Computability Theory, 2nd homework

Benjamin Benčina, 27192018

17. januar 2021

Ex. 1: Given a finite alphabet Σ, we first give a definition in terms of ordinary Turing machines of what it means for a partial function $f\colon \Sigma^* \times \mathbb{N} \rightharpoonup \Sigma^*$ to be computable.

Consider a Turing machine with the alphabet $\Sigma \cup \{0, 1, ;\}$ and a halting state **halt**. A partial function $f\colon \Sigma^* \times \mathbb{N} \rightharpoonup \Sigma$ is computable, if there exists such a Turing machine that the following hold

- for every $(w, n) \in \operatorname{dom} f$ and the associated $(k + l + 1)$-length input $(w_1, \ldots, w_k, ;, n_1, \ldots, n_l)$, where the $w_i \in \Sigma$ constitute a finite word w and the n_i constitute the binary representation of a natural number n, we have

  $$(w, ;, \operatorname{bin}(n))@0 \Downarrow (\mathbf{halt}, f(w, n))@i$$

  for some $i \in \mathbb{Z}$;
- for every $(w, n) \notin \operatorname{dom} f$ the machine does not end up in the **halt** state (regardless of the "output").

Ex. 2: Let q_d be a representation of dyadic rational numbers by words over the alphabet Σ_b, as defined in Lecture 12. Let us show that there exists a computable partial function $f\colon \Sigma_b^* \times \mathbb{N} \rightharpoonup \Sigma_b^*$ that satisfies

(a) $\operatorname{dom}(f) = \{u \in \operatorname{dom}(q_d);\ q_d(u) \neq 0\} \times \mathbb{N}$, and

(b) for any $(u, n) \in \operatorname{dom}(f)$, $|q_d(f(u)) - q_d(u)^{-1}| \leq 2^{-n}$.

Let us first notice that what the above function is trying to calculate is the multiplicative inverse of u with precision of at least n decimal places. We also assume that the addition and multiplication of dyadic rationals are computable functions over Σ_b. Now, let us take a short detour from computability theory and into the world of real analysis.

We recall the following Taylor series

$$\frac{1}{1-u} = 1 + u^1 + u^2 + u^3 + \cdots$$

which calculates the inverse of the real number $1 - u$. In the case where u is a dyadic rational, clearly $1 - u$ is one as well. The problem is, however, that the (strict) convergence radius for this series is 1, so we can calculate inverses for values between 0 and 2 (and with a minus for the negative case). Nevertheless, we can remedy this problem later and we have produced an inverse formula containing merely addition and multiplication of dyadic rationals. To remedy the above problem, consider a positive dyadic rational

$$u = \sum_{i=-n}^{k} u_i 2^i$$

We can "shift" it by simply extracting the largest power of 2 as such

$$u = 2^k \cdot \sum_{i=-n-k}^{0} u_{i+k} 2^i$$

where this is only needed in the case of $k > 0$ (we can probably also work with $k > 1$). The inverse is then

$$u^{-1} = 2^{-k} \left(\sum_{i=-n-k}^{0} u_{i+k} 2^i \right)^{-1}$$

where the sum in the parenthesis is a dyadic rational number between 0 and 2. The Turing machine that would calculate our function then looks something like this:

- For a given input $(w, ;, \mathrm{bin}(n))$, shift w enough to the right so that the number represents a dyadic rational u between 0 and 2, and save the number by which we shifted somewhere on the tape. This can easily be done in a computable way by simply moving the decimal period and noting the number of move operations.
- Calculate a new dyadic rational v such that $u = 1 - v$. Clearly, $v = 1 - u$, which is a formula involving only the addition and multiplication of dyadic rationals, as 1 and -1 are both such.
- Calculate the inverse of u by using v in the above formula until two consecutive estimates have enough common digits after the decimal period, say $n + k$, where k is the saved value we shifted by. The digits before the decimal period also have to be the same. This procedure terminates by Taylor's Theorem in a finite number of steps for any fixed n.
- Calculate the inverse of 2^k by simply mirroring the string that represents it over the decimal point and shifting the decimal point one place to the right. Indeed, this is needed to correct the offset that is inherent to any number system, that is, the first place to the right of the decimal represents the (-1)-th power of 2 while the first place to the right represents the 0-th, not the 1-st, power of 2. Note here that if the shifting was not needed, we can skip this step.
- Multiply the values to obtain the desired approximate inverse, which can be done in a computable way by assumption, since the produced approximation is by construction a finite string and hence represents a dyadic rational.
- If the input was negative in the beginning, make sure the minus is added here as well. Indeed, without loss of generality we can delete the minus at the beginning and reintroduce it at the end, so we are operating merely with positive numbers.

Notice, that this procedure clearly does not work for $w = 0$, regardless of n, for the simple reason that there is no way to shift w in order to obtain a number strictly between 0 and 2. In addition, the inverse formula will not hold, hence the first demand is satisfied. The second demand is satisfied by construction and is guaranteed to hold by Taylor's Theorem.

Ex. 3: We will prove that the partial function $r \colon \mathbb{R} \rightharpoonup \mathbb{R}$ defined by

$$r(x) \simeq \begin{cases} x^{-1} & \text{if } x \neq 0 \\ \uparrow & \text{if } x = 0 \end{cases}$$

is computable.

Recall that a real partial function $r \colon \mathbb{R} \rightharpoonup \mathbb{R}$ is by definition computable precisely when it is $(\gamma_c - \gamma_c)$-computable. Hence, we are searching for a $T2M$-computable partial function $g \colon \Sigma_c^* \rightharpoonup \Sigma_c^*$ such that for every $x \in \mathbb{R}$ and every γ_c-representation p of x we have

- $g(p)$ is defined precisely when r is defined in x, and
- if $g(p)$ is defined, then $g(p)$ is a γ_c-representation for $r(x)$.

Since the dyadic rationals are dense in $\mathbb{R}$, that is, we can approximate any real number with dyadic rationals with arbitrary precision, and since for an approximating sequence of dyadic rationals $(u_n)_n \to x$ by definition $\gamma_c(p) = \lim_{n \to \infty} q_d(u_n)$, we conclude that our function r will be computable precisely when

there exists a Turing machine, that can for a given non-zero real number $x \in \mathbb{R}$ calculate the inverses of the dyadic approximations of x to an arbitrary precision (previously denoted by n). We notice, however, that this is precisely the substance of Exercise 2. Indeed, using Exercise 2, we give the following realizer g for r:

if the input word is $u_1,;,u_2;,u_3,;,\ldots$, we output $f(u_1,1),;,\ldots,;,f(u_n,n),;,\ldots$ as defined in Exercise 2. The partial inverse function r is hence computable.

Ex. 4: Does there exist any total computable function $r'\colon \mathbb{R} \to \mathbb{R}$ such that $r'(x) = x^{-1}$ for all $x \neq 0$?

The answer is no. Suppose that we have an extension r' of r to the real line by defining $r'(0) = r_0 \in \mathbb{R}$. By the Continuity Theorem, r' is a continuous function on the whole $\mathbb{R}$ with respect to the usual topology, which is clearly absurd, as

$$\lim_{x\to 0^+} r'(x) = \lim_{x\to 0^+} r(x) = \infty$$

and

$$\lim_{x\to 0^-} r'(x) = \lim_{x\to 0^-} r(x) = -\infty.$$