Effective Classical Completeness Theorem

Reed Solomon

September 17, 2008

One of the main results which Wesley Calvert will discuss in October is an effective completeness result for continuous first order logic. To set the stage for this talk, I want to discuss effective completeness in the more familiar context of classical first order logic. Because we are interested in effectiveness issues, we fix a countable first order language $\mathcal{L}$ and assume that we have an effective Gödel numbering for this language. That is, the set of function, relation and constant symbols can be coded by an algorithm that knows the arity of each symbol and we have an effective algorithm (a computer program) that assigns natural numbers to each $\mathcal{L}$ formula in a way that we can calculate $\ulcorner \varphi \wedge \psi \urcorner$ from $\ulcorner \varphi \urcorner$ and $\ulcorner \psi \urcorner$, we can calculate $\ulcorner \varphi[t/x] \urcorner$ from $\ulcorner \varphi \urcorner$ and $\ulcorner t \urcorner$, etc. In general, I will not distinguish between a formula and its coding number. (One usually summarizes the existence of this type of effective coding by saying that the language is *computable*. If you want, just think of working in the language of arithmetic.)

Definition 0.1. A *theory* is a set of sentences. (In particular, a theory does not need to be closed under deductive consequence.) We say that a theory T is *consistent* if there is no sentence φ such that $T \vdash \varphi$ and $T \vdash \neg\varphi$. A theory T is *complete* if for any φ, either $\varphi \in T$ or $\neg\varphi \in T$. (Notice that if T is consistent and complete, then exactly one of φ or $\neg\varphi$ is in T.)

Theorem 0.2 (**Classical Completeness for First Order Logic**)**.** *If T is a consistent first order theory, then T has a model.*

The standard proof of the Completeness Theorem involves a Henkin construction. For our purposes, I want to think of this construction as taking place in two steps, because these steps have different computational properties. In the first step, one extends the consistent theory T to a complete consistent theory T^*. In the second step, one adds an infinite set of new constant symbols $C = \{c_0, c_1, \ldots\}$ to the language $\mathcal{L}$ and uses (equivalence classes) of these terms to build a model of T^*, and hence a model of T since $T \subseteq T^*$. I want to focus on the second step first because this step is effective and gives us the Effective Completeness Theorem.

Definition 0.3. A theory T is *decidable* (or *computable*) if there is an algorithm which determines for any sentence φ whether $\varphi \in T$.

Notice that just because a theory is computable, it does not follow that one can determine whether $T \vdash \varphi$ for an arbitrary sentence φ. For example, if we work in the language of

arithmetic, the set of axioms for Peano Arithmetic (PA) is computable, but there is no algorithm to determine whether PA $\vdash \varphi$ for an arbitrary formula φ. However, we do have this property for complete consistent theories.

Proposition 0.4. *If T is a decidable complete consistent theory, then there is an algorithm that determines whether $T \vdash \varphi$ for any sentence φ.*

This proposition follows trivially from the definitions, because for a complete consistent theory T, $T \vdash \varphi$ if and only if $\varphi \in T$. Therefore, if there is an algorithm to decide whether $\varphi \in T$, the same algorithm decides whether $T \vdash \varphi$.

This completes the description of adding effectiveness issues to the syntactic side of completeness. On the syntactic side, we want to work with decidable complete consistent theories. We also need to see what it means for a model to be effective. Recall that an $\mathcal{L}$-structure $\mathfrak{A}$ is determined by the following pieces of information:

- a domain A,
- a distinguished element $c_i^{\mathfrak{A}} \in A$ for each constant symbol c_i,
- a function $f_i^{\mathfrak{A}} : A^{n_i} \to A$ for each function symbol f_i, and
- a relation $R_i^{\mathfrak{A}} \subseteq A^{n_i}$ for each relation symbol R_i.

We say that an $\mathcal{L}$-structure $\mathfrak{A}$ is *computable* under the following conditions.

- The domain A is a subset of the natural numbers ($A \subseteq \mathbb{N} = \{0, 1, 2, \ldots\}$) for which there is an algorithm to determine whether $n \in A$. (In the case of a (countably) infinite model $\mathfrak{A}$, it never hurts to assume that $A = \mathbb{N}$.)
- There is an algorithm which takes two inputs, a term $t(\overline{x})$ (where $\overline{x}$ denotes the variables in t) and a tuple $\overline{a}$ from the domain, and gives as output the interpretation $t^{\mathfrak{A}}(\overline{a})$.
- There is an algorithm which takes two inputs, an **quantifier free formula** $\varphi(\overline{x})$ (where $\overline{x}$ denotes the free variables in φ) and a tuple $\overline{a}$ from A, and gives as output "Yes" if $\mathfrak{A}$ satisfies $\varphi(\overline{a})$ and "No" if $\mathfrak{A}$ does not satisfy $\varphi(\overline{a})$.

We say that $\mathfrak{A}$ is *decidable* if in addition, there is an algorithm which given **any formula** $\varphi(\overline{x})$ and tuple $\overline{a}$ from A determines whether $\mathfrak{A}$ satisfies $\varphi(\overline{a})$.

Let me give an example. Let $\mathcal{L} = \{0, 1, +, \cdot, \leq\}$ and $\mathfrak{A}$ be the $\mathcal{L}$-structure with domain $A = \mathbb{N}$ and the symbols interpreted in the standard way. (That is, $0^{\mathfrak{A}} = 0$, $+^{\mathfrak{A}} =$ addition, etc.) Then a term $t(\overline{x})$ just expresses some arithmetic combination of the variables. For example, $t(x_0, x_1, x_2)$ might be $(x_0 \cdot x_1) + (x_0 \cdot x_2)$. Obviously, we can compute the value of this term under any assignment of n_0 to x_0, n_1 to x_1 and n_2 to x_2. Furthermore, a quantifier free formula is just some proposition combination of $=$ and $\leq$ between terms. For example, $\varphi(x_0, x_1)$ might be

$$\big((x_0 \cdot x_1) + x_1 = (x_1 \cdot 1) + 0\big) \vee \big((x_1 \leq 1) \to (x_0 \cdot x_0 = x_1)\big)$$

Again, we can obviously determine whether this formula holds under any assignment of natural numbers to the variables. Therefore, this model is computable. On the other hand, the model is not decidable. That is, we cannot determine whether or not any quantified statement holds in this model.

Theorem 0.5 (**Effective Completeness for Classical First Order Logic**). *If T is a decidable complete consistent theory (in a computable language), then T has a decidable model.*

What does a proof of the Effective Completeness Theorem look like? It looks essentially just like the regular Henkin construction, but you need to notice that the assumption that there is an algorithm to decide whether $T \vdash \varphi$ for any sentence φ is enough to give an algorithm for the construction, which in turn lets you decide any sentence is satisfied by the constructed model.

Before leaving this topic, I want to say just a little about finding complete consistent extensions of a consistent theory T. This step of the completeness proof is often referred to as Lindenbaum's Lemma.

Lemma 0.6 (**Lindenbaum's Lemma**). *For any consistent theory T (which for our purposes is in a countable language), there is a complete consistent extension T^* of T.*

How is Lindenbaum's Lemma usually proved? Fix a list of all the sentences $\varphi_0, \varphi_1, \ldots$ in our language $\mathcal{L}$. For any φ_i, we let $\varphi_i^1 = \varphi_i$ and $\varphi_i^0 = \neg\varphi_i$. The proof of Lindenbaum's Lemma is to construct a tree S consisting of finite binary strings σ such that $\sigma \in S$ (with $|\sigma| = n$) if and only if

$$T \cup \{\varphi_0^{\sigma(0)}, \varphi_i^{\sigma(1)}, \ldots, \varphi_{n-1}^{\sigma(n-1)}\}$$

is consistent. One shows that the tree is infinite and hence it must have an infinite path. Finally, one shows that any infinite path in the tree gives a complete consistent extension of T.

How does this construction relate to effectiveness issues? Assume that T is a decidable (but not necessarily complete) theory. The decoding from a path to a complete consistent extension is effective, but the step of determining whether $\sigma \in S$ is not effective. That is, in general, one cannot decide whether T is consistent with an additional finite set of formulas.

In fact, one can show that given any infinite computable tree S, there is a consistent decidable theory T (even a propositional theory) such that the complete extensions of T correspond to the infinite paths in S. In computability theory, we know a lot about the complexity of the set of paths through a computable tree. In particular, there are computable infinite trees of this form that do not have any computable paths. Therefore, there are consistent decidable theories T (even propositional ones) that do not have any decidable complete consistent extensions.