Notes from tech report version of "Intuitionistic completeness of first-order logic" APAL 165, 2014

tions. Kleene used indexes of general recursive functions as *realizers*, and by 1952 [39] he viewed realizability as a formal account of BHK semantics under the assumption of Church's Thesis.

By 1982 Martin-Löf [52, 53] building on the work of Kleene [41] and Tait [70] and Howard [36] refined the informal BHK approach and raised it to the level of a *semantic method* for constructive logics grounded in operational semantics [63]. Already in 1970 Martin-Löf proposed using Brouwer's analysis of bar induction as the meaning of Π^1_1 statements and developed a constructive version of completeness for classical first-order logic [51] based on a constructive topological model of Borel sets in the Cantor Space.

1.6 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 [22, 23, 44, 51, 77, 56]. Some efforts led to apparently more technically tractable semantic alternatives to BHK such as *Beth models* [10, 77], *Kripke models* [46], topological models [19, 28, 71, 64, 51], intuitionistic model theoretic validity [74], and provability logic [2]. Dummett [23] 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 [9] with technical progress by 1957 [10]. 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.

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 [47, 49] 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.

The closest correspondingly faithful constructive completeness theorem for *intuitionistic validity* is by Friedman in 1975 (presented in [74]), and the closest classical proof for the Brouwer-Heyting-Kolmogorov (propositions as types/proofs as terms/proofs as programs) semantics for intuitionistic first-order logic may be from 1998 by Artemov using *provability logic* [2]. Results suggest how delicate completeness theorems are since constructive completeness with respect to full intuitionistic validity contradicts *Church's Thesis* [44, 74] and implies *Markov's Principle* as well [55, 56].[7]

1.7 Constructive type theory with an intersection operator

We first informally discuss *evidence semantics* for minimal logc.[8] Using evidence semantics, we introduce the idea of *uniform validity*, a concept central to our results and one that is also classically meaningful.

This concept provides an effective tool for constructive semantics because we can establish uniform validity by *exhibiting even one polymorphic object* among a possibly unbounded number of them that we might find. For example, the propositional formula $A \Rightarrow A$ is uniformly valid exactly because there is an object in the intersection of the family of all evidence types for this formula indexed by each possible choice of proposition A among the *type of propositions*, $\mathbb{P}$.

[7] Church's Thesis is not an issue for us because we do not assume it.

[8] We can extend this semantics to classical logic if oracle computations are allowed to justify the law of excluded middle, $P \vee \sim P$, with an operator $magic(P)$ [17]. We make some observations about classical logic based on this *classical evidence semantics*.

We write this intersection as $\forall[A : \mathbb{P}].A \Rightarrow A$ or as $\bigcap A : \mathbb{P}.A \Rightarrow A$.[9] In this case, given the extensional equality of functions, the polymorphic identity function $\lambda(x.x)$ is the one and only object in the intersection. So the witness for uniform validity, like the witness for provability, can be provided by a single object. Truth tables provide a single (expensive) piece of evidence for classical propositional logic. There are single witnesses for the validity of all uniformly valid first-order formulas. For example, it will be clear after we provide the evidence semantics that the polymorphic term $\lambda(h.\lambda(x.\lambda(p.h(< x, p >))))$ establishes the uniform minimal (logic) validity of

$$\sim \exists x.P(x) \Rightarrow \forall x.(\sim P(x))$$

hence the uniform intuitionistic and classical validity as well.

Another important observation about uniform validity is that *the formulas of first-order logic that are provable intuitionistically and minimally are uniformly valid.* It is also noteworthy that *the law of excluded middle is not uniformly valid in either constructive or classical evidence semantics.*

The meaning of $False$ also raises the semantic issue that leads us to first consider minimal logic and the Friedman embedding of iFOL into mFOL. Consider the intuitionistically valid assertion $False \Rightarrow A$ for any proposition A. One witnesses for uniform validity is $\lambda(x.x)$, and other witnesses include any constant function, say $\lambda(x.17)$. If we designate a *diverging* term such as *div*, then $\lambda(x.div)$ is also evidence because the claim being made is that if x belongs to the evidence type for $False$, then x or 17 or *div* belongs to the evidence type for A.[10] So according to the informal semantics of intuitionistic logic, the claim $False \Rightarrow A$ is "vacuously true" since no element can be evidence for $False$ whose standard evidence is the empty type.

From the constant function with an arbitrary evidence term *evd*, $\lambda(x.evd)$, we cannot reconstruct the proof of $False \Rightarrow A$. The term *evd* might be entirely misleading. The evidence in the constructive metatheory, CTT, for $False \Rightarrow A$ provided by the CTT proof is $\lambda(x.any(x))$. This evidence suggests a way to provide an alternative semantics for $False$, and thus for iFOL, that avoids the issue just discussed and avoids the need for minimal logic in our account. On the other hand, the minimal logic approach is extremely simple, and it is well known and well studied. So we use that method first and then point out how to avoid it.

We are using the new semantics of $False$ and *iFOL* in our Nuprl proof, and we will account for it much more fully in a future article about the formalization of our proof in CTT.

In minimal logic, there is no atomic propositional constant $False$. Instead the *arbitrary* propositional constant $\perp$ is used, and its interpretation allows non-empty types as well as empty ones. For the same reason, avoiding vacuous hypotheses, we require that all domains of discourse for minimal logic can be non-empty.

1.8 Comparison to intuitionistic validity

Results of McCarty [55, 56] demonstrate that unless one changes in significant ways what one means by completeness (or by validity) or otherwise limits the collection of formulas at issue, then a completeness theorem will be impossible to prove intuitionistically. We have opted to change the notion of validity, not by preconceived choice, but by a discovery.

We discovered that provability is captured exactly by *uniform validity*, an intuitively smaller collection of formulas than those constructively valid. Nevertheless, uniform validity is extremely useful in practice when thinking about purely logical formulas precisely because it corresponds exactly to proof and yet is an entirely semantic notion based on evidence semantics, the semantics that enables strong connections to computer science.

[9] 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 $\mathbb{P}_i$.

[10] We can use the fixed point combinator, say $\mathbf{Y}$ or fix to define *div*. For instance, $fix(\lambda(x.x))$ computes to itself, where fix is an operator such as the $\mathbf{Y}$ combinator $\lambda(f.ap(\lambda(x.ap(f; ap(x; x))); \lambda(x.ap(f; ap(x; x)))))$.

1.9 Counterexamples

Soundness with respect to uniform validity provides a simple method of showing that formulas are not provable by showing that they are not uniformly valid. For example, it is trivial to show that $P \vee \sim P$ is not uniformly valid, e.g. to show $\sim \forall[P : Prop].P \vee \sim P$. Suppose there were a uniform realizer, d. It would have to be an element of the disjoint union type, thus either $inr(\star)$ or $inl(\star)$. If it is an inr term, then pick P to be a true proposition, say $True$, and otherwise pick it to be $False$. These choices show that there can be no such uniform d. Once we have this easy result, we can show that $\sim\sim P \Rightarrow P$ is also not uniformly valid, e.g. we show $\sim \forall[P : Prop]. \sim\sim P \Rightarrow P$. We do this by assuming that $\forall[P : Prop]. \sim\sim P \Rightarrow P$ and using the fact that for any P we can prove $\sim\sim (P \vee \sim P)$, thus if we could prove the uniform statement, we could also prove $\forall[P : Prop].P \vee \sim P$ which we just showed is not uniformly true. By the same technique we can show that Pierce's law is not uniformly valid, e.g. $\sim \forall[P, Q : Prop].((P \Rightarrow Q) \Rightarrow P) \Rightarrow P)$. All the formulas that intuitionistically imply $P \vee \sim P$ are provably false by this method.

We can also show that first-order Markov's Principle (MP) is not uniformly valid; that is $\forall x.(P(x) \vee \sim P(x)) \;\&\; \sim \forall x. \sim P(x)) \Rightarrow \exists y.P(y)$ is not uniformly valid. This is because we can choose a two element domain D with elements a and b and consider two predicates, P_1 and P_2 which have opposite values on a and b. The existential quantifier must pick one of a or b, but it will be an incorrect choice for one of the predicates.

2 Proof Rules and Proof Expressions

2.1 Proof expressions

We assign denotational meaning to proofs, according to the "proofs as terms" principle (PAT). The rules include constraints on the subexpressions of a proof. This is especially natural in *refinement style logics* studied by Bates [5] and Griffin [27] and used in CTT and *tableaux systems* [11, 67, 24].

For each rule we provide a name that is the *outermost operator* of a proof expression with slots to be filled in as the refinement style proof is developed. The partial proofs are organized as a tree generated in two passes. The first pass is top down, driven by the user creating terms with slots to be filled in on the algorithmic bottom up pass once the downward pass is complete. Here is a simple proof of the intuitionistic tautology $A \Rightarrow (B \Rightarrow A)$.

$$\vdash A \Rightarrow (B \Rightarrow A) \; by \; \lambda(x.slot_1(x))$$

$$x : A \vdash (B \Rightarrow A) \; by \; slot_1(x)$$

In the next step, $slot_1(x)$ is replaced at the leaf of the tree by $\lambda(y.slot_2(x, y))$ to give:

$$\vdash A \Rightarrow (B \Rightarrow A) \; by \; \lambda(x.slot_1(x))$$
$$x : A \vdash (B \Rightarrow A) \; by \; \lambda(y.slot_2(x, y)) \; for \; slot_1(x)$$
$$x : A, y : B \vdash A \; by \; slot_2(x, y)$$

When the proof is complete, we see the slots filled in at each inference step as in:

$$\vdash A \Rightarrow (B \Rightarrow A) \; by \; \lambda(x.\lambda(y.x))$$
$$x : A \vdash (B \Rightarrow A) \; by \; \lambda(y.x)$$
$$x : A, y : B \vdash A \; by \; x$$

We present the rules in a top down style showing the *construction rules* first, often called the *introduction rules* because they introduce the canonical proof terms or called the *right hand side* rules because they apply to terms on the right hand side of the turnstile. So typical names seen in

the literature are these: for & we say *AndIntro* or *AndR*; for $\Rightarrow$ we say *ImpIntro* or *ImpR*; for $\vee$ we say *OrIn-r* or *OrIn-l*, and for $\forall x$ we say *AllIntro* or *AllR*, and for $\exists$ we say *ExistsIntro* or *ExistsR*.

For each of these construction rules, the constructor needs subterms which build the component pieces of evidence. Thus for *AndR* the full term will have slots for the two pieces of evidence needed, the form will be $AndR(slot1, slot2)$ where the slots are filled in as the proof tree is expanded. When the object to be filled in depends on a new hypothesis to be added to the left hand side of the turnstile, the rule name supplies a unique label for the new hypothesis, so we see a rule name like $ImpR(x.slot(x))$ or $AllR(x.slot(x))$. In the case of the rule for $\exists$, there is a subtlety. The rule name provides two slots, but the second depends on the object built for the first, so we see rule names such as $ExistsR(a; slot(a))$.

For each connective and operator we also have rules for their occurrence on the left of the turnstile. These are the *rules for decomposing* or using or *eliminating* a connective or operator. They tell us how to use the evidence that was built with the corresponding construction rules, and the formula being decomposed is always named by a label in the list of hypotheses, so there is a variable associated with each rule application. Here are typical names: for & we say $AndElim(x)$ or $AndL(x)$. However, there must be more to this rule name because typically new formulas are added to the hypothesis list, one for each of the conjuncts, so we need to provide labels for these formulas. Thus the form of elimination for & is actually $AndL(x; l, r.slot(l, r))$ where l stands for the left conjunct and r for the right one.

Rule names such as *AndR*, *AndL*, OrR_l, OrR_r, *OrL*, and so forth are suggestive in terms of the details of the proof system, but they are not suggestive of the structure of the evidence, the semantics. We will use rule names that define the computational forms of evidence. The *evaluation rules* for these proof terms are given as in ITT or CTT, for instance in the book *Implementing Mathematics* [18] or in the Nuprl Reference Manual [45].

So instead of $AndR(a; b)$ where a and b are the subterms built by a completed proof by progressively filling in open slots, we use $pair(a; b)$ or even more succinctly $< a, b >$, and for the corresponding decomposition rule we use $spread(x; l, r.t(l, r))$ where the binding variables l, r have a scope that is the subterm $t(l, r)$. This term is a compromise between using more familiar operators for decomposing a pair p such as $first(p)$ and $second(p)$ or $p.1$ and $p.2$ with the usual meanings, e.g., $first(< a, b >) =< a, b > .1 = a$. The reason to use *spread* is that we need to indicate how the subformulas of $A\&B$ will be named in the hypothesis list.

The decomposition rules for $A \Rightarrow B$ and $\forall x.B(x)$ are the most difficult to motivate and use intuitively. Since the evidence for $A \Rightarrow B$ is a function $\lambda(x.b(x))$, a reader might expect to see a decomposition rule name such as $apply(f; a)$ or abbreviated to $ap(f; a)$. However, a Gentzen sequent-style proof rule for decomposing an implication has this form:

$H, f : A \Rightarrow B, H' \vdash G$ *by* $ImpL$ *on* f
1. $H, f : A \Rightarrow B, H' \vdash A$
2. $H, f : A \Rightarrow B, v : B, H' \vdash G$

As the proof proceeds, the two subgoals 1 and 2 with conclusions A and G respectively will be refined, say with proof terms $g(f, v)$ and a respectively. We need to indicate that the value v is $ap(f; a)$, but at the point where the rule is applied, we only have slots for these subterms and a name v for the new hypothesis B. So the rule form is $apseq(f; slot_a; v.slot_g(v))$ where we know that v will be assigned the value $ap(f; slot_a)$ to "sequence" the two subgoals properly. So *apseq* is a sequencing operator as well as an application, and when the subterms are created, we can evaluate the term further as we show below. We thus express the rule as follows.

$H, f : A \Rightarrow B, H' \vdash G$ *by* $apseq(f; slot_a; v.slot_g(v))$
$H, f : A \Rightarrow B, v : B, H' \vdash G$ *by* $slot_g(v)$
$H, f : A \Rightarrow B, H' \vdash A$ *by* $slot_a$