1.1. The Kleene Recursion Theorem

This brief note covers Kleene's recursion Theorem and a few applications.

1.1.1 Theorem. (Kleene's recursion theorem) *If $\lambda z\vec{x}.f(z,\vec{x}_n) \in \mathcal{P}$, then for some e,*

$$\phi_e^{(n)}(\vec{x}_n) = f(e,\vec{x}_n) \text{ for all } \vec{x}_n$$

Proof. Let $\phi_a^{(n+1)} = \lambda z\vec{x}_n.f(S_1^n(z,z),\vec{x}_n)$. Then

$$\begin{aligned} f(S_1^n(a,a),\vec{x}_n) &= \phi_a^{(n+1)}(a,\vec{x}_n) \\ &= \phi_{S_1^n(a,a)}^{(n)}(\vec{x}_n) && \text{by S-m-n thm} \end{aligned}$$

Take $e = S_1^n(a,a)$. □

1.1.2 Corollary. *Let $f \in \mathcal{R}$. Then there is an $e \in \mathbb{N}$ such that $\phi_e = \phi_{f(e)}$.*

Proof. $\lambda xy.\phi_{f(y)}(x) \in \mathcal{P}$ by the Normal Form Theorem. By 1.1.1, there is an $e \in \mathbb{N}$ such that $\phi_e(x) = \phi_{f(e)}(x)$, for all x.

For short, $\phi_e = \phi_{f(e)}$. □

1.1.1. Two Applications of the Recursion Theorem

1.1.3 Definition. Recall that a *complete index set* is a set $A = \{x : \phi_x \in \mathcal{C}\}$ for some $\mathcal{C} \subseteq \mathcal{P}$.

We call A *trivial* iff $A = \emptyset$ or $A = \mathbb{N}$ (correspondingly, $\mathcal{C} = \emptyset$ or $\mathcal{C} = \mathcal{P}$). Otherwise it is called *non trivial.* □

1.1.4 Theorem. (Rice) *A complete index set is recursive iff it is trivial.*

Thus, "algorithmically" we can only "decide" trivial properties of "programs".

Proof. (The idea of this proof is attributed in [Rog67] to G.C. Wolpin.)

if-part. Immediate, since $\chi_\emptyset = \lambda x.1$ and $\chi_\mathbb{N} = \lambda x.0$.

only if-part. By contradiction, suppose that $A = \{x : \phi_x \in \mathcal{C}\}$ is *non trivial*, yet $A \in \mathcal{R}_*$. So, let $a \in A$ and $b \notin A$. Define f by

$$f(x) = \begin{cases} b & \textbf{if } x \in A \\ a & \textbf{if } x \notin A \end{cases}$$

Clearly,

$$x \in A \text{ iff } f(x) \notin A, \text{ for all } x \tag{1}$$

By the Corollary, there is an e such that $\phi_e = \phi_{f(e)}$.

Thus, $e \in A$ iff $\phi_e \in \mathcal{C}$ iff $\phi_{f(e)} \in \mathcal{C}$ iff $f(e) \in A$, contradicting (1). □

The second application is about self-referential (recursive) definitions of functions F such as the one below

$$F(\vec{x}_n) = f\Big(\dots F\Big(\dots F(\dots)\dots\Big)\dots F\Big(\dots F\big(\dots F(\dots)\dots\big)\dots\Big)\dots\Big) \tag{1}$$

where nesting of occurrences of F can be anything.

We are interested in just those cases that, as we say, the right hand side of (1) —as a function of $\vec{x}_n$— is partial recursive in F.

1.1.5 Definition. We say that a function is *partial recursive in* F iff it is in the closure of $I \cup \{F\}$ under composition, primitive recursion and (μy). Here I denoted by "I" the standard initial functions of $\mathcal{P}$.

For short, a function is partial recursive in F iff is obtained by a finite number of partial recursive operations *using as initial functions* F *and those in* I. □

1.1.6 Remark. It follows from 1.1.5 that if $F \in \mathcal{P}$, then a function that is partial recursive in F is just partial recursive.

In particular, if we replace F throughout the right hand side of (1) by a partial recursive function $\phi_e^{(n)}$ of the same arity n as F, then we end up with a partial recursive function. □

In (1) F acts as a "function variable" to solve for. A solution h for F is a specific function that makes (1) true for all $\vec{x}_n$ if we replace all occurrences of F by h.

We show that if the right hand side of (1) is partial recursive in F, then (1) always has a partial recursive solution for F. That is,

$$(\exists e)\Big(\text{if we replace } F \text{ in (1) by } \lambda\vec{x}_n.\phi_e^{(n)}(\vec{x}_n)\text{, then the resulting relation is true for all } \vec{x}_n\Big) \tag{2}$$

Indeed, the function $\lambda z\vec{x}_n.G(z,\vec{x}_n)$ given below is partial recursive by 1.1.6.

$$G(z,\vec{x}_n) = f\Big(\dots \phi_z^{(n)}\Big(\dots \phi_z^{(n)}(\dots)\dots\Big)\dots \phi_z^{(n)}\Big(\dots \phi_z^{(n)}\big(\dots \phi_z^{(n)}(\dots)\dots\big)\dots\Big)\dots\Big) \tag{3}$$

By the recursion theorem there is an e such that

$$G(e,\vec{x}_n) = \phi_e^{(n)}(\vec{x}_n)\text{, for all } \vec{x}_n$$

Thus, (3) yields

$$\phi_e^{(n)}(\vec{x}_n) = G(e,\vec{x}_n) = f\Big(\dots \phi_e^{(n)}\Big(\dots \phi_e^{(n)}(\dots)\dots\Big)\dots \phi_e^{(n)}\Big(\dots \phi_e^{(n)}\big(\dots \phi_e^{(n)}(\dots)\dots\big)\dots\Big)\dots\Big) \tag{4}$$

That is, setting the "function variable F" equal to $\phi_e^{(n)}$ we have solved (1), and with a $\mathcal{P}$-solution at that!

1.1.7 Example. Here is a second solution to the question "$\lambda nx.A_n(x) \in \mathcal{R}$?".

$A_n(x)$ is given by

$$A_n(x) = \begin{cases} x+2 & \text{if } n=0 \\ 2 & \text{else if } x=0 \\ A_{n \dot{-} 1}\big(A_n(x \dot{-} 1)\big) & \text{otherwise} \end{cases}$$

We re-write the above using F as a function variable and setting $F(n,x) = A_n(x)$.

Thus, F is "given" by

$$F(n,x) = \begin{cases} x+2 & \text{if } n=0 \\ 2 & \text{else if } x=0 \\ F\big(n \dot{-} 1, F(n, x \dot{-} 1)\big) & \text{otherwise} \end{cases} \tag{5}$$

(5) has the form (1) and all assumptions are met. Thus, for some e, $F = \phi_e^{(2)}$ works. But is this $\phi_e^{(2)}$ the same as $A_n(x)$? Yes, provided (5) has a unique solution! That (5) indeed does have a unique total solution is an easy (double) induction exercise that shows $A_n(x) = B_n(x)$ for all n, x if

$$B_n(x) = \begin{cases} x+2 & \text{if } n=0 \\ 2 & \text{else if } x=0 \\ B_{n \dot{-} 1}\big(B_n(x \dot{-} 1)\big) & \text{otherwise} \end{cases} \tag{5$'$}$$

Indeed we start the proof of $(\forall n)(\forall x)A_n(x) = B_n(x)$ by induction on n:

$n = 0$: $A_0(x) = x + 2 = B_0(x)$.
I.H. fix n and assume for all x: $A_n(x) = B_n(x)$.
I.S. for $n + 1$: Prove for all x and the fixed n: $A_{n+1}(x) = B_{n+1}(x)$.

Do the latter by induction on x:

$x = 0$: $A_{n+1}(0) = 2 = B_{n+1}(0)$.
I.H. fix x and assume for the n above: $A_{n+1}(x) = B_{n+1}(x)$.
I.S. for $x + 1$: For the fixed n and x we provide the last proof step:

$$A_{n+1}(x+1) = A_n(A_{n+1}(x)) \overset{\text{I.H. on } n}{=} B_n(A_{n+1}(x)) \overset{\text{I.H. on } x}{=} B_n(B_{n+1}(x)) = B_{n+1}(x+1) \qquad \Box$$

Bibliography

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.