Topics

(1) Undecidable Theories
(2) Gödel’s incompleteness theorem (1st)
(3) Computation in HA
(4) Constructive Issues
  • Markov’s Principle
    – in iFOL
    – in HA
  • Limited computations
  • Next week Constructive Type Theory

1 Undecidable Theories

Recall, a set of numbers $S$ is definable in a theory $T$ iff there is a predicate $F_S$ in $T$ such that

$$
\begin{align*}
  n \in S & \implies \models_T F_S(n) \quad \text{and} \\
  n \notin S & \implies \models_T \sim F_S(n)
\end{align*}
$$

**Lemma:** If $T$ is a consistent extension of $\mathbb{Q}$, then $True_T(x)$ the Gödel numbers of true formulas of $T$ is not definable in $T$.

**Proof:** Suppose $True_T(x)$ is definable, then by the Diagonal Lemma there is a formula $G$ of $T$ such that

$$
\models_T G \iff \sim True_T(\ulcorner G \urcorner)
$$

then we can prove $\models_T G$ and $\models_T \sim G$, so $T$ is not consistent.

**Qed**

**Definition:** A theory $T$ is recursively decidable iff $True_T(x)$ is recursive.

A theory $T$ is decidable iff $\forall x : \mathbb{N}. (True_T(x) \lor \sim True_T(x))$ is constructively true.
**Theorem 1:** No consistent extension of $Q$ is recursively decidable.

**Proof:** If $T$ is a consistent extension of $Q$ and $T$ is recursively decidable, then some recursive function $d$ decides it, say

$$d(x) = 1 \iff TrueT(x)$$

But since the function $d$ is representable in $Q$, there is a predicate $D(x, y)$ that represents it.

Thus $D(x, 1)$ defines $TrueT(x)$ in $T$. But there is no such formula by the above Lemma.

$Qed$

**Corollary:** $iQ$ is not decidable recursively.

**Theorem 2:** (Church) FOL is not decidable.

**Proof:** $Q$ is finitely axiomatizable in FOL, so if $QAx$ is a finite conjunction of the axioms of $Q$, then

$$\vdash Q A \iff \vdash_{FOL} (QAx \Rightarrow A).$$

So if FOL is decidable, then so is $Q$.

$Qed$

Note, using the completeness theorem for iFOL proved by Mark Bickford and me we know

$$\vdash iQ A \iff \vdash_{iFOL} (QAx \Rightarrow A).$$

This is a basis for extending these results above to $iQ$ and its extensions. We will not do this work in the course.

Now we look at computation in HA. Consider why we know that $\forall x, y : \mathbb{N} . (x = y \lor x \neq y)$. This theorem says that we can decide whether $n = m$ for two numbers. How would we do this?

$$eq(n, m) == \text{ if } n = 0 \text{ then ask if } m = 0 \text{ then true else false } \text{ else if } m = 0 \text{ then false else } eq(n – 1, m – 1) /eq(n – 1, m – 1)$$

This is a fairly good algorithm, but it is NOT primitive recursive, so it does not come directly from an inductive proof. How do we prove this in HA?

See lecture for topics (2)-(4).