Computability and Decidability

Robert Rynasiewicz
Mathematical Logic II

Spring 2022

Super Exponentiation (Tetration)

Let f_1 be addition, f_2 multiplication, f_3 exponentiation, i.e.,

$$\begin{aligned} x \uparrow 0 &=_{df} 1 \\ x \uparrow S(y) &=_{df} x \cdot (x \uparrow y). \end{aligned}$$

Let f_4 be **super-exponentiation (tetration)**, i.e.,

$$\begin{aligned} x \uparrow\uparrow 0 &=_{df} x \\ x \uparrow\uparrow S(y) &=_{df} x \uparrow (x \uparrow\uparrow y), \end{aligned}$$

Example of Tetration

$$
\begin{aligned}
f_4(3,4) &= 3 \uparrow\uparrow S(3) \\
&= 3 \uparrow (3 \uparrow\uparrow 3) \\
&= 3 \uparrow (3 \uparrow\uparrow S(2)) \\
&= 3 \uparrow (3 \uparrow (3 \uparrow\uparrow 2)) \\
&= 3 \uparrow (3 \uparrow (3 \uparrow\uparrow S(1))) \\
&= 3 \uparrow (3 \uparrow (3 \uparrow (3 \uparrow\uparrow 1))) \\
&= 3 \uparrow (3 \uparrow (3 \uparrow (3 \uparrow\uparrow S(0)))) \\
&= 3 \uparrow (3 \uparrow (3 \uparrow (3 \uparrow (3 \uparrow\uparrow 0)))) \\
&= 3 \uparrow (3 \uparrow (3 \uparrow (3 \uparrow 3)))
\end{aligned}
$$

Example of Tetration (cont.)

$$
\begin{aligned}
f_4(3,4) &= 3 \uparrow (3 \uparrow (3 \uparrow (3 \uparrow 3))) \\
&= 3^{3^{3^{3}}} \\
&= 3^{3^{3^{9}}} \\
&= 3^{3^{19,683}}
\end{aligned}
$$

Super-Duper Exponentiation (Pentation)

Let f_5 be **super-duper-exponentiation (pentation)**, i.e.,

$$\begin{aligned} x \uparrow\uparrow\uparrow 0 &=_{df} x \\ x \uparrow\uparrow\uparrow S(y) &=_{df} x \uparrow\uparrow (x \uparrow\uparrow\uparrow y), \end{aligned}$$

so that

$$f_5(3,4) = 3 \uparrow\uparrow 3 \uparrow\uparrow 3 \uparrow\uparrow 3 \uparrow\uparrow 3$$

with right associativity.

I.e.,

$$f_5(3,4) = 3 \uparrow\uparrow 3 \uparrow\uparrow 3 \uparrow\uparrow 3^{19,683}.$$

Arbitrary n-ation

Following Knuth, $x \uparrow^3 y =_{df} x \uparrow\uparrow\uparrow y$. In general, for $n \geq 2$

$$\begin{aligned} x \uparrow^{n+1} 0 &=_{df} x \\ x \uparrow^{n+1} S(y) &=_{df} x \uparrow^n (x \uparrow^{n+1} y). \end{aligned}$$

And, provided $n > 2$, define

$$f_n(x, y) =_{df} x \uparrow^{n-2} y.$$

.

A Computable Function That Is Not P.R.

Now let

$$g(x) =_{df} f_x(x, x).$$

This is certainly computable, but it blows up so fast, it can't be p.r. — any unary p.r. function has a maximum growth rate determined by the "length" of its definition, comparable to $f_n(x, x)$ for some fixed n. But $g(x)$ will eventually grow much, much faster.

So, in addition to the computable but not p.r. function we got by a diagonalization argument, this is now another. Before considering how to generalize to the class of all computable functions, let's retrace the steps in constructing the sequence of f_n's for small values of n, beginning with f_0.

Diagnosis

Stipulate that

$$f_0(x, y) =_{df} S(y).$$

Then

$$\begin{aligned}
f_1(x, S(y)) &= f_0(x, f_1(x, y)) \\
f_2(x, S(y)) &= f_1(x, f_2(x, y)) \\
f_3(x, S(y)) &= f_2(x, f_3(x, y)) \\
f_4(x, S(y)) &= f_3(x, f_4(x, y)) \\
f_5(x, S(y)) &= f_4(x, f_5(x, y))
\end{aligned}$$

So, let $f(n, x, y) =_{df} f_n(x, y)$. This shows a pattern of *double* recursion:

$$f(S(n), x, S(y)) = f(n, x, f(S(n), x, y)).$$

The Ackermann-Péter Function

Let

$$
\begin{aligned}
p(0,y) &=_{df} S(y) && (1)\\
p(S(x),0) &=_{df} p(x,S(0)) && (2)\\
p(S(x),S(y)) &=_{df} p(x,p(S(x),y)). && (3)
\end{aligned}
$$

This is the so-called Ackermann-Péter function. How to compute, say $p(2,1)$, is not immediately obvious.

$$
\begin{aligned}
p(2,1) &= p(1,p(2,0)) \text{ by (3)}\\
&= p(1,p(1,1)) \text{ by (2)}\\
&= p(1,p(0,p(1,0))) \text{ by (3)}\\
&= p(1,p(0,p(0,1))) \text{ by (2)}\\
&= p(1,p(0,2)) \text{ by (1)}\\
&= p(1,3) \text{ by (1)}
\end{aligned}
$$

The Ackermann-Péter Function (cont.)

$$
\begin{aligned}
p(1,3) &= p(0,p(1,2)) \text{ by (3)} \\
&= p(0,p(0,p(1,1))) \text{ by (3)} \\
&= p(0,p(0,p(0,p(1,0)))) \text{ by (3)} \\
&= p(0,p(0,p(0,p(0,1)))) \text{ by (2)} \\
&= p(0,p(0,p(0,2))) \text{ by (1)} \\
&= p(0,p(0,3)) \text{ by (1)} \\
&= p(0,4) \text{ by (1)} \\
&= 5 \text{ by (1)}
\end{aligned}
$$

It's clear that this is not p.r., but involves a "do until" loop, in which you wait for whatever application of p you're on to bottom out to 0. So, how do we enlarge the class of computable fn's?

Regular Functions and Minimization

Defn. Let $g(\vec{x}, y)$ be an $(n+1)$-ary operation on $\mathbb{N}$. $g(\vec{x}, y)$ is **regular** iff for all $\vec{x}$ there is a y s.t. $g(\vec{x}, y) = 0$.

Scholium. The choice of the constant 0 is arbitrary. If we choose n instead (say, 1). Then if g is regular in the original sense, then $g + n$ would be regular in the new sense. The important thing is to pick a constant natural number and to stick with it.

Defn. Suppose $g(\vec{x}, y)$ is regular. Then

$$f(\vec{x}) =_{df} \mu y[g(\vec{x}, y) = 0]$$

is the least y s.t. $g(\vec{x}, y) = 0$ and f is said to be defined by **regular minimization** from g.

Minimization and Computability

Suppose $g(\vec{x}, y)$ is regular *and* computable. Then arguably, $f(\vec{x}) = \mu y[g(\vec{x}, y) = 0]$ is computable as well:

1. $y := 0$
2. do until $\{\ g(\vec{x}, y) = 0\ \}$
3. $\qquad y := y + 1$
4. end do
5. $f(\vec{x}) := y$

The μ-Recursive Functions

Defn. The **set of μ-recursive functions** is the closure of the set consisting of the zero fn., the successor fn. and the projection fns. under composition, primitive recursion, and regular minimization. I.e., M is the smallest set s.t.

- $0, S, I_j^n \in M$ for all $n \geq j > 0$,
- if $g_1, \ldots, g_k, h \in M$ and f is the composition of h with $g_1, \ldots, g_k$, then $f \in M$,
- if $g, h \in M$, and f is defined from g and h by primitive recursion, then $f \in M$, and
- if $g(\vec{x}, y) \in M$ and $g(\vec{x}, y)$ is regular, then

$$\mu y[g(\vec{x}, y) = 0] \in M.$$

Church's Thesis

Claim. The Ackermann-Péter Function is μ-recursive but not p.r.

Pf. [Skip].

Church's Thesis. The effectively computable (total) functions are the μ-recursive functions.

Convergence of "Evidence". Coincides with the class of Turing computable, abacus computable, λ-computable functions and more.

Thesis or definition? (cf. continuous function)

Remarks

- Accepting Church's Thesis allows us to prove quite a bit. E.g., whenever we can specify an effective procedure for computing a value, we can infer the function in question is μ-recursive without actually constructing a μ-recursive definition of the function. (Thus, Church's Thesis leads to non-constructive proofs and is not obviously a *conservative* definition.)
- Why can't we "diagonalize out"? The set of μ-recursive functions is not effectively enumerable — no effective procedure for determining whether an arbitrarily given μ-recursive function is regular.

Representability of the μ-Recursive Functions

Theorem. Suppose $f(x) = \mu y[g(x, y) = 0]$ and g is represented in Q by the wff $G(v_0, v_1, v_2)$. Then f is represented in Q by the Σ_1 wff

$$F(v_0, v_1) =_{df} G(v_0, v_1, 0) \wedge (\forall x < v_1)\exists y(G(v_0, x, y) \wedge y \neq 0).$$

Pf. It's obvious enough that $F(v_0, v_1)$ defines f in $\mathfrak{N}$. It's less obvious that

1. for all $n, m \in \mathbb{N}$, if $f(n) = m$, then $Q \vdash F(\overline{n}, \overline{m})$, and
2. for each $n \in \mathbb{N}$, $Q \vdash \exists! y F(\overline{n}, y)$,

but the details are left as an exercise.

Cor. Any μ-recursive fn. is representable in Q with a Σ_1 wff.

The Converse

Cor. By Church's Thesis, Q is "sufficiently strong": all decidable subsets of $\mathbb{N}$ are representable in Q.

Theorem. If T is a consistent axiomatic extension of Q, then *only* μ-recursive functions are representable in T.

Pf. Suppose $F(v_0, v_1)$ represents f in T. Then, for all $n, m \in \mathbb{N}$, if $f(n) = m$, then $T \vdash F(\overline{n}, \overline{m})$. To compute $f(n)$, effectively enumerate the theorems of T until $F(\overline{n}, \overline{m})$ appears. This is an effective procedure, so f is computable, and, by Church's Thesis, it is μ-recursive. ■

Cor. PA can represent only the functions that Q can represent.

Some Definitions

Terminology. Drop the "μ" when talking about μ-recursive fns.

Defn. A relation R is **recursive** iff its characteristic function χ_R is recursive.

Convention. Under Church's Thesis, we interchangeably speak of a set as being recursive or decidable.

Defn. A set $S \subseteq \mathbb{N}$ is **recursively enumerable** iff if there recursive function $f : \mathbb{N} \to S$ that is onto (f recursively enumerates S).

Convention. When we speak of sets of sentences being recursive or recursively enumerable, we mean the sets of their Gödel numbers.

Some Theorems

Theorem. Any consistent axiomatic extension of Q is undecidable.

Pf. Let T be a consistent axiomatic extension of Q. If T were decidable, then $Prov_T$ would be representable in T, which it's not.

Church's Theorem. The set of logical validities in any language containing $\mathcal{L}_{PA}$ is not decidable.

Pf. Q is finitely axiomatized. Let α be the conjunction of the axioms. If the set of logically valid sentences were decidable, then we could decide whether $Q \vdash \sigma$ by deciding whether $(\alpha \to \sigma)$ is logically valid.

Final Theorem

Theorem. $\#\mathrm{Th}\ \mathfrak{N}$ is not recursively enumerable.

Pf. With all we have in place at this point, there are many essentially different ways to prove this.

- A very easy route is to invoke Craig's theorem: if $\mathrm{Th}\ \mathfrak{N}$ were effectively enumerable, then it would be axiomatizable, and hence a consistent axiomatic extension of Q and PA, and hence incomplete. But $\mathrm{Th}\ \mathfrak{N}$ is complete.

Final Theorem (cont.)

- We could also invoke Tarski's theorem. If $\#\mathrm{Th}\ \mathfrak{N}$ were recursively enumerable, then it would be the range of a recusive function f. Let $F(v_0, v_1)$ represent f in Q (PA). Then $F(v_0, v_1)$ also defines f in $\mathfrak{N}$. Then we would have

$$\#\sigma \in \#\mathrm{Th}\ \mathfrak{N} \text{ iff } \models_{\mathfrak{N}} \exists x\, F(x, v_0)\, [\![\#\sigma]\!],$$

meaning that $\exists x\, F(x, v_0)$ defines $\#\mathrm{Th}\ \mathfrak{N}$, contrary to Tarksi's theorem on the undefinability of arithmetic truth. ■