Lecture 27

Topics

1. Discuss aspirations for planned lessons on type theory.

   We are seeing issues that arise for a *comprehensive theory of computing* that can be a foundation for computer science and mathematics.

   Here is one of them: computation in type theory is key for computer science – but subrecursive or total (Coq PL vs Nuprl PL)?

   Can there be a PL for Herbrand/Gödel - "*all* total computable functions"?

   Some proof assistants don't incorporate computation – HOL.

   Some don't use types – ACL2.

2. Rice's Theorem

   à la Rogers (lecture notes)

   à la CBRFT (proof in *Computational foundations of basic recursive function theory*)

3. Blum Size Thoerem

   Issues- why can't $\mathcal{R}$, the Herbrand-Gödel recursive functions, be a PL like Coq PL?

Rice's Theorem – two proofs, one for $\mathbb{N}$ from Rogers, one from *Computational foundations of basic recursive function theory* for $\overline{T}$.

(i) **Roger's version**

Let $\mathcal{C}$ be a collection of partial recursive functions of one variable, $\varphi_i : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$. Let $\{i : \mathbb{N} \,|\, \varphi_i \in \mathcal{C}\}$ be the indices of $\mathcal{C}$. $\mathcal{C}$ has a recursive characteristic function, say $ch_{\mathcal{C}} : \mathbb{N} \to \mathbb{B}$, if and only if $\mathcal{C}$ is either empty or full (all of $\overline{\mathbb{N}} \to \overline{\mathbb{N}}$).

Proof: If $\mathcal{C}$ is empty, use $\lambda x.false$ (the constant function returning false). If $\mathcal{C}$ is full use $\lambda x.true$, the constant function returning true.

Suppose $ch_{\mathcal{C}}$ is the recursive characteristic function. We show that it must be a constant. So suppose it's non-constant. Thus $ch_{\mathcal{C}}(i)$ is true on some values and false on others. Let φ_d be the everywhere diverging function, $\lambda x.\,\bot$.

We will use $ch_{\mathcal{C}}$ to solve the halting problem does $\varphi_i(i)$ halt, i.e. $\varphi_i(i)\downarrow$? which we previously proved is unsolvable.

First, see where φ_d resides, using $ch_{\mathcal{C}}(d)$. To solve the halting of $\varphi_i(i)$, use the function $\lambda x.\varphi_i(i);\ \varphi_c(x)$ where $\varphi_c(x)$ converges for some x, and $ch_{\mathcal{C}}(c) \neq ch_{\mathcal{C}}(d)$. Recall that $\lambda x.\varphi_i(i);\ \varphi_c(x)$ first computes $\varphi_i(i)$ and then sequences to $\varphi_c(x)$ if $\varphi_i(i) \downarrow$. Call this function $\varphi_{d(i)}$.

We have $\begin{cases} \varphi_i(i) \uparrow \text{ iff } ch_{\mathcal{C}}(d(i)) = 1 \\ \varphi_i(i) \downarrow \text{ iff } ch_{\mathcal{C}}(d(i)) = 0. \end{cases}$

■

(ii) **CBRFT Version**, Theorem 3.10

For all types T, $C_{\overline{T}}$ is decidable iff $C_{\overline{T}}$ is trivial, either empty or full.

Proof:

1. ($\Leftarrow$ Case)

   Use $\lambda x.0$ for empty, $\lambda x.1$ for full. Recall, we use the numeral $\mathbf{2}$ for the Booleans.

2. ($\Rightarrow$ Case)

   If $C_{\overline{T}}$ is decidable, we can use the function $f : \overline{T} \to \mathbf{2}$ to decide whether
   (a) $f(\bot) = 0$ or (b) $f(\bot) = 1$

   In case (a), we show that $C_{\overline{T}}$ is empty.

   In case (b), we show that $C_{\overline{T}}$ is full.

   Case (a). $f(\bot) = 0$.

   Show $\forall t : \overline{T}.f(t) = 0$. Let $t \in \overline{T}$ be arbitrary. We show $f(t) = 0$ by contradiction because equality on $\mathbf{2}$ (Bool) is decidable.

   Assume $f(t) \neq 0$. We show that $div\, k_{\overline{T}}$ is decidable, using the function $h = \lambda x.f((x;t))$ in $\overline{T} \to \mathbf{2}$.

   To show $div\, k_{\overline{T}}$ is decidable, show $h(x) = 0$ iff $x \uparrow$.

   ($\Rightarrow$) $h(x) = 0$ implies $f((x;t)) = 0$, if $x \downarrow$ then $f((x;t)) = f(t) = 0$, but we assumed $f(t) \neq 0$. Hence $x \uparrow$.

   ($\Leftarrow$) If $x \uparrow$ then $f((x;t)) = f(\bot) = 0$ by case (a) assumption.

   So h decides divergence, contrary to Theorem 3.5.

   Case (b). $f(\bot) = 1$

   Exercise for PS4: Finish this case of the proof.
   Optional Exerscie: Can you prove Rice's Theorem for $C_{\overline{T} \to \overline{T}}$ in CBFRT?

Blum Size Theorem

Definition. φ_i is an *acceptable indexing* $\varphi : \mathbb{N} \to (\bar{\mathbb{N}} \to \bar{\mathbb{N}})$ iff it satisfies:

(i) the Universal Machine Theorem, $\exists\, um : (\mathbb{N} \times \mathbb{N} \to \bar{\mathbb{N}})$.

$um(i, x) = \varphi_i(x)$

(ii) the S-m-n theorem, we can find a recursive function s such that $\varphi_i^2(x, y) = \varphi_{s(i,x)}(y)$, for all i, x, y.

Theorem (Rogers): *If φ_i and ψ_i are acceptable indexings, then one is a recursive permutation of the other, e.g. there is a recursive $f : \mathbb{N} \to \mathbb{N}$, $\varphi_{f(i)} = \psi_i$ for all i.*

Definition: a recursive function s is a *size function* iff

(i) There are a finite number of machines of any given size.

(ii) We can compute the size of a machine.

Theorem (Blum Size Theorem). *Let g be any recursive function with unbounded range and let f be any recursive function. We can find indices $i, j \in \mathbb{N}$ such that $\varphi_i = \varphi_{g(i)}$ and $f(|i|) < |g(j)|$, that is, the size of φ_i is considerably smaller than the size of $\varphi_{g(i)}$ although they compute the same function.*

Application – If g enumerates the programs of primitive recursive functions (or Coq PL functions), then a general recursive function φ_i for the same function is considerably smaller.

How could this result possibly be true? The proof is almost magical, using the (strong) *recursion theorem* of Kleene (proof in attached notes from Roger's).

The basic idea is that we can establish a size constraint as part of a general recursive definition for a fixed enumerable sequence of functions known to be total and then take a fixed point.

Optional exercise for PS4: Give an intuitive account of this result as best you can.