springer, 2012.

[14] Michael J. Beeson. Proof and computation in geometry. In Tetsuo Ida and Jacques Fleuriot, editors, *Automated Deduction in Geometry*, volume 7993 of *Springer Lecture Notes in Artificial Intelligence*, pages 1–30. Springer, 2013.

[15] Michael J. Beeson. A constructive version of Tarski's geometry. *Annals of Pure and Applied Logic*, 2015.

[16] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development; Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer-Verlag, 2004.

[17] Mark Bickford. Formalizing category theory and presheaf models of type theory in Nuprl. *arXiv e-prints*, 2018.

[18] Mark Bickford. Connectedness of the continuum in intuitionistic mathematics. *Mathematical Logic Quarterly*, 2019.

[19] Mark Bickford, Fedor Bogomolov, and Yuri Tschinkel. Vladimir Voevodsky – Work and Destiny. *EMS Newsletter December 2017*, pages 30–31, 2017.

[20] Mark Bickford, Liron Cohen, Robert L. Constable, and Vincent 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, New York, NY, USA, 2018. ACM.

[21] Mark Bickford and Robert Constable. Inductive construction in Nuprl type theory using Bar Induction. In *Proceedings of Types 2014*, 2014.

[22] Mark Bickford, Christoph Kreitz, Robbert van Renesse, and Robert Constable. An experiment in formal design using meta-properties. In J. Lala, D. Mughan, C. McCollum, and B. Witten, editors, *DARPA Information Survivability Conference and Exposition II (DISCEX-II)*, volume II of *IEEE Computer Society Press*, pages 100–107, Anaheim, CA, 2001.

[23] E. Bishop. *Foundations of Constructive Analysis*. McGraw Hill, NY, 1967.

[24] E. Bishop. Mathematics as a numerical language. In *Intuitionism and Proof Theory*, pages 53–71. North-Holland, NY, 1970.

[25] E. Bishop and D Bridges. *Constructive Analysis*. Springer, New York, 1985.

[26] Nick Bostrom. *Superintelligence*. Oxford University Press, 2014.

[27] R. S. Boyer and J. S. Moore. *A Computational Logic*. Academic Press, New York, 1979.

[28] Douglas Bridges and Fred Richman. *Varieties of Constructive Mathematics*. Cambridge University Press, Cambridge, 1988.

[29] L.E.J. Brouwer. *Over de grondslagen der wiskunde (On the foundations of mathematics)*. PhD thesis, Amsterdam and Leipzig, 1907.

[30] L.E.J. Brouwer. Intuitionisme en Formalisme. *Clausen, Amsterdam*, 1912.

[31] L.E.J. Brouwer. Intuitionism and formalism. *Bull Amer. Math. Soc.*, 20(2):81–96, 1913.

[32] L.E.J. Brouwer. Über Definitionsbereiche von Funktionen. *Mathematische Annalen*, 97:60–75, 1927.

[33] L.E.J. Brouwer. Consciousness, philosophy, and mathematics. *Proceedings of the 10th international congress on philosophy, Amsterdam*, pages 1235 – 1249, 1948.

[34] L.E.J. Brouwer. An intuitionist correction of the fixed point. *Proceedings of the Royal Society London*, 213:1 – 2, 1952.

[35] L.E.J. Brouwer. Points and spaces. *Canadian Journal of Mathematics*, 6:1 – 17, 1954.

[36] L.E.J Brouwer. Contradictority of elementary geometry. In Aren Heyting, editor, *L.E.J Brouwer, Collected Works*, pages 497–498. North Holland, 1975.

[37] L.E.J Brouwer. *Brouwer's Cambridge Lectures on Intuitionism.* Cambridge University Press, 1981.

[38] L.E.J. Brouwer. Intuitionism and formalism. In P. Benacerraf and H. Putnam, editors, *Philosophy of mathematics: selected writings.* Cambridge University Press, 1983.

[39] L.E.J. Brouwer and A. Heyting. *Collected Works: Philosophy and foundations of mathematics, edited by A. Heyting.* Collected Works. North-Holland Pub. Co., 1976.

[40] Luitzen Egbertus Jan Brouwer. On continuous one-to-one transformations of surfaces to themselves. *Proc. Kon. Nederl. Akad. Wetench.*, Ser A11:788 – 798, 1909.

[41] R. L. Constable. The structure of Nuprl's type theory. In Helmut Schwichtenberg, editor, *Logic of Computation*, volume 157 of *Series F: Computer and Systems Sciences*, pages 123–156, Berlin, 1997. NATO Advanced Study Institute, International Summer School held in Marktoberdorf, Germany, July 25–August 6, 1995, Springer.

[42] Robert Constable and Mark Bickford. Intuitionistic Completeness of First-Order Logic. *Annals of Pure and Applied Logic*, 165(1):164–198, January 2014.

[43] Robert L. Constable. The semantics of evidence (also appeared as Assigning Meaning to Proofs). *Constructive Methods of Computing Science*, F55:63–91, 1989.

[44] Robert L. Constable. Recent results in type theory and their relationship to Automath. In Fairouz Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, pages 1–11. Kluwer Academic Publishers, 2003.

[45] Robert L. Constable. *Transforming the Academy*: Knowledge formation in the age of digital information. *Physica Plus*, 9:428 – 469, 2007.

[46] Robert L. Constable. The triumph of types: Principia mathematica's impact on computer science. In *Principia Mathematica Anniversary Symposium*, pages 1–10, 2010.

[47] Robert L. Constable, Stuart F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, Douglas J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, James T. Sasaki, and Scott F. Smith. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice-Hall, NJ, 1986.

[48] Robert L. Constable and N. P. Mendler. Recursive definitions in type theory. In *Proceedings of the Logics of Programming Conference*, pages 61–78, January 1985. Cornell TR 85–659.

[49] Thierry Coquand. Metamathematical investigations of a calculus of constructions. In P. Odifreddi, editor, *Logic and Computer Science*, pages 91–122. Academic Press, London, 1990.

[50] Thierry Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

[51] Thierry Coquand and G. P. Huet. Constructions: A higher order proof system for mechanizing mathematics. In *EUROCAL '85*, volume 203 of *Lecture Notes in Computer Science*. Springer-Verlag, 1985.

[52] Thierry Coquand, Bengt Nordström, Jan M. Smith, and Björn von Sydow. Type theory and programming. *Bulletin of the European Association for Theoretical Computer Science*, 52:203–228, February 1994.

[53] N. G. de Bruijn. The mathematical language Automath: its usage and some of its extensions. In J. P. Seldin and J. R. Hindley, editors, *Symposium on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 29–61. Springer-Verlag, 1970.

[54] N. G. de Bruijn. Checking mathematics with computer assistance. *Notices of the American Mathematical Society*, 38(1):8–15, January 1991. Computers and Mathematics Column.

[55] N. G. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In R. P. Nederpelt, J. H. Geuvers, and R. C. De Vrijer, editors, *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*, pages 865–935. Elsevier, Amsterdam, 1994.

[56] A. K. Dewdney. *The Turing Omnibus: 61 Excursions in Computer Science*. Computer Science Press, New York, 1989.

[57] A Doxiadis and C. Papadimitriou. *Logicomix: An Epic Search for Truth*. Ikaros Publications, Greece, 2008.

[58] George B. Dyson. *Darwin Among the Machines: The Evolution of Global Intelligence*. Perseus Books, 1997.

[59] Euclid. *Elements*. Dover, approx 300 BCE. Translated by Sir Thomas L. Heath.

[60] N. Gisin. Mathematical languages shape our understanding of time in physics. *Nat.Phys*, 16:114116, 2020.

[61] Stephen Hawking. *Brief Answers*. John Murray, UK, 2018.

[62] A. Heyting. *Intuitionism, An Introduction*. North-Holland, Amsterdam, 1966.

[63] A. Heyting, editor. *L. E. J. Brouwer Collected Works*, volume 1. North-Holland, Amsterdam, 1975.

[64] Fairouz Kamareddine, Twan Laan, and Rob Nederpelt. Types in logic and mathematics before 1940. *Bulletin of Symbolic Logic*, 8(2):185–245, 2002.

[65] Ariel Kellison, Mark Bickford, and Robert Constable. Implementing Euclid's straightedge and compass constructions in type theory. *Annals of Mathematics and Artificial Intelligence*, Sep 2018.

[66] Ariel Kellison, Mark Bickford, and Robert Constable. Implementing euclid's straightedge and compass constructions in type theory. *Annals of Mathematics and Artificial Intelligence*, 2018.

[67] S. C. Kleene. *Introduction to Metamathematics*. D. Van Nostrand, Princeton, 1952.

[68] S. C. Kleene. *Introduction to Meta-Mathematics*. ISHI Press International, 2009.

[69] S. C. Kleene and R. E. Vesley. *Foundations of Intuitionistic Mathematics*. North-Holland, 1965.

[70] S.C. Kleene. On the interpretation of intuitionistic number theory. *J. of Symbolic Logic*, 10:109–124, 1945.

[71] S.C. Kleene. Recursive functions and intuitionistic mathematics. *Proceeding of the International Congress of Mathematicians*, pages 679–685, 1950.

[72] Stephen Cole Kleene and Richard Eugene Vesley. *Foundations of Intuitionistic Mathematics*. North-Holland, 1965.

[73] Xiaoming Liu, Christoph Kreitz, Robbert van Renesse, Jason J. Hickey, Mark Hayden, Kenneth Birman, and Robert Constable. Building reliable, high-performance communication systems from components. In David Kotz and John Wilkes, editors, *17th ACM Symposium on Operating Systems Principles (SOSP'99)*, volume 33(5) of *Operating Systems Review*, pages 80–92. ACM Press, December 1999.

[74] P.F. Mendler. *Inductive Definition in Type Theory*. PhD thesis, Cornell University, Ithaca, NY, 1988.

[75] Donald Michie. *Machine Intelligence 3*. American Elsivier, New York, 1968.

[76] Peter Millican and Andy Clark. *The Legacy of Alan Turing, Vol. 1: Machines and Thought*. Oxford University Press, New York, 1996.

[77] Joan Rand Moschovakis. A classical view of the intuitionistic continuum. *Annals of Pure and Applied Logic*, 81:9–24, 1996.

[78] Joan Rand Moschovakis. The logic of Brouwer and Heyting. In *Logic from Russell to Church*, pages 77–125. 2009.

[79] Joan Rand Moschovakis and Garyfallia Vafeiadou. Intuitionistic mathematics and logic. Technical report, EU Common Fund, 1999.

[80] R. P. Nederpelt, J. H. Geuvers, and R. C. de Vrijer. *Selected Papers on Automath*, volume 133 of *Studies in Logic and The Foundations of Mathematics*. Elsevier, Amsterdam, 1994.

[81] A. Nerode and R. Shore. Logic for Applications. Springer-Verlag, New York, 1994.

[82] Erik Palmgren. Semantics of intuitionistic propositional logic. Technical report, 2009.

[83] R. De Prisco, B. Lampson, and N. Lynch. Revisiting the paxos algorithm. *Theoretical Computer Science*, 243:35 – 91, 2000.

[84] Univalent Foundations Program. *Homotopy Type Theory*. Univalent Foundations Program, 2013.

[85] Vincent Rahli and Mark Bickford. Validating Brouwer's Continuity Principle for numbers using named exceptions. *Mathematical Structures in Computer Science*, 28:942–990, 2018.

[86] Vincent Rahli, Liron Cohen, Mark Bickford, and Robert Constable. Bar induction is compatible with constructive type theory. *Journal of the ACM (JACM)*, 66(2):13:1–13:35, April 2019.

[87] Nicolas Schiper, Vincent Rahli, Robbert van Renesse, Mark Bickford, and Robert L. Constable. Developing correctly replicated databases using formal tools. pages 395–406, 2014.

[88] Michael Shulman. Brouwer's fixed-point theorem in real-cohesive homotopy type theory. *Math. Structures in Comp. Science*, 28:856 – 941, 2018.

[89] R. M. Smullyan. *First–Order Logic*. Springer-Verlag, New York, 1968.

[90] Anne Sjerp Troelstra. *Metamathematical Investigation of Intuitionistic Mathematics*, volume 344 of *Lecture Notes in Mathematics*. Springer-Verlag, 1973.

[91] A.S. Troelstra. *Choice Sequences*. Oxford University Press, Oxford, 1977.

[92] A.S. Troelstra. Realizability. In S.R. Buss, editor, *Handbook of Proof Theory*, pages 407 – 473. Elsivier Science, 1998.

[93] A.S. Troelstra and D. van Dalen. *Constructivism in Mathematics, An Introduction*, volume I, II. North-Holland, Amsterdam, 1988.

[94] A. M. Turing. Computing Machinery and Intelligence. *Mind*, 59:433–60, 1950.

[95] G. Vafeiadou. *Formalizing Constructive Analysis: a comparison of minimal systems and a study of uniqueness principles*. PhD thesis, 2012.

[96] M. van Atten, D. van Dalen, and R. Tiezen. Brouwer and Weyl: The phenomenology and mathematics of the continuum. *Philosophia Mathematica*, pages 1 – 22, 2001.

[97] Mark van Atten. *On Brouwer*. Wadsworth Philosophers Series. Thompson/Wadsworth, Toronto, Canada, 2004.

[98] Mark van Atten. Luitzen Egbertus Jan Brouwer. *Stanford Encyclopedia of Philosophy*, pages 1 – 17, 2020.

[99] Dirk van Dalen. *Brouwer's Cambridge Lectures on intuitionism*. Cambridge University Press, Cambridge, 1981.

[100] Dirk van Dalen. Intuitionistic logic. In *The Blackwell Guide to Philosophical Logic*, pages 224–257. Blackwell, Oxford, 2001.

[101] Dirk van Dalen. Brouwer's epsilon-fixed point and sperner's lemma. *Theoretical Computer Science*, 412(28):3140 – 3144, 2011. Festschrift in Honour of Jan Bergstra.

[102] Walter P. van Stigt. *Brouwer's Intuitionism*. North-Holland, Amsterdam, 1990.

[103] Wim Veldman. Understanding and using Brouwer's continuity principle. *Department of Mathematics, Nijmegen*, pages 1–48, 2000.

[104] Wim Veldman. Understanding and using brouwers continuity principle. In *Reuniting the Antipodes Constructive and Nonstandard Views of the Continuum*, pages 285– 302. Synthese Library. Springer Netherlands, 2001.

[105] Valdimir Voevodsky. Notes on type systems. School of Math, IAS, Princeton, NJ, 2011.

[106] A.N. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 2nd edition, 1925–27.

[107] Freek Wiedijk. A contemporary implementation of Automath. Talk presented at the Workshop on 35 years of Automath, Heriot-Watt University, Edinburgh, Scotland, April 10–13, 2002.

[108] Peter Wohlleben. *The Hidden Life of TREES*. Greystone Books, Vancouver/Berkley, 2016.

[109] J. Zucker. Formalization of classical mathematics in Automath. In *Colloque International de Logique*, pages 135–145, Paris, 1977. Colloques Internationaux du Centre National de la Recherche Scientifique, CNRS.