Intuitionistic Completeness of First-Order Logic – mPC Case

Robert Constable and Mark Bickford

Abstract

We constructively prove in type theory the completeness of the *minimal Propositional Calculus*, showing that a formula is provable in mPC if and only if it is *uniformly valid* in constructive type theory extended with the intersection operator. Our completeness proof provides an effective procedure *Prf* that converts any uniform evidence into a formal an mPC proof. Mark Bickford has implemented *Prf* in the Nuprl proof assistant.

The fundamental idea behind this completeness proof is that even one polymorphic evidence term in extended intuitionistic type theory, *evd*, witnesses the uniform validity of an mPC formula F. Finding one such uniform type theoretic realizer *evd* guarantees validity because $Prf(F, evd)$ builds a first-order proof of F, establishing its validity and providing a purely logical normalized realizer. We reason using symbolic computing on uniform realizers. This style of reasoning and our proof can be formalized in Nuprl's constructive type theory. These results are published in the article *Intuitionistic completeness of first-order logic* [8].

This result demonstrates the value of uniform validity as a semantic notion for studying constructive logical theories, and it provides new techniques for showing that formulas are not intuitionistically provable.

1 Introduction

1.1 Overview

We introduce the concept of *uniform validity* for minimal and intuitionistic propositional logic, mPC and iPC, and show that it opens a new approach to completeness questions for constructive logics. Here we use the idea to prove constructive completeness theorems with respect to *uniform evidence semantics* for both minimal and intuitionistic first-order logic.

We use the term *evidence semantics* for a particular version of the propositions-as-types principle (also called the Curry Howard isomorphism or proofs-as-programs identification) as it is expressed in extensional constructive type theories. We believe that a plausible case can be made that evidence semantics captures the informal notion referred to as the *Brouwer Heyting Kolmogorov (BHK) semantics* for constructive logics. If this belief is widely shared, then our result could be stated this way:

Our evidence semantics is a version of the propositions-as-types semantics (a.k.a. Curry-Howard *isomorphism*, proofs-as-programs *identification*, type theoretic *realizability*) widely used in computer science as a practical semantics for programming languages and their associated programming logics, see [27].[1]. Evidence semantics is quite different from Beth and Kripke semantics, for which there are also intuitionistic completeness theorems [33]. Our uniform evidence semantics is

[1]This widely used semantics in computer science is based on the work of a large number of logicians, computer scientists, and mathematicians. Several background articles will be explicitly referenced including [18, 11, 7, 21, **?**, 16, 23, 15, 27]

a new logical notion that is a natural extension of the propositions-as-types semantics and which we believe will also be practically useful. Evidence semantics with exceptions is an addition we propose in this article for turning the Friedman translation into a semantic notion.

Some scholars believe that the propositions-as-types semantics is very close to semantic ideas suggested by Brouwer, Heyting and Kolmogorov, called BHK semantics.[2] It is quite clear how Kleene's precise notions of *recursive realizability* relate to the propositions as types semantics and to *type theoretic realizability*. We do not try to precisely characterize the connection between evidence semantics and BHK semantics because apparently additional historical work remains to be done to understand more exactly what Brouwer meant in his writings.[3]

We do not rely on Church's Thesis for any of these results, and according to Kleene [17, 30], our use of the Fan Theorem precludes it.[4]

1.2 Logical background

It is widely agreed that Tarski's semantics [28] for classical first-order formulas faithfully captures their intuitive *truth-value based interpretation*. Gödel's classical completeness theorem for first-order logic is nowadays presented with respect to this Tarskian semantics, showing that an FOL formula is provable if and only if it has the value "true" in all Tarski structures. This has become a fundamental result in logic which is widely taught to undergraduates using many excellent textbook proofs, such as Smullyan's enduring *First-Order Logic* [26].[5]

According to Troelstra [29], the BHK semantics for iFOL is the "intended semantics", faithful to an intuitionistic conception of knowledge. In retrospect we can see a strong connection between BHK semantics and propositions-as-types/ evidence semantics. In contrast to the classical situation, there has been no intuitionistic completeness proof with respect to this semantics. We mention some of the excellent attempts later in this section.

1.3 Previous completeness theorems

Over the last fifty years there have been numerous deep and evocative efforts to formulate completeness theorems for the intuitionistic propositional calculus and for intuitionistic first-order logic modeled after Gödel's Theorem [12, 19, 33, 24]. Some efforts led to apparently more technically tractable semantic alternatives to BHK such as *Beth models* [5, 33], *Kripke models* [20], topological models [10, 28, 25], intuitionistic model theoretic validity [31], and provability logic [2]. Dummett [13] discusses completeness issues extensively.

The value of developing precise mathematical semantics for intuitionistic mathematics in the spirit of Tarski's work dates at least from Beth 1947 [4] with technical progress by 1957 [5]. While completeness questions did not drive this exploration of various forms of constructive semantics, nevertheless, each new semantics was tested by the challenge of finding a completeness proof in addition to explaining with increasing precision the differences between classical and constructive notions of knowledge.

[2] See Troelstra [29] where he states on page 12 "The standard informal interpretation of logical operators in intuitionistic logic is the so-called proof-interpretation or Brouwer-Heyting-Kolmogorov interpretation (BHK-interpretation for short)."

[3] We think there is a good case equating BHK with propositions as types and to type theoretic realizability, but others disagree, even about Kleene's motivations. In relating our work to Kleene's 1945 notion of realizability, we came to see that Kleene altered his motivation for recursive realizability from seeing it as exploring Hilbert's notion of incomplete communication to exploring Brouwer's informal computational semantics. Perhaps not everyone agrees. By 1965 Kleene definitely interpreted his earlier work as an investigation of Brouwer's ideas.

[4] The Computational Type Theory which Nuprl implements was designed in 1984 to use an *open-ended* notion of effective computability from the start [9].

[5] For a direct connection of our methods to other results in Smullyan's book see [6].

So the completeness issue *a la BHK* has been identified as important for sixty four years. A very significant early attempt to base completeness on BHK is the (nonconstructive) work of Läuchli [21, 22] who stressed the notion of *uniformity* as important. None of these efforts provides a constructive completeness theorem faithful to BHK semantics (a.k.a. Brouwer realizability) either for the intuitionistic propositional calculus (IPC) or for the full predicate calculus.

2 Main Theorems

2.1 Evidence semantics

We start by defining first-order languages and then define *realizability evidence semantics* in a form that captures BHK semantics.[6] We note that if we use effective computation with respect to classical oracles, the semantics applies to classical logic as well in some sense. This observation is not important for our main results.

In type theory, the propositions, $\mathbb{P}$, are identified with types.[7] A non-empty type is a true proposition and members of the type are the evidence for the truth of the proposition. An empty type provides no evidence and represents a false proposition. We will not treat false for minimal logic.

Definition 1.

$$\begin{aligned} M(\psi_1 \& \psi_2) &= M(\psi_1) \times M(\psi_2) \\ M(\psi_1 \vee \psi_2) &= M(\psi_1) + M(\psi_2) \\ M(\psi_1 \Rightarrow \psi_2) &= M(\psi_1) \to M(\psi_2) \end{aligned}$$

Thus, any $M \in \mathcal{S}(\mathcal{L})$ assigns a type $M(\psi)$ to a sentence (a formula with no free variables) $\psi \in \mathcal{F}(\mathcal{L})$. $M(\psi)$ is synonymous with the proposition $M \models \psi$, and the members of type $M(\psi)$ are the evidence for $M \models \psi$.

Definition 2. *A sentence $\psi \in \mathcal{F}(\mathcal{L})$ is uniformly valid if there is at least one term that is a member of all the types $M(\psi)$ for $M \in \mathcal{S}(\mathcal{L})$. Such a term is a member of the intersection type*

$$\bigcap_{M \in \mathcal{S}(\mathcal{L})} M(\psi)$$

We write an intersection type $\bigcap_{x \in T} P(x)$ as a proposition using the notation $\forall[x : T].\, P(x)$. The square brackets indicate that evidence for the proposition $\forall[x : T].\, P(x)$ is uniform and does not depend on the choice of x.

2.2 Completeness

We first informally prove a special case of completeness in which evidence for a formula of minimal propositional logic is normal and pure. The term is *normal* if there are no *reducible* redexes, and it is pure if only logical operators occur. In this case the termination argument for the proof generating procedure is by a simple induction on the depth of the normal evidence term. Moreover, this

[6] We use the term *evidence semantics* when we want to avoid confusion with Kleene's specific notion of realizability [18] from 1945 about which there is a very large literature and which is based on general recursive functions and an implicit use of Church's Thesis [30].

[7] Recall that we work in a *predicative* metatheory, therefore the type of all propositions is stratified into orders or levels, written $\mathbb{P}_i$. For these results we can ignore the level of the type or just write $\mathbb{P}$.

special case illustrates well the general method and one of the delicate points concerning evidence structures. It also motivates the definition of evidence structures and the rules for generating them given in Section 6. Even this special case is original, although it could be taught to undergraduates.

Theorem - Completeness of Minimal Propositional Logic with respect to normal pure evidence: We can find (from the following proof) an effective procedure *Prf* such that given any a logical formula $F(\bar{P})$ in minimal propositional logic with *normal uniform pure* evidence *evd*, $Prf(nil, F, evd)$ builds a minimal logic proof, pf, of the formula, $F(\bar{P})$, that is, $\vdash^1_{min} F(\bar{P})$ *by* pf.

Proof

The proof procedure *Prf* we build here is part of a more general procedure that we build later as an explicit extension of this one. *Prf* is defined by the rules given in Section 6, however we can also think of it as implicitly defined by this proof. In that role we call it Prf_3.

We proceed by induction on the depth t of the evidence term *evd*. We are given $nil \models F(\bar{p}), evd$, $depth(evd) = t$. From this we build the minimal logic proof inductively by progressively refining evidence structures. Thus to prove $\bar{h} : \bar{H}(\bar{P}) \models F(\bar{P}), evd(\bar{h})$ with satisfiable hypotheses, it suffices to know that the procedure is correct for any uniformly valid $\bar{h'} : \bar{H'}(\bar{P}) \models F'(\bar{P}), evd'(\bar{h})$ with satisfiable hypotheses and evidence term depth t' where $t' < t$. We define these terms precisely in the next three sections.

First decide whether $evd(\bar{h})$ is canonical or noncanonical. This is a decidable property of any normal evidence term based on its outer operator. This property is fundamental to the computation system in computational and intuitionistic type theories.

Consider first the canonical cases, these can be further classified by the outermost constructors **C1-4**. Then we will consider the noncanonical ones, **D1-3**. We show that the key subterms can be classified by the destructor cases of the proof rules.

Constructor Cases C1 to C4

1. **C1-implies:** *evd* is $\lambda(x.f_2(x))$ and F is $F_1 \Rightarrow F_2$. Generate a "derived evidence structure" by adding the hypothesis $x : F_1$ and changing the goal to get $\bar{H}, x : F_1 \models F_2, f_2(x))$. It is easy to see intuitively that in the derived evidence structure, $f_2(x)$ is still valid in a sense we will make precise later. Moreover, the hypotheses are clearly satisfiable.

   Now execute $Prf_3([\bar{H}, x : F_1], F_2, f_2(x))$ where $[H, x : F_1]$ is the new hypothesis list. The procedure is correct on this argument because the evidence term depth is smaller by one. We see intuitively that $f_2(x)$ has the type F_2 in the context of the expanded hypothesis list. We will prove this later in the article as a general fact about evidence structures.

2. **C2-and:** *evd* is $pair(f_1; f_2)$ and F is $F_1 \& F_2$. Consider the evidence structures $\bar{H} \models F_i, f_i$ for $i = 1, 2$. These are clearly valid and the hypotheses remain satisfiable.

   By the induction hypothesis, executing $Prf_3(\bar{H}, F_1, f_1)$ and $Prf_3(\bar{H}, F_2, f_2)$ will produce subproofs for these sequents.

   The rule $pair(slot_1; slot_2)$ indicates how the subproofs fit together. We know by the induction hypothesis that the procedure succeeds since t decreases by one in each case.

3. **C3-or:** *evd* is $inl(f_l)$ or $inr(f_r)$ and F is a disjunction, say $F_1 \vee F_r$. In this case we examine the evidence structures for $\bar{H} \models F_l, f_l$ and $\bar{H} \models F_r, f_r$. If *evd* is $inl(f_l)$, then because the evidence structure is valid, it must be that f_1 is evidence for F_1, and we execute the proof procedure *Prf* on the first evidence structure with its satisfiable hypotheses. Otherwise execute the proof procedure on the other structure.

   It is easy to see that the evidence term depth decreases by one on the recursive calls to *Prf*.

4. **C4-atomic:** *evd* is a variable v and F is atomic. The only normal evidence of this form arises when the variable labels the hypothesis $v : F$. In this *base case* the evidence term has depth 0. The procedure returns $hyp(v)$.

Next are the cases and the steps we take for *non-canonical evidence* for F, i.e. the destructors. We note that there cannot be a term of the form $op(v)$ where v occurs among the hypotheses and the operator is a non-logical operator because the evidence term must be for a logical formula, and the only variables declared in the context have logical type of a subformula of F. So the evidence is called *pure*.

It remains to consider the following non-canonical elements of evidence types given the structure $\bar{H} \models F, evd$ where *evd* is not reducible further and is normal and pure. The method is to analyze the term *evd* by following the evaluation path through the *principal argument positions*, as defined above in the computation rules, until we reach a variable; that variable prevents further reduction, and it is typed in the hypothesis list $\bar{H}$, e.g. it might be d in the term $spread(decide(d; l.t_1; r.t_r); x, y.e')$. We then continue reduction by expanding the evidence term symbolically in the way specified below. The depth of the evidence term decreases when we do this, and we can see it clearly because the evidence term in normal and pure.

So consider the *outermost subterm* of *evd* encountered on the evaluation path, say $g'(x)$, denote its term context by $evd(...g'(x)...)$, with principal argument x typed in $\bar{H}$. We use a mnemonic naming of the principal variable x to suggest its structure, e.g. f when it is a function, p when it is a pair, c when it is a case split, an element of a disjoint union.

Destructor Cases D1 to D3

1. **D1-implies:** $g'(f)$ is $ap(f; f_1)$ and f is a unique label for an hypothesis $f : F_1 \Rightarrow F_2$. We build from the given evidence structure two generated evidence structures, called *input* and *output*. Then we execute the procedure *Prf* on each of them. Termination is guaranteed by induction, provided we can show that the derived evidence structures are valid and satisfiable. It is easy to see that they are satisfiable, but validity requires a deeper analysis. It will involve the use of a class of *finitary models* that we define in Section 4.

   Consider first the **input** evidence structure $\bar{H}, f : F_1 \Rightarrow F_2, \bar{H}' \models F_1, f_1$ whose evidence term f_1 is a subterm of g' and whose context is $\bar{H}, f : F_1 \Rightarrow F_2, \bar{H}' \models F_1, f_1$.

   According to the induction hypothesis, $Prf_3([\bar{H}, f : F_1 \Rightarrow F_2, \bar{H}'], F_1, f_1)$ will construct a proof, say pf_{in}. The induction hypothesis applies since t decreases by one and the lemmas provide validity and satisfiability. Establishing validity is one of the crucial points of the proof, and we need to invoke properties of special finitary models we construct that guarantee f_1 has the type F_1. Essentially the argument is that we can build a model of $F_1 \Rightarrow F_2$ such that the functions in it will diverge on inputs that do not belong to F_1. This model will be one of those in the intersection of all models, hence we know that f_1 must be in the input type.

   Consider next the generated **output** evidence structure $\bar{H}, f : F_1, v : F_2, \bar{H}' \models F, evd$. Replace all occurrences of f in *evd* by the symbolic constant function $\lambda(x.v)$ for a fresh variable v. Then reduce $evd'(...v...)$ to evd'. All explicit occurrences of $ap(f; f_1)$ in *evd* are thus replaced by v, and these are the only occurrences that will arise. It is noteworthy that we can remove the hypothesis about f since we will not need it again. The depth of evd' is less than t. Note that this depends on using the simple symbolic constant function $\lambda(x.v)$ for v a fresh variable.

   We cite the lemmas to know that this generated evidence structure is valid with satisfiable hypotheses, and we note that the substitution for f enforces an implicit constraint that $v = ap(f; f_1)$ in all evidence structures generated from this one. Under this constraint, the

generated structure has the same meaning as the given one. The hypotheses are clearly satisfiable.

The induction hypothesis guarantees termination of *Prf* whose term depth decreases by one, to $t-1$.

We now add the rule $apseq(f; slot_{in}; v.slot_{out})$. Since we know by the induction hypothesis that the application of *Prf* will succeed on $evd(...v...)$, to produce pf_{out}, we return that value for $slot_{out}$. For $slot_{in}$ we use pf_{in}.

2. **D2-and:** $g'(p)$ is $spread(p; l, r.t(l, r))$. We know that $p : A\&B$ appears in the hypothesis list which we modify to $\bar{H}'$ by replacing $p : A\&B$ with $l : A, r : B$. We create the evidence structure $\bar{H}' \models F, evd'$ where evd' is the subterm obtained by replacing $spread(p; l, r.t(l, r))$ in evd by $spread(pair(l; r); l, r.t(\bar{x}, l, r))$ and then reducing $g(...spread(pair(l; r); l, r.t(\bar{x}, l, r))...)$ as much as possible to g'' which has lower term depth $t' < t$. Note that computation preserves the type.

   Note, these reductions might increase the size of the evidence term, but we see that the logical depth of the input to *Prf* has decreased because we remove the destructor and substitute a variable of depth one for the binding variable of the same depth.

   We also need to *update the constraints of the model* in the generated evidence structures. If there is a constraint $p = exp$ in the given structure, then we add the constraint $exp = pair(l; r)$ in the generated structures.

   We add the rule $spread(p; l, r.slot_{and})$, and g'' will go into the slot. Since $A\&B$ is satisfiable, so are the new hypotheses, $l : A$ and $r : B$.

3. **D3-or:** $g'(d)$ is $decide(d; l.t_a(\bar{x}, l); r.t_b(\bar{x}, r))$ and d appears in the hypothesis list as $d : A \vee B$. Now we generate two valid satisfiable evidence substructures $\bar{H}_1, a : A, \bar{H}_2{}' \models F, evd(...g'(inl(a))...)$ and $\bar{H}_1, b : B, \bar{H}_2{}' \models F, evd(...g'(inr(b))...)$, and we reduce the new evidence terms as much as possible. This symbolic computation might increase the size of the evidence terms; however, the logical depth of the typing context has decreased in each term by the Reducibility Lemma (essentially because we remove the destructor and substitute a variable for the binding variable).

   If there is a constraint $d = exp$ in the given evidence structure, then we add the constraint $d = inl(a)$ in the first case and $d = inr(b)$ in the other.

Qed

One classical approach to first-order completeness is based on systematic search for counter examples to a conjecture. Validity of the conjecture is the reason the search fails – halting with a proof. This approach is well illustrated in Smullyan's enduringly valued monograph *First-Order Logic* [26] and Fitting's monograph [14] where the method is adapted to iFOL with Kripke semantics. Both Smullyan and Fitting use tableaux proofs which go back to the work of Beth [4, 5].[8] Like all other classical proofs of completeness, these are not constructively valid, but they are nearly constructive as Underwood showed in her thesis [32].

We came to our approach because we use on a daily basis the fact that from constructive proofs of theorems in Computational Type Theory (CTT) [9, 1] our proof assistant can automatically extract programs or data that meet the specification given by the theorem.

We hope that our results will add more weight to the notion that there is a deep connection between proving a theorem and writing a program. We have long stressed this idea in papers

[8] Beth invented *semantic tableau* as a bridge from semantics to proofs; we use uniform realizers and their *evidence structures*.

treating *proofs as programs* [1, 3, 9] and conversely *programs as proofs*, additionally in papers treating formal constructive *mathematics as a programming language* [9] where types subsume data types. Here we are treating iFOL as an abstract programming language where formulas are specifications given by dependent types. We build the proof from the program/data type which is a uniform Brouwer realizer.

References

[1] Stuart Allen, Mark Bickford, Robert Constable, Richard Eaton, Christoph Kreitz, Lori Lorigo, and Evan Moran. Innovations in computational type theory using Nuprl. *Journal of Applied Logic*, 4(4):428–469, 2006.

[2] 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/`.

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

[4] Evert W. Beth. Semantical considerations on intuitionistic mathematics. *Indagationes mathematicae*, 9:572 – 577, 1947.

[5] Evert W. Beth. Semantic construction of intuitionistic logic. *Koninklijke Nederlandse Akademie van Wetenschappen, Mededelingen, Nieuwe Reeks*, 19 (11):357 – 388, 1957.

[6] Mark Bickford and Robert L. Constable. Polymorphic logic. In *Logic, Construction, Computation*. Ontos Verlag, 2012. Festschrift for Helmut Schwichtenberg.

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

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

[9] 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.

[10] Thierry Coquand and Jan M. Smith. An application of constructive completeness. In *In Proceedings of the Workshop TYPES '95*, pages 76–84. Springer-Verlag, 1995.

[11] H. B. Curry, R. Feys, and W. Craig. *Combinatory Logic, Volume I*. Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1958.

[12] H.C.M. de Swart. An intuitionistically plausible interpretation of intuitionistic logic. *Journal of Symbolic Logic*, 42:564 – 578, 1977.

[13] Michael Dummett. *Elements of Intuitionism*. Oxford Logic Series. Clarendon Press, 1977.

[14] M. Fitting. *Intuitionistic model theory and forcing*. North-Holland, Amsterdam, 1969.

[15] J-Y. Girard, P. Taylor, and Y. Lafont. *Proofs and Types*, volume 7 of *Cambridge Tracts in Computer Science*. Cambridge University Press, 1989.

[16] W. Howard. The formulas-as-types notion of construction. In *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 479–490. Academic Press, NY, 1980.

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

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

[19] G. Kreisel. Weak completeness of intuitionistic predicate logic. *Journal of Symbolic Logic*, 27:139–158, 1962.

[20] Saul A. Kripke. Semantical analysis of intuitionistic logic. In *Formal Systems and recursive functions*, pages 92–130. North-Holland, Amsterdam, 1965.

[21] H. Lauchli. An abstract notion of realizability for which intuitionistic predicate calculus is complete. In J. Myhill, A. Kino, and R.E. Vesley, editors, *Intuitionism and Proof Theory*, pages 227–34. North-Holland, Amsterdam, 1970.

[22] James Lipton and Michael J. O'Donnell. Some intuitions behind realizability semantics for constructive logic: Tableau and Läuchli countermodels. *Annals of Pure and Applied Logic*, 81:187 – 239, September 1996.

[23] Per 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.

[24] David McCarty. Completeness and incompleteness for intuitionistic logic. *Journal of Symbolic Logic*, 73(4):1315–1327, 2008.

[25] H. Raisowa and R. Sikorski. *The Mathematics of Metamathematics.* Panstowe Wydawnictwo Naukowe, Warsaw, 1963.

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

[27] M.H. Sørensen and P. Urzyczyn. *Lectures on the Curry-Howard Isomoprhism.* Elsevier, 2006.

[28] A. Tarski. Der aussangenkalkul und die topologie. *Fundamenta Mathematicae*, 31:103–134, 1938.

[29] Anne Sjerp Troelstra. History of constructivism in the 20th century. *ITLI Publication Series*, ML-91-05:1–32, 1991.

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

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

[32] Judith L. Underwood. Aspects of the computational content of proofs. Department of Computer Science TR94-1460, Cornell University, Ithaca, NY, October 1994.

[33] W. Veldman. An intuitionistic completeness theorem for intuitionistic predicate calculus. *Journal of Symbolic Logic*, 41:159–166, 1976.