THE KLEENE-HERBRAND-GODEL COMPUTABLE FUNCTIONS

In this section, we present another model of computation due to Kleene, Herbrand and Godel. In this model, a function is defined by a finite set of purely recursive equations. By allowing arbitrary recursion, it might seem at first glance that we obtain a larger class of functions than the partial recursive functions. However, this is not the case because the process of computing a function defined by a K-H-G (Kleene-Herbrand-Godel) system of equations can be encoded by primitive recursive functions, the only step requiring partial recursive functions being the "guess" that the computation halts. The reader will note that this time, the functions defined in this model are obviously "recursive," which is not so obvious in the RAM model, the Turing machine model and the definition in terms of closure operations where the only type of recursion allowed is primitive recursion. One might say that the K-H-G model justifies the name given to this class of functions (partial recursive functions). Historically, variants of the K-H-G model were given first by Herbrand (around 1920) and Godel (around 1935) and the version presented here is due essentially to Kleene: Introduction to Meta-Mathematics, by S. C. Kleene, North-Holland, Amsterdam, first printing 1952.

To get the flavor of the K-H-G model, let us look at the following example:

Example 1:

$$F(0,y) = y + 1$$

$$F(x + 1,0) = F(x,1)$$

$$F(x + 1, y + 1) = F(x,F(x + 1,y))$$

The above equations define a function called Ackermann's function. The remarkable fact about this function is that it eventually grows faster than any primitive recursive function, and therefore is not a primitive recursive function. It can be shown that,

$$F(0,x) = x + 1$$

$$F(1,x) = x + 2$$

$$F(2,x) = 2x + 3$$

$$F(3,x) = 2^{x + 3} - 3$$

and $F(4,x) = 2^{2^{2^{\cdot^{\cdot^{\cdot^{2^{16}}}}}}} - 3$ (a stack of x exponentials)

where the stack of exponentials has height x . For instance,

$$F(4,1) = 2^{16} - 3,\ F(4,2) = 2^{2^{16}} - 3 .$$

Actually, it is not even clear that F is recursive, but we will see that this results from the following developments.

Let us give another example and see how computations proceed.

Consider the functions H and F defined by the following system of equations:

Example 2:

$$S \begin{cases} H(x,y) = 7 \\ F(0) = 4 \\ F(x+1) = H(x,F(x)) \end{cases}$$

We claim that S computes the constant function $H(x,y) = 7$ and the function $F(x) =$ if $x = 0$ then 4 else 7.

We now describe a sequence of very elementary steps which constitute a computation of F(2). At each step, we are either allowed to substitute an integer for all occurrences of a variable in an equation, or to substitute the value of an expression where all the arguments have been evaluated. Let us number the equations as follows:

1. $H(x,y) = 7$
2. $F(0) = 4$
3. $F(x+1) = H(x,F(x))$

Computation:

1. Substitute 0 for x in (3) obtaining:

   $F(1) = H(0,F(0))$

2. Replace F(0) by 4 in right-hand side of line 1 obtaining:

   $F(1) = H(0,4)$

3. Substitute 0 for x and 4 for y in (1) obtaining:

   $H(0,4) = 7$

4. Replace right-hand side of line 2 by 7 obtaining:

   $F(1) = 7$

5. Substitute 1 for x in (3) obtaining:

   $F(2) = H(1,F(1))$

6. Replace F(1) by 7 in right-hand side of line 5 obtaining:

   $F(2) = H(1,7)$

7. Substitute 1 for x and 7 for y in (1) obtaining:

   $H(1,7) = 7$

8. Replace H(1,7) by 7 in line 6 obtaining:

   $F(2) = 7$

Line 8 is the desired equation. We say that equation F(2) = 7 has been deduced from the System S. Clearly, each step of the computation, also called deduction, is highly mechanical: either an integer is substituted for all occurrences of a variable in an equation, or a subterm whose arguments are already evaluated is replaced by its value in the right-hand side of an equation.

Our next task is to define precisely the K-H-G model. The main definition is that of a system of equations, and for that, we need some auxiliary definitions given below. This section could have been written for functions with string arguments, however it will be simpler especially in the coding scheme to use functions of the natural numbers. From results in Chapter 2, we see that there is actually no loss of generality.

Variables:

The formal system KHG uses a countable set of individual variables

$\mathcal{V} = \{ x_0, x_1, x_2, \ldots \}$ and

a countable set of function symbols

$\mathcal{F} = \{ F_0^1, F_1^1, \ldots, F_0^2, F_1^2, \ldots, F_0^m, F_1^m, \ldots \}$

where each F_i^m is the i-th function symbol of arity m (taking m arguments).

Terms:

1. Each variable x_i is a term
2. 0 is a term
3. If t is a term, so is S(t), where S is the successor function.
4. If $t_1, \ldots, t_m$ are terms, so is $F_i^m(t_1, \ldots, t_m)$.

A term of the form $S(S(\ldots S(0)\ldots))$ with n occurrences of S is called a numeral and represents the number n. Such a numeral is abbreviated as $\bar{n}$ for convenience.

Equations:

An equation is an expression of the form $t_1 = t_2$ where t_1 and t_2 are terms.

Substitution:

Given a term t, having occurrences of a variable x, we denote as $t_1[t_2 / x]$ the term obtained as the result of substituting t_2 for all occurrences of x in t_1. Substitution can be defined inductively as follows:

1. If t_1 is a variable $y \neq x$, then $t_1[t_2 / x] = y$
2. If $t_1 = x$, then $t_1[t_2 / x] = t_2$
3. If t_1 is the constant 0 then $t_1[t_2 / x] = 0$
4. If $t_1 = S(t)$, then $t_1[t_2 / x] = S(t[t_2 / x])$
5. If $t_1 = F_i^m(s_1, \ldots, s_m)$, then $t_1[t_2 / x] = F_i^m(s_1[t_2 / x], \ldots, s_m[t_2 / x])$.

We now describe the only two rules which allow us to deduce a new equation either from one equation or from two equations.

The Substitution Rule (SR):

If $t_1 = t_2$ is an equation and the variable x occurs either in t_1 or t_2 (or both), then from $t_1 = t_2$ we can infer the new equation $t_1[\bar{n} / x] = t_2[\bar{n} / x]$ where $\bar{n}$ is any numeral. Sometimes, an application of (SR)

is written schematically as:

$$\frac{t_1 = t_2}{t_1[\bar{n} / x] = t_2[\bar{n} / x]}$$

The Replacement Rule (RR):

Given an equation $t_1 = t_2$ with no variable in it, if t_2 is of the form $t_2 = s[F^k(\bar{n}_1, \ldots . \bar{n}_k) / x]$ where F^k is a function symbol of arity k, and $\bar{n}_1, \ldots, \bar{n}_k$ are numerals, and if $F^k(\bar{n}_1, \ldots, \bar{n}_k) = \bar{n}$ is an equation (with $\bar{n}$ a numeral), from $t_1 = s[F^k(\bar{n}_1, \ldots, \bar{n}_k) / x]$ and $F^k(\bar{n}_1, \ldots, \bar{n}_k) = \bar{n}$ we can infer, $t_1 = s[\bar{n} / x]$, the result of substituting $\bar{n}$ for $F^k(\bar{n}_1, \ldots, \bar{n}_k)$ in the right-hand side of $t_1 = t_2$.

Schematically, we write:

$$\frac{t_1 = s[F^k(\bar{n}_1, \ldots, \bar{n}_k) / x] \qquad F^k(\bar{n}_1, \ldots, \bar{n}_k) = \bar{n}}{t_1 = s[\bar{n} / x]}$$

where x does not occur in t_1.

Deduction (Derivation)

Given a finite set $\mathcal{E}$ of equations, we say that a sequence of equations $e_1, \ldots, e_m$ is a deduction of e_m from $\mathcal{E}$, written as $\mathcal{E} \vdash e_m$ if the following conditions hold:

1. For each e_k in the sequence, either:
   a) e_k belongs to $\mathcal{E}$.
   b) $\exists$ $i < k$ and e_k is obtained from e_i by application of rule (SR).
   c) $\exists$ $i,j < k$ and e_k is obtained from e_i and e_j by application of rule (RR).

K-H-G Computability

Given a finite set $\mathcal{E}$ of equations, assuming that the equations are ordered as $\mathcal{E} = \{e_1, \ldots, e_N\}$ and that among the function letters occurring in the equations in $\mathcal{E}$, a function letter F^m is singled out as the "principal function letter," we say that a function ϕ of m argument is K-H-G computable if for all $x_1, \ldots, x_m, y$ in N, $\phi(x_1 \ldots, x_m) = y$ iff $\mathcal{E} \vdash F^m(\bar{x}_1, \ldots, \bar{x}_m) = \bar{y}$.

So, ϕ is K-H-G computable, if for all arguments $x_1, \ldots, x_m$, there is a deduction (computation) of $F^m(\bar{x}_1, \ldots, \bar{x}_m) = \bar{y}$ from the equations in $\mathcal{E}$.

For example, we note that the function $\phi(x) =$ if $x = 0$ then 4 else 7 is K-H-G computable from the system of equations:

$$S\begin{cases} H(x,y) = 7 \\ F(0) = 4 \\ F(x+1) = H(x,F(x)) \end{cases}$$

with principal function letter F.

Our next goal is to show that the class of K-H-G computable functions is exactly the class of partial recursive functions. First, we show that every partial recursive function is K-H-G computable.

The base functions are obviously K-H-G computable:

(1) The function E such that $E(x) = 0$ for all x is computed by the single equation:

$$F^1(x) = 0$$

(2) The successor function S such that $S(x) = x + 1$ for all x is computed by the single equation:

$$F^1(x) = S(x)$$

(3) The projection function $P_i^n(x_1, \ldots, x_n) = x_i$ is computed by the single equation :

$$F^n(x_1, \ldots, x_n) = x_i$$

Closure under composition, primitive recursion and minimization is done in the following Lemmas. We only indicate the main line of the proofs, leaving details to the reader.

Lemma 1

If $g_1, \ldots, g_m$ are m K-H-G computable functions of n arguments and h is a K-H-G computable function of m arguments, then their composition $f = h \circ (g_1, \ldots, g_m)$ is K-H-G computable.

Proof: Assume g_i is computed by the system of equations E_i with principal function G_i, and that h is computed by the system E_{m+1} with principal function H. Then, the system of equations obtained by taking the union of the E_i, $1 \leq i \leq m+1$, together with the equation $F(x_1, \ldots, x_n) = H(G_1(x_1, \ldots, x_n), \ldots, G_m(x_1, \ldots, x_n))$, with principal function F, computes f. (Note that we have to make sure that the function letters used in the E_i, $1 \leq i \leq m+1$ are distinct).

Lemma 2

If g is a K-H-G computable function of n arguments and h is a K-H-G computable function of n + 2 arguments, then the function f of n + 1 arguments obtained by primitive recursion from g and h is K-H-G computable.

Proof: Assume g is computed by a system of equations E_1 with principal function G and H is computed by a system of equations E_2 with principal function H. Then, the system of equations $E_1 \cup E_2$ together with the equations:

$$F(0,x_1, \ldots, x_n) = G(x_1, \ldots, x_n)$$

$$F(S(y),x_1, \ldots, x_n) = H(y,F(x_1, \ldots, x_n),x_1, \ldots, x_n)$$

with principal function F, computes f.

Lemma 3

Let g be a K-H-G computable function of n + 1 arguments. The function f obtained from g by minimization is K-H-G computable, where $f(x_1, \ldots, x_n) = \min y[g(y,x_1, \ldots, x_n) = 0]$.

Proof: Let g be computed by the system of equations E_1 with principal function G. We need an auxiliary function $\bar{g}$ defined by primitive recursion as follows:

$$\bar{g}(0,x_1, \ldots, x_n) = 1$$

$$\bar{g}(y + 1,x_1, \ldots, x_n) = sg(g(y,x_1, \ldots, x_n) \cdot \bar{g}(y,x_1, \ldots, x_n))$$

Since sg and multiplication are primitive recursive, they are K-H-G computable, and since g is K-H-G computable, by Lemma 1 and Lemma 2, $\bar{g}$ is K-H-G computable. Let E_2 be the system of equations computing $\bar{g}$ with principal function $\bar{G}$. Note that $\bar{g}$ has the following property:

For any $x_1, \ldots, x_n$, if $\min y[g(y,x_1, \ldots,x_n) = 0]$ is undefined, then for all y, $\bar{g}(y,x_1, \ldots, x_n) = 1$. If $\min y[g(y,x_1, \ldots,x_n) = 0] = z$, then for $y \leq z$, $\bar{g}(y,x_1, \ldots, x_n) = 1$ and for $y > z$, $\bar{g}(y,x_1, \ldots, x_n) = 0$. In particular, $\bar{g}(z,x_1, \ldots, x_n) = 1$ and $\bar{g}(z + 1,x_1, \ldots, x_n) = 0$. Therefore, if $\min y[g(y,x_1, \ldots,x_n) = 0]$ is defined $\bar{g}$ tells us for which value. Then, $f(x_1, \ldots, x_n) = \min y[g(y,x_1, \ldots,x_n) = 0]$ is computed by the system E_2 together with the two equations:

$$F(x_1, \ldots, x_n) = H(\bar{G}(y,x_1, \ldots, x_n),\bar{G}(S(y),x_1, \ldots, x_n),y)$$

$$H(\bar{1},\bar{0},y) = y,$$

with principal function F.

Collecting the above results, we have shown:

Theorem 4

Every partial recursive function is K-H-G computable.

We now proceed to show the converse of Theorem 4. The technique involved in showing that every K-H-G computable function is partial recursive consists in coding the derivation of equations $F(\bar{n}_1, \ldots, \bar{n}_k) = \bar{n}$ from a system of equations E. Such coding schemes are sometimes referred to as Godel numberings. In order to code derivations, we have to assign codes to variables, 0, terms, equations and derivations. We proceed inductively as follows:

Variable: The variable x_i has the code

$\#i = \langle i,0 \rangle$

Constant 0: The code of 0 is $\#0 = \langle 3,1,0 \rangle$

Terms: Each term of the form S(t) has the code $\#S(t) = \langle 3,1,\#t \rangle$

Each term $F_i^m(t_1, \ldots, t_m)$ has the code $\#F_i^m(t_1, \ldots, t_m) = \langle m + 2, i + 2, \#t_1, \ldots, \#t_m \rangle$

Equations: The code of an equation $t_1 = t_2$ is $\#(t_1 = t_2) = \langle \#t_1, \#t_2 \rangle$

Sequences of Equations

The code of a sequence of equations $e_0, e_1, \ldots, e_m$ is $\#(e_0, e_1, \ldots, e_m) = \langle m + 2, \#e_0, \ldots, \#e_m \rangle$

Next, we define some useful primitive recursive predicates and functions. For every integer n, let code(n) $= \#\bar{n}$, the code of the numeral $\bar{n}$. Also, for any x,

$$\text{val}(x) = \begin{cases} n \text{ if } x = \#\bar{n} \\ x + 1 \text{ otherwise} \end{cases}$$

The predicates are:

Var (x) iff x is the code of a variable

Num (x) iff x is the code of a numeral

Term (x) iff x is the code of a term

Eqn (x) iff x is the code of an equation

We postpone to the next section the proof that the above functions and predicates are primitive recursive.

We also need two predicates Subs and Rep expressing whether an equation has been obtained from another equation by the substitution rule, or whether an equation has been obtained from two equations by the Replacement rule.

Subs (z_1,z_2) hold iff z, and z_2 are codes of equations say e_1 and e_2, and e_2 is derivable from e_1 by the rule (SR).

Rep (z_1,z_2,z_3) holds iff z_1,z_2,z_3 are codes of equations, say e_1,e_2,e_3 and e_3 is derivable from e_1 and e_2 by the rule (RR).

Next, we need a predicate Der expressing that a sequence of equations constitute a derivation of an equation of the form $F_0^1(\bar{n}_1, \ldots, \bar{n}_k) = \bar{n}$. For simplicity, we will consider the principal function letter F_0^1 of one argument.

Der (u,z) holds iff z is the code of a sequence of equations constituting a derivation from E of an equation of the form $F_0^1(\bar{u}) = \bar{n}$, where $\bar{n}$ is a numeral.

Der (u,z) has the following primitive recursive definition. First, given the set of equations E, we also have the predicate EN (z) which holds iff z is the code of an equation in E. Also, let $L(z) = \pi_1(z)$ (the "length" of z).

Then we have:

Der (u,z) iff $\forall\, k \leq L(z) - 2$

[EN($\pi(k + 2, L(z),z)$) or

$\exists\, i \leq k - 1\,(\mathrm{Subs}(\pi(i + 2, L(z), z), \pi(k + 2, L(z), z)))$

or $(\exists\, i \leq k - 1)(\exists\, j \leq k - 1)(\mathrm{Rep}(\pi(i + 2, L(z), z),$

$\pi(j + 2, L(z), z), \pi(k + 2, L(z), z)))]$ and

$\pi_1(\pi(L(z), L(z), z)) = \langle 3, 2, \mathrm{code}(u) \rangle$ and

$\mathrm{Num}(\pi_2(\pi(L(z), L(z), z)))$.

Finally, we leave to the reader to check that the function f defined by the system of equations E has the following definition:

Let $g(u) = \min z[\mathrm{Der}(u, z)]$.

Then, $f(u) = \mathrm{Val}(\pi_2(\pi(L(g(u)), L(g(u)), g(u))))$.

Therefore, f is partial recursive.

Theorem 5

Every K-H-G computable function is a partial recursive function.

Appendix:

Proof that the auxiliary functions and predicates are primitive recursive.

$\mathrm{code}(0) = \langle 3, 1, 0 \rangle$

$\mathrm{code}(n + 1) = \langle 3, 1, \mathrm{code}(n) \rangle$

$\mathrm{val}(z) = \min x \leq z\,[\mathrm{code}(x) = z]$

$\mathrm{Var}(z)$ iff $\exists\, x \leq z[x = \pi_1(z)]$ and $\pi_2(z) = 0$

$\mathrm{Num}(z)$ iff $\exists\, x \leq z[\mathrm{code}(x) = z]$

The predicate term is defined by course of value recursion. Let T be the characteristic function of the predicate term.

$T(0) = 0$

$$T(z+1) = \begin{cases} 1 \text{ if } Var(z+1) \text{ or } z+1 = \langle 3,1,0 \rangle \text{ or} \\ (L(z+1) = 3 \text{ and } \pi(2,3,z+1) = 1 \\ \text{and } T(\pi(3,3,z+1))) \text{ or} \\ (\pi(2,L(z+1),z+1) > 1 \text{ and} \\ (\forall\, i \le L(z+1) - 3)(T(\pi(i+3, L(z+1), z+1)))) \\ 0 \text{ otherwise} \end{cases}$$

(Recall that $L(z) = \pi_1(z)$). Eqn(z) iff $Term(\pi_1(z))$ and $Term(\pi_2(z))$

The predicate tsub(u,v,w) is defined as follows:

$$tsub(u,v,w) = \begin{cases} \#(t_w(t_u / t_v)) \text{ if } u = \#t_u,\ v = \#t_v,\ w = \#t_w \\ w \text{ otherwise} \end{cases}$$

(u,v,w are the codes of three terms t_u,t_v, and t_w). The predicate tsub can be defined by course of value recursion.

$tsub(u,v,0) = 0$

$$tsub(u,v,w+1) = \begin{cases} u & \text{if } v = w+1 \\ w+1 & \text{if}(v \neq w+1 \text{ and } var(w+1)) \text{ or } w+1 = \langle 3,1,0 \rangle \\ h(u,v,w) & \text{if } v \neq w+1 \text{ and } Term(w+1) \text{ and not } var(w+1) \\ & \quad \text{and } w+1 \neq \langle 3,1,0 \rangle \\ w+1 & \text{otherwise} \end{cases}$$

where $h(u,v,w) = \langle L(z+1), \pi(2,L(z+1),z), tsub(u,v,\pi(3,L(z+1),z)), \ldots$

$tsub(u,v,\pi(L(z+1),L(z+1),z)) \rangle$

Then, we can define Subs as follows:

$Subs(z_1,z_2)$ iff $Eqn(z_1)$ and $Eqn(z_2)$ and $(\exists\, n \le z_2)(\exists\, v \le z_1)[Num(n)$ and $Var(v)$ and $(\pi_1(z_2) = tsub(n,v,\pi(z_1)))$ and $(\pi_2(z_2) = tsub(n,v,\pi_2)))]$

Finally, recall that $Rep(z_1,z_2,z_3)$ holds if z_1 is the code of an equation $t_1 = s[F(\bar{n}_1, \ldots, \bar{n}_k) / x]$, z_2 is the code of an equation $F(\bar{n}_1, \ldots, \bar{n}_k) = \bar{n}$ and z_3 is the code of the equation $t_1 = s[\bar{n} / x]$ (also t_1 contains no variable and s contains no variable but x). A predicate Novar(z) which holds iff Eqn(z) and the equation coded by z contains no variable can easily be defined by primitive recursion. We leave the details as an exercise. The predicate special(z) which holds iff Term(z) and z codes a term of the form $F^k(\bar{n}_1, \ldots, \bar{n}_k)$ where $\bar{n}_1, \ldots, \bar{n}_k$ are numerals can also be defined by primitive recursion. We then have the following definition for $Rep(z_1,z_2,z_3)$.

$Rep(z_1,z_2,z_3)$ iff $Eqn(z_1)$ and $Novar(z_1)$ and $Eqn(z_2)$ and $special(\pi_1(z_2))$ and $Num(\pi_2(z_2))$ and $Eqn(z_3)$ and $(\exists\, z \leq bound(z_1,z_2))[Eqn(z)$ and $\pi_1(z_1) = \pi_1(z)$ and $\pi_2(z_1) = tsub(\pi_1(z_2), <1,0>, \pi_2(z))$ and $\pi_1(z_3) = \pi_1(z)$ and $\pi_2(z_3) = tsub(\pi_2(z_2), <1,0>, \pi_2(z))]$.

Note that the auxiliary equation $t_1 = s$ whose code is z is obtained from the equation $t_1 = s[F^k(\bar{n}_1, \ldots, \bar{n}_k) / x]$ whose code is z_1 by substituting x for some occurrences of $F^k(\bar{n}_1, \ldots, \bar{n}_k)$. Therefore, the code of $t_1 = s$ is no more (by monotonicity) than the code of the equation obtained by substituting x for <u>all</u> occurrences of $F^k(\bar{n}_1, \ldots, \bar{n}_k)$ in the first equation whose code is z_1, and so we have:

$$bound(z_1,z_2) = <tsub(<1,0>, \pi_1(z_2), \pi_1(z_1)),\ tsub(<1,0>, \pi_2(z_2), \pi_1(z_1))>$$