Computer Science 513

Rice's Theorem and the Recursion Theorem

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #12

Goals for Today

Goals for Today: Presentation of two additional significant results:

- Rice's Theorem
- The Recursion Theorem

We will see that these can be used to develop simpler and shorter proofs that sets are not computable, or not computably enumerable, than would otherwise be possible.

Parameter Theorem

Once again, recall the ***Parameter Theorem:*** For all positive integers n and m there exists a primitive recursive function

$$S_m^n : \mathbb{N}^{n+1} \to \mathbb{N}$$

such that, for all $x_1, x_2, \ldots, x_m, u_1, u_2, \ldots, u_n, y \in \mathbb{N}$,

$$\Phi^{(m+n)}(x_1, x_2, \ldots, x_m, u_1, u_2, \ldots, u_n, y) = \Phi^{(m)}(x_1, x_2, \ldots, x_m, S_m^n(u_1, u_2, \ldots, u_n, y)).$$

This theorem will be used — repeatedly — to establish additional results today.

Rice's Theorem

Let Γ be some collection of computable partial (or total) functions of one variable.

- The ***index set*** of Γ is the set

$$\begin{aligned} R_\Gamma &= \{n \in \mathbb{N} \mid \Phi_n \in \Gamma\} \\ &= \{n \in \mathbb{N} \mid \text{the program } \mathcal{P} \text{ such that } \#(\mathcal{P}) = n \\ &\qquad\qquad \text{computes a function in } \Gamma\ \}. \end{aligned}$$

- Recall that R_Γ is defined to be a ***computable set*** if the characteristic function $p_{R_\Gamma} : \mathbb{N} \to \{0, 1\}$ such that, for all $n \in \mathbb{N}$,

$$p_{R_\Gamma}(n) = \begin{cases} 1 & \text{if } \Phi_n^{(1)} \in \Gamma \\ 0 & \text{if } \Phi_n^{(1)} \notin \Gamma \end{cases}$$

  is a Turing-computable predicate.

Rice's Theorem

Rice's Theorem: Let Γ be a collection of Turing-computable partial (or total) functions of one variable. Suppose that there exist Turing-computable functions $f, g : \mathbb{N} \to \mathbb{N}$ such that $f \in \Gamma$ and $g \notin \Gamma$ — so that $R_\Gamma \neq \emptyset$ and $R_\Gamma \neq \mathbb{N}$.

Then the set R_Γ is not computable.

Proof:

- Note that R_Γ is computable if and only if $R_{\overline{\Gamma}}$ is computable.
- We may there assume, without loss of generality, that $\varphi \notin \Gamma$, where $\varphi : \mathbb{N} \to \mathbb{N}$ is the function that is ***undefined*** everywhere.

Rice's Theorem

- Let q be the encoding of the $\mathcal{S}$-program (obtained, after macro expansion, from)

$$Z_1 \leftarrow \Phi^{(1)}(X_2, X_2)$$
$$Y \leftarrow f(X_1)$$

- Recall that, for every integer $i \in \mathbb{N}$, $S_1^1(i, q)$ is the encoding of a program computing the same function as

$$X_2 \leftarrow i$$
$$Z_1 \leftarrow \Phi^{(1)}(X_2, X_2)$$
$$Y \leftarrow f(X_1)$$

Rice's Theorem

- It follows by the ***Parameter Theorem*** that the function $S_1^1 : \mathbb{N}^2 \to \mathbb{N}$ is primitive recursive.
- With that noted, let $h : \mathbb{N} \to \mathbb{N}$ be the function such that, for all $n \in \mathbb{N}$,

$$h(n) = S_1^1(n, q).$$

  Then h is primitive recursive as well, since q is a constant.
- It follows that h is a total computable function.
- Recall that the set

$$K = \{n \in \mathbb{N} \mid n \in W_n\}$$

  is computably enumerable but not computable.

Rice's Theorem

- Notice that, for all $n \in \mathbb{N}$,

$$
\begin{aligned}
n \in K &\Rightarrow \Phi^{(1)}(n, n) \text{ is defined} && \text{(by the definition of } K\text{)} \\
&\Rightarrow \Phi^{(1)}(x, S_1^1(n, q)) = f(x) \text{ for all } x \in \mathbb{N} \\
&\qquad\qquad \text{(by inspection of the above programs)} \\
&\Rightarrow \Phi^{(1)}(x, h(n)) = f(x) \text{ for all } x \in \mathbb{N} \\
&\qquad\qquad\qquad\qquad \text{(by the definition of } h\text{)} \\
&\Rightarrow \Phi^{(1)}_{h(n)} \in \Gamma && \text{(since } \Phi_{h(n)} = f \in \Gamma\text{)} \\
&\Rightarrow h(n) \in R_\Gamma .
\end{aligned}
$$

Rice's Theorem

- On the other hand, for all $n \in \mathbb{N}$,

$$
\begin{aligned}
n \notin K &\Rightarrow \Phi^{(1)}(n,n) \text{ is undefined} \quad \text{(by the definition of } K\text{)} \\
&\Rightarrow \Phi^{(1)}(x, S_1^1(n,q)) \text{ is undefined for all } x \in \mathbb{N} \\
&\qquad\qquad \text{(by inspection of the above programs)} \\
&\Rightarrow \Phi^{(1)}(x, h(n)) \text{ is undefined for all } x \in \mathbb{N} \\
&\qquad\qquad \text{(by the definition of } h\text{)} \\
&\Rightarrow \Phi^{(1)}_{h(n)} = \varphi \notin \Gamma \\
&\Rightarrow h(n) \notin R_\Gamma .
\end{aligned}
$$

- Since h is a total computable function it now follows that $K \preceq_m R_\Gamma$ and, since K is not computable, it follows that R_Γ is not computable, either. □

Rice's Theorem: How To Use It

In order to ***apply*** Rice's Theorem, to prove that R_Γ is not computable, for some collection Γ of Turing-computable partial (or total) functions of one variable,

1. Describe a partial or total function f that is provably Turing-computable and belongs to Γ.
2. Describe another partial or total function g that is provably Turing computable and *does not* belong to Γ.
3. Conclude that R_Γ is not computable.

Rice's Theorem: Example Application

Consider the set

$$\begin{aligned}\text{Empty} &= \{n \in \mathbb{N} \mid W_n = \emptyset\} \\ &= \{n \in \mathbb{N} \mid \text{ the } \mathcal{S}\text{-program with encoding } n \\ &\qquad\qquad \text{does not halt on any (single) input.}\}\end{aligned}$$

A ***many-one reduction*** $\overline{K} \preceq_m$ Empty was described in the previous lecture — and it was noted that this implies that the set "Empty" is not computably enumerable. It follows, from this, that the set "Empty"' is not computable.

Rice's Theorem can be used to give a shorter, and simpler, proof of this.

Rice's Theorem: Example Application

Notice that Empty $= R_\Gamma$ where Γ is a set of Turing-computable partial or total functions. In particular, Empty $= R_\Gamma$ when

$$\Gamma = \{\varphi\}$$

for $\varphi : \mathbb{N} \to \mathbb{N}$ such that $\varphi(n)$ is undefined for all $n \in \mathbb{N}$.

- Thus Rice's Theorem might be applicable.

Rice's Theorem: Example Application

1. Let $f = \varphi : \mathbb{N} \to \mathbb{N}$ for the function given above.

   This function is certainly Turing-computable. Indeed, it is computed by the $\mathcal{S}$-program

   $$\begin{array}{ll} & Z_1 \leftarrow Z_1 + 1 \\ [A_1] & \text{IF } Z_1 \neq 0 \text{ GOTO } A_1 \end{array}$$

   Thus $f = \varphi$ is a Turing-computable partial function such that $f \in \Gamma$.

Rice's Theorem: Example Application

2. Let $g : \mathbb{N} \to \mathbb{N}$ such that $g(n) = 0$ for all $n \in \mathbb{N}$.

   Then g is certainly Turing-computable — it is computed by the empty program — and $g \notin \Gamma$.

3. It follows by Rice's Theorem that R_Γ is not a computable set. That is (since Empty $= R_\Gamma$), the set Empty is not computable.

Recursion Theorems

The ***Recursion Theorems*** are a collection of related technical results that involve a form of "self reference".

- Their statements can be confusing, and it might not be clear why they are interesting.
- Indeed, they are interesting mainly as "technical lemmas" which can be used to prove other more understandable (and, possibly, interesting) results.
- The version of the "Recursion Theorem" that is stated and proved here is found in Davis, Sigal and Weyuker's text, *Computability, Complexity and Languages: Fundamentals of Theoretical Computer Science.*

Recursion Theorem

Recursion Theorem: Let $g : \mathbb{N}^{m+1} \to \mathbb{N}$ be a Turing-computable partial (or total) function. Then there exists a number $e \in \mathbb{N}$ such that, for all $x_1, x_2, \ldots, x_n \in \mathbb{N}$,

$$\Phi_e^{(m)}(x_1, x_2, \ldots, x_m) = g(e, x_1, x_2, \ldots, x_m)$$

The proof of this result makes a clever use of the ***Parameter Theorem*** and is included in the supplemental material for this lecture.

Recursion Theorem: Application

Once again, consider the total ***halting*** predicate $p_H : \mathbb{N}^2 \to \{0, 1\}$ such that, for all $x, y \in \mathbb{N}$,

$$p_H(x, y) = 1 \iff \Phi_x^{(1)}(y) \text{ is defined.}$$

- It has already been shown that this predicate is not Turing-computable.
- A shorter and simpler proof of this can be obtained using the Recursion Theorem.

Recursion Theorem: Application

Claim: The halting predicate, p_H, is not Turing-computable.

Proof: By contradiction. Suppose, instead, that the predicate p_H is Turing-computable.

- Then so is the partial function Loop : $\mathbb{N}^2 \to \mathbb{N}$ that is computed by the $\mathcal{S}$-program obtained from the following (after macro expansion):

$$\begin{array}{ll} & Z_1 \leftarrow p_H(X_1, X_2) \\ [A_1] & \text{IF } Z_1 \neq 0 \text{ GOTO } A_1 \end{array}$$

- This is the partial function such that, for all $x, y \in \mathbb{N}$,

$$\text{Loop}(x, y) = \begin{cases} 0 & \text{if } \Phi_x^{(1)}(y) \text{ is undefined,} \\ \uparrow & \text{otherwise.} \end{cases}$$

The Recursion Theorem: Application

- It now follows by the Recursion Theorem that there exists a number $e \in \mathbb{N}$ such that, for all $y \in \mathbb{N}$,

$$\Phi_e^{(1)}(y) = \text{Loop}(e, y).$$

- However, it now follows that

$$\begin{aligned} \Phi_e^{(1)}(y) \text{ is defined} &\iff \text{Loop}(e, y) \text{ is defined} \\ &\iff \Phi_e^{(1)}(y) \text{ is undefined} \\ &\quad \text{(by the definition of Loop).} \end{aligned}$$

- This is certainly impossible — providing *another* proof that the predicate p_H is not Turing-computable. □

The Recursion Theorem: Application

Claim: There exists a number $e \in \mathbb{N}$ such that

$$\Phi_e^{(1)}(y) = e$$

for all $y \in \mathbb{N}$.

Proof:

- Let $g : \mathbb{N}^2 \to \mathbb{N}$ such that, for all $x, y \in \mathbb{N}$,

  $$g(x, y) = x = u_1^2(x).$$

  Since g is primitive recursive it is certainly Turing-computable.

- It follows by the Recursion Theorem that there is a number e such that

  $$\Phi_e^{(1)}(y) = g(e, y) = e$$

  for all $y \in \mathbb{N}$, as claimed. □

The Recursion Theorem: Application

- ***Note*** that the previous result is not easy to prove *without* the Recursion Theorem: The $\mathcal{S}$-program $\mathcal{P}$ consisting of e copies of the statement

$$Y \leftarrow Y + 1$$

also computes this function — but the encoding, $\#(\mathcal{P})$, of this program is ***exponential*** in e instead of equal to it!

Fixed Point Theorem

The following is another useful technical result — which is easily proved using the Recursion Theorem.

Fixed Point Theorem: Let $f : \mathbb{N} \to \mathbb{N}$ be a Turing-computable total function. Then there exists a number $e \in \mathbb{N}$ such that

$$\Phi^{(1)}_{f(e)}(x) = \Phi^{(1)}_{e}(x)$$

for all $x \in \mathbb{N}$ — that is, the program with encoding $f(e)$ computes the same (partial or total) function as the program with encoding e.

Fixed Point Theorem

Proof:

- Consider the function $g : \mathbb{N}^2 \to \mathbb{N}$ such that, for all $x, y \in \mathbb{N}$,

$$g(x, y) = \Phi^{(1)}_{f(x)}(y).$$

  Since f is a Turing-computable total function this is also a Turing-computable (partial or total) function.

- It follows by the Recursion Theorem that there is a number $e \in \mathbb{N}$ such that

$$\Phi^{(1)}_{e}(y) = g(e, y) = \Phi^{(1)}_{f(e)}(y)$$

  for all $y \in \mathbb{N}$ — as claimed. □

Fixed Point Theorem: Application

Let $p : \mathbb{N} \to \{0, 1\}$ be a Turing-computable predicate, and let $g : \mathbb{N} \to \mathbb{N}$ be a Turing-computable total function.

- Let

$$\text{while} : \mathbb{N} \to \mathbb{N}$$

be the partial function such that, for all $n \in \mathbb{N}$, while(n) is the encoding of the $\mathcal{S}$-program obtained from the following, after macro expansion:

$$\begin{aligned}
&X_2 \leftarrow n \\
&Y \leftarrow X_1 \\
&Z_1 \leftarrow \neg p(Y) \\
&\text{IF } Z_1 \neq 0 \text{ GOT } E_1 \\
&Y \leftarrow \Phi^{(1)}_{X_2}(g(Y))
\end{aligned}$$

Fixed Point Theorem: Application

- Notice that — after macro expansion — this program would begin with an $\mathcal{S}$-program consisting of n copies of the statement

$$X_2 \leftarrow X_2 + 1$$

  — a $\mathcal{S}$-program whose encoding can be shown to be a primitive recursive function of n — followed by some fixed $\mathcal{S}$-program that does not depend on n at all.

- This can be used to prove that the function $\text{while} : \mathbb{N} \to \mathbb{N}$ is primitive recursive — so that it is certainly a Turing-computable total function.

Fixed Point Theorem: Application

- It follows, by the Fixed Point Theorem, there this a number $e \in \mathbb{N}$ such that

$$\Phi_e^{(1)}(x) = \Phi_{\text{while}(e)}^{(1)}(x)$$

for all $x \in \mathbb{N}$.

- Note that — by the construction of the above program —

$$\Phi_e^{(1)}(x) = \Phi_{\text{while}(e)}^{(1)}(x) = \begin{cases} x & \text{if } \neg p(x), \\ \Phi_e^{(1)}(g(x)) & \text{otherwise,} \end{cases}$$

because X_2 is set to have value e at the first step of the program while(e).

Fixed Point Theorem: Application

- Thus

$$\Phi_e^{(1)}(x) = \begin{cases} x & \text{if } \neg p(x) \\ g(x) & \text{if } p(x) \wedge \neg p(g(x)) \\ g(g(x)) & \text{if } p(x) \wedge p(g(x)) \wedge \neg p(p(g(x)) \end{cases}$$

  and so on.

- In other words, this performs the same computation as the program with a `while` loop

$$\begin{array}{l} Y \leftarrow X \\ \texttt{while } (p(Y)) \{ \\ \quad Y \leftarrow g(Y) \\ \} \end{array}$$

Another Proof of Rice's Theorem

Recall *Rice's Theorem:*

Rice's Theorem: Let Γ be a collection of Turing-computable partial (or total) functions of one variable. Suppose that there exist Turing-computable partial or total functions $f, g : \mathbb{N} \to \mathbb{N}$ such that $f \in \Gamma$ and $g \notin \Gamma$ — so that $R_\Gamma \neq \emptyset$ and $R_\Gamma \neq \mathbb{N}$.

Then the set R_Γ is not computable.

The ***Recursion Theorem*** can be used to obtain another proof of this result.

Another Proof of Rice's Theorem

Proof: By contradiction. Suppose that R_Γ is computable.

- Let $p_\Gamma : \mathbb{N} \to \{0, 1\}$ be the characteristic function of R_Γ, that is, the function such that for all $x \in \mathbb{N}$,

$$p_\Gamma(x) = \begin{cases} 1 & \text{if } x \in R_\Gamma, \\ 0 & \text{otherwise.} \end{cases}$$

- Let $h : \mathbb{N}^2 \to \mathbb{N}$ such that, for all $x, y \in \mathbb{N}$,

$$h(x, y) = \begin{cases} g(y) & \text{if } x \in R_\Gamma, \\ f(y) & \text{if } x \notin R_\Gamma. \end{cases}$$

Then, since p_Γ is a computable (total) predicate, and f and g are computable (partial or total) functions, h is a computable (partial or total) function as well.

Another Proof of Rice's Theorem

- It follows by the Recursion Theorem that there is a number $e \in \mathbb{N}$ such that, for all $y \in \mathbb{N}$,

$$\Phi_e^{(1)}(y) = h(e, y) = \begin{cases} g(y) & \text{if } \Phi_e^{(1)} \in \Gamma, \\ f(y) & \text{if } \Phi_e^{(1)} \notin \Gamma. \end{cases}$$

- It now follows that, since either $\Phi_e^{(1)} = f \in \Gamma$ or $\Phi_e^{(1)} = g \notin \Gamma$,

$$\begin{aligned} \Phi_e^{(1)} \in \Gamma &\iff \Phi_e^{(1)} = g && \text{(by the above)} \\ &\iff \Phi_e^{(1)} \notin \Gamma && \text{(since } g \notin \Gamma\text{)} \end{aligned}$$

and this is certainly impossible. This ***contradiction*** provides an alternative proof of Rice's Theorem, that is, another proof that the set R_Γ is not computable. □

Expectations for Students

- Please make sure that you understand both Rice's Theorem and the Recursion Theorem.
- It is possible (but not guaranteed) that you will be asked to use Rice's Theorem to prove that a set is not recursive on a future assignment or test.
- ***Given an adequate hint*** (i.e., enough information for you to discover the function g mentioned in the statement of the theorem) you might also be asked to apply the Recursion Theorem in order to solve a problem.

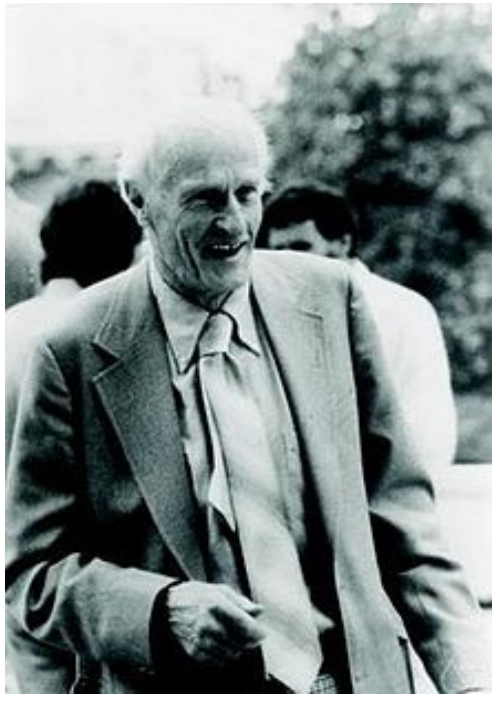
Who Discovered Rice's Theorem?

Rice's Theorem is also called the ***Rice-Myhill-Shapiro Theorem*** — and, indeed, all three of the following mathematicians discovered versions of this result:

- Henry Gordon Rice — who proved the result in his Ph.D. thesis in 1951.
- John R. Myhill Sr. — a British mathematician.
- Norman Z. Shapiro — an American mathematician.

Photographs of these mathematicians do not seem to be easy to find!

Who Discovered the Recursion Theorem?



Stephen Cole Kleene — the discoverer of the ***Normal Form Theorem*** and the ***Parameter Theorem*** — also discovered the ***Recursion Theorem***.