Emulation of Computations — and a "Universal" $\mathcal{S}$-Program Universal Turing Machines

Introduction

While the current set of lecture notes introduces the concept of ***universality*** by describing a universal $\mathcal{S}$-program, you have probably seen this concept introduced already — using a ***universal Turing machine*** instead. This document — which is for interest, only — describes *one* way to describe encodings of Turing machines and their inputs, and to develop a "universal Turing machine" that uses this encoding scheme.

There are many variants of a "Turing machine". The version of a Turing machine, used here, to define computational problems, is the same as the one introduced (for the computation of functions) in the previous lecture. The computation of such a machine on a sequence of k inputs $\omega_1, \omega_2, \ldots, \omega_k \in \Sigma^\star$ will be considered, where Σ is the Turing machine's input alphabet and k is a positive integer.

Consider the problem of representing (or "encoding") a Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{halt}})$$

and a sequence of k input strings $\omega_1, \omega_2, \ldots, \omega_k \in \Sigma^\star$. We will ***assume*** (or require) that $q_0 \neq q_{\mathsf{halt}}$: It is easy to modify any Turing machine, that computes a function, so that it satisfies this assumption.

Input Alphabet

Let

$$\Sigma_{\mathsf{TM}} = \{\texttt{(}, \texttt{)}, \texttt{,}, \texttt{q}, \texttt{s}, \texttt{0}, \texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}, \texttt{5}, \texttt{6}, \texttt{7}, \texttt{8}, \texttt{9}, \texttt{H}, \texttt{L}, \texttt{R}, \texttt{S}, \texttt{\#}\}.$$

Σ_{TM} will be the ***input alphabet*** for the universal Turing machine M_{UTM} that is now being described.

The universal Turing machine M_{UTM} that is being defined should receive a string in $\Sigma_{\mathsf{TM}}^\star$ as input that has the form

$$(\mu\texttt{,}\nu_1\texttt{,}\nu_2\texttt{,}\ldots\texttt{,}\nu_k)$$

where μ is a string in $\Sigma^\star_{\mathsf{TM}}$ encoding some deterministic Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{halt}})$$

and, for $1 \leq i \leq k$, ν_i is a string in $\Sigma^\star_{\mathsf{TM}}$ encoding an input string $\omega_i \in \Sigma^\star$ for M. The encodings of Turing machines and strings, to be used, are described below.

Encoding a Turing Machine

Encoding States. Renaming states if necessary, we may assume that

$$Q = \{q_0, q_1, \ldots, q_t, q_{\mathsf{halt}}\}$$

for some integer $t \geq 0$ — so that $|Q| = t + 2$.

For $0 \leq i \leq t$, state q_i will be encoded by the string $e(q_i) \in \Sigma^\star_{\mathsf{TM}}$ consisting of the letter `q` followed by the ***unpadded*** decimal representation of the number i — so that $e(q_0) =$ "`q0`", $e(q_1) =$ "`q1`", $e(q_{12}) =$ "`q12`", and so on. The state q_{halt} will be encoded by the string $e(q_{\mathsf{halt}}) =$ "`qH`". It follows that the encodings of both the start state and the halting state are easy to recognize.

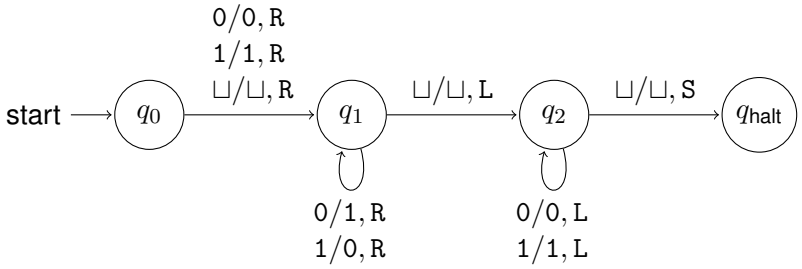
Encoding Tape Symbols. Let $\ell \in \mathbb{N}$ such that $10^{\ell-1} < |\Gamma| \leq 10^\ell$.

- $\sqcup$ will be encoded by a string $e(\sqcup) \in \Sigma^\star_{\mathsf{TM}}$ starting wth `s` and ending with ℓ `0`'s — that is, ending with a ***padded*** decimal representation of number 0 with length ℓ.
- Let $|\Sigma| = h$, so that $1 \leq h \leq |\Gamma| - 1$. Choosing an ordering for the input symbols in Σ we may assume that
$$\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_h\}.$$
- For $1 \leq i \leq h$, σ_i will be encoded by the string $e(\sigma_i) \in \Sigma^\star_{\mathsf{TM}}$ starting with `s` and ending with a ***padded*** decimal representation of i with length ℓ — that is, ending with a representation of i with enough leading `0`'s to make sure that it is a string with length ℓ.
- Choosing an ordering for the remaining tape symbols, we may assume that the symbols in Γ that are different from $\sqcup$ and not in Σ are symbols $\sigma_{h+1}, \sigma_{h+2}, \ldots, \sigma_m$ — where $m \geq h$, and $10^{\ell-1} < |\Gamma| = m + 1 \leq 10^\ell$.
- For $h + 1 \leq i \leq m$, the symbol σ_i will *also* be encoded by the string $e(\sigma_i)$ starting with `s` and ending with the padded decimal representation of the number i with length ℓ.

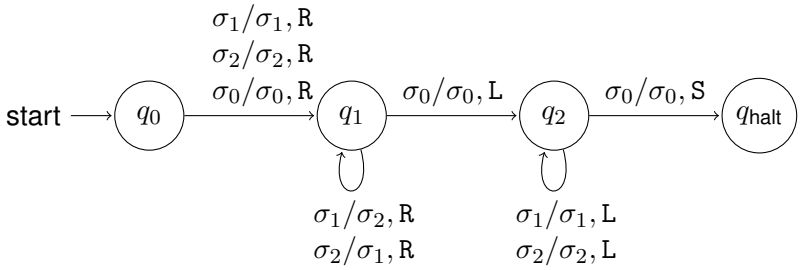
This ensures that ***every*** symbol in Σ is encoded by a string in $\Sigma^\star_{\mathsf{TM}}$ with the same length, $\ell + 1$. This will make it easier to use a tape of M_{UTM} to represent the contents of M's tape. It also makes it easy to recognize the encoding of $\sqcup$ and to decide whether an encoding of a tape symbol is an encoding of a symbol in Σ.

An Ongoing Example

Consider the following Turing machine — which has input alphabet $\Sigma = \{0, 1\}$, and tape alphabet $\Gamma = \{0, 1, \sqcup\}$.



Choosing an ordering for the symbols in Σ, let us set $\sigma_1 = 0$ and $\sigma_2 = 1$. As suggested above, $\sqcup = \sigma_0$. A modified state diagram that uses these names for symbols is as follows.



The transition table for this modified example Turing machine is as follows.

	σ_0	σ_1	σ_2
q_0	$(q_1, \sigma_0, \text{R})$	$(q_1, \sigma_1, \text{R})$	$(q_1, \sigma_2, \text{R})$
q_1	$(q_2, \sigma_0, \text{L})$	$(q_1, \sigma_2, \text{R})$	$(q_1, \sigma_1, \text{R})$
q_2	$(q_{\text{halt}}, \sigma_0, \text{S})$	$(q_2, \sigma_1, \text{L})$	$(q_2, \sigma_2, \text{L})$

Encoding Tape Symbols as Strings in $\Sigma^\star_{\text{TM}}$

Now, since $|\Gamma| = 3$, $\ell = 1$ (since $10^0 < 3 \leq 10^1$), so that

- $e(\sqcup) = e(\sigma_0) =$ "s0"
- $e(0) = e(\sigma_1) =$ "s1"
- $e(1) = e(\sigma_2) =$ "s2"

Encoding Transitions as Strings in $\Sigma^\star_{\mathsf{TM}}$

Each ***transition*** of M now has the form

$$\delta(q, \sigma) = (r, \tau, D)$$

where

- $q \in Q \setminus \{q_{\mathsf{halt}}\} = \{q_0, q_1, q_2, \ldots, q_t\}$,
- $\sigma \in \Gamma = \{\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_m\}$,
- $r \in Q = \{q_0, q_1, q_2, \ldots, q_t, q_{\mathsf{halt}}\}$,
- $\tau \in \Gamma$, and
- $D \in \{\mathtt{L}, \mathtt{R}, \mathtt{S}\}$.

Since $\mathtt{L}, \mathtt{R}, \mathtt{S} \in \Sigma_{\mathsf{TM}}$, each direction of motion can be encoded as a string with length one.

The above transition can be encoded as the string $e(\delta(q, \sigma)) \in \Sigma^\star_{\mathsf{TM}}$ with the following form:

$$\mathtt{(}e(q)\mathtt{,}e(\sigma)\mathtt{,}e(r)\mathtt{,}e(\tau)\mathtt{,}e(D)\mathtt{)}$$

For example, the transition

$$\delta(q_1, \sigma_1) = (q_1, \sigma_2, \mathtt{R})$$

in the ongoing example is encoded by the string

$$e(\delta(q_1, \sigma_1)) = \text{“}\mathtt{(q1,s1,q1,s2,R)}\text{”}$$

Exercise: Write down the encodings of at least a few more of the transitions in the example Turing machine. This should now be reasonably easy to do!

Encoding the Transition Function as a String in $\Sigma^\star_{\mathsf{TM}}$

The entire ***transition function*** δ can now be encoded by a string $e(\delta) \in \Sigma^\star_{\mathsf{TM}}$ that

- starts with “(”,
- continues with the encodings of all the transitions $\delta(q, \sigma)$, separated by commas (“,”) and
- ends with “)”

The encodings of transitions should be listed in ***sorted order:***

- If $0 \leq i < j \leq t$ then the encoding of $\delta(q_i, \sigma)$ should be listed before the encoding of $\delta(q_j, \tau)$ for all $\sigma, \tau \in \Gamma$.
- For $0 \leq i \leq t$, if $0 \leq h < j \leq m$ then the encoding of $\delta(q_i, h)$ should be listed before the encoding of $\delta(q_i, j)$.

Encoding a Turing Machine as a String in $\Sigma^\star_{\mathsf{TM}}$

The Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{halt}})$ can now be encoded as a string $e(M)$ with the form

$$(e(Q),e(\Sigma),e(\Gamma),e(\delta))$$

where

- $e(Q)$ is the unpadded decimal representation of the integer t (such that $|Q| = t + 2$, as above);
- $e(\Sigma)$ is the unpadded decimal representation of the size, h, of Σ;
- $e(\Gamma)$ is the unpadded decimal representation of the integer m (such that $|\Gamma| = m + 1$, as above);
- $e(\delta)$ is the encoding of the transition function, δ, as described above.

Encoding a String

A string $\nu \in \Sigma^\star_{\mathsf{TM}}$ is an encoding of a string

$$\omega = \tau_1\tau_2 \ldots \tau_n \in \Gamma^\star$$

if and only if ν is the following string, "$e(\omega)$":

$$e(\tau_1)e(\tau_2) \ldots e(\tau_n)$$

where the encoding of a symbol $\tau_i \in \Gamma^\star$ is as described above. As noted above, $e(\tau_i)$ is a string with length $\ell + 1$, for every integer i such that $1 \leq i \leq n$, so that $|\nu| = (\ell + 1)n = (\ell + 1)|\omega|$.

Furthermore, if ω is as above then $\omega \in \Sigma^\star$, so that ω might be an "input string", if and only if $\tau_i \in \Sigma$, so that $e(\tau_i)$ is the symbol $\mathtt{s}$, followed by the padded decimal representation of an integer j such that $1 \leq j \leq |\Sigma| = h$, for $1 \leq i \leq n$.

Processing

Describing an Algorithm — Multi-Tape Turing Machines

As noted in the document "Useful Turing Machine Variants" for Lecture #7, a standard Turing machine can simulate the computation of a ***multi-tape Turing machine*** which has a fixed (but arbitrarily large) number of tapes, with independently moving tape heads. The "universal Turing machine" M_{UTM} discussed here will be described as a multi-tape Turing machine; it follows by the above that a standard one-tape, implementing this, also exists.

Organization of Tapes

The multi-tape Turing machine M_{UTM}, being described, will use the following eight tapes.[1]

1. The first tape — the ***input tape*** — initially stores the input and is initially described as in the lecture notes. This will be used to access the transition function of the encoded Turing machine, M, as M_{UTM}'s computation proceeds.
2. The second tape will be used to represent the number of states in the Turing machine M whose encoding is part of the input. In particular, this will be used to store the unpadded decimal representation of a non-negative integer t such that M has $t + 2$ states.
3. The third tape will be used to represent the size, h, of M's input alphabet, Σ. That is, an unpadded decimal representation of h will be stored here.
4. The fourth tape will be used to remember the integer m such that the tape alphabet Γ has size $m + 1$, as described above. That is, an unpadded representation of m will be stored on this tape.
5. The fifth tape will be used to remember the length of an encoding of a tape symbol. In particular, it will store a unary encoding of the integer $\ell = \lceil \log_{10}(m + 1) \rceil$ described above, that is, a string of ℓ copies of `1`.
6. The sixth tape stores the encoding of the current state of M.
7. The seventh tape represents the non-blank part of M's tape, enclosed by copies of the symbol #.
8. The eighth tape will be used to store values needed for computations, so that it functions as a "scratch tape".

Validating an Input — and Initializing the Tapes

As noted above, the input string for the universal Turing machine should be a string in $\Sigma^{\star}_{\text{TM}}$ with the form

$$(\mu, \nu_1, \nu_2, \ldots, \nu_k)$$

where μ is a string in $\Sigma^{\star}_{\text{TM}}$ encoding some deterministic Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{halt}})$$

[1] This certainly uses more tapes than necessary! The organization is intended to make it easier to see how the simulation might proceed.

and, for $1 \leq i \leq k$, ν_i is a string in $\Sigma^\star_{\mathsf{TM}}$ encoding an input string $\omega_i \in \Sigma^\star$ for M. Thus M_{UTM} should start by checking whether *its* input string starts with "(", includes at least k copies of "," and ends with ")"— erasing its inputs and halting (so that the empty string is returned as output) otherwise.

Otherwise, for $1 \leq i \leq k$, the encoding ν_i of M's i^{th} input string does not include a "," — so that ν_k is the string between the rightmost "," and the ")" at the end of the string; ν_{k-1} is the string between the next "," to the left, and the rightmost ",", and so on. Thus, for $1 \leq i \leq k-1$, ν_i is the string between the $k-i+1^{\mathsf{st}}$ and the $k-i^{\mathsf{th}}$ copies of "," from the right end of the input string, and μ is the string between the copy of "(" at the beginning of the string, and the k^{th} copy of "," from the right end of this string. Since k is a constant, each of these strings is easily identified.

1. As noted, above, the string μ should have the form

   $$(e(Q),e(\Sigma),e(\Gamma),e(\delta))$$

   if it encodes a Turing machine $M = (Q, \Sigma, \gamma, \delta, q_0, q_{\mathsf{halt}})$ — so that the string should begin with a left bracket, end with a right bracket, and include at least three commas; M_{UTM} should erase its input and halt, returning the empty string as output (as above) if this is not the case. Otherwise, since strings $e(Q)$, $e(\Sigma)$ and $e(\Gamma)$ do not include commas if they are as described above, the computation can proceed as follows in order to validate the input and initialize the tapes.

2. $e(Q)$ is the string between the leading the left bracket and the first comma in the input. As noted above, this should be an unpadded decimal representation of a non-negative integer t such that $|Q| = t + 2$ — and the empty string should be returned as output, as above, if this is not the case. Otherwise $e(Q)$ should be copied onto the second tape (and the tape head should be moved back to the left of this string) in order to initialize the second tape.

3. $e(\Sigma)$ is the string between the first and second commas in the input. As noted above this should be the unpadded decimal representation of a positive integer h — where $h = |\Sigma|$. Once again, the empty string should be returned as output if this is not the case. Otherwise $e(\Sigma)$ should be copied onto the third tape (and the tape head should be moved back to the left of this string) in order to initialize the third tape.

4. $e(\Gamma)$ is the string between the second and third commas in the input. As noted above this should be the unpadded decimal representation of a positive integer m such that $|\Gamma| = m + 1$ — so that $m \geq h$. Once again, the empty string should be returned as output (and computation should halt) if this is not the case. Otherwise $e(\Gamma)$ should be copied onto the fourth tape (and the tape head should be moved back to the left of this string) in order to initialize the fourth tape.

5. If the decimal encoding of m is a string of i 9's, for some integer i, then $\ell = i + 1$. Otherwise ℓ is the length of the decimal representation. M_{UTM} should check which of these cases is applicable and write a string of ℓ copies of `1` onto the fifth tape (moving the tape head back to the left) in order to initialize the fifth tape.

6. $e(\delta)$ is the string between the third comma and the final symbol, a right bracket, in the input. As noted above, this should consist of a sequence of encodings of transitions — sorted by input state and symbol, and separated by commas. In a first pass over this string one can check whether it consists of a sequence of encodings

$$(e(q),e(\sigma),e(r),e(\tau),e(D))$$

of transitions — so that $q \in Q \setminus \{q_{\mathsf{halt}}\} = \{q_0, q_1, \ldots, q_t\}$, $r \in Q$, $\sigma, \tau \in \Gamma$ and $D \in \{\mathtt{L}, \mathtt{R}, \mathtt{S}\}$. Since t is now stored on the second tape, m is stored on the fourth tape, the length of an encoding of a tape symbol can be checked using the information on the fifth tape, and the encodings of states and tape symbols do not include either commas or left or right brackets, it is easy to check this using essentially a single sweep over the string (moving back to the left as needed to compare decimal encodings of integers).

It remains only to check whether the transitions are sorted by input state and input tape symbol — and to check that every possible transition is included. This process can also be carried out using essentially a single sweep from left to right over the string being provided, provided that the scratch tape is used to store unpadded decimal representations of a pair of integers i and j such that $0 \leq i \leq t$, $0 \leq j \leq m$, and the next transition that one should expect to see in the encoding of δ is a transition with input symbol q_i and tape symbol σ_j: One can then initialize i and j each to be 0 and update these values (using the copies of t and m found on the second and fourth tapes) as needed.

If it determined, at any point in this computation, that $e(\delta)$ does not have the form given above, then the empty string should be returned as output, as above, and the computation should halt. Otherwise the tape head should be moved back to the leftmost symbol in $e(\delta)$, and the tape heads for the second and fourth tapes should be moved back to their initial positions, before the computation continues.

7. The encoding "`q0`" of the start state, q_0, should be written onto the sixth tape, to initialize it, and the tape head should be moved back to the left of this string.

8. The symbol # should be written onto the (initially blank) seventh tape, followed by the encoding $e(\sigma_0) = e(\sqcup)$ of a single "blank".

For $1 \leq i \leq k$ it should be checked that the i^{th} encoded input string, ω_i, is a string in $\Sigma^\star$ — so that each of the symbols in this string is σ_j for an integer j such that $1 \leq j \leq h = |\Sigma|$. Since h is stored on the third tape and the length of an encoding of a symbol can be checked using the information on the fifth tape, this test is easy to carry out. Once again,

the empty string should be returned, as output, if any of these tests fails. Otherwise the encodings of $\omega_1, \omega_2, \ldots, \omega_k$ should be copied onto the seventh tape — with separating commas each replaced by the encoding of a blank.

Another copy of # should be written onto the seventh tape, to the right of the encoding of ω_k, and the tape head should be moved back to the beginning of the encoding of blank (immediately the right of the leftmost #) to complete the initialization of the seventh tape.

9. Any non-blank symbols on the eighth "scratch" tape should be erased, in order to complete the initialization.

At the end of this initialization phase the sixth and seventh tapes of M_{UTM} represent the initial configuration of M on the inputs $\omega_1, \omega_2, \ldots, \omega_k$, as needed for a simulation to begin.

Simulating a Step

Each step in a computation of M can now be simulated as follows.

1. Consulting the encoding of the current state on the sixth tape, sweep over the encoding $e(\delta)$ of the transition function, on the input tape, until encodings of transitions for the current state q have been located. Then, consulting of the encoding of the symbol σ that is currently visible — whose encoding begins at the location of the tape head on the seventh tape — continue to sweep over $e(\delta)$ on the input tape until the encoding of the transition

$$\delta(q, \sigma) = (r, \tau, D)$$

to be applied next has been located.

If $r = q_{\mathsf{halt}}$ then proceed to the "cleanup" phase described in the next subsection.

Otherwise the copy, $e(\sigma)$, of the currently symbol, that the tape head now points to, should be replaced by the encoding $e(\tau)$ of the symbol, τ, that should replace σ (as given by the transition that has been found) — moving the tape head, for Tape #7, back to the copy of "s" that begins the encoding of τ.

The simulation should continue as follows.

2. $D = \mathtt{L}$, $D = \mathtt{R}$, or $D = \mathtt{S}$.

   - If $D = \mathtt{L}$ then the tape head for the seventh tape should next be moved left until either # or s is found. If # was found then the tape head was immediately to the right of this, when this part of the simulation began; insert an encoding of blank by replacing the # now visible with the string "#$e(\sqcup)$" and then move the tape head onto the copy of s that is immediately to the right of the # that has now been written. In either case, the tape head now rests at the beginning of the encoding of a symbol

immediately to the left of the symbol that it pointed to when the simulation of this step began.

- If $D = \mathtt{R}$ then the tape head for the seventh tape should be moved right until either # or $\mathtt{s}$ is found. If # was found then the tape head was at the beginning of the encoding of a symbol immediately to the left of this, when this part of the simulation began; insert an encoding of blank by replacing the # now visible with the string "$e(\sqcup)$#" and then move the tape head onto the copy of $\mathtt{s}$ that begins the encoding of blank that has now been written. In either case, the tape head now rests at the beginning of the encoding of a symbol immediately to the right of the symbol that it pointed to when the simulation of this step began.
- If $D = \mathtt{S}$ then Tape #7 does not need to be further updated.

In each of these cases the encoding of the state q on the sixth tape should be replaced by an encoding of the state r, and the tape head should be moved back to the beginning of this encoding, to complete the simulation of this step.

Cleaning Up

Copies of # at the beginning and end of the string on the seventh tape, and encodings of leading and trailing blanks, should be removed to obtain the string that should be returned as output.

One straightforward way to do this is to move the tape head right until the right copy of # is seen; remove this along with any copies of encodings of $\sqcup$ that are seen, as the tape head moves left, until either the left # or an encoding of a non-blank symbol is found.

If # is seen before an encoding of a non-blank symbol then, after completion of the erasure of the tape, the encoding $e(\sqcup)$ of "$\sqcup$" should be written onto the tape, and the tape head should be located at the single copy of "$\mathtt{s}$" at the beginning of this encoding — so that the tape contents represent the fact that the empty string should be returned, as output, by the Turing machine whose execution is being simulated.

Otherwise the tape head should sweep left until the left copy of "#" is found. This should be erased, along with copies of the encoding of "$\sqcup$", until the encoding of the leftmost non-blank symbol has been reached. The string $e(\sqcup)$ should be written to the left of this encoding, and the tape head should be moved back, left, to the copy of "$\mathtt{s}$" at the beginning of this encoding of "$\sqcup$".

Correctness

It is easily established by induction on i that for any non-negative integer i, if M uses at least i steps when executed on inputs $\omega_1, \omega_2, \ldots, \omega_k$, the sixth and seventh tapes of M_{UTM} correctly represent the current state and the contents and location of M's tape, after the simulation of the first i steps.

It follows by the correctness of the "Cleanup" phase as well that the simulation ends if and only if the execution of M on inputs $\omega_1, \omega_2, \ldots, \omega_k$ halts, and, if the execution ends, then the seventh tape of M_{UTM} includes the encoding of M's output at the end of this simulation of this computation, as required.