THE RECURSION THEOREM (FINAL)

REBECCA WEBER

Kleene's Recursion Theorem, though provable in only a few lines, is fundamental to computability theory and allows strong self-reference in proofs. It is a fixed-point theorem in the sense that it asserts for any total computable function f, there is a number n such that n and $f(n)$ index (or code) the same partial computable function (though we need not have $f(n) = n$). In this paper we will prove the Recursion Theorem, two generalizations, and some simple applications.

Note that these theorems date back to near the beginning of the study of computability theory; those attributed to Kleene may or may not have been published by him at the time of their proof. For Kleene's write-up of his own results, see his text [1]. The *S-m-n* Theorem and all versions of the Recursion Theorem are attributed to him; the Relativized *S-m-n* Theorem is not attributed to anyone, which is likely because it is such a natural and straightforward generalization that many people proved it independently.

We begin with some background. As is standard in computability theory (see, e.g., the text by Soare [6]), let the enumeration of all partial computable functions be denoted $\{\varphi_e : e \in \mathbb{N}\}$. The *index* or *code* of φ_e is e, and there is a computable bijection with computable inverse between functions and codes, which we assume is fixed and known from the beginning. This bijection allows one to obtain φ_e from e and encode any defined function into an index e. We say $\varphi_n = \varphi_m$ if the domains of the two functions are equal and on the (mutual) domain the output values of the functions are always equal. As every function has infinitely-many indices, this is not an empty definition. For other standard computability-theoretic definitions see [6].

The basic theorem needed to prove the Recursion Theorem and its variants is the following, known as the *S-m-n* Theorem or the parametrization theorem.

Theorem 1 (S-m-n Theorem, Kleene). *Given m, n, there is a primitive recursive one-to-one function S^m_n such that for all e, all n-tuples $\bar{x}$, and all m-tuples $\bar{y}$,*

$$\varphi_{S^m_n(e,\bar{x})}(\bar{y}) = \varphi_e(\bar{x}, \bar{y}).$$

Very roughly, $S_n^m(e, \bar{x})$ decodes e into an $n+m$-input function, fills the entries of $\bar{x}$ into the appropriate inputs, and recodes the resultant m-input function. It is important for our use that S_n^m is total, but that it is actually primitive recursive and one to one will not be used. In the proofs using this theorem, the input e will be fixed, and so we will ignore it and think of S_n^m as a function of n variables ($\bar{x}$).

Theorem 2 (Recursion or Fixed-Point Theorem, Kleene)**.** *Suppose that f is a total computable function; then there is a number n such that $\varphi_n = \varphi_{f(n)}$. Moreover, n is computable from an index for f.*

The idea of the proof is as follows. If we could guarantee $f(\varphi_x(x))\downarrow$, then using the slightly circular choice of x as the index of $f \circ \varphi_x$ we would have $f(\varphi_x(x)) = (f \circ \varphi_x)(x) = \varphi_x(x)$, and so the functions indexed by $f(\varphi_x(x))$ and $\varphi_x(x)$ would be the same because those values would be equal. However, there is no guarantee of halting for $f(\varphi_x(x))$, and for a function such as $f(n) = n+1$ we must have divergence. However, we may define a function on two inputs that mimics the desired function:

$$\varphi_e(x, y) = \begin{cases} \varphi_{f(\varphi_x(x))}(y) & \varphi_x(x)\downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

By the S-m-n Theorem 1, this function is equal to $\varphi_{s(x)}(y)$ for a total computable function s. The key fact is that if $\varphi_x(x)\uparrow$, $s(x)$ will index a function that diverges everywhere, but $s(x)$ itself will still be defined.

Proof of Theorem 2. By the S-m-n Theorem there is a total computable function $s(x)$ such that for all x and y

$$\varphi_{f(\varphi_x(x))}(y) = \varphi_{s(x)}(y).$$

Let m be any index such that φ_m computes the function s; note that s and hence m are computable from an index for f. Rewriting the statement above yields

$$\varphi_{f(\varphi_x(x))}(y) = \varphi_{\varphi_m(x)}(y).$$

Then, putting $x = m$ and letting $n = \varphi_m(m)$ (which is defined because s is total), we have

$$\varphi_{f(n)}(y) = \varphi_{f(\varphi_m(m))}(y) = \varphi_{s(m)}(y) = \varphi_{\varphi_m(m)}(y) = \varphi_n(y)$$

as required. □

From the Recursion Theorem we obtain the immediate corollary that there are numbers n, m such that $\varphi_n = \varphi_{n+1}$ and $\varphi_m = \varphi_{2m}$, and we may continue in this manner for any total computable function.

Corollary 3. *If f is a total computable function then there are arbitrarily large numbers n such that $\varphi_{f(n)} = \varphi_n$.*

Corollary 3 is proved by showing that for all N, there is a function that agrees with f on $n > N$ and cannot have a fixed point below N, and hence has only fixed points that are also fixed for f.[1] The following corollary is proved using the *S-m-n* Theorem to define a function from f to which to apply the Recursion Theorem.

Corollary 4. *If $f(x, y)$ is any computable function there is an index e such that $\varphi_e(y) = f(e, y)$.*

Corollary 4 gives the existence of n, m such that $\varphi_n(x) = x^n$ and $\mathrm{Dom}(\varphi_m) = \{m\}$.[2]

Many of the uses of the Recursion Theorem in computability-theoretic constructions can be summed up as building a Turing machine using the index of the finished machine. The construction will have early on a line something like "We construct a partial computable function ψ and assume by the Recursion Theorem that we have an index e for ψ." The construction, which is computable, is itself the function for which we seek a fixed point. When the construction is given the input e to be interpreted as the index of a partial computable function, it can use e to produce e', which is an index of the function ψ it is trying to build. The Recursion Theorem says the construction will have a fixed point, some i such that i and i' both index the same function, which must be ψ. Furthermore this fixed point will be *computable* from an index for the construction itself.

Application of the Recursion Theorem to a construction depends on the construction being *uniform.* A process is uniform if its definition does not depend on the values it is given as input; that is, if it acts like a function, which is defined ahead of time and then acts in particular ways based on the specific inputs. Uniformity allows the construction to *have* an index from which to compute the fixed point.

Our first extension of the Recursion Theorem gives a fixed point of sorts for functions of two inputs.

Theorem 5 (Recursion Theorem with Parameters, Kleene)**.** *If $f(x, y)$ is a total computable function, then there is a total computable function $n(y)$ such that $\varphi_{n(y)} = \varphi_{f(n(y),y)}$ for all y.*

Proof. Let the index e code the function

$$\varphi_e(x, y, z) = \begin{cases} \varphi_{\varphi_x(x,y)}(z) & \text{if } \varphi_x(x, y)\downarrow; \\ \uparrow & \text{otherwise.} \end{cases}$$

[1]Were they not homework assignments, I would prove Corollaries 3 and 4 in the paper.

[2]Likewise, I would explain/derive these applications.

By the *S-m-n* Theorem 1 there is a total computable function $d(x,y)$ such that $\varphi_{d(x,y)}(z) = \varphi_e(x,y,z)$. Since f and d are both partial computable, there is an index v such that $\varphi_v(x,y) = f(d(x,y),y)$. Then $n(y) = d(v,y)$ is a fixed point for f, since unpacking the definitions of n, d and v (and then repacking n) we see

$$\varphi_{n(y)} = \varphi_{d(v,y)} = \varphi_{\varphi_v(v,y)} = \varphi_{f(d(v,y),y)} = \varphi_{f(n(y),y)}.$$

□

In fact we may replace the total function $f(x,y)$ with a partial function $\psi(x,y)$ and have total computable n such that whenever $\psi(n(y),y)$ is defined, $n(y)$ is a fixed point. The proof is identical to the proof of the recursion theorem with parameters. Note that the parametrized version implies the original version by considering functions which depend only on their first input (in that case $n(y)$ is a constant function and may be viewed as simply a value n).

The second generalization of the Recursion Theorem we will include is the Relativized Recursion Theorem, which also allows parameters. It implies the previous two by assigning the oracle $\emptyset$ to all functions. The notion of *relativization* is one of fixing some set A and always working with A as your oracle: working *relative to A*. Then computability becomes computability in A (being equal to φ_e^A for some e, also called A-computability) and enumerability become enumerability in A (being equal to $W_e^A := \mathrm{Dom}(\varphi_e^A)$ for some e). In general when a computability-theoretic theorem is relativized, every function and set acquires an oracle, but the *S-m-n* Theorem and Recursion Theorem are exceptions, as explained below.

Theorem 6 (Relativized S-m-n Theorem)**.** *For every $m,n \geq 1$ there exists a one-to-one computable function S_n^m of $m+1$ variables so that for all sets $A \subseteq \mathbb{N}$ and for all $e, y_1, \ldots, y_m \in \mathbb{N}$,*

$$\varphi^A_{S_n^m(e,y_1,\ldots,y_m)}(z_1,\ldots,z_n) = \varphi_e^A(y_1,\ldots,y_m,z_1,\ldots,z_n).$$

The proof is essentially identical to the proof of the original *S-m-n* Theorem 1, because the definition of an oracle machine does not depend on the particular oracle in use. That is, the definition of an oracle computation is uniform. That allows the function S_n^m to be not just computable in A, but computable. The Relativized Recursion Theorem, below, is as a consequence able to produce a fixed point computably from an index for the function f, instead of just A-computably.

Theorem 7 (Relativized Recursion Theorem (with Parameters), Kleene)**.** *Let $A \subseteq \mathbb{N}$. If $f(x,y)$ is an A-computable function, then there is a* computable *function $n(y)$ such that $\varphi^A_{n(y)} = \varphi^A_{f(n(y),y)}$ for all y. Moreover,*

n does not depend on A; namely, if $f(x,y) = \varphi_e^A(x,y)$, $n(y)$ can be found uniformly in e.

Proof. Let the index e code the function

$$\varphi_e^A(x,y,z) = \begin{cases} \varphi^A_{\varphi_x(x,y)}(z) & \text{if } \varphi_x(x,y)\downarrow; \\ \uparrow & \text{otherwise.} \end{cases}$$

By the Relativized S-m-n Theorem there is a total computable function $d(x,y)$ such that $\varphi^A_{d(x,y)}(z) = \varphi_e^A(x,y,z)$. Since f and d are both computable in A, there is an index v such that $\varphi_v^A(x,y) = f(d(x,y),y)$. Then $n(y) = d(v,y)$ is a fixed point for f, since unpacking the definitions of n, d and v (and then repacking n) we see

$$\varphi^A_{n(y)} = \varphi^A_{d(v,y)} = \varphi^A_{\varphi_v^A(v,y)} = \varphi^A_{f(d(v,y),y)} = \varphi^A_{f(n(y),y)}.$$

□

Our final result is an application of the Relativized Recursion Theorem to the structure of Turing degrees. Recall that the Turing degrees are the quotient structure of $\mathcal{P}(\mathbb{N})$ under Turing equivalence, and they are partially ordered by $\leq_T$, Turing reduction.

Definition 8. The *Turing jump* of a set A, denoted A', is the Halting Set relativized to A. That is, $A' = \{e : \varphi_e^A(e)\downarrow\}$.

If $A \leq_T B$, then $A' \leq_T B'$ (and hence the jump is a well-defined operation on degrees), but it may be that $A <_T B$ and $A' \equiv_T B'$. We recall that the degree of computable sets is denoted $\mathbf{0}$ and hence the degree of the Halting Set is $\mathbf{0}'$. All degrees below $\mathbf{0}'$ must have jumps between $\mathbf{0}'$ and $\mathbf{0}''$. Degrees on the upper and lower extremes are called *high* and *low*, respectively. The following definition generalizes the notions of lowness and highness.

Definition 9. For each $n > 0$, define a degree $\boldsymbol{a} \leq \mathbf{0}'$ to be low_n (high_n) if $\mathbf{0}^{(n)} = \boldsymbol{a}^{(n)}$ ($\boldsymbol{a}^{(n)} = \mathbf{0}^{(n+1)}$). A set A is low_n (high_n) exactly when $\deg(A)$ is. We use low_n and high_n also to denote the collection of all low_n or high_n degrees. For convenience, we set $\text{low}_0 = \{\mathbf{0}\}$ and $\text{high}_0 = \{\mathbf{0}'\}$.

Note that once a degree jumps to a jump of $\emptyset$, its future jumps all match $\emptyset$'s, so $\text{low}_n \subseteq \text{low}_{n+1}$ and $\text{high}_n \subseteq \text{high}_{n+1}$. We state without proof that this containment is proper; the result is a corollary of the Jump Theorem 12. All proofs omitted below may be found in Soare [6], Chapter VIII §3.

Proposition 10. *For all $n \in \mathbb{N}$, $\text{low}_n \neq \text{low}_{n+1}$ and $\text{high}_n \neq \text{high}_{n+1}$.*

In some sense the low degrees are "close to" computable, and the high degrees are "close to" complete. The hierarchy of low$_n$ and high$_n$ degrees gradually carves out more and more of the c.e. degrees as n increases: low$_1$ (or just low) degrees are near $\mathbf{0}$, low$_2$ degrees come a little further up, low$_3$ a little further up yet; meanwhile the high$_n$ degrees are creeping down from near $\mathbf{0}'$. It is easy to see that they cannot overlap, but natural to ask whether they "meet in the middle". Another corollary of the Jump Theorem 12, which uses the Relativized Recursion Theorem, says no, there is a gap. Degrees which are c.e. but neither low$_n$ nor high$_n$ for any n are called *intermediate*.

Proposition 11 (Martin [3], Lachlan [2], Sacks [5])**.** *There is an intermediate c.e. degree $\boldsymbol{a}$. That is, $\mathbf{0}^{(n)} < \boldsymbol{a}^{(n)} < \mathbf{0}^{(n+1)}$ for all n.*

Proposition 11 was proved independently by Martin and Lachlan, using different sorts of priority arguments. Sacks gave a shorter proof based on his Jump Theorem; before giving his proof we state that theorem. The fully general statement below gives the relativized version of the way it is usually stated ($Y = \emptyset$; see Soare [6] Theorem VIII.3.1) and combines it with a further observation by Sacks (adding the set D; see Soare [6] Remark VIII.3.2).

Theorem 12 (Sacks Jump Theorem [4])**.** *Suppose we are given sets Y, S, C, and D such that*

(i) *D is c.e. in Y,*
(ii) *S is c.e. in Y',*
(iii) *$D', Y' \leq_T S$,*
(iv) *$Y <_T C \leq_T Y'$, and*
(v) *$C \not\leq_T D$.*

Then there exists a set A such that $A \not\leq_T Y$, A is c.e. in Y, $A' \equiv_T S$, $C \not\leq_T A$ and $D \leq_T A$. Furthermore, an index of A can be found uniformly from indices for S, C, and D.

In other words, if the set S *could* be the jump of a Y-c.e. set, then up to Turing equivalence it is. If B is a Y-c.e. set, then $Y \leq_T B$ gives $Y' \leq_T B'$ and $B \leq_T Y'$ gives that B' must be c.e. in Y'; the theorem says those properties fully specify degrees that are jumps. Moreover we have control over what sort of Y-c.e. set we have jumped from. The $C \not\leq_T A$ property is *cone avoidance*: we can choose this "jump inverse" A to be outside the Turing cone above any Y'-computable set C that is reasonable; that is, any C that is not Y-computable. We can also ensure A *is* able to compute another specific Y-c.e. set; again, it must be reasonable: if $D' \not\leq_T S = A'$, we can't have $D \leq_T A$. The last condition ensures C and D are compatible. This theorem is proved by an infinite injury construction.

Finally, we give the proof of the existence of an intermediate degree, in more detailed form than that given in Sacks [5] and Soare [6]. Recall that the *join* of two sets, $A \oplus B$, is their disjoint union; its degree is the least upper bound of the degrees of A and B.

Proof of 11 (*Sacks* [5]). Given Y and $S = (W_x^{Y'}) \oplus Y'$ (note that S is c.e. in Y' and computes Y'), apply the Jump Theorem 12 with $C = Y'$ and $D = Y$. Note that we can fix indices of machines that compute exactly their oracle and the jump of their oracle, so (using composition for S) the only index that varies is x. The uniformity of the Jump Theorem then gives a total computable function q such that $W_{q(x)}^Y$ is the set A from the theorem. $A \not\leq_T Y$ and $D \leq_T A$ combine to give $Y <_T A$, and A c.e. in Y and $C \not\leq_T A$ combine to give $A <_T Y'$. In short, for every $x \in \mathbb{N}$ and $Y \subseteq \mathbb{N}$,

$$Y <_T W_{q(x)}^Y <_T Y' \quad \text{and} \quad (W_{q(x)}^Y)' \equiv_T (W_x^{Y'}) \oplus Y'.$$

Now apply the Relativized Recursion Theorem 7 to obtain a fixed point n such that $W_{q(n)}^Y = W_n^Y$ for all $Y \subseteq \mathbb{N}$. Define $\boldsymbol{a} = \deg(W_n^\emptyset)$.

We prove by induction that $\boldsymbol{a}$ is intermediate by proving $(W_n^\emptyset)^{(m)} \equiv_T W_n^{\emptyset^{(m)}}$. Then, since $W_n^{\emptyset^{(m)}} = W_{q(n)}^{\emptyset^{(m)}}$, we have $\emptyset^{(m)} <_T W_n^{\emptyset^{(m)}} <_T \emptyset^{(m+1)}$. The case $m = 0$ is by definition. Suppose for some $m \geq 0$ we have proved $(W_n^\emptyset)^{(m)} \equiv_T W_n^{\emptyset^{(m)}}$. Then

$$\begin{aligned}((W_n^\emptyset)^{(m)})' \equiv_T (W_n^{\emptyset^{(m)}})' = (W_{q(n)}^{\emptyset^{(m)}})' &\equiv_T (W_n^{(\emptyset^{(m)})'}) \oplus (\emptyset^{(m)})' \\ &= (W_n^{\emptyset^{(m+1)}}) \oplus \emptyset^{(m+1)} = (W_{q(n)}^{\emptyset^{(m+1)}}) \oplus \emptyset^{(m+1)}.\end{aligned}$$

Since $\emptyset^{(m+1)} <_T W_{q(n)}^{\emptyset^{(m+1)}}$, the join $(W_{q(n)}^{\emptyset^{(m+1)}}) \oplus \emptyset^{(m+1)}$ is actually Turing equivalent to $W_{q(n)}^{\emptyset^{(m+1)}}$, and hence (using once more the fact that n is a fixed point for q), $(W_n^\emptyset)^{(m+1)} \equiv_T W_n^{\emptyset^{(m+1)}}$. $\square$

References

[1] Kleene, S.C. *Introduction to Metamathematics.* Van Nostrand, New York, 1952.

[2] Lachlan, A.H. On a problem of G.E. Sacks. *Proceedings of the American Mathematical Society* **16**(1965): 972–979.

[3] Martin, D.A. On a question of G.E. Sacks. *Journal of Symbolic Logic* **31**(1966): 66–69.

[4] Sacks, G.E. *Degrees of Unsolvability.* Annals of Mathematical Studies No. 55, Princeton University Press, Princeton, N.J., 1963.

[5] Sacks, G.E. On a theorem of Lachlan and Martin. *Proceedings of the American Mathematical Society* **18**(1967): 140-141.

[6] Soare, R.I. *Recursively Enumerable Sets and Degrees.* Springer-Verlag, 1987.