CS4860-2018fa-Lecture 5

Robert L. Constable

Abstract

We will examine the notion of *evidence semantics* that we illustrated using Euclidean geometry.

1 Review of Evidence Semantics and Further Observations

One way to think about evidence semantics is to mention specific examples of propositions that are well known for their significance and remain as open problems in mathematics, such as the P = NP conjecture in complexity theory. We don't know whether P = NP, and we might never know.

In mathematics there are many questions or conjectures or hypotheses that have remained unsettled for a long time. Here are some of the most famous:the *Riemann Hypothesis* (RH), the *Generalized Riemann Hypothesis* (GRH), the *Yang-Mills conjecture* (YM), the *Hodge Conjecture* (HC), and the *Birch-Swinnerton-Dyer* conjecture (BS-D). There are other very interesting open problems such as whether P is equal to PSpace (P=PS), the *Unique Games Conjecture* (UGC), and the *Schwartz-Zippel Lemma* (SZL). In set theory there is the *Generalized Continuum Hypothesis* (GCH). We will also look at some simpler specific mathematical propositions to understand what constitutes evidence for them.

Researchers have examined connections between RH and P=NP. We point out that much of the research about this now concerns the *Generalized Riemann Hypothesis* (GRH). Also there do not seem to be conclusive results on this subject yet. Some articles claim that the GRH implies the negation of P = NP others claim that GRH implies P = NP. How do we make such inferences when we don't know the "truth value" of either proposition? We investigate this based on what it would mean to prove the GRH, and how the evidence from that proof could be transformed into evidence for P = NP.

The point we care about is that we know what it means to *have evidence* for RH or GRH. We do not simply rely on the Boolean truth value of the claims. We rely on the mathematical meaning of these problems that allow us to explore connections among them.

One way to study the relationships among these problems is to see if we can transform evidence for one of them into evidence for a different one. The question $RH \Rightarrow (P = NP)$? and the question $GRH \Rightarrow (P = NP)$? have been well studied and are both quite intriguing. But the truth value analysis is useless. We need to know whether we can transform the evidence for the Riemann Hypothesis or for the Generalized Riemann Hypothesis into evidence for $P = NP$. This is a matter of analyzing ways of using the *evidence* for RH or GRH to understand non deterministic computation better.

Here we will look at the nature of evidence semantics for the propositional calculus that we just studied in detail using Smullyan's Boolean semantics and his tableaux proof rules. We will present propositional logic rules that depend on evidence rather than truth values.

2 Proof Rules Based on Propositional Evidence

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:

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

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

$$x : A, y : B \vdash A \ by \ x$$

The decomposition rules for $A \Rightarrow B$ is more delicate.

$H, f : A \Rightarrow B, H' \ \vdash \ G \ by \ ImpL \ \ on \ f$
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 there is a rule form $apseq(f; slot_a; v.slot_g(v))$ 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.

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

We evaluate the term $apseq(f; a; v.g(v))$ to $g(ap(f; a))$ or more succinctly to $g(f(a))$. This simplification can only be done on the final bottom up pass of creating a closed proof expression, one with no slots.

Minimal Logic

Construction rules

- **And Construction**

  $H \ \vdash \ A\&B \ by \ pair(slot_a; slot_b)$
  $H \ \vdash A \ by \ slot_a$
  $H \ \vdash B \ by \ slot_b$

- **Implication Construction**

  $H \ \vdash \ A \Rightarrow B \ by \ \lambda(x.slot_b(x)) \ new \ x$
  $H, \ x : A \vdash B \ by \ slot_b(x)$

- **Or Construction**

$H \vdash A \vee B$ *by* $inl(slot_l)$
$H \vdash A$ *by* $slot_l$

$H \vdash A \vee B$ *by* $inr(slot_r)$
$H \vdash B$ *by* $slot_r$

Decomposition rules

- **And Decomposition**

  $H, x : A\&B, H' \vdash G$ *by* $spread(x; l, r.slot_g(l, r))$ *new* l, r

  $H, l : A, r : B, H' \vdash G$ *by* $slot_g(l, r)$

- **Implication Decomposition**

  $H, f : A \Rightarrow B, H' \vdash G$ *by* $apseq(f; slot_a; v.slot_g[ap(f; slot_a)/v])$ *new* v[1]
  $H, f : A \Rightarrow B, H' \vdash A$ *by* $slot_a$

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

- **Simple Implication Decomposition**

  $H, f : A \Rightarrow B, H' \vdash G$ *by* $ap(f; slot_a)$ $H, f : A \Rightarrow B, H' \vdash A$ *by* $slot_a$

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

  The value a provided by filling in $slot_a$ is used by function f to compute the value v of type B used to compute the value $g(v)$ that proves the goal. So the goal is computed by $g(f(a))$. This style of proof works top down to build the evidence term from inside out, e.g. we are given f by assumption, to use it we need an element of A, so we try to find a, then apply f, then apply g to $f(a)$.

- **Or Decomposition**

  $H, y : A \vee B, H' \vdash G$ *by* $decide(y; l.leftslot(l); r.rightslot(r))$

  1. $H, l : A, H' \vdash G$ *by* $leftslot(l)$

  2. $H, r : B, H' \vdash G$ *by* $rightslot(r)$

[1] This notation shows that $ap(f; slot_a)$ is substituted for v in $g(v)$. In the CTT logic we stipulate in the rule that $v = ap(f; slot_a)$ *in* B.

- **Hypothesis**

  $H, d : D, H' \vdash \ d \in D \ by \ obj(d)$

  $H, x : A, H' \vdash \ A \ by \ hyp(x)$
  We usually abbreviate the justifications to *by* d and *by* x respectively.

Intuitionistic Rules

- **False Decomposition**

  $H, f : False, H' \vdash G \ by \ any(f)$

  This is the rule that distinguishes intuitionistic from minimal logic, called "ex falso quodlibet". We use the constant $False$ for intuitionistic formulas and $\bot$ for minimal ones to distinguish the logics. In practice, we would use only one constant, say $\bot$, and simply add the above rule with $\bot$ for $False$ to axiomatize iFOL. However, for our results it's especially important to be clear about the difference, so we use both notations.

Classical Rules

- **Law of Excluded Middle (LEM)**

  Define $\sim A$ as $(A \Rightarrow False)$

  $H \vdash (A \vee \sim A) \ by \ \mathbf{magic}(A)$

  Note that this is the only rule that mentions a formula in the rule name.

References