23 Turing machines encoded, again

23.1 ϕ_m, again

Consider a Turing machine M whose input alphabet is $\{0,1\}^*$, likewise its output alphabet. We assume that whenever it halts, the tape contents are a single bitstring surrounded by blanks, with the read/write head positioned at the leftmost bit (if the bitstring is nonempty). Actually, there is a translation process which converts M to a machine with this property, but it is very slightly easier and only slightly wrong just to assume it.

The machine M (partially) computes a partial function $f : \mathbb{N} \rightarrow \mathbb{N}$. For any $n \in \mathbb{N}$, let z be the unique bit-string whose length-lexicographical encoding is n. If M halts on input z, then it halts with a well-defined output w, and $f(n)$ is the length-lex encoding of w. If M loops on input z then $f(n)$ is undefined.

For any $m \in \mathbb{N}$, if y is the reverse enoding of m, i.e., m is the length-lex encoding of m, and $y \in$ TM, then there is a unique Turing machine M whose code is y. Then ϕ_m is the partial function computed by M, as above.

If $y \notin$ TM then ϕ_m is nowhere defined.

23.2 Configurations of M

- Recall that a *configuration* of a Turing machine M is a combined string $\alpha q \beta$ where $\alpha\beta$ is a string, over the tape alphabet, including all the nonblank part of the tape, q is the state, and the leftmost symbol in β gives the tape square being scanned (blank if $\beta = \lambda$).
- Recall that it is quite easy to define a relation $\alpha q \beta \vdash_M \alpha' q' \beta'$, where the configuration $\alpha q \beta$ yields $\alpha' q' \beta'$ *directly*.
- $\vdash_M^*$ for the 'transitive closure' of the relation $\vdash_M$.
- It is even easier to identify a *halting configuration*.
- Also, the string (w above) found on the tape in a halting configuration.
- Also, the initial configuration on input z: $q_0 z$.

We use σ and σ_i to denote configurations of M (for no particular reason).

23.3 The partial function computed by M, again

Again discussing ϕ_m, where m is the length-lex encoding of a bitstring y which belongs to TM and therefore encodes a Turing machine. (If $y \notin$ TM, it is of little interest).

Call that Turing machine M: the machine encoded by the unique string y whose length-lex encoding is m.

Given $n \in \mathbb{N}$, $\phi_m(n)$ is defined as follows. Let z be the reverse encoding of n, i.e., n is the length-lex encoding of z.

If there exists a sequence $\sigma_0, \sigma_1, \ldots, \sigma_p$ of configurations of M such that

- $\sigma_0 = q_0 z$, the initial configuration on input z,

- for $1 \leqslant k \leqslant p$, $\sigma_{k-1} \vdash_M \sigma_k$, and
- σ_p is halting.

Then M halts on input z, and there is a unique bitstring w on the tape in the halting configuration σ_p, and

$$\phi_m(n) = r$$

where r is the length-lex encoding of w.

If no such sequence exists then M loops on input z and

$$\phi_m(n) \uparrow$$

23.4 $H(m, n, r, s)$

This relation says

- m is the length-lex encoding of a bitstring y which is a valid encoding of a Turing machine, call it M.
- Let z be the string length-lex encoded as n.
- Let w be the string length-lex encoded as r.
- Let σ be the string length-lex encoded as s.
- Then σ encodes a halting sequence, of M, with input string z, and output string (from the halting configuration) w which is encoded as r.

The relation refers repeatedly to concatenation of strings, which we know is a primitive recursive function under the length-lex encoding. But more importantly

(23.1) Proposition *The relation $H(m, n, r, s)$ is primitive recursive (i.e., considered as a function mapping to 0 and 1, it is primitive recursive).*

But every primitive recursive relation is representable (or expressible) in Peano Arithmetic. Remember that

$$\overline{0} = 0, \overline{1} = s(0), \overline{2} = s(s(0)) \ldots$$

are the so-called numerals. There is a formula $A(x_1, x_2, x_3, x_4)$ of Peano arithmetic, with only x_1, x_2, x_3, x_4 free, such that $H(m, n, r, s)$ is true if and only if

$$A(\overline{m}, \overline{n}, \overline{r}, \overline{s})$$

can be proved in Peano Arithmetic.