

Theory of Computation

Lecture 06

Recursive Function Theory

C SC 573 Theory of Computation

Subject: Recursive Function Theory

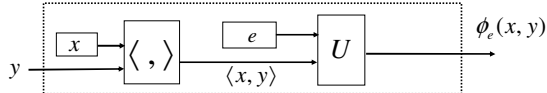
- Or “computable function theory” in text’s parlance
 - “Recursive function” == partial computable function
- Theorems about the functions in some *effective enumeration* of p.c.f.s
 $\phi_0, \phi_1, \phi_2, \dots, \phi_i, \dots$
- Basic results:
 - Enumeration theorem (universality theorem)
 - S-m-n theorem (partial evaluation theorem)
- Advanced results:
 - Recursion theorem
 - Rice’s theorem
- **Enumeration Theorem:** $\exists u. \forall e. \forall x. \phi_u(\langle e, x \rangle) = \phi_e(x)$
 and whenever the right side is \uparrow , so is the left.
Pf: Define $\psi(\langle e, x \rangle) = U(e, x)$. $\exists u. \phi_u(y) = \psi(y)$.
 $\therefore \phi_u(\langle e, x \rangle) = \phi_e(x) \square$

C SC 573 Theory of Computation 2

S-m-n Thm \equiv Partial Evaluation Thm

- Theorem 3.7 ($m=1, n=1$): \exists a fixed t.c.f. $s(\cdot, \cdot)$ such that
 $\forall e, x, y. \phi_{s(e,x)}(y) = \phi_e(x, y)$

Proof: Given fixed e, x one can algorithmically construct TM $S(e, x)$:



Having constructed $S(e, x)$ one can compute its Gödel number $s(e, x)$. So $\phi_{s(e,x)}(y) = \phi_e(x, y) \square$

C SC 573 Theory of Computation 3

S-m-n Thm (cont.)

- Theorem 3.7: $\forall m, n \geq 1 \exists$ t.c.f. $s_n^m(e, \bar{x})$ such that $\forall e, \bar{x}, y \phi_{s_n^m(e, \bar{x})}(y) = \phi_e(x, y)$ where $x = (x_1, \dots, x_m), y = (y_1, \dots, y_n)$
- Corollary 3.3: For every p.c.f. $\lambda x y. \psi(x, y)$ of 2 arguments, there is a t.c.f. f such that $\phi_{f(x)}(y) = \psi(x, y)$

Proof: For some Gödel number $e, \psi = \phi_e$. So by s-m-n, $\psi(x, y) = \phi_e(x, y) = \phi_{s(e,x)}(y)$. Define $f = \lambda x. s(e, x)$ \square

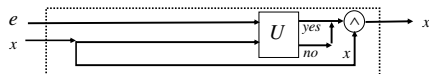
Programming Systems are Cross-Compilable

- Definition: A *programming system* is an effectively enumerable listing ψ_0, ψ_1, \dots that includes all the p.c.f.s of one argument. A programming system is *universal* iff \exists a p.c.f. u such that $u(e, x) = \psi_e(x)$. A programming system is *acceptable* iff it is universal and satisfies the s-m-n theorem.
- Theorem 3.8: For any two acceptable programming systems ψ_0, ψ_1, \dots and ϕ_0, ϕ_1, \dots , there is a t.c.f. t that compiles one to the other, i.e., \exists t.c.f. $t \forall i. \psi_i = \phi_{t(i)}$. *Proof:* Let u be the universal p.c.f. for the system $\{\psi_i\}$. Let $\psi'(\langle e, x \rangle) = u(e, x)$. Since the system $\{\phi_i\}$ contains all p.c.f.s of 1 argument, $\exists k \psi' = \phi_k$. Then $\psi_i(x) = u(i, x) = \psi'(\langle i, x \rangle) = \phi_k(\langle i, x \rangle) = \phi_{s(k,i)}(x)$, the last equality because of the s-m-n theorem. Define the t.c.f. $t(i) = s(k, i)$. \square

S-m-n Applications: More CE Characterizations

- Theorem 3.9: \exists a t.c.f. f such that $range \phi_{f(e)} = dom \phi_e$. *Proof:* Construct a p.c.f. $\psi(e, x) = \begin{cases} x & \text{if } x \in dom \phi_e \\ \uparrow & \text{otherwise} \end{cases}$

by exhibiting a TM for it:



Then for fixed $e, range \lambda x. \psi(e, x) = dom \phi_e$. By Corollary 3.3, \exists t.c.f. f such that $\phi_{f(e)}(x) = \psi(e, x)$. and so $range \phi_{f(e)} = dom \phi_e$. \square
 Homework 3.9: \exists a t.c.f. g such that $dom \phi_{g(e)} = range \phi_e$.

More CE Characterizations (cont.)

- Corollary 3.4: A set S is c.e. iff S is the range of a p.c.f.
Proof: (\Rightarrow). By definition of c.e., S is the range of a t.c.f., unless S is empty; in the latter case S is the range of the TM that never halts for any input.
 (\Leftarrow). Suppose $S = \text{range } \phi_e$. Then Homework 3.9 implies $S = \text{dom } \phi_{g(e)}$. Since S is the domain of a p.c.f., it is c.e. \square
- Homework: Catalog all the characterizations of **CE**.

C SC 573 Theory of Computation 7

The Recursion Theorem

- Consider any sort of (total, algorithmic) program transformation $P \mapsto f(P)$.
 - E.g., $P(x) \mapsto \text{if } x = 0 \text{ then } 1 \text{ else } x \cdot P(x-1)$
 - The transformation f is purely syntactic
- Then there is always a program P^* for which Recursion Theorem

$$\phi_{P^*}(x) = \phi_{f(P^*)}(x)$$
 i.e., P^* is a program whose i/o behavior is identical to that of program $f(P^*)$. P^* is the *fixed point* of f .
- Informally, we are used to writing this fact as:

$$P^*(x) = f(P^*)(x)$$
 - P^* is said to be *recursively defined*
 - E.g., $P^*(x) = \text{if } x = 0 \text{ then } 1 \text{ else } x \cdot P^*(x-1)$

C SC 573 Theory of Computation 8

The Recursion Theorem (cont.)

- Theorem 3.10: \forall t.c.f. $f \exists n \phi_n = \phi_{f(n)}$.
Proof: Define $\theta(e, x) = U(U(e, e), x)$. Then

$$\theta(e, x) = \phi_{\theta_e(e)}(x) \quad (1)$$
 if $\phi_e(e) \downarrow$; \uparrow otherwise. By s-m-n, there is a t.c.f. g such that

$$\phi_{g(e)}(x) = \theta(e, x). \quad (2)$$
 Let f be an arbitrary t.c.f. Then the composition $\lambda x.f(g(x))$ is a t.c.f. It has an index, say k , so that $\phi_k(x) = f(g(x))$. Both sides of this are always defined, so that for any x

$$\phi_{f(g(x))} = \phi_{\phi_k(x)}. \quad (3)$$

C SC 573 Theory of Computation 9

The Recursion Theorem (cont.)

Now set $x = k$ in (3) to get

$$\phi_{f(g(k))} = \phi_{\phi_k(k)}. \quad (4)$$

Since $\phi_k(k) \downarrow$, we may use (1) then (2) to get

$$\forall x. \phi_{f(g(k))}(x) = \phi_{\phi_k(k)}(x) = \theta(k, x) = \phi_{g(k)}(x).$$

Thus $\phi_{f(g(k))} = \phi_{g(k)}$, and setting $n = g(k)$ we have

$$\phi_{f(n)} = \phi_n. \quad \square$$

Corollary 3.5: There is a program that outputs its own code on any input; $\phi_n = \lambda x.n$. \square

Proof: Define $\psi(e, x) = e$. By s-m-n \exists t.c.f. f . $\phi_{f(e)}(x) = \psi(e, x)$ or $\phi_{f(e)}(x) = e$. By recursion thm $\exists n$. $\phi_n(x) = \phi_{f(n)}(x) = n$. \square

C SC 573 Theory of Computation

10

The Recursion Theorem: Applications

- Corollary 3.6: \forall t.c.f. $f \exists n$. $W_n = W_{f(n)}$.
- Example: There is a program n such that $W_n = \{n^2\}$.

Proof: Define $\psi(e, x) = 1$ if $x = e^2$ then 0 else \uparrow .

By s-m-n \exists t.c.f. f . $\phi_{f(e)}(x) = \psi(e, x)$. Then

$$\text{dom } \phi_{f(e)} = W_{f(e)} = \{e^2\}. \text{ By recursion theorem}$$

$$\exists n. W_{f(n)} = W_n = \{n^2\}. \quad \square$$

C SC 573 Theory of Computation

11

The Recursion Theorem: Applications

- Theorem (Undecidable Sentences): Given any consistent formal system F (such as set theory, or Peano arithmetic), there is a TM M such that there is *no* proof in F that M started on any particular input halts, *and* no proof that it does not halt.

Proof: Construct M , a TM computing a 2-input function

$$g(e, x) = \begin{cases} 1 & \text{if there is a proof in } F \text{ that } \phi_e(x) \uparrow \\ \uparrow & \text{otherwise} \end{cases}$$

M enumerates proofs in F in some order, printing 1 if a proof that the e th TM does not halt on x is found. Further, we can construct M so that if $g(e, x) = 1$, then M halts, otherwise M does not halt. By Cor. 3.3, there is a t.c.f. f such that

$$\phi_{f(e)}(x) = g(e, x)$$

C SC 573 Theory of Computation

12

The Recursion Theorem: Applications

By the Recursion Theorem, we may effectively construct an integer n such that

$$\phi_n(x) = \phi_{f(n)}(x) = g(n, x).$$

But $g(n, x) = 1 \Leftrightarrow \exists$ a proof in F that $\phi_n(x) \uparrow$

If F is consistent, it would follow from the existence of this proof that

$$g(n, x) = 1 \Leftrightarrow \phi_n(x) = g(n, x) \uparrow$$

This contradiction establishes that there can be no proof in F that the TM with Gödel number n halts (or does not halt) on any particular input. \square

C SC 573 Theory of Computation

13
